

Doctoral Dissertation

**A Study on Analyzing Cyber Attacks
through Active and Passive Observation**

能動的観測と受動的観測による
サイバー攻撃分析に関する研究

Takashi KOIDE
小出 駿

Graduate School of Environment and
Information Sciences,
Yokohama National University

Supervisor: Professor Tsutomu MATSUMOTO

March 2021

Abstract

With the spread of communication devices such as IoT (Internet of Things) and mobile devices as well as improving the functionality of the web, cyber attacks on the Internet have been becoming more complex and sophisticated. Cyber attacks can be divided into two types: attacks targeting specific groups or individuals, and attacks against many and unspecified users or devices. This thesis focuses on the latter type of attacks, which affect a wider range of users. Current existing countermeasures against this type of attack are effective against individual attacks, but reactive and limited to these attack techniques. These countermeasures will not respond to changes in attack techniques and the evolution of devices in the future. In order to prevent the attacks continuously and proactively, it is necessary to estimate the attackers' purposes and to follow the attack vectors early on. In this thesis, we propose methods for observing cyber attacks targeting many and unspecified users and devices on the Internet by combining passive and active observation. Correlation analysis of the observation from two perspectives allows us for a faster and more comprehensive collection of cyber attacks in the wild. Unlike existing methods that are limited to observing the surface of individual attacks, our methods analyze attackers' purposes and attack vectors, thus enabling long-term observations that are robust to changes in attack techniques and the evolution of network devices.

First, we propose a system for collecting and detecting multi-step social engineering (SE) attacks on the web. Modern web-based attacks use social engineering to make users download malware and leak sensitive information. Some SE attacks leverage a sequence of web pages to psychologically manipulate users to lead them to attackers' purposes. We call them multi-step SE attacks in this study. Our system actively follows the sequences of web pages by automating a web browser and detects multi-step SE attacks by extracting features from the entire sequence. We used our system for a large scale measurement and revealed the psychological tactics of attackers to deceive and persuade users. We also analyzed domain names associated with multi-step SE attacks using passively collected user access statistics and found that a large number of users are affected by these attacks.

Then, we propose a system for identifying malicious web pages that trick users to install fake antivirus software. Fake antivirus (AV) software is a type of malware that disguises as legitimate antivirus software and infects users' devices. Fake removal information advertisement (FRAD) sites, which introduce fake removal information for cyber threats, have emerged as platforms for distributing fake AV software. Although FRAD sites seriously threaten users who have been suffering from cyber threats and need information for dealing with them, little attention has been given to understanding these sites. To shed light on the pervasiveness of this type of attack, we performed a comprehensive analysis of both passively and actively collected data. Our system actively collected a large amount of FRAD sites written in multiple languages. We show that FRAD sites occupy search results

when users search for cyber threats, thus preventing the users from obtaining the correct information.

Finally, we propose a method for detecting network packets sent by malware and network scanning tools that implement their network stack. Network packets created by such malware and network scanning tools have unique characteristics in their header fields. We created signatures based on the active analysis of malware and tools. We evaluated our method by using passively collected large-scale network traffic. We found that there was a large amount of network scanning activities using some network scanning tools and reconnaissance activities by attackers to find vulnerable devices on the Internet.

Contents

Abstract	i
List of figures	v
List of tables	v
1 Introduction	1
1.1 Motivation and Contribution	1
1.2 Organization	3
2 Observing Cyber Attacks on the Internet	4
2.1 Passive Observation	4
2.2 Active Observation	5
2.3 Challenges on Observing Cyber Attacks on the Internet	5
3 An Analysis of Multi-step Social Engineering Attacks on the Web	8
3.1 Introduction	8
3.2 Background	10
3.2.1 User Attraction	10
3.2.2 Browser Interaction	13
3.2.3 Web Navigation	13
3.2.4 Problems on Collecting SE Attacks	13
3.3 StraySheep	14
3.3.1 Landing-Page-Collection Module	14
3.3.2 Web-Crawling Module	15
3.3.3 SE-Detection Module	18
3.4 Evaluation	24
3.4.1 Qualitative Evaluation	24
3.4.2 Experimental Setup	25
3.4.3 Effectiveness of URL Collection	25
3.4.4 Efficiency of Web Crawling	27
3.4.5 Evaluating the SE Detection Module	31
3.5 Detailed Analysis of Detected Multi-step SE Attacks	34
3.5.1 SE Attack Categories	34
3.5.2 Common Infrastructures of Multi-step SE Attacks	36
3.6 Discussion	43

3.6.1	Limitations	43
3.6.2	Ethical Consideration	44
3.7	Related work	45
3.8	Conclusion	46
4	Understanding the Fake Removal Information Advertisement Sites	47
4.1	Introduction	47
4.2	Background	49
4.3	Method	50
4.3.1	Web Crawling	51
4.3.2	Classification	51
4.4	Data Collection	53
4.4.1	Collecting Cyber Threats	53
4.4.2	Web Search	54
4.4.3	Creating the Dataset	54
4.5	Evaluation	55
4.5.1	Detection Accuracy	55
4.5.2	Detecting Unknown FRAD Sites	55
4.6	Measurement Study	57
4.6.1	Incoming Traffic to FRAD Sites	57
4.6.2	Downloads and Page Transitions from FRAD Sites	61
4.6.3	Search Poisoning	63
4.6.4	Distribution Sites of Fake AV software	64
4.6.5	Structure and Content of FRAD Sites	66
4.6.6	Infrastructure of FRAD Sites	68
4.7	Discussion	69
4.7.1	Ethical Considerations	69
4.7.2	Limitation	70
4.8	Related Work	71
4.9	Conclusion	73
5	Detection Method for Malicious Packets with Characteristic Network Protocol Header	74
5.1	Introduction	74
5.2	Feature Extraction of Packet Headers Using Macro and Micro Analysis	75
5.2.1	Morto	75
5.2.2	ZMap	76
5.2.3	Masscan	76
5.2.4	Malware used for DRDoS attacks	77
5.2.5	Malicious Packets Targeting Linux Devices	78
5.3	Method	79
5.3.1	Overview	79
5.3.2	Creating Signatures	80
5.3.3	Limitation	80
5.4	Creating Signatures	81
5.4.1	Morto	81

5.4.2	ZMap	82
5.4.3	Masscan	82
5.4.4	Malware used for DRDoS Attacks	82
5.4.5	Malicious Packets Targeting Linux Devices	83
5.5	Evaluation	83
5.5.1	Signatures of Morto	83
5.5.2	Signatures of ZMap	84
5.6	Measurement Study	85
5.6.1	Malware used for DRDoS Attacks	85
5.6.2	Malicious Packets Targeting Linux Devices	85
5.7	Implementation of the Proposed Method on a Network Observation System	86
5.7.1	Implementation	86
5.7.2	Result of detecting network packets	87
5.7.3	Providing information on infected devices	88
5.8	Summary	88
6	Conclusion and Future Work	90
6.1	Conclusion	90
6.2	Future Work	91
	Acknowledgments	92
	Publications	93

List of Figures

3.1	Sequence of web pages in multi-step SE attacks and phases in each web page.	10
3.2	STRAYSHEEP overview.	12
3.3	Conceptual model of WebTree.	16
3.4	Example of extracting features from a sequence.	19
3.5	CDF of time taken to complete web crawling for each landing page within a 1-hour timeout. Horizontal lines mean the percentage of web crawling completed before timeout.	28
3.6	Overlap of SE pages' domain names observed using STRAYSHEEP and TrueClick.	30
3.7	ROC curves of SE detection results for each feature set.	33
3.8	Examples malware-distribution pages that require user to rename files and execute them.	38
3.9	Examples of multimedia scams.	39
4.1	Overview of fake AV software distribution via FRAD sites.	49
4.2	Common page structure of FRAD sites.	49
4.3	Percentage of incoming traffic to FRAD sites from each channel.	58
4.4	Number of FRAD site domain names with the same URL paths.	66
4.5	Mean and median values of time intervals between entries posted on FRAD sites.	68
4.6	Distribution of the languages used on FRAD sites.	69
4.7	Number of FRAD sites per top 20 countries where the IP addresses are located.	70
4.8	Number of FRAD sites per top 20 ASes.	71
4.9	Number of FRAD sites per top 20 TLDs.	72
5.1	Number of network packets observed in 6 hours of analysis time.	76
5.2	Network packets targeting Linux devices.	78
5.3	Network packets (3389/TCP) observed in the darknet.	83
5.4	Network packets observed at the time of publishing ZMap and detected by signatures of ZMap.	84
5.5	Network packets detected by Dark_embedded_linux.1	86
5.6	Network packets detected in NICTER	87
5.7	UDP packets whose ports are abused for DRDoS attacks. Blue bars indicate the number of packets sent from ZMap and orange bars indicate the number of packets sent from other sources	89

List of Tables

3.1	List of features SE-detection module uses.	19
3.2	Comparison between proposed and previous systems	22
3.3	Results of web crawling starting from landing page collected with each method.	23
3.4	Results for each web-crawling module.	27
3.5	Crawling efficiency of each web-crawling module.	29
3.6	Unique domain names observed at each depth.	30
3.7	Results of web crawling using STRAYSHEEP and TrueClick.	30
3.8	Unique SE pages observed at each depth by using STRAYSHEEP and TrueClick.	30
3.9	SE attack categories	34
3.10	Newly observed SE pages' domain names at each depth in ascending order and their user access investigated by SimilarWeb, Alexa Web Information Service, and DNSDB.	40
3.11	Advertising provider domain names redirected to SE domain names.	41
3.12	Top 10 countries mapped	42
3.13	Top 10 ASes hosting SE attacks	42
4.1	List of terms for each category; used to check the term's frequency in the title, URL paths, domain names, and text content of a web page.	51
4.2	Search queries used by the users to reach FRAD sites.	59
4.3	Top 15 social media that led to FRAD sites.	60
4.4	Categories of referral web pages to FRAD sites.	60
4.5	The percentage of FRAD sites included in search results.	63
4.6	Top 15 ad networks that redirected users to distribution sites of fake AV software.	65
5.1	Signature format of TCP packets.	80
5.2	Signature format of UDP packets.	80
5.3	Signature format of ICMP echo request packets.	80
5.4	TCP Signatures.	81
5.5	UDP Signatures.	81
5.6	ICMP Signatures.	82

Chapter 1

Introduction

1.1 Motivation and Contribution

Nowadays, the Internet is an essential infrastructure in communication. Unfortunately, while technology becomes sophisticated, it also attracts cybercriminals. Such threats, cyber attacks, can be divided into two types: attacks targeting specific groups or individuals, and attacks against many and unspecified users or devices. This thesis focuses on the latter type of attacks, which affect a wider range of users. A typical example of these attacks is a malicious web page that displays fake virus alerts or rewards messages to psychologically induce users coming from public channels such as search engines and social media. Another example is network scanning by malware and attackers to find vulnerable devices. Many previous studies have proposed methods for effectively preventing and reducing this type of cyber attacks. However, with the evolution of devices and attack techniques, existing methods will not be able to cover new cyber attacks. This is because they are reactive and limited to individual attack techniques and vectors. In order to prevent attacks continuously and proactively, it is necessary to estimate the attackers' purposes and to follow the attack vectors early on. By capturing the attackers' purposes and attack vectors, we can sensitively identify changes in attack methods and targets and predict possible future attacks.

In this thesis, we focus on methods for observing cyber attacks that target a large number of unspecified users and devices on the Internet. Observing cyber attacks is an essential starting point to understand cyber attacks in detail. It can be divided into two methods: passive and active observation. Many of the previous studies have chosen active or passive observations to conduct detailed analysis specific to individual attacks. However, there is a lack of understanding the attackers' purposes and attack vectors in those studies. We combine passive and active observations to address the following three main challenges in exploring the attackers' purposes and attack vector. The three challenges are (i) matching the individual cyber attack to the actual scale of the damage, (ii) collecting detailed behavioral information of victims, (iii) linking observed cyber attacks to the implementation of their sources. Correlation analysis of the observation from active and passive perspectives allows us for a faster and more comprehensive collection of attacks in the wild than previous studies. Also, we can develop robust defenses

against attacks targeting new devices and infrastructure in advance based on the analysis of attackers’ purposes and attack vector.

In chapter 3, we propose a system for collecting and detecting multi-step social engineering (SE) attacks on the web. Web-based social engineering (SE) attacks manipulate users to perform specific actions, such as downloading malware and exposing personal information. Aiming to effectively lure users, some SE attacks, which we call multi-step SE attacks, constitute a sequence of web pages starting from a landing page and require browser interactions at each web page. Also, different browser interactions executed on a web page often branch to multiple sequences to redirect users to different SE attacks. Although common systems analyze only landing pages or conduct browser interactions limited to a specific attack, little effort has been made to follow such sequences of web pages to collect multi-step SE attacks. We propose STRAYSHEEP, a system to automatically crawl a sequence of web pages and detect diverse multi-step SE attacks. We evaluate the effectiveness of STRAYSHEEP’s three modules (landing-page-collection, web-crawling, and SE-detection) in terms of the rate of collected landing pages leading to SE attacks, the efficiency of web crawling to reach more SE attacks, and accuracy in detecting the attacks. Our experimental results indicate that STRAYSHEEP can lead to 20% more SE attacks than Alexa top sites and search results of trend words, crawl five times more efficiently than a simple crawling module, and detect SE attacks with 95.5% accuracy. We demonstrate that STRAYSHEEP can collect various SE attacks, not limited to a specific attack. We also clarify attackers’ techniques for tricking users and browser interactions, redirecting users to attacks.

In chapter 4, we present a system for identifying malicious web pages that trick users to install fake antivirus software. Fake antivirus (AV) software is a type of malware that disguises as legitimate antivirus software and causes harm to users and their devices. Fake removal information advertisement (FRAD) sites, which introduce fake removal information for cyber threats, have emerged as platforms for distributing fake AV software. Although FRAD sites seriously threaten users who have been suffering from cyber threats and require information for removing them, little attention has been given to revealing these sites. We develop a system to automatically crawl the web and identify FRAD sites. To shed light on the pervasiveness of this type of attack, we performed a comprehensive analysis of both passively and actively collected data. Our system collected 2,913 FRAD sites in 31 languages, which have 73.5 million visits per month in total. We show that FRAD sites occupy search results when users search for cyber threats, thus preventing the users from obtaining the correct information.

In chapter 5, we propose a method for detecting network packets created by malware and network scanning tools. Some malware and network scanning tools implement their original network stack to quickly generate network packets instead of using sockets provided by operating systems. Network packets created by such malware and network scanning tools have unique characteristics in their header fields. By actively analyzing network packets sent by the malware and tools, we created signatures to identify them. We evaluated our method by using passively collected large-scale network traffic. We found that there was a large amount of network scanning activities using some network scanning tools and reconnaissance

activities by attackers to find vulnerable devices on the Internet.

The contributions of this dissertation are as follows.

- We propose a system called STRAYSHEEP to collect and detect multi-step SE attacks. STRAYSHEEP automates web browsers and recursively follows sequences of web pages. This system identifies malicious web pages by applying machine learning to crawling results. STRAYSHEEP allows us to automatically find malicious web pages hiding deep in the sequence of web pages.
- We present a system for crawling and identifying FRAD sites, which introduce fake information of malware removal to users infected with malware. This system extracts the linguistic, visual, and structural features of web pages and uses machine learning to detect FRAD sites. We reveal an ecosystem of a new malware distribution model by analyzing passively collected user access statistics and actively crawled web pages.
- We propose a method to detect network packets sent by malware and network tools that implement their network stack. This method can identify reconnaissance and intrusive network packets from the large number of packets flowing through the network. We clarify that some types of malware and network scanning tools

1.2 Organization

The rest of this dissertation is organized as follows. Chapter 2 presents the background on the observation of cyber attacks and discuss challenges on observing cyber attacks on the Internet. Chapter 3 proposes a system for collecting and detecting multi-step social engineering attacks on the web. Chapter 4 presents a system for identifying malicious web pages that trick users infected with malware to install fake antivirus software. Chapter 5 proposes a method for identifying malware and network tools that implement their network stack by leveraging characteristics of network packets' header fields. Chapter 6 concludes this dissertation.

Chapter 2

Observing Cyber Attacks on the Internet

2.1 Passive Observation

This section explains methods of passively observing cyber attacks on the Internet. Observing cyber attacks is an essential starting point for capturing their traces and mechanisms. Passive observation is a way of simply monitoring the circumstances of cyber attacks without interfering with them (i.e., accessing attackers' servers or executing malware). A typical method of passive observation is to analyze network traffic at servers and transfer points between networks. On the other hand, interacting with an attackers' infrastructure or malware to induce the attacks is called active observation, which will be discussed in detail in the next section. Many studies observed network packets passing through IDS and proxy servers to follow the trail of malware and attack tools [1, 2]. These studies extract network packets and payloads that characterize malware and tools from a large amount of network traffic to analyze trends of cyber attacks and block them. By observing the darknet, which consists of unused IP addresses, we can capture the signs of an attack and exploration activities of attackers [3, 4]. Since it is unusual for normal users to communicate to IP addresses where none of the servers are located, network packets arriving on the darknet are the result of a system malfunction or malicious behavior. Also, we can identify malware behavior and malicious sites indirectly by observing DNS cache servers. Some studies analyzed malicious activities on the Internet in real-time and at low cost by using passive DNS, which integrates traffic observed at multiple servers, rather than observing only a single DNS cache server [5, 6, 7]. The advantage of passive observation is that by continuously monitoring the same point, we can identify cyber attacks by the slight differences that imply activities of attackers or malware. It is also unlikely that these methods will make the attackers aware that they are being observed, or that our experiments will cause unintended harm to other users. On the other hand, its disadvantage is that the desired observation result cannot always be obtained because we do not control the occurrence of the attacks.

2.2 Active Observation

We explain methods of actively observing cyber attacks. As opposed to passive observations, which only collect network traffic flowing through the same point, active observations deliberately generate and monitor malicious network traffic. In other words, active observations intentionally generate attacks or trigger accesses from attackers by analyzing attackers' tools and simulating the environment of victims. For example, attack packets and network scanning can be generated by executing malware and network tools on the sandbox (testing environment for malware analysis). This analysis allows us to access attackers' infrastructure such as malicious web pages and command-and-control (C&C) servers. Previous studies identify malicious network packets by actually running malware and capturing network traffic from them to create signatures [8, 9]. Other studies detect drive-by download attacks by accessing malicious web pages and simulate the victim's vulnerable environment, which is called a client honeypot [10, 11]. There are other active observation methods that automate web browser and crawl web pages to identify cyber attacks that use social engineerings, such as fake virus warning and fake reward [12, 13, 14, 15, 16]. It is also effective to wait for access from attackers in the decoy environment that simulates user devices or servers working as attack springboards. By deploying web server honeypots on the Internet, we can lure attackers and observe their attack methods in detail [17, 18]. Also, the DRDoS honeypot, which is a honeypot for observing some types of DDoS (Denial-of-service) attacks that uses amplification of the response from servers to attack a target, can be deployed on the Internet and collect these attacks in the wild [19, 20]. The advantage of the active methods is that we can observe cyber attacks more accurately in a shorter time than passive observations that monitor a large number of network packets. By intentionally causing malicious activities and accessing malicious servers to simulate victims, we can collect detailed and direct information. However, attackers may be aware of our experiments. Our experiments may inadvertently cause negative effects to other users, so we need to conduct them carefully.

2.3 Challenges on Observing Cyber Attacks on the Internet

In this section, we discuss our roadmap and challenges on observing cyber attacks. Cyber attacks can be divided into two types: attacks targeting specific groups or individuals [21], and attacks against many and unspecified users or devices. Our ultimate goal is to analyze all of those attacks and reveal attackers' purposes and attack vectors. In this thesis, as a starting point for our roadmap, we focus on the latter attacks. In particular, we analyze and reveal malicious activities on the network and attackers' psychological strategies on the web, which are threatening to many and unspecified victims in recent years.

Each of the passive and active observation is usually used to analyze cyber attacks in previous studies. We have explained the advantages of each observation

method in the above section. Previous studies, which counter cyber attacks that target an unspecified but a large amount of users and devices, were designed to be limited to individual types of cyber attacks. They are useful to tackle individual attacks at the moment, however, they cannot respond immediately to the emergence of new cyber attacks. In order to prevent attacks continuously and proactively, it is necessary to estimate the attackers' purposes and to follow the attack vectors early on. Attackers' purposes are the reasons for attackers to manipulate and harm users and their devices to achieve attackers' goal, e.g., stealing users' money and personal information. By revealing them, we can understand not only the mechanism of the attack, but also the possible actions of the attacker during certain phases of the attack. Attack vectors are intrusion routes used by attackers or malware, and traps set by the attackers to deceive users. By understanding the attack vectors in detail at each step of the attacks, we can develop strong defenses. This thesis proposes methods of passive and active observation, and compensate for the disadvantages of previous methods. We can anticipate and defend against possible attack techniques by designing a versatile approach that is robust to the aforementioned changes. Correlation analysis combining these two methods provides comprehensive attack observation.

Here, we address the following three challenges by combining passive and active observations to reveal attackers' purposes and attack vectors. The first challenge is to match the individual cyber attack to the actual scale of the damage, which is necessary to prioritize countermeasures. Active methods are often used to identify cyber attacks in detail. For example, in analyzing cyber attacks on the web that require user interactions, previous studies [13, 12] used active methods, which is automating web browsers, to analyze malicious web pages. However, these studies did not reveal how many users reached each of the identified malicious web pages and how many were affected. The active methods of observing cyber attacks on the web are uncertain in terms of effectiveness and comprehensiveness due to the lack of objective information to measure the threat level of the collected attacks. To address this challenge, in Chapter 3, we propose a system for actively collecting malicious web pages by simulating behaviors of victims and passively analyze statistical data of user accesses to estimate the threat impact of each web page.

The second challenge is to collect detailed behavioral information of users affected by cyber attacks, i.e., the route of accessing each malicious web page and what actions of users caused the attack. Attackers use illegal ways of gaining access to their web pages such as Blackhat SEO (search engine optimization) and abuse social media postings. Such web pages not only exploit users directly and take control of them but also try to profit from redirecting traffic to other malicious web pages by using their web pages as relay points. The active methods of analyzing these web pages by crawling them can discover possible paths to the web pages and observe types of attacks that arise from them. However, they cannot identify what paths actual users mainly take to reach the malicious web pages and what kind of psychological stimulus leads users to the cyber attacks. This is because the active methods do not reveal which of the multiple paths are effective in tricking users. Thus, Chapter 4 focuses on the infrastructure of distributing

fake antivirus (AV) software, which is involved in more than half of all malware distribution on the web [22]. We propose a system to actively collect and detect new infrastructure of distributing fake AV software, called FRAD (fake removal information advertisement) sites. We also analyze user access traffic passively to identify the most effective psychological tactics and path of guiding users, and understand the ecosystem of malware distributions.

The third challenge is to link observed cyber attacks to the implementation of their sources. Previous studies, which passively observed cyber attacks on the Internet and identified malicious traffic, did not link observed attacks to the causes of the traffic. While passive observation can detect traces of cyber attacks from a large amount of traffic, it is difficult to link them to an actual source of the attacks (e.g., specific families of malware and network tools). Also, when there is abnormal traffic in the network, sometimes it is hard to distinguish whether a malicious activity or a system failure. The reason for those disadvantages is the lack of detailed profiles of malware and network tools. Therefore, Chapter 5 proposes a method for detecting malicious packets based on the analysis of actively and passively observed traffic. We can identify the causes that contribute to DDoS attacks and network scanning activities, allowing us to understand cyber attack trends and take countermeasures, e.g., takedowns of specific malware families.

In summary, this thesis addresses each of the three challenges with combinations of active and passive observations and analysis in Chapter 3, 4, and 5. These observations not only compensates for the disadvantages of each method, but are also robust to changes in attack vectors and targeted systems (e.g., devices, operating systems, and web browsers). This is because it provides continuous observations of current cyber attacks from both passive and active perspectives. In this thesis, we focus on cyber attacks against many and unspecified users/devices. Once the effectiveness of our methods are confirmed, they will be a stepping stone to passive and active observation of all attacks, including attacks targeting specific groups and individuals. We can conduct a large-scale and long-term analysis of cyber attacks in the future.

Chapter 3

An Analysis of Multi-step Social Engineering Attacks on the Web

3.1 Introduction

Attackers use social engineering (SE) techniques to lure users into taking specific actions. Modern web-based attacks leverage SE for malware infections [23, 24] and online frauds [25, 14, 12], which are called *web-based SE attacks* (or simply *SE attacks*). Attackers skillfully guide a user’s browser interaction through attractive web content or warning messages to make users download malware or leak sensitive information. For example, to download pirated games, a user clicks a download button on an illegal downloading web page. Then, a popup window with a virus-infection alert is displayed. A user who believes the fake information clicks a “confirm” button and downloads fake anti-virus software [26].

Common systems to automatically collect SE attacks involve accessing web pages collected from search engines [13, 14, 12]. These systems use a web browser to crawl web pages and identify a particular SE attack by extracting features only from each web page. However, some types of SE attacks constitute a sequence of web pages starting from a landing page and require browser interaction (e.g., clicking an HTML element) at each web page to reach the attacks, which we call *multi-step SE attacks*. This is because each web page gradually convinces a user by using different psychological tactics [24]. Also, different browser interactions executed on a web page often branch to multiple sequences, redirecting users to different SE attacks, because there are multiple attack scenarios corresponding to a user’s interests or psychological vulnerabilities. Although current systems analyze only landing pages or conduct browser interactions limited to a specific attack, little effort has been made to follow such sequences of web pages to collect multi-step SE attacks.

We propose STRAYSHEEP, a system to automatically crawl the sequence of web pages and detect diverse multi-step SE attacks derived from a landing page. STRAYSHEEP is based on two key ideas. The first idea is to simulate the multi-step browsing behaviors of users, that is, intentionally follow the sequence of web pages by selecting possible elements that psychologically attract users to lead them to

SE attacks. STRAYSHEEP not only follows a single sequence of web pages but also crawls multiple sequences derived from a landing page. The second idea is to extract features from reached web pages as well as an entire sequence of web pages. Unlike previous approaches that extract features from a single web page they have visited [12, 14] or identify malicious URL chains automatically caused without user interactions (i.e., URL redirections) [27, 28, 29], STRAYSHEEP extracts features from the entire sequence of web pages it has actively and recursively followed. That is, STRAYSHEEP analyzes image and linguistic characteristics of reached web pages, browser events (e.g., displaying popup windows and alerts) that occurred before reaching the web page, and browser interactions that lead users to SE attacks. These features represent common characteristics of all SE attacks, i.e., persuading and deceiving users. Therefore, by combining these features to classify sequences, STRAYSHEEP detects various multi-step SE attacks more accurately. We implemented STRAYSHEEP with three distinct modules (*landing-page-collection*, *web-crawling*, and *SE-detection*) to automatically collect landing pages, crawl the web pages branching from them, and detect SE attacks using the results of web crawling, respectively.

To determine the effectiveness of STRAYSHEEP’s three modules, we conducted three evaluations: the rate of collected landing pages leading to SE attacks, the efficiency of web crawling to reach more SE attacks, and accuracy in detecting the attacks. The first evaluation demonstrated that landing pages gathered by the landing-page-collection module led to 20% more SE attacks than Alexa top sites and search results of trend words. The second evaluation demonstrated that the web-crawling module is five times more efficient at crawling than simple crawling modules. The third evaluation revealed that the SE-detection module identified SE attacks with 95.5% accuracy.

We analyzed collected multi-step SE attacks STRAYSHEEP in detail. As a result of categorizing SE attacks, we found that STRAYSHEEP reached a variety of SE attacks such as malware downloads, unwanted browser extension installs, survey scams, and technical support scams. We also found that 30% of SE attacks were reached from 25 different advertising providers.

The main contributions of this chapter are as follows:

- We propose STRAYSHEEP, which detects multi-step SE attacks by automatically and recursively crawling sequences of web pages branching from landing pages. STRAYSHEEP can crawl and detect these attacks by simulating multi-step browsing behaviors of users and extracting features from an entire sequence of web pages.
- We evaluated STRAYSHEEP’s three modules. The landing-page-collection module led to 20% more SE attacks than Alexa top sites and search results of trend words. The web-crawling module was five times more efficient at crawling than a simple crawling module. The SE-detection module identified SE attacks with 95.5% accuracy.
- We conducted a detailed analysis of multi-step SE attacks collected using STRAYSHEEP. We found that STRAYSHEEP collected various SE attacks,

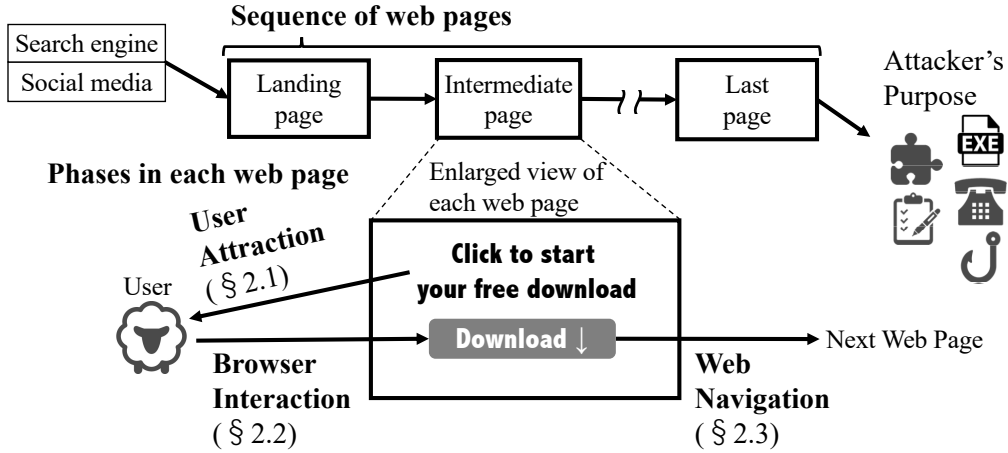


Figure 3.1: Sequence of web pages in multi-step SE attacks and phases in each web page.

not limited to a specific attack. We analyzed attackers' techniques of luring users and browser interactions leading users to attacks.

3.2 Background

SE is used to manipulate people into performing a particular action by exploiting their psychology and has been widely used in various types of web-based attacks, such as malware downloads [24, 30], malicious browser extension installs [31, 32, 33], survey scams [12], and technical support scams [25, 14]. Malware downloads and malicious browser extension installs are achieved by masquerading as legitimate software. Survey scams recruit users attracted by fake survey rewards to trick them into providing sensitive information and accessing web pages controlled by attackers. Technical support scams are carried out by persuading users to make a call to a fake technical support desk and install keystroke loggers, remote access tools, or malware.

