2017

Data-Driven Network-Centric Threat Assessment

Dae Wook Kim
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Repository Citation
Kim, Dae Wook, "Data-Driven Network-Centric Threat Assessment" (2017). Browse all Theses and Dissertations. 1729.
https://corescholar.libraries.wright.edu/etd_all/1729

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact corescholar@wwwlibraries.wright.edu.
DATA-DRIVEN NETWORK-CENTRIC THREAT ASSESSMENT

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

by

DAE WOOK KIM
M.S., Syracuse University, 2007
B.S., Michigan State University, 2003

2017
WRIGHT STATE UNIVERSITY
I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Dae Wook Kim ENTITLED Data-Driven Network-Centric Threat Assessment BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Junjie Zhang, Ph.D.
Dissertation Director

Michael L. Raymer, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Robert E. W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School

Committee on Final Examination

Junjie Zhang, Ph.D.

Adam Robert Bryant, Ph.D.

Bin Wang, Ph.D.

Xuetao Wei, Ph.D.
ABSTRACT


As the Internet has grown increasingly popular as a communication and information sharing platform, it has given rise to two major types of Internet security threats related to two primary entities: end-users and network services. First, information leakages from networks can reveal sensitive information about end-users. Second, end-users systems can be compromised through attacks on network services, such as scanning-and-exploit attacks, spamming, drive-by downloads, and fake anti-virus software. Designing threat assessments to detect these threats is, therefore, of great importance, and a number of the detection systems have been proposed. However, these existing threat assessment systems face significant challenges in terms of i) behavioral diversity, ii) data heterogeneity, and iii) large data volume.

To address the challenges of the two major threat types, this dissertation offers three unique contributions. First, we built a new system to identify network users via Domain Name System (DNS) traffic, which is one of the most important behavior-based tracking methods for addressing privacy threats. The goal of our system is to boost the effectiveness of existing user identification systems by designing effective fingerprint patterns based on semantically limited DNS queries that are missed by existing tracking efforts.

Second, we built a novel system to detect fake anti-virus (AV) attacks, which represent an active trend in the distribution of Internet-based malware. Our system aims to boost the effectiveness of existing fake AV attack detection by detecting fake AV attacks in three challenging scenarios: i) fake AV webpages that require user interaction to install malware, instead of using malicious content to run automatic exploitation without users consent (e.g., shellcode); ii) fake AV webpages designed to impersonate real webpages using a few representative elements, such as the names and icons of anti-virus products from
authentic anti-virus webpages; and iii) fake AV webpages that offer up-to-date solutions (e.g., product versions and threat names) to emerging threats.

Finally, we built a novel system to detect malicious online social network (OSN) accounts that participate in online promotion events. The goal of our work is to boost the effectiveness of existing detection methods, such as spammer detection and fraud detection. To achieve our goal, our framework that systematically integrates features that characterize malicious OSN accounts based on three of their characteristics: their general behaviors, their recharging patterns, and their currency usage, and then leverages statistical classifier for detection.
Contents

1 Chapter 1: Introduction .......................................................... 1
  1.1 Background ........................................................................... 1
  1.2 Objectives and Challenges ................................................. 3
  1.3 Overview of Solutions ...................................................... 5
  1.4 Organization .................................................................... 7

2 Chapter 2: Detecting Users’ Identities in DNS Traffic ................. 9
  2.1 Motivation .......................................................................... 9
  2.2 Related Work ..................................................................... 11
  2.3 System .............................................................................. 13
    2.3.1 Problem Formulation .................................................. 14
    2.3.2 Patterns ..................................................................... 17
    2.3.3 System Implementation .............................................. 21
  2.4 Evaluation .......................................................................... 22
    2.4.1 Experimental Setup .................................................... 22
    2.4.2 Experimental Results .................................................. 24
  2.5 Discussion ......................................................................... 30
  2.6 Summary .......................................................................... 31

3 Chapter 3: Detecting Fake Anti-Virus Webpages ....................... 33
  3.1 Motivation .......................................................................... 33
  3.2 Related Work ..................................................................... 36
  3.3 System .............................................................................. 38
    3.3.1 Collecting & Labeling Data ......................................... 39
    3.3.2 Discovering Trendy and Diverse Security Keywords ...... 41
    3.3.3 Designing Detection Features ..................................... 42
  3.4 Evaluation .......................................................................... 53
    3.4.1 Experimental Setup .................................................... 53
    3.4.2 Experimental Results .................................................. 53
  3.5 Discussion ......................................................................... 59
  3.6 Summary .......................................................................... 61
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Chapter 4: Detecting Malicious Accounts for Online Promotions in Social Network</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>4.1 Motivation</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>4.2 Related Work</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>4.2.1 Spammer Detection</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>4.2.2 Fraud Detection</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>4.2.3 Limitations</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>4.3 System</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.3.1 Collecting &amp; Labeling Data</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.3.2 System Design</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>4.4 Evaluation</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>4.4.1 Experimental Setup</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>4.4.2 Experimental Results</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>4.5 Discussion</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>4.6 Summary</td>
<td>85</td>
</tr>
<tr>
<td>5</td>
<td>Chapter 5: Conclusion and Future Work</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>5.1 Conclusion</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>5.2 Future Work</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>5.3 Closing Remarks</td>
<td>90</td>
</tr>
</tbody>
</table>

Bibliography                                      91
List of Figures

2.1 *DNSMiner* architecture; Reprinted from [39] ............................ 13
2.2 *DNSMiner* implementation of identifying persistent patterns; Reprinted from [39] ................................................................. 21
2.3 The CDF distribution of the maximum $\text{supp}(\cdot)$ for all persistent IP addresses in $D_1$ (i.e., all IPs in $P_1$). A significant percentage (64.18%) of IPs in $P_1$ have patterns that are persistent across 7 epochs; Reprinted from [39] .... 23
2.4 The CDF distribution of the highest contrast confidence for each IP address that has at least one persistent pattern. Approximately 70% have unique persistent patterns (i.e., with contrast confidence of 1); Reprinted from [39] 26
2.5 The CDF distribution of the number of fingerprint patterns for each IP address. IP addresses with fingerprint patterns tend to have a large number of fingerprint patterns; Reprinted from [39] ....................... 26
2.6 The trend of detection rates when $\alpha$ increases given a fixed value for $\beta$, where *DNSMiner* achieves the best accuracy of 98.74% when $\alpha = 5/7$ and $\beta = 60\%$; Reprinted from [39] .................................................. 28

3.1 An example screenshot of a fake AV webpage; Reprinted from [38] ...... 34
3.2 Example of Fake AV Webpage Attack [79] ................................. 35
3.3 System Architecture of *DART*; Reprinted from [38] ................... 38
3.4 Examples of icons from authentic AV software; Reprinted from [38] ....... 44
3.5 Image similarity between an authentic McAfee image (Left) and fake one (Right); Reprinted from [38] ................................. 44
3.6 Image Identity Feature; Reprinted from [38] ............................... 46
3.7 Domain Identity Feature; Reprinted from [38] ............................. 47
3.8 Content Identity Feature; Reprinted from [38] ............................. 47
3.9 Path Keyword Feature; Reprinted from [38] .............................. 48
3.10 Content Keyword Feature; Reprinted from [38] .......................... 48
3.11 Redirection Feature; Reprinted from [38] ................................. 52
3.12 Maliciousness Score Feature; Reprinted from [38] ........................ 52
3.13 ROC comparison between light-weight and all features; Reprinted from [38] 54
3.14 Correlation matrix of detection features; Reprinted from [38] .......... 57
3.15 Detection features factor map: Projection of coordinates of 7 features on the PCA variable factor map; Reprinted from [38] ................. 59
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The integration of OSN accounts and financial accounts; Reprinted from [92]</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>Virtual Currency Flow for Malicious OSN Accounts; Reprinted from [92]</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>The Architectural Overview of the System; Reprinted from [92]</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>Feature 1 - The Ratio of Active Days; Reprinted from [92]</td>
<td>70</td>
</tr>
<tr>
<td>4.5</td>
<td>Feature 2 - The Number of Friends for An Account; Reprinted from [92]</td>
<td>71</td>
</tr>
<tr>
<td>4.6</td>
<td>Feature 3 - The Number of Services Purchased By An Account; Reprinted from [92]</td>
<td>72</td>
</tr>
<tr>
<td>4.7</td>
<td>Feature 4 - The Average Recharge Amount; Reprinted from [92]</td>
<td>73</td>
</tr>
<tr>
<td>4.8</td>
<td>Feature 5 - The Percentage of Recharge from Promotion Activities; Reprinted from [92]</td>
<td>74</td>
</tr>
<tr>
<td>4.9</td>
<td>Feature 6 - The Total Amount of Expenditure; Reprinted from [92]</td>
<td>75</td>
</tr>
<tr>
<td>4.10</td>
<td>Feature 7 - The Percentage of Expenditure from Banks; Reprinted from [92]</td>
<td>76</td>
</tr>
<tr>
<td>4.11</td>
<td>Feature 8 - The Percentage of Expenditure as Gifts; Reprinted from [92]</td>
<td>77</td>
</tr>
<tr>
<td>4.12</td>
<td>ROC curve on the proposed 8 features; Reprinted from [92]</td>
<td>79</td>
</tr>
<tr>
<td>4.13</td>
<td>Upper triangular matrix; Reprinted from [92]</td>
<td>82</td>
</tr>
<tr>
<td>4.14</td>
<td>The variables factor map (PCA); Reprinted from [92]</td>
<td>83</td>
</tr>
</tbody>
</table>
## List of Tables

2.1 Transactions and their associated domains for two users across 5 epochs, where \{a,b\} becomes the domain set fingerprint pattern for \(u_1\); Reprinted from [39] .................................................. 18

2.2 Window-Aware Patterns; Reprinted from [39] .................................................. 20

2.3 A sequence of DNS queries; Reprinted from [39] .................................................. 20

2.4 The # of IPs in \(D_1\) and \(D_2\), # of persistent IPs (\(|P_1|\) and \(|P_2|\)), # of IPs with persistent patterns in \(P_1\) and \(P_2\), and # of IPs with fingerprint patterns in \(D_1\) (i.e., \(|FP_1|\)); Reprinted from [39] .................................................. 23

2.5 The accuracy of identifying users in a new DNS stream \(D_2\) using fingerprint patterns extracted from a historical DNS stream \(D_1\). Among 69.63% IPs that are identified by fingerprint patterns, 98.74% are correctly revealed; Reprinted from [39] .................................................. 28

2.6 The detection performance under different \(\alpha\) and \(\beta\) values. “PI” indicates the percentage of IPs in \(P_1\) that have fingerprint patterns; “II” is denoted as the percentage of IPs in \(FP_1 \cap P_2\) that are detected by fingerprint patterns; “DR” and “FP” refer to the detection rate and false positive rate, respectively; Reprinted from [39] .................................................. 28

2.7 The detection performance of \textit{DNSMiner} for each category of patterns: “Domain name” refers to the domain set pattern; “Inter-domain relationship” includes the domain sequence pattern and window-aware domain sequence pattern; “Temporal behavior” contains period and hourly behavior pattern; Reprinted from [39] .................................................. 30

3.1 Examples of security keywords discovered in each step; Reprinted from [38] 41

3.2 Domain examples for authentic and fake anti-virus landing pages; Reprinted from [38] .................................................. 46

3.3 False positive rates for Security-popular and Fake AV webpages; Reprinted from [38] .................................................. 55

3.4 AUC for each classifier; Reprinted from [38] .................................................. 55

3.5 A percentage of average computation time for each of light-weight features; Reprinted from [38] .................................................. 56
3.6 Rank of feature importance in *DART*; variable importance is measured as the mean decrease in accuracy and features with large variable importance are more influential for classification of the data; Reprinted from [38] . . . 56

4.1 AUCs for Three Classifiers; Reprinted from [92] . . . . . . . . . . . . . . . 80

4.2 Feature importance rank of *ProGuard* by Random Forest; Reprinted from [92] 81
ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor, Prof. Junjie Zhang, for all his encouragement and support of my research and thesis in the Cybersecurity Lab at Wright State University. Without his advice and insightful feedback, this PhD would not have been achievable. Prof. Zhang has been a great advisor. Thanks to his professional training that has truly benefited me, his constant scientific guidance, and my learning from his research expertise, I have grown as a researcher scientist in my field. Prof. Zhang was and remains my best role model for a research scientist, mentor, and instructor for my future career.

I would also like to thank Prof. Adam R. Bryant, Prof. Bin Wang, and Prof. Xue-tao Wei for serving as my dissertation committee members. Their brilliant comments, insightful suggestions, and warm encouragement helped me overcome the weaknesses in my dissertation and improve its quality. My thanks also go out to the support I received from the collaborative work by Prof. Yadong Zhou, Lili Liu, and Peiying Yan to complete my dissertation. I am grateful to Yu Li, Jacob H. Saunders, Ademola A. Ikusan, Soham Kelkar, David R. Sutherin, Zach Sisco, and Xiaotian Wang in the Cybersecurity Lab for their friendship and assistance throughout my dissertation work.

Finally, I want to thank my family, including my parents, Won Ju Kim and Jin Ock Kim, for their endless encouragement and constant support. I would especially like to thank my beloved wife and son, Eun Hee Ko and Joseph J. Kim. Without their love and support throughout this work, my thesis could not have been completed.
To Eun Hee Ko and Joseph J. Kim,
for their endless love and support.
Chapter 1: Introduction

1.1 Background

The rise and popularity of the Internet as an essential part of life has ensured that Internet services play a significant role in peoples lives [30]. According to prior surveys [73], as of 2016, the total number of Internet users worldwide is 3.5 billion, including approximately 680 million Internet users in China and 282 million in the USA. However, the rapid growth of the social phenomenon of Internet usage has produced significant threats to Internet security. Specifically, two primary entities end-users and network services are at risk of cyber threats.

End-users, who are also referred to as regular Internet users, typically face two types of major threats to Internet security. First, information leakages from the Internet threaten the privacy of end-users. Specifically, increasing numbers of applications, such as Internet measurements, traffic engineering, and network-based intrusion detection, require network traffic to be preserved and analyzed. As a consequence, network users sensitive information could be leaked. Second, end-users systems might be compromised by attackers. Attackers frequently exploit software vulnerabilities to compromise end-users systems using a variety of methods. These methods could include scanning-and-exploit attacks [55], spamming [25], drive-by downloads [16], and fake anti-virus software [27].

Network services are Internet services that rely on secure infrastructures to operate. An increasing number of network services rely specifically on social networks; in these
cases, attackers may create accounts, perform malicious activities, and finally damage the security of the infrastructure. For example, Tencent QQ [63], a leading Chinese OSN, supports its own virtual currency (the Q coin) through an online financial service that links an online banking account with a virtual currency account. Attackers might compromise QQ accounts by participating in online promotion events and collecting the virtual currency. Subsequently, they can use spam methods to launder the virtual currency to other users.

Assessing such threats is crucially important. However, threat assessments face three significant challenges. First, attackers exhibit a wide diversity of behaviors, making it difficult to design static rules for threat assessment. For example, the attackers (i.e., malicious OSN users) are likely to show three categories of distinct behaviors within OSN dataset. These attackers behaviors cover general behavior (e.g., chatting, photo sharing), virtual-currency collection (e.g., online banking, promotion), and virtual-currency usage (e.g., online shopping, gifting). Intrinsically, each of these behaviors has different characteristics, resulting in the need for robust and seamless threat assessment methods.

Second, addressing data heterogeneity and assessing multiple factors simultaneously is of utmost importance. For example, fake anti-virus (AV) webpages inherently comprise heterogeneous types of data based on user-perception content (e.g., images), search engine optimization (e.g., URL keywords), and networking infrastructures (e.g., HTTP protocol). Characterizing these data effectively requires the use of a variety of threat assessment techniques.

Third, the volume of data involved is huge; thus, the methods used must be scalable. For instance, data collected from the DNS resolver tend to include large volumes of DNS traces. It is difficult to handle these large data using traditional data manipulation or statistical methods on single-host infrastructures. Instead, big data platforms (e.g., Hadoop) must be applied to process and analyze large-scale DNS datasets for high scalability.
1.2 Objectives and Challenges

In this dissertation, we address three important and challenging problems. First, we seek to assess the extent to which users’ fingerprints can be leaked through DNS traffic. Second, we wish to design an automated method capable of detecting fake AV distribution webpages. Third, we design a system to identify malicious accounts used for online promotion behaviors in social network.

First, we are interested in the feasibility of DNS-based behavioral fingerprinting techniques that rely on users’ DNS activities. Since DNS traces can be leaked and used to derive fingerprints that characterize end-users’ DNS activities, attackers can use these traces to deanonymize the users in DNS traffic. This is one of the most significant privacy threats associated with end-users. However, in order to identify users of DNS datasets accurately, it is necessary to derive effective DNS fingerprint patterns. To tackle this problem and design DNS-based fingerprint systems, we must address three specific challenges. First, DNS traces use few semantics and do not include explicit identifiers (i.e., only DNS queries are visible), making it difficult to design effective fingerprint patterns. Second, users’ DNS activities are diversified, thus increasing the challenge of distinguishing among users. Third, the extremely large volume of DNS queries leads to problems related to the scalability of the systems needed to handle the massive DNS datasets. Although some existing methods [46, 11, 85, 84, 61] to address these problems have been proposed, they are not sufficient to fully address these challenges.

Second, fake anti-virus (AV) attacks represent a new and active trend in the distribution of malware to end-users. According to a Google case study [65], fake AV attacks are responsible for 50% of all malware delivered via Internet advertising, and their prevalence is growing. Such attacks, therefore, represent a serious threat to end-users. With respect to phishing threat assessments, the development of fake AV webpage detection systems is an important challenge. To address this challenge, we develop a new automatic system
that can generate detection features based on the properties of fake AV webpages and use statistical classifiers to discriminate fake AV webpages from legitimate ones. To achieve high levels of detection performance, the main challenge is to design systematic and robust features using the heterogeneous data extracted from the fake AV webpages. To address this challenge, we propose detection features characterized by the three aspects that are most important for successful fake AV attacks. Some existing fake AV detection systems [16, 19, 49, 66] are easily evaded by attackers due to the nature of fake AV webpages (i.e., data heterogeneity), and their detection abilities are significantly constrained, resulting in an inability to effectively detect fake AV attacks.

