Securing Enterprise Networks with Statistical Node Behavior Profiling

Su Chang
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd
Part of the Electrical and Computer Engineering Commons

Recommended Citation
https://lib.dr.iastate.edu/etd/11483

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Securing enterprise networks with statistical node behavior profiling

by

Su Chang

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Thomas E. Daniels, Major Professor
Douglas W. Jacobson
Julie Dickerson
Sung Song
Zhao Zhang

Iowa State University
Ames, Iowa
2010

Copyright © Su Chang, 2010. All rights reserved.
I would like to dedicate this thesis to my wife Ye Cheng and to my parents Youzhen Chang and Xi Zhang, without whose support I would not have been able to complete this work.
TABLE OF CONTENTS

LIST OF TABLES .......................................................... vii
LIST OF FIGURES ......................................................... viii
LIST OF ABBREVIATIONS ................................................ xi
ACKNOWLEDGEMENTS .................................................... xiv
ABSTRACT ................................................................. xvi

CHAPTER 1. Introduction ................................................. 1
  1.1 Problem Statement .................................................. 1
  1.2 Thesis Statement .................................................... 2
  1.3 Introduction .......................................................... 2
  1.4 Motivations & Research Focus ................................. 3
  1.5 Research Objectives and Overview ............................. 6
  1.6 Contributions ......................................................... 8
  1.7 Organization of the Thesis ....................................... 10

CHAPTER 2. Correlation based Node Behavior Profiling for Enterprise
  Network Security ....................................................... 12
  2.1 Introduction .......................................................... 12
  2.2 Related Work ........................................................ 15
  2.3 Correlation based Node Behavior Profiling ................. 18
     2.3.1 Common Node Behavior Cycle ......................... 18
     2.3.2 Attribute Selection and Problem Formulation ........ 19
     2.3.3 Behavior Clustering ........................................ 21
### 2.3.4 Metrics to Measure the Abnormal Change of Node Behaviors

### 2.3.5 Discussions

### 2.4 Evaluations

- 2.4.1 Trace Information
- 2.4.2 A Detailed Example
- 2.4.3 False Positive Rates

### 2.5 Conclusion and Discussions

### CHAPTER 3. A Framework for P2P Botnets

- 3.1 Introduction
- 3.2 Related Work
  - 3.2.1 Botnet Overview
  - 3.2.2 Centralized Botnets
  - 3.2.3 Advanced Botnets
- 3.3 Botnet Framework & Lcbot
  - 3.3.1 A Framework for Botnets/Worms
  - 3.3.2 Metrics for A Good Botnet under the Proposed Framework
  - 3.3.3 Predicted Botnet (lcbot)
  - 3.3.4 Propagation Method
- 3.4 Evaluations
  - 3.4.1 Evaluation of the Propagation Scheme
  - 3.4.2 Performance Comparison of P2P Botnets
- 3.5 Conclusion

### CHAPTER 4. Node Behavior based C&C Channel Detection for P2P Botnets

- 4.1 Introduction
- 4.1.1 Overview of the Proposed Approach
- 4.2 Background & Related Work
  - 4.2.1 Introduction to SPRT
  - 4.2.2 Related Work
CHAPTER 5. A Novel Background Traffic Modeling Structure for NIDS

Evaluations ................................................................. 98

5.1 Introduction .......................................................... 98

5.2 Related Work ........................................................ 102
   5.2.1 Real Trace vs Realistic Modeling ......................... 102
   5.2.2 Packet Level Modeling .................................. 103
   5.2.3 Advanced Modeling .................................. 104

5.3 Node Behavior based Traffic Modeling ....................... 107
   5.3.1 Summarization & Comparison of Current Work .......... 107
   5.3.2 Framework Overview .................................. 109
   5.3.3 Behavior Profiling Extraction Engine .................. 112

5.4 Evaluations .......................................................... 114
   5.4.1 Behavior Profiles vs Modeling Precision ............... 115
   5.4.2 Traffic Homogeneity .................................. 116
   5.4.3 Realistic Traffic Generation .......................... 120
5.4.4 Discussions and Future Work .................................. 121
5.5 Conclusion .............................................................. 122

CHAPTER 6. SUMMARY AND DISCUSSION .......................... 125
6.1 Conclusion .............................................................. 125
6.2 Future Work ........................................................... 127
   6.2.1 Evaluations of Anonymization Schemes .................. 127
   6.2.2 Improvement of the Existing Approach .................. 127
   6.2.3 Coupling with Multi-Variable Analysis .................. 128
   6.2.4 Mitigation & Traceback ........................................ 128
   6.2.5 High-speed & Large-scale Detection .................... 128
   6.2.6 Testbed Construction .......................................... 129

APPENDIX A. Detailed Information of Each Behavior Profile ........ 130
APPENDIX B. Main Functions in Agobot ............................. 131
APPENDIX C. Main Functions in SDbot05 ............................. 133
APPENDIX D. Formulation of SPRT-FOD using Cost ................. 134
   D.1 System Model Considered in SPRT-FOD .................... 134
   D.2 Problem Formulation ............................................ 134
APPENDIX E. Another Solution to SPRT Optimization ............... 137
BIBLIOGRAPHY ............................................................ 139
LIST OF TABLES

Table 2.1  Parameters considered for attributes for host $i$ at interval $t$  

Table 2.2  Compression and representation ratio  

Table 2.3  Node behaviors at different time intervals  

Table 2.4  Summary of node activities in terms clusters within one hour  

Table 2.5  Acceptance regions for cluster 1, 2, 3, 4, 5, 6, $\alpha = 0.05$  

Table 2.6  Proportion test results (1: reject; 0: not reject)  

Table 3.1  An example of well-known botnet history (slightly updated from [97])  

Table 3.2  Evolution of bot techniques  

Table 3.3  Framework for Botnet/Worm  

Table 4.1  Data information  

Table 4.2  Proportion test results (1: reject; 0: not reject)  

Table 4.3  t-test results (1: reject; 0: not reject)  

Table 4.4  False positive rates (fpr) for each trace  

Table 4.5  False positive rates  

Table 4.6  Detection rate of ITD, CTD & SPRT-FOD under simple case  

Table 5.1  Traffic information (log)  

Table 5.2  Kolmogorov-Smirnov Test  

Table A.1  Detailed behavior information of trace 1526(0107) (logscale)
| Figure 1.1 | General picture of this research | 8 |
| Figure 2.1 | Examples of node behavior overlap | 14 |
| Figure 2.2 | Corresponding dendrogram | 26 |
| Figure 2.3 | Intra-cluster distance over $T_h$ | 26 |
| Figure 2.4 | Cluster size over $T_h$ | 27 |
| Figure 2.5 | Cluster sizes at $T_h = 0.21$ | 27 |
| Figure 2.6 | Intra-cluster distance & std | 32 |
| Figure 2.7 | Inter-cluster distance & std | 32 |
| Figure 2.8 | Cluster sizes with worm present | 33 |
| Figure 2.9 | Intra-cluster distance | 33 |
| Figure 3.1 | Typical (IRC) infection cycle (modified from [127]) | 44 |
| Figure 3.2 | Possible patterns of lcbot | 52 |
| Figure 3.3 | Algorithm Description | 53 |
| Figure 3.4 | Distribution of # of parents | 55 |
| Figure 3.5 | Variation of # of parents | 55 |
| Figure 3.6 | MC after random removal | 56 |
| Figure 3.7 | MC after random removal | 56 |
| Figure 3.8 | MC after select removal | 57 |
| Figure 3.9 | Overhead ratio | 57 |
| Figure 3.10 | Traceback probability | 58 |
| Figure 3.11 | CDF of C&C speed in hops | 58 |
Figure 5.18 # of sessions subnet021 ............................................. 119
Figure 5.19 # of packets subnet008 ............................................. 119
Figure 5.20 NetBottle project ...................................................... 123
Figure 5.21 Request size subnet021 ............................................. 124
Figure 5.22 Response size subnet021 ........................................... 124
Figure 5.23 Starting time subnet021 ............................................. 124
Figure 5.24 Inter arrival subnet021 ............................................. 124
Figure 5.25 # of packets subnet021 ............................................. 124
Figure 5.26 # of sessions subnet021 ............................................. 124

Figure B.1 Main() function in Agobot3-0.2.1-pre4-priv .................. 131
Figure B.2 Agomainctrl() function in Agobot3-0.2.1-pre4-priv .......... 132
Figure C.1 Main() function in sdbot5b ...................................... 133
Figure D.1 Formulation of SPRT-FOD ........................................ 136
Figure E.1 Optimization on $p_1$ ($p_0 = 0.6$) ............................... 138
Figure E.2 Optimization on $p_0$ ($p_0 = 0.6$) ............................... 138
Figure E.3 Optimization on $p_1$ ($p_0 = 0.5$) ............................... 138
Figure E.4 Optimization on $p_0$ ($p_0 = 0.5$) ............................... 138
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADU</td>
<td>Application Data Unit</td>
</tr>
<tr>
<td>BINO</td>
<td>Binomial Distribution</td>
</tr>
<tr>
<td>BMDT</td>
<td>Behavior Mean Distance based Test</td>
</tr>
<tr>
<td>BPT</td>
<td>Behavior Proportion based Test</td>
</tr>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>CC</td>
<td>Command and Control</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>CIA</td>
<td>Confidentiality, Integrity and Availability</td>
</tr>
<tr>
<td>CTD</td>
<td>Correlated Test Detection</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial-of-Service</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DNSBL</td>
<td>DNS-based Blackhole List</td>
</tr>
<tr>
<td>ECDF</td>
<td>Empirical CDF</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In First-Out</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Message Access Protocol</td>
</tr>
<tr>
<td>IMAPS</td>
<td>Secure Internet Message Access Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>IPP</td>
<td>Internet Printing Protocol</td>
</tr>
<tr>
<td>IPT</td>
<td>Inter Packet Time</td>
</tr>
<tr>
<td>IRC</td>
<td>Internet Relay Chat</td>
</tr>
<tr>
<td>ISS</td>
<td>Internet Security Systems</td>
</tr>
<tr>
<td>KS-Test</td>
<td>Kolmogorov-Smirnov Test</td>
</tr>
<tr>
<td>LBNL</td>
<td>Lawrence Berkeley National Laboratory</td>
</tr>
<tr>
<td>LCBOT</td>
<td>lcbot Loosely Coupled P2P Botnet</td>
</tr>
<tr>
<td>LSH</td>
<td>Locality Sensitive Hashing</td>
</tr>
<tr>
<td>MC</td>
<td>Mean Coverage</td>
</tr>
<tr>
<td>MD5</td>
<td>Message-Digest Algorithm 5</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NETBIOS-SSN</td>
<td>NETBIOS Session Service</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Controller</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network Intrusion Detection System</td>
</tr>
<tr>
<td>OR</td>
<td>Overhead Ratio</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>POP3S</td>
<td>POP3 over TLS/SSL</td>
</tr>
<tr>
<td>RRE</td>
<td>Request-Response-Exchange</td>
</tr>
<tr>
<td>RED</td>
<td>Random Early Detection</td>
</tr>
<tr>
<td>RTT</td>
<td>Round-Trip Time</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SPRT</td>
<td>Sequential Probability Ratio Test</td>
</tr>
<tr>
<td>SPRT-FOD</td>
<td>SPRT based Fast and Optimized Detection</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>Acronym</td>
<td>Full Form</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TG</td>
<td>Traffic Generator</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TP</td>
<td>Traceback Probability</td>
</tr>
<tr>
<td>TTCP</td>
<td>Test TCP</td>
</tr>
<tr>
<td>TTL</td>
<td>Time-to-Live</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VOD</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>VOIP</td>
<td>Video on Demand</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation.

First and foremost, I would like to express my deepest appreciation to my major advisor, Dr. Thomas E. Daniels. Without his knowledge, perceptiveness, patience, guidance, encouragement and support, I would have never finished my thesis. I also thank him for spending countless hours with me from discussing research problems to developing promising solutions, from reviewing my papers to guiding my studies. His direction during my study at ISU has been an invaluable part in my whole life. I not only got invaluable advice on how to conduct good research, but also found constant encouragement on how to pursue my life goal. I would also remember the moments we bore the failures together and finally shared the successes for ever.

I would like to thank my committee members: Dr. Julie A. Dickerson, Dr. Douglas W. Jacobson, Dr. Sung Song and Dr. Zhao Zhang, for their valuable time and effort in reviewing this dissertation and providing insightful comments and suggestions. I would also like to thank Dr. Yong Guan for his early support and guidance in my study. I would additionally like to thank all the lecturers in Department of Electrical and Computer Engineering and other departments, from whom I learnt a lot. Also, many thanks to Mrs. Pamela J Myers, Mrs. Ginny Anderson and other department staffs for their help in last four years.

I would like to thank all my labmates and friends at Iowa State University. I am only able to list a few here: Ben Anderson, Majd Al-Kofahi, Jason Erbskorn, Shantanu Gattani, Amy Joines, Paras Malhotra, Alex Pease, Yongping Tang, Wei Wang, Yawen Wei, Yang Yang, Zhen Yu and Linfeng Zhang. I appreciate that all of you provided me a wonderful life in Ames, Iowa.
between January 2006 and December 2009. I would remember your faces, your words, and your deeds, deeply and gratefully.

This work is supported by the National Science Foundation under Grant No. 0627409, the Center for Information Protection, and by the Iowa State University Information Assurance Center.
The substantial proliferation of the Internet has made it the most critical infrastructure in today’s world. However, it is still vulnerable to various kinds of attacks/malwares and poses a number of great security challenges. Furthermore, we have also witnessed in the past decade that there is always a fast self-evolution of attacks/malwares (e.g. from worms to botnets) against every success in network security. Network security thereby remains a hot topic in both research and industry and requires both continuous and great attention.

In this research, we consider two fundamental areas in network security, malware detection and background traffic modeling, from a new viewpoint of node behavior profiling under enterprise network environments. Our main objectives are to extend and enhance the current research in these two areas. In particular, central to our research is the node behavior profiling approach that groups the behaviors of different nodes by jointly considering time and spatial correlations. We also present an extensive study on botnets, which are believed to be the largest threat to the Internet. To better understand the botnet, we propose a botnet framework and predict a new P2P botnet that is much stronger and stealthier than the current ones. We then propose anomaly malware detection approaches based directly on the insights (statistical characteristics) from the node behavior study and apply them on P2P botnet detection. Further, by considering the worst case attack model where the botmaster knows all the parameter values used in detection, we propose a fast and optimized anomaly detection approach by formulating the detection problem as an optimization problem. In addition, we propose a novel traffic modeling structure using behavior profiles for NIDS evaluations. It is efficient and takes into account the node heterogeneity in traffic modeling. It is also compatible with most current modeling schemes and helpful in generating better realistic background
traffic. Last but not least, we evaluate the proposed approaches using real user trace from enterprise networks and achieve encouraging results. Our contributions in this research include: 1) a new node behavior profiling approach to study the normal node behavior; 2) a framework for botnets; 3) a new P2P botnet and performance comparisons with other P2P botnets; 4) two anomaly detection approaches based on node behavior profiles; 4) a fast and optimized anomaly detection approach under the worst case attack model; 5) a new traffic modeling structure and 6) simulations and evaluations of the above approaches under real user data from enterprise networks.

To the best of our knowledge, we are the first to propose the botnet framework, consider the worst case attack model and propose corresponding fast and optimized solution in botnet related research. We are also the first to propose efficient solutions in traffic modeling without the assumption of node homogeneity.
CHAPTER 1. Introduction

1.1 Problem Statement

The wide deployment and continuous popularity of the Internet also lead to dramatic increase of attacks and fraudulent activities. Especially, botnet, a state-of-the-art malware, has become a primary platform for most malicious activities on the Internet [97]. With the distributed nature and controllable characteristic, it is now considered as the largest threat to the Internet. To prevent against this emerging threat, better detection techniques to identify botnets (bots and C&C channel) and better environments (realistic background traffic) to evaluate those detection techniques (to remove any potential weakness and problems) are both important fields of study.

Past studies on botnet detection have been focused on centralized ones using specific C&C protocol (e.g. IRC), without taking into account the potential advances in C&C protocol and structure. This results in the incapability of most proposed schemes to detect P2P botnets using multiple protocols in C&C. Similarly, past studies on traffic modeling model the network traffic by individual application or application mix, without considering the differences between different nodes. This leads to the problem in traffic generation that, for different network nodes, the choice of applications/destinations and the traffic characteristics for the same application are from the same underlying distributions respectively. As a consequence, all network nodes need to be considered homogeneous in terms of traffic generation, which is not realistic in general for traffic modeling.
1.2 Thesis Statement

Although there is no obvious correlation between detection and traffic modeling, in this thesis, we show that it is possible to specify a behavior-based model for both fields. The behavior-based model is useful for reasoning about the mechanisms of modeling traffic from heterogeneous network nodes. The behavior-based model is also useful for reasoning about the difference between normal and abnormal traffic at the behavior level and new detection algorithms used for P2P botnet detection. In addition, we also show that it is possible to incorporate botnets of all different kinds under an underlying framework to better understand the inherent correlation between them.

Furthermore, given that any good detection scheme should still perform well by taking into account the impact of attackers, we extend our proposed detection algorithms and propose a new one under the worst case attack model, where attackers know all the detection parameter values and can adjust the attack strategy to avoid being detected. Similarly, for any new traffic modeling schemes, it is better to make use of existing mature techniques to accelerate the development period. We show in this thesis that the newly proposed traffic model is not only capable of generating better realistic network traffic, but also compatible with most current major models in the literature.

1.3 Introduction

In the past decade, the substantial proliferation of the Internet in both wired and wireless fields has made it the universal communication platform and part of the most critical infrastructure in today’s world. According to [99], the Internet has distributed to over 233 countries and world regions and has more than 1.596 billion users by the end of March 2009. The average growth rate of Internet users is over 340% worldwide from 2000 to 2008. Especially, in some places such as Mideast, this rate is as high as 1296% [99]. In the meantime, there has been explosive growth of technologies and services over the Internet. For instance, during the past decade, we have witnessed the popularity of Internet services ranging from email to WWW (World Wide Web), from FTP (File Transfer Protocol) to P2P (Peer to Peer), from VOIP
(Voice over IP) to online video or VOD (Video on Demand), from online chatting to online gaming. Furthermore, the Internet has become an integral part of our lives and more and more people rely on it for their daily lives ranging from amusement to study and work. It is also a key platform to exchange crucial information such as financial information and personal private data.

All these discussed above lead to the common belief that the future global information transport network will be a hybrid network that has several different domains (e.g. wide-area Internet, enterprise and wireless networks) interconnected by IP (Internet Protocol). However, the network (e.g. Internet) was not originally designed to protect against malwares/attacks and has many potential vulnerabilities from the security perspective, e.g. buffer overflow. Therefore, to ensure the Internet its further popularity and prevent against any potential financial or personal losses, confidentiality, integrity and availability (CIA) have to be taken into account to provide reliable and risk free network services. For example, online transactions must be secure enough to prevent credit card information theft by attackers.

Moreover, as the Internet is increasing dramatically in its capacity, speed and coverage, there is always a great demand on new security techniques to improve the performances of the current ones and cope with the latest malwares. Therefore, in this research, we study and design efficient and effective techniques to provide security guarantee for the Internet. In particular, two fundamental areas in network security are considered: malware detection and realistic background traffic generation. We hope that the proposed schemes can serve as fundamental components for network security research, and the key observations made in this research can provide further insight to solve the hard problems in network security.

1.4 Motivations & Research Focus

The increasing reliance on the Internet also poses a number of great security challenges [97]. Network security thereby has become more and more important in providing security guarantee to ensure the proper functionality of the Internet. However, despite a great number of network security techniques such as firewalls and intrusion detection systems have been
proposed and deployed in the Internet, today’s Internet is far from secure and vulnerable to various malwares/attacks.

In contrast, malwares also evolve quickly and one significant current evolution is bot/botnet. It is known to be the platform of many current attacks. Moreover, as pointed out in [38], the sudden disappearance of worm after 2004 is probably not because the Internet is much more secure, but more likely because attackers no longer focus on infecting a large number of computers just to attract media attention, instead, their attention has shifted to compromising and controlling victim computers, an attack scheme which provides more potential for personal profits. Therefore, our research work in this thesis is motivated by the current Internet situation and summarized as follows:

• The fast increase of the Internet. With the goal of providing AAA (anywhere, anytime and anyone) services, the Internet is increasing dramatically in capacity, speed and coverage. Accordingly, it also poses great demands on security techniques to match the scale of the Internet.

• The increasing reliance on the Internet. As people use the Internet for various purposes ranging from amusement to study and work, there is also an increasing number of security problems in every aspect of the Internet. For example, security issues (e.g. clickfraud) arise when online advertisement and pay-per-click payment method are used.

• Malwares never stop evolving and evolve at a very fast speed. The evolution refers to both variations of the origin malware and new types of malware. For example, evolution from common worm to polymorphic or self-stopping worm, from worm to botnet. Therefore, there is always a requirement to design countermeasures against the latest and future malwares.

• Attackers are becoming more sophisticated and covert. Since the sudden disappearance of worms after 2004 is very likely due to the fact that attackers have shifted their attention to make more profits [38], various ways can be used by attackers to avoid being detected by
the known and traditional detection techniques. Consequently, better security techniques are needed to detect more sophisticated attacks.

- The monopoly of operating systems (e.g. Windows) used throughout the Internet hosts. This feature makes malwares spread more easily and widely once a single vulnerability is found. That is, once a vulnerability is found in a host in a network, it is very likely that a majority of hosts in the network have the same vulnerability due to system monopoly, which is favorable for malwares.

- The current user awareness of network security is not going to change much in the near future. As pointed out in [96], even many latest security related patches or updates are available online, many users are unaware of installing them for security purposes. Further, the normal user may have difficulty to differentiate the modern attacks, e.g. phishing.

Accordingly, to design good security countermeasures, many security aspects have to be considered jointly, including understanding of the latest malwares and their variations; designing detection schemes at different levels (e.g. host and network levels); speeding up the detection schemes for fast response; designing mitigation schemes to minimize the damages; designing traceback schemes to seize the real person behind the attack; evaluating the proposed schemes to fix any possible weakness and publish corresponding patches or updates to prevent any future damage. In this research, our main goal is to address the issue of network security and exclusively concentrate on two important areas: malware detection and background traffic modeling in an enterprise network environment, with the reasons given below:

- Malware detection is usually the basis for many other security countermeasures. For example, mitigation and traceback techniques can be applied for further analysis after the detection of malware. Moreover, if a better detection scheme is implemented, it can greatly reduce attackers’ attempts in doing nefarious work, as they will have a higher chance to be caught.
A thorough evaluation is needed before implementing any detection scheme or comparing the performances of different detection schemes. Consequently, modeling/generating realistic background traffic and attack traffic are both the basis for such evaluations. However, unlike the attack traffic generation which is relatively easy and straightforward, generation of realistic background traffic is much more difficult as it requires considering many different aspects of the normal traffic.

The enterprise network is not well studied as compared to the wide-area Internet. In [10], the authors claim that the nature of traffic inside enterprises remains almost wholly unexplored more than 15 years after studies of wide-area Internet traffic began to flourish. As the enterprise network is also an important part of the current Internet, we will focus ourselves mainly on this system.

1.5 Research Objectives and Overview

There are many potential directions/techniques to solve the problems discussed above. In this research, we consider solving them from the perspective of node behavior profiling. Node behavior profiling refers to profiling the traffic characteristics captured at different layers from different nodes. It is a good candidate to above issues. Because a good node profile scheme can help network administrators and researchers detect abnormal behaviors caused by either simple malicious attacks or misconfigurations. We also believe and will show in this research that, when the host-based behavior profiles are coupled with new network-based techniques, the node behaviors are also promising in detecting advanced malwares, such as botnets. In addition, nodes of similar behaviors can further be grouped together for traffic modeling. However, limited research has been done to this end and most of them focus on detecting misconfigurations or older attacks (e.g. scanning).

Therefore, our objectives in this research can be summarized as follows:

- Understand the normal node behavior from the network perspective using a new node behavior profiling approach. Provide key information that can be used for further analysis in detection and modeling.
• Understand the latest malware and its variations using a new malware framework and comparisons among themselves.

• Detect the network anomaly based on the understanding of both normal and abnormal node behaviors from the network perspective.

• Model realistic background traffic for NIDS evaluations based on the understanding of the traffic from normal behaviors.

To achieve the above goals, we conduct research in three topics as illustrated in Figure 1.1. To be specific, we first propose a correlated node behavior profiling approach which is able to characterize normal behaviors of different kinds. The main idea of behavior profiling is to extract the correlated behavior information shared by different nodes. Due to the way people use the Internet, this behavior correlation happens frequently among different nodes. We then study in detail not only the properties of each individual behavior, but also the possibility of using them in worm detection. The study on the behavior profiles also provides key observations that can be further utilized for detection and modeling.