Multi-step SE attacks use multiple web pages leveraging different psychological tactics to effectively lure users to the succeeding web page. Figure 3.1 shows a sequence of web pages in multi-step SE attacks and three simplified phases in each web page: *user attraction*, *browser interaction*, and *web navigation*. Therefore, the three phases can be repeated multiple times, starting from a landing page, which appears in response to clicking on a search-engine result or social-media link. Different user interactions on a single web page also lead to different SE attacks.

3.2.1 User Attraction

The user-attraction phase attracts a user psychologically by using the content of the web page to deceive and persuade the user to induce browser interaction [24]. For example, these web pages advertise free downloads of video games, threaten

users with fake virus warnings, and request bogus software updates. The main purpose of this psychological attraction is to make the user interact with an HTML element (e.g., `a` and `div`) that navigates to malware downloads or a web page controlled by an attacker. We call such HTML elements *lure elements*. What is common with lure elements is that they contain words or shapes indicating the behavior or category of an element. A lure element is characterized by its visual effects, such as easily understandable download buttons containing “Click here to download” and movie play buttons containing “WATCH NOW” or a triangle pointing right. A lure element is also characterized by containing words such as “`download-btn`” and “`video-play-link`” in their text content and document object model (DOM) attributes such as `id`, `class`, and `alt`. Multiple lure elements may be arranged on a single web page. In this case, clicking these lure elements results in different SE attacks.

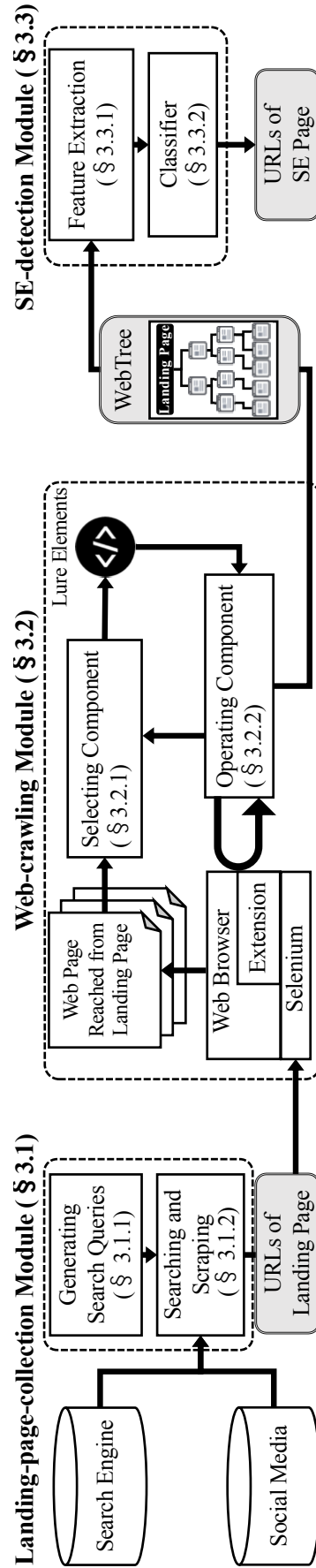


Figure 3.2: STRAYSHEEP overview.

3.2.2 Browser Interaction

Users who are acted upon by the previous user-attraction phase are guided to interact with lure elements on the web page. This browser-interaction phase is mainly an explicit click on the lure element but also includes an unintended click [13]. For example, unintended clicks include clicking an overlay on the entire web page, context menu, and the browser’s back button. These clicks are forcibly generated by JavaScript to redirect a user to a new web page or show a popup window against the user’s intention.

3.2.3 Web Navigation

In the web-navigation phase, browser events occur as a result of browser interaction. These browser events redirect to another web page in the current window or a new window (popup), display alert dialogs, and download files. Web-page redirection occurs in an intermediate step of a multi-step SE attack, guiding the user to the next web page. On the next page, another user attraction, browser interaction, and web navigation could occur again. The purpose of repeatedly making a user reach multiple web pages without completing the attack on one web page is to gradually convince the user and increase the success rate of the attack. For example, to increase the attack-success rate of a user who watches a movie on an illegal streaming site, attackers display a popup that offers a dedicated video player with an alert dialog such as “Please install HD Player to continue.” instead of providing an automatic software download on the first web page. Also, the multiple sequences of web pages in a multi-step SE attack often branch from the landing or intermediate web pages because such web pages contain two or more lure elements leading to different pages.

3.2.4 Problems on Collecting SE Attacks

There are three approaches to automatically collect SE attacks: tracing web traffic, archiving with a crawler, and crawling with a web browser. We give a brief introduction of these approaches and their limitations then present the requirements of collecting SE attacks.

The first approach is of reconstructing SE attacks from web traffic obtained through passive network monitoring [24]. To take measures against SE attacks, revealing a single SE attack reached from the landing page is useful, but uncovering all attacks that branch from web pages is more critical. However, this approach is used to observe only a single sequence of web pages accessed by the user. Also, it cannot be used to observe SE attacks starting from arbitrary web pages. That is, it cannot be used to observe attacks from which a user was not affected but another user could be affected.

The second approach is to visit each web page using crawlers such as Heritrix [34] and GNU Wget. Such crawlers extract links from a downloaded HTML source code of a web page and crawl them recursively. This approach can solve the problem with the first approach, in which it cannot collect SE attacks that the

user did not reach because it can input an arbitrary URL. However, these crawlers can only execute simple content downloads and static content parsing. SE attacks often use web content dynamically generated by JavaScript, which require user interactions to navigate to the next pages; thus, these types of crawlers cannot collect most SE attacks.

The third approach is of web-browser automation using a tool such as Selenium [35]. Web-browser automation enables us to simulate user interaction to all elements on each web page. With this approach, we can solve the problems with the second approach. If we apply the idea of following all links with the second approach to web-browser automation, that is, clicking all elements on each web page, we can ideally collect all multi-step SE attacks derived from a landing page. However, recursively following all elements takes a significant amount of time because the browser requires time to run JavaScript and render web pages.

In summary, to efficiently observe multi-step SE attacks in a short time, the number of elements to crawl must be reduced by selecting possible lure elements from thousands of HTML elements on each web page. To analyze multi-step SE attacks in detail, it is also necessary to recursively follow multiple sequences of web pages that lead to SE attacks derived from a landing page rather than tracing only a single sequence of web pages. Therefore, requirements for collecting and analyzing multi-step SE attacks are crawling with the web-browser-automation approach, selecting lure elements that will likely lead to SE attacks, and recursively interacting with lure elements.

3.3 StraySheep

We propose a system called STRAYSHEEP that automatically collects landing pages that lead to SE attacks, crawls web pages, and detects multi-step SE attacks. STRAYSHEEP consists of three modules: *landing-page-collection*, *web-crawling*, and *SE-detection*. An overview of STRAYSHEEP is shown in Fig. 3.2. The landing-page-collection module gathers URLs of web pages leading to SE attacks by leveraging search engines and social media. The web-crawling module starts recursive web crawling from the URLs collected by the landing-page-collection module, selects and clicks on lure elements, and outputs a WebTree. A WebTree consists of tree-like abstract data, including logs such as web navigation, browser interaction, and snapshot (screenshot and HTML source code) observed at each web page branching from a landing page. The SE-detection module extracts features from a WebTree and identifies the multi-step SE attack using a classification model.

3.3.1 Landing-Page-Collection Module

The landing-page-collection module leverages search engines and social media to find landing pages as input for the web-crawling module. Many SE attacks use web pages that have copyright infringement, such as illegal downloads and free video streaming, to draw the attention of incautious users [13, 36]. To induce a user to access such web pages, attackers use search-engine-optimization techniques [37,

11, 38] and post messages on social media, which include links to the landing pages [39, 40, 41]. Examples of such social-media postings are an instruction video for illegally installing software and a message introducing a free game download site. To collect such landing pages effectively, the landing-page-collection module uses a web-search-based approach consisting of two steps: *generating search queries* and *searching and scraping*.

Generating Search Queries

The landing-page-collection module generates search queries to search the URLs of possible landing pages leading to SE attacks. To generate the search queries, we design the module so that it collects *core keywords*, which stand for a title or name of paid content (e.g., “Godzilla” and “Microsoft Office”) and concatenates them with predefined *qualifiers* (e.g., “free download,” “crack,” and “stream online”). To collect core keywords, the module automatically scrapes popular electronic commerce (EC) sites and online database sites by using predefined scraping logic in accordance with each site and groups the core keywords by content category (e.g., video, software, and music). These core keywords can regularly be updated by recollecting ranking and new release information.

The aim of using qualifiers is (1) limiting the coverage of search results including illegal downloads and streaming, not legitimate sites, and (2) increasing the variation in search results. We manually prepare qualifiers in advance using auto-suggest/related search functions on a search engine. When a user queries a certain word in a search engine, these search functions provide a list of corresponding keyword predictions. We input some titles of paid content to the search engine and collect qualifiers for each category because the qualifiers we require vary depending on the core keyword’s category. For example, qualifiers of video are “stream”, “movie”, and “online”. For another example, qualifiers of the software category are “download”, “crack”, and “key”.

Searching and Scraping

This module retrieves URLs from a search engine or social media by using the generated search queries. It inputs them into the search engine and search forms on social media to widely collect corresponding URLs. Some social media do not always provide comprehensive search results due to a minimum required search function; thus, the module also uses a search engine to collect social-media postings. Finally, it outputs the URLs collected from the search results and links scraped from social media postings as input for the web-crawling module.

3.3.2 Web-Crawling Module

The web-crawling module automates a web browser to recursively crawl a URL collected by the landing-page-collection module and outputs a WebTree as a crawling result. Figure 3.3 shows a conceptual model of a WebTree representing sequences of web pages derived from the landing page and visited by the web-clawing module.

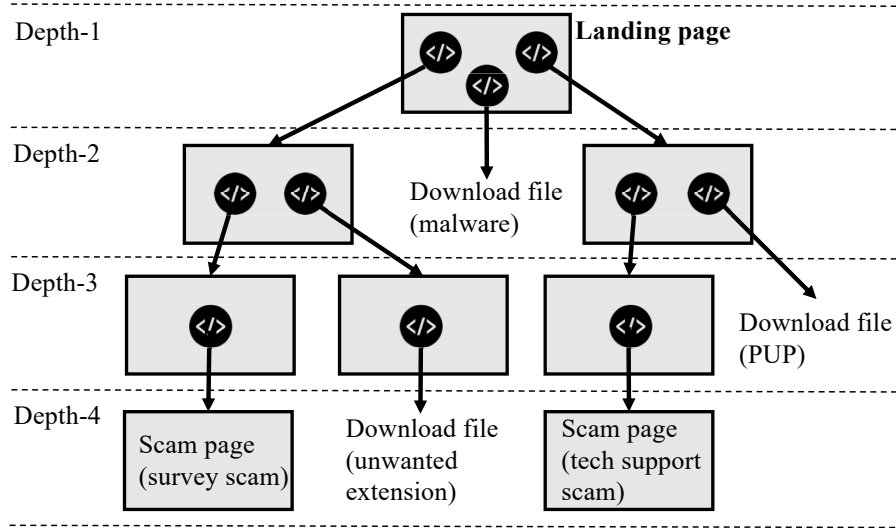


Figure 3.3: Conceptual model of WebTree.

The web-crawling module starts from the landing page, clicks on multiple lure elements on the web pages, and recursively follows multiple web pages derived from the landing page. The depth indicates the recursion count of web crawls. The depth increases when this module reaches a web page that completes loading and is waiting for browser interaction. This module uses Selenium and our original browser extension to automatically control and monitor a web browser. For the prototype of our system, we chose Google Chrome as a browser, but Selenium can also control other web browsers; thus, the web-crawling module can use different browsers. In the following section, we describe two components of the web-crawling module: *selecting* and *operating*.

Selecting Component

The selecting component collects a lure element that causes web navigation leading to SE attacks by analyzing an HTML source code and a screenshot of a web page. As mentioned in Section 3.2.1, a word representing the category or action of an element tends to be used for the lure element’s DOM attributes, text content, and the text drawn inside the button graphic, for example, “download” in “`download-btn`” of the `class` attribute and “click” in “Click Now” of the text drawn inside a clickable button. To select elements containing such keywords as lure elements, we design the selecting component so that it parses an HTML source code and executes image processing of a web page’s screenshot. The purpose of the selecting component is not to accurately detect elements leading to SE attacks but to select possible lure elements to reduce the number of elements with which to interact. By following only selected elements, the web-crawling module can efficiently reach diverse SE attacks. Note that there could be multiple lure elements on the same web page; thus, this component analyzes all elements on the web page. The reason the selecting component also executes image processing

is to complement the acquisition of character strings drawn in the button image (i.e., `img` element), which cannot be acquired from the HTML source code. This component also identifies lure elements by their shape such as the triangular video play button.

We explain a statistical method of preparing keywords for selecting lure elements. We compare elements that have actually redirected users to SE attacks (lure elements) with other elements that have not redirected users to any SE attacks (*non lure elements*) and extract words specific to lure elements. More specifically, we extract attribute, text content, and strings drawn on buttons from the collected elements and divide these words into two documents: a document of lure elements and one of non lure elements. We then calculate the term frequency-inverse document frequency (tf-idf) of the two documents and manually choose words that have high tf-idf values from the lure-element document. The process of keyword selection is shown in Section 3.4.2.

In the analysis of HTML source codes, if an element matches at least one of the following four rules, this component determines it to be a lure element.

- One of the keywords is used in the element’s text content.
- A keyword is set in `id`, `class`, or `alt` DOM attributes.
- A keyword is used as the file name indicated by the URL of the link (`a` element) or image (`img` element).
- An executable file (e.g., `.exe` or `.dmg`) or a compressed file (e.g., `.zip` or `.rar`) is used as a link extension.

The purpose of the analysis of image processing is to find rectangular buttons and video play buttons. This component extracts character strings written in each element from the screenshot and matches keywords used in the HTML source code analysis. This component leverages OpenCV to find rectangle contours representing the button areas in the screenshot and identify the coordinates and size of buttons. It also uses optical character recognition (OCR) using Tesseract OCR [42] to extract character strings from the rectangles the component found. This component executes keyword matching with extracted character strings and determines an element containing one of the keywords in the area to be a lure element. To acquire video play buttons as lure elements, the module also finds a triangle contour pointing right. Finally, the component outputs multiple lure elements that may lead to SE attacks from the web page.

Operating Component

The operating component executes browser interactions (i.e., clicking on lure elements), monitors web navigation, and constructs a WebTree. It simulates clicking on lure elements with the `CTRL` key pressed to open the web page in a new browser tab because the current page may be transferred to another web page by a simple clicking. As a result, links or popup windows can be opened in new tabs without

changing the original tab. The operating module also clicks a `body` element, `body` element with context click, and the browser’s back button to simulate unintended clicks described in Section 3.2.2. When the new tab is opened, the selecting component finds lure elements again, and the operating component executes browser interactions with a depth-first order, unless it reaches a predetermined maximum depth. We explain the maximum depth we used in the following experiment in Section 3.4.2.

The operating component also monitors web navigation. For monitoring JavaScript function calls, this component hooks the existing JavaScript function to detect the executed JavaScript function name and its argument. The JavaScript functions to be monitored by this component are `alert()`, `window.open()`, and the installation function of the browser extension (e.g., `chrome.webstore.install()`). The function `alert()` is frequently used in SE attacks that threaten a user by suddenly displaying dialog with messages inducing user anxiety. The `window.open()` function opens a new browser window and is used for popup advertisements. This component also hooks the installation function of the browser extension and detects what type of browser extension was installed from the argument. This component also monitors URL redirection, which navigates a user to another URL. URL redirection is divided into client-side redirection and server-side redirection. A web browser may conduct client-side redirection such as JavaScript function `location.href` when this component clicks the lure element. On the other hand, a web server conducts server-side redirection to navigate to another web page before loading a web page. This component monitors the URLs the browser passed during server-side redirection to identify the server that navigates users to SE attacks, such as advertising providers.

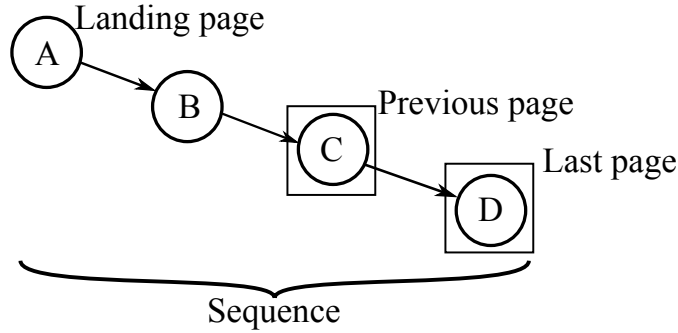
The operating component conducts browser interactions and monitors web navigation until it finishes clicking on all selected lure elements. This component aggregates information from sequences of web pages (i.e., screenshots, the HTML source codes of web pages, browser interactions, and web navigation) and finally outputs a WebTree as input for the SE-detection module.

3.3.3 SE-Detection Module

The SE-detection module extracts features from a WebTree output by the web-crawling module and identifies multi-step SE attacks using a classification model. This module first extracts *sequences* from the WebTree. A sequence is defined as a series of rendered web pages from the landing page (a root node) to the last pages (leaf nodes). Note that the sequence does not represent a URL redirection chain (an automatic process of forwarding a user to another URL multiple times) but a series of displayed web pages through user interaction. This module then extracts features from each sequence that reaches web pages of depth of two or more. Unlike conventional methods that examine structural similarity of URL redirection chains [27, 28, 29], this module extracts features specific to multi-step SE attacks from the entire sequence: contents of web pages, browser interactions that trigger page transitions, and web navigation. Finally, it identifies whether the last page of each sequence is the SE page using a classifier and outputs URLs of

Table 3.1: List of features SE-detection module uses.

	Target	Feature	# of dimensions
User attraction	Last page	Image features (Bag-of-visual-words)	128
	Last page	Color histogram	30
	Last page	Linguistic features (Doc2Vec)	300
	Last page	HTML tag histogram	40
	Last page	Length of text field	1
	Previous page	Image features (Bag-of-visual-words)	128
	Previous page	Color histogram	30
	Previous page	Linguistic features (Doc2Vec)	300
	Previous page	HTML tag histogram	40
	Previous page	Length of text field	1
Browser interaction	Sequence	# body clicks	1
	Sequence	# body context clicks	1
	Sequence	# left clicks	1
	Sequence	# back button clicks	1
	Sequence	# <a>tags clicked	1
	Sequence	# <iframe>tags clicked	1
	Previous page	Coordinates (x,y)	2
	Previous page	Size (width, height)	2
Web navigation	Sequence	Depth	1
	Sequence	# alert dialogues	1
	Sequence	# popup windows	1
	Sequence	# server side redirections	1
	Sequence	# client side redirections	1
	Last page	File downloads	1
	Last page	Extension installs	1
	Previous page	File downloads	1
	Previous page	Extension installs	1

**Figure 3.4:** Example of extracting features from a sequence.

the detected web pages. Ground truth data for identifying SE attacks is explained in Section 3.4.3.

Feature Extraction

To classify web pages that trick users into interacting, it is common to use information that can be acquired after visiting the web page, such as image and HTML features [13, 12]. However, if a classifier uses such features, it cannot detect an SE page similar to the legitimate page, such as a fake software-update web page that

closely resembles a legitimate Flash update page or fake infection-alert page using the logo of security vendors. Therefore, we designed feature vectors using not only features extracted from a single web page but also all features extracted from the entire sequence. Specifically, it analyzes the last page of the sequence, page before the last page (previous page), and the entire sequence, as shown in Fig. 3.4. Table 3.1 shows features extracted from each sequence and grouped into the three phases of SE attacks: user attraction, browser interaction, and web navigation. To the best of our knowledge, STRAYSHEEP is the first system that automatically collects these features from the entire sequence by recursively crawling web pages from the landing page. In terms of the user-attraction-based features, STRAYSHEEP extracts appearance, meaning of a document, and structure of HTML from the last and previous pages. It then finds features based on browser interaction, such as actions performed on the web pages and lure elements from the previous page and entire sequence. The SE-detection module also analyzes web navigation that occurred on the last page, the previous page, and the entire sequence. We explain a feature extraction method for each SE-attack phase below in detail.

User Attraction The appearance of a web page and the semantic properties of text content include the intention of the attacker to trick a user. The HTML document structure is also an important indicator for analyzing the similarity of web pages using the same document template. The SE-detection module extracts image and linguistic features from the last and previous pages of the sequence. It also calculates an HTML tag histogram, RGB color histogram, and the length of the text field from both the last and previous pages. These features are useful for identifying web pages that use the same page templates and images as other malicious web pages. To extract image features, we use AKAZE [43], which is a bag-of-visual words algorithm that detects local image features. The SE-detection module extracts 128-dimensional image features from the screenshots of the last and previous pages using a trained model we previously constructed. We use Doc2Vec [44] as a document-modelling algorithm to extract linguistic features. The purpose of this is to capture attackers’ intentions, such as deceiving or threatening users, based on linguistic characteristics. The module extracts the 300-dimensional features from the text content of the last and previous pages by using a doc2vec model trained beforehand. The text content of a web-page document is extracted by cleaning out HTML tags from an HTML source code. The SE-detection module also calculates a histogram of the RGB (red, green, and blue) values of the screenshot with ten bins for each color and a histogram of HTML tags of the text content. This module uses up to 40 HTML tags (e.g., `div`, and `img`) frequently appearing on the web pages we collected in advance. It counts the number of characters in the text content.

Browser Interaction The SE-detection module analyzes lure elements and actions that caused SE attacks. Browser interaction is an important indicator that characterizes multi-step SE attacks because the destination web pages change depending on the types of actions taken by users and clicked elements. To extract features from browser interactions, we design this module so that it counts the number of left clicks and unintended clicks (body clicks, body-context clicks, and back-button clicks) the web-crawling module performed in the sequence. This

module also counts the types of clicked lure elements (`a` and `iframe`) in the sequence and determines the size (x,y) and coordinates (width, height) of lure elements on the previous page.

Web Navigation The SE-detection module analyzes browser events that occurred as a result of browser interaction. File downloads and extension install indicate events that are directly related to SE attacks such as malware downloads and unwanted extension installs. Since SE attacks are often delivered via advertising providers, redirection has characteristics unique to SE attacks. The method of navigation (e.g., redirection and popup window) is important for analyzing SE attacks. This module determines whether file downloads and extension installs occurred on the last and previous pages. It counts the times popup windows were displayed and the number of URLs observed during server-side and client-side redirection. It also checks the number of displayed alert dialogues and the length of the sequence, i.e., crawling depth.

Classifier

We combine the features extracted from sequences to create features vectors and construct a binary classifier to identify SE web pages. We use Random Forest as a learning algorithm because we can measure the importance of each feature that contributes to the classification. Evaluation results compared with other algorithms are given in Section 3.4.5.

Table 3.2: Comparison between proposed and previous systems

Collecting method	STRAYSHEEP	Surveylance [12]	Rafique et al. [13]	ROBOVIC [25]	Srinivasan et al. [14]	TrueClick [36]	WebWitness [23, 24]
	Active	Active	Active	Active	Active	Active	Passive
Interacting with HTML elements	●	● (survey filling)	● (overlay video ad)	○	○	○	○
Following multiple lure elements on web page	●	○	○	○	○	○	○
Features (image)	●	●	●	○	○	●	○
Features (HTML)	●	●	●	○	○	○	○
Features (linguistic)	●	●	●	● (heuristic)	●	○	○
Features (sequence)	●	○	○	● (heuristic)	○	○	● (network level)
Source of landing-page collection	Search engine, social media	Search engine	Search engine	Parked domain, URL shortener	Search engine	File sharing site	Depending on users
Type of SE attacks to collect	All multi-step SE attacks	Survey scams	FLIS aggregator pages	Tech support scams	Trick bannars	SE downloads	

●: Fully Covered, ●: Partially Covered, ○: Not Covered

Table 3.3: Results of web crawling starting from landing page collected with each method.

	Search Engine (STRAYSHEEP)	Social Media (STRAYSHEEP)	Alexa Top Sites (Baseline)	Trend Words (Baseline)	Core Keywords (Baseline)
# of landing pages	5,000	5,000	5,000	5,000	5,000
# of landing pages lead to SE attacks	1,060 (21.2%)	808 (16.2%)	33 (0.7%)	65 (1.3%)	46 (0.9%)
# of unique visited URLs	50,587	25,722	27,818	16,240	30,133
# of unique URLs of visited SE pages	4,716 (9.3%)	1,633 (6.3%)	80 (0.3%)	105 (0.6%)	628 (2.1%)
# of unique visited domains	4,984	3,619	8,046	3,537	4,507
# of unique domains of visited SE pages	446 (8.9%)	151 (4.2%)	42 (0.5%)	27 (0.6%)	95 (2.1%)
# of unique downloaded Malware samples	160	186	50	3	41

3.4 Evaluation

We evaluated the three modules of STRAYSHEEP (landing-page-collection, web-crawling, and SE-detection). We first evaluated the qualitative advantage of STRAYSHEEP by comparing it with previous systems for collecting SE attacks. We then evaluated the effectiveness of the landing-page-collection module by comparing its two collection methods (search engine and social media) to three baseline URL-collection methods in terms of the number of landing pages leading to SE attacks and total visited malicious pages and domain names. Also, we conducted a crawling experiment to determine the efficiency of the web-crawling module by comparing its crawling method with two baseline crawling methods in terms of the number of malicious domain names reached per unit of time. Finally, we confirmed the effectiveness of the SE-detection module in terms of detection accuracy.

3.4.1 Qualitative Evaluation

We qualitatively compared STRAYSHEEP with the previous systems to collect SE attacks from five perspectives. Table 3.2 summarizes the results.

Collecting method. The previous systems [23, 24] for passively observing HTTP traffic to analyze SE attacks, can only collect attacks triggered by users’ real download events. On the other hand, actively crawling arbitrary web pages with STRAYSHEEP enables us to proactively detect SE attacks before many users reach the web pages.

Interacting with elements. To observe multi-step SE attacks, we need to interact with HTML elements and recursively follow page transitions. Surveylance [12] is a system to detect survey gateways, which are landing pages displaying survey requests, and interact with their survey content and survey publisher sites. A system proposed by Rafique et al. [13] detects free live streaming (FLIS) pages and interacts with overlay video ads on them. While these systems focus on survey scams or FLIS services, STRAYSHEEP can collect various SE attacks and observe different types of survey scams originating from web pages deeper than the landing pages (see Section 3.5.1).

Extracting features. As stated in Section 3.3.3, STRAYSHEEP extracts features such as images, HTML structures, and linguistic context from reached web pages and analyzes sequences to accurately detect multi-step SE attacks. As shown in Table 3.2, none of the previous systems use all the features used in STRAYSHEEP.

Source of landing-page collection. STRAYSHEEP collects landing pages from two common platforms: search engines and social media. STRAYSHEEP is the only system that uses both platforms.

Type of SE attacks to collect. While the previous systems are limited to detecting a specific attack, STRAYSHEEP collects various multi-step SE attacks by following lure elements on each web page.

In summary, STRAYSHEEP is the first system to collect multi-step SE attacks not limited to specific attacks by recursively following multiple lure elements on web pages. STRAYSHEEP also detects multi-step SE attacks by extracting various types of features from reached web pages and sequences.

3.4.2 Experimental Setup

We implemented STRAYSHEEP for Google Chrome 69 with Ubuntu 16.04. It simultaneously ran up to 32 instances on a virtual machine assigned with Intel Xeon 32 logical processors and 256-GB RAM. For the browser setting, a user agent was set as Google Chrome of Windows 7, and browser cookies were reset for every landing-page access. Our crawling experiment spanned from November to December 2018, and STRAYSHEEP used a single IP address. We need to set a timeout for performance evaluation because the two baseline web-crawling modules mentioned in Section 3.4.4 require an enormous amount of time (a few weeks at most) to complete web crawling. About 90% of web crawling conducted with STRAYSHEEP finished within an hour in our preliminary experiment (similar results are shown in Fig. 3.5); therefore, we set the timeout to one hour. To find the best maximum depth for collecting the most malicious domain names when we used the timeout, we changed the depth from two to six. The number of malicious domain names monotonically increased up to depth four and decreased as the depth increased. Therefore, we set the maximum depth to four in the following experiments.

To determine keywords for selecting lure elements, we followed the statistical method described in Section 3.3.2. First, we manually browsed landing pages (e.g., game download, movie streaming, and torrent sites) and clicked on various HTML elements. We also browsed intermediate pages navigated from them, such as fake virus alerts, file downloading, and advertising pages served by URL shorteners. We then gathered 1,447 lure elements from 978 web pages, which we confirmed finally led to SE attacks. To determine if the reached web pages contained SE attacks, we used URL/domain blacklists (Google Safe Browsing, Symantec DeepSight [45], and hpHosts [46]) to match visited web pages and checked the MD5 hash values of the downloaded binaries with VirusTotal. We defined an *SE page*, which matched the blacklist whose label was associated with SE attacks (e.g., phishing, tech support scam, and survey scam) or started downloading malware or potentially unwanted programs (PUPs) [47, 48]. We used the same method of checking SE pages in the following experiments. We randomly selected 5,000 non-lure elements that did not redirect to any SE pages from the landing and intermediate pages. We created lure and non-lure elements’ documents containing words extracted from attributes and text content to calculate tf-idf. Finally, we chose 31 keywords specific to the lure elements by excluding proper nouns (e.g., game and movie titles) and words with zero tf-idf values.

3.4.3 Effectiveness of URL Collection

To show the effectiveness of STRAYSHEEP’s landing-page-collection module, we validated landing pages collected by this module; thus, we used the web-crawling module to recursively crawl the landing pages and identified whether visited web pages caused SE attacks. We compared the number of collected landing pages that led to SE attacks across the five methods, i.e., the landing-page-collection module’s two methods (search engine and social media) and three baseline methods (Alexa top sites, trend words, and core keywords). We collected 5k landing pages for each

method.

Search Engine (StraySheep’s Method) This method collected a total of 3k core keywords from EC/database sites, such as amazon.com, steampowered.com, billboard.com, and imdb.com, which we chose from Alexa top 500 sites. These core keywords were divided into five categories: software (game and applications), video (movie, animation, and TV series), music, eBook, and comic. We can increase the variety of landing pages by collecting different types of core keywords, which are often used in illegal sites to lure users. This method generated 90k search queries by concatenating the core keywords with an average of 30 predefined qualifiers for each category. When we search for only core keywords, many legitimate sites, such as official sites of movies or games, are included in the search results. However, we can collect more landings pages leading to SE attacks, including illegal sites, by adding qualifiers to core keywords. It searched the queries using Microsoft Bing Web Search API [49] (Bing API) and collected about 1M unique URLs. In that web search, it gathered URLs from up to 30 search results for each search query. Note that the search queries containing the same core keywords with different qualifiers sometimes returned duplicate search results, and some search queries returned less than 30 search results. Finally, we randomly sampled 5k URLs from the collected 1M URLs to crawl for the crawling experiment.