Third, we focus on malicious OSN accounts that attackers create to launch malicious attacks, such as spamming or phishing, to worsen the protection of OSN network services. Specifically, malicious OSN accounts participating in online promotions represent a novel threat to OSN network services. The attackers who control malicious OSN accounts can illegally transfer virtual currencies collected as rewards from promotion events to other users as gifts. The mostly important problem in addressing this threat is detecting these malicious OSN accounts, which can enable network services to mitigate malicious activities. A number of detection systems [25, 12, 86, 1, 81, 77] have been proposed to address such threats as OSN spamming or fraudulent activities; however, these systems have been largely constrained by two major challenges that make it difficult for existing methods to detect malicious OSN accounts: First, these malicious OSN accounts do not rely on any malicious content or network infrastructure (e.g., phishing URLs, exploits), and second, no social interaction among OSN accounts is required. As a result, it is difficult to apply existing methods to detect malicious OSN accounts.

To summarize, existing threat assessments may fail to detect malicious threats and identify privacy leaks in terms of behavioral diversity, data heterogeneity, and data volume as a result of three practical challenges. Since the effectiveness of these threat assessments is significantly limited due to an insufficient ability to detect new attacks, novel threat
assessments based on networks are required to address emerging challenges.

1.3 Overview of Solutions

In order to address the challenges of behavioral diversity, data heterogeneity, and data volume, we build two threat assessment systems: an automatic and scalable system to re-identify network users in DNS traffic and an automated and effective system to detect fake AV webpages on the Internet. Finally, we build a detection system to identify malicious OSN accounts that participate in online promotion events.

First, we design a system [39] to re-identify end-users in a passively collected DNS dataset, which represents a threat to users privacy. The goal of our system is to boost the effectiveness (e.g., detection power) and scalability (e.g., time computation) of existing user re-identification systems by designing effective patterns based on semantically limited DNS queries that are missed by existing efforts. To accomplish this goal, our system adopts five features to systematically characterize users diverse behaviors along three different identifiers: their domain names, their inter-domain relationships, and their temporal behaviors. In addition, we use the MapReduce platform to efficiently process a large volume of DNS traces (467 million DNS queries). Experimental results based on a massive DNS dataset demonstrate that our system can identify 69.63% of users who have behavioral fingerprints in the historical DNS stream and who experience persistent DNS activities in the new DNS stream. Among these identifiable users, we find that our system can achieve a high accuracy of 98.74% and a low false positive rate of 1.26%.

Second, we design a system [38] to detect fake AV software distribution webpages, which represent a malicious threat related to end-users compromised systems. The goal of our system is to boost the effectiveness of existing fake AV methods by detecting fake AV attacks in three challenging scenarios: i) fake AV webpages that require users interaction to install malware, instead of using malicious content to run automatic exploitations
without users consent (e.g., shellcode); ii) fake AV webpages designed to impersonate webpages through a few representative elements, such as the names and icons of anti-virus products from authentic anti-virus webpages; and iii) fake AV webpages that provide up-to-date solutions (e.g., product versions and threat names) to emerging threats. To fulfill our goals, our system first collects fake AV webpages, which are labeled by public domain reputation services (e.g., MalewareDomainList) and malware detection engine-based manual inspection services (e.g., VirusTotal), to draw executable(s) from each webpage. Our system further extracts three categories of detection features, including human-perception features, SEO features, and networking features, to characterize the intrinsic elements of fake AV webpages. Based on the features extracted from heterogeneous fake AV data, our system then distinguishes fake AV webpages from legitimate ones using statistical classifiers. Experimental results based on a real dataset of fake AV webpages demonstrates that our system can achieve a high detection rate (90.4%) and a very low false positive rate (0.2%).

Finally, we build a novel system [92] to detect malicious OSN accounts that participate in online promotion events, which represent a major threat in terms of network services. Since online social networks integrate financial capabilities by enabling the use of both real and virtual currency, they serve as new platforms for hosting a variety of business activities, such as online promotion events, in which users can collect virtual currency as a reward for participating. Both OSNs and business partners are concerned about attackers using a set of accounts to collect virtual currency from these events, thus reducing the events’ effectiveness and causing significant financial loss. Therefore, it is important to proactively detect malicious OSN accounts prior to online promotion activities and to subsequently decrease their reward priority. The goal of our work is to boost the effectiveness of existing detection methods, such as spammer detection and fraud detection. To achieve this goal, we propose a new framework for systematically integrating the features that characterize OSN accounts from three perspectives: general behaviors, recharging patterns, and cur-
rency usage. To demonstrate the detection accuracy of our system, we perform extensive experiments based on a real OSN dataset (the Tencent QQ dataset) with built-in financial management activities and then ProGuard can accomplish a high detection rate of 96.67% at very extremely low false positive of 0.3%.

1.4 Organization

The rest of the dissertation is organized as follows:

In Chapter 2, we discuss the design of a new system, DNSMiner, to distinguish network users from DNS traffic. We present a survey of existing DNS-based behavioral tracking techniques and compare them to our system. We then elaborate on the system design of DNSMiner, including its fingerprint pattern mining and matching approaches to reveal the presence of specific users within large-scale DNS traffic. The experimental evaluation is also discussed in terms of its detection rate and its false positive rate.

In Chapter 3, we discuss the design of our detection system, DART, to detect fake AV attacks. Existing methods for fake AV attacks are presented, and we introduce ways to collect and label fake AV webpages. The proposed features and classification algorithms used to distinguish fake AV webpages from legitimate ones are discussed. We subsequently present our evaluation results.

In Chapter 4, we discuss ProGuard, a novel system to detect malicious accounts in OSN-based online promotion activities. We provide a statement for a new problem and discuss why the problem should be solved and also describe related work and limitations. In addition, we explain our ProGuard architecture including data labelling and the detection features design based on three aspects of malicious OSN accounts. The evaluation on real-world OSN dataset is presented, followed by the detection performance and feature importance and their correlation.

In Chapter 5, we summarize our contributions and discuss potential future work, as
well as closing remarks.
Chapter 2: Detecting Users’ Identities in DNS Traffic

2.1 Motivation

The analysis of Domain Name System (DNS) data plays an important role in a variety of applications such as traffic engineering [69, 80], malicious domain detection [33, 6, 8, 5], and cover-channel analysis [62]. While the demand for DNS traces is generally increasing, the collected DNS traces may introduce significant privacy concerns. For example, DNS queries that are triggered by the prefetching mechanisms of popular browsers can leak users’ search engine queries [42]; DNS queries can also reveal the types of operating systems [53]. Therefore, we study a new type of privacy risk introduced by DNS traces: to which extent network users can be uniquely identified merely based on the way they issue DNS queries? We accomplish this objective by designing behavioral fingerprints, where a DNS-based fingerprint aims to uniquely identify its corresponding user in DNS traffic and stay immune to the change of time.

DNS-based fingerprints, once successfully derived, have strong privacy implications. For example, they can be used to de-anonymize the DNS traces with anonymized sources. To be more specific, when DNS traces are shared, the source (e.g., the IP address) that issues the DNS query is usually anonymized (e.g., by obscuring the IP address using...
hash functions). However, one can learn behavioral fingerprints from un-anonymized DNS traces and use the acquired fingerprints to reveal the presence of specific users in (other) anonymized traces. In addition, if one can get access to DNS traces collected from multiple access networks (e.g., through open DNS services or collecting traces from multiple networks), he/she can track users’ locations across different networks by using behavioral fingerprints to reveal users in DNS traces.

In this chapter, we aim at investigating the extent to which behavioral fingerprints can be derived and measuring their accuracy on identifying the presence of corresponding network users. As a means towards this end, we have proposed a set of new patterns, which collectively form behavioral fingerprints. We also built a distributed, scalable system, namely DNSMiner, to automatically derive DNS-based behavioral fingerprints from a massive amount of DNS traces. Specifically, we make the following contributions.

- We have designed five new patterns including domain set, domain sequence, window-aware domain sequence, period behavior, and hourly behavior, which collectively form behavioral fingerprints. These patterns systematically characterize DNS behaviors from three aspects including the domain name, the inter-domain relationship, and the temporal behavior. Although more patterns might be discovered to enhance behavioral fingerprints, our proposed patterns serve as a lower bound of the capabilities to use DNS behaviors to fingerprint network users.

- We have built a system, namely DNSMiner, to automatically mine behavioral fingerprints from a massive amount of DNS traces. The design of the system leverages the MapReduce distributed infrastructure to scale up the system performance. After being deployed in a 15-nodes Hadoop platform, DNSMiner can process more than 467 million DNS queries using approximately 4 hours.

- We have performed extensive evaluation based on a large volume of DNS queries collected from a large campus network across two weeks.
The experimental results demonstrated that the behavioral fingerprints derived from a historical DNS stream can effectively identify users in a new DNS stream. To be more specific, 69.63% of users, who have behavioral fingerprints in the historical DNS stream and experience persistent DNS activities in the new DNS stream, can be identified using their behavioral fingerprints. Among these identifiable users, our system accomplishes a high accuracy of 98.74% and a low false positive rate of 1.26%.

2.2 Related Work

Information leakage through collected network data has been recognized as a significant privacy concern, thereby attracting a lot of research efforts. A rich body of literature [75, 47, 11, 85, 84, 89] have been proposed to infer application-level users’ activities from (encrypted) network traffic. Chen et al. [11] have leveraged communication patterns of HTTP connections to infer the activities taken by browser users. In [85, 84], Wright et al. have built statistical models to reveal languages and even spoken phases from encrypted VoIP traffic. Zhang et al. [89] designed a hierarchical classification system to identify users’ online activities (i.e., a user’s running applications) based on network-level traffic patterns. Sun et al. [75] also created traffic signatures to reveal webpages visited by users in encrypted network traffic. Different from these works that focus on inferring users’ activities, our work targets at inferring users’ identities.

Pang [61] et al. generated user fingerprints based on encrypted wireless traffic patterns. However, compared to deriving user fingerprints from wireless traffic, fingerprinting users based on DNS traffic is faced with unique challenges since DNS traffic has less semantics. Particularly, although encrypted, the wireless traffic can expose the set of SSIDs, packet sizes, and MAC protocol fields used by a user. Comparatively, DNS queries only make visible the domain name and the timestamp if the source IP is anonymized. There-
fore, how to design effective patterns based on semantic-limited DNS queries becomes the key of our solution. The work closest to ours is [32], where Herrmann et al designed a learning-based approach to attribute sessions of DNS queries to their corresponding users. However, our work significantly differs from the method proposed in [32] from two perspectives. First, a single feature, the visiting frequency of popular domains for each host, was adopted in [32] to characterize users’ behaviors while we designed multi-faceted features (i.e., total 5 features) to systematically characterize users’ behaviors from three different perspectives. Second, the method [32] needs to separate a DNS stream into sessions according to the timestamp of DNS queries, which implies the necessity for fine-grained timing information for DNS queries. Despite the fact that our current implementation also used timestamp for DNS queries, the first pattern (i.e., the domain set pattern) is time-independent; the second and third patterns (i.e., the domain sequence and window-aware domain sequence patterns) only concern the order in which DNS queries are issued in each day. This implies that our method can be used in DNS streams with coarse-grained timing information. In fact, the domain sequence and window-aware domain sequence patterns collaboratively accomplished a high detection rate of 90.72% in our experiment. A few projects [42, 53] investigated information leakage from the same type of network traffic used by our work - the passively collected DNS packets. However, their objectives are different from ours. To be specific, Krishnan et al. [42] aimed at recovering search engine queries by investigating correlated domain names and Matsunaka et al. [53] intended to fingerprint operating systems rather than network users. Several methods [15, 82, 57] have been proposed to de-anonymize network data. Specifically, Coull et al. [15] has proposed techniques to de-anonymize network flows by comparing the objects from the unanonymized and anonymized network data directly. Narayanan et al. [82] and Wondracek et al. [57] have leveraged the topology of an unanonymized social network to effectively identify users in an anonymized social network. Despite the fact that our method leverages different data sources, we do not need auxiliary information (e.g.,
the context of the anonymized data and additional topologies of unanonymized social networks). Nevertheless, DNS behavioral fingerprints extracted by our method complement existing methods [61, 15, 82, 57].

2.3 System

The architectural overview of DNSMiner is presented in Fig. 2.1. DNSMiner takes as input a set of DNS-query streams, which is denoted as \( S = \{S_1, S_2, \ldots, S_N\} \). Each stream (e.g., \( S_i \)) contains DNS queries issued by a user (e.g., \( u_i \)) over a certain time period (e.g., several days). A stream is a series of tuples, where each tuple is denoted as \( <u, \text{domain}, \text{timestamp}> \). \( u, \text{domain}, \) and \( \text{timestamp} \) refer to the user identity, domain name, and the querying time, respectively because DNS queries are in a good position for deriving fingerprint patterns in [32] and our DNS-query streams form is suitable for reflect the behaviors of users DNS activities. In a network where an IP address can be associated with a user, we can use IP addresses to represent users’ identities. DNSMiner aims at generating a DNS-based behavioral fingerprint, namely \( F_i \), for a user \( u_i \), where \( F_i \) is defined as a finite set of patterns (i.e., \( F_i = \{F_i^1, F_i^2, \ldots, F_i^K\} \)). Each pattern in the fingerprint is named as a fingerprint pattern. Ideally, fingerprint patterns should be i) unique to their corresponding user (i.e., persistent to their corresponding users) and ii) immune to the change of time.

To illustrate the detailed design of DNSMiner, we first formulate the mining process.
of fingerprints (see Section 2.3.1). Next, we will discuss specific patterns used by DNS-Miner and the motivations behind their design (see Section 2.3.2). Finally, we briefly describe the implementation of DNSMiner that takes advantage of MapReduce [22] to achieve high scalability (see Section 2.3.3).

2.3.1 Problem Formulation

2.3.1.1 Pattern Mining

DNSMiner aims at mining fingerprint patterns that exhibit both significant persistence and uniqueness to a user. Towards this end, we start from defining persistence and uniqueness of a fingerprint pattern. DNSMiner aggregates the DNS stream from a user (e.g., $u_i$) into a set of transactions (denoted as $T_i = \{T_i^1, T_i^2, \ldots, T_i^M\}$), where each transaction $T_i^k$ is a set of tuples issued by $u_i$ within the same epoch. Since Internet activities usually exhibit strong diurnal patterns [68], we currently use one day to represent an epoch. We denote “$T_i^k$ satisfies $F$” if the pattern $F$ is observed in $T_i^k$. The specific meaning of “satisfy” varies for different patterns and we will illustrate it along with the introduction of the patterns. For instance, if $F$ is a set of domains, then $T_i^k$ satisfies $F$ when all domains in $F$ are contained in the set of domains that are extracted from all tuples in $T_i^k$. We introduce a function $mt(F, T_i)$ that returns all transactions in $T_i$ that satisfy $F$. Specifically, $mt(F, T_i)$ is defined as

$$mt(F, T_i) = \{T_i^k \in T_i \mid T_i^k \text{satisfies } F\}$$

(2.1)

We subsequently define a function $supp(F, T_i)$ to quantify the persistence of a pattern (i.e., $F$) across the transactions generated by a user $u_i$. Its formal definition is presented as

$$supp(F, T_i) = \frac{|mt(F, T_i)|}{|T_i|}$$

(2.2)
The $\text{supp}(F, T_i)$ characterizes two trends. If a pattern $F$ is persistent to $u_i$, $\text{supp}(F, T_i)$
tends to be large. In contrast, a transient pattern is inclined to yield small $\text{supp}()$ value. We
use a pre-defined threshold, namely $\alpha$, to discriminate between persistent patterns and tran-
sient ones. Specifically, $F$ is considered to be persistent to $u_i$ if $\text{supp}(F, T_i) \geq \alpha$. We de-
note the set of persistent patterns for a user $u_i$ as $P(T_i)$, where $P(T_i) = \{ F | \text{supp}(F, T_i) \geq \alpha \}$.

However, the high persistence of a pattern does not guarantee its uniqueness since a
persistent pattern for $u_i$ could also be a persistent pattern for another user. We therefore
define another metric, namely contrast confidence, to quantify uniqueness of a persistent
pattern (e.g., $F$) for a user $u_i$ (i.e., how well it $F$ can differentiate $u_i$ from other users).

$$\text{conf}(F, T_i) = \frac{\text{supp}(F, T_i)}{\sum_{F \in P(T_j)} \text{supp}(F, T_j)}, \text{ where } F \in P(T_i) \quad (2.3)$$

$\text{conf}(F, T_i)$ characterizes the following trends: if a pattern is persistent to many users,
then its contrast confidence tends to be low; otherwise, its contrast confidence tends to
be high. Again, a threshold $\beta$ is introduced in our current design to differentiate these
two trends. A persistent pattern $F$ will be considered as a fingerprint pattern for $u_i$ if
$\text{conf}(F, T_i) \geq \beta$.

### 2.3.1.2 Pattern Matching

Given an unknown user $u_u$ and his/her associated DNS stream, the pattern matching phase
of $\text{DNSMiner}$ aims at identifying whether this DNS stream can be attributed to any known
user. To this end, $\text{DNSMiner}$ will first follow the same method discussed in Section 2.3.1.1
to obtain persistent patterns for $u_u$. Specifically, we will derive a set of DNS transac-
tions (denoted as $T_u$) for the unknown user $u_u$ and subsequently identify persistent patterns
$P(T_u)$. It is worth noting that the same criteria for epoch representation (e.g., 24 hours)
and the same value of $\alpha$ will be applied. Next, we will evaluate the similarity between an unknown user $u_u$ and a known user $u_i$, whose fingerprint is denoted as $\mathcal{F}_i$. A distance function, denoted as $\text{dist}(u_u, u_i)$, is consequently defined as

$$
\text{dist}(u_u, u_i) = 1 - \frac{\sum \text{conf}(F^k_i, \mathcal{T}_i)}{\sum \text{conf}(F^j_i, \mathcal{T}_i)},
$$

where $F^k_i \in P(\mathcal{T}_u) \cap \mathcal{F}_i$ and $F^j_i \in \mathcal{F}_i$.