Moreover, as botnets pose a significant danger to the Internet, we consider P2P botnets as the malware in this research. To better understand botnets and their future variations. We study not only the botnet analysis from the literature, but also the source codes of several well-known bots. We then propose a botnet framework capable of integrating all known botnets and design a new P2P botnet named lcbot (loosely coupled bot). The proposed botnet is much stronger and stealthier than the existing botnets from evaluation results.

Furthermore, based on the insights of behavior difference gained from the above work, we propose a series of anomaly detection approaches based on the proposed behavior profiling approach and statistical testing to detect the existence of P2P botnets in enterprise network environments. A fast and optimized anomaly detection approach is also proposed under the worst case attack model. In addition, by considering behaviors of the same pattern are the real source of the network traffic, we propose a new traffic modeling structure which is able to model the traffic from heterogeneous nodes and applications. Finally, we validate the assumptions
made in this research and evaluate the proposed approaches using real network user traces under different metrics and achieve encouraging results.

![Figure 1.1 General picture of this research](image)

1.6 Contributions

In summary, central to our approaches in this research is the node behavior profiling approach that groups the behaviors of different nodes by jointly considering time and spatial correlations. The detection and modeling approaches are based mainly on the behaviors identified by it. In particular, our contributions in this research include:

1. A new node behavior profiling approach which captures time and node correlations to study the behavior of normal nodes. We show that it is easy to understand the node behavior using the profiles and the normal behaviors are generally of different types. We also show in our evaluations that the behavior profiles are of different popularity and part of them can be considered statistically stable from the network perspective.
In addition, we give an example on worm detection to demonstrate that the profiles are capable of being used for further analysis and detection when coupled with different statistical testing methods;

2. A framework for botnets, which is able to capture botnet structures of all known kinds and predict new botnets;

3. Based on the proposed framework, we predict a new botnet that we call the Loosely Coupled P2P Botnet (lcbot). It is stealthy and can be considered as an extension of existing P2P botnet structures. We also design several new metrics and conduct experiments to compare the performances between lcbot and other P2P botnets in the literature to gain insight understanding of P2P botnets;

4. Two behavior based anomaly detection approaches on P2P botnets by coupling the statistical tests with the node behavior profiles identified. In brief, the key difference between our work and previous work in botnet detection is that, instead of trying to filter out the botnet behaviors in a network, we detect them by measuring their impacts on one or more normal behaviors in a statistical way;

5. A fast and optimized anomaly detection approach on P2P botnets. We make use of SPRT (Sequential Probability Ratio Test) to measure the impacts of the C&C behavior from P2P botnets ($H_1$) on normal behaviors ($H_0$) in a fast manner. We propose an approach that simplifies the behavior of each node in a stable way suitable for fast detection via SPRT. Then, under the worst case attack model, we formulate the C&C detection problem as an optimization problem and derive the optimal values for $H_1$ mathematically. An SPRT based fast and optimized detection approach is further evaluated under different traces;

6. A new traffic modeling structure for NIDS evaluations and several important observations that are helpful to improve the realistic level of current modeling schemes. By taking into account that the node behavior is the real source of the network traffic, we take a step further to propose a novel traffic modeling structure that captures the traffic
characteristics by jointly considering the node heterogeneity with other known features. We also apply empirical CDFs to model the common traffic features under this structure. In our initial evaluations, the empirical CDF and the modeling structure work well under the real user data from enterprise networks. We also obtain several important new observations that can be further used to improve the current traffic models.

7. Validation of the assumptions made in this research. Simulations and evaluations of the above approaches are further conducted under real user data from enterprise networks.

To the best of our knowledge, we are the first to propose the botnet framework, consider the worst case attack model and propose corresponding fast and optimized solution in botnet related research. We are also the first to propose efficient solutions in traffic modeling without the assumption of node homogeneity.

1.7 Organization of the Thesis

The rest of the dissertation is organized as follows: Chapter 2 presents the proposed research in node behavior profiling; we introduce the related work in both node behavior and traffic profiling. We then give a detailed example to use node behavior profiles to analyze the normal node behavior and discuss the possibility of worm detection using common node behaviors. In Chapter 3, we present our study on botnet, we review the related studies on botnet from both literature and our analysis on the bot codes we obtained. We then propose a botnet framework and predict a new botnet. The new botnet is further evaluated and compared with current bots in the literature.

A series of anomaly detection techniques are proposed in Chapter 4 which in general can be used to detect abnormal behaviors in the network, the techniques are based on the insights from the understanding of normal node behaviors. As an example, we apply our detection techniques on the Command and Control channel (C&C) detection of a known botnet. Chapter 5 discusses our techniques for realistic traffic modeling for NIDS evaluations. We do a solid survey on the recent major traffic models in the literature and find out that all of them make
the assumption on node behavior homogeneity. Although this assumption holds for many other cases at the aggregate level, it is too strong for NIDS evaluations, especially anomaly NIDS evaluations. Therefore, we take a step further to relax this assumption, build a traffic modeling structure which is able to yield better realistic traffic generation and compatible with most current schemes. We finally summarize our research and discuss our future work in Chapter 6.
CHAPTER 2. Correlation based Node Behavior Profiling for Enterprise Network Security

Profiling is a promising tool for many aspects in network security [4] [20] [7] [5] [16] [13] [8] [21] [7]. In this research, our main goal is to couple node behavior profiles with statistical tests on enterprise network security. Limited work has been done in the literature. In this chapter, we first propose a correlation based node behavior profiling approach to study normal node behaviors in enterprise network environments. We then propose candidate metrics to measure the change of node behaviors with worm present. In our evaluations, we evaluate the proposed approach and behavior metrics using real enterprise data (LBNL traces [10]). The results show that the correlation based node behavior profiling approach can well capture normal behaviors of different types. Consequently, the node behavior profiles are promising for anomaly detection when coupled with statistical methods.

2.1 Introduction

Profiling has a wide usage in many areas of computer networks [5], such as traffic/flow profiling, email profiling, web server profiling, topology profiling and resource profiling. When used to profile a behavior, it refers to extract information which is representative of behavior or usage patterns [5]. Little attention has been paid to node behavior profiling [5], especially for security purposes. In this research, we consider node behavior profiling as capturing the node (end host) behaviors in terms of node level traffic characteristics which can be further used for security purposes.

In general, normal behaviors/traffic can be considered different from abnormal ones. Therefore, when used properly, good node behavior profiles can help network administrators and
researchers to detect abnormal traffic in the network caused by either malicious attackers or misconfigurations. They also can help the design of background traffic models for security product evaluation (e.g. NIDS), as traffic from similar profiles can be considered of same type, which makes the modeling more accurate. Consequently, to build good node profiles, several important issues have to be taken into account jointly, including:

1. Diversity in applications and destinations from different nodes, which refers to the preference/habit of different users;
2. Dynamics of user preference/habit over time;
3. Diversity in traffic pattern of different users (upstream and downstream volume, frequency of visit);
4. Capability of interpreting normal traffic and detecting abnormal traffic at the node level, which is easy to understand.

In addition, similar to [5], the profile should be in a compact and concise representation, which means easy to manage and convenient for further analysis; it should also be easy to update for new behaviors.

In the literature, there are several initial studies that profile the node behavior in terms of traffic characteristics [20, 5, 8]. The basic idea of most current work is illustrated in Fig. 2.1, where different patterns represent different node behaviors at the corresponding time interval, “None” indicates the node is silent in that time interval.

In brief, most current work only profiles the node behavior on individual node within the time domain, neglecting the correlation between nodes. To achieve a good profile, they assume that most part of behaviors in continuous time intervals are stable. This could be true if the time interval is large enough (e.g. 1 day). However, to detect abnormal network traffic, small time intervals are preferred for more timely analysis and detection response (e.g. it is better to detect the worm during its early propagation than after its outbreak). When the time interval is small (e.g. 10 min), the above assumption might be too strong as node behavior will be
quite different at this granularity on two or more consecutive intervals. For example, current schemes may work well to profile node $k$ in Fig. 2.1, but will lose much information for node $i$ and $j$.

On the other hand, given the network applications/destinations are of different popularity, we observe that the popular applications/destinations are more likely to be used by different users at different times. Consequently, different users may share similar behaviors at different times (e.g. behavior at time 2 of node $i$ and behavior at time 1 of $j$ in Fig. 2.1). Therefore, in this Chapter, we build the node behavior profiles by considering correlations among different nodes and make the following contributions:

1. Study normal node behaviors using a newly proposed correlation based node behavior profiling approach. We also show in our evaluations that the behavior profiles are of different popularity, which is consistent with our observation.

2. Propose new metrics and study the impact of worm propagation on these metrics in popular behaviors. In particular, we are interested in the case where worms are used to
form new botnets. In such situations, multiple new worms (either new type of worms or polymorphic ones) can be used, and usually the propagation is from an initial set (e.g. a group of existing bots). Consequently, the use of multiple new worms indicates different ports will be used for propagation, which is hard to detect. Moreover, as there are enough hosts to be used for propagation, there will be no slow start phase and no need for fast spreading, which makes most traditional anomaly detection schemes less effective.

3. Evaluate the proposed schemes via real enterprise data. As mentioned in [10], despite the extensive study on wide-area Internet traffic, the traffic inside Internet enterprises remains almost wholly unexplored. Therefore we focus ourselves mainly on the enterprise network data in the evaluations.

The following of this Chapter is organized as follows: Chapter 2.2 describes the related work in the literature on node profiling for network security. We introduce the correlation based profiling scheme and candidate metrics for malware detection in Chapter 2.3. Our detailed evaluations are shown in Chapter 2.4. In Chapter 2.5, we discuss future directions of this work and make the conclusion.

2.2 Related Work

Profiling has been widely used in many areas of computer networks [7, 13, 16, 17, 19, 25]. We mainly focus ourselves on node related profiling in this section. Prior to node level profiling, many studies have been focused on the flow level profiling. In [17] [18] [104], the authors characterize the Internet traffic for traffic classification by profiling the traffic at flow level. Flow level features are considered during profiling, including flow duration, port number, inter-arrival time, bandwidth, payload size. Flows are further clustered into different groups such that traffic type from an unknown port can be identified if the flow characteristics match to a known traffic type group. However, as the Internet traffic varies broadly across different networks, these approaches either encounter performance challenges or produce unstable outputs for different
traces [8]. In addition, statistical metrics such as mean and variance used in above research papers require sufficient samples of a flow before traffic type identification, which is not suitable for short-term malware traffic.

Therefore, one solution to the above issues is to use node (level) behavior profiling. In the literature, node behavior profiling can be generally categorized as interest-based [23], entropy-based [4, 21], and attribute-based [5, 8, 20, 105, 106]. These classes are determined from their selection of features and the relevant values used for profiling. The interest based approach [23] aims at preventing the worm from fast propagation through building the host profile and a set of if-then like rules. The host profiles consist of different combinations of four tuples (protocol, srcIP, dstIP, dstport) from historical data. Discovering that historical data is not a good predictor for the usage of ephemeral ports, the authors in [23] set no constraint for node pairs if they have communicated via ephemeral ports in the profile. To find ephemeral ports, they associate two tuples (number of connections, number of servers) with all destination ports, and then use k-means clustering. In the evaluations, they show that their scheme can prevent the simulated worm propagation very well. However, it is not certain in their paper if the scheme can detect the worm propagation.

For entropy based approaches [4, 21], the main idea is to use entropy to summarize the feature distributions to lower the dimension of features. In detail, srcIP, srcport, destIP and destport are considered as four features for each given host. Each time interval the entropy of each feature is updated. By use of clustering the known attacks, the authors in [21] are able to find different synthetically injected attacks such as DOS and worm scan. In [4], by studying the entropy change of other three features with one fixed, the authors are able to extract important behaviors for that given feature. And in their evaluations, they show that it is possible to differentiate the normal behaviors from the abnormal behaviors.

For attribute based approaches, the authors in [20] propose BLINC to study node behavior profiles. It is a graphlet consisting of five tuples (srcIP, protocol, dstIP, srcport, dstport), and capable of modeling transport layer flow patterns of specific applications for each host. Given the pre-known flow patterns for different applications, BLINC is further used for traffic
classification at aggregate level. As shown in [20], it can classify approximately 80%-90% of the total number of flows in each trace with 95% accuracy. However, as pointed out by [5], this work is only suitable in a supervised manner, given the flow pattern of different applications already known.

An extended version of graphlet is further proposed in [5] by the same authors. By introducing an additional tuple (dstIP) to the original five tuples for the graphlet, and a set of rules (e.g. delay-accept, aging) in constructing activity and profile graphlets, they achieve a good balance of concise node behavior representation and meaningful characterization in their initial study on real user data. Similarly, in [8], the authors consider another five tuples (dailydestnum, dailybytenum, tcpport, udpport, communication similarity) to capture the node behavior. Rather than using graph representations, the profiles are in XML-like format and agglomerative clustering is used to group the nodes of similar values in those five tuples. Evaluations on real user data showed their scheme has the potential to detect worm outbreak. In addition, the authors in [105, 106] build a relatively simple host behavior profile which only uses the number of destinations contacted and a list of destination port numbers.

The authors in [102] propose an architecture for network security by sharing and reporting past behavioral patterns about network hosts. However, the behavioral patterns are only attack behaviors observed from network entities; no normal behavior pattern is considered. In addition, as pointed out in [8], such design brings in trust problem, and it can not detect the anomalies instantaneously with online traffic.

To summarize, although the above work did a good job in detecting the abnormal traffic, most of them are at the stage of using examples to illustrate the possibility of their schemes. Moreover, few of them evaluate their schemes thoroughly by considering the false positive rates. In the literature, the work in [8] is the most similar one to ours. However, our work mainly differs from theirs in the following ways:

1. The main difference is that we profile the node behavior in a way which can be coupled with further statistical analysis, and we consider a more complicated case where multiple new worms are used which could be further applied for botnet construction, all these
were not studied in [8];

2. As an important metrics, we evaluate the proposed detection metric by false positive rate, while the false positive rate is hard to measure hence not considered in [8];

3. We profile all the nodes in the network, while in [8], only about 17% of nodes have their profiles (they can only profile most active normal nodes, which may not be enough for malware detection at early stage);

4. We use a shorter time interval and jointly consider the time and node correlation to build profile, while they built the profile on individual node in an accumulative level at longer time intervals (1 day).

In addition, it is uncertain that behavior profiles in [8] can be interpreted in a reasonable way, but we have shown in this Chapter that it is easy to interpret network traffic at the node level by our approach.

2.3 Correlation based Node Behavior Profiling

In this section, we first discuss the common node behavior cycle from which we choose the attributes. We then formulate and simplify the behavior profiling problem. Clustering is used to solve the simplified problem. We also propose a simple heuristic to find the proper threshold in clustering. Node behavior based metrics that can be used for worm detection are further discussed.

2.3.1 Common Node Behavior Cycle

The node behavior cycle is defined as a sequence of basic actions of a node in the network. Unlike previous approaches which search for attributes of node behavior at different levels (packet, flow and node level), we begin by choosing the common node behavior cycle for normal nodes, and then identify important attributes in the behavior cycle.

The basic action sequences during the active time interval are as follows:
1. A normal user first decides which application/service he wants to use (e.g. in order to do some searching, HTTP is used).

2. Among those destinations providing the wanted service, he then determines which destination(s) he is interested in, this usually refers to the contents provided by the destination (e.g. to do a search, the user could use either Yahoo or Google, and he may choose Google).

3. Traffic is further generated when he visits the chosen destination.

One point to emphasize is that, most of time, the sequence of actions is very important and differs greatly between the normal user and the malicious attacker. For example, during worm propagation, any worm host may first generate traffic to find the available victims on certain ports, and then compromise them. It acts this way to account for the fact that traffic has to be generated to find victims. On the other hand, it is unlikely that a normal user (or a majority of normal users) uses the network this way (generate traffic to find available hosts). This difference provides inherent guidelines in selecting and simplifying the attributes for node profiling.

2.3.2 Attribute Selection and Problem Formulation

For each node behavior cycle, the first attributes of interest are the possible services to which the node initiates connections. Within each service, the attributes of interest are possible destinations a node can choose and corresponding traffic information for each specific service at that destination. Here we are especially interested in traffic information like the number of packets sent and the number of bytes sent or received. We do not consider the inter-arrival time for each service because it is very sensitive to network conditions, such as congestion level, routing policy, buffer management, and the location of monitoring. Therefore, it may not be a good indicator of host behaviors.

Using the identified attributes, the node/host behavior profiling problem can be formulated as: Consider a network consisting of $N$ end hosts, running up to $M$ applications, to $D$ desti-
Table 2.1 Parameters considered for attributes for host $i$ at interval $t$

<table>
<thead>
<tr>
<th>parameters</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$pkt_{i,j,t}$</td>
<td>number of packets generated for node $i$ on application $j$ at interval $t$, $0 \leq j \leq M$, $M$: total number of applications</td>
</tr>
<tr>
<td>$bs_{i,j,t}$, $br_{i,j,t}$</td>
<td>number of bytes sent and received for node $i$ on application $j$ at interval $t$</td>
</tr>
<tr>
<td>$pkt_{d_{i,j},t}$</td>
<td>number of packets generated for node $i$ on destination $d$ at interval $t$, $1 \leq d \leq D$, $D$: total number of destinations</td>
</tr>
<tr>
<td>behavior $x_{i,t}$</td>
<td>$pkt_{i,j,t}$, $bs_{i,j,t}$, $br_{i,j,t}$, $pkt_{d_{i,j},t}$, $0 \leq i \leq N$, $N$: total number of nodes</td>
</tr>
</tbody>
</table>

For any host $i$ at time interval $t$ ($t$ is the time interval used to extract host behavior, for a trace, assume $T$ such intervals in total), then the behavior for host $i$ at time $t$ is denoted as $x_{i,t} = \{(pkt_{i,j,t}, bs_{i,j,t}, br_{i,j,t}, pkt_{d_{i,j},t})|1 \leq i \leq N, 1 \leq j \leq M, 1 \leq t \leq T, 1 \leq d_{i,j,t} \leq D\}$ (the above parameters are described in Table 2.1), our objective is to find if there is any group of similar behaviors shared by different nodes at different times using clustering. Given the total time interval is $T$, we could have up to $N \times T$ such $x_{i,t}$’s, which is denoted as $X = \{x_{i,t}|1 \leq i \leq N, 1 \leq t \leq T\}$. We want to find clusters $A_1, A_2, ..., A_k$, such that:

1. $k << N \times T$

2. for $A_i, i = 1, 2, ..., k, \bigcup_i A_i = X$ and $A_i \cap A_j = \emptyset, i, j = 1, 2, ..., k, i \neq j$

3. for any $x$ in $A_i$, it is similar or has low distance to all other members in $A_i$, but is dissimilar or has large distance to members not in $A_i$.

This is a multidimensional dataset, and some dimensions are quite large, e.g. $D$ may be quite large. Therefore, we consider a simplified version of this problem and only examine behaviors using TCP, as most of worms will use TCP in propagation. To simplify the problem, we need to select attributes more favorable for normal behaviors while unfavorable for abnormal behaviors.

Firstly, normal users, although capable of choosing arbitrary destinations, usually associate themselves on a small range of destinations of different popularity (in-degree, a host in-degree
is the number of hosts connecting to it). However, for worm, the selection of destinations is random regardless of destination popularity. Therefore, we believe replacing destinations with in-degree will have negligible impact on normal behaviors. Further, we categorize the destinations as least, less, moderate and most popular categories (named as Q1, Q2, Q3, Q4 respectively), and only consider the corresponding packets generated in each category (represented as $pkt_{d,i,t}$). This is achieved by sorting the in-degree of destinations, dividing the in-degree into four quartiles and counting the packets in each quartile for each host.

Secondly, normal users have complete freedom in choosing the type of services, and the combination of such services can also be considered as normal behaviors. Therefore, we preserve all the service attributes for node profiling. Within each application/destination, several possible parameter candidates (e.g. number of connections, number of packets sent or number of bytes sent) can be further used to characterize the relative interest on application/destination choice of the node behavior. Among these parameters, the number of packets or bytes sent are much more accurate than the number of connections to represent node behaviors, as it is very likely that two nodes having different interests on a destination/application may have similar number of connections (but different bytes or packets sent). Further, as bytes mainly represent the accumulated level of packets, we believe the node behavior can be better characterized by the packets sent rather than the bytes sent. In addition, the number of packets is also used in other profiling schemes, including [21] [4].

Further, we consider the total amount of traffic (in bytes) generated and received using $tg_i$ and $tr_i$ for host $i$. Thus the redefined $x_{i,t}$ for host $i$ at time $t$ is:

$$x_{i,t} = \{(pk_{i,j,t}, tg_{i,t}, tr_{i,t}, pkt_{d,i,t})|1 \leq i \leq N, 1 \leq j \leq M, 1 \leq t \leq T, 1 \leq d \leq 4\}$$

By simplifying the problem in this way, we can reduce the complexity of analysis without losing too much information about normal node behaviors.

### 2.3.3 Behavior Clustering

To find out the common behavior profiles/clusters shared by different nodes, clustering is used in a way such that behavior correlations between times and different nodes are jointly
considered. In other words, we aim at finding the common profiles/clusters among all the \( x_{i,t} \)s in the corresponding network.

Consequently, we first identify the active hosts initiating the connection, because those hosts are the sources of most, if not all, network events \(^1\). We use the method described in [8] to identify the active hosts, which is based on the SYN packet. At the same time, we calculate the in-degree of each destination, and sort those destinations by their in-degrees. Then, at a fixed time interval, we calculate the values of the attributes for each active host, which is \( x_{i,t} \) in the problem statement.

We further remove the \( x_{i,t} \)s with \( tr_{i,t} \) equal to 0. By doing so, we can easily remove 1) the inactive \( x_{i,t} \)s, that generates nothing in the corresponding time interval, 2) incorrectly configured hosts, that may continuously send packets but get no response and 3) scanning hosts that do not get a response in the time interval. We also remove the traffic from ports rarely seen in the network (less than 1% of the total traffic). Doing this way, we can preserve the main characteristics of behaviors. Finally, we compute the log values of those attributes.

As we do not have a prior knowledge on how many clusters of behavior in the network, we use agglomerative clustering [6] to find possible clusters. The classic agglomerative clustering algorithm treats each \( x_{i,t} \) as a cluster at the beginning, then calculated the pairwise distance for any two points (or \( x_{i,t} \)s), and combined two points or clusters into one cluster if the distance between them is below a threshold \( T_h \). The criterion to compare the distance on combining clusters is the largest distance. The distance is measured by the extended Jaccard distance which is defined as [11]:

\[
d(x_1, x_2) = 1 - \frac{x_1' x_2}{||x_1||^2 + ||x_2||^2 - x_1' x_2}
\]

The extended Jaccard distance measures the degree of difference between two points. It also accounts both for angle as well as magnitude, so it outperforms Euclidean distance as shown in [11] in web-page clustering. It is also widely used in gene analysis [12] and automatic detection of malwares [123].

\(^1\)In practice, removing scanning traffic and other known malicious traffic should be done at the very beginning, removal had been done in [10] for the data set under consideration in this Chapter
2.3.4 Metrics to Measure the Abnormal Change of Node Behaviors

In general, abnormal behaviors could coexist with normal ones. By coexist, we mean the abnormal behaviors can not only be generated together with normal behaviors from an individual node, but also associate with other normal node behaviors in a network. In contrast, in our study, the behavior profiles are of different popularity (as shown in Chapter 2.4) and popular behaviors have relatively large sizes. Therefore, if there is a certain number of hosts infected by a malware, these popular behaviors will be impacted.

To measure the change of popular behaviors, intra-cluster distance, behavior profile size or its proportion value are among the possible candidate metrics. This is because when abnormal behaviors are present, two possible scenarios would happen for popular behaviors: one is that there are other normal behaviors which are closer to those infected behaviors, causing the change in size (hence proportion) of popular behavior profiles; the other is that the original behavior is still the closest one to those infected behaviors, but the intra-cluster distance is much larger. In the literature, there is no study to measure the proportion change of node behaviors, so we will mainly focus on the characteristics of this metric when worm is present. In addition, as shown in the evaluation part, unlike behavior profile size, the proportion value can be fitted in a well-know statistical model.

On the other hand, although many studies including [8] consider the intra-cluster distance as a metric in detection, they are mainly based on certain heuristics rather than well-grounded statistical models. Consequently, given the node behaviors are different in different networks; it is hard to apply the heuristics to be universally suitable to different networks. In addition, it is also difficult to control the false positive rates. We will propose new solutions for this problem in Chapter 4.

2.3.5 Discussions

To achieve meaningful and representative behavior clusters, we need a small $T_h$. However, if $T_h$ is too small, the clusters identified are of very small size. In the extreme case, each behavior is a cluster, which makes it hard to analysis network traffic at the user level. Therefore, a proper
Th is needed.