Social Media (StraySheep’s Method) This method also searched seven social-media platforms (Facebook, Twitter, Youtube, Dailymotion, Vimeo, Flickr, and GoogleMap) using the same search queries as the above search-engine experiment. Attackers post fake messages on social media such as free downloads of games and streaming of movies to lure users into accessing their links. By collecting such social media posts, we can also gather landing pages that do not appear in search engine results. This method extracted links from posting messages (from Facebook, Twitter, and Flickr), descriptions of uploaded video (from Youtube, Dailymotion, and Vimeo), and descriptions of GoogleMap’s My Maps. It used search forms on Youtube, Dailymotion, and Facebook because they have flexible search mechanisms and searched Bing API for the other social-media platforms to gather up to 30 social media postings for each search query. It searched for 10k search queries (sampled from 90k search queries) for each social-media platform and found a total of 130k unique social-media postings. These search queries often returned less than 30 search queries. This method then gathered 45k unique links by scraping these 130k social-media postings. Some social-media postings did not include any links or included multiple links. Finally, we randomly sampled 5k URLs from the 45k links for the crawling experiment.

Alexa Top Sites (Baseline Method) We gathered the top 5k domain names from Alexa top sites and converted them to 5k URLs by adding “http://” to the domain names.

Trend Words (Baseline Method) We searched the top 1k trend words collected from Google Trends using Bing API and randomly selected 5k URLs from the 30k search results (retrieved 30 results per query).

Core Keywords (Baseline Method) We simply searched the same set of 3k core keywords we used for the above Search Engine method and randomly sampled 5k URLs from the 90k search results (retrieved 30 results per query).

Table 3.4: Results for each web-crawling module.

	STRAYSHEEP		ElementCrawler		LinkCrawler	
	SE pages	Total	SE pages	Total	SE pages	Total
# of Total pages	9,374 (5.4%)	173,060	13,559 (2.4%)	562,708	19,241 (3.6%)	540,822
# of Unique visited pages	6,283 (8.5%)	73,906	5,998 (3.1%)	191,901	5,445 (3.0%)	180,920
# of Unique visited domains	513 (6.7%)	7,660	437 (3.2%)	13,545	335 (3.4%)	9,734

Table 3.3 lists the results of web crawling for each method. The landing pages that led to SE attacks and collected with the search-engine and social-media methods accounted for 21.2 and 16.2% for each 5k landing pages. While, those of the three baseline methods (Alexa top sites, trend words, and core keywords) were much smaller, 0.7, 1.3, and 0.9%, respectively. From the results of the search-engine and social-media methods, the numbers of unique visited URLs and domain names were larger than those of the three baseline methods. Since StraySheep’s methods, which use qualifiers, collected about 20 times as many landing pages lead to SE attacks as the baseline method (Core Keywords) when using the same set of core keywords, qualifiers are effective in collecting landing pages. The number of malware samples reached from the URLs collected with the search-engine and social-media methods was also larger than that of the other three methods.

3.4.4 Efficiency of Web Crawling

To evaluate the efficiency of STRAYSHEEP’s web-crawling module, especially the function to follow lure elements selected by the selecting component, we compared the ratio of SE pages in visited web pages and the time to reach SE attacks among three web-crawling modules: that of STRAYSHEEP’s web-crawling module and two baseline web-crawling modules. Then, we compared the crawling performance of STRAYSHEEP with that of TrueClick [36].

Comparison of crawling performance with baseline web-crawling modules and StraySheep We implemented the two baseline modules: *ElementCrawler*, which extracts all visible elements on the web pages and simply clicks them, and *LinkCrawler*, which purely selects all the link elements (HTML a tag with href attribute) and clicks them. Note that elements selected by ElementCrawler contain all those selected by LinkCrawler or STRAYSHEEP’s web-crawling module. ElementCrawler and LinkCrawler are alternative implementations of STRAYSHEEP’s web-crawling module, which are implemented by replacing the selecting component (see Section 3.3.2) with the function of selecting all elements or all links from an HTML source code. The landing pages we input to the three modules were the same 10k URLs as those collected by the landing-page-collection module, as mentioned in Section 3.4.3, which are the 5k URLs collected from a search engine and another 5k URLs collected from social media. We newly crawled the 10k landing pages using ElementCrawler and LinkCrawler under the same condition mentioned in Section 3.4.3. We compared these crawling results with those of the above experiment in which STRAYSHEEP’s web-crawling module crawled the 10k landing pages. In the same manner as the above experiment, we identified SE pages using blacklists and VirusTotal.

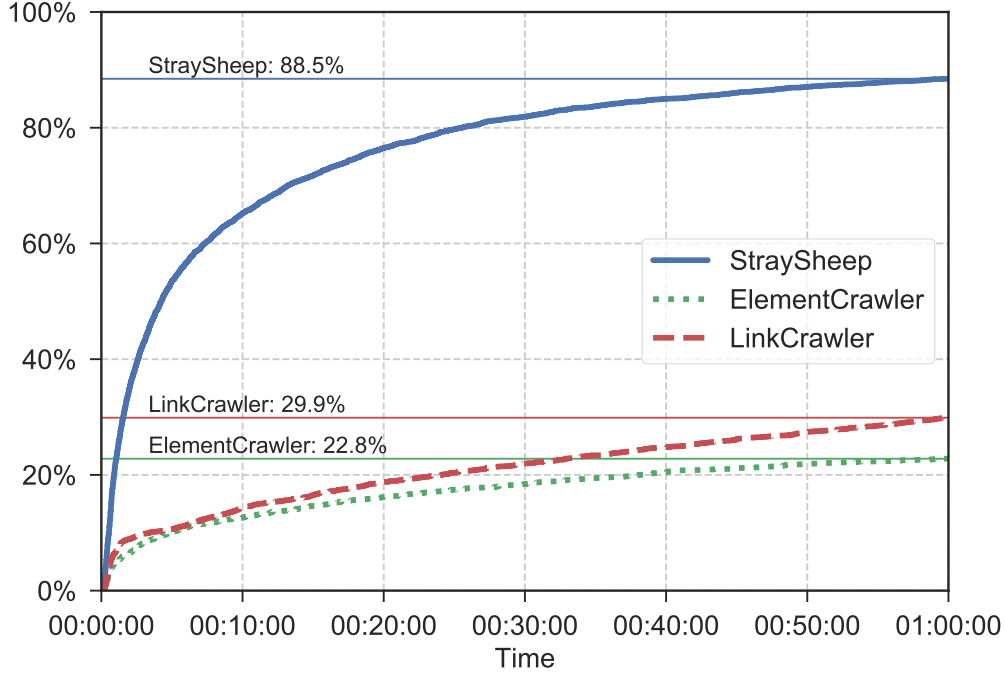


Figure 3.5: CDF of time taken to complete web crawling for each landing page within a 1-hour timeout. Horizontal lines mean the percentage of web crawling completed before timeout.

Table 3.4 shows the number of total pages, unique visited pages, and domain names for each web-crawling module. The numbers of unique visited pages and domain names of SE pages visited with STRAYSHEEP’s web-crawling module were 6,283 pages and 513 domain names, which were larger than those of the baseline modules, and the percentages of pages and domain names of SE pages were also larger than those of the baseline modules (8.5 and 6.7%, respectively). Although the numbers of total pages of ElementCrawler and LinkCrawler were three times larger than that with STRAYSHEEP’s web-crawling module, STRAYSHEEP’s web-crawling module had the best percentage (5.4%) for all SE pages. This is because STRAYSHEEP’s web-crawling module selected lure elements from thousands of elements to crawl web pages likely to cause SE attacks, while ElementCrawler and LinkCrawler simply took turns to click elements and reached many benign web pages. In short, ElementCrawler may crawl all potential SE attacks by taking an enormous amount of time; however, STRAYSHEEP can reach SE attacks in a shorter time by selecting lure elements.

Next, we analyzed the efficiency of each web-crawling module by comparing the time taken to complete visiting web pages branching from the landing page. Figure 3.5 is a cumulative distribution function (CDF) of the time for each web-crawling module, which shows the percentage of web crawling finished at a certain time out of all web crawling starting from 10k landing pages. We found that 88.5% of STRAYSHEEP’s web crawling module finished within one-hour timeout. In contrast, ElementCrawler finished only 22.8% of web crawling within the timeout,

Table 3.5: Crawling efficiency of each web-crawling module.

	STRAYSHEEP	ElementCrawler	LinkCrawler
# of unique domains of visited SE pages	513	437	335
Total crawling time [sec]	8,429,288	29,698,118	28,421,460
Crawling efficiency [/sec]	$6.1 \cdot 10^{-5}$	$1.5 \cdot 10^{-5}$	$1.2 \cdot 10^{-5}$

and LinkCrawler finished 29.9%. The average time to complete the web crawling for each landing page was 14 minutes for STRAYSHEEP’s web-crawling module, 49 minutes for ElementCrawler, and 47 minutes for LinkCrawler.

To measure the web-crawling modules’ ability to reach SE attacks per total crawling time, we calculated *crawling efficiency*.

$$\text{Crawling Efficiency [}/\text{sec]} = \frac{\# \text{ Unique domains of visited SE pages}}{\text{Total crawling time [sec]}}.$$

Crawling efficiency indicates the ability to reach the unique domain names of SE pages per unit of time. Higher crawling efficiency implies that the module can efficiently reach new SE pages.

We show the crawling efficiency for each web-crawling module in Table 3.5. Total crawling time in Table 3.5 represents the sum of the times to complete crawling 10k landing pages. The crawling efficiency of STRAYSHEEP’s web-crawling module was 4.1 times higher than that of ElementCrawler and 5.1 times higher than that of LinkCrawler, making it the most efficient module to reach SE attacks. As described in Section 3.3.2, since STRAYSHEEP’s web-crawling module detected lure elements that led to SE pages by using the selecting component, it visited more SE pages in less time than the two baseline modules.

We also examined the ability to visit SE attacks that can be reached via multiple web pages. Table 3.6 shows the number of unique domain names observed at each depth. Note that each depth may have duplicate domains because the web-crawling modules visited the same domains at different depths. Also, the number of domain names observed at a depth of 1 was the same because each module visited the same landing pages. The number of domains of SE pages show that STRAYSHEEP’s web-crawling module efficiently visited more domains of SE attacks at every depth than the baseline modules. As the depth became deeper, the percentages of an SE page’s domains that ElementCrawler and LinkCrawler detected decreased. On the contrary, the percentages of an SE page’s domains that STRAYSHEEP’s web-crawling module visited were 5.5% at a depth of 2, 6.3% at a depth of 3, and 9.4% at a depth of 4; thus, the deeper STRAYSHEEP’s web-crawling module crawled, the more it efficiently visited SE pages. As described in Section 3.3.2, STRAYSHEEP selects lure elements that lead to SE attacks so that the web-crawling module can reach more of an SE page’s domains even though it crawls deeper.

Comparison of crawling performance with TrueClick and StraySheep

We also conducted an additional experiment comparing the crawling performance of STRAYSHEEP with that of TrueClick in terms of the ability to reach SE pages and collect malware executables. TrueClick is a tool that distinguishes fake advertisement banners (trick banners) from genuine download links. TrueClick has

Table 3.6: Unique domain names observed at each depth.

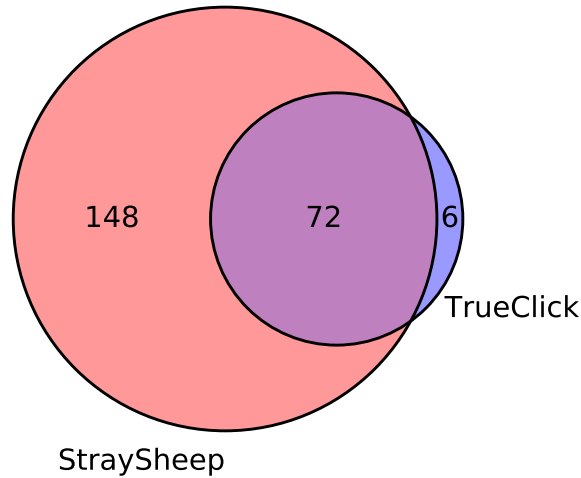
Depth	STRAYSHEEP		ElementCrawler		LinkCrawler	
	SE	Total	SE	Total	SE	Total
1	91 (2.2%)	4,187	91 (2.2%)	4,187	91 (2.2%)	4,187
2	223 (5.5%)	4,043	159 (4.1%)	3,882	126 (4.6%)	2,726
3	231 (6.3%)	3,692	171 (3.5%)	4,895	148 (3.9%)	3,844
4	348 (9.4%)	3,685	299 (2.8%)	10,694	303 (3.4%)	8,939

Table 3.7: Results of web crawling using STRAYSHEEP and TrueClick.

	Unique visited pages (domain names)	Unique visited SE pages (domain names)	Unique malware samples
STRAYSHEEP	48,524 (5,809)	3,897 (219)	266
TrueClick	7,917 (2,978)	523 (78)	1

Table 3.8: Unique SE pages observed at each depth by using STRAYSHEEP and TrueClick.

depth	SE pages crawled using StraySheep (domain names)	SE pages crawled by TrueClick (domain names)
1	97 (44)	97 (44)
2	845 (86)	356 (35)
3	1068 (104)	48 (12)
4	2302 (106)	25 (12)
Unique SE pages	3,897 (219)	523 (78)

**Figure 3.6:** Overlap of SE pages' domain names observed using STRAYSHEEP and TrueClick.

the similar purpose as STRAYSHEEP for finding HTML elements that are made to deceive users and direct to a malicious site or malware executable, but it only finds elements displayed by advertising providers regardless of the web site owner's

intention.

Since the source code of TrueClick has not been published, we re-implemented TrueClick based on the implementation details of the paper [36] using a manually collected dataset containing 87 trick banners and 51 genuine banners, which is almost equivalent to the amount of the original dataset (165 trick banners and 94 genuine download links), to train a machine learning model. The trained model identifies trick banners with 98.6% accuracy. We then created a baseline crawling module by replacing STRAYSHEEP’s selecting component (Section 3.3.2) with TrueClick implementation.

To equivalently compare the crawling results under the same experimental condition in terms of the period of landing-page collection and web crawling, we have collected 5k URLs in the same manner as that mentioned in Section 3.4.4 and crawled them using both STRAYSHEEP’s web-crawling module and the baseline module as of November 2019. Since this experiment was conducted at a different period than the one explained above, we newly collected 2.5k landing pages each from a search engine and social media as input URLs. Table 3.7 summarizes the results. STRAYSHEEP visited more SE pages than TrueClick because it follows not only trick banners but also buttons and links intentionally placed by web site owners to lead to SE attacks. While STRAYSHEEP successfully downloaded 266 malware samples, TrueClick downloaded only 1 malware sample. This is because, in most cases, genuine download links distribute malware samples instead of trick banners on web pages redirected from the first trick banners on landing pages. Table 3.8 shows the number of unique SE pages observed at each depth. Similar to the results in Table 3.6, STRAYSHEEP reached more SE attacks as it crawled deeper. Conversely, the number of SE pages that TrueClick reached considerably decreased deeper than depth three. The reason for this is that as we crawl deeper from the landing page, the number of trick banners decreases. Additionally, intentionally placed lure elements including genuine download links mainly lead to SE attacks at deeper depths. Figure 3.6 shows the overlap of SE pages’ domain names observed using each crawler. Although STRAYSHEEP did not visit a small number of SE pages dynamically served by ads, it covered most of the SE pages observed by TrueClick. In summary, to collect more multi-step SE attacks, we need not only to detect trick banners but also follow lure elements.

3.4.5 Evaluating the SE Detection Module

We evaluated the effectiveness of STRAYSHEEP’s SE-detection module using WebTrees, which are the outputs of STRAYSHEEP’s web-crawling module. We used 30k WebTrees constructed from the results of web crawling starting from 30k landing pages. These WebTrees consisted of the 10k landing pages crawled by STRAYSHEEP’s web-crawling module (Section 3.4.4) and additional 20k landing pages. The 20k landing pages were collected and randomly sampled in the same manner as for the 10k landing pages mentioned in Section 3.4.3. We carried out the web crawling in the same environment in the same period to output additional 20k WebTrees.

To create datasets for evaluation, we extracted malicious and benign sequences

from the 30k WebTrees. The 30k WebTrees contained a total of 243,914 unique web pages (13,415 unique domains) of visited web pages. To label these web pages as SE pages, we used blacklists (same as in Sections 3.4.3 and 3.4.4) and VirusTotal. We labeled 51,501 unique web pages (unique 1,066 domains) as SE pages and extracted 1,066 sequences, which reached 1,066 different domain names from distinct landing pages. We excluded unreachable or parking domain pages and created 1,045 sequences as the malicious dataset. To create a benign dataset, we randomly sampled 1,045 sequences that did not visit SE pages.

To evaluate the detection accuracy of the SE-detection module, we conducted a 10-fold cross-validation (CV) on the labeled dataset. The SE-detection module classified our dataset with a precision of 97.4%, recall of 93.5%, and accuracy of 95.5%. When we changed the learning algorithm from random forest to support vector machine, logistic regression, and decision tree, their accuracies were 93.6%, 90.8%, and 90.7%, respectively. The percentage of feature importance accounted for 65.2% of features extracted from the last page (*last page features*), 28.2% of features extracted from the previous page (*previous page features*), and 6.6% of features extracted from the entire sequence (*sequence features*), as shown in Table 3.1.

Although the SE-detection module can accurately identify multi-step SE attacks, the evaluation result contained some false positives and false negatives. We discuss ideas for reducing these false positives and false negatives. The false positives included popular shopping and casino sites that were redirected from pop-up ads triggered by unintended click. We can reduce these false positives by extracting long-term stable and popular domain names from domain lists such as Alexa to create a white list. We also found false negatives that were listed on blacklists but not detected by the SE-detection module. Since we only trained web pages written in English in this experiment, Some web pages written in non-English languages were included in false negatives. Ad providers may change web pages to serve depending on the region of a source IP address. Therefore, we can accurately detect multi-step SE attacks by automatically translating web pages to English or by training web pages written in a specific language corresponding to the region of the source IP address.

To show the relationship between detection accuracy and features, we divided the features into four feature sets: last page, previous page, sequence, and the combination of last and previous page (feature sets without our proposed sequence features). We conducted 10-fold CVs using all feature sets and four divided feature sets with the same dataset discussed in Section 3.4.5. Figure 3.7 shows the receiver operating characteristic (ROC) curves for the classification results. The most accurate result was the CV using all features in order of the combination of last and previous page, last page, sequence, and previous page feature sets. The area under the curve (AUC) for each result was 0.965, 0.955, 0.948, 0.923, and 0.829. This experiment revealed that our original page-level features that analyzed linguistic, image, and HTML characteristics were useful in detecting various types of SE attacks, i.e., not limited to a specific SE attack. However, we can classify more accurately by using the features of a previous page and sequence together that STRAYSHEEP automatically collects.

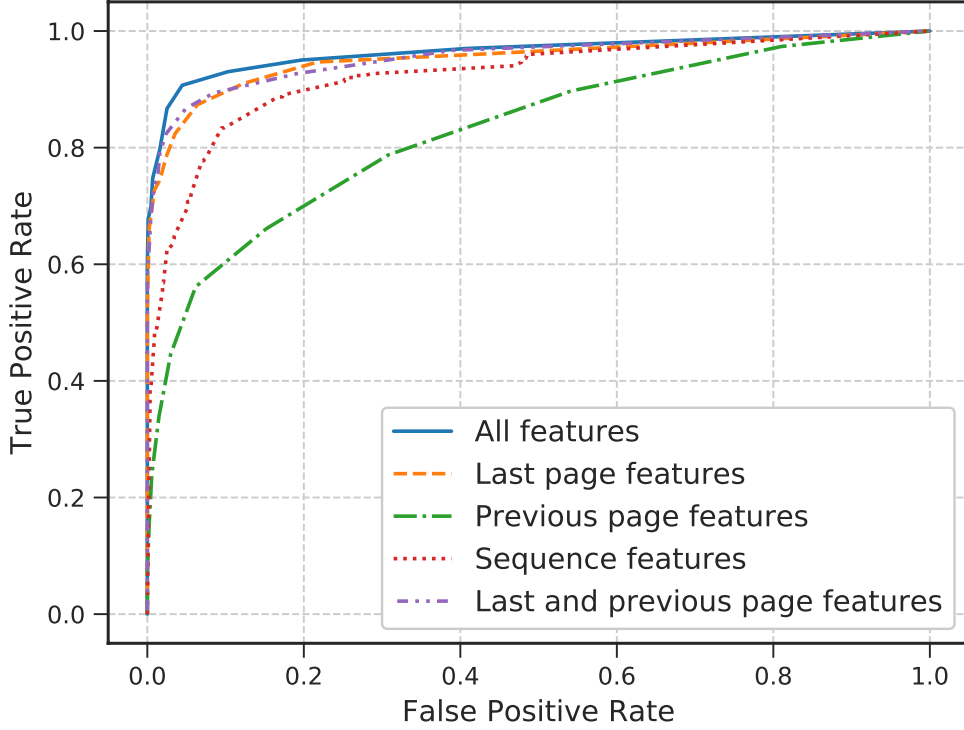


Figure 3.7: ROC curves of SE detection results for each feature set.

Some web pages had similar appearances to known SE pages but were not blacklisted. To find such potentially unknown SE pages, we leveraged the SE-detection module to classify the remaining 192,620 sequences of the 11,304 domain names not used in the evaluation. As a result of manually excluding false positives (27 domains) from the classification results, we found 359 unknown domain names associated with SE attacks. We not only detected web pages where page contents were shared across multiple domain names to expand attack campaigns (e.g., Fig 3.8 and Fig 3.9), but also discovered unreported domain names associated with technical support scams and survey scams. This process was conducted by analyzing screenshots to check whether suggested software and extensions or login pages are associated with legitimate services. One example of the false positives was a Facebook login page opened by a popup that redirected from an illegal software-download blog by clicking a share button. Another example was a download page of legitimate anti-virus products that transferred by clicking advertising in an iframe. We finally found a total of 1,404 unique domain names (the 1,045 blacklisted domain names and newly detected 359 domain names), and 56,922 sequences reached the 1,404 domain names. The number of sequences’ steps (i.e., the number of page transitions) from one to three is 11,855 (20.8%), 13,813 (24.3%), and 31,254 (54.9%), respectively.

Table 3.9: SE attack categories

Category	SE domain names
PUP	566 (40.2%)
Malware	310 (22.1%)
Unwanted browser extension	181 (12.9%)
Multimedia scam	94 (6.7%)
Phishing	70 (5.0%)
Survey scam	25 (1.8%)
Tech support scam	20 (1.4%)
Fake browser history injection	16 (1.1%)
Malvertisement redirection	13 (0.9%)
Cryptojacking	3 (0.2%)
Other SE attacks	109 (7.8%)
Total	1,404 (100%)

3.5 Detailed Analysis of Detected Multi-step SE Attacks

We conducted a detailed analysis of the collected multi-step SE attacks mentioned in Section 3.4.5 (1,404 domain names and 56,992 sequences). To show that STRAYSHEEP found a wide variety of SE attacks, we categorized the observed SE page’s domain names and investigated the attacker techniques to deceive and persuade users for each SE attack category. We then analyzed the browser interactions and advertising providers that led to SE pages to clarify the cause of SE attacks. Finally, we investigated network infrastructures hosting SE attacks.

3.5.1 SE Attack Categories

To clarify the types of multi-step SE attacks detected by STRAYSHEEP, we categorized the 1,404 domain names into 11 categories, as shown in Table 3.9. We used labels of blacklists (Google Safe Browsing, Symantec DeepSight, hpHosts) and virus scan results of VirusTotal to categorize the attacks. We leveraged AV-Class [50] to classify detected binaries as PUPs or malware. We also checked the appearance of these domain names’ web pages to complement categorization.

PUP and Malware The most common categories we identified were PUP (566 domain names) and malware (310 domain names). These categories are SE attacks where PUPs and malware were downloaded due to browser interactions. STRAYSHEEP downloaded 6,924 unique binary executable files (e.g., `.exe` or `.dmg`). For example, we found that these binaries were disguised as fake game installers, fake anti-virus software, and fake Java/Flash updaters. Out of the 6,924 binaries, we detected 1,591 unique binaries including 1,090 malware samples and 501 PUPs by checking their MD5 hash in VirusTotal and using AVClass. We confirmed that 3,336 unique binaries were never uploaded to VirusTotal. Although the remaining 1,997 unique binaries were already uploaded, they were not detected by any anti-virus software in VirusTotal.

The 2,141 out of the 3,336 binaries that were not uploaded had 1,347 unique

filenames, which were automatically set according to the previous page (e.g., “[*the title of the previous page*].exe.rename”). Figure 3.8 shows examples of these web pages. The web pages that downloaded these binaries contained instructions to entice users to remove “.rename” and execute them. We found 504 unique domain names downloading these binaries. The 175 out of these 504 domain names matched the blacklists and the other 329 domain names were newly detected by STRAYSHEEP. The reason for making users change the file extension is to circumvent the download-protection function of web browsers. Since the hash values of these binaries also changed at every downloading, none were ever uploaded to VirusTotal. To check whether these binaries were malicious, we chose ten samples from the binaries and uploaded them to VirusTotal. Then, all ten samples were detected as “StartSurf” or “Prepscream” family names.

Unwanted Browser Extension We categorized 181 domain names as distributing unwanted browser extensions. We confirmed that these domain names were detected as “Fake Browser Extension Download” or “Unwanted Extension”, which led to install pages (<https://chrome.google.com/webstore>) of 128 unique Google Chrome browser extensions. However, we found that 119 (93.0%) extensions were still available on the browser extension install pages a month after the crawling. By investigating these browser extensions, we found that 18 (14.1%) extensions were search tool bars, and 14 (10.9%) extensions were file converters. Security vendor blog postings and online forums stated that some extensions were malicious extensions or browser hijackers that modify web browser settings, track user’s browsing, and inject unwanted advertisements [51]. As a result of our dynamic analysis of some browser extensions using a real browser, we observed suspicious behavior such as displaying popup advertisements and changing the default browser’s homepage and search engine to web pages hard coded in many malware samples. To determine the popularity of these browser extensions, we searched each extension name on a search engine. We then found that the search results of 100 extensions (78.1%) consisted of one or more web pages explaining “How to remove [*browser extension name*]” or “Virus removal guide”. Surprisingly, most of these web pages introduced not only removal methods but also suggested yet more fake removal tools, which were detected as PUPs or malware. Attackers prepared the web pages for tricking technically unsophisticated users who disrupt these browser extensions. Thus, even if the users successfully remove the unwanted browser extensions, they also become victims of other SE attacks. STRAYSHEEP’s SE-detection module newly found 21 domain names out of the 181 domain names we categorized. STRAYSHEEP successfully finds unwanted browser extensions by analyzing distribution web pages and sequences that led to them instead of analyzing their source codes and behaviors.

Multimedia Scam We found web pages (94 domain names) that ask for credit card registration in exchange for offering free access to movies or music. Their content, such as input forms, logos, and background images, were shared among each other. We call them *multimedia scams* in this study. Only 27.7% (26/94) of domain names were listed in blacklists; however, STRAYSHEEP’s SE-detection module newly found 72.3% (68/94) domain names. Some security vendor blog postings and online forums reported that these web pages fraudulently charge

credit cards [52]. We found that some words (e.g., `media`, `play`, and `book`) were frequently used in the domain names, such as `etnamedia.net`, `kelpmedia.com`, `dewymedia.com`, `parryplay.com`, `cnidaplay.com`, and `mossyplay.com`.

Figure 3.9 shows examples of multimedia scams. These web pages suggest users to register for free membership to obtain movies, music, or games. When users are tricked to input their credit card numbers, the web pages fraudulently charge them.

Phishing We observed 94 domain names detected as *phishing*, which were attempting to steal user’s sensitive information such as email addresses or passwords.

Survey Scam We found 25 *survey scam* domain names, which spoofed famous companies and promised rewards such as iPhones and gift cards. Although Surveyance [12] only identified landing pages that have survey content and interacted with them to reach survey scams, STRAYSHEEP recursively followed lure elements to detect survey scams reached from landing pages that did not have survey content.

Tech Support Scam We observed 20 *tech support scam* domain names that displayed fake virus-infection messages and telephone numbers of support centers to urge users to call. STRAYSHEEP reached the scams from sequences of web pages starting from the search engine’s results and social media postings, which are not observed with other systems [25, 14].

Fake Browser History Injection We found 16 *Fake browser history injection attacks* domain names, which injected URLs into the browser’s history to force users to redirect to another SE page when the browser’s back button is clicked. To interact with such attacks, STRAYSHEEP attempted clicking the back button for each web page and determined that the action led to other SE pages.

Malvertisement Redirection We found 13 *Malvertisement website redirect* domains [53, 54] that also led users to other SE pages.

Cryptojacking We found three *Cryptojacking* domain names that secretly used user’s CPU resources to mine cryptocurrencies by injecting JavaScript codes.

Other SE Attacks We observed various SE attacks other than those mentioned above, such as one just indicates the “Social engineering” label.

3.5.2 Common Infrastructures of Multi-step SE Attacks

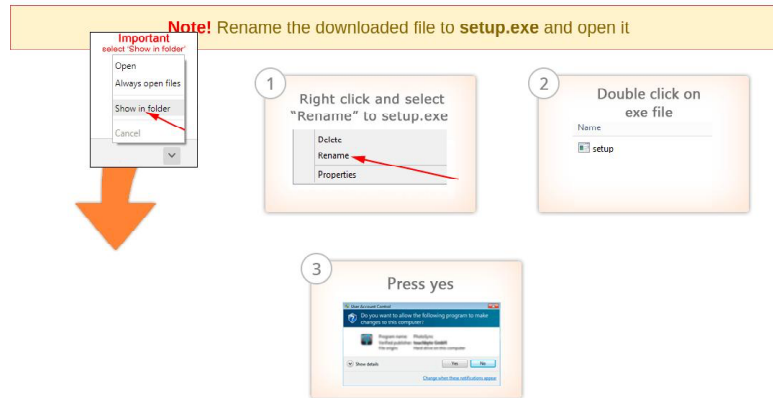
To clarify the common infrastructures of multi-step SE attacks and attacker’s techniques leading to the attacks, we analyzed the 56,922 sequences (see Section 3.5) that led to SE pages.