$\sum \text{conf}(F^k_i, \mathcal{T}_i)$ is the accumulated confidence for all patterns that belong to the intersection of $u_i$’s fingerprint patterns and $u_u$’s persistent patterns; $\sum \text{conf}(F^j_i, \mathcal{T}_i)$ is the accumulated confidence for all patterns in $u_i$’s fingerprint. If $P(\mathcal{T}_u) \cap \mathcal{F}_i$ accounts for a large percentage of patterns in $\mathcal{F}_i$, which implies that two users tend to be similar, the distance tends to be small. If multiple users who have fingerprints have non-zero distance with $u_u$, we assign $u_u$ to the user who has the smallest distance.

It is worth noting that a user with transient DNS behaviors may introduce a large volume of noises when discovering persistent patterns. For example, if a user is only active for one epoch (i.e., there is only one transaction for this user), then all of patterns for this user would be persistent since they are active for that transaction, resulting 100% for the $\text{supp()}$ function. A large number of “persistent” patterns generated by transient users may significantly affect the effectiveness for both pattern generation and matching. In the pattern generation phase, these patterns may drastically decrease the contrast confidence of persistent patterns for persistent IPs. In the pattern matching phase, a transient user is likely to have a large overlap with a known user with respect to their patterns, which implies a false positive. Therefore, in our current design, we only consider those users (or IP addresses) that are sufficiently persistent by themselves. Specifically, if a set of users (or IP addresses) subject to analysis have up to $M$ transactions, our implementation only considers those IP addresses that are active for at least $\frac{M}{2}$ transactions. For example, if a
set of IP addresses have up to 7 transactions, we will only analyze their users that are active for at least 4 transactions.

2.3.2 Patterns

The querying behaviors of DNS are closely related to networking activities of individual users. For example, visiting a website or starting a network application (e.g., an instant messenger) usually triggers the resolution of associated domain(s). The routine and personal networking activities of a user may lead to persistent DNS patterns that are unique to him/her. Based on this intuition, we have designed five types of DNS patterns that characterize a user’s DNS querying behaviors from three perspectives, including the domain name (i.e., Pattern 1), the inter-domain relationship (i.e., Pattern 2 and 3), and temporal behavior (Pattern 4 and 5). In this section, we will present the definitions of these patterns and the motivation behind their design.

Pattern 1 - Domain Set: A user may have steady interest for certain websites and use some applications routinely. These activities are likely to result in a set of domains that are repeatedly queried by this user across multiple epochs. Since the interest and application usage patterns are highly personal, the repeatedly queried domains may vary drastically across different network users. We therefore introduce the domain set pattern (denoted as $F_{domain}$), which is simply a set of domains that meets the requirements of persistence and uniqueness. Particularly, a transaction $T$ satisfies the domain name pattern $F_{domain}$ if all domains in $F_{domain}$ are observed in transaction $T$.

In order to identify $F_{domain}$ ideally, we can enumerate all possible domain set based on all domains derived from each transaction of a user, where the smallest domain set contains a single domain from this transaction and the largest domain set contains all domains in this transaction. We can then evaluate the persistence and uniqueness of these domain sets. Unfortunately, when the number of domains involved in a transaction is large, the sheer volume of domain sets will become overwhelming. In order to solve this problem, we
Table 2.1: Transactions and their associated domains for two users across 5 epochs, where \{a, b\} becomes the domain set fingerprint pattern for \(u_1\); Reprinted from [39]

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Domains</th>
<th>(u_1)</th>
<th>(u_2)</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_1)</td>
<td>a, b, c, d</td>
<td>a, c</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>(T_2)</td>
<td>a, b</td>
<td>a, e</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>(T_3)</td>
<td>b, a, f, k</td>
<td>a, b, c</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>(T_4)</td>
<td>e, f</td>
<td>b, k</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>(T_5)</td>
<td>c, d</td>
<td>b</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

generate domain sets that contain up to \(N\) unique domains, where \(N = 2\) for our current implementation.

Table 2.1 presents an illustrative example: two users, \(u_1\) and \(u_2\), are active across five consecutive epochs, resulting in five transactions, respectively. All domains queried by \(u_1\) and \(u_2\) for each epoch are listed in the second and third columns in Table 2.1. If we configure \(\alpha = \frac{3}{5}\), the \(u_1\) has persistent \(F_{\text{domain}}\) patterns including \{a\}, \{b\} and \{a, b\} since \(\text{supp}\{a\}, T_1\} = \frac{|\text{mt}\{a\}, T_1\}|{|T_1|} = \frac{3}{5} \geq \frac{3}{5}\), \(\text{supp}\{b\}, T_1\} = \frac{|\text{mt}\{b\}, T_1\}|{|T_1|} = \frac{3}{5} \geq \frac{3}{5}\), and \(\text{supp}\{a, b\}, T_1\} = \frac{|\text{mt}\{a, b\}, T_1\}|{|T_1|} = \frac{3}{5} \geq \frac{3}{5}\). Similarly, \(u_2\) will have two persistent patterns including \{a\} and \{b\}, where \(\text{supp}\{a\}, T_2\} = \frac{|\text{mt}\{a\}, T_2\}|{|T_2|} = \frac{3}{5} \geq \frac{3}{5}\) and \(\text{supp}\{b\}, T_2\} = \frac{|\text{mt}\{b\}, T_2\}|{|T_2|} = \frac{3}{5} \geq \frac{3}{5}\). Considering only these two users, it is easy to reach a conclusion that \(\text{conf}\{a\}, T_1\} = \frac{1}{2}\), \(\text{conf}\{b\}, T_1\} = \frac{1}{2}\), \(\text{conf}\{a, b\}, T_1\} = 1\), \(\text{conf}\{a\}, T_2\} = \frac{1}{2}\), and \(\text{conf}\{b\}, T_2\} = \frac{1}{2}\). If we set \(\beta = 60\%\), \{a, b\} becomes the fingerprint pattern for \(u_1\).

**Pattern 2 - Domain Sequence:** A network user’s routine networking activities could involve his/her individualized preferences and the order in which network activities are carried might be able to reflect such preferences. We consequently define a domain sequence pattern denoted as \(F_{\text{seq}}\), where \(F_{\text{seq}}\) is a finite sequence of domains. Given two domains in \(F_{\text{seq}}\) (i.e., \(d_i \in F_{\text{seq}}\) and \(d_j \in F_{\text{seq}}\)), \(d_i \preceq d_j\) means that \(d_i\) is issued before \(d_j\).

Similar to the domain set pattern, the ideal implementation to derive domain sequence patterns should consider domain sequences with all possible lengths derived from a transaction. Unfortunately, the ideal solution could result in a prohibitively huge volume of
domain sequence patterns when the number of domains contained in a transaction becomes large. Therefore, we only generate domain sequence patterns composed of two domains. To be more specific, $F_{\text{seq}} = (d_i, d_j)$ where $d_i \preceq d_j$ in the transaction.

Compared to domain set patterns, domain sequence patterns offer an additional dimension to differentiate two users. For example, if two users visit facebook and twitter routinely, they will have two identical $F_{\text{domain}}$ patterns (i.e., “www.facebook.com” and “www.twitter.com”). However, if the first user always visits facebook before twitter while the second user follows the reverse order, DNSMiner will generate two disparate persistent domain sequence patterns (i.e., (www.facebook.com, www.twitter.com) and (www.twitter.com, www.facebook.com)) for these two users, respectively.

**Pattern 3 - Window-Aware Domain Sequence:** DNSMiner further expands the domain sequence patterns by incorporating the first and last time when a domain is visited. Specifically, rather than considering every possible pairwise sequence for $d_i$ and $d_j$ from all tuples within a transaction, DNSMiner considers the tuples in which $d_i$ and $d_j$ are first and last observed. To this end, we extract a 3-tuple for each domain (e.g., $d_i$) in a transaction denoted as $< d_i, s_i, e_i >$, where $s_i$ and $e_i$ refer to the first and last time $d_i$ is observed in the transaction, respectively. In order to illustrate the design of this pattern, we consider two domains, $d_i$ and $d_j$, whose 3-tuples are $< d_i, s_i, e_i >$ and $< d_j, s_j, e_j >$, respectively. Without loss of generality, we assume that $d_i$ alphabetically precedes $d_j$. The comparison of both starting and ending times of these two domains will result in four 4-tuples as illustrated in Table 2.2. The third element in a 4-tuple indicates how two domains are compared. For example, “ss” indicates that $d_i$’s starting time is compared to $d_j$’s starting time and “se” indicates the comparison between $d_i$’s starting time and $d_j$’s ending time. The second column in Table 2.2 shows rules we have used to assign values for the fourth variable. It is worth noting that these four window-aware sequence patterns might not be independent. For example, if $p_3$ in $< d_i, d_j, es, p_3 >$ is 0, which means that the last time we observe $d_i$ precedes the first time we observe $d_j$, then all $p_*$ variables in other 4-tuples for
Table 2.2: Window-Aware Patterns; Reprinted from [39]

<table>
<thead>
<tr>
<th>Window-Aware Patterns</th>
<th>$p_i$’s Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;d_i, d_j, ss, p_1&gt;$</td>
<td>if($s_i &lt; s_j$) $p_1 = 0$; else $p_1 = 1$;</td>
</tr>
<tr>
<td>$&lt;d_i, d_j, se, p_2&gt;$</td>
<td>if($s_i &lt; e_j$) $p_2 = 0$; else $p_2 = 1$;</td>
</tr>
<tr>
<td>$&lt;d_i, d_j, es, p_3&gt;$</td>
<td>if($e_i &lt; s_j$) $p_3 = 0$; else $p_3 = 1$;</td>
</tr>
<tr>
<td>$&lt;d_i, d_j, ee, p_4&gt;$</td>
<td>if($e_i &lt; e_j$) $p_4 = 0$; else $p_4 = 1$;</td>
</tr>
</tbody>
</table>

Table 2.3: A sequence of DNS queries; Reprinted from [39]

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

d_i and d_j will always be 0. We exploit such dependency in our implementation to reduce the number of patterns yielded for each pair of domains.

Table 2.3 illustrates a series of domains queried by a user together with their timestamps, where all these domains belong to one transaction and $t_0 < t_1 \ldots t_4 < t_5$. For this user, two 3-tuples in the form of $<d_i, s_i, e_i>$ will be derived, including $<a, t_0, t_5>$ and $<b, t_1, t_4>$. For example, $<a, t_0, t_5>$ indicates that the domain a is first and last queried in this transaction at $t_0$ and $t_5$, respectively. We follow the definition of window-aware patterns as indicated in Table 2.2 to derive four window-aware patterns for this example, which includes $<a, b, ss, 0>$, $<a, b, se, 0>$, $<a, b, es, 1>$, and $<a, b, ee, 1>$. As indicated in this example, some patterns may imply others, making it possible to simplify the generation of window-aware patterns. For example, if we know $<a, b, ss, 0>$, we can directly conclude that $<a, b, se, 0>$ without generating it from data.

**Pattern 4 - Period Behavior:** Network users’ networking activities often exhibit strong temporal patterns. For example, a user could visit a news website every morning while another user surfs it over every afternoon. Consequently, each domain together with its temporal information may well represent a user. We therefore introduce the Period Behavior pattern (denoted as $F_{period}$), which is defined as a domain-period combination. The “period” refers to a tag indicating “morning”, “afternoon”, and “evening”. In order to derive such pattern, we first map the timestamp of each tuple into one of three period tags,
where “morning”, “afternoon”, and “evening” stand for [5:00AM, 11:00AM), [11:00AM, 5:00PM), and [5:00PM, 5:00AM), respectively. Next, for each tuple, we integrate its domain and its corresponding period tag into a domain-period combination. For example, (www.facebook.com, 2013-09-17 08:30:23), a tuple in a DNS stream, will generate (www.facebook.com, morning) as its Period Behavior pattern.

**Pattern 5 - Hourly Behavior:** We further introduce the Hourly Behavior pattern to characterize a user’s networking activities at a finer granularity. Rather than mapping a timestamp into a period tag, DNSMiner maps a timestamp to its corresponding hour, thereby leading to a domain-hour combination denoted as $F_{\text{hourly}}$. For instance, the tuple (www.facebook.com, 2013-09-17 08:30:23) will be mapped into (www.facebook.com, 08).

### 2.3.3 System Implementation

A network user may generate a large number of DNS queries. As the number of network users increases, the scalability of DNSMiner becomes a concern. To address the challenge, we have implemented DNSMiner using the Hadoop MapReduce platform. The two phases of Map and Reduce workflows in the implementation are presented in Fig. 2.2. DNSMiner first identifies persistent patterns for each user. Since the identification of persistent patterns for each user is independent to that for other users, we can easily parallelize the
computation by partitioning/mapping tuples (i.e., \(<uid, domain, timestamp >\)) into reducers based on their \(uids\) (i.e., the step ① in Fig. 2.2). Each reducer will then enumerate all patterns for each transaction of \(u_i\); for each derived pattern \(F^j_i\), its \(supp()\) value in the context of \(u_i\) will be subsequently calculated; we consequently apply the predefined threshold \(\alpha\) and preserve all persistent patterns (i.e., patterns whose \(supp()\) values are greater than \(\alpha\)). These three actions together are performed in reducers for the step ② in Fig. 2.2. Next, we partition patterns together with their associated \(uids\) and \(supp()\) values into reducers, where the pattern serves as the key (the step ③ in Fig. 2.2). Finally, each reducer will calculate the contrast confidence for each pattern with respect to each user and yield those unique ones in the step ④ (e.g., \(conf(F, T) \geq \beta\)).

### 2.4 Evaluation

We have implemented a prototype system named **DNSMiner**, and evaluated it using DNS queries collected from a large campus network. Our evaluation aims at answering three questions: “Can DNS-based fingerprints effectively identify their corresponding network users?” “How do parameter values impact **DNSMiner**'s effectiveness?”, and “How effective is each category of patterns?”. The following sections describe the experimental setup and evaluation results.

#### 2.4.1 Experimental Setup

We obtained DNS queries collected from a large campus network of Xi’an Jiaotong University, China, where the DNS queries are collected below the major recursive DNS servers used by the campus network. Aiming at facilitating the network management, the campus network assigns static IP addresses to the vast majority of its users after they register at the network management center (i.e., DHCP is not supported for the network users). Only a few buildings use dynamic IP addresses and we have excluded DNS queries issued from
Table 2.4: The # of IPs in $D_1$ and $D_2$, # of persistent IPs ($|P_1|$ and $|P_2|$), # of IPs with persistent patterns in $P_1$ and $P_2$, and # of IPs with fingerprint patterns in $D_1$ (i.e., $|FP_1|$); Reprinted from [39]

<table>
<thead>
<tr>
<th>Week</th>
<th># of IPs</th>
<th># of Persistent IPs</th>
<th># of IPs with Persistent Patterns</th>
<th># of IPs with Fingerprint Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1 ($D_1$)</td>
<td>55,459</td>
<td>16,003</td>
<td>12,900</td>
<td>11,921</td>
</tr>
<tr>
<td>Week 2 ($D_2$)</td>
<td>54,751</td>
<td>9,120</td>
<td>7,119</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 2.3: The CDF distribution of the maximum $supp()$ for all persistent IP addresses in $D_1$ (i.e., all IPs in $P_1$). A significant percentage (64.18%) of IPs in $P_1$ have patterns that are persistent across 7 epochs; Reprinted from [39]

their corresponding subnets. Sensors were deployed to collect DNS queries that are issued by all hosts in campus network. For each DNS query, three pieces of information were extracted, including the domain name, the timestamp, and the IP address that issues this query. We collected two sets of DNS queries from two consecutive weeks at September 2013, which are denoted as $D_1$ and $D_2$, respectively. As illustrated in the second column of Table 2.4, $D_1$ and $D_2$ contain 55,459 and 54,751 unique IP addresses, respectively. Both $D_1$ and $D_2$ contain a large number of DNS queries (i.e., 467,388,490 queries in $D_1$ and 238,993,575 in $D_2$).

As Internet activities typically show diurnal patterns [68, 21], we considers one day as one epoch. Specifically, an epoch starts from 5:00AM and lasts for 24 hours. Both
transaction-sets for fingerprint extraction and matching contain 7 epochs (i.e., for 7 con-
secutive days). We configure $\alpha = \frac{5}{7}$, which means that a fingerprint pattern has to be
persistent for at least $\frac{5}{7}$ out of the active days for its corresponding IP address. We also set
$\beta = 60\%$.

We use the queries of the first 7 days (i.e., $D_1$) to derive DNS fingerprints and those
of the remaining week (i.e., $D_2$) to evaluate the extent to which the fingerprints can effec-
tively de-anonymize users in a new DNS stream. As we have discussed in Section 2.3, IP
addresses with transient DNS behaviors are likely to introduce noises. Therefore, we only
consider those IP addresses that experience sufficient persistence by themselves. Specifi-
cally, since $D_1$ and $D_2$ contain up to 7 transactions, we preserve those IP addresses that are
active for at least half of the 7 transactions (i.e., for at least 4 transactions). We use $P_1$ to
represent a set of persistent IPs in $D_1$ and $P_2$ in $D_2$. As illustrated in Table 2.4, $P_1$ and $P_2$
contain 16,003 and 9,120 IP addresses, respectively.

2.4.2 Experimental Results

2.4.2.1 Fingerprint Extraction

The first step of DNSMiner is to assess the persistence of patterns for each IP address in
$P_1$. Specifically, for each IP address in $P_1$, we extract all of its patterns, investigate their
$supp()$ values, and preserve those whose $supp()$ values are greater than the predefined
threshold $\alpha$. We identify the maximum $supp()$ value for each IP address and plot the
distribution of maximum $supp()$ value for all IPs in $P_1$ in Fig. 2.3. As illustrated in the
distribution, a significantly large percentage of persistent IPs (i.e., IPs in $P_1$) indeed have
persistent patterns. Particularly, 64.18% of IPs in $P_1$ have the maximum $supp()$ value of 1,
indicating that each of these IPs has repeatedly shown at least one pattern across entire 7
epochs. In addition, a large percentage of 80.60% of IPs in $P_1$ have at least one persistent
pattern whose $supp()$ value is greater than $\alpha = \frac{5}{7}$. This results in 12,900 IPs with persistent
patterns in $P_1$, which account for totally 313,248,287 persistent patterns.