In our experiments, we find that behavior profiles are of different popularity (as the case in our analyzed data shown in Chapter 2.4), and we especially want this popular behavior profiles to be meaningful and stable to help the analysis. Therefore, we propose the following heuristic to find the proper \( T_h \): First find the largest \( l \) (\( l = 4 \) in our evaluations) clusters as the major clusters; instead of using only one threshold, we choose different threshold values, check the intra-cluster distances of the corresponding major clusters under each threshold, we then pick the largest \( T_h \) after which at least \( r \) major clusters are stable (\( r \leq l \)). In the next section, we will give an detailed example on finding the proper \( T_h \). We apply this heuristic to different subnets from real enterprise network trace and find that generally the choice of \( T_h \) is within \([0.2, 0.3]\).

With the behavior profiles identified, it is straightforward to build node behavior profiles for each node in the network. In detail, after identifying the representative profiles \((A_1...A_k)\), for any future observed behavior within certain time interval, we could find the closest \( A_i \) to it, as well as its deviation \( \delta \) to \( A_i \). That is, any node behavior can be recorded as two tuples \((A_i, \delta)\). A node profile may consist of a set of such two tuples. With just two parameters corresponding to the underlying behavior profiles to represent the node behavior in the future, the profile generated by our approach is concise and meaningful.

Another issue of this approach is the scalability problem during clustering. As the sample size \( n \) increases, \( n^2 \) distances need to be calculated for clustering. Recently, in [123], a scalable clustering scheme has been proposed which avoids calculations of \( n^2 \) distances, it is mainly based on locality sensitive hashing (LSH), since LSH provides a sublinear solution to the approximate nearest neighbor problem (\( \epsilon-\text{NNS} \)) [123]. Consequently, it is shown that the fast scheme is able to cluster more than 75 thousand samples in less than three hours. Therefore, a similar idea can be also applied in our approach, and we will consider it in our further studies.
2.4 Evaluations

2.4.1 Trace Information

We consider the LBNL enterprise trace data [10] which is publicly available and the latest enterprise traces to evaluate our approaches in this and following Chapters. The total traces are over 100 hours and cover 8,000 internal active addresses at the Lawrence Berkeley National Laboratory and 47,000 external addresses. As mentioned in [10], unlike a site’s Internet traffic, which we can generally record by monitoring a single access link, an enterprise of significant size lacks a single choke-point for its internal traffic. Therefore, in the LBNL trace collection [10], the authors mainly recorded the traffic from different subnets by monitoring (one at a time) the enterprise’s two central routers.

To capture the traffic, four NICs are used for four unidirectional traffic streams for two subnets. Timestamps are further used to merge each two unidirectional streams of one subnet into one trace. The four NICs then switch periodically such that they are able to capture traffic traces from 18-22 different subnets, with most traces being one hour long. We then choose several traces from different subnets (each subnet consists of multiple one-hour traces) as the training data, and other traces of the same subnet as the testing data.

Specifically, we consider the trace data captured in subnet002, subnet005, subnet017, subnet018 and subnet021 in this work (for clear representation, we replace the prefix port in the origin paper with subnet). The results on traces of different subnets are similar. As an example, we present one of them in the section, which is based on subnet002 in the LBNL trace. There are six traces for subnet002, which in total is 1.64 GB, over four hours long and consists of over 130 source nodes in total. In addition, since the trace data does not include the traffic within its own subnet, we only evaluate our approach by the traffic from and to the subnet of subnet002. The time interval used to extract node behavior is 10 minutes. To identify the services used in the subnet, wireshark [14] is used.
2.4.2 A Detailed Example

To illustrate how our approach works, we first give a detailed example on one trace, which is trace 1526(0107).

2.4.2.1 Determination of $T_h$

The dendrogram in Fig. 2.2 illustrates the general relationship among all the behaviors in the training data. Clearly, the dendrogram indicates certain structure among those behaviors, which means that profiling by jointly considering both time and node correlation yields meaningful behavior clusters. In Fig. 2.2, we can find there are many potential $T_h$ values that can be used for clustering. However, the dendrogram alone can not tell us which $T_h$ is proper in this network. Therefore, we need to find a proper value of $T_h$ so as to find good representations of those behaviors.

![Figure 2.2 Corresponding dendrogram](image1)

![Figure 2.3 Intra-cluster distance over $T_h$](image2)

We apply the heuristic discussed in the previous section to find a proper value for $T_h$. It is illustrated in Fig. 2.3 and 2.4, where we plot the cluster properties when $T_h$ is from 0.25 to 0.15, down at 0.02 interval.

To be specific, Fig. 2.3 and 2.4 illustrate the change of intra-cluster distance and cluster
size at different $T_h$ values. We can find that, the most significant change happens when $T_h$ value changes from 0.25 to 0.23, and 0.23 to 0.21, but there is no significant change after that (except for cluster 2). This is also true on cluster size change. In addition, given the intra-cluster distances of these clusters are very small, these clusters can be considered tightly coupled. Therefore, by setting $r = 3$ and $l = 4$, we find a proper $T_h$ which is 0.21 in this case.

### 2.4.2.2 Compression and Representation

Table 2.2 illustrates the compact representation of our methods. We list four largest traces (each is one hour long) of subnet002. The number before the bracket indicates the capture time, while the number in the bracket indicates the capture date. We define the compression ratio as the ratio of the total number of records of active nodes (uncompressed data) over the clusters identified (compressed data). The coverage is defined as the fraction of data used for clustering over the total number of data (similar to [5]). Since we consider both the number of bytes and packets generated by a node, the coverage is further divided into byte coverage and packet coverage. Finally, we pool all four traces and do clustering for the pooled trace; the result is shown in the overall column. To achieve a fair comparison, we set the distance threshold $T_h$ to be 0.21 for all traces.
Table 2.2  Compression and representation ratio

<table>
<thead>
<tr>
<th>trace information</th>
<th>0209(1215)</th>
<th>2146(1215)</th>
<th>1626(0106)</th>
<th>1526(0107)</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>compression ratio</td>
<td>19.45</td>
<td>24.06</td>
<td>12.48</td>
<td>8.69</td>
<td>21.68</td>
</tr>
<tr>
<td>packet coverage</td>
<td>98.11%</td>
<td>97.38%</td>
<td>96.29%</td>
<td>96.36%</td>
<td>97.23%</td>
</tr>
<tr>
<td>byte coverage</td>
<td>99.75%</td>
<td>99.99%</td>
<td>99.85%</td>
<td>97.53%</td>
<td>99.89%</td>
</tr>
</tbody>
</table>

From Table 2.2, we can find all the compression ratios are around or larger than 9. This means all behaviors across different nodes at different time intervals can be represented by about 12% (or less) behavior clusters of different types. The variations in these ratios indicate the activity diversity of the subnet at different times, e.g. high diversity in activities is expected in the day time, leading to low compression ratio. Consequently, the high compression ratio in each trace suggests certain overlap of behaviors within each trace.

On the other hand, all the coverage ratios are larger than 96%, which means our approach captures the majority of the node behaviors in this network. Moreover, since each trace is of same length in time, and the number of records in each trace used for clustering is similar, the relative higher compression and coverage ratio in the pooled trace (overall column) indicates the existence of behaviors overlap across different times in the same subnet. This also means the behavior profiles are stable across different times for a network. The results are similar when applying our approach to traces of other subnets. Hence it is reasonable to conclude (at least for the traces we have analyzed) that node behaviors in a network have correlations in both time and node domain, and clustering on behavior is effective.

2.4.2.3 Cluster Behavior Information

Fig. 2.5 shows the detailed cluster information when setting $T_h = 0.21$ for trace 1526(0107). In general, we can find large variation of cluster sizes, indicating their relative popularities. There are six clusters whose sizes are over 10. There is one dominant cluster indicating the
dominant behavior in this subnet (according to our analysis, different subnets tend to have different dominant clusters).

We then take a further look into the details of each cluster, and try to associate the real user pattern with each cluster. In particular, the main behavior in the largest cluster (id 1) is printing, with a large amount (around 3 in logscale, we will use logscale in the following analysis) of traffic going through the TCP 631 port (IPP protocol), coupled with a very small amount of IMAP (0.4) traffic. A majority of packets are directed to the destinations Q3 with large total traffic size in both incoming and outgoing directions. This suggests that, in this subnet during capturing period, the major behavior is printing associated with minor behaviors on email. Almost all behaviors are interested in the Q3 destinations. This is not surprising; as printing is a pretty normal task in such enterprise network and people would do something else while waiting for the printing results.

Some other clusters (id 7, 12, 15, 24 and 27) also have major behavior in generating IPP traffic, but they either generate much less IPP traffic than cluster 1 or have other major parallel applications. To be specific, the IPP traffic is around 2 less in cluster 12 and 27, and 0.4 less in cluster 15 and 24 respectively; additional HTTP traffic is generated in cluster 15 and 24, while HTTPs traffic is generated in cluster 7. Moreover, IMAPs traffic is much larger in cluster 24 and 27, and cluster 27 also generates traffic on port 8372, which is unassigned according to [124] and [125] (it might be the port for Trojan NetBoy.100 [126], it could also be the service port specifically used for that subnet, we will consider this issue in our future work). In addition, the choice of destinations is also different, a notable portion of traffic in cluster 7 is directed to destinations in Q1; cluster 24 is interested in destinations of all categories, and cluster 27 shows interest in destination Q1, Q3 and Q4, with a major interest in Q1. Especially, the packets from port 8372 seem to be entirely generated to Q1.

Another notable behavior is cluster 4. Unlike those behaviors in cluster 1, the major behaviors in cluster 4 are HTTP (around 2.2) and IMAPs (around 1.2), with large total traffic size in both incoming and outgoing directions. Less than half of the traffic in cluster 2 is directed to destination Q3 and more than half of the traffic is directed to destination Q1. This
represents one of the typical uses of HTTP, and the choice of destination Q1 and Q3 indicates information on user preference of HTTP and Email destinations.

Cluster 5, 8, 16 and 24 are similar to cluster 4 in generating similar amount of HTTP and Email packets. This is also true for cluster 9, 19 and 29, except no email packets are generated in these clusters. On the other hand, cluster 5 differs from cluster 4 mainly in the choice of destinations, where cluster 5 mainly generates packets to destination Q1 and Q4. Cluster 8 generates additional SSH packets, and the amount of SSH packets is similar to that of HTTP traffic. Cluster 9 and 19 differ from cluster 4 as they only generate HTTP traffic to destination Q4. Additional HTTPs traffic is generated in cluster 16, and the choice of destination is across all four destination categories.

In addition, cluster 3 is fairly quiet. Clusters 18 generates nfs (or shilp) traffic to destination Q2, while cluster 23 generates additional ssh traffic to Q1, Q2 and Q3. Cluster 2 and 26 only generate netbios-ssn traffic but different in volume and destinations (destination Q1 for cluster 2, destination Q1 and Q2 for cluster 26), which we believe is an automatic host behavior in the subnet. Cluster 10, 14, 17, 20, 21 and 30 generate mainly IMAPs traffic but differ in traffic volumes and selection of destinations. To summarize, all such clusters suggest the scenario of people doing their jobs in this subnet, but with different preferences in terms of additional associated services and popularity of destinations.

We also include the details of centroids of trace 1526(0107) in Appendix A.1 in this thesis. In brief, most results obtained by the proposed approach are explainable as reasonable user/computer behavior, suggesting the effectiveness of our approach. In addition, Table 2.3 shows how hosts jump between behaviors from time to time (0 means no activity in that time interval). We list the first five hosts with largest out-degree, it is very clear that some hosts dynamically jump from clusters to clusters, while some are stable on some specific behaviors. We also summarize the proportion of hosts that remain in 1, 2, 3, 4, 5 and 6 clusters within one hour time interval in Table 2.4. It is clear that around 49% nodes are relatively stable, because they only stay within one cluster, while about 51% of nodes are relatively dynamic, jumping between 2 to 4 clusters in the 1 hour time interval.
Table 2.3  Node behaviors at different time intervals

<table>
<thead>
<tr>
<th>node index</th>
<th>cluster id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 1, 1, 1, 1, 15)</td>
</tr>
<tr>
<td>2</td>
<td>(10, 24, 5, 5, 27, 27)</td>
</tr>
<tr>
<td>3</td>
<td>(4, 4, 4, 4, 4, 4)</td>
</tr>
<tr>
<td>4</td>
<td>(16, 8, 8, 8, 8, 8)</td>
</tr>
<tr>
<td>5</td>
<td>(10, 10, 5, 5, 0, 0)</td>
</tr>
</tbody>
</table>

Table 2.4  Summary of node activities in terms clusters within one hour

<table>
<thead>
<tr>
<th>1 cluster</th>
<th>2 clusters</th>
<th>3 clusters</th>
<th>4 clusters</th>
<th>5 or 6 clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>48.89%</td>
<td>20.00%</td>
<td>17.78%</td>
<td>13.33%</td>
<td>0%</td>
</tr>
</tbody>
</table>

2.4.2.4  Cluster Properties

To illustrate the distinct nature of the behaviors, we plot the average intra-cluster and inter-cluster distances for each cluster, as well as the corresponding standard deviations in Fig. 2.6 and Fig. 2.7. As shown in Fig. 2.7, each cluster can be well separated. Moreover, from Fig. 2.6, the intra-cluster distances of all the clusters are very small. However, there are clusters with 0 standard deviation, this is because the cluster only has size 1. There are two possible reasons: 1) the training data is not large enough to cover all those behaviors, or 2) the behavior is so rare and has no one similar to it. For example, the behavior in cluster 30 (with cluster size 1) is sending and receiving huge traffic (mainly on IMAPs) which is not common. One important observation is, the relatively small intra-cluster distance and standard deviation of some major clusters indicates the behavior in those clusters are tightly coupled and stable, which is useful in detecting abnormal behaviors if we can monitor the intra-cluster distance change in those stable clusters in future data set.

2.4.2.5  Detecting Worm Propagation

We generate simulated worm traffic (similar to [23]) as an initial test. To do so, we randomly pick up one time interval $t_1$ from another data set (trace 1626(0106)), we call the original training data set $t_0$. In $t_1$, in addition to generating normal traffic, all the hosts generate certain amount of worm traffic for propagation. This is the situation where a new worm is
spread out from an initial set of nodes (e.g. existing botnet).

To be specific, in addition to the normal traffic, we let all the active hosts in this interval generate worm behavior which uses one of the common TCP ports 21, 25, 80, 135 and 139 for propagation (e.g. W32.Blaster, Nimda, etc.) with small payload (5-15KB), low random scanning speed (5 hosts/minute), and 0.5% chance to find another vulnerable host for infection. In addition, we considered all the destinations chosen by worm are from the destinations in the training data. Therefore, the connection failure rate will be very small. In brief, we simulate a very stealthy worm for the environment.

We then check the cluster size distribution for the infected case, and compare them to the normal case. Fig. 2.8 shows the cluster size distribution of worm behaviors coupled with normal behaviors of $t_1$. Basically, the pattern of the cluster size distribution for $t_1$ is different to Fig. 2.5. We also calculate the proportion value of the first largest 6 behavior profiles, which are cluster 1 to 6 respectively. Further, we compare these values with the ones in the training data in a statistical way. That is, let $p_i$ be the estimate of the proportion of behavior $i$ in the node profiling from training data $t_0$, since the training data is large enough, the estimate can be considered accurate. Then considering any new time interval $t_1$ as a random sample from the same population, we can run the population proportion test (population proportion testing
In practice, the acceptance region is of special interest, since once we know the number of records in sample $t_1$, the acceptance region can be directly derived and used for detection: if the number of records in behavior $i$ is out of the range of its acceptance region, there is an indication of abnormal behaviors. Given the number of active nodes in this network is small, we mainly apply the small sample case test in [26]. The acceptance region $[a, b]$ at a given significance level $\alpha$ is given by [26]:

\begin{align}
\frac{\alpha}{2} & \leq Bino(a; n, p_i) \\
Bino(b - 1; n, p_i) & \leq 1 - \frac{\alpha}{2}
\end{align}

For example, we calculate that $p_1 = 0.175$ for cluster 1 in $t_0$. Given the total sample size of $t_1$ 76 in Fig. 2.8, at significance level $\alpha = 0.05$, we can find the acceptance region is within $[7, 20]$ for cluster 1. Similarly, we can find the acceptance regions for cluster 2 to 6 which are listed in Table 2.5.

From Table 2.5, it is easy to tell cluster 1, 4, 5 and 6 in Fig. 2.8 are within the acceptance region at $\alpha = 0.05$. However, cluster 2 and 3 are not within the acceptance region for the time
interval considered. They are within the rejection region, indicating the existence of abnormal behaviors.

Table 2.5  Acceptance regions for cluster 1, 2, 3, 4, 5, 6, $\alpha = 0.05$

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Acceptance Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster 1</td>
<td>$7 \in [7, 20]$</td>
</tr>
<tr>
<td>cluster 2</td>
<td>$1 \notin [2, 11]$</td>
</tr>
<tr>
<td>cluster 3</td>
<td>$0 \notin [1, 10]$</td>
</tr>
<tr>
<td>cluster 4</td>
<td>$1 \in [1, 10]$</td>
</tr>
<tr>
<td>cluster 5</td>
<td>$2 \in [1, 9]$</td>
</tr>
<tr>
<td>cluster 6</td>
<td>$4 \in [1, 8]$</td>
</tr>
</tbody>
</table>

Further, when looking into the intra-cluster distance distribution (cluster 1 and 6 for example), as shown in Fig. 2.9, the intra-cluster distance distributions of cluster 1 and cluster 6 of the worm case are much higher than the normal case, which also indicates abnormal behaviors. Especially, since cluster 1 is quite tightly coupled and the intra-cluster distance is always less than 0.1 in our experiment, the change in the intra-cluster distance provides much information about the simulated abnormal behaviors. On the other hand, considering the threshold we choose is 0.21, the large intra-cluster distance in cluster 6 also suggests the existence of abnormal behaviors.

In addition, the intra-cluster distances are also very large for cluster 25 and 29, but it is out of the scope of this research. We also simulate different worms by adjusting the spreading speed and payload size, in most cases, there will be changes in both proportion test and intra-cluster distances. Therefore, given the specific subnet profiles, our initial study shows it is possible to detect worm existence during its propagation. We can then look into the suspicious records for further analysis.

2.4.3  False Positive Rates

As different networks tend to have different characteristics, to test the false positive rates when proportion value is used for detection, in addition to subnet002, we consider multiple subnet traces each of which is of two-hour long (combining two one-hour long traces), ranging
Table 2.6 Proportion test results (1: reject; 0: not reject)

<table>
<thead>
<tr>
<th>trace</th>
<th>cluster id</th>
<th>results</th>
<th>trace</th>
<th>cluster id</th>
<th>results</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet002</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet003</td>
<td>(1,2,3)</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>subnet005</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet006</td>
<td>(1,2,3)</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>subnet008</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet010</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>subnet015</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet017</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>subnet018</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet021</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>subnet024</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
<td>subnet026</td>
<td>(1,2,3)</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>overall false positive rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0556</td>
</tr>
</tbody>
</table>

from 100 to 1100 nodes. In addition, we preserve all the application attributes identified in the profiling and consider the first 3 popular behavior profiles for the test. We use the last time interval in each trace as the testing data, and the rest data as the training data. The test results are listed in Table 2.6.

From Table 2.6, 2 out of 36 times (2/36 = 0.0556) that the test wrongly rejects the null hypothesis, which is not far from the significance level at $\alpha = 0.05$. Especially, the first popular behavior in subnet003 and the second popular behavior in subnet006 wrongly reject the hypothesis for the proportion test. But other results do show certain stability of the proportion metric.

We also apply our approach to detect the worm behavior in the above subnet traces. We use the same method described in the previous part to generate worm behavior in the last time interval of each trace, and use the left intervals of the same trace as the training data. An alert is generated if at least one out of three popular clusters fails to pass the test. The results are encouraging, as our approach can detect the existence of worm for all the traces in the experiments.

2.5 Conclusion and Discussions

In this Chapter, based on the observation of behavior correlation across the nodes and times, we proposed a correlation based node profiling approach for network traffic analysis. We also considered a heuristic to find a proper distance threshold to cluster the host behaviors
at different time intervals. The analysis on the real user data from the enterprise network suggests that it is easy to understand the node behavior using the proposed approach.

In addition, we also obtain several important observations:

1. The node behaviors are correlated across different times and nodes, which is consistent with our observation made in the beginning of this Chapter;

2. The node behavior profiles are of different popularities, this is consistent with the fact that the Internet destinations and applications are of different popularities;

3. At least part of the popular behavior profiles tend to be tightly coupled in terms of shorter intra-cluster distance, this is also consistent with the fact that general user operations (represented by our behavior profiles) on popular destinations and services tend to be similar (e.g. operations in printing or browsing a popular website like CNN), although they may be different in content (e.g. content of printing and detailed news browsed). Such stable behavior profiles can be further used for detection.

A consequent question based on the above observations is that, as nodes have different behavior types, does the traffic generated by different behaviors share the same characteristics or not? In the literature, most recent traffic generators model the traffic from the application level. That is, application is considered as the source of the network traffic. Models based on either individual applications or application mix are further proposed. However, as we can find in the common node behavior cycle discussed in this Chapter, the network traffic is not driven by applications, but node behaviors of different types. Consequently, modeling the network traffic from the node behavior perspective should yield better models in generating realistic network traffic. On the other hand, as extensive work has been done in traffic modeling, it will save much time if we can incorporate previous work with the node behavior based modeling. We will address these issues in Chapter 5.

In this Chapter, we further considered detection metrics based on statistical test for worm propagation. Our example on real user data suggests that the profiles have the potential to detect abnormal behaviors in the network when coupling with the statistical test. Moreover,
we calculate the false positive rates of the proposed detection scheme using different subnets in the trace data and achieve good performance. In practice, depending on the speed of the worm propagation, different significance level can be selected to adjust the sensitivity of the proposed detection scheme.

On the other hand, given the aimless nature in scanning and large payload in exploit during the worm propagation, potentially there are many ways to detect the existence of its propagation. For example, both measurement of proportion change and intra-cluster distance change in different behavior profiles will generate an alarm.

However, the detection scheme alone is not suitable to detect the C&C channel in botnets, especially P2P botnets. This is because that usually the bots in the botnet know whom to communicate with, and the command is small in size as compared to worm payload. Moreover, the bots can also generate additional traffic to mimic the normal behavior when doing C&C. Consequently, the change caused by the C&C behavior may not be significantly enough to generate the alarm by the same approaches used for worm detection. Therefore, more advanced approaches have to be designed to detect the existence of the C&C channel, which will be discussed in Chapter 4. In the mean time, understanding of different botnets is also important to design the corresponding detection approaches, and we will discuss it in Chapter 3.
CHAPTER 3. A Framework for P2P Botnets

Botnets, like worms in the past, are the most serious danger towards the Internet. To effectively protect against them, researchers should not only focus on the known ones, but also the inherent relationships among them and those to appear in the future. In this Chapter, we first propose a framework capable of characterizing the inherent relationships between all different kinds of current (existing and suggested in the literature) botnets as well as worms. Based on the proposed framework, we predict a new botnet that is called Loosely Coupled P2P Botnet (lcbot). It is stealthy and can be considered as an extension of the existing P2P botnet structures. We then conduct experiments to compare the performances between lcbot and other P2P botnets in the literature, and gain insight understanding of P2P botnets. We also discuss potential mechanisms to detect the existence of P2P botnets. To the best of our knowledge, we are the first to propose the framework for botnets, the lcbot concept in P2P botnet research.

3.1 Introduction

A bot is common parlance on the Internet for a software program that is a software agent \cite{32}. Traditionally, a botnet refers to a collection of bots, which are compromised machines running programs such as worms, Trojan horses, or backdoors under a common command and control (C&C) infrastructure. A botnet’s controller/botmaster (real person controlling the botnets) usually controls a group of bots remotely for nefarious purposes.

One of the main advantages of botnet usage is that bots can utilize different combinations of existing advanced malware techniques easily, such as keylogging and rootkit. Moreover, by use of exploitation or social engineering approaches such as email and instant messages, the botnet
can self propagate on the Internet to increase its size like worms. In contrast, unlike worms which are usually out of anyone’s control during propagation, botnets are much smarter that they can be controled by the botmaster, or sold to people who wants to use them. Therefore, it is widely believed to become the platform for many current attacks towards the Internet.

To make effective countermeasures against botnets, it is very important to not only study the existing ones of various kinds separately, but the inherent relationships among different botnets/worms (since most current botnets make use of worms to propagate), as well as the ones to appear in the future. There are several papers in the literature each of which designs a new botnet of some special kind [38] [42] [65], but none of them considered the inherent relationship between the botnet they designed with other existing botnets and worms, nor did they designed any experiments to compare different P2P botnets.

In this chapter, we address the above issues and make the following contributions:

1. Propose a general botnet framework for understanding worm and botnet of different kinds;

2. Predict a new lcbot from the botnet framework, and propose a propagation scheme by modifying an existing scheme in the literature;

3. Propose new metrics for botnet evaluations, use them with known ones to compare the performance of the new botnet with other known P2P botnets.