SE Attacks Caused by Unintended Clicks We observed opening popup/pop-under windows caused by unintended clicks such as clicking anywhere on a web page and on the browser’s back button. Such popups are often set by JavaScript codes provided by advertising providers to the web page’s owner. The following three files are the most frequently loaded on web pages leading users to SE pages: “`c1.popads[.]net/pop.js`”, “`cdn.popcash[.]net/pop.js`”, and “`cdn.cpmstar[.]com/cached/jspopunder_v101.pack.js`”. Since such advertisements are common infrastructures for SE attack distribution, they are used in various web pages. The sequences in which popups caused by unintended clicks occurred were

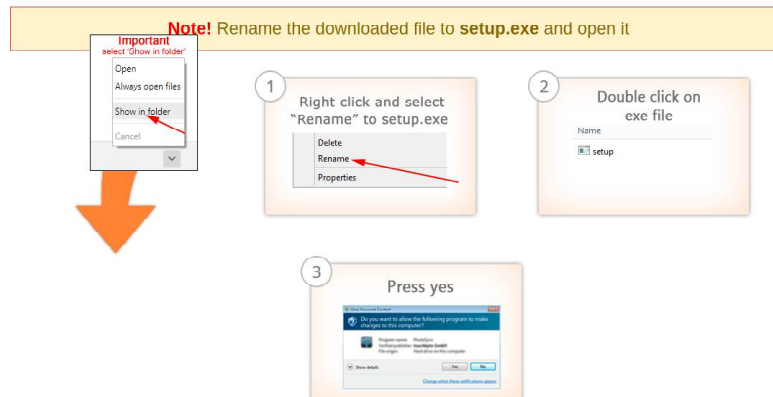
20.0% (11,373/56,922) of all sequences. The sequences in which popups caused by unintended clicks occurred in the landing pages were 8.7% (4,952/56,922) of all sequences. We also observed exit-driven redirections that were triggered by clicking on the browser’s back button, which was 4.5% (2,578/56,922) of all sequences.

Alert Dialog Of all sequences, 2.9% (1,651/56,922) included a web page that displayed more than one alert dialog. We found 66 distinct alert messages, such as those of fake virus infection and fake rewards, which might strongly influence user psychology. To investigate the relationship between the content of alert messages and SE attacks, we categorized the 66 alert messages into the three attack classes of *comply*, *alarm*, and *entice*. These classes were defined in a previous study [24]. We found 30 *Comply* alerts that were often used on fake Java/Flash update web pages for luring users to install PUPs and malware, such as “Please install Java to continue.” and “Your Flash Player might be out of date. Please install update to continue.” We found 19 *Entice* alerts that made users input sensitive information, such as “CONGRATULATIONS! Your IP address has been selected to receive a Year of FREE Netflix!” We found 17 *Alarm* alerts that showed warning messages such as “IMMEDIATE ACTION REQUIRED We have detected a trojan virus” with alert sounds in some cases (e.g., `<audio src="alert.mp3" autoplay>`). Users were directed to install fake anti-virus software or call fake technical support centers.

H1Z1-keygen download is ready!



Warcraft 2 Beyond the Dark Portal Download PC download is ready!



PLAYSTATION 2 ALL BIOS FILES COLLECTION POST download is ready!

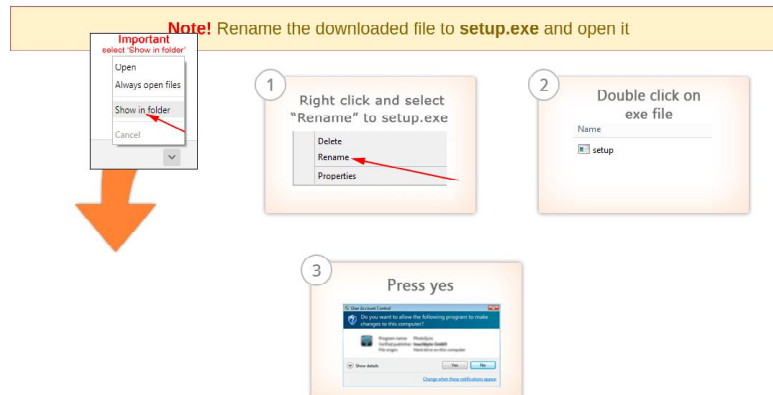


Figure 3.8: Examples malware-distribution pages that require user to rename files and execute them.

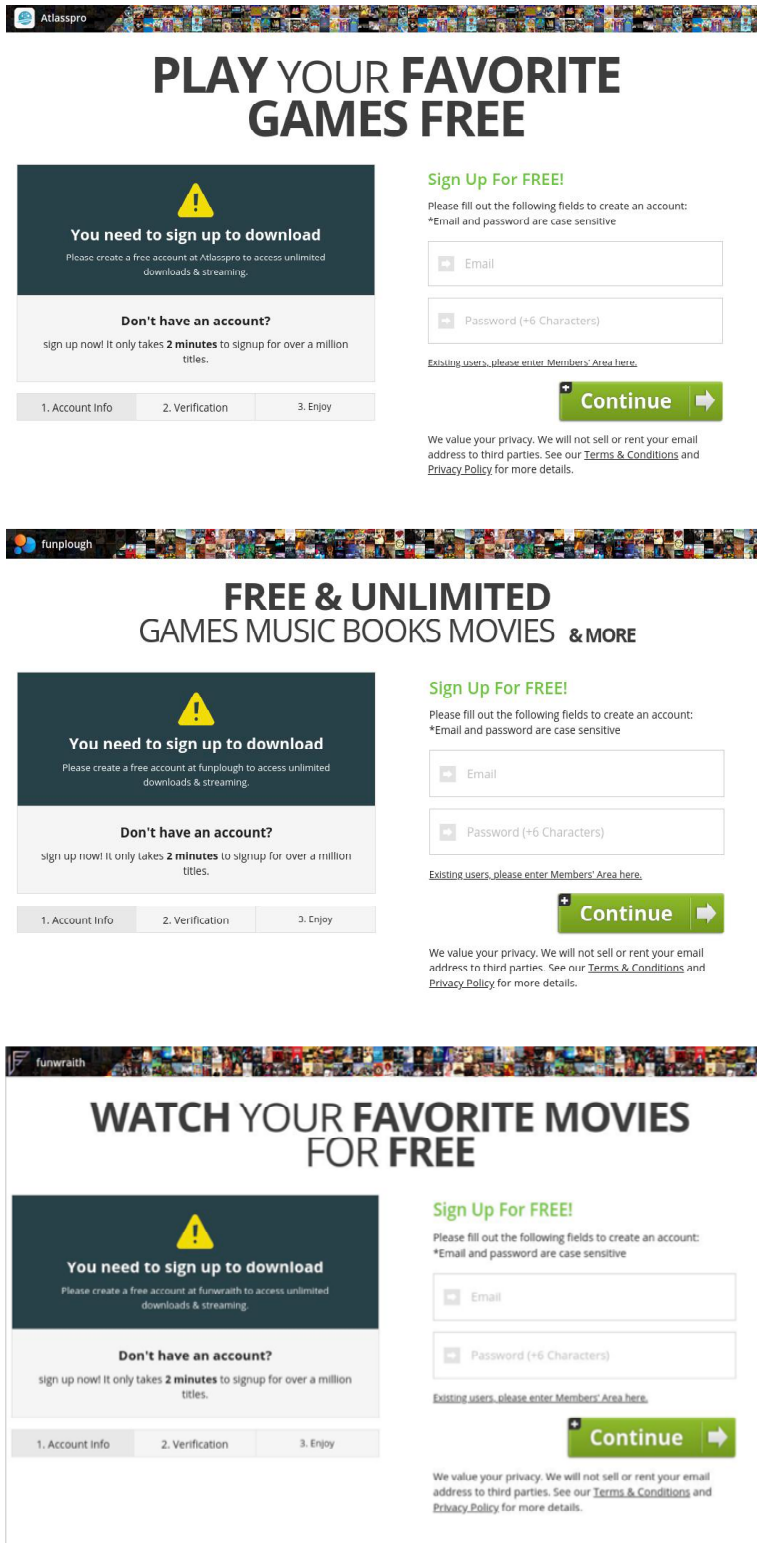


Figure 3.9: Examples of multimedia scams.

Table 3.10: Newly observed SE pages’ domain names at each depth in ascending order and their user access investigated by SimilarWeb, Alexa Web Information Service, and DNSDB.

Depth # of domains	SimilarWeb (total visits per month)					Alexa Web Information Service (pageviews per million users)					DNSDB (DNS queries)				
	# of domain names with valid data	Min	Max	Sum	Mean	# of domain names with valid data	Min	Max	Sum	Mean	# of domain names with valid data	Min	Max	Sum	Mean
1	94	39 16,616	110,300,000	184,891,612	4,740,811		68	0.01	41.70	143.49	2.11	83	1 30,896,251	58,835,762	708,865
2	680	261 9,926	205,800,000	1,483,161,357	5,682,611		344	0.01	631.50	1684.62	4.90	425	2 19,727,061	52,593,168	123,749
3	288	108 16,326	92,460,000	301,856,918	2,794,971		144	0.01	264.00	476.95	3.31	181	1 61,561,400	302,969,603	1,673,865
4	342	28 15,191	115,400,000	158,933,466	5,676,195		51	0.01	8.62	34.32	0.67	93	2 11,357,814	63,461,471	682,381
Total	1,404	436 9,926	205,800,000	2,128,843,353	4,882,668		607	0.01	631.50	2339.38	3.85	782	1 61,561,400	477,860,004	611,074

Table 3.11: Advertising provider domain names redirected to SE domain names.

Advertising provider	# of unique SE domains redirected from ad provider
newstarads.com	155
traktrafficflow.com	134
mybestmv.com	123
revimedia.com	122
naganoadigei.com	99
doubleclick.net	90
googleadservices.com	80
adk2x.com	75
clksite.com	52
cobalten.com	41
bodelen.com	41
googlesyndication.com	36
cpmstar.com	29
go2affise.com	28
inclk.com	23
digitaldsp.com	21
dtiserv2.com	19
tradeadexchange.com	19
adf.ly	19
adreactor.com	19
friendlyduck.com	16
revcontent.com	16
servedbytrackingdesk.com	16
reimageplus.com	14
adnetworkperformance.com	14
All advertising provider	427

Advertising Domain Names Online advertising often results in SE attacks [24, 12]. To analyze SE attacks delivered by advertising providers, we extracted advertising providers’ domain names (ad domain) from server-side redirection on the sequences. We leveraged public advertising provider lists [55] to identify ad domains. Table 3.11 shows a list of ad domains and the number of unique domain names of SE pages redirected from each ad domain. We found 25 ad domains that led to SE attacks. Categories of SE attacks frequently distributed by ad domains were multimedia scam, unwanted browser extension, fake anti-virus software (PUP/malware category), and fake Java update (PUP/malware category). The ad domain that redirected to the most SE page’s domain names was `newstarads.com`, which led to 155 unique domain names. Two domain names (`doubleclick.net` and `googleadservices.com`) redirected to 89 and 79 unique domain names of unwanted browser extension and they also redirected to the same phishing domain names. We found that 30.4% (427/1,404) of the total SE domain names were reached from these advertising domain names.

Prevalence of SE attacks We analyzed the statistics of user accesses to measure how many users encountered multi-step SE attacks. We used SimilarWeb¹, Alexa

¹<https://www.similarweb.com/>

Table 3.12: Top 10 countries mapped

Country	Percentage
UnitedStates	72.9%
Thailand	2.3%
Mexico	1.9%
Vietnam	1.9%
Canada	1.8%
Brazil	1.6%
Korea	1.4%
Indonesia	1.0%
Philippines	0.9%
Taiwan	0.8%

Table 3.13: Top 10 ASes hosting SE attacks

AS	Percentage
Amazon.com, Inc.	50.5%
Google LLC	7.6%
Cloudflare Inc	2.9%
Uninet S.A. de C.V.	1.0%
VNPT Corp	1.0%
Level 3 Parent, LLC	0.6%
JasTel Network International Gateway	0.6%
Telefonica Brasil	0.6%
TOT Public Company Limited	0.6%
FPT	0.4%

Web Information Service (AWIS)², and DNSDB³ to investigate website traffic volumes of 1,404 domain names that STRAYSHEEP collected, as mentioned in Section 3.4. SimilarWeb and AWIS provide website traffic statistics of domain names. DNSDB is a passive DNS database that provides the total number of DNS queries of domain names. Table 3.10 lists the numbers of unique domain names newly observed at each depth in ascending order and statistics (minimum, maximum, sum, mean) of website traffic and DNS queries. Note that *# domain names with valid data* means the number of domain names excluding the data that are zero or not available in the data sources. Since most SE pages' domain names were observed at depth two, there were still 44.9% (630) of domain names observed at deeper depths. In other words, there are many domain names at deeper depths that can only be reached by following multiple web pages with STRAYSHEEP. The statistics of user accesses and DNS queries show that these websites have the same level of population with domain names observed at shallow depths, some of which are covered by previous systems. For example, the mean of SimilarWeb's total visits at depth four (5,676,195) is almost the same as that at depth two (5,682,611) and is larger than that at depth one (4,740,811). Also, the sum of AWIS's pageviews per million at depths three and four is 511.27, which is

²<https://awis.alexa.com/>

³<https://www.dnsdb.info/>

21.9% of the total. In the data of DNSDB, the number of valid domain names at depth three (181) is less than depth two (425); however, the sum of DNS queries (302,969,603) is larger than that at depth three (52,593,168). Therefore, we showed that there are many malicious domain names that StraySheep reaches by following multiple web pages from landing pages. Also, these domain names, which previous systems cannot reach, have a large number of user accesses. One reason for this is that these domain names are distributed by large-scale advertising providers, as shown in Table 3.11.

IP Addresses Used for SE Attacks To analyze the relationship between each SE page’s domain name, we leveraged DNSDB. The DNSDB enables us to find IP addresses historically associated with domain names. If the same IP address is set in the A record of different domain names, we assume that these domain names are related. As a result of investigating the 1,404 domain names, we detected a total of 96,544 IP addresses associated with 1,349 domain names (55 domain names were not found in the DNSDB). Note that multiple IP addresses were associated with one domain name; thus, there are more IP addresses than domain names. We found that 29.6% (28,617/96,544) of IP addresses were shared among more than two domain names we detected, and these IP addresses (28,617) were associated with 39.5% (554/1,404) of domain names. The 554 domain names were mainly used for multimedia scams, PUP/malware distributions, survey scams, and unwanted browser extension installs. We now focus on 94 multimedia scam domains and their corresponding 20,589 IP addresses. We found that 87.8% (18,086/20,589) IP addresses were shared among more than two multimedia scam domains. One of these IP addresses was shared with 84 multimedia scam domains we detected.

Geographical Attribution We analyzed the geographical attribution of IP addresses used for SE attacks. We used the same 96,544 IP addresses as the above analysis. We queried GeoIP2 Databases⁴ for the country and Autonomous system (AS) information associated with the IP addresses. Table 3.12 shows the top 10 countries whose IP addresses were used for distributing SE attacks. United States accounted for 72.9% of all IP addresses, Thailand for 2.3%, Mexico for 1.9%, and Vietnam for 1.9%. Table 3.13 shows the top 10 ASes. We confirmed CDNs and cloud hosting providers are frequently abused for SE attacks. Amazon, Google, and Cloudflare accounted for 50.5, 7.6, and 2.9%.

3.6 Discussion

In this section, we discuss the limitations of STRAYSHEEP and ethical considerations during our study.

3.6.1 Limitations

There are limitations with STRAYSHEEP in terms of system environment, system implementation, and evasion of our system.

⁴<https://www.maxmind.com/en/geoip2-databases>

System Environment In the evaluation discussed in this chapter, STRAYSHEEP was run in a single environment. Some SE pages may not serve the same web page every time due to an ad network or cloaking technology. Specifically, a website changes the web page to be delivered according to the source IP address, web-browser environment, and browsing history. In this case, there are SE attacks that cannot be reached in the current STRAYSHEEP environment. However, as described in Section 3.3.2, STRAYSHEEP does not depend on the selected browser environment and connection network. Thus, preparing multiple browser environments and connection networks enables us to collect environment-dependent SE attacks.

System Implementation STRAYSHEEP implements web-search-based URL collection methods; thus, attacks originating from other types of sources (e.g., email) are out of its scope. Since SE attacks attempt to lure more users to their web pages, attackers should prepare landing pages that can be easily visited from popular web platforms, i.e., search engines and social media. STRAYSHEEP covered these platforms and retrieve landing pages using easily customizable search queries. Since interacting HTML forms are not implemented in STRAYSHEEP’s current web-crawling module, it cannot crawl web pages that require login, account creation, and survey. However, STRAYSHEEP’s SE-detection module can identify these web pages because it uses not only features of the reached web page but also features extracted from the entire sequence.

Evasion There may be an evasion technique against STRAYSHEEP’s web-crawling module to create a web page that redirects users to SE attacks without preparing any lure elements. This technique leads to a lowering of the collection efficiency of SE attacks of STRAYSHEEP. There may be another evasion technique that introduces CAPTCHA authentication in the middle of an SE attack. An SE attack with CAPTCHA authentication cannot be collected with the current implementation of STRAYSHEEP. However, these evasion techniques greatly reduce the number of potential victims, which leads to a reduction in the success rate of attacks. Therefore, we believe that it is unlikely that an attacker actually carries them out, as it goes against the current trend of SE attacks.

There may also be an evasion technique against STRAYSHEEP’s SE-detection module designed as a classification approach. Attackers modify an SE page’s appearance to evade the future design and structure of web pages. However, this module also extracts features from the entire sequence of web pages, such as the occurrence of popup windows displaying fake infection alerts and redirections caused by a user’s unintended clicks. Thus, we believe it is still difficult for attackers to evade because these features represent attackers’ effective techniques to lure users to their web pages.

3.6.2 Ethical Consideration

Our study followed research ethics principles and best practices [56, 53, 54, 14]. While we conducted parallel crawling for various websites, each of our crawling sessions sequentially traversed web content on the same website, so only a restricted amount of traffic to the website was generated, which did not increase website

workload. Our crawling carefully created web requests according to the manner of a real web browser and did not create any harmful web requests breaking or exploiting websites. Due to using a real web browser, our crawling faithfully performs according to the natural behavior of web browsers. Furthermore, the intention of our automated crawling is not to thwart the monetization model of benign web ads. There is no alternative and realistic way to directly observe SE attacks except for active crawling; however, there is a risk of unexpectedly contributing to malicious pay-per-click (PPC) or pay-per-install (PPI) monetization. Our crawling did not intentionally concentrate on specific PPC or PPI services.

3.7 Related work

Web-based SE attacks and their defenses have been gaining the attention of researchers. We review related work in terms of collecting these attacks and analyzing the attack mechanisms. Duman et al. focused on the visual properties of trick banners, which lure users into clicking on fake links [36]. They built a Firefox browser extension called TrueClick to detect such trick banners based on image processing and machine learning. STRAYSHEEP finds lure elements including trick banners, and interacts with them to confirm whether they actually lead users to SE attacks. Rafique et al. analyzed free live streaming services and their ecosystems [13]. They found that users of these services are exposed to ads, malware, and unwanted browser extensions. Our analysis found that not only live streaming services but also web pages showing illegal content, such as music and games, use lure elements to lead users to malware and unwanted browser extensions. Nelms et al. studied the sequences of visited web pages preceding malware downloads in drive-by download and SE attacks [23]. They proposed a system called WebWitness to passively trace back the visited web pages to analyze how users reach the attacks. They also presented a systematic study on successful SE attacks leading to malicious and unwanted software [24]. They categorized and identified the tactics used in such SE attacks to gain users' attention. While these studies [23, 24] passively traced back real victim's traffic, STRAYSHEEP actively collects SE attacks and does not rely on real victims. Vadrevu et al. developed a specific web-browser system called ChromePic to enable the reconstruction of SE attacks [30]. ChromePic introduces a detailed snapshot of logging into Chromium to enable the investigation of SE attacks. Whereas ChromePic focuses on forensics after users reached SE attacks, proactive and large-scale crawling of the latest SE attacks. Miramirkhani et al. conducted the first systematic analysis of technical-support-scam web pages [25]. Specifically, they developed a system that can identify such web pages and collect them to show their prevalence, the abused infrastructure, and illicit profits. Srinivasan et al. analyzed technical support scams by focusing on search-engine results and corresponding sponsored advertisements [14]. They generated technical-support-related special search-engine queries to discover previously unknown technical support scams. STRAYSHEEP also finds identified technical support scams based on search engine results, as well as scams that require multiple interactions to reach. Kharraz et al. proposed a system called Surveylance [12] to identify survey scams using search engines and

a web-crawling approach. While this system identifies only landing pages having survey content (e.g., advertisement in iframe), STRAYSHEEP also identifies survey scams by clicking lure elements, which do not display survey content.

3.8 Conclusion

We proposed a system called STRAYSHEEP to crawl web pages and detect multi-step SE attacks. Our key idea is based on (1) simulating multi-step browsing behavior of users to efficiently crawl web pages leading to SE attacks and (2) extracting features from reached web pages as well as the entire sequence of web pages to accurately detect such attacks. Our experimental results indicate that STRAYSHEEP can lead to 20% more SE attacks than Alexa top sites and search results of trend words, crawl five times more efficiently than a simple crawling module, and detect SE attacks with 95.5% accuracy. STRAYSHEEP will be useful for security vendors, search engine providers, and social-media companies in terms of analyzing trends in SE attacks.

Chapter 4

Understanding the Fake Removal Information Advertisement Sites

4.1 Introduction

Antivirus (AV) software is an essential tool for endpoint protection. The major AV software market was valued at 3,770 million USD in 2018 [57], and attackers focus on the needs of such pervasive AV software to gain financial benefits. Specifically, *fake AV software*, which are rogue applications disguised as legitimate AV software, is used to manipulate users' devices and steal money or sensitive information [58, 22]. For example, once fake AV software is installed, the software displays fake virus scan results to get users to purchase additional licenses [59, 60].

Fake AV software is a traditional cyber threat that can effectively spread malware and unwanted software on the web [61, 62]. To infect users and gain more profit, attackers take advantage of online advertisements that target many people to distribute fake AV software [63]. The web pages served by these advertisements typically show fake virus infection alerts or messages claiming the necessity of installing their software. These web pages also attract users with promises of speeding up their machines [64]. Attackers use such social engineering techniques that exploit users' psychological vulnerabilities to lure users to download fake AV software. These web pages are known to be major distribution paths for fake AV software [65, 66, 67].

In this chapter, we focus on new techniques that psychologically encourage users to install fake AV software from the web. Attackers create web pages that introduce fake information for handling specific cyber threats, such as malware infection or visits to malicious web pages, and suggest fake AV software. We call these web pages *fake removal information advertisement (FRAD) sites*, which target users who have already suffered from security problems and which make them victims of another one. For example, users who notice their malware infection try to search for removal information using the malware detection names given by virus scanners, and they reach the FRAD sites from search results. Believing the FRAD information, the users follow the instructions and inadvertently install the suggested fake AV software. Although it is well known that attackers induce

users to install fake AV software using scaring or attracting messages—such as fake infection alerts or promises to speed up their machines—little attention has been given to analyzing the FRAD sites.

Here, we propose a system that automatically crawls the web pages and detects FRAD sites. Using the linguistic and visual features of the web pages, we accurately identify FRAD sites with 98.8% true positives and only 3.3% false positives. We used our system for a large-scale collection of FRAD sites and found 2,913 distinct domain names of FRAD sites written in 31 languages. The total user accesses to these FRAD sites was 73.5 million visits per month. We observed that these FRAD sites are not adequately reported by existing blacklists.

To reveal the ecosystem of FRAD sites, we performed a measurement study using both passively collected statistical data on user accesses and actively crawled data. We first investigated the incoming traffic to FRAD sites to determine what types of user behaviors are at risk of reaching FRAD sites. We found that many users not only accessed these sites from search engines directly but also reached FRAD sites from videos or messages posted on social media by attackers’ accounts. To determine what kinds of attacks users encounter from FRAD sites, we then analyzed the transferred web pages and downloaded files from the FRAD sites. We confirmed that the FRAD sites led to 76 fake AV software families by directly distributing installers and luring users to payment and distribution sites. Also, we investigated search results for the names of specific cyber threats, and we found that 82.6% of the top 10 search results were occupied by FRAD sites. In other words, search results for information concerning cyber threats are poisoned by FRAD sites, making it difficult for users to obtain correct removal information. To the best of our knowledge, this is the first study that has revealed the prevalence and ecosystem of FRAD sites.

In summary, our contributions are as follows:

- We propose a system to crawl the web and detect FRAD sites automatically. By extracting linguistic and visual features from crawled web pages, our system detected FRAD sites with 98.8% true positives and 3.3% false positives.
- We performed a large-scale collection of FRAD sites on the web by leveraging a search engine, which is the most common channel used to reach FRAD sites. Using our system, we discovered 2,913 domain names of FRAD sites written in 31 languages. We found that attackers widely deploy FRAD sites targeting users in various countries to increase the number of page views.
- We conducted a comprehensive measurement study using both passively collected statistics data and actively crawled data to reveal the ecosystem of FRAD sites. Our measurement study also clarified the typical incoming channels employed by users to reach FRAD sites and the types of potential threats directed from the FRAD sites. We also found that it is difficult for users who need removal information for specific cyber threats to reach correct information, because most of the search results concerning cyber threats are poisoned by the FRAD sites.

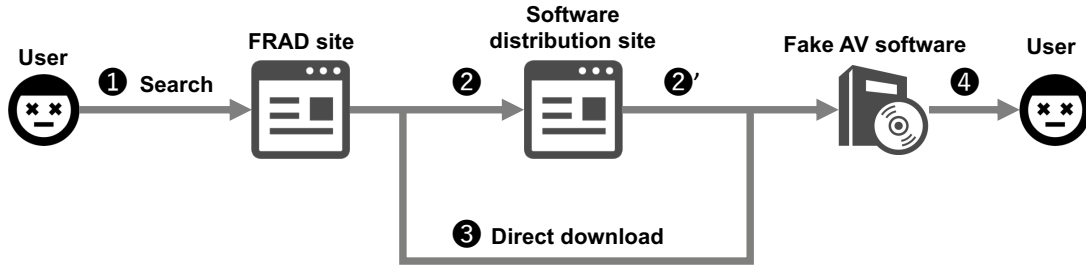


Figure 4.1: Overview of fake AV software distribution via FRAD sites.



Figure 4.2: Common page structure of FRAD sites.

4.2 Background

We first consider an attack technique for distributing fake AV software via FRAD sites. The purpose of the FRAD sites is to deceive users who need ways to deal with cyber threats, i.e., malicious acts that damage the users' devices and steal their sensitive information. Examples of cyber threats include malware infection, fraudulent popup messages, and malicious browser extensions. Attackers post multiple entries on FRAD sites that introduce fake threat removal guides, using the names of specific cyber threats, such as malware detection names or the domain names of malicious sites. For instance, there can be more than 15k entries in a single FRAD site, and dozens of new entries are added to the FRAD site every day. Figure 4.2 shows a common structure of the entries of FRAD sites. FRAD sites often include a phrase related to malware removal in domain names, URL paths, titles, and textual content. They also show the package images of fake AV software and logo images of security vendors and software certification companies. More detailed characteristics of FRAD sites are discussed in Section 4.3.2. When users

notice that they have security issues by looking at the results from legitimate virus scanners or from suspicious alert messages on web pages, they search for information to remove them. Users who reach FRAD sites and are deceived by false information install fake AV software, which makes matters worse. We focus on such scams on the web in this chapter.

Figure 4.1 shows an overview of the distribution of fake AV software via FRAD sites. First, users who have security problems reach FRAD sites by searching for the specific names of cyber threats they want to remove (①). Attackers leverage search engine optimization (SEO) techniques that target specific names of cyber threats to increase the web traffic to FRAD sites. Attackers also post fake videos on YouTube that introduce ways to remove the threats, and they post similar articles on Facebook and other social media to lure users to click on links to FRAD sites. Forum and community sites where anyone can post messages are also used by the attackers in the same manner. Thus, users not only visit FRAD sites from results provided by search engines but also reach FRAD sites through social-media postings and other web pages hit by the search results. The FRAD sites contain detailed fake removal guides for individual threats as well as large buttons or banners to direct users to fake AV software. The FRAD sites usually display the logos of famous security vendors or third-party organizations (e.g., software certification companies) to make them look as if they are legitimate web pages. Users who click on the buttons or banners are navigated to software distribution sites (②). Most of the software distribution sites use domain names containing the names of the fake AV software and disguise themselves as official sites for legitimate AV software by displaying product information and purchase menus. These sites are also reachable through search engines and even provide customer support such as web chats or toll-free calls. On these web pages, users follow the payment and download instructions and then obtain fake AV software installers (②'). These installers can also be downloaded from the FRAD sites directly (③). Users install the fake AV software and thus become victims of other cyber threats (④).

Some social engineering techniques are already known, such as threatening users using fake infection alerts or attracting them by the prospect of improving computer performance. However, it has not been clarified whether attackers use techniques for distributing fake AV software that exploit the weaknesses of users who have already suffered from cyber threats.

4.3 Method

In this section, we introduce our system for collecting and detecting FRAD sites on the Internet automatically. The system consists of two steps: web crawling and classification.

Table 4.1: List of terms for each category; used to check the term’s frequency in the title, URL paths, domain names, and text content of a web page.

Category	Example terms
way	“how to”, “guide”, “solution”, “tips”, “report”, “instruction”
removal	“remove”, “get rid of”, “uninstall”, “delete”, “fix”, “clean”, “kill”, “block”, “repair”, “anti”, “entfernen”, “eliminar”, “verwijderen”, “deinstallieren”, “desinstalar”, “supprimer”, “remuovere”, “usunac”
problem	“virus”, “malware”, “spyware”, “trojan”, “backdoor”, “adware”, “threat”, “infection”, “ransom”, “error”, “pop up”, “redirect”
device	“computer”, “pc”, “windows”, “mac”, “browser”

4.3.1 Web Crawling

The implementation of a web crawler that collects and stores browser-level information from web pages is the first step in our system. The requirement of the crawler is to extract linguistic and image features from a web page rendered by a web browser and to compose a feature vector for the result. To analyze the FRAD sites in detail, we also need to capture the network traffic to and perform browser interactions on the web page. To achieve this, we designed and implemented the crawler using Scrapy¹, which is a web crawling framework for Python, in order to develop functions for monitoring and managing logged data. We used Selenium² as the middleware for Scrapy to automate a real web browser. We used Google Chrome as the default web browser for the crawler. To monitor network traffic in detail, we used Chrome DevTools API³. This is necessary, because we collect network-level information such as HTTP requests and responses that Selenium API does not handle directly. The collected information such as screenshots, HTML source codes, and network traffic are stored to MongoDB. We use those kinds of information for the next step, classification.

4.3.2 Classification

In the second step, our system extracts features from the information collected from the web pages and identifies FRAD sites using a supervised machine learning approach. In particular, the system analyzes term frequencies in web pages and URLs, the presence of logo images on screenshots, and HTML structures, such as the number of tags, and combines them into a feature vector. We explain the detail of each feature below.