The second step of DNSMiner is to investigate the uniqueness of persistent patterns based on their contrast confidence (i.e., $\text{conf}(F^j_i, T_i)$). Again, $\text{conf}(F^j_i, T_i)$ quantifies the uniqueness of a pattern $F^j_i$ to its corresponding user $u_i$. In order to visualize the experiment results, for each IP with persistent patterns, we derive the highest contrast confidence for all its persistent patterns; we then present the distribution of the highest contrast confidence values for these IPs in Fig. 2.4. As illustrated in Fig. 2.4, about 70% percentage of IPs with persistent patterns have patterns whose contrast confidence is 1, which indicates that these patterns are unique for their corresponding users. In DNSMiner, we use the predefined threshold $\beta = 60\%$ to further identify those persistent that also experience significant uniqueness (i.e., fingerprint patterns). Totally, DNSMiner has identified 11,921 IP addresses that have fingerprint patterns, where these IP addresses form a set namely $FP_1$ and $FP_1 \subseteq P_1$. DNSMiner totally generated 222,508,026 fingerprint patterns, among which the domain set pattern, the domain sequence pattern, the window-aware domain sequence pattern, the period pattern, and hourly behavior pattern account for 16.43%, 11%, 72.51%, 0.02%, and 0.04%, respectively. We count the total number of fingerprint patterns for each IP address and plot their distribution in Fig. 2.5. The distribution indicates that these IPs tend to have a large number of DNS fingerprint patterns, implying strongly discriminative DNS behaviors. Particularly, more than 78% of IP addresses in $FP_1$ have at least 100 fingerprint patterns.

### 2.4.2.2 Fingerprint Matching

As introduced in Section 2.4.2.1, $FP_1$ represents a set of IPs in $D_1$ whose DNS behavioral fingerprints have been derived by DNSMiner. We also use $P_1$ and $P_2$ to represent sets of persistent IPs for $D_1$ and $D_2$, respectively. For fingerprint matching, our objective is to use fingerprint patterns for IPs in $FP_1$ to reveal their presence in $P_2$. Specifically, we
Figure 2.4: The CDF distribution of the highest contrast confidence for each IP address that has at least one persistent pattern. Approximately 70% have unique persistent patterns (i.e., with contrast confidence of 1); Reprinted from [39]

Figure 2.5: The CDF distribution of the number of fingerprint patterns for each IP address. IP addresses with fingerprint patterns tend to have a large number of fingerprint patterns; Reprinted from [39]
perform the pattern matching as discussed in Section 2.3 to identify all IPs in $P_2$ whose distance (i.e., $\text{dist}(u_u, u_i)$) is smaller than 1 compared to any IP in $FP_1$, where these IPs together form a set named as $K$. $K$ can be further divided into two sets, namely $KC$ and $KI$, which represent the IPs that are correctly and incorrectly identified, respectively (i.e., $K = KC \cup KI$). Subsequently, we define the following three metrics to quantify the effectiveness of fingerprint patterns.

- The percentage of identified IP addresses ($\Pi$): $\frac{|K|}{|FP_1 \cap P_2|}$. We expect DNSMiner to identify all IPs in $FP_1 \cap P_2$ since IPs in $FP_1 \cap P_2$ indeed have fingerprint patterns in the first week and are persistent in the second week. $\frac{|K|}{|FP_1 \cap P_2|}$ represents the overall effectiveness on identifying IPs in a new DNS stream.

- The detection rate: $\frac{|KC|}{|K|}$ (DR). This ratio shows the ratio of the number of correctly identified IPs over the number of all identified IPs.

- The false positive rate: $\frac{|KI|}{|K|}$ (FP). This ratio shows the ratio of the number of incorrectly identified IPs over the number of all identified IPs.

We have performed the evaluation of fingerprint matching using the DNS stream of $D_2$, where the evaluation results are presented in Table 2.5. Specifically, 4,894 IPs in $P_2$ (i.e., persistent IPs in the second week) have fingerprint patterns in the first week (i.e., $|FP_1 \cap P_2| = 4,894$). In other words, the ideal objective is to identify all these 4,894 IPs in the DNS stream of the second week (i.e., $D_2$) using their fingerprint patterns extracted from the first week (i.e., $D_1$). The matching results show that totally 3,408 IPs have been identified, resulting in the percentage of identified IPs of 69.63%. Among these 3,408 IP addresses, 3,365 IPs are correctly attributed to those IPs in $FP_1$, resulting a high detection rate of 98.74% and a low false positive rate of 1.26%.

We have deployed DNSMiner on a Hadoop platform with 15 nodes. The entire process for both extracting and matching fingerprint patterns consumes approximately 4 hours.
Table 2.5: The accuracy of identifying users in a new DNS stream $D_2$ using fingerprint patterns extracted from a historical DNS stream $D_1$. Among 69.63% IPs that are identified by fingerprint patterns, 98.74% are correctly revealed; Reprinted from [39]

| Week 2 ($D_2$) | $|P_2|$ | $|F_P_1 \cap P_2|$ | $|K|$ | $|K_C|$ | II(%) | DR(%) | FP(%) |
|----------------|-------|---------------------|------|-------|-----|-------|-----|
| Week 2 ($D_2$) | 9,120 | 4,894               | 3,408| 3,365 | 43  | 69.63 | 98.74| 1.26 |

Table 2.6: The detection performance under different $\alpha$ and $\beta$ values. “PI” indicates the percentage of IPs in $P_1$ that have fingerprint patterns; “II” is denoted as the percentage of IPs in $F_{P_1} \cap P_2$ that are detected by fingerprint patterns; “DR” and “FP” refer to the detection rate and false positive rate, respectively; Reprinted from [39]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\alpha = 4/7$</th>
<th>$\alpha = 5/7$</th>
<th>$\alpha = 6/7$</th>
<th>$\alpha = 7/7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta = 30%$</td>
<td>PI(%)</td>
<td>II(%)</td>
<td>DR(%)</td>
<td>FP(%)</td>
</tr>
<tr>
<td>$\beta = 40%$</td>
<td>73.28</td>
<td>69.45</td>
<td>93.33</td>
<td>6.67</td>
</tr>
<tr>
<td>$\beta = 50%$</td>
<td>72.66</td>
<td>70.02</td>
<td>96.09</td>
<td>3.91</td>
</tr>
<tr>
<td>$\beta = 60%$</td>
<td>71.98</td>
<td>70.43</td>
<td>95.79</td>
<td>4.21</td>
</tr>
<tr>
<td>$\beta = 80%$</td>
<td>71.86</td>
<td>70.49</td>
<td>94.78</td>
<td>5.22</td>
</tr>
</tbody>
</table>

Figure 2.6: The trend of detection rates when $\alpha$ increases given a fixed value for $\beta$, where DNSMiner achieves the best accuracy of 98.74% when $\alpha = 5/7$ and $\beta = 60\%$; Reprinted from [39]
2.4.2.3 Evaluating The Impact of Parameter Values

*DNSMiner* needs two parameters including $\alpha$ and $\beta$ to be configured. While the evaluation result based on the current configuration ($\alpha = \frac{5}{7}$ and $\beta = 60\%$) yields a high detection rate, we further investigate how parameter values affect the system effectiveness. Specifically, we assign a wide range of values to $\alpha$ (i.e., $\alpha = \frac{4}{7}, \frac{5}{7}, \frac{6}{7}, \frac{7}{7}$) and $\beta$ (i.e., $\beta = 30\%, 40\%, 50\%, 60\%, 70\%, 80\%$) and then perform the fingerprint extraction and matching for each combination of $\alpha'$ and $\beta'$ values. The experimental results are summarized in Table 2.6, where each cell in the table contains i) the percentage of IPs in $P_1$ that have fingerprint patterns (i.e., $\frac{|FP_1|}{|P_1|}$), ii) the percentage of identified IPs (i.e., $\frac{|K|}{|FP_1 \cap P_2|}$), iii) the detection rate (i.e., $\frac{|KC|}{|K|}$), and iv) the false positive rate (i.e., $\frac{|KI|}{|K|}$). Fig. 2.6 visualizes the trend of detection rates when $\alpha$ increases from $\frac{4}{7}$ to $\frac{7}{7}$ for a fixed value of $\beta$.

As indicated by the experimental results, when both $\alpha$ and $\beta$ increase, the percentage of IPs in $P_1$ that have fingerprint patterns drops. For example, 74.04% of IPs in $P_1$ have fingerprint patterns given $\alpha = \frac{4}{7}$ and $\beta = 30\%$ while the percentage is 47.15% given $\alpha = \frac{7}{7}$ and $\beta = 80\%$. The changes of $\alpha$ and $\beta$ affect $K$ and $FP_1$ simultaneously, thereby impacting the percentage of identified IP addresses (i.e., $\frac{|K|}{|FP_1 \cap P_2|}$). This measure stays very stable (i.e., close to 70%) when $\alpha = \frac{4}{7}, \frac{5}{7}$ and all $\beta$ values under investigation. When $\alpha \geq \frac{6}{7}$, this measure drops significantly (i.e., around 63% for $\alpha = \frac{6}{7}$ and 44% for $\alpha = \frac{7}{7}$). Despite the fluctuation of the percentage of persistent IPs with fingerprint patterns and the percentage of identified IP addresses along with the changes of $\alpha$ and $\beta$, *DNSMiner* accomplishes high detection performance. Specifically, for all combinations of $\alpha$ and $\beta$ values in our experiments, the detection rates are above 87.80%. Particularly, when we configure $\frac{4}{7} \leq \alpha \leq \frac{6}{7}$, all $\beta$ values lead to detection rates higher than 90%. Such experiment results imply that our method accomplishes the high detection accuracy over a wide range of parameter values. Nevertheless, considering the percentage of users with fingerprint patterns (i.e., “PI”) and the percentage of identified IPs (i.e., “II”), $\alpha \in \left[\frac{4}{7}, \frac{5}{7}\right]$ and $\beta \in [40\%, 70\%]$ yield
Table 2.7: The detection performance of *DNSMiner* for each category of patterns: “Domain name” refers to the domain set pattern; “Inter-domain relationship” includes the domain sequence pattern and window-aware domain sequence pattern; “Temporal behavior” contains period and hourly behavior pattern; Reprinted from [39]

<table>
<thead>
<tr>
<th>Pattern Category</th>
<th>II(%)</th>
<th>DR(%)</th>
<th>FP(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain name</td>
<td>29.65</td>
<td>85.83</td>
<td>14.17</td>
</tr>
<tr>
<td>Inter-domain relationship</td>
<td>43.21</td>
<td>90.72</td>
<td>9.28</td>
</tr>
<tr>
<td>Temporal behavior</td>
<td>32.50</td>
<td>87.90</td>
<td>12.10</td>
</tr>
</tbody>
</table>

the best detection performance with approximately (i.e., approximately 70% for both “PI” and “II”, and detection rates higher than 95%).

We have also investigated the detection performance of *DNSMiner* when only a category of patterns are used and the experiment results are presented in Table 2.7, where $\alpha = \frac{5}{7}$ and $\beta = 60\%$. As indicated in Table 2.7, patterns belonging to the category of the inter-domain relationship resulted in the best detection rates (i.e., a detection rate of 90.72% and a false positive rate of 9.28%) compared to patterns in the other two categories. Nevertheless, all these patterns collectively accomplish the best detection performance as indicated in Table 2.6, indicating that all patterns complement each other in *DNSMiner*.

### 2.5 Discussion

*DNSMiner* currently concentrates on network users whose DNS activities are persistent. For example, network users who were active for at least 4 days out of 7 days were considered in our experiments. Despite the fact that such design mitigates the noises caused by network users with transit DNS activities, it may actually result in limitations for the practical usage of *DNSMiner*. First, *DNSMiner* by design cannot generate fingerprint patterns for those network users with transit DNS activities. Second, *DNSMiner* requires that DNS queries can be attributed to their corresponding users over a relative long period (e.g., across the epochs for fingerprint generation). Specifically, when we use an IP address to represent a user, the IP address should not change across the epochs for pattern gener-
tion and matching. For networks using static IP addresses, this limitation can be easily overcome, which is actually the case for our evaluation. However, when the IP address associated with a user changes frequently (e.g., in networks that use dynamic IPs with small lease time), it becomes a challenging problem to directly attribute IP addresses to their corresponding users across a series of epochs.

We acknowledge such limitations in the current design and our future work will focus on systematically addressing them. Specifically, a few potential improvements can be explored. First, we plan to design an algorithm that can adaptively define epochs for each IP address and aggregate them into transaction set according to the DNS activities of this IP address. Particularly, the transaction set will be discovered in a way that it is very unlikely for the host to change its IP address across the epochs belonging to this transaction set. Second, rather than manually defining fingerprint patterns, we intend to propose methods that can automatically generate patterns and perform pattern selection. Particularly, we expect that the patterns will give more weight on characterizing the short-term DNS activities of a user. Lastly, our proposed methods and features are possibly adopted to identify the users on mobile devices (e.g., smartphones) for future work.

2.6 Summary

We present a novel system, DNSMiner, to automatically derive behavioral fingerprints from DNS queries, where behavioral fingerprints are expected to reveal the presence of their corresponding users in new DNS streams whose identities are unknown (e.g., anonymized). A behavioral fingerprint is composed of a collection of patterns that systematically characterize each user’s DNS activities from three different perspectives including the domain name, the inter-domain relationship, and the temporal behavior. The extensive evaluation based on DNS queries collected from a large campus network has demonstrated that these patterns can accomplish a high detection accuracy of 98.74% and a low false positive rate
of 1.26%. Despite its high detection accuracy, more patterns could be discovered and incorporated into *DNSMiner*. Nevertheless, *DNSMiner* demonstrates the lower bound of the effectiveness of using DNS-based patterns to reveal users’ presence in network traffic.
Chapter 3: Detecting Fake Anti-Virus Webpages

3.1 Motivation

Tremendous cyber-security concerns have led to computer systems with enhanced security features. As a result, it becomes increasingly difficult for attackers to directly compromise end users’ systems by exploiting software vulnerabilities. As an alternative strategy, social engineering attacks, which take advantage of humans’ psychological vulnerabilities, rapidly gain their popularity. Among various types of social engineering attacks such as spamming and phishing, fake Anti-Virus (AV) attacks have become one of the most significant threats.

A fake AV attack is to disguise malware as anti-virus software and lure end users into installing it. As web browsers become the dominating network applications, webpages have become a major way for attackers to launch fake AV attacks [27]. Specifically, upon being visited by a user, a fake AV webpage usually claims that it offers a customized anti-virus product and encourages the user to install it, where the offered anti-virus product is actually a malicious executable. In order to stimulate users’ action, a fake AV webpage could claim that it has identified (new) infections on end users’ systems after a thorough scanning. Both the appearance of the anti-virus product and scanning process impersonate those of popular
security vendors such as Microsoft Security Essentials and Symantec Norton. Fig. 3.1 presents an example of a typical fake AV webpage, which impersonates a safe webpage to download a free Symantec Norton Antivirus 2014. A user who approves the installation of the fake anti-virus software will immediately render his/her system infected. Specifically, a typical fake AV webpage asks users to follow four steps to launch a malicious attack [79]. First, a user visits a fake AV webpage using information (e.g., news, images) from search engine and downloads a fake AV file. Next, the fake AV software shows false alarms that warn a user of the danger of infected files, and subsequently request him to purchase a rogue AV software as a solution for stealing his/her sensitive information (see Fig. 3.2).

As a result of users’ increasing awareness of malware threats and growing familiarity of anti-virus software, fake AV attacks are surprisingly successful. In fact, a recent study from Google has indicated that fake AV attacks are responsible for 50% of all malware delivered via Internet advertising and their prevalence keeps growing [65].

Detecting fake AV webpages is therefore of great importance. However, it is a challenging task due to several typical characteristics for fake AV webpages. First, fake AV webpages require users’ interaction to install malware and hence do not require any malicious contents to perform automatic exploitation without users’ consent (e.g., shellcode).
In addition, a fake AV webpage does not need to impersonate a whole authentic anti-virus webpage. Instead, it often uses a few representative elements from the impersonated webpage, such as the names and icons of anti-virus products, while the remaining webpage components could exhibit arbitrary semantics. Finally, in order to trick users into believing that the masqueraded products offer up-to-date solutions to emerging threats, attackers may frequently update the information for both products and threats (e.g., the product versions and threat names). These characteristics easily distinguish fake AV attacks from other prevalent web-based attacks such as drive-by downloads and phishing, and consequently impede the direct application of the detection methods for these attacks to detect fake AV attacks. For example, drive-by download detection methods that require the observation of webpage content for exploitation [16], [19] or automatic binary downloading [49] will fail to detect fake AV webpages due to the absence of malicious content for exploitation. Phishing-webpage detection methods that rely on the similarity between a potential malicious webpage and its impersonated authentic webpage [66] can be easily circumvented.

In this chapter, in order to effectively detect fake-AV detection and overcome the aforementioned challenges, we propose a novel system, namely **DART**. **DART** employs a collection of features to profile a webpage and further discriminate between fake AV webpages and benign webpages by integrating these features using a statistical classifier. These features aim to characterize a webpage by answering three questions that are critical...
for the success of a fake AV webpage. First, how does a fake AV webpage increase the opportunity to be visited by an end user? What tricks does it use to convince an end user to install the executable? Where is a fake webpage actually located? Specifically, DART automatically extracts features that profile search engine optimization techniques used by fake AV webpages, various identities to impersonate authentic security webpages, and the network infrastructures. Since malware threats are constantly evolving, identities related to authentic anti-virus software, such as names of the product and malware, may actively change. In order to solve this challenge, DART can automatically discover diverse and trendy security keywords by performing semantic analysis based on messages from popular social networks such as Twitter. We have performed extensive evaluation based on data collected from the real-world network. Our experimental results have demonstrated that DART can achieve a high detection rate (90.4%) with a very low false positive rate (0.2%).