To the best of our knowledge, we are the first to propose the framework for botnets/worms, the lcbot concept in botnet and related fields. The Chapter is organized as follows: we first give a brief overview of botnets and the related work in the literature in Chapter 3.2. In Chapter 3.3, we propose a general framework which can characterize worms and botnets of various kinds. Based on the framework, we predict a new botnet called Loosely Coupled P2P Botnet. We also discuss the evaluation metrics and propagation methods for lcbot in this section. In Chapter 3.4, we evaluate the performance of lcbot with other P2P botnets proposed in the literature. We also discuss possible ways to detect the existence of botnet. The conclusion is made in Chapter 3.5.
3.2 Related Work

3.2.1 Botnet Overview

In the past several years, different kinds of botnet have been designed and captured. Table 3.1 lists several examples of well known botnets according to their first appearance in the wild.

Table 3.1 An example of well-known botnet history (slightly updated from [97])

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>C&amp;C protocol</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002/04</td>
<td>SDbot</td>
<td>IRC</td>
<td>Centralized</td>
</tr>
<tr>
<td>2002/10</td>
<td>Agobot</td>
<td>IRC</td>
<td>Centralized</td>
</tr>
<tr>
<td>2003/04</td>
<td>Spybot</td>
<td>IRC</td>
<td>Centralized</td>
</tr>
<tr>
<td>2004</td>
<td>Rbot</td>
<td>IRC</td>
<td>Centralized</td>
</tr>
<tr>
<td>2004/03</td>
<td>Phatbot</td>
<td>WASTE</td>
<td>P2P</td>
</tr>
<tr>
<td>2004/05</td>
<td>Bobax</td>
<td>HTTP</td>
<td>Centralized</td>
</tr>
<tr>
<td>2006/04</td>
<td>Nugache</td>
<td>Random</td>
<td>P2P</td>
</tr>
<tr>
<td>2008/03</td>
<td>Kraken</td>
<td>Self-defined</td>
<td>Centralized</td>
</tr>
<tr>
<td>2009/05</td>
<td>Torpig</td>
<td>HTTP</td>
<td>Centralized</td>
</tr>
</tbody>
</table>

According to [33], the damages of botnets can be classified as follows:

- Distributed Denial-of-Service (DDoS) Attacks: The botnet can be used to launch DDoS attacks against any Internet system such as a web server. The purpose of the DDoS attack is to use up all system resources (e.g. bandwidth) so that the system can no longer provide available services to normal users. According to our source code analysis of well-known botnets, the attacks can be in the format of Ping flood, UDP flood, spoofed UDP flood, spoofed SYN flood, HTTP flood, etc.. It is believed that all DDoS attacks are launched through botnets [97], and they are very hard to prevent and defend against, because of the size, the accumulated bandwidth, and the distributed nature of the botnet.

- Spamming: Each day, the spam accounts for several billion spam messages [97] in Internet traffic and is used to frustrate, confuse and annoy Internet users. According to [98], more than 95% emails on the Internet are spam. Consequently, given the size and distributed nature of botnets, most of these spam messages are believed to be sent from botnets. In
addition, some bots also implement a special function to harvest email addresses, and it is often the case that the spam you are receiving was sent from, e.g. grandma’s old Windows computer sitting at home [33], your workmates or friends.

- **Sniffing Traffic:** The bot on the compromised machine can use packet sniffer to watch for interesting clear-text data and gather sensitive information like usernames and passwords. In addition, other interesting information can also be gathered. For example, if the compromised machine belongs to two or more different botnets (infected more than once), it is possible to gather the key information of the other botnet, and steal it.

- **Keylogging:** When encrypted communications (e.g. HTTPS or POP3S) are used in compromised machines, sniffing alone is of little use as no key is available to decrypt the packets. Keylogging is then used to retrieve sensitive information like credit card number, passwords in this situation. For example, an implemented filtering mechanism (e.g. "I am only interested in key sequences near the keyword 'paypal.com'”) further helps in stealing secret data [33].

- **Spreading new malware:** Usually, to avoid the slow start phase, new malwares can be spread on top of the existing botnet. Lacking of slow start phase makes the new malware spread much easier and cause more harm. For example, the Witty worm, which attacked the ICQ protocol parsing implementation in Internet Security Systems (ISS) products is suspected to have been initially launched by a botnet due to the fact that the attacking hosts were not running any ISS services [33].

- **Clickfraud (Installing Advertisement Addons, Browser Helper Objects and Google AdSense abuse):** With large size and distributed nature, botnets can also help the botmaster to make profit easily by clicking the online ads. These clicks can be launched by receiving the corresponding command from the botmaster. Moreover, this process can be further enhanced if the bot hijacks the start-page of a compromised machine so that the "clicks” are executed each time the victim uses the browser [33].
• Attacking IRC Chat Networks: As most centralized botnets are based on IRC for C&C, they can be used to attack the IRC network. It is similar to DDoS attack, with the purpose to shut down the victim IRC network by massive service requests.

• Manipulating online polls/games: The bots can disguise to be different identities to win the game or poll. Given each bot having a distinct IP address, every vote will have the same credibility as a vote cast by a real person [33]. Online games can be manipulated in a similar way.

• Phishing (Mass identity theft): By sending spam emails that pretend to be legitimate (such as fake PayPal or banks), criminals usually trick normal users to visit the fake phishing sites (hosted by bots) so as to gather sensitive user information like usernames/passwords and credit card numbers on the real websites. This is the combination of different functionalities described above.

Traditionally, DDoS and Spamming are usually highly concerned. However, more and more applications from Clickfraud to Phishing are being used for profit purposes. For example, according to ClickForensics (http://www.clickforensics.com), traffic from botnets was responsible for 31.4% of all click fraud traffic in Q4 2008. That’s up from the 27.7% rate reported for Q3 2008 and the 22.0% rate reported for Q2 2008.

In addition, we have also included our study on SDbot and Agobot source codes in Fig. B.1 B.2 C.1 in Appendix 2 and 3. In brief, besides the damages mentioned above, from our analysis on the source codes of Agobot and Sdbot, we found the botmaster in Sdbot (with NB spreader) is interested in the Cdkeys like Half-Life CDKey, CSKEY, Neverwinter Nights CDKey. In general, there is usually more than one botmaster controlling the botnet, our analysis shows one version of Sdbot allows up to four botmasters to control the IRC channel simultaneously. This feature increases the botnet usage and makes it much harder for traceback. Other notable observations from the source code analysis include that in Agobot, up to four propagation methods are used, including scan.netbios, scan.locator, scan.dcom, and scan.dcom2; it will find and kill any antivirus processes in its list every 10 seconds if it fails to
join the IRC channel.

### 3.2.2 Centralized Botnets

Current research on botnets mainly focuses on the behavior of centralized botnets, such as IRC or HTTP based botnets. Fig. 3.1 illustrates the typical life cycle of an IRC based botnet.

There are mainly 5 steps in the botnet operation [127]. Vulnerable hosts are first infected by bots in the origin botnet; the infection can make use of any existing techniques (worms, Trojan horses, etc). As the IRC server may be detected, shut down; no hard IP address is embedded in infected hosts. Instead, only a domain name is embedded, which requires hosts after infection to first use DNS to find the IRC server. Then, infected hosts connect to the IRC server to join the IRC channel under the botmaster’s control. The botmaster is the creator of the IRC channel, who can then issue commands to listening bots. In addition to common IRC messages, the bots in the same channel will automatically parse and execute the command (usually in the format of private messages) from the botmaster. Authorization is achieved via a channel password. For HTTP based botnet, it differs from the IRC based botnet in the last step, where the HTTP bots will visit the server via HTTP protocol from time to time to check the latest command from the botmaster.

Recently, a new type of centralized botnet named Torpig [96] is captured and reported in the literature. It is also a centralized botnet, but with slight variations. According to [96], to increase the reliability of its C&C infrastructure, Torpig uses a more advanced technique which is called domain flux by the authors. With domain flux, each bot periodically and independently (but by the same algorithm and parameter values) generates a list of domains that it contacts. The first host that sends a reply that identifies it as a valid C&C server is considered genuine, until the next period of domain generation is started [96]. Therefore, given the same algorithm and parameter values used to generate the server list in each bot, all the bots will share the same list and check the same server at similar times. The botmaster can issue any command whenever needed, as long as he knows the corresponding time and server lists. By controlling a wild Torpig botnet for over ten days, the authors in [96] make several
important and latest observations on centralized botnets:

- They found that previous evaluations of botnet sizes based on the count of distinct IPs might be grossly overestimated. In particular, they found that the number of unique IP was one order of magnitude larger than the actual number of infected hosts.

- The victims of botnets are users with poorly maintained machines that choose easily guessable passwords to protect access to sensitive sites.

- The botnet targets a variety of applications, and gathers a rich and diverse set of information from the infected victims, such as bank accounts, credit card numbers and username and password pairs.

3.2.3 Advanced Botnets

Despite the advances in centralized botnets, many new features of botnets have been discussed in the recent literature, which make the existing countermeasures less effective. For example, in addition to P2P structures, encryption, use of multiple protocols for C&C are also within the main directions of the evolution. Encryption makes it more secure for a botmaster to
control the botnet, resulting in the inefficacy of schemes based on signatures or anomaly detections using character distribution (e.g., n-gram distribution). Normally, symmetric encryption is expected, but it is possible to make use of the PKI structure [128].

C&C by other commonly used protocols makes the communication among bots more covert [129]. Consequently, there are reports of botnets using Skype [61] and Gmail [62] in C&C. It is also possible that a botnet could use multiple protocols for one C&C cycle [129], making detection even more difficult. In Table 3.2, we summarize and compare the differences between latest and traditional botnets. In this chapter, we mainly focus on the discussion of the C&C structure of botnets. And the predicted botnet is believed to be compatible with the encryption and variation in C&C protocols discussed above.

In contrast, P2P structure makes the botnet robust and resilient to bot removal/repair. Therefore, several schemes for constructing different P2P botnets are proposed as hybridbot [65], randombot [38] and superbot [42]. [31] lists the timeline of captured botnets using P2P.

The main idea of P2P botnets is that each bot has a “buddy list” or routing information (we replace these terms with peerlist in this thesis) consisting of IP addresses of n other infected hosts. The peerlist in randombot is built in the following way [38]: When an infected host i infects a new victim j, its peerlist is passed to the victim. Host i chooses with a probability whether or not to replace one IP address in its own peerlist with host j’s IP address. If host j has already been infected before, host j updates a part of its own peerlist with the new one sent from host i. The peerlist construction of supernode in [42] is similar to [38] except that only client nodes can infect supernodes and entire peerlist information is exchanged.

Further, the botnets discussed above can be considered as “PUSH” based botnets. In contrast, the idea of botnet structure in [65] is similar to [42], except that the clients periodically communicate with any servant bot in their peerlist to grab the command. This can be considered as “PULL” based botnet. However, it is expected that the periodical connection in [65] yields extra information on detection and unexpected latency in command delivery. Therefore, in this Chapter, after building the framework for all the existing botnets and worms, we mainly concentrate on the evaluations of “PUSH” based botnets.
The most similar work in the literature is in [35], the authors in [35] did a thorough study on botnet structures and show that random botnet is highly resistant to different removals. However, they focus more on the structure difference of the known botnets. Our work focuses on the underlying relationship among different botnet structures proposed in the literature and can be considered as an extension of the previous work.

### Table 3.2 Evolution of bot techniques

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Botnet size</td>
<td>Large [50]</td>
<td>Small [50]</td>
</tr>
<tr>
<td>Bot bandwidth</td>
<td>Usually small [50]</td>
<td>Large [50]</td>
</tr>
<tr>
<td>C&amp;C</td>
<td>Centralized (IRC)</td>
<td>Fully or partially distributed [35] [38] [65] [42]</td>
</tr>
<tr>
<td>Authentication</td>
<td>Plain text</td>
<td>MD5 [36], PKI [65]</td>
</tr>
<tr>
<td>C&amp;C channel</td>
<td>IRC protocol</td>
<td>Various protocols, e.g. HTTP [127], Voip [61], gmail [62], random ports [129]</td>
</tr>
<tr>
<td>Communication</td>
<td>Plain text</td>
<td>SSL [62], encryption, information hiding or covert channel</td>
</tr>
<tr>
<td>Disguise IP</td>
<td>No</td>
<td>Yes, fake BGP route announcement [37]</td>
</tr>
<tr>
<td>Propagation</td>
<td>Pure worm-like</td>
<td>Propagate by command two-phase [38], superbot [42]</td>
</tr>
<tr>
<td>Systems</td>
<td>MS Windows</td>
<td>Windows Linux, Mac, etc.</td>
</tr>
<tr>
<td>Others</td>
<td>Kill Anti-Virus</td>
<td>Kill Anti-Virus Honetypot aware [38], NAT enabled [38], mimic human responding time interval more intelligent DNSBL lookup</td>
</tr>
</tbody>
</table>

### 3.3 Botnet Framework & Lcbot

#### 3.3.1 A Framework for Botnets/Worms

To make the analysis in the following sections more clear, we use the notion of rootbot to denote the first host (bot) receiving and sending commands in the botnet. Obviously, the rootbot has direct relationship to the botmaster and it is important to locate the rootbot to traceback the botmaster. Similarly, the server is the host (bot) capable of initiating connections for command delivery, and the client is the host (bot) only capable of receiving commands. For a network composed of either a worm or a botnet, each infected host $i$ is associated with three parameters $p_{si}$, $p_{ci}$, and $k_i$, which are defined as follows:
• \( p_{si} \in \{0, 1\} \): “Can the infected host \( i \) be a server in the botnet?”

• \( p_{ci} \in \{0, 1\} \): “Can infected host \( i \) be a client in the botnet?”

• \( k_i \): the number of hosts with which an infected host \( i \) can communicate.

From the viewpoint of communication in command delivery, we can integrate various botnets/worms into a framework by setting different values to these three parameters, as illustrated in Table 3.3.

<table>
<thead>
<tr>
<th>Table 3.3 Framework for Botnet/Worm</th>
</tr>
</thead>
<tbody>
<tr>
<td>General worm</td>
</tr>
<tr>
<td>IRC bot</td>
</tr>
<tr>
<td>HTTP bot</td>
</tr>
<tr>
<td>Torpig bot [96]</td>
</tr>
<tr>
<td>Randombot</td>
</tr>
<tr>
<td>Superbot</td>
</tr>
<tr>
<td>Hybridbot</td>
</tr>
</tbody>
</table>

From Table 3.3, we can easily differentiate worms or botnets of different kinds. To be specific, since a worm needs not to be controlled, any worm host will neither be a server, nor will it be a client. In other words, it is independent and maintains no information about others. For a centralized bot, all the infected hosts need to connect with a common IRC server for C&C. For any host, the parameter to be a client is 1, while there is no chance for any of them to be a server, nor do they maintain a list of others except the IRC server.

For randombot, each infected host is a server, and meanwhile, it is a client of other bots during command delivery. Therefore, both \( p_{si} \) and \( p_{ci} \) are 1, and each host maintains a peerlist of others for communication. For commands to be successfully relayed to all bots in the botnet and robustness concerns, \( k_i \) should not be too small (if \( k_i = 1 \), the botnet is either a circle or
partially connected). In addition, for any \( i \neq j \), the larger distinction of the IPs in the peerlist of \( i \) and \( j \), the better.

In the case of superbot, certain number of hosts with \( p_{si} = 1 \) choose to be a supernode, the number of super nodes is determined by the botmaster. All the hosts are clients: they are either the clients of a supernode, or the clients of multiple super nodes, indicating all \( p_{ci} = 1 \). To ensure the network is well connected, \( k_i \) needs to be large enough. For hybridbot, according our definition, each host can grab commands from bots in its peerlist, resulting in all \( p_{si} = 1 \). On the other hand, the servant bot in [42] is also a client in our framework, it has \( p_{ci} = 1 \), and all the other bots have \( p_{ci} = 0 \).

It is worth mentioning that, although the Torpig bot is discovered after the publishing date of our paper, it is still a variation from our framework and hence can be captured by the framework. This further validates the effectiveness of the proposed framework.

3.3.2 Metrics for A Good Botnet under the Proposed Framework

For any botnet/worm considered in the above framework, the key objective is to hide itself as covert as possible; be robust against portion removal (random or select), be hard to be traced back when part of it is detected, be efficient as much as possible in command delivery, which are discussed in the following.

By covert, we mean there should be no obvious anomaly in communications among bots, especially during C&C. Since the botmaster needs to issue the command to control the bots in the botnet to launch attacks, if the attack is active attack (DDOS), once the communication is detected, security professionals will have more time to take relative actions against it or even stop it before it happens. If the attack is passive attack (keylogging), certain alert can be announced beforehand to warn the public and prevent unnecessary loss. So in any sense, the covert issue is one of the most important factors when designing the botnet.

As the anomaly can be in both content and behavior, we only consider the one in behavior. Because it is relatively easy to use encryption or polymorphism to remove the anomaly in content. Consequently, we use the C&C overhead ratio (or) to represent the level of covertness.
It is defined by the number of messages actually generated when issuing a command divided by the minimum number of messages needed to issue this command. Take an example of randombot, since each bot maintains a list of $k$ other bots, for a botnet of size $M$, there will be $kM$ command messages generated within a short time. However, the minimum number of messages needed is $M$. If $M$ is large, say $10^4$, and $k$ is chosen to be 22 (one version of Nugachebot [129]), from the help of the network monitor system used in worm detection, it is much easier to recognize the C&C behavior when $2.2 \times 10^5$ duplicated command messages are generated by $10^4$ hosts than only $10^4$ messages are generated. On the other hand, it is worth mentioning that centralized botnet only needs $M$ messages in C&C, indicating the ratio is 1.

By robust, we mean even after removing certain portion of bots, the command can still be delivered to the majority of the remaining botnet. This removal can be either active or passive. Active refers to the case where bots are removed by the security professional permanently, or patched with the latest update. Passive refers to the case where bots are removed from the botnet due to other factors like power shut down or network connection unavailable from the victim side.

We use the mean coverage ($mc$) to represent the robustness of a botnet. The coverage refers the maximum percentage of bots botmaster can reach after each removal (random or select). Consequently, the mean coverage is the average value of coverage over all the removals. It is important to use mean here, as this maximum percentage is not a fixed value and tends to vary at each removal. If the $mc$ is small, the botnet may be divided into multiple pieces, each of which is independent. Similar metrics are also considered in other botnet studies [35].

By hard to be traced back, we mean once a bot is detected or honeypot is placed into the botnet, it should be practically impossible to trace back the rootbot. This is because if there is a high possibility to trace back the rootbot within a short time, similar to the centralized case, it is possible to find the botmaster. In other words, the botmaster will prefer the botnet with lower trace back probability on rootbot. We use the traceback probability ($tp$) for this purposes. It is defined by the probability of detecting the rootbot in the botnet. Obviously, the detection of rootbot can be either from the detection of rootbot itself or detection of the
bots that the rootbot connects to. In addition, \( tp \) can be affected by many factors, such as the number of IPs in the peer-list, the number of honeypots under our control.

By efficient, we mean under low \( tp \) rate, how fast the botmaster can relay the command to all the bots in the botnet. This is also important for decentralized botnets, as a good bot army should be responsible to any command. Therefore, we use the \( speedcdf \) in terms of hops to represent the command spreading speed. On the other hand, it is important to keep in mind the precondition of low \( tp \). For example, the centralized bot is extremely fast in command relay (one 1 hop is needed), but it also the easiest one to locate the centralized server.

Under the botnet framework, it is easy to correlate the parameters with one or more above metrics. Generally, if the number of bots having \( psi = 1 \) is large, it indicates a large portion of bots will be the server, the \( mc \) will be high or more robust. The \( or \) will be also high, or more easily to be detected. On the other hand, there should be at least a portion of bots having \( psi \) equal to 1 to make the botnet function properly. Further, if \( ki \) is too high, the command speed is faster, which also results in higher value of \( tp \) (easy to trace back). However, if \( ki \) is too low, \( mc \) will be low indicating the network may not be well connected.

### 3.3.3 Predicted Botnet (lcbot)

From above discussions, the values of \( psi \) and \( ki \) are important to current botnets. On one hand, the botmaster wants the number of bots having \( psi = 1 \) and \( ki \) as low as possible to make the C&C control more covert. On the other hand, given certain portion of bots in the botnet will be turned off or cleaned at any time, these values have to be large enough to maintain the connectivity within the remaining botnet.

Normally it is expected that attackers can adjust the above values to balance the tradeoff in these proposed botnets under specific situations. However, in the real world, instead of balancing the tradeoff between \( psi \) and \( ki \), we believe attackers are smart enough to design more intelligent botnets. In other words, they can achieve a good stealthiness level and high coverage.

The lcbot is one such example, and it is directly derived from the proposed framework.

The basic concept of lcbot is to consider the botnet being composed of many groups of
different group codes, and decouple $p_{si}$ into $p_{isi}$ and $p_{osi}$. $p_{isi}$ is defined as the parameter of being a server inside its own group, $p_{osi}$ refers to being a server (or outlink bot) to bots outside its own group. Any bot in the lcbot has $p_{isi}$ equal to 1, and the peerlist contains all the other bots in the same group. Within each group, a small number of bots have $p_{osi}$ equal to 1, and each of these bots has only one peer in other groups to communicate with.

No current schemes differentiate these two parameters clearly, but it is possible to achieve a good covertness level and high coverage through this decoupling. Fig. 3.2 illustrates different possible patterns of the lcbot. Specifically, case 1 refers to the most compact mode, where each bot in a group connects to one bot in other groups and can be connected by one bot in other groups. Case 2 refers to the compact mode, where each of five bots (black, called outlink bot in the following) is able to connect one bot in other group, and each of the other give bots (gray) can receive the command from the bot in other groups. Case 3 refers to the redundant mode, it is similar to the compact mode, but there is a certain number of bots (white) that can not connect or being connected by the bot in other groups. In addition, all the bots within a group can communicate with each other. There are many possible ways to construct such botnet. Therefore, in this Chapter, we mainly focus on the final structure of lcbot and consider a propagation scheme modified from [42] for its propagation.

One obvious advantage of this structure is that, like superbots or centralized botnets, only a small number of messages are needed for C&C. On the other hand, it is robust as randrobots when there is bot removal (further evaluation results are provided in Chapter 3.4). In addition, if we want to control the number of bots taking part in the attack, we can simply specify the code of each group, all bots in a group will launch the attack if the code for this group is included in the command. Encryption or unrelated data can be added to avoid the detection.

Further enhancements on command relaying can be added to it. That is, if a bot is an outlink bot, it can send the command to the outpeer (say group B) with a time-to-live value 2. The bot receiving this new command in group B chooses a peer randomly in B, sends the command to that bot, decreases the time-to-live value to 1. Consequently, the bot receiving this new command with time-to-live value 1 broadcasts the command to all other group members.
If the bot receives duplicate commands, the commands will be discarded. By doing so, we can lower the chance of being detected in presence of honeypots. As honeypots can tell who relays the command to them, there is a chance that a honeypot is in the peerlist of the rootbot. The chance of being detected is much lower if the rootbot does not broadcast to all its peers. Since the botnet is believed to be reused multiple times, and the botmaster can choose any bot known to him as rootbot, the botmaster has to pay great attention not to be detected due to reuse.

### 3.3.4 Propagation Method

The lcbot is open to any propagation scheme or can be established from any existing centralized botnet. As an example, we consider a propagation scheme modified from [42] for lcbot propagation, which is illustrated in Fig. 3.3.

To summarize, step 0 and 1 decide the necessary parameters for establishing the botnet. The code pool and the values of $M_{bot}$, $N_{group}$ and $N_{og}$ are all chosen by the botmaster. Step 2
Step 0: Preinfection
Botmaster determines the number \((M_{\text{bot}})\) of bots he wants to infect, the number of groups \((N_{\text{group}}, \text{each group is associated with a group code})\), and number \(N_{\text{og}}\) of outlink bots. Then start with an infector (we name it bot\(_{A,1}\)) with the current group code, say A.

Step 1: Infection new hosts
On infecting a new victim (e.g. bot\(_{A,2}\)), the bot (bot\(_{A,1}\)) updates its inpeerlist (peerlist within group) with bot\(_{A,2}\), and divide the number of remaining bots \((M_{\text{bot}}/N_{\text{group}} - 2, \text{in this example})\) to form group A by 2. Then, bot\(_{A,1}\) sends the group code, the inpeerlist, the number of new bots needed to the new bot (bot\(_{A,2}\) in this case). Repeat step 1 till no more bots are needed to form the group. In addition, schemes to detect the VMwares or Honeypots can be used before the success of infection.

Step 2: Establish group
All the bots in group (e.g. group A) communicate with peers in its inpeerlist to update within-group peerlist information. And choose to be the outlink bot with probability of \(N_{\text{og}} \times N_{\text{group}} / M_{\text{bot}}\). If no more group is needed to infect, go to step 4.