Term Frequencies

To capture the linguistic characteristics of FRAD sites, frequencies of terms are used as a feature. To improve the SEO ranking and ensure an easy web page topic for users to understand, FRAD sites use terms meaning for the removal of cyber threats in the titles, URL paths, domain names, and text content of their web pages. Examples of such titles are “*Remove Trojan.ZeroCleare (Virus Removal Guide)*” and “*Remove Magiballs.com (Free Guide)*.” The URL paths include forms such as “/2019/12/27/how-to-remove-my-login-hub-virus-removal-guide/” and “/uninstall-nvux-xyz-from-windows-7-8-8-1-10.” Examples of domain names are `uninstallmalwarefrompc[.]example` and `virusremovalguide[.]example`. The text content of the web page is written with a summary of the cyber threat and specific removal information for it.

Our key insight is that the FRAD sites must include a phrase composed of the following four categories of terms: *way*, *removal*, *problem*, and *device*. Table 4.1 shows a list of example terms. As the feature vector, we use the number of occurrences of each term category in the following four fields: the title, URL path, domain name, and text content. The terms in the four categories are intended to capture phrases such as “*how to remove Trojan.ZeroCleare virus from my PC*.” Because the FRAD sites are created in many languages, we leverage machine translation services such as Cloud Translation API⁴ and Amazon Translate⁵. We translate the title and text content of the crawled web pages into English and then calculate the frequencies of the terms.

To create the list of terms, we extracted all terms that match each category from the title, URL paths, domain names, and text content of 300 FRAD sites that were randomly selected from our created dataset, as discussed below in Section 4.4. Some domain names include non-English terms in the *removal* category, such as “entfernen” in German and “eliminar” in Spanish. Because these domain names are difficult to translate, we manually obtained such terms as much as possible. To this end, we separated the domain names by “.” or “-” and used word segmentation⁶ and then searched for the meaning of each extracted word.

Logo Images

We next consider features that specify logo images on the FRAD sites. The FRAD sites include download buttons and software packages that may be shared among multiple FRAD sites. The FRAD sites also display logos of security vendors, operating system (OS) vendors or software certification companies in order to pretend to be legitimate sites. These logos are copied from vendors’ sites or used as image files modified from the original images. To find such visual characteristics, our system uses an image matching approach on the basis of our logo image database.

¹<https://scrapy.org/>

²<https://selenium.dev/>

³<https://developer.chrome.com/extensions/devtools>

⁴<https://cloud.google.com/translate/>

⁵<https://aws.amazon.com/translate/>

⁶<http://www.grantjenks.com/docs/wordsegment/>

Specifically, the system extracts images from `img` tags and crops images for which the area matches `a` or `button` tag elements from screenshots. It calculates the perceptual hash⁷ of these images and compares them to the image database. If the target image is more than 85% similar to the image in the database, the system determines it to be a logo image. Three types of images are stored in the database: logos of security vendors or software certification company (19 images), package images of fake AV software (33 images), and images of the download buttons (56 images). We extracted images belonging to the three types from the 300 FRAD sites used in the above. Our system counts the number of images that match each type to create feature vectors.

HTML Structure

Here, we explain the features extracted from the HTML structure that we use for identifying FRAD sites. As with previous works that identify specific types of malicious web pages [65, 68], the numbers of `a` and `iframe` tags are important indicators of FRAD sites. Also, FRAD sites often re-use web page templates so that they have similar structures of HTML source codes. In other words, the frequency of HTML tags and combinations of those numbers characterize FRAD sites. To find such features, the system counts the number of appearances of HTML tags. The HTML tags to be counted are the top 30 tags frequently used in the 300 FRAD sites mentioned above.

4.4 Data Collection

We explain the method used to collect FRAD sites in the wild in order to make the dataset employed to evaluate our classification model. We first collected the names of cyber threats. Then, we searched for and gathered candidates of FRAD sites using the names of those cyber threats. Finally, we manually created a labeled dataset for our evaluation experiment.

4.4.1 Collecting Cyber Threats

We collected the names of cyber threats to make search queries to find candidate FRAD sites. As described in Section 4.2, FRAD sites prepare many entries that introduce ways of removing specific cyber threats such as malware detection names and malicious domain names. To collect such names efficiently, we crawled the database pages of security vendors (e.g., Symantec Security Center⁸) and a security community site (e.g., `malwaretips[.]com`) in October 2019. We collected 806 names of threats, including 500 malware detection names, 200 malicious domain names, and 106 popup messages.

⁷<https://github.com/JohannesBuchner/imagehash>

⁸<https://www.symantec.com/security-center/a-z>

4.4.2 Web Search

We created search queries using the collected names of cyber threats and gathered the URLs of web pages using a search engine. To collect FRAD sites efficiently, we added “how to remove” to the name of the cyber threat to create the search query, instead of searching only for the name of the threat. We found that we can collect more FRAD sites by searching with “how to remove” in our experiment described in Section 4.6.3. To collect search results systematically, we used Microsoft Bing Web Search API ⁹ and gathered 34k URLs. We chose one URL for each domain name from among the gathered URLs. As a result, we extracted 4,188 URLs with 4,188 unique domain names to crawl.

4.4.3 Creating the Dataset

We crawled 4,188 web pages using our system and created a labeled dataset. Since there is no existing URL blacklist that accurately identifies FRAD sites, we manually labeled them by analyzing the crawled web pages and actually accessed them as necessary. To efficiently conduct this process, we created a web application that displays screenshots and buttons to choose labels (FRAD and non-FRAD sites). This application extracts information about the crawled web pages from our MongoDB database and generates the web pages for labeling. We implemented it using Node.js and the Express¹⁰ framework. We labeled web pages as FRAD sites if they satisfied following heuristic rules. If not, we labeled the web pages as non-FRAD sites.

- i. We check whether a web page introduces a removal guide for a specific cyber threat. If so, we check rule ii.
- ii. We check whether the web page has visual characteristics specific to FRAD sites, as described in Section 4.3.2. Specifically, we check whether the web page has an image of a fake AV software package or a logo of a security vendor or a software certification company. We also check screenshots of the removal instructions or download buttons, which are often shared with multiple FRAD sites. If the web page has these characteristics, we identify it as an FRAD site. If not, we further check rule iii.
- iii. We confirm that clicking a download button on the web page triggers a download of a fake AV software installer or initiates a web transition to a distribution or payment site for fake AV software. We performed this process by manually accessing the web page and clicking the download button.

From the 15-h labeling process, we obtained 804 web pages of FRAD sites with 804 unique domain names. To create a dataset, we randomly selected 800 web pages from these FRAD sites. We also randomly selected 800 web pages from non-FRAD sites, which are the web pages remaining after excluding the 804 web

⁹<https://azure.microsoft.com/en-us/services/cognitive-services/>

¹⁰<https://expressjs.com/>

pages of FRAD sites. Since we collected the non-FRAD sites using the same search queries as for the FRAD sites, they often introduce removal information for cyber threats, details of malware, or introductions to legitimate AV software, just as FRAD sites do. Thus, it is a challenging task to identify FRAD sites accurately from these similar web pages.

4.5 Evaluation

We next evaluated the detection capability of our system in terms of its capability to classify web pages accurately as FRAD sites or non-FRAD sites. We also conducted an experiment to discover unknown FRAD sites in the wild using the trained classification model.

4.5.1 Detection Accuracy

We first evaluated the detection accuracy of our system using the balanced dataset including 800 FRAD sites and 800 non-FRAD sites. We used a random forest classifier as the machine learning algorithm for two-class classification, because we can easily tune it due to the small number of hyper parameters to be considered. We conducted a 10-fold cross validation to determine how accurately our system performed classifications. We found that our system classified web pages with a 98.8% true positive (TP) rate ($= \frac{TP}{TP+FN}$), where FN = false negative, a 3.3% false positive (FP) rate ($= \frac{FP}{FP+TN}$), and with 96.8% precision ($= \frac{TP}{TP+FP}$). The system identified 26 non-FRAD sites as FRAD sites (FPs). Examples include articles from security vendors that introduce malware information, ranking web pages for legitimate AV software, and blog entries that describe correct removal instructions. Five FPs were security vendors’ web pages that often appear in search results when searching for removal information for cyber threats. We can therefore reduce FPs by placing the domain names of major security vendors on a whitelist. Examples of false negatives include web pages with domain names that do not include words such as “remove” or “malware.” Other false negatives do not contain visual features such as images of fake AV software packages or logos of security vendors.

4.5.2 Detecting Unknown FRAD Sites

To collect unknown FRAD sites that have not been found in Section 4.5.1, we conducted additional data collection and detection using our classification model, which has high detection accuracy.

Additional Data Collection

We first describe additional data collection to find more FRAD sites in the wild, such as non-English FRAD sites and FRAD sites with content copied from other sites. In the process of creating the dataset described in Section 4.4, we found many

FRAD sites written in various languages. Some of them were translated automatically according to the browser’s language setting when the web pages were loaded. Some web pages were also written in multiple languages to enable users to switch languages. In addition, we found FRAD sites dedicated to certain languages. In such cases, the domain names contain words in those languages (e.g., “entfernen” in entfernen-spyware[.]example and “eliminar” in eliminarvirus[.]example), as described in Section 4.3.2. We also found that FRAD sites are often copied from other FRAD sites and from legitimate sites that introduce specific malware removal information. These FRAD sites not only use the names of cyber threats extracted from legitimate sites but also copy page titles or entire articles from them. To find such FRAD sites, we collected page titles from legitimate sites (malwaretips[.]com and malwarefixes[.]com) and from the 804 FRAD sites we labeled, which include non-English sites, and we searched for the titles using Bing API. Although it is difficult to create search queries in multiple languages to collect non-English FRAD sites, we can gather them efficiently in this way. We gathered 16k page titles from these web pages and collected 836,731 URLs (111,161 domain names) from these search. We extracted up to three URLs from each domain name and crawled them (120,577 URLs) using our system.

Detection Result

As a result of the classification of additionally crawled web pages, we identified 6,130 URLs as FRAD sites. To find FPs, we manually checked web pages classified as positive in the same way as described in Section 4.4.3. Examples of FPs include the following. Some technical-support scam [69, 70] sites were falsely identified as FRAD sites, because they offered support for malware removal and displayed noticeable phone numbers and web-chat support. These FPs are not FRAD sites, however, because they did not lead users to fake AV software but instead are actually malicious web pages themselves, which are listed in VirusTotal¹¹. Moreover, our system falsely detected pirate web pages that introduce free downloads of fake AV software. Although such fake AV software is useless and not very well-known, some web pages illegally offered such software. Other FPs include software review and download sites, which distribute fake AV software as well as legitimate software. We also found FPs similar to those described in Section 4.5.1. By excluding these FPs, we finally determined 5,780 URLs (2,109 domain names) as FRAD sites. The precision of this classification result was 94.3%. Although this precision is somewhat less than the results obtained in Section 4.5.1, we accurately identified FRAD sites. The reason for this decrease in detection capability is that we changed the search queries from “how to remove” and the name of threats (used in Section 4.4.2) to page titles of known FRAD sites, so that the types of web pages in the search results were somewhat changed.

¹¹<https://www.virustotal.com>

Summary of Collected FRAD Sites

Overall, in this chapter we have identified 2,913 domain names, including the newly discovered 2,109 domain names, to be FRAD sites. To confirm the FRAD sites already reported by security vendors, we searched for all 2,913 domain names in VirusTotal. Of the total, 32.7% (952 domain names) of the domain names had URLs that had already been detected by one or more vendors. We also found 21.5% (626/2,913) of the domain names had URLs that are sources of detected files. Although some FRAD sites have been detected by a small number of security vendors, most of the FRAD sites we found in this chapter have been unreported to date. These FRAD sites are less likely to be filtered out from search results, even if they were reported as malicious. Thus, most of these FRAD sites remain easily accessible to users and remain threatening to them.

4.6 Measurement Study

We measured the ecosystem and risk of FRAD sites using both passively collected statistical data of user accesses and actively crawled data. In the experiment described above, we found FRAD sites using our system and simply checked the detection status for each of them on VirusTotal. Here, we analyze deeply the 2,913 domain names of FRAD sites that we found in Section 4.5 in terms of incoming traffic to those FRAD sites, the distribution of fake AV software from those sites, and poisoned search results that are occupied by FRAD sites.

4.6.1 Incoming Traffic to FRAD Sites

To find out what browsing behaviors of users are at risk of reaching FRAD sites, we analyzed the incoming channels (i.e., ① in Figure 4.1 in Section 4.2) of the FRAD sites that we found in Section 4.5. To this end, we need data on the history of user accesses to and traffic volumes of those web pages. Thus, we leveraged the statistical data provided by SimilarWeb¹², which passively observes hundreds of millions of global devices and covers over 220 countries and territories. Using this approach, we collected statistical data from October to December in 2019 that we used in the measurement studies described below.

Overview of Incoming Traffic

We first show an overview of seven types of incoming traffic to FRAD sites. We investigated 1,451 domain names of FRAD sites for which data are available in SimilarWeb (out of 2,913 domain names of the FRAD sites we discovered in this study). Note that statistical data of web pages with few user accesses are not provided. These FRAD sites have 73.5 million visits per month in total. Figure 4.3 shows the percentage of traffic to the FRAD sites from each incoming channel. The channels consist of seven labels: *Search* (accessed from a search engine), *Direct* (directly accessed by entering URLs in a web browsers), *Referral* (accessed

¹²<https://www.similarweb.com/>

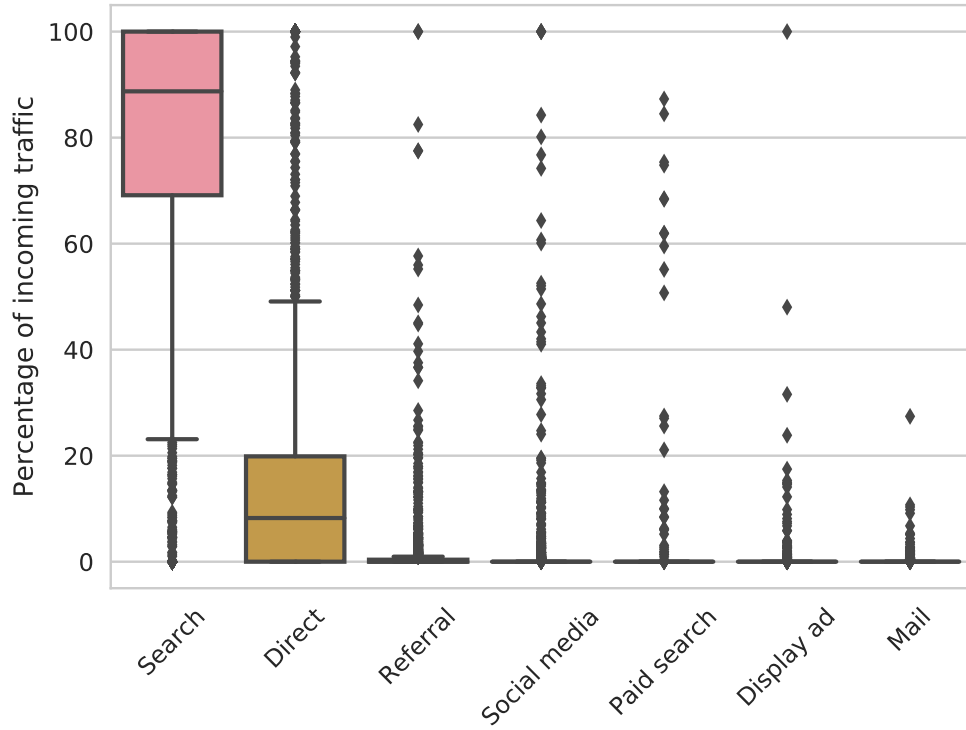


Figure 4.3: Percentage of incoming traffic to FRAD sites from each channel.

from other web pages), *Social media* (accessed from Social Media), *Paid search* (accessed from keyword advertisements on search engines), *Display ad* (accessed from advertisements on web pages), and *Mail* (accessed from hyperlinks on email). Note that the incoming traffic measured as Mail comes only from web mail. Incoming traffic from email client software or other applications is measured as Direct. The mean values of Search, Direct, Referral, and Social media were 76.7%, 16.5%, 1.7%, and 1.7%, respectively. The value for each of the other three channels is less than 0.6%. Paid search, Display ad, and Mail have few data for further investigation. Also, we only know the amount of incoming traffic that we have shown here from the data of Direct. Therefore, in the following, we analyzed the detail of three channels: Search, Referral, and Social media.

Search

To find out how users reached FRAD sites via search engines, we investigated the statistics of the search queries. We extracted the top 10 English search queries (4,510 unique queries in total) for each FRAD site and categorized them. Table 4.2 shows the categories and the number of search queries. We found that 47.5% (2,143/4,510) of the search queries were related to the names of specific cyber threats. They included malware detection names (e.g., trojan:win32/bearfoos.a!ml), malicious domain names, and alert dialog messages (e.g., “your computer is in-

Table 4.2: Search queries used by the users to reach FRAD sites.

Category	Search query	#
Cyber threats	how to <remove><threat>	576
	<remove><threat>	438
	<threat>	849
	is <threat> safe ?	27
	what is <threat>	113
	<error>	140
Download	download <software>	421
	crack <software>	101
Fake AV software	<fake AV software>	66
Other	<other>	1,802
Total		4,510

fectected with dangerous viruses”). Among them, 12.8% (576/4,510) are search queries combining “how to” with words meaning removal (e.g., “remove”, “delete”) and the names of cyber threats. We found that 9.7% (438/4,510) of the search queries combined words meaning removal with the names of cyber threats. Users also searched for the names of cyber threats alone (18.8%, 849/4,510) or for software or OS error messages (e.g., “MSVCP140.dll missing”). Thus, many users reach FRAD sites by searching for cyber threats and corresponding removal guides. The names of fake AV software were also used as search queries to reach FRAD sites (66/1,802). We found that 11.6% (522/4,510) of the search queries were used to search for downloads of software such as office software or video games and guides of cracking them. Forty percent (1,802/4,510) of the search queries were not included in these categories.

Social Media

We also analyzed incoming traffic from social media. We investigated 167 FRAD sites for which statistical data for queries incoming from social media is available from SimilarWeb. Table 4.3 shows the top 15 social media that led users to FRAD sites and the number of FRAD sites to which users were redirected from each type of social media. Users visited 95.8% (160/167) of FRAD sites from YouTube and 66.5% (111/167) of those from Facebook. Attackers create social-media accounts for these FRAD sites and post videos or messages to lure users to FRAD sites. These accounts pretended to be official accounts that use the web-site names or domain names of FRAD sites. They introduce removal information for cyber threats in the same way as entries for FRAD sites, and they put hyperlinks leading to FRAD sites in the description of their videos and messages. We found that some accounts post such instruction videos on YouTube several times a day. These videos got as many as 700k views. We also found that attackers created such accounts across multiple social media. In summary, attackers not only optimize search results to lead users directly to FRAD sites, but also they use various social

Table 4.3: Top 15 social media that led to FRAD sites.

Social media	# of FRAD sites
Youtube	160
Facebook	111
Reddit	58
Quora	35
Pinterest	22
Pocket	9
Twitter	7
Linkedin	6
Instagram	5
WhatsApp	2
SoundCloud	2
Google Groups	2
DeviantArt	2
Yammer	2

Table 4.4: Categories of referral web pages to FRAD sites.

Category of referral web pages	#
Computers Electronics and Technology	517
Games	29
News and Media	25
Science and Education	22
Business and Consumer Services	20
Arts and Entertainment	19
Hobbies and Leisure	8
Adult	8
Reference Materials	7
E-commerce and Shopping	6
Vehicles	2
Reference Materials	2
Gambling	2
Community and Society	2

media to increase user accesses to FRAD sites.

Referrals

In addition, we investigated referral traffic that leads users to FRAD sites. In other words, we analyzed the incoming traffic to FRAD sites when users accessed them from other web pages, excluding search engines and social media. We found that users visited 891 web pages belonging to various categories before reaching FRAD sites. Table 4.4 shows the SimilarWeb categories of these referral web pages. The most common category of referral web page is Computers Electronics and Technology, which includes forum and community sites such as `social.technet.microsoft[.]com.`, `ubuntuforums[.]org`, and `discussions.apple[.]com`. In

most cases, attackers abuse these sites, where anyone can post messages, to impersonate good users who introduce removal information for cyber threats with URLs of FRAD sites. The web pages categorized as Games (e.g., steamcommunity[.]com) were used in the same manner. Attackers also posted FRAD sites' URLs in comment sections in articles in News and Media and other categories. In short, attackers leverage popular web pages where they can post comments and hyperlinks to lure users to visit FRAD sites.

4.6.2 Downloads and Page Transitions from FRAD Sites

To identify threats that occur when users access FRAD sites, we performed an additional crawling experiment. While we simply found FRAD sites using our system in Section 4.5, and we investigated users' incoming traffic to them in Section 4.6.1, the malicious activity derived from them was not revealed by these experiments. Therefore, we actively crawled the FRAD sites and collected installers of fake AV software and their respective distribution sites. To this end, we added a function to the crawler of our system to enable it to detect a download button on an FRAD site and click it. Then we analyzed the downloaded files and transferred the web pages from those FRAD sites.

Collecting File Downloads and Web-Page Transitions

We first describe the details of the new function that enables our crawler to interact with the FRAD sites. The crawler crops images with areas that match the `a` tag and `img` tag elements of FRAD sites. If the crawler finds a “download” string in the images using optical character recognition, it clicks on that area. We used two types of UserAgent with different OS (Windows 10 and macOS v10.14). This is because FRAD sites change the fake AV software to be distributed according to the UserAgent's OS, typically Windows or Mac. To collect the URLs of FRAD sites to crawl, we searched for the 2,913 domain names of FRAD sites using Bing API and selected up to three URLs based on the search results for each domain name. The reason for this is that web pages of FRAD sites with the same domain names can lead to different destinations (e.g., different software distribution sites) depending upon their URLs. To find more fake AV software, we collected 8,099 URLs and crawled them twice with two types of UserAgent. As a result, the crawler downloaded 4,548 files with 594 unique MD5 hash values and reached 136 domain names (630 URLs) of web pages from FRAD sites. In the following, we investigated the downloads of fake AV software originating from the FRAD sites (i.e., ③ in Figure 4.1 in Section 4.2), web pages transferred from those sites (i.e., ② in Figure 4.1), and redirectors that relayed these downloads and web page transitions.

Fake AV Software Downloaded from FRAD Sites

We analyzed the files that our crawler downloaded (see ③ in Figure 4.1) to identify the installers of fake AV software. First, we checked 594 files with unique MD5

hash values on VirusTotal and found that 89 of those files had been detected. To specify fake AV software families from the detected files, we manually analyzed and searched them using their filenames and metadata (e.g., product name, legal copyright, and file description) read by ExifTool¹³. We examined whether the 89 files were related to malware removal, registry fix, or speed up based on the above information and on the software distribution sites that we obtained from the search results. We classified 84 files into 58 unique fake AV software families with different software names. All 58 fake AV software families have software distribution sites reachable from search engines. The software distribution sites profess to be official sites for these fake AV software families. For example, these sites show download and purchase menus and provide customer support such as web chats or toll-free calls. The remaining five detected files were not fake AV software but instead were malware that pretend to be installers of legitimate software, such as music-production software and video games.

To find more fake AV software from the 505 undetected files, we compared their filenames and metadata with those of the classified 58 fake AV software families. As a result of determining files with the same strings as the fake AV software, we additionally found 189 files to be fake AV software. Overall, we found 278 files (31 `dmg` files and 247 `exe` files) of the 58 fake AV software families.

Web Pages Transferred from FRAD Sites

We also analyzed the web pages of 136 domain names that our crawler reached after clicking on download buttons (see ② in Figure 4.1). In the above measurements, we investigated fake AV software directly downloaded from FRAD sites. However, FRAD sites also navigate users to software distribution sites that lure them to purchase and download fake AV software. To find such web pages, we analyzed the crawled data (e.g., screenshots of web pages) and manually classified the malicious web pages. We first checked the 136 domain names on VirusTotal and found that 57 domain names were detected. We then specified the web pages that offered license purchases of known fake AV software or were related to malware removal, registry fixes, and speed-up from the web pages of the 57 detected domain names. We found that 34 domain names were related to distributions of fake AV software, including six domain names of payment sites and 27 domain names of software distribution sites. The payment sites required inputting credit card numbers and personal information to purchase fake AV software. Out of the 27 domain names, we found that 18 domain names were distribution sites for 18 new fake AV software families in addition to the measurements described above, where we found 58 fake AV software families. Thus, we found 76 fake AV software families in total. The detected domain names also included five domain names of FRAD sites that we found in Section 4.5. That is, users may be transferred from one FRAD site to another. We also found malicious web pages that distribute malicious Chrome extensions. We found 14 domain names associated with such threats and four domain names related to distributions of other types of malware.

¹³<https://exiftool.org/>

Table 4.5: The percentage of FRAD sites included in search results.

	threat name <threat name>	remove <threat name>	how to remove <threat name>
Malware	69.4%	87.9%	87.9%
Domain name	88.5%	93.5%	88.0%
Extension	36.1%	85.1%	87.2%
Total	70.6%	89.7%	87.8%

Redirectors

To reveal the network infrastructure related to the distribution of fake AV software, we investigated the redirectors that relayed the above fake AV software downloads and web page transitions. We analyzed the network traffic that our crawler captured and extracted redirectors for which the effective second-level domains (e2LD; e.g., example.com is a e2LD of www.example.com) are different from those of the source web pages (i.e., the FRAD sites) and destination web pages. We found 169 domain names (38 e2LD names) as redirectors of 1,048 URL redirections associated with fake AV software downloads and web transitions to software distribution sites. Nine of these domain names were known advertising domain names listed in EasyList¹⁴. In addition, we found a small number of redirectors that were involved in many fake AV software distributions. For example, we found that 76.4% of the URL redirections were associated with just two domain names: safecart[.]com and revenuewire[.]net. These two redirectors navigated to 17 and 14 fake AV software families, respectively. The domain name safecart[.]com not only is a redirector but also is a payment web page that prompts users for their credit card numbers. Some redirectors, such as reimageplus[.]com and paratologic[.]com, which are software distribution sites, navigated to other software distribution sites.

4.6.3 Search Poisoning

We conducted a further measurement experiment to analyze the percentage of FRAD sites in the search results. In Section 4.6.1, we used statistical data to investigate search queries that users used to reach FRAD sites. Then, we determined the risk of users reaching these FRAD sites by actually searching with those search queries and analyzing the search results. When users search for specific names of cyber threats to find removal information, many FRAD sites prominently show up in search results. To confirm these poisoned search results, we investigated 150 search queries, combining 50 cyber threats and three search patterns. The three search patterns are those that users frequently use, as found in the measurements in Section 4.6.1: “how to remove” and the name of a cyber threat, “remove” and the name of a cyber threat, and only the name of a cyber threat. We extracted the latest names of cyber threats from public lists: 20 malware detection names from Symantec Security Center and 20 malicious domain names from malwaretips[.]com. Also, we randomly chose 10 malicious browser extensions out of 14 browser extensions that we found in Section 4.6.2. We investigated the top 10 search results

¹⁴<https://easylist.to/>

for each search query, which are the top result pages from popular search engines such as Google and Bing.

We collected 1,461 web pages from the top 10 search results for each of the 150 search queries in total. By matching the 2,913 domain names of the FRAD sites collected in Section 4.5.2, we found that 1,207 web pages (82.6%) were FRAD sites. Table 4.5 shows the percentages of FRAD sites included in the search results for each search query and the names of the cyber threats. When we searched for the names of cyber threats with “how to remove” or “remove,” the percentages of FRAD sites were 87.8% and 89.7%, respectively. The FRAD sites were also included at a high rate in the results of searching only for the names of cyber threats. In particular, 88.5% of search results for the domain names were FRAD sites. Search results for malicious browser extensions did not include many FRAD sites (36.1%), but there was less useful information available for users to use to remove the threats or determine whether they are malicious. We also found 22 YouTube web pages as search results, with videos and descriptions that introduced FRAD sites. We found that 26.7% (40/150) of the search queries returned search results for which the top 10 web pages were all FRAD sites. In summary, we found that most of the search results were occupied by FRAD sites when users searched for removal information for cyber threats, making it difficult for users to reach correct information.

4.6.4 Distribution Sites of Fake AV software

We analyzed the incoming traffic to distribution sites of fake AV software using SimilarWeb’s statistical data. From crawling results and manual web search, we identified 76 distribution sites, which were web sites pretending to be the official sites of the 76 fake AV software identified (Section 4.6.2). The purpose of this measurement was to identify the channels other than FRAD sites through which users accessed the distribution sites of fake AV software. We extracted 59 distribution sites for which data are available in SimilarWeb. The total user access to these sites was 22.6 million per month. Our analysis of the incoming traffic in terms of referral web pages, ad networks, and ad publishers is presented in the following sections.

Referrals

We first investigated referral web pages, which are web pages that the users reached before reaching the distribution sites of fake AV software by clicking links or URL redirections. We found 33 distribution sites with valid referral traffic in SimilarWeb data. The traffic originates from 469 domains of referral web pages. In total, 66 domains of FRAD sites (14.1%) were included in the referral web pages. Note that actually a greater incoming traffic may be associated with FRAD sites because in some cases, users reach FRAD sites via intermediate sites such as ad networks and payment sites. Forum (18 domain names) and community sites (5 domain names) were also used to direct users to the distribution sites as in the case of the incoming traffic to FRAD sites (Section 4.6.1). Attackers pretended to be legitimate users

Table 4.6: Top 15 ad networks that redirected users to distribution sites of fake AV software.

Ad network	# of distribution sites
Google Display Network	12
Skimlinks	6
RevenueWire	6
AdSupply	4
Yahoo Advertising	3
Outbrain	3
ClickBank	3
Zedo	2
TripleLift	2
TrafficShop	2
TrafficHunt	2
TORO Advertising	2
PopMyAds	2
Impact	2
Digital River	2

to post links to the distribution sites on such forum and community sites.

Ad Networks

Our crawler often reached FRAD sites via advertising domain names (Section 4.6.2). We analyzed the advertising traffic of distribution sites of fake AV software. We examined the traffic delivered by ad networks and found that SimilarWeb reported 19 distribution sites that were accessed by users. Table 4.6 lists the top 15 ad networks. Even major ad networks (e.g., Google Display Network and Yahoo Advertising) distributed some of the distribution sites of fake AV software. Meanwhile, some ad networks such as RevenueWire that distribute installers of fake AV software were also involved in the delivery of multiple distribution sites of fake AV software. Some ad networks (e.g., PopMyAds) that were reported to be associated with malicious popup ads [63] were used to deliver the distribution sites.