3.2 Related Work

Extensive measurement efforts [65], [74] have been invested to study the severity of fake AV attacks and gained in-depth understanding of their infrastructures and operations. For example, Rajab et al. [65] performed large-scale study based on malicious webpages collected by Google and concluded that fake AV accounted for 15% of all malware detected on the web. Stone-Gross et al. [74] managed to acquire back-end servers for several fake AV campaigns and revealed sophisticated techniques adopted by attackers to accomplish robustness and agility of their networking infrastructures.

A few methods [23], [67], which target at detecting fake AV attacks, have been proposed. Dietrich et al. [23] designed a method to detect fake AV attacks by dynamically analyzing suspicious binaries. Specifically, binaries are executed and their user interfaces will be collected. The method will aggregate binaries with similar user interfaces into clusters. If several binary instances in a cluster are known to be a fake AV, their maliciousness
can be propagated to other binaries in the same cluster. Seifert et al. [67] proposed a method to detect fake AV webpages, which focuses on detecting webpages that show an animation of an anti-virus scan. This method first extracts a collection of features from the snapshot of a webpage, which aim at characterizing visual elements related to anti-virus scanning. It further employs a statistical classifier to discriminate between fake AV webpages and legitimate ones based on the derived features. These methods have demonstrated promising detection performance. However, they suffer from several limitations, which may impede their practical deployment and effectiveness. Specifically, the first method [23] mandates the execution of a binary, making it extremely challenging to deploy it in end users’ hosts. In addition, the execution of binaries implies considerable computation resources (e.g., mounting and cleaning virtual images), thereby significantly limiting its efficiency. The second method [67] relies on the observation that fake AV webpages are likely to use the animation of an anti-virus scan to “scare” end users. Unfortunately, this observation is not always true. In fact, a fake AV webpage can simply display a text description to lure end users into installing the binary. Consequently, the second method is prone to generating false negatives.

Our method differs from existing detection approaches in the following aspects. First, the system [23] detects a fake AV webpage by performing the dynamic behavior of the binary associated with the fake AV webpage. This implies the detection engine needs a controlled environment (e.g., a well-managed virtual machine) to perform detection. The cost to maintain (e.g., clean and restart the image) is very high. In contrast to [23], our method leverages webpage information, rather than dynamic behavior of a binary, to detect fake AV webpages, thereby becoming more efficient compared to [23]. The system [67] mainly leverages the visual appearance (the scanning animation to be more specific) of a fake AV webpage for detection. In addition to the potential high overhead implied by image analysis, a fake AV webpage does not necessarily need to adopt scanning animation and therefore may evade the detection of the system [67]. Comparatively, DART uses multi-
dimensional features rather than exclusively relying on the visual appearance. In fact, in our evaluation of lightweight features, DART can still achieve high detection accuracy without image-based features. This implies a higher level of robustness of DART compared to the system [67].

3.3 System

DART involves two phases, namely training phase and detection phase. In the training phase, a statistical classifier is built based on a set of pre-labelled fake AV webpages and benign webpages. Specifically, given the URL of a landing page, DART instructs our browser to load this webpage and then extract a number of features. In other words, we use a feature vector to represent a landing page and the webpages that are subsequently loaded when the browser renders it. In the detection phase, an unknown landing page will be converted to a feature vector and then analyzed by the statistical classifier to assess its maliciousness. The Fig. 3.3 presents the architectural overview of DART. Considering the fact that various well-known statistical classifiers have been widely used, how to design features capable of distinguishing between fake AV webpages and benign webpages becomes of central importance. To illustrate the detailed design of DART, we first describe
how we collect fake AV webpages on the web (see Section 3.3.1). Next, Section 3.3.2
details a method of discovery trendy and diverse security keywords and provides such real-
world keyword examples from our study data. Finally, we discuss specific patterns used by
DART and the motivations behind their design (see Section 3.3.3).

3.3.1 Collecting & Labeling Data

We have collected fake AV webpages to motivate our design and evaluate its performance.
To this end, we have built an instrumented browser based on csexwb2 [18], which is capa-
able of executing dynamic content embedded in each webpage. Our instrumented browser
records detailed information for a landing page and all webpages that are subsequently
loaded by visiting the landing page, where the landing page [90], which is also known
as the root page in other literature [50], is referred to as the first webpage rendered by a
browser. To be more specific, in addition to taking a snapshot of the landing page after it
is fully loaded, the instrumented browser will keep the URL and the source code (or the
image content) of each webpage.

In order to get the ground-truth fake AV webpages, we instruct our browser to col-
lect fake AV webpages based on search engine results. We first obtain a set of security-
related keywords from a keyword research tool by WordStream [83], which provides over
a trillion keywords in their database for Search Engine Optimization (SEO). We then select
keywords relevant to virus or anti-virus, and enter a keyword together with a randomly gen-
erated word into mainstream search engines such as Google. Upon receiving search results,
our browser will visit the top 200 search results (i.e., the landing page URLs) and record
their information, yielding a set of webpages denoted as $S_{\text{google}}$. $S_{\text{google}}$ totally contains
18,970 unique landing pages. Aiming at labelling fake AV webpages, we manually inves-
tigate the snapshot of each landing page and preserve those that encourage users to install
anti-virus software. Among them, we follow two steps to identify fake AV webpages. First,
by consulting public domain reputation services such as Malware Domain List [51]
and SURBL Lists [76], if the domain name of a landing page has been detected as suspicious, we will label this landing page as a fake AV webpage. For remaining webpages, we manually download the executable(s) from each landing page (if applicable) and leverage a collection of malware detection engines such as VirusTotal [78] and ClamAV [40] to detect malicious binaries. A landing page will be labeled as a fake AV webpage if any of its associated executable(s) is classified as malware. Such practice has resulted in a set of total 1,230 fake AV landing pages, which is denoted as $S_{fake}$ ($S_{fake} \subseteq S_{google}$).

Next, we prepare the benign dataset. Our benign dataset is composed of three types of webpages. First, the benign dataset contains those popular security-related webpages from our search engine queries. These webpages are likely to be returned to users through various search engines given their high popularity. To derive this part, we reuse the $S_{google}$ dataset. Specifically, if the domain of a webpage in $S_{google} - S_{fake}$ appears in Alexa [3] top 100,000 most popular sites, this webpage will be considered as a popular benign webpage and then aggregated into $S_{security-popular}$. Through this approach, we have identified 210 popular webpages (i.e., $|S_{security-popular}| = 210$). Webpages in $S_{security-popular}$ are mainly composed of popular authentic anti-virus webpages and security-related articles (e.g., blogs). Second, the remaining webpages in $S_{google}$ will be aggregated into $S_{security-unpopular}$ (i.e., $S_{security-unpopular} = S_{google} - S_{security-popular} - S_{fake}$), where $|S_{security-unpopular}| = 17530$. These webpages represent those security-related webpages whose popularity is relatively low. For example, many of webpages in $S_{security-unpopular}$ are unpopular digital forums in which security topics are discussed. Third, our benign dataset needs to contain those webpages that are irrelevant to security. To this end, we again resort to Alex [3] list to visit 538 pages that are randomly selected from top 10,000 most popular webpages that are irrelevant to security, where these webpages are aggregated into $S_{security-irrelevant}$ (i.e., $|S_{security-irrelevant}| = 538$).
### 3.3.2 Discovering Trendy and Diverse Security Keywords

*DART* follows three steps to identify security keywords. It starts with a set of “seed” security keywords from public encyclopedia repositories, which maintain a category of information related to popular security vendors and their products. The repository leveraged by *DART* is Open Directory Project [60], one of the most popular public encyclopedia repositories, where *DART* enumerates all words belonging to the directory of “*Malicious Software:*:Detection and Removal Tool”. Then we manually eliminate over-generic words (e.g., “application”) as well as stopping words (e.g., “and” and “of”) and aggregate the remaining ones to a set denoted as $W_{\text{seed}}$. As these repositories are well maintained, words in $W_{\text{seed}}$ are likely to imply core and stable concepts in the context of anti-virus products and security. Unfortunately, $W_{\text{seed}}$ might be insufficient due to two facts. First, the public encyclopedia repositories may lack comprehensiveness, thereby missing certain important keywords. Second, the repositories might be updated infrequently, and hence words in $W_{\text{seed}}$ could fail to capture trendy security keywords promptly.

Next, *DART* leverages DISCO [41], a public API capable of retrieving the semantic similarity between arbitrary words, to discover more diverse security keywords by expanding $W_{\text{seed}}$. DISCO takes advantage of English Wikipedia to explore the semantic similarity between two words. Specifically, given each security keyword $w$ in $W_{\text{seed}}$, we use DISCO

### Table 3.1: Examples of security keywords discovered in each step; Reprinted from [38]

<table>
<thead>
<tr>
<th>Seed Security Keywords</th>
<th>Expanded by DISCO</th>
<th>Expanded by LSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>Cybercrook</td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td>Typosquatting</td>
<td></td>
</tr>
<tr>
<td>Anti-spyware</td>
<td>RogueAntivirus</td>
<td></td>
</tr>
<tr>
<td>Anti-spam</td>
<td>Zombiecomputer</td>
<td></td>
</tr>
<tr>
<td>Rootkits</td>
<td>Maladvertising</td>
<td></td>
</tr>
<tr>
<td>Backdoor</td>
<td>Snoopware</td>
<td></td>
</tr>
<tr>
<td>Privacy</td>
<td>Ransomware</td>
<td></td>
</tr>
<tr>
<td>KeyBoy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
API to derive top 10 words that are most similar to \( w \), and then aggregate them together with \( w \) into \( W_{\text{disco}} \).

Third, since social networks have become a dominating way for information dissemination, \textit{DART} uses Twitter to mine trendy and more diverse security keywords by performing \textit{Latent Semantic Analysis} (LSA) \cite{24}. The LSA method measures the semantic similarity by applying singular value decomposition (SVD) \cite{52} on a term-document matrix. \textit{DART} collected a set of 10.9 million tweets using the public streaming API offered by Twitter. It filtered out stopwords in each tweet. It then constructed a term-document matrix for Latent Semantic Analysis, where the “term” and “document” are corresponding to “word” and “tweet”, respectively. The high co-occurrence of two words in the matrix indicates that they are likely to share similar semantic meaning. \textit{DART} enumerates all words that exhibit high semantic similarity to words belonging to \( W_{\text{disco}} \). Specifically, we calculated semantic similarity between any two words using cosine similarity and set a threshold \( \theta \) value to 0.75. We obtained this threshold by performed empirical evaluation: we tested thresholds between 0.5 and 0.95 with a 0.05 increment and found that most of the security-related words in a corpus of sampled tweets were successfully retrieved at the threshold of 0.75.

### 3.3.3 Designing Detection Features

In this section, we explore various features and demonstrate their effectiveness on discriminating fake AV webpages from benign ones. The proposed features are categorized into three classes, which characterize three aspects critical to the success of a fake AV attack respectively.

- **Human-Perception Features**: Fake AV webpages need to impersonate identities of authentic anti-virus products. Features in this class therefore characterize textual and visual identities that are used by attackers to trick users into trusting the authenticity
of the fake AV webpages.

- **Search Engine Optimization (SEO) Features**: Search engines become a major method for attackers to disseminate fake AV webpages. Hence, fake AV webpages actively promote their rankings in search engine results. Features in this class profile webpage elements (e.g., security keywords in URL and webpage source code) that can be used for boosting the search rankings of fake AV webpages.

- **Networking Features**: Attackers actively increase the resilience of their networking infrastructures against legal disruption efforts. Features in this category describe how attackers employ service redirection and domain-/fast-flux to improve the agility of their network infrastructures used by fake AV webpages.

### 3.3.3.1 Human-Perception Features

In our current implementation, *DART* bootstraps from a $W_{seed}$ with 51 words. The DISCO API enlarges the set to 510 words. By performing LSA analysis on a large corpus of 10.9 million tweets, *DART* yields 1050 security keywords (i.e., $W_{security}$). Table 3.1 presents examples of security keywords that are discovered in the three steps, respectively. Specifically, the keywords (i.e., “*firewall*”, “*privacy*”, and, “*anti-spyware*”) triggered by DISCO method present the most widely stuffed keywords within a fake AV webpage. The trendy security keywords (e.g., “*KeyBoy*”, “*scareware*”, and “*typosquatting*”) might reflect trendy topics in which Internet users are interested with respective to fake AV software or cyber attacks.

Since the success of a fake AV attack requires the explicit approval of an end user, fake AV webpages need to convince end users that they offer authentic anti-virus products. Practically, attackers usually embed the textual and visual identities of the impersonated anti-virus products in fake AV webpages. In addition, names of emerging malware threats are also adopted by attackers to “scare” end users [67] and stimulate their installation ac-
Figure 3.4: Examples of icons from authentic AV software; Reprinted from [38]

Figure 3.5: Image similarity between an authentic McAfee image (Left) and fake one (Right); Reprinted from [38]

Accordingly, we propose three features to characterize such observations.

- **Feature 1: Image Identity.** This feature quantifies the extent to which the images loaded by a landing page are similar to those images of the authentic anti-virus webpages. Specifically, \( DART \) aggregates all images loaded by a landing page into a set denoted as \( V_{\text{web}} \). Given a set of images \( (V_{\text{auth}}) \) composed of authentic anti-virus logos or icons, \( DART \) enumerates each pair of images \( v_i \) and \( v_j \) where \( v_i \in V_{\text{web}} \) and \( v_j \in V_{\text{auth}} \), and then computes their similarity score using an image similarity function denoted as \( \text{similar}(\cdot) \). The visual identity feature for a landing page is represented by the maximum value of similarity scores.

Since the visual identities (e.g., the icons) of authentic anti-virus webpages are relatively stable, it is practically feasible to manually collect their representative images. Specifically, we used Wikipedia to collect the logos and icons of all anti-virus companies according to the Open Directory Project [60], resulting in 56 images belonging to \( V_{\text{auth}} \) since it guaranteed that we can find the high resolution images which are identi-
cal with those from authentic anti-virus webpages. Fig. 3.4 presents several examples. Since images loaded by landing pages (i.e., images belonging to $V_{\text{web}}$) usually have poorer resolutions compared to those from authentic anti-virus webpages, we use a normalized RGB based histogram [29] and Bhattacharyya measure [7] to calculate a similarity score between the probability distribution of images. Fig. 3.5 presents an example of image similarity measurement: even if two images of the same visual identity exhibit disparate resolution, $DART$ can still yield a high similarity score of approximately 0.9345. Fig. 3.6 presents the distribution of values for the image identity feature with respect to fake AV, benign (security-popular), benign (security-unpopular), and benign (security-irrelevant): more than 90% benign (security-popular) have feature values higher than 0.97, and fake AV landing webpages also have high similarity value 0.88; comparatively, the majority of benign (security-irrelevant) and benign (security-unpopular) have small values for this feature. We assumed that fake AV attackers can copy images from third parties or authentic AV webpages and reused them on their local hosting. Thus, it is not surprising that the feature value for fake AV webpages have high similarity value. Despite this, it is worth noting that this feature takes advantage of differentiating between fake AV webpages and legitimate ones (i.e., webpages in $S_{\text{security-irrelevant}}$ or $S_{\text{security-unpopular}}$) in terms of image similarity.

Commercial webpages usually embed their textual identities in the domain names of their landing pages, and authentic anti-virus webpages are not exceptions. Particularly, security vendors usually position their textual identities in their second-level domains. The first row of Table 3.2 presents a few domain names from several authentic anti-virus landing pages, where their textual identities such as “avast” and “norton” reside in the second-level domains. Attackers can leverage the same way for deception. However, unlike authentic anti-virus webpages, fake AV webpages tend to embed textual identities in lower-level domains for a number of reasons. First, the second-level domain with textual identities usually have been registered by security vendors. Second, many fake AV webpages are hosted
in public hosting service [10], [70], [71], and their second-level domains are managed by various hosting services. Third, mobile devices such as tablets and smart-phones usually have limited space in browser address bar to display domains, and hence the low-level domains are more likely to be presented to end users. Attackers can take advantage of such limitation and deliberately embed textual identities in low-level domains to mislead mobile device users [4]. The second row of Table 3.2 illustrates a few examples of fake AV landing pages domains, where the textual identities such as “avast” and “norton” appear in the 3- or 4-level domains. In addition, textual identities are also frequently presented in the texts and images of fake AV landing pages after they are fully loaded. We therefore design two features, including domain identity and content identity to characterize our observations.

- **Feature 2: Domain Identity.** We split the domain name of a visited landing page into tokens by the delimiter “.”, where each token defines a single level of the domain name and the level increases from right to left. The rightmost level is corresponding to the 1-level domain (a.k.a., top-level domain). We identify all tokens that contain any word in \( W_{\text{security}} \) and then accumulate their levels as the value for
Feature 3: Content Identity. Although we can directly extract security keywords from preserved source codes for all webpages associated with a landing page, many security keywords are dynamically generated (e.g., by JavaScript) or actually presented in images. Therefore, DART performs Optical Character Recognition [59] analysis on the snapshot of the fully loaded landing page and extract all words. The value of this feature represents the total occurrence of words that belong to $W_{security}$.

Fig. 3.7 presents the distribution of the values of the domain identity feature for fake AV, benign (security-popular, security-unpopular, and security-irrelevant) webpages, respectively. As indicated in the distribution, more than 93% fake AV webpages have feature values higher than 15 while larger than 97% benign (security-popular), benign (security-unpopular), and benign (security-irrelevant) have their domain features lower than 5. It showed that most fake AV webpages indeed commonly embed security keywords in their
lower-level domains. Fig. 3.8 compares content identity feature for fake AV webpages, benign (security-popular), benign (security-irrelevant), and benign (security-unpopular) webpages, where fake AV webpages present much more security keywords to end users compared to other webpages.

3.3.3.2 Search Engine Optimization (SEO) Features

While it is critical for attackers to trick users into installing fake anti-virus software, how to ensure a fake AV webpage has the chance to be visited by end users is equally important. Search Engine Optimization (SEO) is widely used by attackers to accomplish this objective. As search engines often give heavy weight for keywords in both URLs and contents of webpages, it becomes a common practice to embed in them with security keywords [36]. We accordingly design two features to characterize such observation.