Step 3: Establish consequent groups
The infectors in the remaining \(N_{\text{group}} - 1\) groups are further infected by the bots in group A (each bot in A infects upto \((N_{\text{group}} - 1) \times N_{\text{group}} / M_{\text{bot}}\) infectors. Each infector in other groups repeats step 1 and 2.

Step 4: Establish lcbot
All the outlink bots keep scanning. When finding a bot having a different group code, the outlink bot replace its outpeerlist (peers outside its group) with probability of 50%. The scanning stops after enough time or enough number of bots visited by the outlink bot. Other bots can also be used to in this step to find the bots in other groups [42].

Step 5: Be dynamic
Each outlink bot \(i\) will decide to pass his outpeer information to its group peer by a probability function \(P(t)\), which is related to the time \(t\). Once \(P(t)\) is larger than a threshold e.g. 0.5, a new bot is randomly chosen in the same group and the outpeer is sent to the new bot. The previous bot deletes the outpeer information and is no longer an outlink bots.

Figure 3.3 Algorithm Description
and 3 describe how a group is established. Different combination of worms can be used in these steps. Step 4 shows how to build each group into a botnet. Step 5 enables a bot to change the role of whether to discard the outpeer with a probability depending on time. Algorithms similar to domain flux can also be used in Step 5. Once the probability is over a threshold, it will pass the outlink information to another peer in the same group. By dynamically changing the outlink bot, the importance of the specific outlink bot is further decreased, making it more robust against removal.

3.4 Evaluations

To analyze and compare the performance of the proposed botnet with others in the literature, we consider the botnet of size 1000 (consists of 1000 online bots), as we believe it is the common size of the current botnet. The botnets chosen for comparison are randombot [38] and superbot [42]. Matlab is used to obtain the simulation results. For each metric under consideration, we simulate 1000 times and average the results. All bots considered in this Chapter are homogeneous. For superbot, we consider the group size to be 25, which is same as the group size of lcbot. All bots in randombots, or supernodes in superbot, have peerlist of size five to communicate with other randombots or supernodes respectively. Accordingly, each group of the lcbot has five outlink bots each of which can communicate with one peer in another group.

3.4.1 Evaluation of the Propagation Scheme

A good propagation scheme should spread the bots evenly in the peerlists. In other words, the distribution of parents (number of bots sending command to a bot) shared by each bot should be balanced. This is because the unbalanced parent distribution can leak additional information on the botnet. For example, even if the peerlist size is the same for all bots in the botnet, if the bot is not selected by the peers in a uniform way during propagation, once the bots having larger number of parents are captured, it is quite similar to centralized botnets. As it is easy to find a large portion of other bots in the botnet, which leads to the possibility
to stop the whole botnet. Ideally, we want the parents to be evenly distributed in each bot after propagation.

![Figure 3.4 Distribution of # of parents](image1)

![Figure 3.5 Variation of # of parents](image2)

Fig. 3.4 and 3.5 illustrate the distribution of parents when the propagation scheme is applied. We can find that, most bots in the botnet are the parents of other 24-26 bots. And the variation is pretty small, this also confirms the correctness of the propagation scheme in [38].

### 3.4.2 Performance Comparison of P2P Botnets

#### 3.4.2.1 Mean Coverage

To obtain the $mc$ values for different bots, we consider both random and select removals. The removal happens either when the bot host is turned off or lost network connection. Therefore, the random removal is considered to happen randomly; indicating any bot in the corresponding botnet has the same probability to be removed. On the other hand, the select removal refers to the removal of important bots (e.g. supernode in superbot).

Fig. 3.6 and 3.7 show the mean coverage of randombot, superbot, and the proposed botnet when random removal is present. In Fig. 3.6(3.7), the peerlist size is 4(5) for randombot,
the peerlist size among supernodes is 4(5) for superbot, and the number of outlink bots for a group is 4(5) in lcbot. As we can find, in presence of small removal, the proposed botnet achieves highest mean coverage. However, when the percentage is larger than 0.2, there is a sharp decrease in mean coverage for our scheme. This can be explained as follows, when the removal increases to certain value, the nodes that outlink bots connect to might be removed. Therefore, even these outlink bots are still active, the mean coverage is as bad as in superbot in the worst case. One direct enhancement is then to enable the outlink bot to have more than one peers, and choose one at a time.

Fig. 3.8 shows the case where select removal happens. The important nodes are the supernode or the outlink bot in lcbot. As we can find, since the important node in lcbot is less important than the supernode in superbot, it achieves a high coverage when the number of select removal is same.

### 3.4.2.2 Overhead Ratio

Another important metric for P2P botnet is the level of covertness during C&C, since the more number of the same or similar commands; the more likely the C&C can be detected. Fig. 3.9 shows the overhead ratio of each botnet via simulations. We can find that the overhead
ratio of randombot increases quickly with the increase of peerlist size. On the other hand, the overhead ratio values for the other two schemes keep low and nearly unchanged under different peerlist size. This tells us the randombot is more likely to be detected during C&C. In terms of this metric, superbot and lcbot achieve similar performances.

3.4.2.3 Traceback Possibility

Further, we consider the possibility of being traced back. If the bot in the first and second hop of C&C happen to be the honeypots, it is possible to locate the rootbot. It is analogue to the case in the centralized botnet where IRC server is located. And usually there will be more than one honeypot in the botnet. Consequently, the traceback probability is given by:

$$tp = 1 - (1 - p)^{n \times N}$$

where $p$ refers to the probability that at least one honeypot happens to be within the first two hops in C&C, $n$ is the percentage of the honeypot in the botnet, and $N$ is the total number of bots in the botnet, which is 1000 in this evaluation. In our case, given the size of the botnet, the size of the peerlist in each bot and the C&C methods discussed in lcbot, we can find the $p$ for randombot and lcbot is 0.006 and 0.002 respectively. For
superbot, we consider two cases: case 1 refers to the situation where the botmaster only knows the super nodes, case 2 refers to the general case where any bot is known to the botmaster. In case 1, as each supernode can communicate with 24 nodes in the group and 5 other super nodes, \( p \) is equal to 0.03. In case 2, since each node can be chosen randomly, \( p = \frac{\text{average#of honeypots in the first 2 hops}}{\text{botnet size}} \) = \( 1 + \frac{p_{\text{supernode}}N_{\text{supernode}} + p_{\text{clientnode}}N_{\text{clientnode}}}{N} \). Where \( p_{\text{supernode}} \) and \( p_{\text{clientnode}} \) refer to the probability of being supernode and clientnode respectively, and \( p_{\text{supernode}} = 1 - p_{\text{clientnode}} \).

We can find from Fig. 3.10 that, given the same traceback probability, lcbot requires more honeypots to be placed in the botnets than others. This shows that the lcbot is less likely to be traced back when compared with other P2P botnets. As we can find, the superbot case1 has the lowest performance, because the number of super nodes is small and it is very easy to locate the supernode. On the other hand, the superbot case2 outperforms the randombot, indicating in the general case, randombot has the worst performance in traceback probability for the botnet parameters used in our evaluation. It is also worth mentioning that, if randombot switches to the C&C method discussed in lcbot, the traceback probability will be the same with lcbot. Accordingly, the speed of C&C in randombot will also be the same with lcbot, which is shown in the next section.

![Figure 3.10 Traceback probability](image1.png)

![Figure 3.11 CDF of C&C speed in hops](image2.png)
3.4.2.4 C&C Hops

Fig. 3.11 shows on average how many hops are needed for a command to be delivered to the whole botnet. The proposed botnet has the lowest speed in terms of hops. However, when regarding to time cost, since the end-to-end delay of each hop may be in the amount of seconds, 15 hops in delay may only cause several seconds or minutes, which is tolerable. On the other hand, the fewer the hops, the more likely that the rootbot can be detected. For example, the centralized bot only have 1 hop to all the bots which makes the server (HTTP or IRC) easy to detect. So we believe the hops required by lcbot are still acceptable.

To summarize, with regard to the above metrics, the lcbot demonstrate better performances when compared with other two, and it is pretty likely that this kind of botnet will appear in the near future.

3.4.2.5 Potential Ways of Detection

In general, to detect the membership of botnet, we believe it is important to understand the life cycle of botnet such that proper schemes can be designed for specific phases. In addition to [127], we classify another five phases for botnet which is useful in detection. The five phases are propagation phase, C&C phase, silent phase, updating phase, and attack phase. Propagation phase refers to the time that a botnet spreads to form a new botnet; C&C phase refers to the period of command delivery; silent phase is the time botnet waits for commands from the botmaster; updating phase refers to the time botnet is updating itself to more advanced one; and attack phase refers to the specific actions (DDOS, etc.) required by the botmaster in the C&C phase.

Among these five phases, updating is also an action after C&C, but it causes no damage to the Internet, so we classify it as an independent phase. Consequently, of these five phases, it is nearly impossible to detect botnet in silent phase, because all bots behave like normal hosts. In the attack phase where attack is launched, it is usually too late for detection. In contrast, mitigation is more important in this phase. Therefore, proper detection schemes can be designed on the other three phases, which are propagation phase, updating phase and C&C
We will discuss the detection on C&C in the next chapter. In this section, we only consider the propagation phase. From the viewpoint of infection during propagation, there is always a vulnerability used for infection. Unlike other detection schemes proposed in the literature which is based on anomaly of botnet from the normal traffic, possible detection can be designed in a reversed way: there should be something botnet can not have, while it always exists in the normal traffic.

Take an example of buffer overflow, which is a commonly used vulnerability in various infections (bot/worm), the contents bot/worm can not have but normal traffic can have are: \(x00, x0A\ 0x0D, 0x5C\) (optional) and \(0x90\) (optional, if too many \(0x90\), there may exist shellcode). This is because usually shellcode is used in buffer overflow. To have the shellcode work, \(x00\) is not permitted, since it indicates the end of buffer, resulting in all the following codes unable to be executed. On the other hand, normal traffic does not have such kind of limitation as they never need to run this way. \(x0A\ 0x0D\) are \(r\) which are frequently used in the protocol headers (like HTTP) for information exchange. But \(r\) means enter when used in shellcode, leading to the uselessness of the codes afterwards. \(0x5C\) is not allowed for certain bot/worm, like Agobot. To better illustrate what we discuss above, we attach a piece of Agobot code in Fig. 3.12.

As we can find in the above code, as long as there is any of \(x00, x0A\ 0x0D, 0x5C\), the shellcode is XORed with the xorkey, which is \(0x98\) in the code. On the other hand, since the buffer overflow usually happens when victim machine is handling the malicious request, the request is always associated with certain protocol headers (like RPC, HTTP). Therefore, there should be enough \(x00, x0A\ 0x0D, 0x5C\) among the normal requests. In addition, to enable network communication, network related function like socket; bind should be used in the shellcode, leading to the length of shellcode larger than the corresponding normal request in certain protocol (e.g. GET in HTTP).

Based on the above discussion, a threshold based scheme to detect bot/worm can be used. For example, we count the number of \(x00, x0A\ 0x0D, 0x5C\) and calculate the ratio of this
number to the entire length of the request. If the ratio is less than a certain value, shellcode should exist in the request. Of course, different protocols may have different threshold values. Further implementation and evaluation of this scheme is considered as one of our future work.

### 3.5 Conclusion

In this chapter, we first propose a framework integrating the botnet/worm for a thorough understanding on the inherent relationships between botnets and worms. We then predict a new botnet capable of achieving a better overall performance in terms of the metrics we concerned so far and the observation is further supported by the simulations. In addition, although it is hard to detect the existence of P2P botnets, we discuss potential mechanism that can be used for botnet detection. Consequently, in the next Chapter, we further extend this research to detect the C&C commands in advanced P2P botnets by making use of the node behavior profiling approach proposed in Chapter 2.
CHAPTER 4. Node Behavior based C&C Channel Detection for P2P Botnets

Due to the distributed nature and controllable characteristic, botnets are believed to be the platform for most current attacks towards the Internet. We have studied the relationships between different botnet structures and compared their performances under different metrics in Chapter 3. In this Chapter, we consider the problem of botnet detection. It is one of the most important problems in botnet research, as many techniques such as mitigation and traceback are directly based on the detection of botnets. In the literature, most recent studies on botnet detection primarily rely on two assumptions: prior knowledge of potential C&C channels (e.g. IRC channel) and capability of monitoring them. However, when botnets switch to a P2P structure and utilize multiple protocols for C&C, the above assumptions no longer hold. Consequently, the detection of P2P botnets is more difficult. Little work has been done in this direction in the current literature.

In this chapter, we relax the above assumptions and focus on C&C channel detection for P2P botnets. We first use the proposed node behavior profiling approach to capture the node behaviors in a network; then propose two anomaly detection schemes using formal statistical tests on popular behaviors in this network. In brief, the key difference between our work and previous work in botnet detection is that, instead of trying to filter out the botnet behaviors in a network, we detect them by measuring their impacts on one or more normal behaviors in a statistical way. Moreover, we make use of SPRT to measure the impacts of the C&C behavior from P2P botnets ($H_1$) on normal behaviors ($H_0$) in a fast manner. This is achieved by simplifying the node behavior profiling approach suitable for SPRT for fast detection. Furthermore, by considering the worst case attack model where the botmaster knows
all the parameter values used in SPRT detection, we formulate the SPRT-based C&C detection problem as an optimization problem and derive the optimal values of $H_1$ mathematically. An SPRT-based fast and optimized detection approach is further proposed. We validate the assumptions made in this Chapter under different real user traces from enterprise network environments. To evaluate the performances of the proposed approaches, we consider both simple and realistic cases and achieve encouraging results in terms of high detection and low false positive rates. To the best of our knowledge, we are the first to consider the worst case attack model and propose corresponding solution that derives the optimal value of $H_1$ for SPRT in botnet related research.

4.1 Introduction

“A botnet is comparable to compulsory military service for windows boxes” - Stromberg [33]. According to [34], a typical botnet is a network geographically distributed, consisting of 40 to 400,000 computer systems with average of about 2,000 to 10,000 machines. The largest documented botnet yet is believed to be comprised of 1.5 million compromised PCs [36].

The botnet has also spread widely across different carriers and platforms in recent years. According to the Cyber Threats Report 2009 from www.gtiscsecuritysummit.com, about 15 percent of all online computers are infected with bots. It is even possible to see the botnet problem infiltrate the mobile world in 2009 [116]. Moreover, in addition to Windows systems, security researchers have also discovered that payloads delivered by Trojans in pirated versions of iWork and Photoshop earlier 2009 are being used to create a Mac botnet [117]. On the other hand, the new trend of botnets is characterized by smaller botnet size, more advanced C&C (P2P structure) [38], higher capacity/ability of each bot [52], and stronger self-resilience [38].

Most current research on botnets focuses on C&C detection for centralized botnets, usually under two inherent assumptions: prior knowledge of potential C&C channels (e.g. IRC or HTTP) and capability of monitoring them. However, when botnets switch to more advanced ones, those assumptions no longer hold, e.g. no knowledge about the potential C&C channel, thereby no place to monitor, making the current detection schemes unable to detect the more
advanced botnets.

By more advanced botnets in this Chapter, we refer to the ones not only with a P2P (peer-to-peer) structure, but also capable of using different ports for C&C. Utilizing multiple ports in C&C for P2P botnets is easy to implement and possibly widely used in current botnets. For example, as noted in [129], once the Nugache bot was altered to use random high-numbered service ports for each bot, the malware dropped off nearly everyone’s radar.

Therefore, to make effective countermeasures against botnets, it is very important to design detection schemes for P2P botnets (for concise representations, the term P2P botnets and more advanced botnets are exchangeable in the following of this Chapter, unless otherwise specified). Consequently, in this Chapter, our focus is to design detection schemes without previous assumptions used in the current centralized botnet detection schemes and make use of node behavior profiling with statistical testing.

Since the two assumptions used for centralized botnet detection no longer hold in P2P case, to detect the P2P based botnets, protocol and structure independent schemes have to be considered [27]. Usually, the design of such schemes requires continuously processing the normal traffic before detection. For example, Botminer [27] uses three filters to remove normal traffic at the first stage. However, as the C&C traffic is usually of short lifetime and intermittent in the network, there is a large amount of time that the network traffic does not include any C&C traffic. Consequently, processing such network traffic causes unnecessary computation overhead and slows down the response time. Hence, the design of fast detection schemes is also very important but little work has been done in the literature.

4.1.1 Overview of the Proposed Approach

In general, from the studies of P2P botnets in [65] [38] [68] [42], the advantages (e.g. robustness) of P2P botnets are mainly from the structure rather than the specific P2P protocol. Consequently, many P2P botnets [65] [38] [68] [42] proposed in the literature can randomly choose from either P2P protocols and/or non-P2P protocols for C&C. In addition, utilizing multiple non-P2P ports in C&C for P2P botnets is acknowledged in the literature and possibly
Figure 4.1 Examples of node behavior based detection

widely used in current botnets, such as the variation of Nugachebot [129], which uses random high-numbered service ports for each bot. Therefore, in this Chapter, given the trace data we have, we only consider the latter one and regard the detection of P2P botnets using P2P protocols as our further work.

Given the unique characteristics (small volume, short lifetime from a group of bots and mixed with normal traffic) of C&C behavior, and the correlation of node behavior profiles, we are able to design anomaly detection schemes based on formal statistical tests to determine if there are notable underlying botnet C&C behaviors in the network. The proposed approaches are primarily based on node behavior profiles proposed in Chapter 2, which characterizes the node behavior by jointly considering spatial and temporal correlations. As illustrated in Fig. 4.1, the main idea of our approaches is to find node behavior characteristics that are statistically stable in the training data, and detect the abnormal behaviors by measuring the statistical changes on the existing profiles, especially popular ones.

Further, we propose an approach to fast detect C&C channel of P2P botnets by using SPRT (Sequential Probability Ratio Test) in an optimal way on the node behavior profiles in Chapter 2. To be specific, by simplifying each node behavior into a Bernoulli variable, we
make use of SPRT for fast detection. As we do not have full knowledge of $H_1$, we consider the worst case attack model, formulate the C&C channel detection as an optimization problem and derive the optimal values for SPRT mathematically. An SPRT based fast and optimized detection scheme (SPRT-FOD) is further proposed to make use of the optimal values.

In the evaluation part, similar to [27] [97], since the enterprise network is less studied than the Internet [10], we consider the enterprise network from [10] in this Chapter. Moreover, in addition to simply combining two independent traffic types (bot traffic and normal traffic) during the evaluation (simple case), we consider the case where the bot traffic is correlated with the normal traffic (realistic case). That is, to avoid being detected, the botmaster can obtain important information (e.g. services ports and destination range) about the network where his bots locate, randomly assign the C&C ports and peers within the same range of this network to each bot. This correlation is usually not considered in the current research, but we believe this case is more realistic and could be adopted by the botmaster easily (e.g. slight change of Nugachebot in [129]). In particular, we make the following contributions in this Chapter:

1. Capture the normal traffic using node behavior profiles, formulate the problem of detecting C&C traffic as detecting additional behaviors over the existing normal behavior profiles.

2. Propose two anomaly detection schemes using statistical tests on popular behavior profiles, under the assumption that certain metrics of popular behaviors will be statistically stable.

3. Propose a fast anomaly detection scheme using SPRT, derive the optimal parameter values for the worst case attack model by formulating the C&C detection problem as an optimization problem. To the best of our knowledge, we are the first to consider the worst case attack model and propose corresponding solution that derives the optimal value of $H_1$ for SPRT in botnet related research.

4. Validate the assumptions made in this Chapter using real traces from enterprise networks.
5. Evaluate the detection schemes in both simple and realistic scenarios.

The following of this chapter is organized as follows: Chapter 4.2 describes the background and related work in the literature on P2P botnets and botnet detection. We introduce the correlation based node behavior profiling approach in brief and propose two detection schemes using statistical tests in Chapter 4.3. The fast and optimized detection is further proposed in Chapter 4.4. In Chapter 4.5, we first validate the assumptions made for detection, and evaluate our schemes by using P2P botnet C&C traffic with the real trace from enterprise network environments, detecting known and advanced P2P botnets (simple and realistic case) is further considered. We discuss the limitations and our future work in Chapter 4.5 and conclude in Chapter 4.6.

4.2 Background & Related Work

4.2.1 Introduction to SPRT

The SPRT has been shown [130] to be optimal under $H_0$ and $H_1$ in the sense of minimizing the average sample size among all the tests (sequential or nonsequential) under the same false positive and false negative rates. The minimum average sample size in SPRT thereby provides means of fast detection when SPRT is used in anomaly NIDS. In the literature, the basic idea of SPRT is to calculate the likelihood ratio on a given sequence of samples (observed as $X_1, ..., X_n$) to determine whether these samples are from $H_0$ or $H_1$. Assume the samples $X_i$ are i.i.d. (independent and identically-distributed), we have:

$$S_n = \ln \prod_i \frac{\Pr(X_i|H_1)}{\Pr(X_i|H_0)} = \sum_i \ln \frac{\Pr(X_i|H_1)}{\Pr(X_i|H_0)}$$

According to [132], [131], to calculate this likelihood $S_i$, on each observed sample $X_i$, $S_i$ is used in a way such that it ($S_i = 0$ for $i = 1$) increases with length $\ln(p_1/p_0)$ when $X_i = 1$, and decreases with length $\ln(1-p_1)/(1-p_0)$ when $X_i = 0$. With user chosen $\alpha$ and $\beta$ (false positive rate and false negative rate), if $S_i$ goes down to reach the lower bound $A = \ln \beta/(1-\alpha)$, we reach the conclusion that there is no evidence to reject the non-hypothesis ($H_0$) and the current network is normal. If $S_i$ hits the upper bound $B = \ln(1-\beta)/\alpha$, there is evidence to
reject the non-hypothesis, so we generate an alert. For the case where $A < S_i < B$, it indicates that more sample data are needed before making the decision. In most practical applications, when SPRT finishes, the actual false positive and false negative rate ($\hat{\alpha}$ and $\hat{\beta}$) are usually close to $\alpha$ and $\beta$ [133].

Consequently, the expected number of sample needed before decision can be derived as following [131]:

$$E[n_0] = \frac{(1 - \alpha) \ln \frac{\beta}{1 - \alpha} + \alpha \ln \frac{1 - \beta}{\alpha}}{p_0 \ln \frac{p_1}{p_0} + (1 - p_0) \ln \frac{1 - p_1}{1 - p_0}}$$ (4.1)

$$E[n_1] = \frac{\beta \ln \frac{\beta}{1 - \alpha} + (1 - \beta) \ln \frac{1 - \beta}{\alpha}}{p_1 \ln \frac{p_1}{p_0} + (1 - p_1) \ln \frac{1 - p_1}{1 - p_0}}$$ (4.2)

However, despite the advantages, one of the most important issues with SPRT is the determination of $p_1$, which requires full knowledge of $H_1$. If $p_1$ is not determined carefully, the resulting average sample size can be considerably larger than the sample size needed by fixed-sample-size tests [134].

4.2.2 Related Work

4.2.2.1 P2P Botnet

The structure of P2P botnets has been widely discussed in the literature. In [31], the authors list the timeline of captured P2P botnets. The authors in [35] did a thorough study on botnet structures and show that random botnets are highly resistant to different removal methods. Several different P2P botnets have also been proposed such as hybridbot [65], randombot [38], lcbot [68] and superbot [42].

To be specific, as illustrated in Fig. 4.2 (each bot has a peerlist of 5), each bot in randombot/hybridbot maintains a peerlist for communication during C&C; superbot divides the botnet into multiple groups, with each group having a supernode (middle node in Fig. 4.2), the C&C among supernodes is like randombot, while the C&C within each group is like centralized botnets; lcbot is a mixture of superbot and randombot, each bot has a peerlist with one peer outside its group and others within its group, it has a better performance in terms of mean coverage after certain removals and low overhead during C&C as shown in [68].
PUSH or PULL based communications [27] can be further used for each kind of P2P botnet. It is also worth pointing out that, in all these P2P botnets, there is no limitation to use any specific protocols for C&C, thus different protocols can be used for C&C, (e.g. there are reports of botnets using gmail and skype for C&C). In practice, it is easy to implement multiple protocols for C&C in P2P botnets, and it is very likely to happen widely nowadays (e.g. Nugache bot using random ports for each bot during C&C [129]).

![P2P botnet structures](image)

**Figure 4.2** P2P botnet structures

### 4.2.2.2 Botnet Detection

In the literature, many detection schemes have been applied with a focus on gaining full understanding of botnets and their threat. Specially, the authors in [50] conducted several basic studies of botnet dynamics and the authors in [107] analyze the bot source code to provide an inside look at the botnets. In [36], a DNS sinkholing technique is proposed and the global diurnal behavior of botnets is studied. Further, in [96], after analyzing the detailed bot code of Torpig, the authors bought two domain names found in the server list of individual Torpig bot and take the botnet under control for weeks to observe the bot behaviors in sensitive information gathering.
Many detection schemes have also been proposed in the literature to detect botnets with a centralized structure. To summarize, those schemes are based on one or more of the following techniques: honeypot/honeynet, DNS inspection, DNSBL inspection, traffic pattern recognition/clustering, temporal or spatial correlation, and many others.