Publishers

To identify the web pages that were the source of advertising traffic to distribution sites of fake AV software, we examined publisher sites, which display advertisements on their content, using SimilarWeb data. We found 528 domain names for web pages showing ads that led to 20 distribution sites of fake AV software. Of those domain names, 14 belonged to FRAD sites. We found that users accessed the distribution sites from social media platforms such as YouTube (to 7 distribution sites) and Reddit (to 3 distribution sites). Some legitimate web pages with heavy traffic (e.g., play.google[.]com and mail.google[.]com) also had advertisements that led to the distribution sites. Surprisingly, such advertisements were even placed on the web pages of legitimate antivirus software such as norton[.]com, mcafee[.]com,

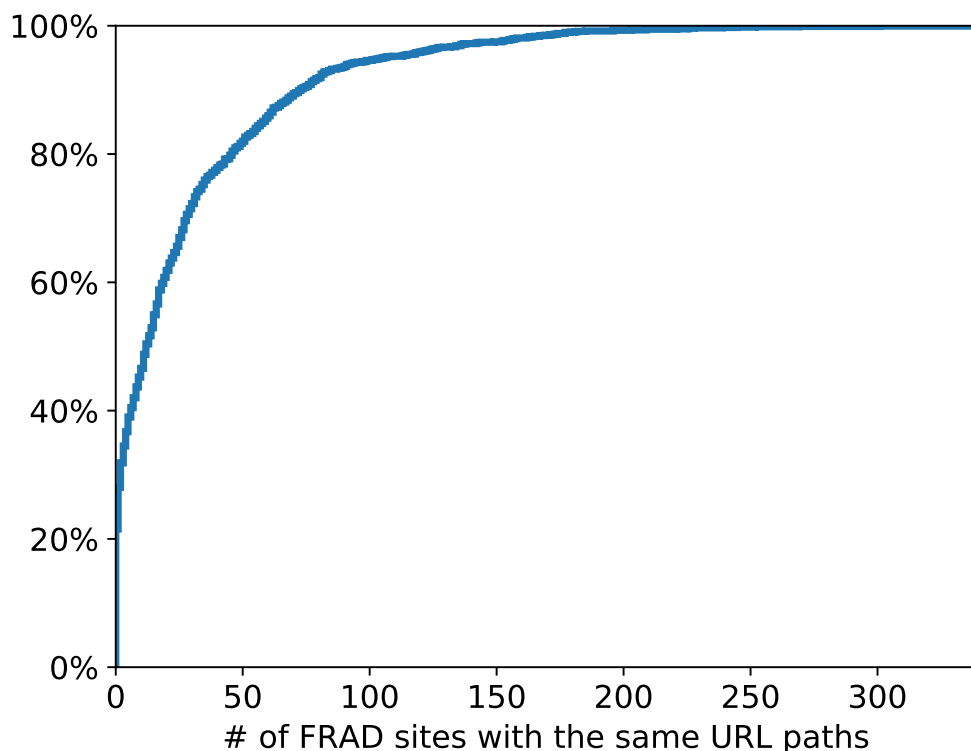


Figure 4.4: Number of FRAD site domain names with the same URL paths.

and `avg[.].com`.

4.6.5 Structure and Content of FRAD Sites

In order to understand the FRAD site structure, duplication between FRAD sites, and update frequency of these sites, we analyzed the sitemaps of FRAD sites. The sitemaps provide structured listings of their web pages and page content. We automatically crawled the sitemaps by accessing `robots.txt` and `sitemap.xml` at the root of FRAD sites and collected the URLs and the date of last modification. Out of the 2,913 domain names of FRAD sites that we discovered (Section 4.5), only 1,833 domain names had sitemap pages. We analyzed in detail the information on each of these FRAD sites.

Duplicate Entries across FRAD Sites

We determined the number of domain names with the same URL paths to find shared page content between FRAD sites, as such shared content imply associations between FRAD sites. We extracted the URLs with identical paths components at the end. For example, `frad1[.]example/trojan/how-to-remove-trojan-abc.html` and `frad2[.]example/2019/12/how-to-remove-trojan-abc.html` are considered to share a URL path. To exclude common URL paths (e.g.,

index.html and contact.html), we ignored the paths with less than 10 characters except for extensions (e.g., .html and .xml). Figure 4.4 shows a cumulative histogram of the domain names of FRAD sites that share one or more URL paths with other domain names. We found that 1,438 domain names (78.5%) out of 1,833 domain names shared at least one URL path with other domain names and that 340 domain names shared at most one URL path. We found that 522,261 URL paths of a certain FRAD site were used on one or more other FRAD sites.

Next, we compared the page content of FRAD sites with the same URL paths. While the names of the cyber threats for which the FRAD sites introduce removal methods were the same, the page titles and textual content of the sites did not exactly match. However, we found a number of web pages with partial matches of the textual content. These web pages were created by using synonyms and alternative terms, for example, “Trojan.EXAMPLE1 is a highly vicious computer infection that belongs to Trojan Horses family” and “Trojan.EXAMPLE2 is a highly malicious computer infection that comes from Trojan Horses Family.” To evade judgment as similar web pages by search engines, the owners of FRAD sites first prepare multiple templates of the textual content for each category of cyber threats such as Trojan, Browser Hijacker, and Ransomware. Then, they generate page entries by using synonyms and similar-meaning phrases in the templates.

We found URL paths shared by many FRAD sites were also used in user posts in SlideShare. For example, “[www.slideshare\[.\]net/username/URLpath](#).” Such web pages included the slide format converted from entries from the original FRAD sites and displayed links of these sites. We found 2,680 posts on SlideShare with the same URL paths of 300 FRAD sites.

Frequency of Posting Entries on FRAD Sites

Attackers post multiple entries for each cyber threat (e.g., malware detection names and malicious domain names) on FRAD sites. We investigated the time intervals between the posting of entries on FRAD sites by checking the date of the last modification of each entry. Figure 4.5 shows cumulative histograms of the mean and median values of time intervals between the two closest entries for each FRAD site. On average, 69.0% of FRAD sites were updated at least once a day. The median values are smaller than the mean values because many entries were posted in succession in particular periods of the days. The median values with time intervals less than 1 min were 10.1% of the total. Given the large number of entries posted in a short period of time, FRAD sites may automatically generate entries by using the names of cyber threats collected from the database pages of security vendors and other FRAD sites.

Languages of FRAD Sites

We investigated the distribution of languages used on FRAD sites. We automatically identified the languages of FRAD sites by applying langdetect¹⁵ to the textual content of web pages accessed by our crawler. Note that these sites may support

¹⁵<https://pypi.org/project/langdetect/>

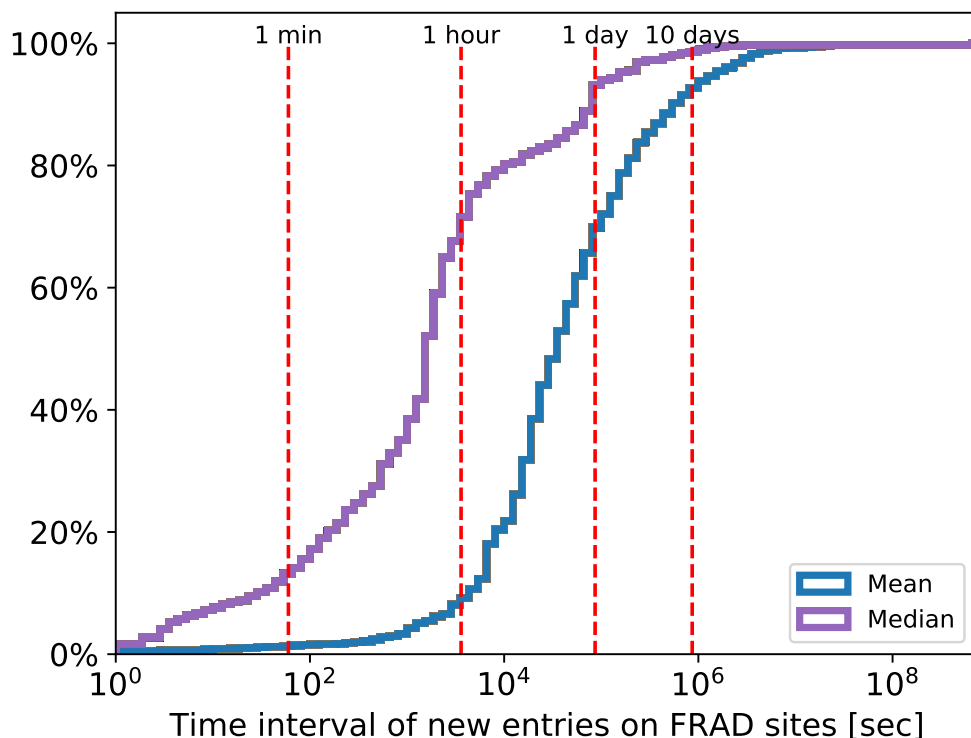


Figure 4.5: Mean and median values of time intervals between entries posted on FRAD sites.

more languages than indicated by this result. This is so because users can choose multiple languages on some FRAD sites, and these sites may be automatically translated depending on the browser language setting of the user (Section 4.5.2). Figure 4.6 shows the number of FRAD sites for each language. We found that 50.4% of the FRAD sites were written in English, and the remainder was comprised sites in as many as 30 languages in all, for example, German (6.3%), Spanish (4.2%), French (3.6%), and Portuguese (3.0%). Thus, FRAD sites were deployed in various languages to lure more users from all over the world. Attackers not only prepare web pages in multiple languages for each FRAD site, but also create web pages that specialize in particular languages by including terms in a particular language in the domain names.

4.6.6 Infrastructure of FRAD Sites

We investigated infrastructure of FRAD sites that we discovered by querying GeoIP2 databases [71] in terms of locations, ASes, and top-level domains (TLDs). Figure 4.7 and Figure 4.8 show countries and organizations to which the IP addresses of the FRAD sites belong. We found that 78.4% of IP addresses were located in the United States. This finding is attributed to the fact that the owners of many FRAD sites use content delivery networks and hosting providers in

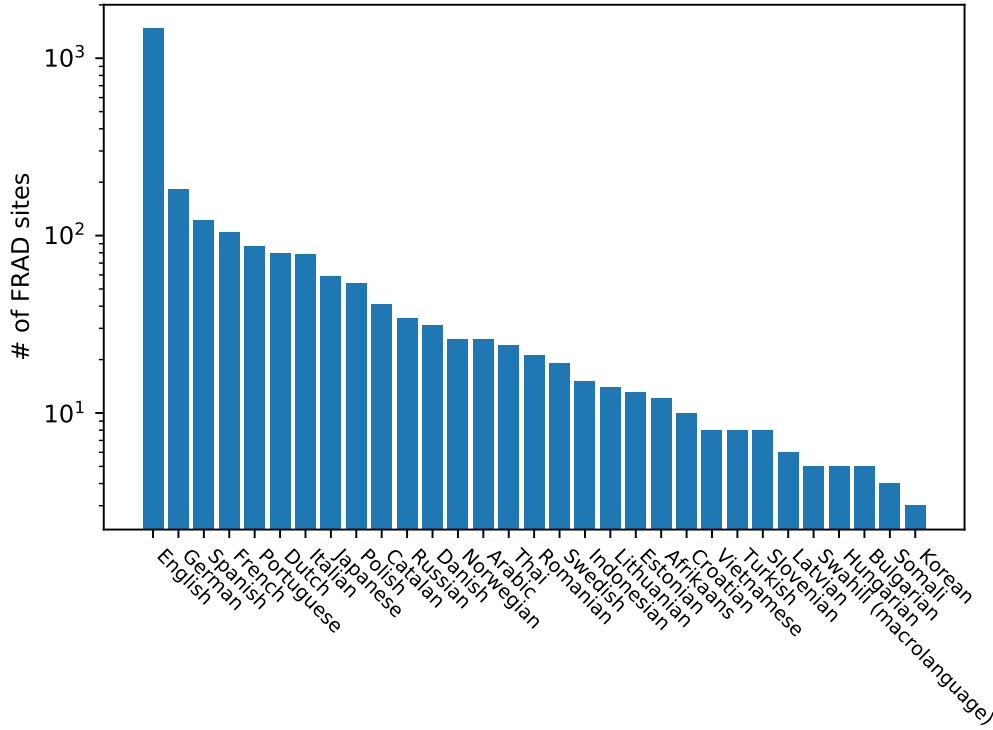


Figure 4.6: Distribution of the languages used on FRAD sites.

the US (see Figure 4.8), such as Cloudflare and Google. The US is followed by Germany (2.6%), Canada (1.8%), Netherlands (1.4%), and France (1.3%), which together host 7.1% of the IP addresses. The remaining 14.5% of the IP addresses were located across 37 countries. In summary, the IP addresses of FRAD sites were owned by 195 organizations located across 42 countries. Figure 4.9 shows top 20 FRAD sites in terms of the TLDs. We found that 59 TLDs were used for FRAD sites; 78.1% of the FRAD sites had the `.com` TLD, of which 20.8% were subdomains of `blogspot[.]com`, which is a blog-publishing service. Some TLDs of FRAD sites provided useful information such as `.info` (33 domains), `.guide` (16 domains), and `.help` (1 domain).

4.7 Discussion

4.7.1 Ethical Considerations

We followed research ethics principles and best practices to conduct this study [56]. We analyzed users' behavior to visit FRAD sites using anonymized statistical data on user accesses for this study. We purchased a license to access data that is legally collected based on SimilarWeb's privacy policy. The information extracted from the web pages we crawled is publicly available data. To reduce server load, our experiment that interacted with download buttons was performed only once for

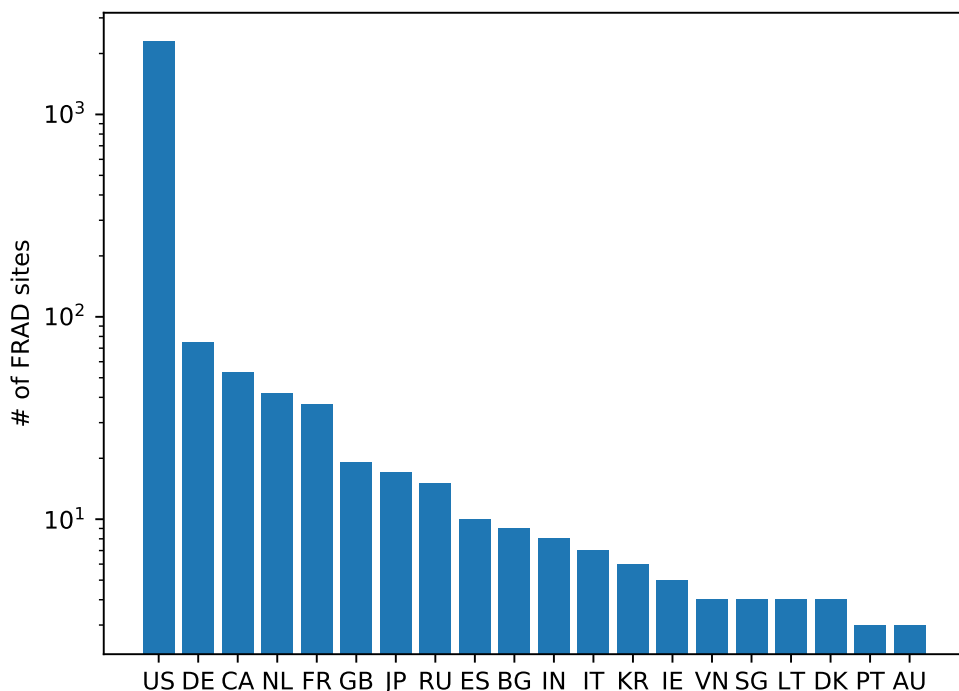


Figure 4.7: Number of FRAD sites per top 20 countries where the IP addresses are located.

each web page that we identified as an FRAD site.

4.7.2 Limitation

Although our system can accurately identify FRAD sites, there are some limitations. Since our system is specialized for collecting and detecting FRAD sites, which are the important platforms used by attackers to distribute fake AV software, detecting software distribution sites is out of scope for this study. We identified software distribution sites that pretended to be official sites for legitimate AV software on the basis of detection results from VirusTotal and manual analysis. We showed that we can visit various software distribution sites from FRAD sites by clicking on the FRAD sites. We also found that these software distribution sites share common network infrastructures, such as ad networks and redirectors. Thus, further analyses focusing on the web pages arriving from the FRAD sites collected by our system should support efficient collections of software distribution sites.

We then discussed a technique that can be used to evade our classification of FRAD sites. Developers of FRAD sites employ phrases related to the removal information for threats in domain names, URLs, titles, and text contents. This is because they use the topic of the web pages to attract or persuade users. They also place logos of trusted companies to disguise FRAD sites as legitimate sites.

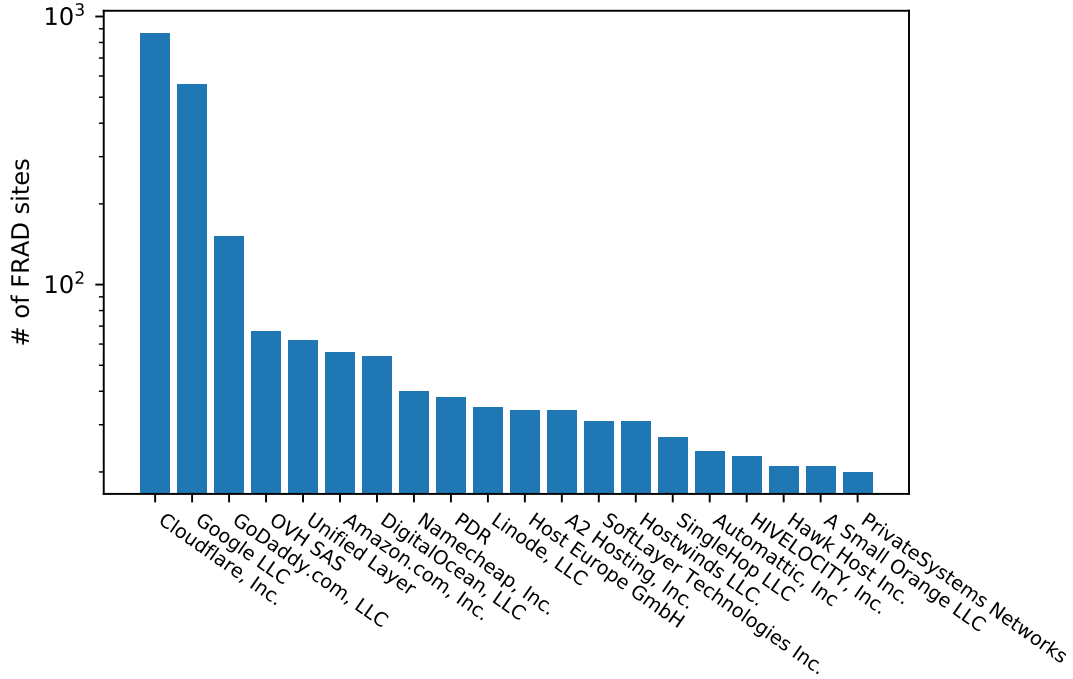


Figure 4.8: Number of FRAD sites per top 20 ASes.

A possible evasion technique would be to remove these characteristics that psychologically affect users. However, this also would reduce the interest of users and the usefulness of the FRAD sites to the attackers. In addition, excluding phrases related to malware removal lowers the SEO rankings of FRAD sites and user accesses. Since our system relies on these characteristics to identify FRAD sites, we can accurately detect high-risk FRAD sites that strongly affect the users' psychology.

Since our collection of FRAD sites depends on search engine results, we have not collected all FRAD sites on the Internet. To efficiently collect FRAD sites, we used the names of the cyber threats that are mainly used by attackers to lure users and leverage search engines, which are the most common channel to lead a user to FRAD sites. As a result, our analysis found that FRAD sites are created in many languages and have a large amount of user access. Our system is useful for continuously collecting FRAD sites to create URL blacklists and for analyzing trends for this type of attack.

4.8 Related Work

We have reviewed related work that investigated the distribution infrastructure for fake AV software and the social engineering techniques attackers use to trick

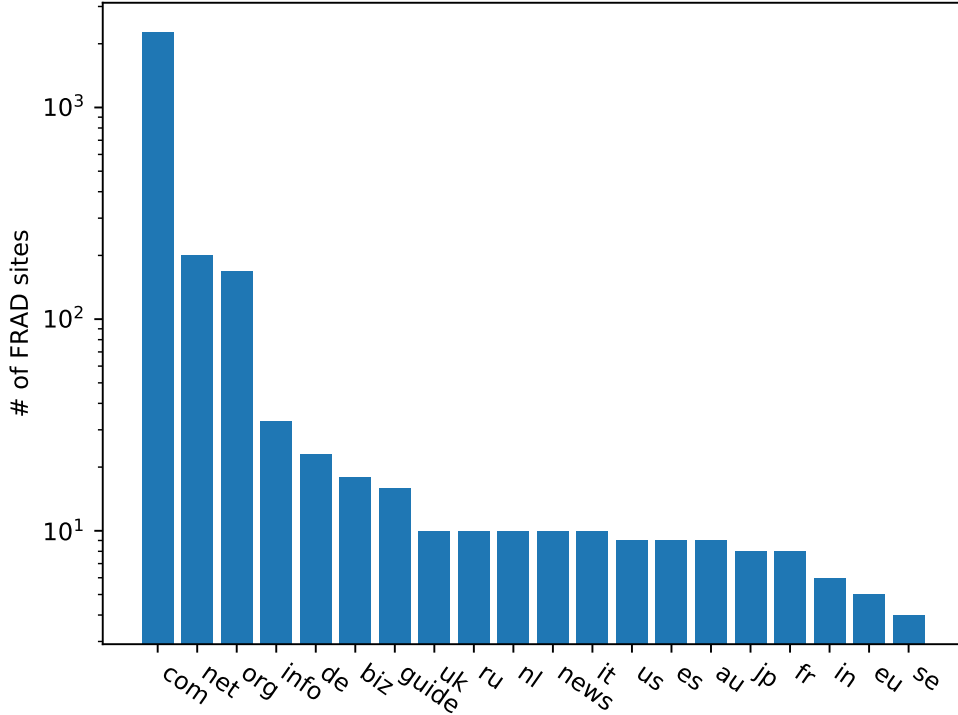


Figure 4.9: Number of FRAD sites per top 20 TLDs.

users. Using a combination of unsupervised, graph-based clustering, Cova et al. analyzed the network infrastructure (e.g., domain registration information and IP addresses) of fake AV software distributions to reveal their ecosystem and attack campaigns [58]. Although they investigated the relationship of servers hosting fake AV software, they did not discuss how users access these web pages. Rajab et al. conducted a measurement study that discovered web pages related to the distribution of fake AV software from data collected by Google [22]. They showed the prevalence of fake AV software in malware distributions on the web. Stone-Gross et al. proposed an economic model and estimated attackers’ revenue by analyzing back-end servers that attackers used to support fake AV software businesses [60]. They identified the incoming channels that users employ to reach distribution sites, such as landing pages that exploit browsers to redirect users. They also described the social engineering techniques used to install fake AV software using web pages that display fake infection alerts. Although these studies analyzed the infrastructure and traditional distribution techniques for fake AV software—such as drive-by downloads and fake infection alerts—new distribution tactics using FRAD sites have not been revealed. There is also related work that describes case studies of fake AV software distribution from social engineering aspects [72, 59, 73, 74, 75, 76, 77, 78, 69, 66, 79, 2, 80, 70, 81, 67, 82, 83]. In most studies, they analyzed fake infection alerts via advertisements that threaten or attract users to install fake AV software. However, no previous study has focused

on the FRAD sites or analyzed attackers' techniques that exploit the psychological weakness of users who are suffering security problems.

4.9 Conclusion

We developed a system that can automatically crawl the web and identify FRAD sites, which lure users to install fake AV software by providing fake removal information for cyber threats. Using our system, the first comprehensive measurement study was conducted to disclose the ecosystem of distributing fake AV software via FRAD sites. We have analyzed both passively collected statistical data on user accesses and actively crawled data to clarify users' risky behavior that leads them to reach FRAD sites and which exposes them to attacks navigated from FRAD sites. Our findings emphasize that it is very difficult for users who are suffering from cyber threats to reach correct removal information, because search results related to the specific cyber threats are poisoned by FRAD sites. Our system is useful for search engine providers and security vendors for excluding and blocking FRAD sites.

Chapter 5

Detection Method for Malicious Packets with Characteristic Network Protocol Header

5.1 Introduction

Most operating systems provide network sockets for sending and receiving data across the network. Network packets sent by the network sockets may have unique characteristics in their header fields. The OS Fingerprinting is a technique that identifies the source operating system by analyzing these packet-level characteristics. p0f [84], which is a typical software that implements this technique, passively observes TCP packets and matches their original signatures to identify operating systems that send the packets. Some types of malware have capabilities of performing the Distributed Reflection Denial-of-Service (DRDoS) attacks [85] or network scanning. Since such malware can implement its own network stack, network packets sent from it also have unique characteristics for each malware. Previous studies proposed methods for identifying malicious packets with unusual header values that are different from those generated by the network stack of operating systems [8, 9]. However, little effort has been done to accurately identify the specific type of malware based on characteristics of network packets.

In this chapter, we propose a method for identifying malware and network tools, which implement their own network stack, by analyzing header fields of network packets. This method uses original signatures and match values on header fields extracted from each network packet in the same way as p0f, which identifies operating systems from each network packet. We created 19 signatures based on macro and micro analysis on network traffic generated by the malware and network tools.

We conducted experiments that evaluate our method by using large-scale network traffic. We used two signatures that identify Morto malware and the network scanning tool ZMap [86]. As a result of extracting network traffic that matches these signatures, the traffic occurred at the same time as the appearance of the malware and the release of tools.

We report a measurement study of malicious activities observed in a large-scale network traffic. We first analyzed network traffic observed by our honeypot that is specialized in observing DRDoS attacks. We analyzed network traffic that matches a signature of malware that is used in DRDoS attacks. The result shows that attackers did not usually use the malware in the actual DRDoS attacks. We then analyzed darknet traffic, which is traffic arriving at unused IP addresses, and extracted network packets that matches a signature of IoT malware. We associated the packets with intrusion and exploit methods of malware.

We also report long-term observations of a large-scale network using our method. We implemented our method in NICTER (Network Incident Analysis Center for Tactical Emergency Response) [87], a cyber-attack observation and analysis system owned by the National Institute of Information and Communications Technology (NICT). As a result of analyzing network-scanning packets related to DRDoS attacks, we found attackers massively used ZMap to find vulnerable servers.

5.2 Feature Extraction of Packet Headers Using Macro and Micro Analysis

In this section, we show that malware and network tools that implements their own network stack send network packets with characteristics in their header fields. As micro-analysis, we performed both static and dynamic malware analysis to analyze their network stack. We also investigated source codes of network scanning tools. As macro-analysis, we used traffic observed in the darknet, which consists of 65536 IP addresses and provided by NONSTOP (NICTER Open Network Security Test-out Platform).

5.2.1 Morto

We performed dynamic analysis of Morto [88, 89] and analyzed network packets created by it. Morto is malware that uses Remote Desktop Protocol (RDP) to infect Windows devices. Security vendors reported that this malware occurred in late August 2011. Once Morto is installed on devices, it starts scanning the local network and the Internet to find devices listening for the RDP port (3389/TCP). Then, Morto tries to remotely intrude other devices by performing dictionary attacks, which use a list of possible password templates (e.g., root/admin) to log in to the devices. We performed dynamic malware analysis and capture network packets to extract features from them. The environment for the dynamic malware analysis is the same as the one used in the previous study [90]. The access control that only allows outgoing TCP SYN packets is configured to prevent exploiting devices on the Internet.

- Malware hash value: 0475c97ddb96252febff864fb778b460 (MD5)
- Experimental period: August 26, 2012 (6 hours)
- Operating system: Windows XP SP2

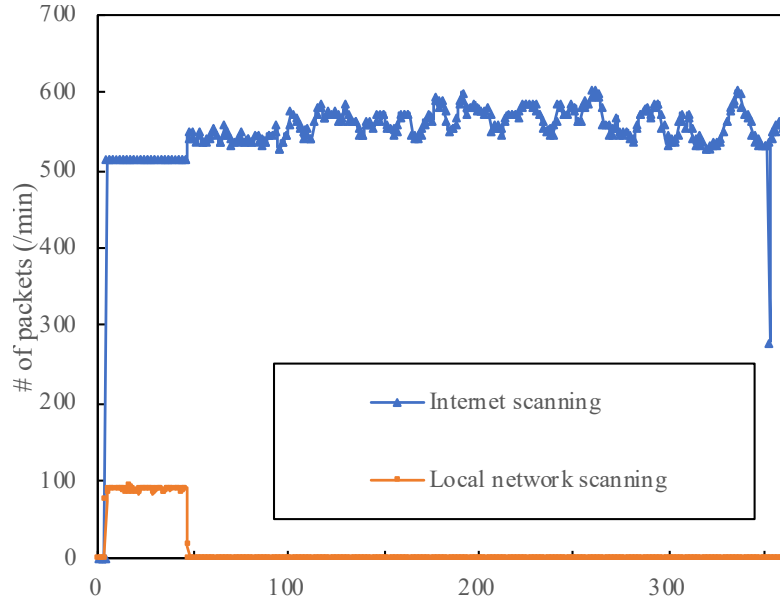


Figure 5.1: Number of network packets observed in 6 hours of analysis time.

In our dynamic malware analysis, network scanning traffic to the RDP port (3389/TCP) is observed. The traffic is divided into two types of network scanning: network scanning for the Internet and network scanning for local networks around the IP address at which the malware was executed. Figure 5.1 shows the number of network packets per minute. All TCP SYN packets that are sent to the local networks is identified as Windows XP by p0f. On the other hand, TCP SYN packets to the global IP addresses can be divided into a small number of packets with characteristics of Windows XP and a large number of packets with no characteristics of any operating system. The latter packets have three fixed header fields: the sequence number of the TCP header is 2406000322, the source port number is 4935, and the ID value of the IP header is 9496. Common operating systems such as Windows and Linux dynamically set values for those header fields. This result shows that Morto implements their original network stack for fast network scanning for the Internet, which require creating more packets than network scanning for local networks. Once Morto finds devices listening for the RDP port, it tries to log in using the network stack of operating systems.

5.2.2 ZMap

We analyzed source codes of the network scanning tool ZMap to investigate its network stack. ZMap is a open-source software developed by University of Michigan for high-speed network scanning. We found that network packets created by ZMap are always set to 54321 in the ID values of the IP header.