- Feature 4: Path Keywords. DART divides the path of a URL into tokens using the delimiter “/” from left to right, where each token usually represents a directory in the web server. For example, “Spyware” resides in the 2nd level of the fake AV
URL “www.xyz.com/Antivirus/Spyware/Dist/worm.html”. Second, we accumulate the levels of directories that contain security keywords from $W_{\text{security}}$. For instance, suppose $\{\text{antivirus, spyware, worm}\} \subset W_{\text{security}}$, the value of this feature for the aforementioned URL will be 7 since $1^{st}$, $2^{nd}$, and $4^{th}$ directories contain “Antivirus”, “Spyware”, and “worm”, respectively.

- **Feature 5: Content Keywords.** Words in the webpage source codes are commonly employed for search engine for the webpage indexing. Attackers excessively tend to inject words of various security semantics into the webpage source codes, which can be easily analyzed by search engines. This feature represents the occurrence of words belonging to $W_{\text{security}}$ in the source codes of a landing page and all other webpages loaded by it.

Fig. 3.9 and Fig. 3.10 present the distribution of these two features, respectively. As indicated in these figures, fake AV webpages tend to have much more security keywords in both URL paths and webpage source codes.

### 3.3.3.3 Networking Features

Attackers strive to make their networking infrastructures resilient against legal disruption efforts. Two networking-level strategies are commonly adopted. First, attackers usually use service redirection (by either standard HTTP redirection protocols or executable scripts) to obfuscate the locations of web servers that actually offer malicious services. Specifically, upon visiting a fake AV landing page, the browser might be redirected through a chain of redirection servers before it reaches at the server that hosts malicious contents such as the malicious binaries for download. Compromised hosts (e.g., bots), which are distributed over the Internet, are often employed by attackers as redirection servers [90] [43]. Consequently, a fake AV landing page may redirect the browser to traverse a large number of networks, which can be characterized by networking Autonomous System Numbers.
(ASNs), before its content is fully loaded. Second, fast- and domain-flux techniques are widely used to improve the agility of malicious networking infrastructures [90]. On the one hand, a collection of IP addresses can be registered for the same fake AV domain, and thus even if a portion of the IP addresses (and their associated servers) are disrupted, the remaining ones can still provide malicious services. On the other hand, a set of fake AV landing pages controlled by the same attacker can be hosted in the same server (i.e., the domain names will be mapped to the same IP address). In this case, even if certain domains are taken down, others can still function. Therefore, if two domains (say $d_i$ and $d_j$) corresponding to two fake AV landing pages are under the control of the same attacker, it is highly likely that their resolved IPs (say $IPSet_i$ and $IPSet_j$) tend to have certain overlap (i.e., $|IPSet_i \cap IPSet_j| > 0$). In other words, if $d_i$ has been identified as a domain used for a fake AV attack, its maliciousness can be “propogated” to $d_j$ due to the overlap of their IP addresses. Moreover, a large IP-set overlap between $d_i$ and $d_j$, which can be measured by $|IPSet_i \cap IPSet_j|/|IPSet_i \cup IPSet_j|$, indicates a high possibility that $d_j$ is used for malicious purpose. In this section, we propose two features that aim at profiling the service redirection and quantifying the maliciousness propagation.

- **Feature 6: Redirection.** For each webpage that is triggered by rendering the landing page, DART identifies the IP address(es) for the domain name in its URL and subsequently acquires the Autonomous System Number(s) (ASN) for the IP address(es) using public IP-to-ASN services [20]. This operation will result in a set of ASNs, which is denoted as $ASN_{All}$. If we denote the set of ASN(s) for the landing page as $ASN_{Landing}$ ($ASN_{Landing} \subseteq ASN_{All}$), then the value of this feature is defined as $|ASN_{All} - ASN_{Landing}|/|ASN_{All}|$.

- **Feature 7: Maliciousness Score.** An increasing number of domains are created and registered by attackers for malicious users such as fake AV webpages. The malicious score is to quantify the maliciousness of a collection of observed domains (denoted as $D_{observed}$) given their correlation with a set of known fake-AV domains (denoted
as $D_{seed}$). We start by merging $D_{observed}$ and $D_{seed}$ into a set of domains denoted as $D = \{d_1, d_2, \ldots, d_n\}$. We then collect the IP address(es) for each domain $d_i$, we collect its IP addresses (denoted as $IPSet_i$) by issuing DNS queries. We then construct a matrix $R = \{r_{i,j}\}_{n \times n}$, where $r_{i,j} = |IPSet_i \cap IPSet_j|/|IPSet_i \cup IPSet_j| \in [0, 1]$. This matrix can be considered as an undirected graph $G$, where each node is corresponding to a domain. Two nodes, $d_i$ and $d_j$, will share an edge if $r_{i,j} > 0$, where $r_{i,j}$ can represent the weight for this edge. Accordingly, $R^d$ quantifies the aggregated weights for two nodes (i.e., domains) to be connected through exact $d$ hops. We initialize a vector $S_0 = \{s_i\}_{n \times 1}$, where $s_i = 1$ if $d_i \in D_{seed}$ and $s_i = 0$ otherwise.

$$S_k = \sum_{d=1}^{d=k} R^d \cdot S_0 \quad (3.1)$$

$R^d \cdot S_0$ indicates that we propagate the maliciousness of known fake AV domains to other domains that are $d$ hops away from them. Hence, $S_k$ accumulates the maliciousness score of a domain that is propagated from known malicious domains that are at most $k$ hops away. If the maliciousness score for a domain exceeds 1, we will set it as 1 to prevent the maliciousness score from growing to infinity.

Fig. 3.11 shows the distribution of the redirection feature for fake AV, benign (security-popular), benign (security-unpopular), and benign (security-irrelevant) webpages respectively. The comparison indicates that fake AV webpages indeed involve more diverse networks for redirection than other webpages. Certain benign (security-popular), benign (security-unpopular) and benign (security-irrelevant) have the feature values greater than 0.8, where our manual analysis revealed that some of benign (security-popular), benign (security-unrelevant), and benign (security-unpopular) massively integrated advertising contents from numerous networks.

In order to evaluate the maliciousness score feature, we need to first obtain $D_{seed}$. 


which contains domains that have been known to be used for fake antivirus webpages. However, since $S_{fake}$ represents the ground truth, we have known that all domains involved in the landing pages belonging to $S_{fake}$ are used for fake AV webpages. In order to evaluate the effectiveness of this feature, we pretend that only a portion of landing pages domains in $S_{fake}$ are known to be malicious. Specifically, we randomly selected 10% of the landing pages in $S_{fake}$ and aggregate their domains into $D_{seed}$. The domains of remaining 90% landing pages are aggregated into $D_{observed}$. Fig. 3.12 presents the distribution of the maliciousness score feature for fake AV, benign (security-popular), benign (security-irrelevant), and benign (security-unpopular) webpages given $D_{seed}$, where we set $k = 10$. As indicated in the figure, approximately 90% of benign (security-popular), benign (security-unpopular), and benign (security-irrelevant) webpages have maliciousness scores lower than 0.2 for their domains. Comparatively, domains of fake AV webpages tend to have overlaps with each other with respect to the sets of IP addresses, thereby resulting in high maliciousness scores for their domains. For example, more than 60% of fake AV webpages have maliciousness scores higher than 0.2.
3.4 Evaluation

We have performed extensive evaluation of DART, which focuses on the overall detection performance, the detection accuracy when only light-weight features are employed, the relative importance of the designed features, and the correlation among these features.

3.4.1 Experimental Setup

Since DART adopts statistical classifiers to integrate various features for detection, evaluating DART implies that a dataset has relatively balanced benign and malicious samples. To this end, we use all 1,230 fake AV webpages in \( S_{fake} \) as malicious samples. The total number of benign samples (i.e., \( |S_{security-popular} \cup S_{security-unpopular} \cup S_{security-irrelevant}| \)) is much larger than \( |S_{fake}| \), where \( S_{security-unpopular} \) contribute most of the benign samples. Therefore, we randomly sample 538 webpages from \( S_{security-unpopular} \), where 538 is actually the size of \( |S_{security-irrelevant}| \). We then aggregate \( S_{security-popular}, S_{security-irrelevant}, \) and these 538 webpages randomly selected from \( S_{security-unpopular} \) into \( S_{benign} \). Hence, \( S_{benign} \), together with \( S_{fake} \), forms an approximately balanced evaluation dataset (i.e., \( |S_{benign}| = 1286 \) and \( |S_{fake}| = 1,230 \)), and meanwhile experiences significant diversity.

3.4.2 Experimental Results

3.4.2.1 Detection Accuracy

We equipped DART with the Random Forest statistical classifier and evaluated its detection accuracy. We utilized 10-fold cross-validation, where we randomly partition the data set into 10 folds and then employ 9 folds for training and the remaining 1 fold for fake AV webpages detection. The receiver operating characteristic (ROC) is presented in Fig. 3.13 and it demonstrated that the features used by DART can achieve high detection accuracy when
they are integrated by a statistical classifier for detection. For example, given a false positive rate of 0.2%, the Random Forest classifier can achieve a high detection rate of 90.4%.

To measure DART that can distinguish fake AV from security-popular \( (S_{\text{security-popular}}) \) webpages, we check the false positive rate for security-popular and fake AV webpages. Note that the false positive rate is defined as the fraction of all security-popular webpages classified as fake AV ones. The low false positive rate in Table 3.3 is achieved by the Random Forest using all the features or light-weight features on the dataset \( S_{\text{fake}} \) and \( S_{\text{security-popular}} \). The result indicates that the 0.4% of false positive rates are acceptable for DART to detect fake AV webpages in the Internet. We also evaluated the sensitivity of DART to the selection of statistical classifiers. Specifically, we used both Support Vector Machine (SVM) and Gradient-Boosted Tree [31] to perform 10-fold cross-validation. Both classifiers accomplished high detection accuracy comparable to the Random Forest classifier, where their area under the ROC curve (AUC) values are summarized in Table 3.4. This experimental result demonstrated that proposed features are not sensitive to the selection
### Table 3.3: False positive rates for Security-popular and Fake AV webpages; Reprinted from [38]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Light-Weight only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>0.004</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
</tr>
</tbody>
</table>

### Table 3.4: AUC for each classifier; Reprinted from [38]

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.997</td>
</tr>
<tr>
<td>Gradient-Boosted Tree</td>
<td>0.998</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.999</td>
</tr>
</tbody>
</table>

3.4.2.2 Detection Accuracy Using Light-Weight Features

Although the integration of these features results in high detection accuracy, extracting certain features actually implies high computational costs and may introduce significant delays when a large number of webpages are analyzed. Specifically, the image identity feature mandates the similarity measurement between each icon from all authentic anti-virus webpages and every image of an unknown webpage; the content identity requires OCR analysis for each unknown webpage, which is computationally expensive; calculating the value of the IP-overlap feature needs to perform pairwise IP-set interaction between each labelled webpage and the unknown webpage as well as iterative matrix multiplication. As a consequence, it may become a performance bottleneck when DART is used for the environments where efficiency becomes a concern (e.g., used by mainstream search engines to analyze a massive number of webpages).

In order to address this challenge, DART can selectively employ light-weight features for detection. Specifically, the domain identity, path keywords, and content keywords only require text-based comparison while AS redirection merely needs ASN lookups. We have evaluated the detection accuracy of DART when only these features are used and presented the detection result in Fig. 3.13. Despite the fact that the light-weight features lead to slightly lower detection accuracy (e.g., 87.5% detection rate given the false positive rate of
0.2%) compared to all features, the time used for detection has been tremendously reduced. To be specific, the average detection time has been reduced from 5.5 minutes to 36.8 seconds for each webpage. In addition, if a false positive rate of 0.5% is tolerable, we can obtain approximately 90% detection rate.

In spite of the huge reduction for time consumption, 36.8 seconds are not negligible. It is mainly attributed to our implementation, which is not fully optimized. For example, all features are sequentially generated; keyword identification is implemented using the most basic substring matching; identifying ASNs relies third-party IP-to-ASN service where network interactions are involved. We present the percentage of time consumption for each light-weight feature in Table 3.5, where Redirection dominates the time consumption (i.e., about 84% of total average computation time) since network interactions are involved. Nevertheless, these light-weight features enable significant room for performance improvement. For example, a fast substring matching algorithm and a local IP-to-ASN database would fundamentally decrease the time consumption. We will take the performance optimization as our future work.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Variable importance</th>
<th>Decrease of detection rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Identity</td>
<td>7.981</td>
<td>0.537</td>
</tr>
<tr>
<td>Content Identity</td>
<td>6.178</td>
<td>0.418</td>
</tr>
<tr>
<td>Content Keywords</td>
<td>5.401</td>
<td>0.365</td>
</tr>
<tr>
<td>Path Keywords</td>
<td>5.297</td>
<td>0.358</td>
</tr>
<tr>
<td>Maliciousness score</td>
<td>3.272</td>
<td>0.221</td>
</tr>
<tr>
<td>Image Identity</td>
<td>2.796</td>
<td>0.139</td>
</tr>
<tr>
<td>Redirection</td>
<td>1.182</td>
<td>0.080</td>
</tr>
</tbody>
</table>

Table 3.6: Rank of feature importance in DART; variable importance is measured as the mean decrease in accuracy and features with large variable importance are more influential for classification of the data; Reprinted from [38]

Table 3.5: A percentage of average computation time for each of light-weight features; Reprinted from [38]
3.4.2.3 Feature Importance and Correlation

We explored the relative importance of the proposed features in the context of Random Forest classifier, which has accomplished the best detection accuracy according to our experiments. Specifically, we conducted variable importance measurements [9] for our features using permutation test of Random Forest function for feature selection [45]. The variable importance for each feature is presented in the second column of Table 3.6. In addition, we evaluate the decrease of the detection rate (given a fixed false positive rate of 0.2%) of DART in absence of each feature, which is presented in the third column of Table 3.6. Furthermore, the results have demonstrated that the light-weight features tend to contribute more to the detection accuracy. Specifically, Domain Identity and Content Keywords features demonstrate high detection ability since the decrease rates of the two features (0.537 and 0.365, respectively) indeed rank 1st and 3rd among all features.

We also evaluated the correlation among various features, where the correlation implies the extent to which a feature might be redundant given other features. Two strategies...
have been adopted. First, we used the correlation matrix to identify if correlated predictors exist in the features, where each element in the matrix represents the Pearson’s $r$ correlation coefficient [44] of two features. The Pearson’s correlation coefficient $r \in [-1, 1]$ of two features $P$ and $Q$ can be defined as

$$r = \frac{\sum (P - \bar{P})(Q - \bar{Q})}{\sqrt{\sum (P - P)^2} \sqrt{\sum (Q - Q)^2}}$$

(3.2)

where $\bar{P}$ and $\bar{Q}$ denote the means of the two features. Fig. 3.14 shows that the majority of features are not linearly correlated to other features. For example, a pair of two features (Path Keywords, Domain Identity) represents that the highest negative correlation score is 0.39 and the highest positive correlation between Content Keywords and Domain Identity is at most 0.15.

Second, we performed Principal Component Analysis (PCA), which can be used to analyze correlation between variables with respect to the variance of the data [37]. We presented our evaluation results using PCA variable factor map [64] in Figure 3.15. In a factor map, an arrow represents a feature. The angle between the two arrows of features explains the relative correlation of the respective features on the first two principal components. Specifically, if the angle between the two arrows is close to 90 degrees, the corresponding features are not correlated. As illustrated in Figure 3.15, the angles between majority of features approximate 90 degrees (e.g., Content Identity and Redirection onto the principal component 2 space), implying a weak correlation between features. According to the correlation matrix and PCA variable factor map, we concluded that most of the features tend to be linearly independent.
3.5 Discussion

Attackers may attempt to evade our detection after they know the design of DART, which represents a general challenge for all detection systems. Specifically, attackers can deliberately design their fake AV webpages so that the feature values of these webpages are indistinguishable from those of benign ones. However, since DART’s detection features characterize elements of fake AV webpages that are critical to the success of attacks, manipulating such elements may fundamentally constrain the attack effectiveness. For example, attackers can significantly reduce visual and textual identities in fake AV webpages to disrupt the human-perception features, while it would decrease the possibility to convince end users to install fake anti-virus binaries. In addition, attackers can also abandon SEO techniques to make the SEO features ineffective. Unfortunately, this strategy will tremendously lower the ranks of fake AV webpages in search engine results, thereby intrinsically diminishing their exposure to end users. Further, attackers could organize a fake AV
landing page and its associated webpages in the hosts belonging to the same autonomous system. Unfortunately, achieving this objective will impair the agility and robustness of their networking infrastructures against legal disruption efforts. Attackers may also partition different landing pages to distinct network infrastructures that share little overlap of IP addresses. Nevertheless, it necessitates a large amount of network resources (e.g., IP addresses) from attackers especially when the number of fake AV landing pages grows.

Although our goal of this work is focused on detecting fake AV webpages, the proposed methods and features could be adopted to detect other types of malicious webpages. Specifically, since the networking features are independent to webpage content, they can be used to detect all types of malicious webpages. Other features are relevant to the context of antivirus software. For example, all keywords are related to virus or antivirus; all images are related to icons of popular antivirus software. Therefore, directly using these features to detect other malicious webpages is unlikely to work. However, one can modify the seed data (e.g., the keywords and icons) to detect other types of malicious webpages. For example, we can use the keywords and icons related to bank, in order to detect phishing websites related to finance. In addition, the method to propagate maliciousness score from known malicious domains to unknown domains based on their correlation can be used to detect malicious domains beyond fake AV webpages. For example, we can use this method to detect botnet C&C domains that use domain-flux [87] techniques.

Similar to many methods of detecting malicious webpages, DART can be deployed as a crawler to detect fake AV webpages proactively, helping a variety of critical Internet services such as search engine and social networks. Although the current average detection time (36 seconds) makes DART unlikely to be directly deployed in end users’ browsers, we still envision such possibility after optimizing its implementation (e.g., parallelizing feature extraction and using fast algorithms). Systematically reducing system overhead and increasing scalability are definitely of great importance for detection systems and improving DART towards this direction falls in our future work.
3.6 Summary

This chapter presents a novel system, DART, to automatically detect fake AV webpages. DART leverages three categories of features including human-perception features, SEO features, and networking features. Deriving these features necessitates our knowledge of diverse and trendy security keywords, which has been accomplished by performing semantic analysis based on messages from social networks. Experimental results based on real fake AV webpages collected from the Internet have demonstrated the detection accuracy of DART, which has achieved a high detection rate of 90.4% given an extremely low false positive rate of 0.2%. In addition, statistical analysis has revealed that the proposed features complement to each as they tend to be linearly independent. Further, Since these features characterize the elements in a fake AV webpage that is critical to its success, circumventing these features will fundamentally limit the effectiveness of fake AV attacks.