Among all the techniques proposed in the literature, honeypot/honeynet should be the most successful approach to detecting and analyzing a botnet. The basic idea of honeypot is to emulate an exploited computer to be infected by a bot. Therefore, a computer solely used for this purpose is called honeypot and multiple honeypots can be setup to form a honeynet. The honeypot is generally isolated from the network in the sense that only incoming traffic is allowed, preventing any further infection by the honeypot. Virtual machines can be used together with honeypots to emulate the virtual network environment. Several variations of honeypot are also proposed in.

However, despite the wide use and successful application of honeypots, honeypot has several limitations.

1. As pointed out by: honeypots of low-interaction can capture attacks from a limited number of known exploits that they emulate; and honeypots of high-interaction can neither scale well nor implementing all services.

2. Although it is easy for honeypots to capture malwares exploiting vulnerabilities on remote machines, it is hard for them to capture malwares using emails or web-download, and there is no guarantee on the volume of malwares honeypots can capture because they are used in passive mode.

3. Malwares can detect honeypots or virtual machine environment commonly used for honeypots to evade capture. For example, the authors in proposed a honeypot aware approach to detect and remove the honeypot by using some bots as sensors, under the assumption that the security professionals deploying honeypots have liability constraints such that they cannot allow their honeypots to participate in real
(or too many real) attacks [38].

Therefore, due to the above weaknesses, honeypot alone is not sufficient enough to detect botnet of different variations. More research is needed. Consequently, the authors in [52] propose DNSBL lookup counting. Unlike the honeypot approach and the protocol dependent ones, DNSBL lookup counting is based on the observation of there are some basic requirements for a botnet to function properly.

To be specific, for the botnet sending spams, each bot needs to know if the spam it sends will be successfully delivered to the receiver. Since the mail server on the receiver end may check the DNSBL and any email from the machine on the DNSBL will be blocked. The botnet needs to check the DNSBL before sending spam. Based on the observation that the ratio of the querying requests sent from a host to the query requests on that host will be of different values for a botnet and for a normal user, the author in [52] designed a counting scheme on detecting bot. Although it is also easy to be overcome by the bot, (e.g. bot can use proxy for DNSBL query, or generate fake queries) it provides good insight on the detection of the bot. However, it is only focused on finding botnet members generating spams. As pointed out in [97], their approach may be useful in some cases, but not generally valid and can cause many false positives.

In [55], the authors propose Rishi to detect IRC botnets by matching known nickname patterns of IRC bots. Since it is based on signature of bot nicknames in IRC, it is accurate only if a comprehensive signature set is available. However, it is trivial for botmaster to change the nicknames for bots in the botnet to avoid detection. The authors in [111] [111] [56] extract features under specific IRC protocol such as bytes per second and apply machine learning based approaches for botnet detection. However, their approaches are dependent on specific protocol and will be ineffective when other protocol is used in C&C.

There are also some recent advances in botnet detection. To be specific, in [24], by aggregating traffic of similar payload, same external destination, and internal hosts with similar OS platforms, the authors proposed a system called TAMD to detect malware including botnets. The authors in [27] [28] [29] proposed three kinds of detection approaches: BotHunter,
BotMiner and BotSniffer. BotHunter detects the bots by associating IDS events to a user-defined bot infection dialog model, and it is a passive detection system. BotSniffer is designed to detect centralized botnets by using horizontal correlation. The authors in [28] used SPRT on the character distribution (2-gram) to detect C&C channels of centralized botnets, under the assumption that within certain time interval (e.g. 10 min), the bot commands have high similarity in pattern while normal IRC private messages are not.

However, the BotSniffer works only for centralized IRC botnets and the authors do not consider the worst case performance of their scheme. As it is usually possible that botmaster knows all the parameters used for SPRT, it is very easy for the botmaster to evade detection. In this Chapter, on the other hand, as the character distribution can be easily removed by encryption or compression, we focus on node behaviors instead of character distribution. Moreover, we consider the detection of P2P botnets without any assumptions used in centralized botnet detection (used in BotSniffer). Furthermore, we consider the worst case (botmaster knows all the parameters in SPRT) in detecting P2P botnets to find the optimal values for SPRT test.

BotMiner [27] also utilizes a horizontal correlation approach that examines correlation across multiple hosts [27] and is also designed for P2P botnet detection. It did a good job of detecting the C&C channel of P2P botnets under the assumption of frequent communication between peers and capability of detecting attacks launched by bots after C&C.

However, as four histograms in C&C communications are needed during detection, it mainly focuses on detecting P2P botnets requiring frequent communication between peers (e.g. PULL based). As botnets do not require frequent communications among peers in general (except the bot located in the network using NAT), it is uncertain that their scheme can detect general P2P botnets (e.g. the sample needed for histogram may not be large enough for general botnets). Moreover, detection of attacks generated by bots is a necessary condition in [27] to detect the C&C channel. Given many attacks used by botnets (e.g. clickfraud) are much harder to detect, it is uncertain that their scheme can detect C&C channel of P2P botnets launching more sophisticated attacks.
To summarize, most schemes rely on the two assumptions for detection or are based on the detection of worms or other attacks. Unfortunately, they become ineffective when botnets shift to other structures, other propagation methods, use encryption in C&C or launch more sophisticated attacks (e.g. clickfraud). Therefore, more work on detecting C&C of P2P botnets needs to be done. In this Chapter, our work aims to detect C&C channel without the above constraints, and only rely on the statistical nature of the node behavior profiles and formal statistical tests including SPRT.

4.3 P2P Botnet Detection using Statistical Tests and Correlation based Node Behavior Profiling

In this section, we describe our detection algorithms, which perform statistical tests on the node behaviors. We first briefly introduce the attributes used for profiling. Clustering is then used to find the representative behavior profiles. Classical statistical tests and SPRT are further discussed to construct the detection algorithms.

4.3.1 Attribute Selection & Behavior Clustering

We have discussed the node behavior profiling approach in Chapter 2. In this section, we introduce the main idea in brief. According to [66], the attributes are chosen from the observation of the action sequence of normal users. The action sequence differs greatly between the normal user and the botnet. Since the botnet is dynamic: peers in the botnet can be dynamically shut down or removed from the botnet at any time, a bot may send commands to both online and offline peers on certain ports from its peer list. On the other hand, it is very unlikely that a normal user (or a majority of normal users) generates the normal behavior this way. Therefore, as discussed in Chapter 2, this difference provides inherent guidelines in selecting and simplifying the attributes for node profiling.

Consequently, for attributes with large dimensions, e.g. the dimension of destinations may be quite large, we need to simplify the attributes in a way more favorable for normal behaviors while unfavorable for C&C. Firstly, we categorize the destinations as least, less, moderate
and most popular categories, and only consider the corresponding packets generated in each category (represented as $\text{pkt}_{d,i,t}$). Secondly, we consider all the services as attributes for node profiling, as the combination of all the normal services should also be considered normal. In addition, we only consider TCP traffic, as most of C&C traffic uses TCP (it is similar to design behavior profiles under UDP). Within each application, we consider the packets sent for that application. This is because the node behavior can be better characterized by the packets sent rather than the bytes sent, as bytes only represent the accumulated level of packets. Instead, we consider the total amount of traffic (in bytes) generated and received using $tg_i$ and $tr_i$ for node $i$. Thus the node behavior $x_{i,t}$ for node $i$ at time $t$ is $x_{i,t} = \{(\text{pkt}_{i,j,t}, tg_{i,t}, tr_{i,t}, \text{pkt}_{d,i,t}) | 1 \leq i \leq N, 1 \leq j \leq M, 1 \leq t \leq T, 1 \leq d \leq 4\}$ (the above parameters are listed in Table 2.1).

As there is no prior knowledge on how many common behavior clusters are shared by different nodes in the network, agglomerative clustering is used to find possible clusters. Similar to [66], after identifying the active nodes that initiate connections and removing the silent nodes, the classic agglomerative clustering algorithm is further used [66]. It treats each $x_{i,t}$ as a cluster at the beginning, then calculated the pairwise distance for any two points (or $x_{i,t}$s), and combined two points or clusters into one cluster if the distance between them is below a threshold $T_h$. The criterion to compare the distance on combining clusters is the largest distance, and we set $T_h = 0.25$ in the evaluation section. The distance is measured by the extended Jaccard distance which is defined as [11]:

$$d(x_1, x_2) = 1 - \frac{x'_1 x'_2}{|x_1| + |x_2| - x'_1 x'_2}.$$ 

4.3.2 Design of Node Behavior based Detection Algorithms

Of all the behaviors (or clusters) identified in the training data, we are interested in those which are most common (or popular). This is because we can get enough samples from those popular behaviors, and have more accurate estimations on the metrics of interest. Therefore, in terms of detection, if there is any group behavior in the new sample data which is unseen in the training data, as shown in Fig. 4.3, most of time, it will either cause the change of proportion of popular behaviors (as illustrated in case 1, the original behavior $i$ may switch to another behavior $j$, affected by the unseen behavior in the new data); or the change of
intra-cluster distances within each popular behavior (as illustrated in case 2, if the affected behavior still belongs to cluster $i$).

Generally, both above cases will happen simultaneously in one or more behavior clusters, so we can design algorithms based on different statistical tests for each case. Consequently, the assumption made in our detection approaches is that the metrics for popular behaviors should be statistically stable across training data and new data (we validate this assumption in next section).

![Diagram showing node behavior change with/without C&C traffic](image)

Figure 4.3 Node behavior change with/without C&C traffic

### 4.3.2.1 Behavior Proportion based Test (BPT)

For the $l$ popular behavior clusters identified (in general, we consider the first $l$ popular behaviors, e.g. $l = 3$), consider the $ith$ popular behavior ($i \in l$) a specified property in the new data, statistical test on population proportion of specified property can be applied in testing if the proportion of the $ith$ behavior in the new sample is from the same population as the training data.

To be specific, given the training data sufficiently large, the estimate of proportion of the popular behavior $i$ can be considered as an accurate estimate for the corresponding population. For any new sample captured, if it is from the same population as the training data, under a predetermined significance level $\alpha$, the test will give no evidence of rejecting the non-hypothesis. On the other hand, if they are from different populations, the test will reject the non-hypothesis.
In detail, the new sample size $n$ and the number of occurrences of popular behavior $i$ (denoted by $Num_i$, $Num_i$ is then a random variable) may vary from time to time. At time interval $t$, two possible cases have to be considered, one is called the large sample case, with both $np_{i,0} \geq 10$ and $n(1-p_{i,0}) \geq 10$, where $p_{i,0}$ is the true value of the proportion that popular behavior $i$ in the population. The other is called the small sample case, where either $np_{i,0} < 10$ or $n(1-p_{i,0}) < 10$, or both. Two kinds of tests are needed for these two cases. From [26], for the large sample test, both $Num_i$ and the estimator $\hat{p}_{i,\text{new}} = \frac{Num_i}{n}$ are approximately normally distributed. Therefore, for the null hypothesis $H_0 : \hat{p}_{i,\text{new}} = p_{i,0}$ (with $H_a : p_{i,\text{new}} \neq p_{i,0}$), the test statistic

$$z = \frac{\hat{p}_{i,\text{new}} - p_{i,0}}{\sqrt{p_{i,0}(1-p_{i,0})/n}} \quad (4.3)$$

The above null hypothesis will be rejected if $z \geq z_{\alpha/2}$ or $z \leq -z_{\alpha/2}$, otherwise, there is no evidence to reject the null hypothesis at significance level $\alpha$.

For the small sample case [26], the small-sample test can be used, and it is based on binomial distribution. Usually in this test, a rejection region or acceptance region is used at given significance level $\alpha$. We use the acceptance region in this Chapter, it is given by $[a, b]$, where

$$\alpha/2 \leq \text{Bino}(a; n, p_{i,0}) \quad (4.4)$$

$$\text{Bino}(b-1; n, p_{i,0}) \leq 1 - \alpha/2 \quad (4.5)$$

If $Num_i$ in the new sample does not belong to $[a, b]$, we reject the null hypothesis at significance level $\alpha$.

### 4.3.2.2 Behavior Mean Distance based Test (BMDT)

Another metric of interest is the mean of intra-cluster distance of each popular behavior cluster. If there is a group of subtle behavior added to the network, the mean value of intra-cluster distance may also change. Given the training data is large enough, we can accurately estimate the mean value for each behavior cluster in the population. On the other hand, since
the number of each popular behavior in any new sample is usually not large enough (< 30, as the time interval under consideration is short), to test if the mean values are statistically equal, one-sample t-test can be used [2]. So we have the null hypothesis $H_0 : d_{i,new} = d_{i,0}$ (with $H_a : d_{i,new} > d_{i,0}$), where $d_{i,new}$ is the mean intra-cluster distance of behavior cluster $i$ in the new sample, and $d_{i,0}$ is the true value of the mean. The $t$ statistic is given by [2] [26]:

$$t = \frac{d_{i,new} - d_{i,0}}{s_i/\sqrt{n_i}}$$

(4.6)

where $n_i$ is the size of behavior $i$ in the new sample, $s_i = \sqrt{\frac{\sum(y_j - d_{i,new})^2}{n_i - 1}}$, and $y_j$ the intra-cluster distance of member $j$ in behavior cluster $i$. The degree of freedom of the t-test is $df = n_i - 1$. We reject the null hypothesis if $t \geq t_{\alpha,n_i-1}$, as we allow that $d_{i,new} < d_{i,0}$, which refers to the case the behavior is closer to the centroid of behavior cluster $i$.

### 4.3.2.3 Node Behavior based Detection Algorithm

With two kinds of formal statistical tests, it is possible to design several detecting algorithms. In this Chapter, we consider two of them. One is called independent test detection algorithm (ITD), which treats two tests (BMDT or BPT) independently, and generates an alarm when there are two or more rejections from any test in the $l$ popular behaviors in the new sample data. The ITD algorithm is straightforward and simple.

The next algorithm is called correlated test detection algorithm (CTD). It uses two parameters ($r, l$) and examines the first $l$ popular behaviors. For each behavior cluster, if there is at least 1 rejection of tests by BPT or BMDT, the corresponding behavior cluster is said to be test positive. If at least $r$ out of $l$ clusters in the new sample are test positive, an alarm is generated, otherwise, there is no alarm.

The reason of considering $r$ behaviors together for detection in CTD is that, in practice it is impossible to set the $\alpha$ to be 0. That is to say, there are always a small number of false alarms. On the other hand, considering the subtle group behavior will affect more than one popular behaviors of the testing data, if there is no unseen group behavior, it would rarely happen that two or more popular behaviors do not pass the statistical tests at the same time.
Step 1: Training Data Processing
Identify first \( l \) popular behavior clusters and metrics of interest, e.g. proportion value, and mean intra-cluster distance from the training data.

Step 2: New Sample Data Processing
Remove the known attacks from the new sample, and then group each behavior in the new sample data to the closest behavior cluster by measuring the distance. Identify the first \( l \) popular behaviors in the new sample data. Calculate the metrics of interest.

Step 3: Behavior Proportion based Test
For each popular behavior \( i \) in the sample data, if it satisfies the large sample case, use equation (4.3) for the test, otherwise, use equation (4.4) and (4.5), at a given significance level \( \alpha \).

Step 4: Behavior Mean Distance based Test
Apply t-test using equation (4.6) for each popular behavior in the sample data, at the significance level \( \alpha \).

Step 5: Detection
For CTD: For each popular behavior \( i \) in the new sample data, if either test (or both) is rejected, call popular behavior \( i \) test positive; if \( r \) out of \( l \) such popular behaviors are test positive, generate an alarm, go to step 6. Otherwise, go to step 2.
For ITD: If there are two or more rejections in any test from \( l \) behaviors, generate an alarm, go to step 6. Otherwise, go to step 2.

Step 6: Identification of C&C Nodes
For each behavior \( Y_i \) in the new data, find the closest behavior \( X_j \) in the training data, update \( Y_i \) by \( Y_i = |Y_i - X_j| \). Use 2-means clustering to differentiate the C&C behaviors from normal behaviors, as normal behaviors will have \( Y_i = |Y_i - X_j| \) around zero vector, while C&C behaviors will be centered at a vector point other than zero (this step is only designed for the simple case discussed in next section). Go to step 2 for another new sample data.

Figure 4.4 CTD & ITD
interval (it also means that the false positive rates of CTD will be less than \( \alpha \)). In addition, the main difference between two schemes is that CTD takes into account the proportion and intra-distance effects jointly. Consequently, we formulate the above detection algorithms in Fig. 3.3.

### 4.4 Fast and Optimal Detection using SPRT

In this section, we propose our fast and optimized detection algorithm, which performs SPRT on node behaviors. We first propose a scheme to simplify node behaviors using Bernoulli variables which can be further used by SPRT for fast detection. Further, by formulating the detection problem as an optimization problem, we derive the optimal parameters for SPRT under the worst case attack model, which can be used for optimized detection.

#### 4.4.1 Fast and Optimized SPRT based Detection

##### 4.4.1.1 Design of SPRT based Detection

For each behavior profile/cluster identified, we are interested in the intra-cluster distance of each node (the distance from the node to its corresponding cluster centroid). Given the training data large enough, intuitively, the empirical distribution of all these intra-cluster distances can be considered a specific feature of the corresponding network. For any new sample data, if it is from the same network, the distribution of intra-cluster distances should match with the empirical distribution of the training data in a statistical manner. Statistical tests (e.g. Kolmogorov-Smirnov Test) can be further used to compare these two distributions.

However, since the behavior itself may be affected by many unknown/unpredictable factors, the node behaviors can be very dynamic in the training data and new sample data (that is, many outliers) under the corresponding network. Therefore, rather than the distribution itself, the median (or values around median) of the empirical distribution should be more stable than the distribution or other parameters. Moreover, the stability of median in a distribution is widely acknowledged and applied in many other areas, such as study of housing price change and personal income change [3]. In addition, detection using KS test is also less efficient since
it requires measuring all the nodes in the new sample before decision. In contrast, the SPRT is the most optimal among all the tests under the same criterion (requires less sample size on average to make the same decision) [15], so we first construct the node behavior in a way suitable for SPRT.

Given the empirical distribution from the training data, we divide the distribution into two areas by a value $p_0$ around median ($0.4 < p_0 < 0.6$), set the nodes in the lower $1 - p_0$ percentile with value 0 and those in the upper $p_0$ percentile with value 1. In this way, we bernoullize the empirical distribution using a Bernoulli variable. Since each node is independent from others and can generate behavior randomly chosen from the behavior clusters, we further associate each node with a Bernoulli variable $B(p_0)$ (our results in the evaluation part show this Bernoullization works very well in terms of low false positive rates).

Consequently, with the empirical distribution of the training data, we obtain the corresponding distance threshold (the smallest distance in the empirical distribution to be 1) under $H_0$, and use it to decide the Bernoulli value of the node behavior in the new sample data. If there is any subtle C&C behavior in the new sample, the corresponding probability of being 1 in the new sample data will be larger than $p_0$, by choosing this value to be $p_1 > p_0$, SPRT can be used for fast detection (in terms of minimum sample size used when compared with other statistical tests).

### 4.4.1.2 Optimization of SPRT based Detection

In general, $p_1$ can take any value larger than $p_0$ in the SPRT detection, and the smaller the difference between $p_1$ and $p_0$, the more sensitive/precise of SPRT. However, the determination of $p_1$ should not be arbitrary especially in the worst case where botmaster knows the parameter values used for SPRT.

That is to say, the assumption that the botmaster does not know any parameters used for detection does not always hold in practice. Therefore, in the worst case, it is possible that the botmaster knows all these parameters (e.g. by social engineering) and can further design strategies to maximize the number of bots he can use during C&C while avoiding detection
at the same time. Consequently, the value $p_1$ should be optimized in a way to minimize the distance between $p_1$ and $p_0$ and the number of bots that the botmaster can use without being detected simultaneously. To achieve the above objective, we consider the following metrics for SPRT.

First, we define the precision loss $l_p$, the difference between $p_1$ and $p_0$, denoted as $l_p = p_1 - p_0$ (the $p_1$ value is not fixed here and in the following part of this section). As we can find from Eq. 4.1, given $p_1 > p_0$, with all other parameter values known except $p_1$ and $E[n_0]$ (we denote $E[n_0]$ as $n$ in the following for simplicity), $n$ is only determined by $p_1$ and vice versa. From the viewpoint of the detector, we want to measure the change in proportion in the new sample data as subtle as possible, this means we want $p_1$ to be small (or as close as possible to $p_0$) under the same detection criterion, so does $l_p$. Since in Eq. 4.1, the numerator is a constant, the decrease of $p_1$ will cause the increase on $n$. In this sense, we want $n$ to be as large as possible, given the size of the new sample data is $N$, we want $n$ to be close to $N$ (if the size of new sample can be mapped to a distribution, e.g. Poisson, $N$ can be chosen to be the mean of the distribution, but it is out of the scope of this research).

![Graph 4.5](image1.png)

**Figure 4.5** Combined loss ($p_0 = 0.5$)

![Graph 4.6](image2.png)

**Figure 4.6** Combined loss ($p_0 = 0.6$)

On the other hand, consider the botmaster knows the choice of $p_1$ and other parameters ($p_0$, $\alpha$, $\beta$ and hence $n$). Since the botnet needs to be reused multiple times, the botmaster has to lower the chance of being detected to a certain level (denoted as $\gamma$, determined by the
botmaster). Then $1 - \gamma$ is the new false negative rate that the botmaster wants to achieve with C&C traffic present and is denoted as $\beta'$ ($\beta' = 1 - \gamma$). Consequently, given any new sample with C&C traffic, he may want the SPRT test to be finished with average size $n'$ equal to $n$ (since the change of $n'$ from $n$ is also a sign of anomaly) without being detected at $\beta'$ level. This further indicates that the proportion ($p'$) in the new sample with C&C traffic should not cause the change of $n$ at $\beta'$ level with $p_0$ and $\alpha$ unchanged. Therefore, from Eq. 4.1 we have:

$$n' = E_0(n') = \frac{(1 - \alpha) \ln \frac{\beta'}{1 - \alpha} + \alpha \ln \frac{1 - \beta'}{\alpha}}{p_0 \ln \frac{p'}{p_0} + (1 - p_0) \ln \frac{1 - p'}{1 - p_0}} = n$$ (4.7)

$$p_0 \ln \frac{p'}{p_0} + (1 - p_0) \ln \frac{1 - p'}{1 - p_0} = \frac{(1 - \alpha) \ln \frac{\beta'}{1 - \alpha} + \alpha \ln \frac{1 - \beta'}{\alpha}}{n}$$ (4.8)

From Eq. 4.7 and 4.8, we know that $p'$ is a function of $p_1$ (hence also a function of $n$ based on Eq. 4.1). Further, from the botmaster viewpoint, it is very likely that all the bots doing C&C are tested in the SPRT (e.g. one way to achieve this is to inspect the infection history of each node, and use those having longer infection histories first in the SPRT) and have value 1. Therefore, the maximum percentage of nodes in the sample data that can be used by the botmaster without being detected is given by: $p' \ast n/N$ ($N$ is the size of the new sample data). Denote this percentage as the uncontrollable loss $l_u$ ($l_u = p' \ast n/N$). We want this value to be as small as possible to limit the maximum number of nodes used by the botmaster without being detected.

To summarize, from the perspective of the SPRT detector, we want both precision and uncontrollable losses to be as small as possible. Without loss of generality, define the cost function $c = l_u + l_p$, then we want to find an optimized $n$ (hence $p_1$, since they are only dependent on each other, denote $n = f(p_1)$) such that the total cost is minimized. Therefore, we formulate the detection problem as the following optimization problem:

$$\min_{1 < n < N, n = f(p_1)} c(n) = l_u(n) + l_p(n)$$

where

$$l_u(n) = p_1(n) - p_0, l_p(n) = n \ast p'(n)/N$$
For each value of \( n \), the values of \( p_1(n) \) and \( p'(n) \) can be derived from Eq. 4.7 and 4.1. As there is no closed form solution for this problem, a numerical method is further used to find the optimum value of \( c(n) \), with other parameters user chosen and fixed (including \( \alpha, \beta, p_0, \gamma \)). For example, given \( p_0 = 0.5 \) or \( p_0 = 0.6 \), \( \alpha = \beta = 0.01, \gamma = 0.05 \) and \( N = 200 \), the relations between \( c \) and \( n \) are illustrated by Fig. 4.5 and 4.6 respectively. And it is easy to find that when \( n = 40 \) or \( n = 30 \), the cost function reaches its minimum at each \( p_0 \). The corresponding \( p_1 \) can be further determined by numerical method from Eq. 4.1. In practice, given empirical distribution from the training data, user chosen fixed parameter values and the range of the total sample size \( N \) in a network, a table of optimal values for SPRT can be further designed in advance to speed up the optimization process. Therefore, the optimal SPRT detection algorithm can be summarized in Fig. 4.7:

In general, step 2 is not necessary, as any attack traffic including P2P botnet C&C traffic should deviate from the normal traffic and can be detected by SPRT-FOD scheme. However, it is possible that there are attacks unrelated to the P2P botnets in the network, and usually the volume of the C&C traffic is much smaller than these attacks, to detect and identify the C&C traffic more efficiently, it is better to remove the known attacks before SPRT-FOD.