5.2.3 Masscan

We analyzed source codes of the network scanning tool Masscan, which is a open-source software developed in 2013 [91]. Masscan can transmit 10 million packets

per second using its original network stack. Previous study [92] showed Masscan sets an exclusive logical combination of three values, the destination IP address, the destination port number, and the sequence number, in the ID value of the IP header of TCP SYN packets. We also found that Masscan sets the IP header's ID value of some types of UDP packets in the similar way as creating TCP SYN packets. Network packets that are sent to three types of protocols using UDP, DNS (53/UDP), NetBIOS (139/UDP), and SNMP (161/UDP), are generated by replacing the TCP sequence number with the ID value of the DNS header, NetBIOS header, and SNMP header, respectively. The following is the method of calculating the ID value of the IP header for each protocol.

$$\text{Ip.id} = \text{Ip.dstaddr} \oplus \text{Tcp.dstport} \oplus \text{Tcp.seq.} \quad (5.1)$$

$$\text{Ip.id} = \text{Ip.dstaddr} \oplus \text{Udp.dstport} \oplus \text{Dns.id.} \quad (5.2)$$

$$\text{Ip.id} = \text{Ip.dstaddr} \oplus \text{Udp.dstport} \oplus \text{Ntb.id.} \quad (5.3)$$

$$\text{Ip.id} = \text{Ip.dstaddr} \oplus \text{Udp.dstport} \oplus \text{Snmp.id.} \quad (5.4)$$

5.2.4 Malware used for DRDoS attacks

We performed both static and dynamic malware analysis to investigate network stack of two malware families IptabLes[93] and XOR botnet[94]. They occurred in 2014 as botnets used for DRDoS attacks.

IptabLes

Iptables is a DRDoS botnet that targets Linux devices. We executed Iptables on our dynamic malware analysis environment using Ubuntu 10.04. As a result, we observed a large number of TCP SYN packets and DNS packets. The TCP SYN packets do not have no characteristics of any operating systems, but have 848 byte payloads, which should normally be zero byte. Also, their TCP sequence numbers were all 848 and the ID values of the IP headers were all 0.

- Malware hash value: b826fb1253a52a3b53afa3b7543d7694 (MD5)
- Experimental period: July 17, 2014 - July 18, 2014 (25 hours)
- Operating system: Ubuntu10.04 (32bit)

On the other hand, we did not find any fixed values in header values of DNS packets. Then, we performed static malware analysis (reverse engineering) and identify the calculation method of packet creation. This analysis revealed that this malware first generates a 2-byte random value for the ID value of the IP header, and calculates the ID value of the DNS header and the UDP source port.

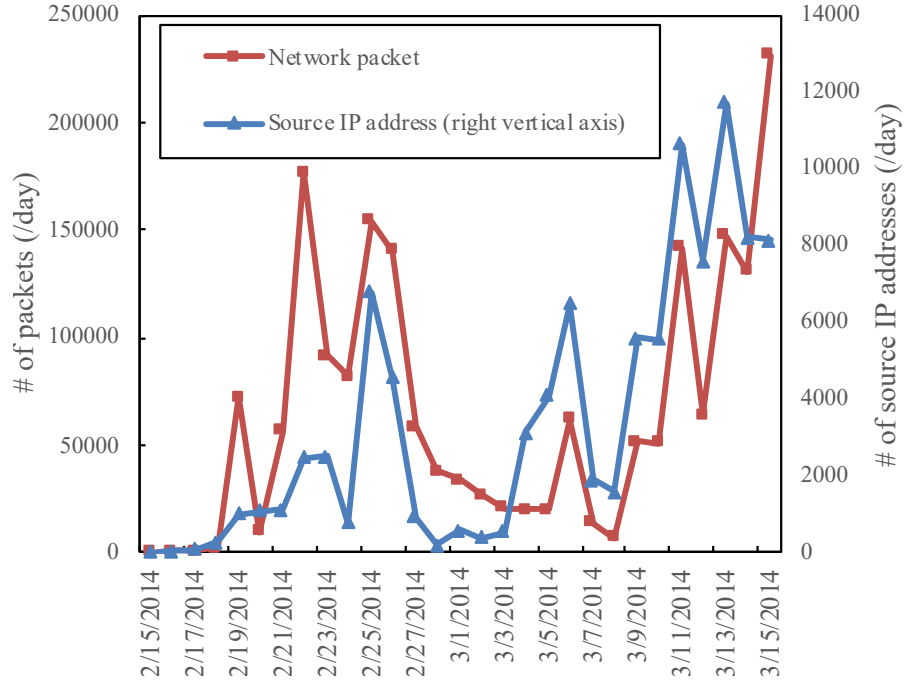


Figure 5.2: Network packets targeting Linux devices.

These values are calculated as the following Eqs. 5.5, 5.6, and 5.7. The ID value of the DNS header is also converted to network byte order after the calculation.

$$\text{Ip.id} = \text{Random_value}. \quad (5.5)$$

$$\text{Dns.id} = \text{Random_value} \& 0x2345. \quad (5.6)$$

$$\text{Udp.srcport} = \text{Random_value} \% 4996 + 1400. \quad (5.7)$$

XOR Botnet

We performed static malware analysis of XOR botnet and investigated the algorithm for generating network packets.

- Malware hash value: 7c903107ebc28a2e98e3247dbde71f02 (MD5)

This analysis revealed that this malware also generate a 2-byte random value for the ID value of the IP header, as well as the UDP source port and the ID value of the DNS header.

$$\text{Ip.id} = \text{Udp.srcport} = \text{Dns.id} = \text{Random_value}. \quad (5.8)$$

5.2.5 Malicious Packets Targeting Linux Devices

We show network packets that were targeting Linux devices and were observed by the darknet. Since February 16, 2014, we have observed a large number of

TCP packets that were sent to 22/TCP (SSH), 23/TCP (Telnet), 8080/TCP, 32764/TCP, and 58455/TCP port of the darknet’s IP addresses. These packets have the same characteristics: the TCP sequence number is 1112425812, the ID value of the IP header is 0, and the TCP window size is 300. Figure 5.2 shows the number of these packets and their source IP addresses per day observed in the darknet between February 15 and March 15, 2014. Security vendors reported that 32764/TCP port is a backdoor port of router intentionally created by the manufacturer and 58455/TCP port is a backdoor port created by Linux.Darilloz, a malware family targeting Linux devices, to control infected devices [95, 96]. The network packets sent to these ports can be used for network scanning to find such backdoors.

5.3 Method

In this section, we propose a method for identifying malware and network tools by analyzing header fields of network packets.

5.3.1 Overview

We describe a brief overview of our proposed method in this section. P0f is a tool for identifying operating systems by analyzing packet-capture data or traffic through network interfaces and matching signatures. p0f extracts the Time to live (TTL) value, the maximum segment size (MSS), the window scale value, and TCP flag values from TCP SYN packets. As we shown in Section 5.2, network packets created by some malware and network tools have unique characteristics in their header fields that is dynamically set by operating systems, for example, fixed values and unique calculation methods are used. These fields are normally used by OS to manage connections between servers and clients. Some malware and network tools designed to send network packets in a non-interactive manner, such as DRDoS attacks and network scanning. Therefore, malware and network tools that implements their own network stack can set those fields arbitrarily.

In this chapter, we propose a method for identifying malware and network tools by analyzing the header fields used by p0f, as well as the fields dynamically set by operating systems. We create signatures for each malware and network tool by micro and macro analysis. Protocols that this method identifies are TCP, UDP packet, and ICMP echo request. This method first extract header fields from a network packet. Then, this method matches the extracted value with the signature one by one or check the results of the calculation. Finally, this method determine the packet to be positive if all fields match the signature.

The originality of this method is to check header values that is dinamically set by operating systems and are not used by p0f. In addition, this method can identify not only TCP packets, but also UDP packets and ICMP echo request packets by analyzing their header values.

Table 5.1: Signature format of TCP packets.

Header	Field
IP header	ID
	TTL
TCP header	Source port
	Destination port
	Sequence number
	ACK number
	Window size
	Option

Table 5.2: Signature format of UDP packets.

Header	Field
IP header	ID
	TTL
UDP header	Source port
	Destination port
Application protocol header (if exists)	ID (DNS, SNMP, and NetBIOS)

Table 5.3: Signature format of ICMP echo request packets.

Header	Field
IP header	ID
	TTL
ICMP echo request header	ID
	Sequence number

5.3.2 Creating Signatures

We explain the way to create signatures of TCP, UDP, and ICMP echo request packets. Signatures constitute different formats for each protocol. Tables 5.1, 5.2, and 5.3 show formats of each protocol's signature. Since TCP packets constitute IP headers and TCP headers, The signatures of TCP packets have two fields of the IP header and 6 fields of the TCP header. The signatures of UDP packets have two fields of the IP header, two fields of the UDP header, and one field of the application protocol header such as DNS and SNMP. The signatures of ICMP packets consist of two fields of the IP header and two fields of the ICMP echo request header.

We then explain the fields of signatures. When a fixed value is set in each header field of a network packet created by malware or network tools, we set the value such as a single value, any of multiple values, and a wild card (any of all possible values) in the header field. On the other hand, when a header field is calculated by a specific method that uses other header fields, we set the equation to match the signature.

5.3.3 Limitation

The purpose of this method is limited to identify network packets created by malware and network scanning tools that implements their original network stacks. Malware that implements the network stack which does not set a fixed value in

Table 5.4: TCP Signatures.

Signature	IP header				TCP header			
	ID	TTL	Sequence number	ACK number	Source port	Destination port	Window size	Option
Morto_scan	9496	0-64	2406000322	0	4935	3389	65535	
Morto_scan_NAT	9496	0-64	2406000322	0	*	3389	65535	
Dark_dst3389_1	256	65-128	1210253312	0	*	3389	16384	
Dark_dst3389_2	256	65-128	2284205602	0	*	3389	512	
Dark_embedded_linux_1	0	0-64	1112425812	0	22, 23, 8080, 32764, and 58455	300	*	
Zmap_tcp	54321	129-255	*	0	*	*	65535	
Dark_ipid0_1	0	0-64	*	0	*	0	8192	
Iptables_tcp_1	0	129-229	848	0	*	*	16000-18999	848byte payload
Srizbi_1	*	65-128	6509	0	4099	24000	0x1F219A	
Linuxddos1_tcp_1	0	129-255	0	0	*	*	6000	960-980byte payload
Masscan_tcp	Eq. 5.1	129-255	*	0	*	*	1024	
Dark_scan_1	256	65-128	*	0	12200	*	8192	

Table 5.5: UDP Signatures.

Signature	IP header		UDP header		Application protocol header	
	ID	TTL	Source port	Destination port		
Zmap_udp	54321	129-255	*	*		
Masscan_dns	Eq. 5.2	129-255	*	53		
Masscan_ntb	Eq. 5.3	129-255	*	139		
Masscan_snmp	Eq. 5.4	129-255	*	161		
Iptables_dns	Eq. 5.5	129-255	5.7	53	Eq. 5.6 (DNS header's ID)	
Xor_dns	Eq. 5.8	0-178	Eq. 5.8	53	Eq. 5.8 (DNS header's ID)	

the header field or use a specific formula to calculate it is out of scope

5.4 Creating Signatures

We describe the way of creating signatures based on the macro and micro analysis of network packets. We created 19 signatures that identify 19 types of network packets created by 10 malware families and 2 network scanning tools. These signatures are shown in Tabs. 5.4-5.6.

5.4.1 Morto

We created signatures of malware Morto based on the analysis of Section 5.2.1. Signature Morto_scan has fixed values of the ID value of the IP header, the TCP sequence number, the source port number, the destination port number, and the

Table 5.6: ICMP Signatures.

Signature	IP header		ICMP header	
	ID	TTL	ID	Sequence number
Zmap_icmp	54321	129-255	*	0

TCP windows size. This malware sets 128 on the initial TTL value, however, this value is decremented every time that the network packet goes through routers. Therefore, we set this field to a range of 64 to 128. Since the source port number can be changed by NAT (Network Address Translation), we also created signature `Morto_scan_NAT`, whose source port number is set wildcard.

5.4.2 ZMap

We created signatures `Zmap_tcp`, `Zmap_udp`, `Zmap_icmp` based on the analysis of Section 5.2.2. The IP header's ID value of the three signatures are set to a fixed value 54321 and the range of the TTL value is set to 129 to 255. `Zmap_tcp` also has fixed value in the TCP window size field. `Zmap_udp` is a signature that identifies network packets using UDP. Since ZMap set 0 to the ICMP header's sequence number of the ICMP scanning packet, we set this fixed value to `Zmap_icmp`.

5.4.3 Masscan

We created four signatures of Masscan based on the analysis of Section 5.2.3. Masscan uses unique methods (Eqs. 5.1-5.4) to calculate the IP header's ID value of TCP SYN, DNS, NetBIOS, and SNMP packets. Therefore, we extract header values and calculate these Equation to match the signatures.

5.4.4 Malware used for DRDoS Attacks

In this section, we explain signatures of malware `Iptables` and XOR botnet. We created signatures `Iptables_tcp_1` and `Iptables_dns` based on the analysis of Section 5.2.4. The IP header's ID value and the TCP sequence number of `Iptables_tcp_1` have fixed values of 0 and 848. This signature also confirms whether the size of the payload is 848 bytes. `Iptables_dns` uses methods of calculating the ID value of the IP header and the ID value of the DNS header (Equation 5.6 and 5.7).

We created the signature `Xor_dns` based on the analysis of Section 5.2.4. This malware generate a random value to set the ID value of the IP header, the UDP source port, and the ID value of the DNS header. Therefore, this signature confirms that the three type of header files set the same value.

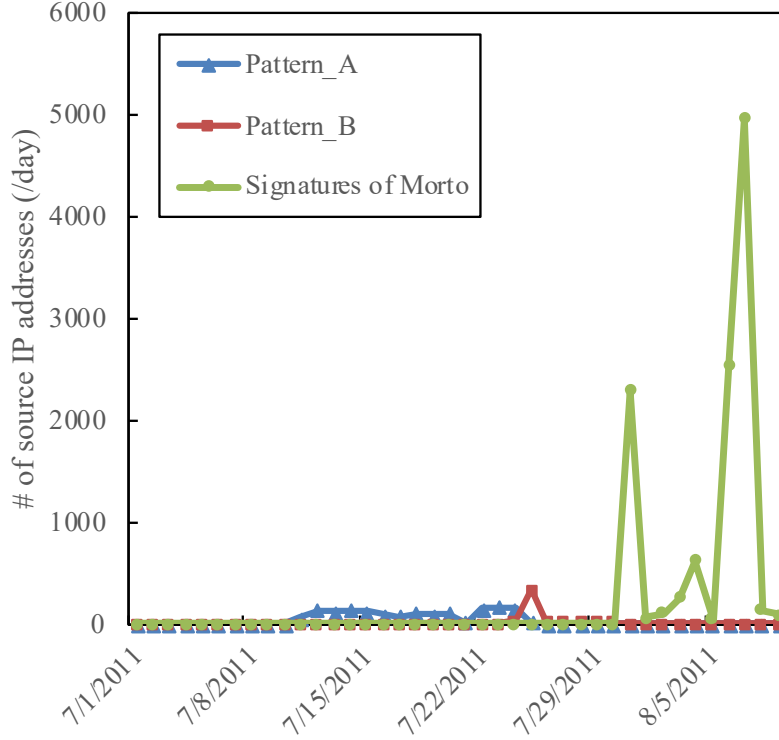


Figure 5.3: Network packets (3389/TCP) observed in the darknet.

5.4.5 Malicious Packets Targeting Linux Devices

We created the signature `Dark_embedded_linux_1`, which identifies network packets targeting linux devices based on the analysis of Section 5.2.5. `Dark_embedded_linux_1` detects network packets of 5 TCP ports: 22/TCP, 23/TCP, 8080/TCP, 32764/TCP, 58455/TCP. This signature has a fixed IP header's ID value, TCP sequence number, and TCP source port. The initial TTL value is set to 64.

5.5 Evaluation

In this section, we evaluate the effectiveness of the proposed method by analyzing the correlation between macro and micro analysis of network packets created by each malware and network scanning tool. We also evaluated false positives and false negatives when we applied the method to real network traffic.

5.5.1 Signatures of Morto

We analyzed network traffic observed in the darknet and extracted network packets that matched `Morto_scan` and `Morto_scan_NAT`. The network traffic we used is explained in Section 5.2 We analyzed traffic during the period around the emergence of the malware Morto (from July 1 to August 9, 2011) and examined the increase of the number of network packets that matched the signatures. Figure 5.3

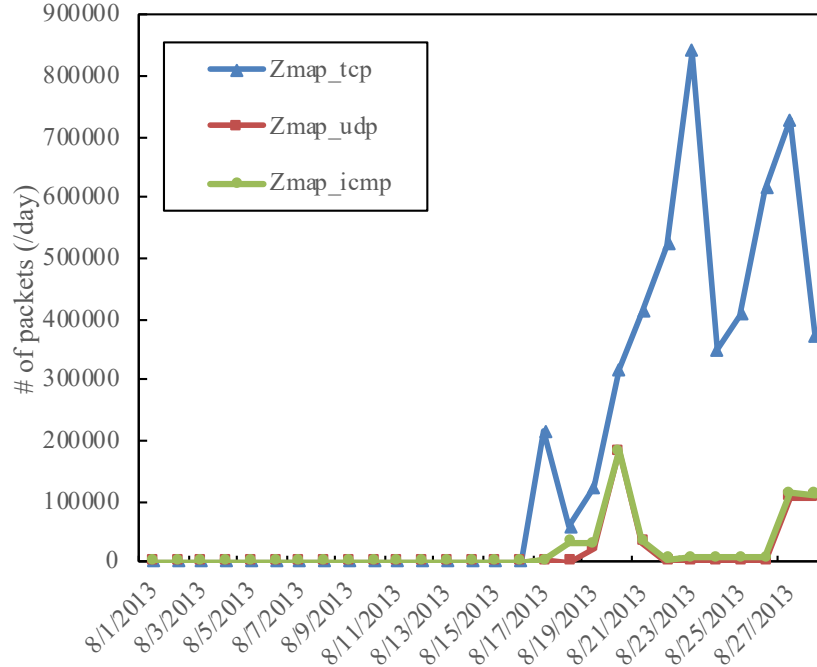


Figure 5.4: Network packets observed at the time of publishing ZMap and detected by signatures of ZMap.

shows the number of source IP addresses of detected network packets per day. We observed network packets detected by `Morto_scan` and `Morto_scan_NAT`, which have occurred from July 30. This was just before the time that security vendors issued alerts of malware `Morto` in late August. We also observed network packets that have similar characteristics of those created by malware `Morto`. The packets were shown in this figure as the `Pattern_A` and `Pattern_B`. A large number of `Pattern_A`'s packets were observed from July 12 to 27, 2011, but as of 2016, a small number of packets have been observed in the darknet. Network packets of `Pattern_B` were only observed in between July 20 and August 1, 2011. These packets could be network packets for attacker's reconnaissance or those created by prototype implementation of malware `Morto`. In summary, we extracted network packets of malware `Morto` using our method and signatures, and revealed that these packets had occurred and increased prior to the security vendors' alert.

5.5.2 Signatures of ZMap

We extracted network packets that matched signatures of ZMap from the darknet traffic around the time this tool was released (August 2013). Figure 5.4 shows the number of packets that matched `Zmap_tcp`, `Zmap_udp`, `Zmap_icmp`. After the tool was published on github on August 11, 2013, the packets began to be observed on the darknet from August 15, 2013. Almost all packets were sent from several IP addresses, which were owned by University of Michigan (the developer of this tool) and Rapid7, which is an organization that uses the tool. Therefore, our method

and signatures correctly identified network packets that created by ZMap.

5.6 Measurement Study

We performed a measurement study of analyzing malicious traffic in the wild by using proposed method. We analyzed network traffic observed in honeypots that we operate. The honeypots were used for observing DRDoS attacks by simulating servers using UDP, such as DNS and NTP. The purpose of the honeypots is that attackers use them as reflector servers of DRDoS attacks. The technical specifications were given in our previous study [85].

5.6.1 Malware used for DRDoS Attacks

To identify network packets related to DRDoS attacks that originated from Malware, we analyzed DNS traffic that our honeypots observed between January 1 and April 30, 2015. By using the two signatures `Iptables_dns` and `Xor_dns`, we observed no network packets from our honeypots. The reasons for this is that our honeypots the two types of malware were not used for DRDoS attacks in the wild; or the network packets created by the these types of malware did not reach our honeypots.

5.6.2 Malicious Packets Targeting Linux Devices

We analyzed the malicious network packets targeting Linux devices to understand what kind of attacks they were associated with. The malicious network packets were observed in the darknet as described in Section 5.2.5 and can be identified by the signature `Dark_embedded_linux_1`. We used our honeypots [97], which are developed for observing attacks targeting Linux-based IoT devices. Our honeypots simulate Telnet (23/TCP) servers to capture attackers' login attempts. We extracted network packets that matched the signature and analyzed sessions with the same IP addresses as those of the network packets observed on the same day. A session is defined as a series of network packets starting with a client's TCP SYN packet and ending with a TCP FIN packet.

Figure 5.5 shows the number of the sessions and the number of source IP addresses of the network packets observed between February 22 and October 19, 2015. There were login attempts with easy passwords such as "root/admin" in the sessions. We used `p0f` to identify clients' operating systems and found that all network packets were sent from Linux 2.x or Linux 3.x kernel. Then, we tried to manually access the source IP addresses of them by using a web browser. Some IP addresses returned login web pages of routers and network cameras. This result indicates that routers and network cameras were controlled by any types of malware and tried to infect other vulnerable devices on the Internet.

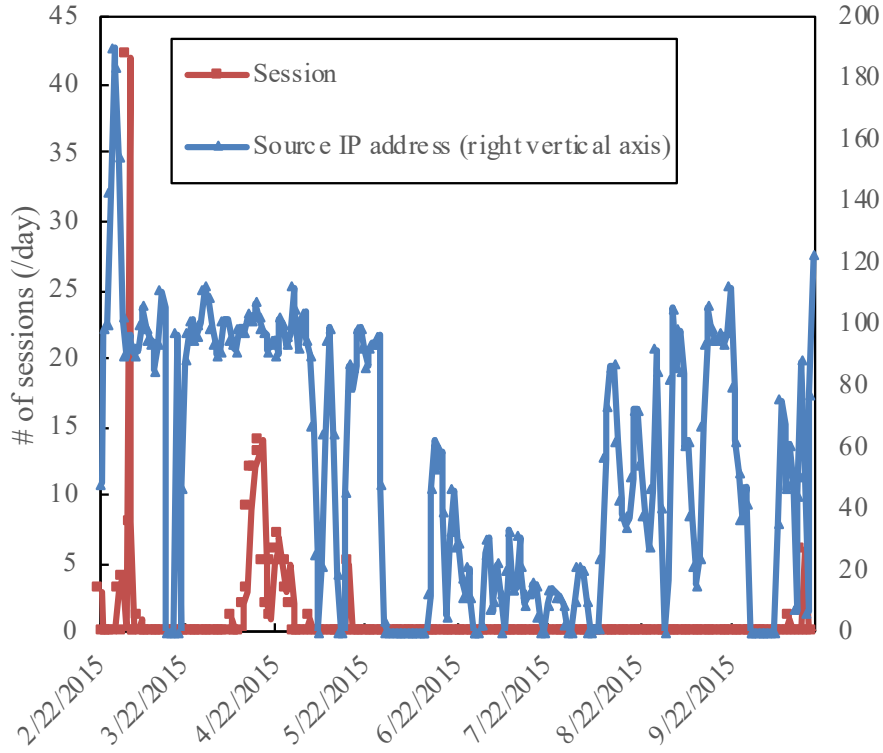


Figure 5.5: Network packets detected by Dark_embedded_linux_1

5.7 Implementation of the Proposed Method on a Network Observation System

In this section, we explain our implementation of the proposed method on a large-scale network observation System, NICTER. NICTER has multiple components, which enable macro and micro analysis such as the malware dynamic analysis and the statistical analysis of the network traffic. Our implementation uses network packets observed in the darknet as the input and automatically detect malicious and reconnaissance ones using signatures. We show the effectiveness of this implementation by analyzing network scanning packets related to DRDoS attacks.

5.7.1 Implementation

Our implementation first receives network packet data in real time from the observation server of the darknet traffic. Then, this implementation detects network packets using our signatures and stores results into a database system. The detection results are stored as one record per network packet in the MySQL database. The record has eight fields: the source IP address, the destination IP address, the source port number, the destination port number, the name of the signature, the version of the signature, the ID of the darknet censor, and packet ID. The signatures will be updated when the implementation accesses our signature server and download the latest signatures. The latest signatures are also available on the

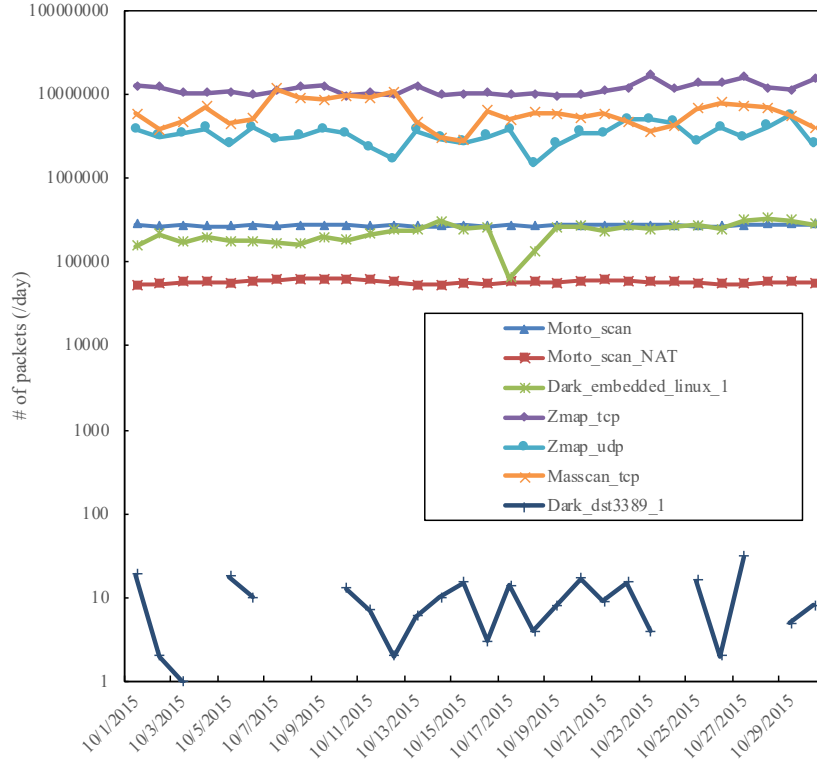


Figure 5.6: Network packets detected in NICTER

website [98]

5.7.2 Result of detecting network packets

Figure 5.6 shows the result of detecting network packets observed in the NICTER between October 1 and October 30, 2015. Network packets were detected by our seven signatures during this period. The most common packets were those detected by signatures of ZMap and Masscan, which are related to network scanning. We also network packets created by Malware created by Morto and those targeting Linux devices. In summary, by using our method and signatures, we can distinguish a large number of network packets created by malware and network scanning tools that implements their own network stack from large-scale darknet traffic.

Then we analyzed network packets detected by the signature ZMap_udp to reveal the network scans to protocols that are often abused in DRDoS attacks. In our study [85], we have shown that most attackers abused nine protocols for DRDoS attacks: QOTD (17/UDP), CharGen (19/UDP), DNS (53/UDP), NTP (123/UDP), NetBIOS (137/UDP), SNMP (161/UDP), SSDP(1900/UDP), 27015/UDP, and 27960/UDP Figures 5.7a-5.7i are stacked bar charts that show the number of UDP packets observed in the darknet. The numbers of network packets to the nine protocols were increasing in the period of analysis. Network packets sent by Zmap were also increasing from March 2014. Also, they occupied most of observed

network packets to some protocols, such as DNS and 27960/UDP. Now, we define a scanner as a source IP address that sends network packets to 64 or more IP addresses in the darknet. We found that 12,653 scanners send network packets and 10.1% (1,296 scanners) of them were IP addresses owned by universities, security vendors, or network scanning projects such as Shodan and Shadowserver. Out of all scanners, those using ZMap are 727 scanners (5.7%). We found that 507 scanners, whose IP addresses belonged to universities, security vendors, and network scanning projects, used ZMap. Therefore, network scanning for finding reflectors of DRDoS attacks was performed not only by attackers and malware, but also by hosts for research and investigative purposes.

5.7.3 Providing information on infected devices

We discuss alerting users and ISPs with information on malicious devices, which behave abnormally, such as network scanning. The results in Section 5.4.1 and 5.4.4 showed that some devices infected with malware performed network scanning. We are continually observing such malicious activities in the darknet and finding a large number of malicious devices. This information will be useful for users that have infected devices and are unaware of the infection. By collaborating with ISPs, we can alert users and provide information on what malware their devices are infected with. Our future work is to alert users using the implementation on NICTER and provide users with a guide to remove malware according to their infection status.

5.8 Summary

In this chapter, we proposed a method for identifying network packets created by malware and network tools that implements their own network stacks. We evaluated the method, which uses our original signatures, by analyzing correlation of micro and macro analysis such as malware dynamic analysis, reverse engineering, observation of the darknet, and operating honeypots. We reported a measurement study that analyzed malicious and reconnaissance activities on the Internet by using the method and signatures. Our future work is to implement our method to a large-scale observation system and to notify ISPs and organizations (e.g., CSIRT) of alerts in real time.