In the future work, we plan to extend DART more scalable to analyze large-scale fake AV webpages whose the volume of features keeps growing over time. In particular, the possible direction is to use a distributed computing technique such as MapReduce to carry out big-data processing. Another option would be valuable to design a novel automated browser with respect to labelling fake AV webpages and downloading binary files. Lastly, we will explore new approaches to detect foreign languages fake AV webpages as part of our future work.
Chapter 4: Detecting Malicious Accounts for Online Promotions in Social Network

4.1 Motivation

In the chapter 4, we will discuss a new challenge to detect malicious OSN accounts that participate in online promotion events. Since OSNs provide a platform to support not only socializing but also sharing of users’ information, the OSN application is also becoming increasingly popular. Specifically, an OSN account which integrates with financial activities is commonly associate with accounts for both online banking and virtual currency. Figure 4.1 presents such an example, where a QQ account, the most popular OSN account of Tencent in China, is associated with an online banking account for real currency and a virtual account for virtual currency (i.e., Q coin). As shown in Figure 4.1, an OSN user (i.e., a QQ user) has two methods to recharge his/her virtual currency account: i) online banking and ii) online promotion event. Through online banking services, the OSN user can directly deposit real currency into his/her online banking account and then he/she can recharge his/her virtual currency account from his/her banking account. Alternatively, the OSN user who participates online promotion events, can also recharge his/her virtual cur-
currency account by collecting rewards from the promotion events. Then, the user can expend from his/her account in two typical ways. First, he/she can use real or virtual currency to buy both real and virtual goods (e.g., via online shopping). Second, he/she can transfer both real and virtual currency to another user by sending out gifts.

As discussed in Section 1.2, attackers could control a large number of accounts, either by registering new accounts or compromising existing accounts, to participate in the online promotion events for virtual currency. We found that the security risk posed by the use of OSN accounts in terms of online promotion events is feasible in general. Figure 4.2 presents the typical virtual currency flow when malicious accounts participate in online promotion events. The flow is composed of three phases including i) collecting, ii) multi-layer transferring, and iii) laundering the virtual currency. In first phase, an attacker controls a set of accounts to participate in online business promotion activities and each account possibly gets a certain amount of virtual currency as return. In the second phase, the attacker will instrument these currency-collection accounts to transfer the virtual currency to other accounts. Multiple layers of transferring activities might be involved to obfuscate the identities of malicious accounts used for participating online promotion activities. At the end of the second phase, a large amount of virtual currency will be aggregated into a few
laundering accounts. In the third phase, the attacker will control the laundering accounts to trade the virtual currency into real cash by selling it to individual buyers. Attackers usually employ two methods to solicit individual buyers including sending spams and advertising through major e-commerce websites such as www.taobao.com and www.tmall.com. In order to compete with regulated sources for virtual currency (i.e., purchasing virtual currency using real currency), attackers usually offer a considerable discount. Our objective is to design a detection system capable of identifying malicious accounts that participate in online promotion events for virtual currency collection (at the collection phase) before rewards are committed.

In Section 4.2, we remark that the existing techniques may not effective for identifying such malicious OSN accounts with the high detection accuracy.

### 4.2 Related Work

We review some of the existing works on spammer detection across online social networks (OSNs). Effective extracting of detection features from the sources for spammer detection plays an important role in OSNs. Therefore, selection of appropriate features is vital as it could lead to detection performance. In this section, we also review some of the currently
available financial fraud detection methods.

### 4.2.1 Spammer Detection

Since online social networks play an increasing important role in both cyber and business world, detecting malicious users in OSNs becomes of great importance. Many detection methods have been consequently proposed [35, 34, 14, 13, 25, 12, 86, 54]. Considering the popularity of spammers in OSNs, these methods almost exclusively focus on detecting accounts that send malicious content. A spamming attack can be considered as an information flow initiated from an attacker, through a series of malicious accounts, and finally to a victim account. Despite the diversity of these methods, they generally leverage partial or all of three sources for detection including i) the content of the spam message, ii) the network infrastructure that hosts the malicious information (e.g., phishing content or exploits), and iii) the social structure among malicious accounts and victim accounts. For example, Gao et al. [28] designed a method to reveal campaigns of malicious accounts by clustering accounts that send messages with similar content. Lee et al. [43] devised a method to first track HTTP redirection chains initiated from URLs embedded in an OSN message, then group messages that lead to webpages hosted in the same server, and finally use the server reputation to identify malicious accounts. Yang et al. [88] extracted a graph from the “following” relationship of twitter accounts and then propagate maliciousness score using the derived graph. Wu et al. [86] proposed a social spammer and spam message co-detection method based on the posting relations between users and messages, and utilized the relationship among user and message to improve the performance of both social spammer detection.
4.2.2 Fraud Detection

Detecting fraudulent activities in financial transactions has also attracted significant research efforts [1, 81]. For example, Olszewski et al. [58] represented the user account records in 2-dimensional space of the Self-Organizing Map grid, and proposed a detection method based on threshold-type binary classification algorithm to solve problems of credit card fraud and telecommunications fraud. Lin et al. [48] ranked the importance of fraud factors used in financial statement fraud detection, and investigated the correct classification rates of three algorithms including Logistic Regression, Decision Trees, and Artificial Neural Networks. Throckmorton et al. [77] proposed a corporate financial fraud detection method based on combined features of financial numbers, linguistic behavior, and non-verbal vocal.

4.2.3 Limitations

Based on the literature survey presented here, the existing methods are limited to the following.

1. It is difficult that the proposed approaches for spammer detection in OSNs detect malicious OSN accounts participating in online promotion activities. Different from traditional spammer accounts, neither do the OSN accounts with financial activities utilize spamming messages, nor require any malicious network infrastructures to spread malwares and launch illicit activities automatically. In addition, these accounts don’t necessarily rely on social connection for i) users and users (i.e., user-user relations); ii) users and messages (i.e., user-message relations); and iii) OSN messages (i.e., message-message relations).

2. Compared to the studied financial fraud detection problems, account behaviors of collecting and using the virtual currency in online promotion activities are almost
completely different with traditional financial systems since they do not only involve financial activities but also networking and online promotion activities. In this case, it is hard to develop new and more robust features to detect any financial fraud based on the traditional detection solutions.

These present new challenges to directly making use of existing methods for effective malicious accounts detection in OSNs with online promotion events. To overcome those limitations, in this dissertation, we address a new problem caused by new trend of integrating online social networks and financial activities and propose a novel system, ProGuard to automatically detect malicious OSN accounts that participate in online promotion events.

4.3 System

4.3.1 Collecting & Labeling Data

We have collected labelled data from Tencent QQ, a leading Chinese online social network that offers a variety of services such as instant message, voice chat, online games, online shopping, and e-commerce. All these services support the usage of the Q coin, the virtual currency distributed and managed by Tencent QQ. Tencent QQ has a giant body of 899 million active QQ accounts with a reportedly peak of 176.4 million simultaneous online QQ users. Tencent QQ is one of the global leading OSNs that are actively involved in virtual-currency-based online promotion activities. We collect total 56,000 OSN accounts. Our data set is composed of 28,000 malicious accounts and 28,000 benign accounts, where all of these accounts are randomly sampled from the accounts that participated in Tencent QQ online promotion activities in August 2015. The labeling process starts from identifying laundering accounts (i.e., accounts that are associated with virtual currency spams and accounts that sell virtual currency in major e-commerce websites). Specifically, if an account transfers virtual currency to any account that engages in virtual-money laundering
activities, this account will be labeled as malicious. Such “trace-back” process may involve multiple layers of transferring, which is visualized at the bottom in Figure 4.2. It is worth noting that although both malicious and benign accounts are labelled based on their activities in Phase-2 (i.e., currency transferring) and Phase-3 (i.e., laundering), the data used for building the detection system are collected before the launch of the online promotion event. The reason is that the objective of our detection system is to identify malicious accounts before the rewards are committed. The top of Figure 4.3 presents the temporal relationship among the data collection process, online promotion events, and the account labeling process. Therefore, it is worth noting that an account may not have any historical financial activities (even for virtual currency collection activities) since it participates in the online promotion for the first time.

![Figure 4.3: The Architectural Overview of the System; Reprinted from [92]](image)

Although the aforementioned “trace-back” method is effective in manually labeling malicious accounts, using it as a detection method is impractical. First, it requires a tremendous amount of manual efforts for forensic analysis such as identifying suspicious virtual-currency dealers in external e-commerce websites, correlating spamming contents with user accounts, and correlating sellers’ profiles with user accounts. In addition, evidence for such forensic analysis will be only available after malicious accounts participate in online promotion events. Therefore, this data labeling process, if used as detection method, cannot
guide business entities to mitigate their financial loss proactively. In contrast, our method is designed to detect malicious accounts prior to the reward commitment. For each account, we collect a variety of information including 1) login activities, 2) a list of anonymized accounts that this account has sent instant messages to, 3) service purchase activities, 4) the recharging activities, and 5) the expenditure activities.

4.3.2 System Design

*ProGuard* is composed of two phases, namely the training phase and the detection phase. In the training phase, a statistical classifier is learnt from a set of pre-labelled malicious and benign accounts. In the detection phase, an unknown account will first be converted to a feature vector and then analyzed by the statistical classifier to assess its maliciousness. The bottom of Figure 4.3 presents the architectural overview of *ProGuard*. As a variety of statistical classifiers have been developed and widely used, designing features which is capable of discriminating between malicious accounts and benign accounts becomes of central focus. In this section, we will introduce various features and demonstrate their effectiveness on differentiating malicious accounts from benign ones. We propose three general guidelines to steer the feature design.

- **General Behaviors**: Benign accounts are usually used by regular users for variety of activities such as chatting, photo sharing, and financial activities. In contrast, malicious accounts are more likely to be driven by online promotion events. Therefore, the benign accounts tend to be more socially active compared to malicious accounts.

- **Currency Collection**: The malicious accounts under investigation focus on using online promotion activities to collect virtual currency. In contrast, benign users are likely to obtain virtual currency from multiple resources.

- **Currency Usage**: Attackers’ ultimate objective is to monetize the virtual currency. In contrast, benign users use their virtual currency in much more diversified ways.
4.3.2.1 General-Behavior Features

Malicious accounts tend to be less active compared to benign accounts with respect to the non-financial usage. Attackers usually control their accounts to only participate in online promotion activities. In contrast, benign accounts are more likely to engage in active interaction with other users.

- **Feature 1: The Ratio of Active Days.** This feature represents the ratio of the number of active days of an account for the passed one year. Specifically, if an account is logged in at least once for a day, this day will be labeled as “active” for this account.

Attackers usually login malicious accounts for participating in online promotion activities that involve virtual currency. Therefore, malicious accounts tend to be silent in the absence of online promotion activities. The availability of promotion activities is significantly influenced by timing and spatial factors. For example, promotion activities are intensive over holiday seasons, special dates, and regional events while occasionally available for other time periods. As a consequence, malicious accounts tend to be inactive generally. Comparatively, benign accounts are used by regular users and their logins are driven by the daily usage such as chatting and photo sharing. Many users configure their applications to automatically login upon the bootstrap of the underlying system (e.g., a smartphone),
which further facilitates volatility of benign accounts. Figure 4.4 presents the distribution of feature values for both malicious accounts and benign accounts. As illustrated in the figure, the vast majority of malicious accounts (i.e., approximately 98% of malicious accounts) are active for less than 20% of total days whereas only a small percentage of benign accounts (i.e., less than 20%) experience the same active level (i.e., being active for less than 20% of one year).

- **Feature 2: The Number of Friends.** This feature summarizes the number of friends for each account.

![Figure 4.5: Feature 2 - The Number of Friends for An Account; Reprinted from [92]](image)

As a common feature for almost all online social networks, each OSN account has a list of friends. It usually implies a considerable amount of user-user interaction for one user to add another one as her friend. It is common for a benign user to maintain a relatively lengthy friend list for various social activities such as chatting and photo sharing. In contrast, an attacker usually lacks the motivation to maintain a friend list since it contributes little to promotion participation but costs significant efforts such as solving captcha challenges. Figure 4.5 presents the distribution of values for this feature, where malicious accounts tend to have much less friends compared to benign accounts. Specifically, approximately 80% of malicious accounts have less than 40 friends while about 70% benign accounts have more than 200 friends.
• **Feature 3: The Number of Services Purchased By An Account.** This feature represents the total number of types of upgraded membership that an account has paid for through all possible methods.

![CDF of the Number of Services Purchased By An Account](image)

Figure 4.6: Feature 3 - The Number of Services Purchased By An Account; Reprinted from [92]

It is a common feature in many online social networks that an user can upgrade his/her account by making a certain amount of payment through various ways such as credit card, wire transfer, and virtual currency. In the Tencent dataset, we consider 8 types of most popular upgraded membership including QQ VIP, Qzone, SVIP, QQ Music, Hollywood VIP, QQ Games, QQ books, and Tencent Sports. An upgraded account can a wide range of paid benefits such as advanced capabilities for an online game avatar, enriched decoration for the account appearance, and expanded visibility of visitors. While a certain amount of benign users are inclined to be motivated to upgrade their accounts, accounts controlled by attackers are extremely unlikely to participate in such paid upgrade since the upgraded membership contributes nothing to their collection of virtual currency. Figure 4.6 presents the distribution for this feature: while approximately 37% of benign users purchased at least one type of upgraded membership, the vast majority of malicious accounts do not make any purchase.
4.3.2.2 Currency Collection Features

In addition to collecting virtual currency by participating in online promotion activities, an OSN user can recharge her account with virtual currency through various ways such as wire transfer, selling virtual goods, and transferring from other accounts. Generally, benign users should be more active with respect to recharging their accounts. We propose two features to characterize this trend from two aspects including the amount of recharging and the important sources for recharging.

• Feature 4 - The Average Recharge Amount of Virtual Currency. This feature represents the average amount of virtual currency for each recharge regardless of the sources for recharging.

Benign users who participate in online promotion activities are usually also interested in other online financial activities. Therefore, these benign users tend to actively recharge their accounts. The recharge amount for each time by a benign user is commonly considerably large since users tend to decrease the hassle of recharging. In contrast, if a malicious account has been recharged, the amount of virtual currency for each recharge is usually bounded by a relatively small volume offered by the online promotion activity. Figure 4.7 presents the distribution of this feature for benign and malicious accounts, respectively.

Figure 4.7: Feature 4 - The Average Recharge Amount; Reprinted from [92]
Specifically, the average recharge amount is higher than 1100 Chinese cents\(^1\) for more than 50% of benign users, where only a small percentage (i.e., approximately 15%) of malicious users has an average amount that is higher than 140 Chinese cents.

We then consider the sources for recharge. Despite a variety of possibilities, we focus on one source that is rewards from promotion activities, and accordingly design one self-explanatory feature as follows:

- **Feature 5 : The Percentage of Recharge from Promotion Activities.**

  The feature intuitively profiles how significantly online promotion activities contribute to the wealth of an account. Benign users are inclined to employ a variety of sources for recharge. Comparatively, malicious accounts usually exclusively rely on online promotion activities to collect virtual currency. Figure 4.8 presents the distribution of this feature. Both the majority (approximate 88%) of benign and malicious accounts have not collected any virtual currency from promotion activities. In spite of the similarity, malicious accounts differentiate themselves from benign ones by experiencing a strong bipolar distribution: approximate 88% of malicious accounts do not collect virtual currency from online promotion activities at all where as the remaining (i.e., approximate 12%) accumulate their

\(^1\)Each Chinese cent is approximately \(\frac{1}{6}\) of a U.S. cent
wealth exclusively from online promotion activities; the number of malicious accounts, whose currency is partially collected from online promotion activities, is negligible. This implies that it is the first time for about 88% of malicious accounts to participate in online promotion events where 12% are reused accounts.

4.3.2.3 Features of Usage Activities

As an increasing number of business capabilities are integrated into social networks, users conduct a variety of activities such as shopping and gifting. Features in this category characterize how users spend their wealth. As a means towards this end, we propose three features.

- **Feature 6: The Total Amount of Expenditure.**

![Figure 4.9: Feature 6 - The Total Amount of Expenditure; Reprinted from [92]](image)

This feature characterizes the total amount of expenditure of an account regardless of the possible sources such as the associated bank accounts, the virtual currency, and other online social network platforms. As the popular online social networks are integrated into almost all mainstream e-business infrastructures, shopping and gifting through these accounts becomes prevalent. Users keep recharging their accounts, persistently associate their bank accounts with OSN accounts, and actively engage in shopping and gifting. Therefore,
we expect that benign accounts accumulate a high amount of expenditure. Comparatively, the total amount of currency controlled by each malicious account is constrained by the total number of virtual currency collected from online promotions, which is expected to be relatively small. Figure 4.9 presents the distribution of this feature. At least 50% of benign accounts have spent more than 4000 Chinese cents. Comparatively, only a tiny percentage of 0.9% malicious accounts spent more than 4000 Chinese cent; the vast majority of malicious accounts never committed any spending.

- Feature 7: The Percentage of Expenditure from Banks.

![CDF of Percentage of Expenditure from Banks](image)

Figure 4.10: Feature 7 - The Percentage of Expenditure from Banks; Reprinted from [92]

As we have introduced, a user can associate her bank account with the OSN account. This bank account can be directly used for shopping and gifting in addition to recharging the OSN account with virtual currency. Such association may greatly facilitate financial activities but result in exposure of users’ bank identities in case of law enforcement. Figure 4.10 presents the evaluation of this feature based on the real-world data. A very tiny percentage of malicious accounts expended their currency from bank accounts. Comparatively, this percentage is considerably high for benign users (i.e., around 45%).