The optimized problem above can be also considered in another way, from the perspective of \( p_0 \) and \( p' \). Although different, the two schemes reach the same value of \( n \) (hence \( (p_1) \)) when the combined cost is minimized. And it is discussed in Appendix 5. In addition, the SPRT-FOD can be also formulated in terms of cost, which is discussed in Appendix 4.

4.4.2 Discussions

In this and previous sections, we first proposed two anomaly detection algorithms based on the behavior profiles identified and formal statistical tests. There are two assumptions made in our detection schemes. One is that we need to assume the proportion metric is statistically stable across normal traffic samples, and this should also hold for different networks. The other is that the mean of intra-cluster distance has to be statistically stable in different networks. Therefore, in the evaluation part of this Chapter, we first validate the two assumptions using
Step 1: Training Data Processing
Identify all the behavior clusters, measure the intra-cluster distances from the training data, and derive the empirical distribution of the intra-cluster distance. Bernoullize the distance using value $p_0$ and record the smallest distance $d_s$ of upper $p_0$ percentile.

Step 2: New Sample Data Processing
Remove the known attacks from the new sample.

Step 3: Random Selection and Bernoullization
Randomly choose a node from the new sample, find its belonging behavior cluster and corresponding intra-cluster distance, bernoullize the distance $Y_i$ using value $d_s$.

Step 4: Apply SPR-T-FOD
Apply the SPRT using the optimal value $p_1$, if $Y_i = 0$, decrease $S_i$ with $ln(1-p_1)/(1-p_0)$, otherwise, increase $S_i$ $ln p_1/p_0$. If $S_i$ goes down to reach the lower bound $A$ or upper bound $B$, go to step 5. Otherwise, go to step 3.

Step 5: Decision
If $A$ is reached, pass the sample, go to step 2. Otherwise, generate an alarm, go to step 6.

Step 6: Identification (optional)
For each behavior vector $Y_i$ in the new data, find the closest behavior $X_j$ in the training data, update $Y_i$ by $Y_i = |Y_i - X_j|$. Use 2-means clustering to differentiate the C&C behaviors from normal behaviors, as normal behaviors will have $Y_i = |Y_i - X_j|$ around zero vector, while C&C behaviors will be centered at a vector point other than zero (this step is only designed for the simple case detection discussed in next section). Go to step 2 for another new sample data.

Figure 4.7  SPRT-FOD
real user traces. The traces are from different networks and different dates and times. Most of the traces considered suggest our assumptions hold in real environments.

Secondly, we proposed an SPRT based fast and optimized anomaly detection algorithm (SPRT-FOD) based on bernoullized behaviors and SPRT tests. We consider the fact that, although behaviors from different nodes can be considered independent, they are of different types. Hence, the node behaviors are not identical in general. However, by considering the intra-cluster distances (or residuals) and removing the outliers’ impact (by Bernoullizing), we can consider the bernoullized intra-cluster distance approximately identical.

Therefore, we can achieve approximate i.i.d. distribution of node behavior, which is a prerequisite for SPRT. And the evaluation results of SPRT-FOD further confirm this approximation. Moreover, the derivation of the optimum solution for SPRT is not only applicable for SPRT-FOD in this Chapter, but also applicable for general SPRT based schemes in network security (e.g. BotSniffer in [28]). In addition, the SPRT-FOD should be able to work with any behavior based approach, and we believe the performance of SPRT-FOD can be further improved by any approach that can better capture the node behaviors.

4.5 Evaluations

We use the LBNL enterprise trace data which is the latest publicly available trace to evaluate our approaches. Six representative subnets are considered, each subnet is combined by two traces captured at different times and dates. We use time intervals of 10 minutes to profile the node behavior. To obtain more accurate results, we validate and evaluate the proposed approaches in a more rigorous way than that in Chapter 2. That is, we pick each time interval as the new data set, and the rest time intervals as the training data. Therefore, for each 2-hour trace, 12 possible tests are done for each subnet in the evaluation. Specifically, we consider the trace data captured in subnet002, subnet003, subnet008, subnet010, subnet021 and subnet026 in this Chapter. Table 4.1 lists the information of the trace under consideration. We do not limit ourselves to the trace of a specific time, the largest gap in time is over 10 hours (20041215-0510, 20041216-1618).
Table 4.1 Data information

<table>
<thead>
<tr>
<th>Trace</th>
<th>Size</th>
<th>Date</th>
<th>Duration</th>
<th>Distinct Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet002</td>
<td>291M</td>
<td>20050106-1626, 20050107-1526</td>
<td>2h</td>
<td>88</td>
</tr>
<tr>
<td>subnet003</td>
<td>1.07G</td>
<td>20041215-1142, 20041216-1418</td>
<td>2h</td>
<td>1005</td>
</tr>
<tr>
<td>subnet008</td>
<td>340M</td>
<td>20041215-0510, 20041216-1618</td>
<td>2h</td>
<td>1086</td>
</tr>
<tr>
<td>subnet010</td>
<td>199M</td>
<td>20041215-1443, 20041216-1719</td>
<td>2h</td>
<td>241</td>
</tr>
<tr>
<td>subnet021</td>
<td>239M</td>
<td>20041215-1012, 20041216-1016</td>
<td>2h</td>
<td>194</td>
</tr>
<tr>
<td>subnet026</td>
<td>285M</td>
<td>20050106-1423, 20050107-1323</td>
<td>2h</td>
<td>174</td>
</tr>
</tbody>
</table>

4.5.1 Assumption Validation of ITD & CTD

As discussed in the previous section, it is important that the assumptions are valid (over a relatively long time, e.g. days or weeks) to ensure the detection approaches work in a real environment. Two assumptions were made for the statistical tests. One is that the proportion of the popular behaviors should be statistically stable across different times, for any single subnet. The other assumption is the mean intra-cluster distance should be statistically stable at different times, for any single subnet.

Table 4.2 Proportion test results (1: reject; 0: not reject)

<table>
<thead>
<tr>
<th>Trace</th>
<th>Cluster</th>
<th>t=1</th>
<th>t=2</th>
<th>t=3</th>
<th>t=4</th>
<th>t=5</th>
<th>t=6</th>
<th>t=7</th>
<th>t=8</th>
<th>t=9</th>
<th>t=10</th>
<th>t=11</th>
<th>t=12</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet002</td>
<td>(1,2,3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet003</td>
<td>(1,2,3)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet008</td>
<td>(1,2,3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet010</td>
<td>(1,2,3)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet021</td>
<td>(1,2,3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet026</td>
<td>(1,2,3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

For the proportion case, from Table 4.2, 18 (tests with value 1) out of 216 times (18/216 = 0.083) that the test wrongly rejects the null hypothesis, which is not far from the significance level at $\alpha = 0.05$. Especially, the first popular behaviors in subnet003, subnet008 and subnet010 are not very stable for the proportion test. But other results do show certain stability of the
proportion metric, so we believe the assumption on the proportion test can be considered valid at least for the LBNL trace.

Similarly, from Table 4.3, 8 out of 216 times (8/216 = 0.037) that the test wrongly rejects the null hypothesis, this also matches the significance level at $\alpha = 0.05$ very well. In addition, the test on mean intra-cluster distance is more stable (as compared with proportion metric) across the traces under consideration. Therefore, we conclude that the assumptions are valid for the analysis in this Chapter.

### 4.5.2 False Positive Rate

#### 4.5.2.1 False Positive Rate of ITD & CTD

Table 4.4 shows the false positive rates when different detection schemes are used. We first set $l = 3$ and $r = 2$ for CTD and $l = 3$ for ITD. We can find that both schemes achieve low overall false positives rates (both less than 0.05). Especially, the CTD based detection reports only one false positive (subnet003 at interval 4) for the data we have analyzed. Further, if we remove the first behavior cluster in both subnet008 and subnet003 in the detection, there is no false alarm from both ITD and CTD schemes and the overall false positive rate drop to 0 for the traces we have analyzed. Therefore, from Table 4.4, we can find that although there are chances that the tests are wrongly rejected, by jointly taking into account the test results from different behavior clusters (especially stable ones), we can achieve a lower false positive rate.

<table>
<thead>
<tr>
<th>Trace</th>
<th>ITD ($l = 3$)</th>
<th>ITD ($l = 2$)</th>
<th>CTD ($(l, r) = (3, 2)$)</th>
<th>CTD ($(l, r) = (2, 2)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet002</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet003</td>
<td>8.3%</td>
<td>0</td>
<td>8.3%</td>
<td>0</td>
</tr>
<tr>
<td>subnet008</td>
<td>8.3%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet021</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subnet026</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Overall</td>
<td>2.78%</td>
<td>0</td>
<td>1.39%</td>
<td>0</td>
</tr>
</tbody>
</table>


4.5.2.2 False Positive Rate of SPRT-FOD

Table 4.5 shows the false positive rates for SPRT-FOD when different networks are considered. We set $\alpha = \beta = 0.01$ for all the networks considered and $p_0$ is chosen from $[0.4 - 0.6]$. For each trace, we also divide it into 12 time intervals, and do the SPRT-FOD 1000 times to get the average false positive rate of that interval. The final false positive rate is averaged over the 12 time intervals. We can find the SPRT-FOD approach achieves low false positives rates (all around 0.02) in most traces which are not far from the predetermined $\alpha = 0.01$ value. And the overall false positive rate is also around 0.02. Therefore, as discussed in [133], the practical false positive rate is usually close to the predetermined $\alpha$ value, we believe the false positive rates are acceptable (at least) for the networks we have analyzed.

<table>
<thead>
<tr>
<th>trace</th>
<th>false positive rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet003</td>
<td>0.012</td>
</tr>
<tr>
<td>subnet008</td>
<td>0.021</td>
</tr>
<tr>
<td>subnet010</td>
<td>0.030</td>
</tr>
<tr>
<td>subnet021</td>
<td>0.024</td>
</tr>
<tr>
<td>subnet026</td>
<td>0.015</td>
</tr>
<tr>
<td>overall</td>
<td>0.0204</td>
</tr>
</tbody>
</table>

4.5.3 Detecting the C&C in P2P Botnets

In this section, we consider two possible scenarios. One is called simple scenario where known botnet (Nugache bot) traffic [129] [135] is used for evaluation. This is achieved by adding the nodes generating Nugachebot traffic to the normal trace for evaluation. This is similar to the test used in the literature [27], where nodes using fixed port to generate bot traffic are independent with nodes generating normal traffic. Further, by noticing that any P2P botnet can randomly choose any port for C&C (especially those used in the training data), and the botmaster can avoid being detected by using those ports for C&C, we also consider this more realistic case in the evaluation. That is to say, in addition to generating the C&C traffic from a randomly chosen port, we enable each bot to randomly choose the port recorded in the training data for C&C, in addition, we let the bots only communicate with peers recorded in
the training data to test our schemes.

4.5.3.1 Detection in Simple Case

According to [135] [129] [136], the captured Nugache bot has a peerlist containing 22 peers and uses TCP port 8 for C&C. To evaluate the performance of our detection schemes in the simple case, we use the last time interval as the testing data, and the earlier time intervals as the training data. For the testing data, in addition to the nodes generating the normal trace traffic, we add 100 Nugachebot nodes generating C&C traffic. To obtain the bot traffic, we run Nugache bot over VMware and capture the traffic on port 8. Since we also want to take into account PUSH based P2P botnets, we only consider one round of C&C (e.g. there is no frequent communication among peers). We use the detection rate for the evaluation; the detection rate is defined as the number of bots identified over the total number of bots in the data. In CTD algorithm, we set $l = 3$ and $r = 2$. Table 4.6 shows the results of detection rate under trace subnet002, subnet010, subnet021 and subnet026. We can find, for all these traces, both of our schemes achieve 100% detection rate.

<table>
<thead>
<tr>
<th>Trace</th>
<th>CTD detection rate</th>
<th>ITD detection rate</th>
<th>SPRT-FOD detection rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet002</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>subnet010</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>subnet021</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>subnet026</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

From Table 4.6, our schemes achieve good performance in terms of detection rates under the simple case. This is not surprising, and the reason behind is that, usually TCP port 8 is not a common port, and connections of 22 IPs on port 8 are also rare. As nodes used for bots are independent of normal nodes in the trace, the behaviors generated by the Nugache bot should deviate from any normal behavior clusters greatly, resulting in sufficient evidences for detection.

However, the results in this simple case only provide limited support for evaluation of our schemes, as it is very easy to design a series of variation of Nugache bot in a way such that each bot can choose any port used by normal users in the same network for C&C. And it is
not necessary that two bots share the same port for C&C (e.g. one can choose TCP port 80, another can choose TCP port 21, etc.).

Moreover, usually the nodes generating bot traffic should not be independent of nodes generating normal traffic (e.g. one node can generate both bot and normal traffic from bot program and normal user respectively). The Nugache bot of this kind is much smarter and stealthier than the original one; it is also more practical in implementation. Therefore, we believe experiments of detecting this more advanced bot should be considered for more thorough and general evaluations.

### 4.5.3.2 Detection in Realistic Case

Since there are no traces available for the more advanced Nugachebot discussed above, we simulate the Nugachebot C&C behavior and add it to the normal nodes to evaluate the performance of our detection approaches.

In detail, we let the node generating normal traffic at the testing time interval will also generate a small amount of P2P botnet C&C traffic. The P2P botnet traffic at each node is generated to a port randomly chosen from the recorded ports in the training data, and the peers are also chosen at random from the destinations recorded in the training data. That is, we consider the bot knows the approximate range of destinations and services of normal traffic in its network, only communicates with peers and uses the port recorded in the training data for C&C. Mixing the bot traffic with normal traffic in each node means the bot traffic can hide itself in the normal traffic. In addition, we reduce the peerlist to be 10. By simulating the environments this way, we simulate the realistic case that is most favoring the P2P botnet.

In this realistic case, what we are interested is the alarm rate, which refers to detecting bot existence and generating an alarm, because this is the most important step without which there is no chance to identify bots. Also in this case, it is much harder to identify the bot, as bot traffic is correlated with the normal traffic. Therefore, we evaluate our schemes in this case using alarm rate and leave the identification of individual bots and detection rate as future work.
For each trace, we pick one time interval as new sample data, simulate P2P botnet behaviors only on the active nodes in that interval, and use the rest intervals as the training data. We run the simulation 1000 times for each interval, therefore, there are all together 12000 instances of the P2P botnet scenarios for a 2-hour long trace. We then calculate the alarm probability for each trace. We also set the P2P botnet behavior occurs in only a portion of the total nodes (randomly chosen) to test the sensitivity of our approach (each test also runs 12000 times). Each bot will connect to 10 peers during C&C. Like the common case, there are no frequent connections between any peers. Among all its peers, we consider there is 50% chance that a peer is online (a peer can be turned off, thus unavailable in the botnet). So on average, each bot contacts ten peers, and finds five online peers available for communication. Also, we consider the most concise communications between peers. That is, each peer sends the command using TCP (or any protocol over TCP), so taking into account the setup phase, there are at least 3 packets sent for one command from the sender.

A. ITD & CTD

![Graph showing average alarm rate of ITD & CTD for subnet003, 008](image)

Figure 4.8  Average alarm rate of ITD & CTD for subnet003, 008
We consider trace subnet003 and subnet008 in this case as an example to evaluate the ITD and CTD schemes. Since the first behavior cluster is not stable in previous sections, we only consider the second and third clusters in this section. That is, we set $l = 2$ for ITD and $l = 2$ for CTD. The alarm rates of CTD and ITD in Fig. 4.8 are very good ($> 95\%$) when the percentage of nodes being bots is over 70%. This indicates both schemes perform well when there are enough samples of bots available in the network. In addition, the small standard deviations shown at the bottom also suggest that both schemes are stable.

However, when the percentage of nodes being bots decreases from 70% to 30%, CTD based scheme decreases much faster than ITD based scheme. This indicates ITD is less affected by the percentage change at this interval. This also shows that ITD based scheme is much more robust than CTD, as CTD treats BPT and BMDT jointly, which is much more conservative. Therefore, during implementation, the ITD and CTD schemes can be further used jointly with certain number of honeypots randomly deployed in the network. That is, if a large portion of honeypots used are infected by the bot, it indicates that a large portion of nodes may also be infected, then either CTD or ITD can be used. Otherwise, ITD is preferred.

On the other hand, when the percentage is below 30%, both schemes decrease sharply, and is mainly because the bot samples are not enough to cause notable statistical changes on behavior clusters. A direct enhancement is to consider additional stable popular behavior clusters during detection. On the other hand, from the perspective of the botmaster, since the botnet will be reused multiple times, to ensure the chance of being detected smaller than a certain value (e.g. less than 10%), the number of bots in the network where our detection scheme (for example, CTD) is applied should be no more than 10% of the total nodes, if the botmaster has a good knowledge of the network where the bots are located.

B. SPRT-FOD

From Fig. 4.9, we can find the SPRT-FOD works quite well ($> 90\%$) when there are over 14 percent of nodes in the corresponding network generating C&C behavior. This indicates that SPRT-FOD can achieve good performance in different networks even when there are only a low percentage of nodes used by the botmaster for C&C in the realistic scenarios. Especially,
the detection rate is over 95% when only 15% of nodes is used as bots for all the networks considered. The small standard deviations shown at the bottom also suggest that the proposed behavior based SPRT-FOD scheme is stable. In contrast, when the percentage is lower than 10%, there is a sharp decrease in detection rate.

Moreover, since we do not have any infection history data on each node in the network, we just randomly select the nodes for SPRT. In practice, the infection history can be further used, so we regard the alarm rates in this section the lower bound of our scheme. On the other hand, we find that, when the node percentage is as low as 5%, although the alarm rate is not good from the security perspective (around 35%), it is also not good for the botmaster, as it is much larger than the predetermined $\gamma = 0.05$ by the botmaster.

In addition, as shown in [51], when the number of online peers a bot can communicate is less than a certain number (e.g. 4), the P2P botnet can not be well connected and is sensitive to removals. This is also confirmed by our study on P2P botnets. Therefore, we believe the peerlist size chosen in this evaluation can be also considered realistic. And the performance of SPRT-FOD can be further improved if the botmaster increases the peerlist size in P2P botnets.

4.5.4 Discussions and Future Work

4.5.4.1 Discussions

As we can find in the evaluation part, in general, the SPRT-FOD scheme outperforms the ITD and CTD schemes. The main reason is because in SPRT-FOD scheme, we make use of all the network information to do the SPRT test. In contrast, in CTD/ITD scheme, only partial network information is used in detection, e.g. up to three most popular behaviors are considered. Therefore, when using more stable behaviors information in the test, it is possible to improve the performance of CTD and ITD.

However, this improvement requires checking more number of nodes during BMDT and BPT. On the other hand, only a small number of nodes need to be checked in SPRT-FOD, which is the direct benefit from SPRT. We can also apply our detection schemes by monitoring a subset of nodes that has a higher probability of being infected, depending on the version of
OS, history of being infected, number of abnormal behaviors in the past, etc.

In addition, as shown in the evaluations on the realistic case, we consider all the destinations bots communicates with being recorded in the training data. In practice, it will rarely happen and at least a portion of destinations used by bots will be unseen in the training data. Therefore, an additional attribute recording the number of unseen destinations during behavior profiling will further improve the performance of ITD/CTD and SPRT-FOD in detection. On the other hand, we regard the evaluation results the lower bound performance of all proposed schemes.

4.5.4.2 Possible Evasion

There are several possible ways for P2P botnets to evade proposed detection approaches. Firstly, the botmaster can evade the detection by using random and long response (hours or days) delay in C&C. This is to reduce the bot sample size available in each detection period such that there is no evident statistical change in the node behaviors. If this happens, possible
solutions include extending the detection time window (e.g. from 10 to 20 minutes or longer) and considering additional attributes such as unseen destination and DNS. As the P2P botnet may connect to the destinations unseen in the training data and use DNS during C&C. We can also use weight for different attributes according to the importance to the detection. Currently, we set all attributes with weight 1.

On the other hand, as noted in [97], if bots are forced to use a very long response delay, the utility of the botnet to botmaster is reduced or limited because the botmaster can no longer command his bots promptly and reliably. For example, the computers infected with bot may be shut down or disconnected from the Internet by the users or security professionals during this time, hence become removed from the botnet.

Furthermore, it is also possible that botnets may try to utilize popular destinations and services that match the ones used in the corresponding subnet for C&C to evade detection, e.g. use HTTP and IRC protocols and corresponding servers for C&C. However, for any subnet, the number popular destinations are very small when compared to the unpopular ones, and the popular destinations usually have higher security protection, which makes it harder for C&C. Moreover, the use of popular services in C&C gives additional information on potential protocols used in C&C. Consequently, the detection problem of this kind is very similar to the ones considered in centralized botnet detection, protocol-specific detection schemes can be further applied, which is out of the scope of this Chapter.

Last but not least, covert channels [118] can be used to by botnets hide their actual C&C communications. It is widely acknowledged and in general, communication randomization, mimicry attacks and covert channel represent limitations for all traffic-based detection approaches [97]. Therefore, to achieve a better performance, detection schemes at different levels (e.g. network-wide, protocol-specific, host-based and content-based) should be used jointly to make evasion more difficult.
4.5.4.3 Future Work

To detect the P2P botnet using multiple protocols for C&C, the most difficult part is to detect its existence. As shown in the evaluation section, the proposed ITD, CTD and SPRT-FOD schemes can achieve good performance in both simple and realistic cases. On the other hand, once the existence of C&C is detected, either attack correlation based techniques \[27\] or active approaches \[30\] can be used to locate the bot. However, to achieve a more accurate detection for ITD and CTD in the realistic case, in addition to considering more popular behavior profiles, time effects (daytime or nighttime, weekday or weekend) should be further considered during behavior profiling and detection, as the behavior profiles will be quite different in different times, e.g. backup traffic is usually generated in the nighttime.

In addition, there are normal behaviors that only occur occasionally, e.g. payroll at the end of each month or final grades at the end of each semester. In this case, a whitelist of normal but uncommon behavior profiles will help to reduce the false positive rates greatly, especially for ITD and CDT. Further, more trace data from different enterprise networks are also required for a thorough evaluation. To implementation of our schemes, TCP traffic can be divided into P2P and non-P2P traffic, detection schemes of each traffic part can be further implemented. We will consider the above in our future work.

On the other hand, as noted in the previous section, the node behavior simplification approach and the SPRT-FOD/ITD/CTD scheme in this Chapter should be able to work with any other node behavior profiling schemes. And the performance of SPRT-FOD could be further improved by any other better behavior profiling schemes in the literature. Possible enhancements include considering additional representative attributes, using weight for different attributes according to their importance for the detection, classifying the destinations into inner and outer parts.

One main concern of our scheme is the unbounded sample size needed under certain situations for SPRT \[130\] \[132\]. In this case, truncated SPRT can be further used if the sample size required is too large before decision. Also, since each round of SPRT requires making a decision (stop or continue), the performance can be further improved by the group based
SPRT method [132].

It is also worth mentioning that for the t-test used in detection, in addition to independency, it requires that the sample has equal variance and residuals are normally distributed. In our analysis of the data, we found that a small number of samples do not have equal variance and their residuals are not normally distributed. To solve these issues, transformations (e.g. Box-Cox) can be used before the t-test to achieve equal variance, nonparametric test (e.g. One Sample Wilcoxon Sign Rank Test) can be used to solve the normality issue. As the t-test provides correct results for the data considered in this Chapter, we regard it one of our future work.

4.6 Conclusion

In this Chapter, we discussed the problem of detecting the P2P botnet via its C&C. Based on the observation of correlations of node behaviors at different times, we used a correlation based node behavior profiling scheme which can be further used for statistical tests. By validating the assumptions, we designed algorithms based on statistical tests to check if there is any unseen subtle group activity from C&C in P2P botnets. We also propose a fast and optimized detection scheme based on SPRT under the worst case attack model. The results of the experiments on the real user traces from enterprise networks are encouraging in terms of high detection and low false positive rate.
CHAPTER 5. A Novel Background Traffic Modeling Structure for NIDS Evaluations

Realistic background traffic modeling is one of the most important problems in applications such as NIDS (network intrusion detection system) evaluation. However, although traffic is driven by users (or nodes) in a network, modeling it by individual node does not scale well and requires longer traces. Most current researches in the literature model the aggregate network traffic by individual application or application mix in the network. They thereby lose the traffic characteristic at the node level, e.g. they model/generate traffic of the same application from different nodes by the same underlying distribution, and choice of destinations by different nodes are also from the same underlying distribution. Consequently, the traffic can not be considered realistically generated from real nodes. In this Chapter, we propose a novel traffic modeling structure that not only models the traffic at the node level but also has good scalability. This is achieved by designing a behavior based traffic modeling framework. We also show that most traffic models are special cases of this framework. In the initial evaluations, we evaluate the proposed framework under real user traces from enterprise networks and obtain several important observations which could be helpful in generating better realistic background traffic.