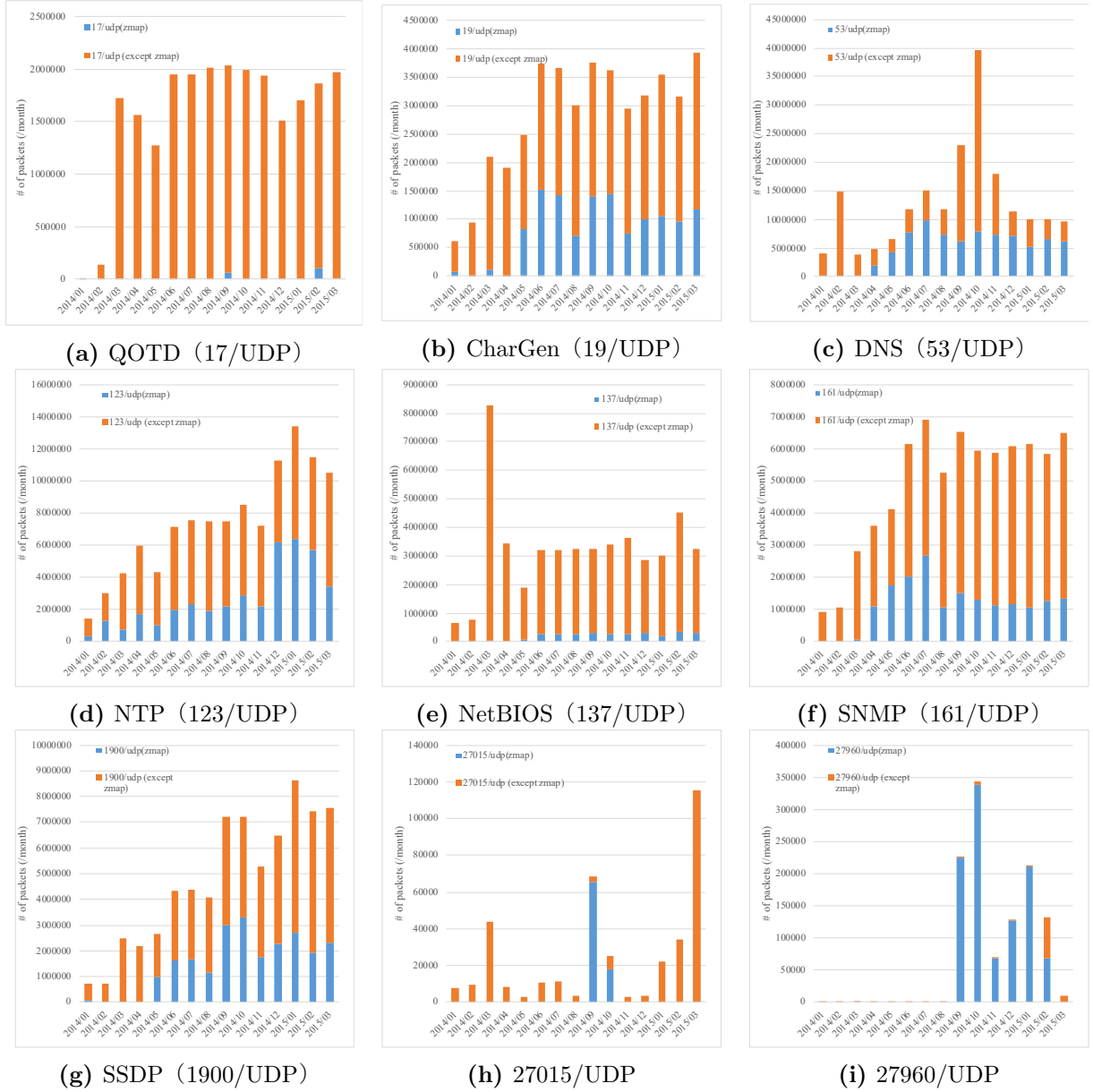


Figure 5.7: UDP packets whose ports are abused for DRDoS attacks. Blue bars indicate the number of packets sent from ZMap and orange bars indicate the number of packets sent from other sources

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In recent years, cyber attacks have become more diverse due to the increasing complexity of communication devices and the web. Existing methods for protecting devices target specific attacks so that they will not be able to capture new attacks in the future. It is important to estimate the attackers' purposes and follow attack vectors to prevent possible future attacks. In this thesis, we focused on methods for collecting and detecting cyber attacks on the Internet by combining passive and active observation. One of the contributions of this research is to present robust approaches to changes in devices and attack trends. In addition, our measurement studies using the proposed approaches have discovered attackers' modus operandi and purposes, which can be useful to improve future countermeasures.

In chapter 3, we proposed a system for crawling and identifying multi-step social engineering (SE) attacks on the web. SE attacks leverage a sequence of web pages to manipulate users to lead them to attackers' purposes. Since previous studies have analyzed only the surface of the attacks, they have failed to capture those attacks lurking behind multiple web pages. Our system actively follows the sequences of web pages by automating a web browser and detects such SE attacks. We used our system for a large scale measurement and discovered the psychological tactics of attackers to trick users.

In chapter 4, we proposed a system for detecting new distribution channels of fake antivirus software. We have defined fake removal information advertisement (FRAD) sites as malicious web pages that introduce fake removal information for cyber threats to trick users to install fake antivirus software. To shed light on the pervasiveness of FRAD sites and their ecosystem, we performed a comprehensive analysis of both passively and actively collected data. We showed that FRAD sites occupy search results when users search for cyber threats, thus preventing the users from obtaining the correct information.

In chapter 5, we proposed a method for detecting network packets created by malware and network scanning tools. Some malware and network scanning tools use their network stack to generate network packets faster than operating systems. Using signatures we created on the basis of the active analysis of malware and tools, we applied our method to a large-scale network. We revealed that there

was a large amount of network scanning activities using some network scanning tools and reconnaissance activities by attackers to find vulnerable devices on the Internet.

6.2 Future Work

This thesis presented methods for passively and actively observing cyber attacks on the Internet. We showed that we can analyze attackers' purposes and trends by using our methods to collect the cyber attacks, which were difficult to capture with existing methods. Our experimental results, which focused on cyber attacks against many and unspecified users and devices, will be a stepping stone to passive and active observation of all attacks, including attacks targeting specific groups and individuals. It also helps prevent possible future attacks. Our methods should be applied to large systems to continuously observe and analyze cyber attacks at multiple locations. Also, we should focus on designing and developing more robust systems for new attacks based on our findings.

Acknowledgments

First, I would like to express my deepest appreciation to my supervisor, Professor Tsutomu Matsumoto, for his tremendous supports, comments and encouragements during the course of my study. His insights into my research and his valuable suggestions greatly inspired me in my research pursuits. I am extremely grateful to Associate Professor Katsunari Yoshioka, who taught me how to conduct research and write technical papers. Without his help I would not have been able to complete my dissertation. I also gratefully acknowledge Professor Junji Shikata whose insightful comments and advice significantly contributed to improving my dissertation. I would like to thank Professors Tatsunori Mori and Dr. Shinichi Shirakawa for serving on my dissertation committee. Their valuable comments and feedback were extremely helpful.

My sincere thanks also go to Dr. Akiyama Mitsuaki and Dr. Daiki Chiba and all my colleagues at NTT Secure Platform Laboratories for fruitful discussions, insightful comments, and the beneficial working environment. I am also indebted to members of the Matsumoto Laboratory, especially Ms. Mio Narimatsu and Ms. Tomoko Ishidate. Finally, I would like to thank my family for their support and sacrifices.

List of Publications

Reviewed Papers in Journals

- **Takashi Koide**, Shogo Suzuki, Daisuke Makita, Kosuke Murakami, Takahiro Kasama, Mio Suzuki, Jumpei Shimamura, Masashi Eto, Daisuke Inoue, Koji Nakao, Katsunari Yoshioka, and Tsutomu Matsumoto, “Detection Method for Malicious Packets with Characteristic Network Protocol Header,” Journal of Information Processing, Vol. 57, No. 9, pp. 1986-2002, 2016. (in Japanese).
小出 駿, 鈴木 将吾, 牧田 大佑, 村上 洸介, 笠間 貴弘, 鈴木 未央, 島村 隼平, 衛藤 将史, 井上 大介, 中尾 康二, 吉岡 克成, 松本 勉, “通信プロトコルのヘッダの特徴に基づく不正通信の検知手法,” 情報処理学会論文誌, Vol. 57, No. 9, pp. 1986-2002, 2016.
- **Takashi Koide**, Daiki Chiba, Mitsuaki Akiyama, Katsunari Yoshioka, and Tsutomu Matsumoto, “To Get Lost is to Learn the Way: An Analysis of Multi-step Social Engineering Attacks on the Web,” IEICE Transactions on Information and Systems, Vol.E104-A, No.1, pp.162-181, 2021.
- **Takashi Koide**, Daiki Chiba, Mitsuaki Akiyama, Katsunari Yoshioka, and Tsutomu Matsumoto, “Understanding the Fake Removal Information Advertisement Sites,” Journal of Information Processing, Vol. 29, 2021.

Reviewed Papers in International Conference Proceedings

- **Takashi Koide**, Daiki Chiba, Mitsuaki Akiyama, “To get lost is to learn the way: Automatically collecting multi-step social engineering attacks on the web,” the 15th ACM Asia Conference on Computer and Communications Security, ASIA CCS ’ 20, 2020.
- **Takashi Koide**, Daiki Chiba, Mitsuaki Akiyama, Katsunari Yoshioka, and Tsutomu Matsumoto, “It Never Rains but It Pours: Analyzing and Detecting Fake Removal Information Advertisement Sites,” The 17th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, DIMVA 2020, 2020. [**Best Paper Award**]

Poster in International Conference

- **Takashi Koide**, Daisuke Makita, Katsunari Yoshioka, Tsutomu Matsumoto, “Observation and Analysis of TCP-based Reflection DDoS Attacks Using Honeypot”, The 18th International Symposium on Research in Attacks, Intrusions and Defenses (RAID2015), Poster session, 2015.

Technical Reports

- 小出駿, 鈴木将吾, 牧田大佑, 村上洸介, 笠間貴弘, 島村隼平, 衛藤将史, 井上大介, 吉岡克成, 松本勉, “通信プロトコルのヘッダの特徴に基づく不正通信の検知・分類手法”, 情報処理学会, コンピュータセキュリティシンポジウム 2014 (CSS2014) 論文集, pp. 48-55, 2014. [CSS 学生論文賞]
- 田辺瑠偉, 筒見拓也, 小出駿, 牧田大佑, 吉岡克成, 松本勉, “Linux 上で動作するマルウェアを安全に観測可能なマルウェア動的解析手法の提案”, 情報処理学会, コンピュータセキュリティシンポジウム 2014 (CSS2014) 論文集, pp. 1007-1014, 2014.
- 鈴木将吾, 小出駿, 牧田大佑, 村上洸介, 笠間貴弘, 島村隼平, 衛藤将史, 吉岡克成, 松本勉, 井上大介, “複数国ダークネット観測による攻撃の局地性分析”, 情報処理学会, コンピュータセキュリティシンポジウム 2014 (CSS2014) 論文集, pp. 40-47, 2014.
- 村上洸介, 蒲谷武正, 千賀渉, 鈴木将吾, 小出駿, 島村隼平, 牧田大佑, 笠間貴弘, 衛藤将史, 吉岡克成, 井上大介, 中尾康二, “複数のダークネット観測拠点で同時期に急増する攻撃を検知する手法の提案”, 情報処理学会, コンピュータセキュリティシンポジウム 2014 (CSS2014) 論文集, pp. 32-39, 2014.
- 牧田大佑, 西添友美, 小出駿, 筒見拓也, 金井文宏, 森博志, 吉岡克成, 松本勉, 井上大介, 中尾康二, “早期対応を目的とした統合型 DRDoS 攻撃観測システムの構築”, 電子情報通信学会, 暗号と情報セキュリティシンポジウム 2015(SICS2015) 予稿集, 2A3-1, 2015.
- 小出駿, 牧田大佑, 吉岡克成, 松本勉, “TCP リフレクション攻撃の観測と分析”, 情報処理学会, コンピュータセキュリティシンポジウム 2015 (CSS2015) 論文集, pp. 923-930, 2015.
- 小出駿, 牧田大佑, 笠間貴弘, 鈴木未央, 井上大介, 中尾康二, 吉岡克成, 松本勉, “通信プロトコルのヘッダの特徴に基づくパケット検知ツール tkiwa の実装と NICTER への導入”, 電子情報通信学会, 情報通信システムセキュリティ研究会 (ICSS), 信学技報, vol. 115, no. 334, ICSS2015-38, pp. 19-24, 2015.
- 柴原健一, 筒見拓也, 小出駿, 森博志, 村上洸介, 中尾康二, 吉岡克成, 松本勉, “DRDoS 攻撃を観測可能なダークネットを用いたリフレクタの分析”, 電子情報通信学会, 暗号と情報セキュリティシンポジウム 2016(SICS2016) 予稿集, 1B1-5, 2016.
- 小出駿, 千葉大紀, 高田雄太, 秋山満昭, 八木毅, 波戸邦夫, “ユーザ操作が起点となる Web 上の攻撃の収集”, 電子情報通信学会, 情報通信システムセキュリティ研究会 (ICSS), 信学技報, vol. 117, no. 481, ICSS2017-66, pp. 91-96, 2018. [情報通信システムセキュリティ研究賞]

- 源平祐太, 中川 雄太, 高田一樹, 小出駿, 金井文宏, 秋山満昭, 田辺瑠偉, 吉岡克成, 松本勉, “悪性 Web サイトに到達しやすい危険検索単語の検知,” 情報処理学会, コンピュータセキュリティシンポジウム 2019 (CSS2019) 論文集, pp. 1390–1397, 2019. [CSS 優秀論文賞]

References

- [1] Cisco, “Snort - Network Intrusion Detection & Prevention System.” <https://www.snort.org/>, 2020.
- [2] M. Sharif, J. Urakawa, N. Christin, A. Kubota, and A. Yamada, “Predicting impending exposure to malicious content from user behavior,” the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, 2018.
- [3] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric, “All things considered: An analysis of iot devices on home networks,” 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019, ed. N. Heninger and P. Traynor, pp.1169–1185, USENIX Association, 2019.
- [4] K. Fukuda, J.S. Heidemann, and A. Qadeer, “Detecting malicious activity with DNS backscatter over time,” *IEEE/ACM Trans. Netw.*, vol.25, no.5, pp.3203–3218, 2017.
- [5] S. Torabi, A. Boukhtouta, C. Assi, and M. Debbabi, “Detecting internet abuse by analyzing passive DNS traffic: A survey of implemented systems,” *IEEE Commun. Surv. Tutorials*, vol.20, no.4, pp.3389–3415, 2018.
- [6] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee, “DNS noise: Measuring the pervasiveness of disposable domains in modern DNS traffic,” 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014, pp.598–609, IEEE Computer Society, 2014.
- [7] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, “Exposure: A passive DNS analysis service to detect and report malicious domains,” *ACM Trans. Inf. Syst. Secur.*, vol.16, no.4, pp.14:1–14:28, 2014.
- [8] K. Kisamori, A. Shimoda, T. Mori, and S. Goto, “Analysis of malicious traffic based on tcp fingerprinting,” vol.52, no.6, pp.2009–2018, 2011. (in Japanese).
- [9] R. Yamada and S. Goto, “Using abnormal ttl values to detect malicious ip packets,” the Asia-Pacific Advanced Network (APAN), pp.27–34, 2013.

- [10] M. Cova, C. Krügel, and G. Vigna, “Detection and analysis of drive-by-download attacks and malicious javascript code,” the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, pp.281–290, ACM, 2010.
- [11] L. Invernizzi and P.M. Comparetti, “Evilseed: A guided approach to finding malicious web pages,” IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA, pp.428–442, IEEE Computer Society, 2012.
- [12] A. Kharraz, W.K. Robertson, and E. Kirda, “Surveyance: Automatically detecting online survey scams,” 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA, pp.70–86, 2018.
- [13] M.Z. Rafique, T. van Goethem, W. Joosen, C. Huygens, and N. Nikiforakis, “It’s free for a reason: Exploring the ecosystem of free live streaming services,” 23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016, pp.21–24, The Internet Society, 2016.
- [14] B. Srinivasan, A. Kountouras, N. Miramirkhani, M. Alam, N. Nikiforakis, M. Antonakakis, and M. Ahamad, “Exposing search and advertisement abuse tactics and infrastructure of technical support scammers,” the Web Conference, WWW 2018, Lyon, France, April 23-27, 2018, 2018.
- [15] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, “Understanding malvertising through ad-injecting browser extensions,” Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, ed. A. Gangemi, S. Leonardi, and A. Panconesi, pp.1286–1295, ACM, 2015.
- [16] K. Tian, S.T.K. Jan, H. Hu, D. Yao, and G. Wang, “Needle in a haystack: Tracking down elite phishing domains in the wild,” Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018, pp.429–442, ACM, 2018.
- [17] The Honeynet Project, “Glastopf – The Honeynet Project.” <https://www.honeynet.org/projects/old/glastopf/>, 2020.
- [18] T. Yagi, N. Tanimoto, T. Hariu, and M. Itoh, “Design of provider-provisioned website protection scheme against malware distribution,” IEICE Trans. Commun., vol.93-B, no.5, pp.1122–1130, 2010.
- [19] J. Krupp, M. Backes, and C. Rossow, “Identifying the scan and attack infrastructures behind amplification ddos attacks,” Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, ed. E.R. Weippl, S. Katzenbeisser, C. Kruegel, A.C. Myers, and S. Halevi, pp.1426–1437, ACM, 2016.

- [20] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, “Exit from hell? reducing the impact of amplification ddos attacks,” Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014, ed. K. Fu and J. Jung, pp.111–125, USENIX Association, 2014.
- [21] A. Niakanlahiji, J. Wei, M.R. Alam, Q. Wang, and B. Chu, “Shadowmove: A stealthy lateral movement strategy,” 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, ed. S. Capkun and F. Roesner, pp.559–576, USENIX Association, 2020.
- [22] M.A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao, “The nocebo effect on the web: An analysis of fake anti-virus distribution,” 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET ’10, 2010.
- [23] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, “Webwitness: Investigating, categorizing, and mitigating malware download paths,” 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015., pp.1025–1040, USENIX Association, 2015.
- [24] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, “Towards measuring and mitigating social engineering software download attacks,” 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., pp.773–789, USENIX Association, 2016.
- [25] N. Miramirkhani, O. Starov, and N. Nikiforakis, “Dial one for scam: A large-scale analysis of technical support scams,” 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26-March 1, 2017, 2017.
- [26] C.J. Dietrich, C. Rossow, and N. Pohlmann, “Exploiting visual appearance to cluster and detect rogue software,” the 28th Annual ACM Symposium on Applied Computing, SAC ’13, Coimbra, Portugal, March 18-22, 2013, pp.1776–1783, ACM, 2013.
- [27] G. Stringhini, C. Kruegel, and G. Vigna, “Shady paths: leveraging surfing crowds to detect malicious web pages,” 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013, pp.133–144, 2013.
- [28] H. Mekky, R. Torres, Z. Zhang, S. Saha, and A. Nucci, “Detecting malicious HTTP redirections using trees of user browsing activity,” 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014, pp.1159–1167, 2014.
- [29] T. Taylor, X. Hu, T. Wang, J. Jang, M.P. Stoecklin, F. Monroe, and R. Sailer, “Detecting malicious exploit kits using tree-based similarity searches,” Proceedings of the Sixth ACM on Conference on Data and Application Security

and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016, pp.255–266, 2016.

- [30] P. Vadrevu, J. Liu, B. Li, B. Rahbarinia, K.H. Lee, and R. Perdisci, “Enabling reconstruction of attacks on users via efficient browsing snapshots,” 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26-March 1, 2017, 2017.
- [31] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, “Hulk: Eliciting malicious behavior in browser extensions,” the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014., pp.641–654, USENIX Association, 2014.
- [32] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, “Understanding malvertising through ad-injecting browser extensions,” the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, pp.1286–1295, ACM, 2015.
- [33] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M.A. Rajab, “Ad injection at scale: Assessing deceptive advertisement modifications,” 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015, pp.151–167, IEEE Computer Society, 2015.
- [34] Internet Archive, “Heritrix.” <https://github.com/internetarchive/heritrix3>, 2019.
- [35] Selenium Developers Group, “Selenium.” <https://www.seleniumhq.org/>, 2019.
- [36] S. Duman, K. Onarlioglu, A.O. Ulusoy, W.K. Robertson, and E. Kirda, “Trueclick: automatically distinguishing trick banners from genuine download links,” the 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, December 8-12, 2014, pp.456–465, ACM, 2014.
- [37] L. Lu, R. Perdisci, and W. Lee, “SURF: detecting and measuring search poisoning,” the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011, pp.467–476, ACM, 2011.
- [38] H. Yang, X. Ma, K. Du, Z. Li, H. Duan, X. Su, G. Liu, Z. Geng, and J. Wu, “How to learn klingon without a dictionary: Detection and measurement of black keywords used by the underground economy,” 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, pp.751–769, IEEE Computer Society, 2017.
- [39] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B.Y. Zhao, “Detecting and characterizing social spam campaigns,” the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010, pp.35–47, ACM, 2010.

- [40] S. Lee and J. Kim, “Warningbird: Detecting suspicious urls in twitter stream,” 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, The Internet Society, 2012.
- [41] N. Nikiforakis, F. Maggi, G. Stringhini, M.Z. Rafique, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, and S. Zanero, “Stranger danger: exploring the ecosystem of ad-based URL shortening services,” 23rd International World Wide Web Conference, WWW ’14, Seoul, Republic of Korea, April 7-11, 2014, pp.51–62, ACM, 2014.
- [42] “Tesseract Open Source OCR Engine.” <https://github.com/tesseract-ocr/tesseract>, 2019.
- [43] P.F. Alcantarilla, J. Nuevo, and A. Bartoli, “Fast explicit diffusion for accelerated features in nonlinear scale spaces,” British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9-13, 2013, 2013.
- [44] “Doc2vec paragraph embeddings.” <https://radimrehurek.com/gensim/models/doc2vec.html>, 2019.
- [45] Symantec, “DeepSight Intelligence.” <https://www.symantec.com/services/cyber-security-services/deepsight-intelligence>, 2019.
- [46] Malwarebytes, “hpHosts.” <http://www.hosts-file.net/>, 2019.
- [47] K. Thomas, J.A.E. Crespo, R. Rasti, J.M. Picod, C. Phillips, M. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M. Courteau, L. Ballard, R. Shield, N. Jagpal, M.A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, “Investigating commercial pay-per-install and the distribution of unwanted software,” 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., pp.721–739, USENIX Association, 2016.
- [48] P. Kotzias, L. Bilge, and J. Caballero, “Measuring PUP prevalence and PUP distribution through pay-per-install services,” 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., pp.739–756, USENIX Association, 2016.
- [49] “Microsoft Cognitive Services Bing Search Engine APIs.” <https://azure.microsoft.com/en-us/services/cognitive-services/search/>, 2020.
- [50] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, “Avclass: A tool for massive malware labeling,” Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings, pp.230–253, 2016.
- [51] P. Arntz, “Stolen security logos used to falsely endorse pups.” <https://blog.malwarebytes.com/threat-analysis/social-engineering-threat-analysis/2018/01/stolen-security-logos-used-to-falsely-endorse-pups/>, 2018.

- [52] “Web of Trust.” <https://www.mywot.com/en/scorecard/etnamedia.net>, 2019.
- [53] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, “Knowing your enemy: understanding and detecting malicious web advertising,” the ACM Conference on Computer and Communications Security, CCS’12, Raleigh, NC, USA, October 16-18, 2012, pp.674–686, ACM, 2012.
- [54] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, “The dark alleys of madison avenue: Understanding malicious advertisements,” the 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, November 5-7, 2014, pp.373–380, ACM, 2014.
- [55] “hosts-blocklists.” <https://github.com/nottracking/hosts-blocklists>, 2019.
- [56] D. Dittrich and E. Kenneally, “The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research,” tech. rep., U.S. Department of Homeland Security, 2012.
- [57] MarketWatch, Inc., “Global Antivirus Software Market Report 2019 and Future Opportunity Assessment 2024.” <https://www.marketwatch.com/press-release/global-antivirus-software-market-report-2019-and-future-opportunity-assessment-2024-2019-09-30>, 2019.
- [58] M. Cova, C. Leita, O. Thonnard, A.D. Keromytis, and M. Dacier, “An analysis of rogue AV campaigns,” Recent Advances in Intrusion Detection, RAID 2010, 2010.
- [59] C. Grier, L. Ballard, J. Caballero, N. Chachra, C.J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, N. Provos, M.Z. Rafique, M.A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, and G.M. Voelker, “Manufacturing compromise: the emergence of exploit-as-a-service,” the ACM Conference on Computer and Communications Security, CCS’12, pp.821–832, 2012.
- [60] B. Stone-Gross, R. Abman, R.A. Kemmerer, C. Kruegel, and D.G. Steigerwald, “The underground economy of fake antivirus software,” 10th Annual Workshop on the Economics of Information Security, WEIS 2011, 2011.
- [61] Malwarebytes Corporation, “2020 State of Malware Report.” https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf, 2020.
- [62] E. Stein, “Hong Kong Based Malvertiser Brokers Traffic To Fake Antivirus Scams.” <https://blog.confiant.com/hong-kong-based-malvertiser-brokers-traffic-to-fake-antivirus-scams-over-100-million-ads-300e251eff06>, 2019.

- [63] P. Vadrevu and R. Perdisci, “What you see is NOT what you get: Discovering and tracking social engineering attack campaigns,” *Proceedings of the Internet Measurement Conference, IMC 2019*, pp.308–321, 2019.
- [64] K. Thomas, J.A.E. Crespo, R. Rasti, J.M. Picod, C. Phillips, M. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M. Courteau, L. Ballard, R. Shield, N. Jagpal, M.A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, “Investigating commercial pay-per-install and the distribution of unwanted software,” *25th USENIX Security Symposium, USENIX Security 16*, pp.721–739, 2016.
- [65] A. Kharraz, W.K. Robertson, and E. Kirda, “Surveylance: Automatically detecting online survey scams,” *IEEE Symposium on Security and Privacy, SP 2018*, 2018.
- [66] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad, “Towards measuring and mitigating social engineering software download attacks,” *25th USENIX Security Symposium, USENIX Security 16*, pp.773–789, 2016.
- [67] T. Vissers, W. Joosen, and N. Nikiforakis, “Parking sensors: Analyzing and detecting parked domains,” *Network and Distributed System Security Symposium, NDSS*, 2015.
- [68] M.Z. Rafique, T. van Goethem, W. Joosen, C. Huygens, and N. Nikiforakis, “It’s free for a reason: Exploring the ecosystem of free live streaming services,” *23rd Annual Network and Distributed System Security Symposium, NDSS*, 2016.
- [69] N. Miramirkhani, O. Starov, and N. Nikiforakis, “Dial one for scam: A large-scale analysis of technical support scams,” *24th Annual Network and Distributed System Security Symposium, NDSS*, 2017.
- [70] B. Srinivasan, A. Kountouras, N. Miramirkhani, M. Alam, N. Nikiforakis, M. Antonakakis, and M. Ahamad, “Exposing search and advertisement abuse tactics and infrastructure of technical support scammers,” *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*, pp.319–328, 2018.
- [71] MaxMind, “GeoIP2 Databases.” <https://www.maxmind.com/en/geoip2-databases>, 2020.
- [72] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, “Measuring pay-per-install: The commoditization of malware distribution,” *USENIX Security Symposium*, 2011.
- [73] L. Invernizzi and P.M. Comparetti, “Evilseed: A guided approach to finding malicious web pages,” *IEEE Symposium on Security and Privacy, SP 2012*, 2012.

- [74] J.P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi, “deseo: Combating search-result poisoning,” 20th USENIX Security Symposium, 2011.
- [75] Y. Kwon, B. Saltaformaggio, I.L. Kim, K.H. Lee, X. Zhang, and D. Xu, “Self destructing exploit executions via input perturbation,” 24th Annual Network and Distributed System Security Symposium, NDSS, 2017.
- [76] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, “Knowing your enemy: understanding and detecting malicious web advertising,” the ACM Conference on Computer and Communications Security, CCS’12, pp.674–686, 2012.
- [77] L. Lu, R. Perdisci, and W. Lee, “SURF: detecting and measuring search poisoning,” Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, pp.467–476, 2011.
- [78] H. Mekky, R. Torres, Z. Zhang, S. Saha, and A. Nucci, “Detecting malicious HTTP redirections using trees of user browsing activity,” 2014 IEEE Conference on Computer Communications, INFOCOM 2014, pp.1159–1167, 2014.
- [79] N. Nikiforakis, F. Maggi, G. Stringhini, M.Z. Rafique, W. Joosen, C. Kruegel, F. Piessens, G. Vigna, and S. Zanero, “Stranger danger: exploring the ecosystem of ad-based URL shortening services,” World Wide Web Conference, WWW, 2014.
- [80] Y. Shen, E. Mariconti, P. Vervier, and G. Stringhini, “Tiresias: Predicting security events through deep learning,” Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp.592–605, 2018.
- [81] P. Vadrevu, J. Liu, B. Li, B. Rahbarinia, K.H. Lee, and R. Perdisci, “Enabling reconstruction of attacks on users via efficient browsing snapshots,” 24th Annual Network and Distributed System Security Symposium, NDSS 2017, 2017.
- [82] M. Zhang, W. Meng, S. Lee, B. Lee, and X. Xing, “All your clicks belong to me: Investigating click interception on the web,” 28th USENIX Security Symposium, USENIX Security 2019, pp.941–957, 2019.
- [83] J. Zhang, C. Yang, Z. Xu, and G. Gu, “Poisonamplifier: A guided approach of discovering compromised websites through reversing search poisoning attacks,” Research in Attacks, Intrusions, and Defenses, RAID 2012, 2012.
- [84] M. Zalewski, “p0f v3.” <https://lcamtuf.coredump.cx/p0f3/>, 2020.
- [85] Z. Durumeric, M. Bailey, and J.A. Halderman, “An internet-wide view of internet-wide scanning,” Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014, ed. K. Fu and J. Jung, pp.65–78, USENIX Association, 2014.
- [86] M. Zalewski, “The ZMap Project.” <https://zmap.io/>, 2020.

- [87] N.I. of Information and C. Technology, “NICTERWEB.” <https://www.nictcr.jp/>, 2020.
- [88] Microsoft, “Microsoft: Encyclopedia entry: Worm: Win32/Morto.A, Malware Protection Center.” <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Worm:Win32/Morto.a>, 2017.
- [89] “Worm:W32/Morto.A Description — F-Secure Labs.” https://www.f-secure.com/v-descs/worm_w32_morto_a.shtml, 2020.
- [90] K. Yoshioka, K. Murakami, and T. Matsumoto, “Network scan method and its automated signature generation for detecting malware infected hosts,” vol.51, no.9, pp.1633–1644, 2010. (in Japanese).
- [91] R. Graham, “MASSCAN: Mass IP port scanner.” <https://github.com/robertdavidgraham/masscan>, 2020.
- [92] Z. Durumeric, M. Bailey, and J.A. Halderman, “An internet-wide view of internet-wide scanning,” Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014, ed. K. Fu and J. Jung, pp.65–78, USENIX Association, 2014.
- [93] “IPTABLES AND IPTABLEX DDOS BOTS THREAT ADVISORY.” <https://www.stateoftheinternet.com/resources-web-security-threat-advisories-2014-iptables-iptablex-linux-bots-botnet.html>, 2014.
- [94] “XOR DDOS [HIGH RISK].” <https://www.stateoftheinternet.com/resources-web-security-threat-advisories-2015-xor-ddos-attacks-linux-botnet-malware-removal-ddos-mitigation-yara-snort.html>, 2015.
- [95] S. Gallagher, “Backdoor in wireless DSL routers lets attacker reset router, get admin, arstechnica.” <http://arstechnica.com/security/2014/01/backdoor-in-wireless-dsl-routers-lets-attacker-reset-router-get-admin/>, 2014.
- [96] Trend Micro, “Worm.Linux.DARLLOZ.AA.” <https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/worm.linux.darlloz.aa>, 2019.
- [97] Y.M.P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “Iotpot: Analysing the rise of iot compromises,” 9th USENIX Workshop on Offensive Technologies, WOOT ’15, Washington, DC, USA, August 10-11, 2015, ed. A. Francillon and T. Ptacek, USENIX Association, 2015.
- [98] “Packet Detection Tool tkiwa.” <http://ipsr.ynu.ac.jp/tkiwa/index.html>, 2020.