- Feature 8: The Percentage of Expenditure as Gifts.
After malicious accounts collect virtual currency from the online promotion activities, they will transfer it to malicious accounts used for trading. Sending online gift cards becomes the best option for malicious accounts to transfer currency for two reasons. First, sending online gift cards inside an OSN usually does not incur any cost. Second, such transfer is independent to any bank, thereby requiring no personal information and consequently minimizing the exposure of attackers. We therefore design this feature to quantify the percentage of all expenditure that is used for gifts. Figure 4.11 presents the empirical analysis of this feature. Specifically, malicious accounts show a strong pattern of bipolar distribution. Particularly, approximate 81% of malicious accounts never sent any gifts. Most likely, these accounts have never successfully participated any promotion activities and therefore have nothing to transfer. The rest of them (i.e., about 19%) spend all of their expenditure on gifts, which implies that they transfer all of their wealth to other accounts. Similar to malicious accounts, most of benign accounts (i.e., about 80%) never engaged in any gifting activities. However, the remaining benign accounts (i.e., about 20%) seem reluctant to use all of their wealth as gift.

![Figure 4.11: Feature 8 - The Percentage of Expenditure as Gifts; Reprinted from [92]](image-url)
4.4 Evaluation

4.4.1 Experimental Setup

We conducted an extensive evaluation of ProGuard that focused on overall detection accuracy, feature importance, and the correlation between these features. In total, we used 56,000 samples that featured 28,000 malicious accounts and 28,000 benign accounts; this well-balanced dataset served well for training a statistical classifier [2].

4.4.2 Experimental Results

4.4.2.1 Detection Accuracy

We evaluated ProGuard’s detection capabilities using the normalized random forest (RF) method, which uses multiple unpruned classification trees that use normalized bootstrapped trees from the original samples for the training set; in this method, the classification is chosen by the majority voting over all the trees in the random forest. We used the RF method [9] as a binary classification method for evaluating the accuracy of malicious account detection. We also performed 10-fold cross-validation, a model validation technique that assesses the generalization of the trained RF classifier to the original dataset. Specifically, we divided the entire dataset into 10 equal-sized subsets (i.e., 10 non-overlapping partitions) and iteratively assigned nine non-overlapping partitions to training and the remaining single non-overlapping partition to testing. To ensure the best classification model, the RF classifier was tuned by 3,000 trees to grow, and four features were randomly sampled for each tree splitting [45]. We selected the ROC curve, which is a plot between a true-positive rate and false-positive rate, to demonstrate ProGuard’s overall detection performance. Fig. 4.12 shows that the proposed eight features used by ProGuard achieved high detection accuracy in malicious detection in cases where they were integrated by the
RF classifier. For instance, *ProGuard* achieved a high detection rate of 96.67% with a low false-positive rate of 0.3%.

![ROC Curve](image)

**Figure 4.12**: ROC curve on the proposed 8 features; Reprinted from [92]

In addition, regarding the proposed detection features, it is practically important to study the performance of alternative classification methods. Therefore, we also evaluated *ProGuard*’s sensitivity to traditional statistical classifiers, such as the support vector machine (SVM) [17] and gradient-boosted tree [31] classifiers. Specifically, we used 10-fold cross-validation on each classifier and compared the area under the ROC curve (AUC) [26], as this area is equal to the probability that a randomly chosen sample of malicious accounts will have a higher estimated probability of belonging to malicious accounts than a randomly chosen sample of benign accounts; this is a widely-used method for measuring the quality of supervised classification models. Since AUC is cutoff-independent and AUC values range from 0.5 (i.e., no predictive ability) to 1.0 (i.e., perfect predictive ability), the classifier with the highest AUC indicates better prediction performance, regardless of the cutoff selection. Table 4.1 represents the summary information on AUC values for the classifiers selected in the experiments. Both the SVM and gradient-boosted tree classifiers achieved fairly good detection results that were comparable with the RF classifier, though...
the RF classifier had the best performance overall regarding AUC. These experimental results revealed that our detection features are robust regardless of the selected statistical classifiers.

Table 4.1: AUCs for Three Classifiers; Reprinted from [92]

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>0.9959</td>
</tr>
<tr>
<td>SVM</td>
<td>0.9753</td>
</tr>
<tr>
<td>Gradient-Boosted Tree</td>
<td>0.9781</td>
</tr>
</tbody>
</table>

4.4.2.2 Feature Importance and Correlation

Overall, these top three features which include Feature 1 (*The Ratio of Active Days*), Feature 4 (*The Average Recharge Amount of Virtual Currency*), and Feature 7 (*The Percentage of Expenditure from Banks*) tend to complement each other in terms of our feature design which covers General Behaviors, Currency Collection, and Currency Usage.

We also performed feature importance analysis to evaluate the detection capabilities of each of ProGuard’s proposed features. Since RF outperformed the other classification methods in Table 4.1 and can reasonably model the relationship between features and OSN accounts, we applied each feature’s variable importance to the RF classification model using permutation test [45]. The values of variable importance were computed using the mean decrease in accuracy, which was defined as a prediction error rate after permuting each feature [45]. The rank of features based on their variable importance is shown in Table 4.2. Specifically, the ratio of active days (Feature 1) and the average recharge amount of virtual currency (Feature 4) represent the most significant features for detection. Overall, the top three features: Feature 1 (*The Ratio of Active Days*), Feature 4 (*The Average Recharge Amount of Virtual Currency*), and Feature 7 (*The Percentage of Expenditure from Banks*) tended to complement each other in terms of our feature designs, which cover general behaviors, currency collection, and currency usage.
Table 4.2: Feature importance rank of ProGuard by Random Forest; Reprinted from [92]

<table>
<thead>
<tr>
<th>Rank</th>
<th>Variable importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1</td>
<td>465.4</td>
</tr>
<tr>
<td>Feature 4</td>
<td>349.9</td>
</tr>
<tr>
<td>Feature 7</td>
<td>246.6</td>
</tr>
<tr>
<td>Feature 2</td>
<td>61.31</td>
</tr>
<tr>
<td>Feature 5</td>
<td>56.91</td>
</tr>
<tr>
<td>Feature 8</td>
<td>52.17</td>
</tr>
<tr>
<td>Feature 6</td>
<td>46.44</td>
</tr>
<tr>
<td>Feature 3</td>
<td>35.63</td>
</tr>
</tbody>
</table>

We also evaluated the correlation among various features, where correlation implies the extent to which a feature might be redundant given other features. We adopted two strategies in our experiments. First, we used the correlation matrix to identify whether correlated predictors exist in the features, where each element in the upper triangular matrix represented the Pearson’s $r$ correlation coefficient [44] of two features. The Pearson’s correlation coefficient $r \in [-1, 1]$ of two features, $X$ and $Y$, can be defined as

$$r = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2} \sqrt{\sum (Y - \bar{Y})^2}}$$

(4.1)

where $\bar{X}$ and $\bar{Y}$ denote the means of the two features. Fig. 4.13 shows that the majority of features have a weak linear correlation to other features. For example, a pair of features, Feature 1 (The Ratio of Active Days) and Feature 8 (The Percentage of Expenditure as Gifts), represented the highest negative correlation score at 0.07. The highest positive correlation between Feature 4 (The Average Recharge Amount of Virtual Currency) and Feature 6 (The Total Amount of Expenditure) was 0.82.

Second, we performed principal component analysis (PCA), which is used to analyze the correlation between variables with respect to the data variance [37]. We presented our evaluation results using the PCA variable factor map [64] in Figure 4.14. In a factor map, an arrow represents a feature. The angle between the two feature arrows explains the correlation between the respective features of the two principal components. Specifically, if
Figure 4.13: Upper triangular matrix; Reprinted from [92]

the angle between the two arrows is close to 90 degrees, the corresponding features are not correlated. As illustrated in Figure 4.14, the angles between the majority of features were approximately 90 degrees (e.g., Feature 3 (The Number of Services Purchased By An Account) and Feature 5 (The Percentage of Recharge from Promotion Activities) onto the third and fourth principal components), implying a weak correlation between features. According to the correlation matrix and PCA variable factor map, which show little correlation, we concluded that most of the features complement each other based on their linear independence.

4.5 Discussion

Attackers who create multiple malicious OSN accounts intend to evade our detection efforts once they comprehend the ProGuard’s framework. This phenomenon is not a systematic flaw specific to the proposed detection technique but a typical challenge for all
Figure 4.14: The variables factor map (PCA); Reprinted from [92]

detection systems. Specifically, attackers can instrument their malicious OSN accounts, and consequently, in effect, their behavioral patterns cannot be differentiated from benign OSN accounts. However, since ProGuard’s detection features target the characteristics of malicious accounts that play a key role in the success of their attacks and their stealthiness against other detection efforts, the attackers’ successful evasion could essentially limit their malicious capabilities. For example, attackers can intentionally increase the number of active days in malicious accounts. However, this reveals their malicious accounts to existing bot-account detection systems that utilize frequent login patterns to identify malicious accounts [91]. In addition, attackers can significantly increase their number of friends by adding their other malicious accounts as friends. However, this can be detected by many systems that leverage social structures, such as [72, 88, 56]. Attackers can also use diverse methods to recharge their resources, recharge amounts, and bank account expenditure amounts. Nevertheless, their attempts directly increase the financial cost associated with laundering the attacks, which may render the attacks themselves useless. Moreover,
attackers could try to reduce the percentage of expenditures through online gifting, but this fundamentally constrains their bandwidth to launder the virtual currency collected from online promotion events.

It is possible that attackers can hack some benign users’ accounts and leverage these benign accounts to participate in online promotion events. However, it is worth noting that hacking a large number of benign accounts is a nontrivial task that requires significant costs. Moreover, the majority of social networks have generally provided efficient recovery options to protect hacked accounts. Nonetheless, attackers can create a significant number of free OSN accounts dedicated to persistent malicious activities. Overall, this reduces attackers’ motivation to employ hacked benign accounts for this type of attack. However, if attackers indeed attempt to use the hacked accounts for such attacks, these accounts may demonstrate both benign and malicious behavior. In this scenario, if the malicious behaviors are dominant in the OSN dataset (i.e., the online financial activities from benign accounts are trivial), then we expect that our system will still perform well in detecting these accounts. On the contrary, if benign behaviors dominate (i.e., these accounts are considerably active in online financial activities), then the accounts are likely to yield a false negative. Addressing false negatives in this scenario is an important problem, and our future work will look for efficient solutions.

Considering the active trend of integrating financial capabilities into OSNs, the ability to detect malicious accounts that engage in suspicious financial activity is centrally important. Although the ProGuard’s design and evaluation are based on real-world data collected from Tencent QQ, a leading Chinese OSN with 899 million active accounts, its features and detection framework can be easily applied to other OSNs that integrate financial activities. Specifically, all the proposed features are based on essential financial functions, such as recharging and gifting. In addition, all current features rely on coarse-grained information that minimizes privacy concerns, which may foster the deployment of the proposed system in a detection-as-service model.
Since integrating OSNs with financial capabilities is an active trend in online business, the efforts to detect malicious accounts, which interact with suspicious financial activities, is becoming more important. Even if the design and evaluation of ProGuard are originated from real-world dataset collected from Tencent QQ, a leading Chinese OSN with 899 million active accounts, the proposed features and the detection framework can be easily fitted to other mainstream OSNs that integrate financial activities. Particularly, our proposed features are designed on the basis of fundamental financial functions such as recharging and gifting. Furthermore, all the features depend on coarse-grained information that minimizes the OSN users’ privacy issue, which could develop the real deployment of our ProGuard system in a detection-as-service model. In spite of the fact that ProGuard system can efficiently detect malicious accounts employed for collecting virtual currency via online promotion events, the current implementation of our system has the limitation to detect malicious accounts used in transferring and laundering steps. Therefore, extending ProGuard to handle such malicious detection capabilities is considered as our future work.

4.6 Summary

This chapter presents ProGuard, a novel system that can automatically detect malicious accounts that participate in online promotion events on social networks. ProGuard’s features use three aspects to characterize malicious OSN accounts: general behavior, currency collection, and currency usage. Applied to a real OSN dataset collected from Tencent QQ, a global leading Chinese OSN company, ProGuard achieved a high detection rate of 96.67% and an extremely low false-positive rate of 0.3%, indicating that the system has value for real deployment.
Chapter 5: Conclusion and Future Work

5.1 Conclusion

The Internet is a popular platform for communication and information sharing. However, since the Internet inevitably poses security risks for end-users and network services, data-driven network-centric threat assessment is of the utmost importance. Many such threat assessment systems have been proposed. Yet, these threat assessment systems face three challenges that can reduce the effectiveness and efficiency of their practical deployment. First, attackers display a diverse range of behaviors in launching threats in order to increase their effectiveness, as existing threat assessment systems rely on static patterns to detect threats. Second, regarding network-centric threats, the heterogeneity of the data used is fundamentally significant, as the assessment systems rely on the systematic characterization of this data to assess threats. Third, the volume of data in network-centric systems is rapidly growing, and this growth requires threat assessment systems to operate on a distributed architecture in the cloud computing environment in order to circumvent their scalability limitations. This research aims to address these three challenges by designing three new threat assessment systems.

In the first phase of our dissertation work, we designed DNSMiner [39], a novel system that identifies the presence of network users using DNS behavioral tracking, which is one of the most important methods for addressing privacy concerns within the Domain Name System (DNS). DNSMiner improves the effectiveness and scalability of existing user iden-
tification systems by automatically deriving DNS-based fingerprints, which can reveal the users’ unique DNS activities. In building this system, we made the following contributions:

- We proposed five new patterns to accurately characterize network users’ DNS behaviors using their DNS queries.
- We built a scalable MapReduce framework to automatically generate DNS fingerprints from a large DNS dataset.
- To demonstrate the effectiveness of the DNSMiner system for practical deployment, we present an extensive evaluation strategy using a real DNS dataset from an on-campus network.

The second phase of our dissertation work focuses on developing DART [38], a new system that detects fake antivirus webpages. This system boosts the effectiveness of existing detection methods by applying multifaceted features that are extracted from the semantic analysis of diverse trendy security keywords and the characteristics of their malicious networking infrastructure. Through this work, we made the following contributions:

- We built a novel machine learning-based detection system that can effectively detect fake antivirus webpages before the end-user’s system is fully infected.
- We designed diverse detection patterns that systematically capture the essential invariance of fake antivirus webpages, specifically regarding three aspects, including human-perception behavior, search engine optimization, and networking infrastructure.
- We evaluated the DART system’s effectiveness using actual fake antivirus webpages on the Internet, which revealed the system’s high detection accuracy and extremely low false-positive rate.
In the third phase of our dissertation work, we developed *ProGuard* [92], a novel detection system that identifies malicious accounts that participate in social network-based promotion events. *ProGuard* can proactively detect malicious OSN accounts before the reward is completed. In addition, our detection system can improve the effectiveness of existing detection systems that target only spam or financial fraud. By designing the system, we offer the following contributions:

- We designed a detection system that is capable of identifying malicious OSN accounts used for online promotion events during the virtual currency collection phase.
- We proposed eight novel features that characterize how malicious OSN accounts behave using three factors: general profile, virtual currency collection, and virtual currency usage.
- We developed an automatic detection system based on supervised learning algorithms and evaluated the effectiveness of our system using a real OSN dataset.

### 5.2 Future Work

In addition to addressing centrally important research challenges for building effective and scalable network-centric threat assessment systems, we present the following avenues for future research.

- **DNS Session-based Fingerprinting for Short-Term Network User Identification**

  As discussed in Chapter 2, we developed a DNSMiner system that can automatically derive users’ behavioral fingerprints from large-scale DNS queries (i.e., that can passively collect DNS traces). This system uses these behavioral fingerprints to identify the presence of unknown users in new DNS streams. The experimental results demonstrate that the system has high detection accuracy in deanonymizing network
users’ presence. However, currently, the implementation of DNSMiner focuses only on network users that are persistently active for at least four days out of the past seven days. As such, the design is limited in its ability to detect networks users engaging in short-term DNS activities. Tracking these users’ behaviors is challenging, as extracting the periodic DNS fingerprints of short-term active users is difficult. Hence, to improve the practical usage of the system for detecting users engaging in transient DNS activities, we will study the characteristics of these users’ behavior-based tracking between multiple gap-sessions based on the observed DNS queries during a short period (e.g., one day or three days) and leverage machine learning and natural language processing (NLP) techniques to achieve better accuracy in identifying these users in the future.

**Fake Antivirus Webpages Detection System in a Distributed Architecture**

Due to the nature of social engineering attacks in malware distribution, attackers are more likely to launch a large number of fake antivirus webpages, resulting in an important challenge regarding the DART system’s scalability (see Chapter 3). To enhance the systems scalability, future research could design a distributed infrastructure that can be implemented in cloud computing environment (e.g., Hadoop or Spark).

**Detecting Malicious OSN Accounts during Multi-Layer Transferring or Laundering Phases**

Currently, the ProGuard system can efficiently detect malicious OSN accounts as they collect virtual currency. However, to evade detection systems, attackers actively extend the resilience of their malicious infrastructure using obfuscation methods. For example, it is possible that attackers hack accounts benignly to circumvent the ProGuard system’s detection capabilities, as benignly hacked accounts would appear as normal, unhacked accounts. We plan to address this important challenge in our future work by investigating malicious OSN accounts that are used for transferring
and laundering.

5.3 Closing Remarks

In our dissertation work, we have addressed significant research challenges in network-centric threat assessment. Specifically, we focused on new three challenges that existing network-centric threat assessment systems must overcome: behavioral diversity, data heterogeneity, and large-volume data. First, we proposed a novel threat assessment system that can identify network users using DNS behavioral-based tracking and, as a result, disclose the DNS information leakage of end-users against privacy threats. Second, we built a new system for detecting fake antivirus webpages, an active trend in malware distribution. Finally, we presented a systematic framework that can detect malicious OSN accounts used to launder virtual currency through online promotion events. Together, these contributions can improve the effectiveness and scalability of existing threat assessment methods. Since network-centric threats are continuously evolving, we are confident that the solutions proposed in this dissertation will contribute to future research on network-centric threat assessment.
Bibliography


[31] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining: concepts and techniques. Morgan kaufmann, 2006.