5.1 Introduction

Traffic modeling is widely used in many areas of computer networks, such as protocol evaluations, buffer management evaluations and NIDS evaluations. When used for NIDS evaluations in network security, it usually refers to modeling both attack and background traffic. Its main application is to thoroughly evaluate and compare different NIDS (or distributed
NIDS) performances in terms of robustness and capabilities [83]. It is also used to eliminate any weakness [83] in NIDSs before practical implementation. With the extensive growth in both diversity and volume of malwares in the current Internet, there has been an increasing demand on NIDSs. Hence, traffic modeling plays a more and more important role for NIDS evaluations.

In this Chapter, we focus on background traffic modeling to support NIDS evaluations. Unlike common traffic generators, the traffic generator for NIDS (especially anomaly based NIDS) requires realism in a much wider range. For example, other than the packet size and inter arrival time information maintained by traffic generators for protocol evaluations, user level information (in this Chapter, unless otherwise stated, the notions of user and node are exchangeable) has to be considered, including choice of services and destinations, user think time and payload content. This makes the modeling problem more challenging.

In general, in addition to privacy issues, the challenges for realistic background traffic modeling lie in:

1. User behavior heterogeneity: users differ in their preferences on services, destinations and the corresponding amounts of traffic generated. Moreover, these preferences are dynamic over time.

2. Host heterogeneity: hosts have different processing capacities, protocol types, versions and parameters (e.g. TCP window size).

3. Diversity in traffic types and characteristics: for example, packet size, inter-arrival time.

4. Network size: a network may consist of many individual nodes, which makes it difficult and unscalable to model traffic from each individual node.

5. Incomplete sampling: in high speed networks, it is impossible to capture all the traffic information by sampling, which makes modeling more difficult.

To solve the above challenges, most current work ([81] [87] [88] [59] [77] [82] [76] [79]) in the literature models the aggregate network traffic by individual application or application
mix. To summarize, for individual-application based models, they extract the important traffic characteristics of each application from individual node. Since nodes can be considered independent of each other, they pool each traffic characteristic across all the nodes in the network to build traffic models. Similarly, for application-mix based models, they abstract all the application communications as file transfers, and pool the important traffic characteristics in each individual node in the network to model the total application mix.

However, despite successes in modeling the aggregate network traffic, current modeling schemes lose the traffic characteristic at the node level, e.g. they model/generate traffic of the same application from different nodes by the same underlying distribution, and choice of destinations by different nodes are also from the same underlying distribution. Consequently, the traffic can not be considered realistically generated from real nodes. In other words, as will be shown in Section 3, most current models assume the nodes are homogeneous or identical in behaviors. The node behavior considered in this Chapter refers to capturing the preference patterns of using the network from different users. As a consequence, current modeling schemes may have difficulties when there is no assumption on behavior homogeneity. For example, as will be shown in the evaluation section, current models may have difficulties in modeling the case where in a network, certain nodes continue generating HTTP requests with small mean or variation in size, while others (more active ones) keep generating HTTP requests with large mean or variation in size. In addition, they also have problems modeling the realistic traffic from nodes with the same mix of applications (such as HTTP and Email), but towards different destination ranges.

Therefore, we propose a solution to the above issue and extend the current work by relaxing the assumption on node behavior homogeneity. This is achieved by proposing a behavior based traffic modeling framework that captures the network traffic from the viewpoint of node behaviors. In addition, although the trace data used for the experiments in this Chapter can not represent all the traffic in different networks, our main purpose is to use them as an example to help understand and generate better realistic background traffic. Specially, we want to answer the following questions on traffic modeling:
1. From the node behavior perspective, how do node behaviors affect the precision in network traffic generation. We propose a way to relate the node behavior to the precision of traffic generation.

2. From the node behavior perspective, each node behavior may consist of a combination of different applications, and it is very likely certain application exists in two or more node behaviors. Consequently, does the traffic from the same application but different node behaviors share same properties? We show it is not necessary for such traffic to share all the properties from our experiments and statistical tests.

3. From the node behavior perspective, how does the framework relate to the current work in traffic modeling? We show that most traffic models can be considered as special cases of our framework.

In this Chapter, as a majority of the network security related traffic is transmitted via TCP, we focus ourselves on the traffic modeling framework for TCP, and it will be straightforward to extend the framework to support UDP. In our evaluations, as the enterprise network is less studied for over 15 years [10], we evaluate the proposed framework under real user data from enterprise networks and make several important observations in Chapter 5.4 which could be helpful in generating better realistic background traffic.

The following of this Chapter is organized as follows: Chapter 5.2 describes the related work in the recent literature on traffic modeling. In Chapter 5.3, we bring forward the proposed framework by first giving a detailed comparison on assumptions used in the existing work, and then discuss the correlation based node behavior extraction engine. In Chapter 5.4, we evaluate the framework using the real trace data from enterprise networks and answer the above questions by the experiment results and statistical test. As empirical modeling is widely used in current traffic models, we also evaluate the performance of empirical modeling under this framework on selected parameters for web traffic. We conclude the Chapter in Chapter 5.5.
5.2 Related Work

Traffic modeling has been widely used in many areas of computer networks, such as buffer management and QoS provision. One of the classic examples to illustrate how great impact a traffic model can have on the evaluation results is discussed in [94], where performances of RED (Random Early Detection) and FIFO (First-In First-Out) under web traffic models are discussed. It is shown in [94] that RED has little benefit when compared with FIFO in the case where realistic web-like traffic (finite sources and small data transfer) is used in evaluations. In contrast, as shown in [93], if only unrealistic traffic (infinite sources and large data transfer) is used, RED can significantly outperform FIFO. This difference in performance is mainly caused by the different traffic models. Therefore, accurately modeling and generation of the Internet traffic plays a very important role in evaluations of any network mechanism including NIDS.

5.2.1 Real Trace vs Realistic Modeling

The traffic generator for NIDS proposed in the literature can be classified into two groups: real trace (e.g. DARPA data sets) and realistic modeling. A natural approach for addressing representativeness in both flows and content is to take empirical traces from real networks for offline analysis [59], and the DARPA data set, which was developed at Lincoln Lab in 1998-1999, is one of the most well known traces for offline NIDS testing. However, as pointed out by [89], there is no statistical information on the real traffic published; the data rates and their variations are never a variable in the test environment; there is no knowledge on how many false alarms background traffic alone would generate. In addition, the size of the training data may be insufficient. Therefore, although real traces naturally preserves all the contents needed for NIDS testing, they suffers from the issues such as privacy concern and inflexible to generate counterparts in experiments. In this Chapter, we focus ourselves mainly on the realistic modeling in the literature.
5.2.2 Packet Level Modeling

Realistic traffic modeling, in contrast, is able to generate traffic synthetically. In principle, it can produce traces for offline tests or in live streams for online tests [59]. To generate such traffic, the simplest way is to generate packets at packet level according to the packet characteristics (exact copy or preserve some statistical properties) extracted from the monitored network. Generators of this kind e.g. TCPreplay [90], are used to replay packets that were previously captured with the tcpdump program. The packet level generators are not suitable to generate background traffic for NIDS evaluations. This is mainly because there is no transport or upper layer information maintained in these generators; hence the packets generated are unaware of the current network condition. In other words, they are called the open-loop generator [92], as there is no feedback loop preserved between the endpoints and the network. Therefore, they are often used to generate attack traffic [91], such as SYN flood.

![Figure 5.1 Relations between different models](image-url)
5.2.3 Advanced Modeling

More advanced traffic generators on source level that takes feedback into account have been proposed in the literature. They can generate closed-loop traffic that reacts to the network conditions as real endpoints do, which is far more realistic than the packet level generators. The general relationship between the closed-loop traffic generators in the latest literature is illustrated in Fig. 5.1. Among all the generators, Harpoon [81] and a-b-t model [87] [88] have been widely acknowledged. To be specific, as illustrated in Fig. 5.2, Harpoon operates at flow level and takes into account the destination diversity and models the distribution of file size, connection arrival, src IP, dst IP and active sessions for traffic generation in TCP. It also takes into account the traffic in UDP, where constant packet rate, a fixed-interval periodic ping-pong, and an exponentially distributed ping-pong are considered.

![Figure 5.2 Structure of Harpoon [81]](image)

The a-b-t model models the traffic from any application in TCP at connection level by the request size (a), response size (b), and waiting time of the next request (t). According to [75], a sequential a-b-t connection vector has the form $C_i = (e_1, e_2, \ldots, e_n)$ with $n \geq 1$ epoch tuples. An epoch tuple has the form $e_j = (a_j, t_{a_j}, b_j, t_{b_j})$ where $a_j$ is the jth ADU (application data unit) sent from the connection initiator to the connection acceptor; $b_j$ is the jth ADU sent in the opposite direction; $t_{a_j}$ is the duration of the quiet time between the arrival of the last segment of $a_j$ and the departure of the first segment of $b_j$; and $t_{b_j}$ is either the duration of the quiet time between $b_j$ and $a_j + 1$ (for connections with at least epochs), or the quiet time between the last data segment. And as mentioned in [88], it is easy to extend
the \textit{a-b-t} model to support UDP. For example, the HTTP application in Fig. 5.3 is modeled as \(((329,0,403,0.12),(403,0,25821,3.12),(356,0,1198,15.3))\).

There many schemes proposed in the literature on traffic modeling for individual application, we are only interested in the recent and representative ones in this section. In [59], the authors propose a traffic generation framework named Trident, the background traffic generator is modified from Harpoon [81]. In brief, as illustrated in Fig. [59] they use three or more states to represent the evolution of any given application, and generate traffic from the packet pool of each state. Accordingly, an empirical distribution of the packet content is maintained in each state. Other parameters like flowsize, numberpackets, activesessions are considered to regulate the overall traffic volume such that application mix can be tuned as desired. Like Harpoon, destination and source IP distributions are further considered, with the simplification that each flow is transferred between points randomly selected from those distributions. Unlike other models in the literature, the authors in [59] feed the traffic from Trident to the known NIDSs such as Bro, Snort and Bleeding Snort to evaluate its performance in terms of false alarms.

Given characters in the payload of the background traffic are not uniformly distributed, the authors in [58] find out that the payload distribution could affect the signature based NIDSs greatly. Consequently, they propose a scheme to model the payload distribution for signature based NIDS testing. They propose a payload model that is based on empirical distributions of single-byte pattern for different applications. Several important observations are made in
this Chapter, including: 1) NIDS performance is highly sensitive to the underlying traffic and that simple stream-based analysis (e.g., performed with tools like ttcp) as well as trace-based analysis can easily be misleading; 2) they have compared the payloads from single-byte pattern modeling to real traffic traces, showing that the model error is in most cases insignificant. Therefore, it is promising in generating realistic traffic for signature based NIDS testing.

In [83], a Markov chain based approach is used to model the traffic of individual application. With a focus on HTTP traffic from a web server perspective, the authors in [83] treat each web link as a Markov state, and build the state transition and delay matrix from the trace data to model the dynamic of HTTP traffic. In addition, the Markov chain is also used in [74]. The authors proposed a traffic modeling scheme at packet-level, where Inter Packet Time (IPT) and Packet Size (PS) are considered. It models the traffic generated by each individual Internet application (HTTP, SMTP, Online gaming, Instant messaging) running on single hosts.

In [77] [82] [76], the authors propose to use analytical distributions to model the traffic from individual application, e.g. HTTP, TELNET and FTP. Each application is associated with a predefined distribution (e.g. Poisson/Exponential, Lognormal, Gamma, Pareto and Multimodal distributions) and the parameters (e.g. mean and variance) are derived from the trace data. However, as pointed out in [88], the use of analytic models of specific TCP applications doesn’t scale to developing workload models of application mixes comprised of hundreds of applications.

The importance of structural models is well documented [86] [119], but a generic structural
model for Internet applications does not exist to date [79]. In [79], a structure model is proposed which captures the traffic characteristics at user level, connection level and network level. In detail, parameters describing user think time between RRE (Request-Response-Exchange), number of RREs, time between connection starts, RREs per connection, response size, user think time between connection, packet size, packet arrival, network delay and loss rate are considered in the modeling. In the evaluations, it is shown that SWING is able to capture the burstiness in the origin traffic. However, as pointed out in the same Chapter, they do not consider destination distribution and interactions between different applications (hence node heterogeneity), and regard them as future work. Similar work on layered modeling is also proposed in [84] with a focus on the framework.

5.3 Node Behavior based Traffic Modeling

5.3.1 Summarization & Comparison of Current Work

To model/generate realistic background traffic, it is required to consider multiple factors with large dimensions jointly, such as users in the network, applications and destinations. However, it is a quite complex problem. In order to make the problem doable, different levels of assumptions are made in the literature.

The fundamental differences between different modeling schemes in the literature are illustrated in the lower part of Fig. 5.1. Firstly, although nodes in a network should have different characteristics in terms of behavior, taking into account each individual nodes in the modeling requires much more sample traces (for each node) and makes the modeling process less scalable. Consequently, the first assumption made for all current models in the literature is that, all the nodes in the network are homogeneous in behavior. Under this assumption, traffic is modeled either within individual application or application mix.

Secondly, as the number of destinations in a network is also quite large, further assumption on destination homogeneity is made in most traffic models. As we can find in Fig. 5.1, most of work assumes that there is only one (or several) unified destination in the modeling to make further simplification, only Harpoon and Trident model the network traffic without assuming
destination homogeneity.

In contrast, as shown in the upper part of Fig. 5.1, by acknowledging the differences in node behaviors, we consider the traffic as being driven by node behaviors instead of individual node or applications. This is based on the observation that, although nodes are different from each other, there must be common behaviors share by different nodes because of the way people use the Internet. For example, two or more nodes may visit same popular websites, share same popular songs or files and use the same search engine such as google. Further, the same application may exist in two or more behaviors, such as the application \( i \) in Fig. 5.1 or the attribute 1 in behavior \( i, j \) and \( k \) in Fig. 5.5. As a consequence, it is possible to model network traffic by first finding out distinct node behaviors in the network.

![Figure 5.5 Traffic generation from behavior perspective](image)

It is also worth mentioning that, from the node behavior perspective, it can be interpreted that only one behavior type is considered in the current traffic models (application-mix and individual-application based). Fig. 5.5 illustrates one such example where two attributes and four possible behavior types \((i, j, k, l)\) are considered. Since current modeling schemes do not differentiate different behavior types, they treat the behavior \(i, j, k, l\) as one behavior type, denoted by the biggest ellipse in Fig. 5.5. Consequently, they will generate behaviors other than the existing ones \((i, j, k, l)\). That is, current modeling schemes may generate unnecessary traffic combinations all over the dashed square in Fig. 5.5. In contrast, depending on the
precision level required in modeling, it is possible that different behavior types be combined for traffic modeling, such as behavior i and l in Fig. 5.5. Therefore, it is possible to relate previous work on modeling under the behavior based framework, which will be discussed in the following sections.

5.3.2 Framework Overview

Fig. 5.6 illustrates the behavior based framework for traffic modeling using behavior profiles. Taking into account that node behaviors are the real source of all the network traffic, the framework first uses node behavior profiling to identify all the distinct behavior profiles in a network, traffic models are further used to model each individual application or application mix within each behavior profile. This way, we can utilize all the existing models in the framework and future models can also be proposed under this framework.

In Fig. 5.6, the core items are the behavior profile extraction engine, behavior models for each node and traffic models for each profile (if the data is not clean, removal of known attacks is needed before profile extraction engine). We first introduce the function of each part in the framework and then the design of behavior profile engine in detail.

5.3.2.1 Behavior Profile Extract Engine & Behavior Modeling

There are two behavior profile extraction engines in the framework. Both of them are used to extract the representative behavior profiles. Depending of the behavior attributes used, each behavior profile may consist of a combination of applications, destinations or other features such as RTT and TTL. The one on the left side is used to extract behavior profiles from the captured network traces, and the extracted behaviors are further used for traffic and behavior modeling. Many behavior models can be used; one candidate is to use the behavior profile CDF for each node because of its simplicity. To generate emulated traffic, behavior profile id is first generated for each individual node according to its behavior CDF. Traffic models within each behavior profile are further used to generate the traffic.

The behavior profile extraction engine on the right side is used to extract the behavior
Figure 5.6 Behavior based traffic modeling framework
information in real time from the monitored network. Once the behavior profile \( id \) is obtained, traffic can be directly generated from the corresponding traffic models. This is because as long as the traffic models have been built for each behavior profile in the same network, only behavior \( id \) is needed for real time generation. It makes it possible that researchers generate the real time traffic of the remote network locally, as long as they have the traffic models for each behavior profile. Therefore, two types of generation are supported in this framework, real time and emulated traffic.

### 5.3.2.2 Traffic Modeling

For traffic modeling within each behavior profile, the framework is open to any existing or new traffic models. That is, the framework improves the performance of existing traffic models by preprocessing the network traffic into categories, each of which is driven by the same behavior. Especially, if the applications within one behavior profile are considered independent, it is better to use individual-application based model such as \( TG(1,1) \) and \( TG(1,k) \). Otherwise, it is better to use application-mix based model such as \( TG(1,1-k) \). Unlike [81], the application mix within behavior profiles is a subset of the total application mix. In this Chapter, we consider independent applications under each behavior profile. Therefore, in this framework, it can happen that the same application(s) exists in different behavior profiles. For example, application 1 is contained in both profile 1 and \( n \). We hypothesize the corresponding traffic generated by any traffic generator (\( TG(1,1) \) and \( TG(1,n) \) in Fig. 5.6) does not need to share all the properties because the traffic is from different behavior profiles. Statistical test is used to test this hypothesis in Chapter 5.4.

At current stage, we use empirical CDFs (ECDF) to model the traffic characteristics including request size, response size, session starting time, connection inter-arrival, \# of packets within connection, \# of sessions within application. Similar to [81], session is defined as the srcIP and desIP pair of a given application. It is worth mentioning that, although the above parameters do not fully cover all the traffic parameters in traffic modeling, they are among the most representative ones. Moreover, the results obtained on these parameters are enough to
show the heterogeneity in traffic of same application but different behaviors.

5.3.2.3 Privacy Rules & Precision Level

In the framework, the privacy rules are used to remove the sensitive information in the traffic. For example, source IP addresses should be anonymized during the capture of the trace. The precision level refers to the precision in modeling the traffic. By adjusting the precision level of behavior profile extraction engine, we can achieve the flexibility of precision in the modeling. This is achieved through agglomerative clustering (discussed in the next section) used in the behavior profile extraction engine, if the clustering threshold (determined by the administrator) is low, there will be more clusters and hence more precise model of each profile and vice versa. This feature can make the traffic generator suitable for different scenarios of evaluation, e.g. for network traffic modeled for external or internal use, the precision in traffic models could be different, as the more precise of the traffic models, the more likely some important nodes can be identified.

5.3.2.4 Relating to Current Traffic Models

As we can adjust the threshold in the behavior profile extraction engine, it is possible that in the extreme case, the threshold is set to be large enough or the maximum value (1 in this Chapter). Consequently, there will be only one behavior profile in traffic modeling and our framework backs to the system considered in the existing traffic models (individual application or application mix models). Therefore, the current modeling schemes can be considered as special cases of the framework.

5.3.3 Behavior Profiling Extraction Engine

Although there are many possible ways to build the profiling engine, we design it mainly based on scheme proposed in [66]. Other candidate schemes include [8], the main difference between [8] and [66] is that nodes are grouped in [8], while similar behaviors are grouped in [66].
The node behavior profiling approach is discussed in detail in Chapter 2. Therefore, we only cover the main idea in this section. The main idea of the profiling engine is illustrated in Fig. 5.7. As the network applications and destinations are of different popularities, the behaviors from two or more nodes could share similar patterns (or same profile). For example: for hosts $i$, $j$, the behavior at time 2 of node $i$ is similar to the behavior at time 1 of $j$. Consequently, realistic traffic models can be further built for each behavior profile.

Usually, there are a large number of attributes to profile the node behavior and some may have large dimensions. For example, the dimension of destinations may be quite large. Therefore, we need to simplify the attributes in a way more favorable for normal behaviors. Firstly, the destinations are categorized as least, less, moderate and most popular categories (denoted as Q1, Q2, Q3 and Q4 respectively), and the corresponding packets generated in each category are considered. Secondly, all the services are considered as attributes for node profiling, as the combination of all the normal services should also be considered normal. In addition, only TCP traffic is considered. Within each application/destination, the number of packets sent for that application/destination is considered. In addition, the total amount of traffic (in bytes) generated and received using $tg_i$ and $tr_i$ for node $i$ are further considered.
Thus the node behavior \( x_{i,t} \) for node \( i \) at time \( t \) is \( x_{i,t} = \{(pk_{i,j,t}, tg_{i,t}, tr_{i,t}, pk_{d,i,t})|1 \leq i \leq N, 1 \leq j \leq M, 1 \leq t \leq T, 1 \leq d \leq 4\} \) (the above parameters are listed in Table 2.1). To obtain the main characteristics of the node behavior, we remove the inactive \( x_{i,t}s \). For each service attribute, we count the total traffic (in terms of packets) on this attribute and divide it by the total traffic added over all the service attributes. If this ratio is too low and less than a predetermined value (1\% in our Chapter), we discard all the traffic information in that attribute. Thus, we can preserve the main characteristics of node behaviors. Finally, we take the log value of those attribute values. To identify the profiles, given no prior knowledge on how many common behavior clusters are shared by different nodes, agglomerative clustering with extended Jaccard distance (defined as [11]: \( d(x_1, x_2) = 1 - \frac{x_1^t x_2}{||x_1||^2 + ||x_2||^2 - x_1^t x_2} \)) is used to find possible profiles (clusters) [66]. Largest distance is considered in the clustering. The basic idea of agglomerative clustering is to treat each \( x_{i,t} \) as a cluster at the beginning, then calculated the pairwise distance for any two points, and combined two points or clusters into one cluster if the distance between them is below a threshold.

5.4 Evaluations

We use the LBNL enterprise trace data [10] which is the latest and publicly available enterprise trace to evaluate the framework. We consider several subnets to evaluate the framework and achieve similar results. Especially, we mainly include the results of subnet008 and subnet021 (represented by port008 and port021 in [10]) in this Chapter. Two one-hour traces are considered in each subnet (20041216-1618 and 20041215-0510 in subnet008 and 20041216-1016 and 20041215-1012 in subnet021), and there are 1086 nodes in subnet008 and 194 nodes in subnet021. The time interval used for behavior profile extraction engine is 10 minutes, where we assume a majority of connections are finished within this period. Similar assumption is also made in [81].
5.4.1 Behavior Profiles vs Modeling Precision

As node behaviors can be considered the real source to generate the network traffic, the more node behavior profiles are identified, the more precise it is to model the traffic. Fig. 5.8 and 5.9 illustrate the dendrograms after behavior profile extraction. We can find that the node behaviors can be well characterized at different thresholds by the behavior based approach. Especially, when the threshold is set to be 0.4, there are 14 and 13 behavior profiles in each subnet. Moreover, when the threshold is chosen from 0.4 to 0.85, the resulting profiles range from 1 to 14 in subnet008 and 1 to 13 in subnet021. That is, it is possible the precision level be adjusted by tuning the threshold in clustering. For each threshold, corresponding traffic can be modeled in each behavior profile. This also indicates that, the normal traffic (from normal behaviors) is of different types and characteristics; only at certain condition (large threshold) can we consider the normal traffic as one type.

In contrast, as shown in Fig. 5.8 and 5.9, there are only limited numbers of behavior profiles. Therefore, once the threshold is determined, it is possible to consider the node behavior profile as an invariant parameter which is useful in the modeling [86]. In addition, although the number of behavior profiles is larger than 1, it only increases gradually with the decrease of threshold and is much smaller than the number of nodes in the corresponding network.
Therefore, it will not increase the modeling time dramatically when existing models are used in the framework. The different profiling scheme used in [8] also indicates similar results on another data set, which is out of the scope of this research.

Further, as an example for comparison, Table 5.1 illustrates the detailed behavior information of behavior profile 6 and 12 in subnet008, 8 and 12 in subnet021, when threshold is 0.4 for each subnet. We can find that, both these behavior profiles consist of traffic from web applications, but at different volume and interest. For example, the web traffic is the main traffic in behavior profile 8 and 12 from subnet021. However, the web traffic from profile 8 shows great interest in those popular destinations (Q4), while the web traffic from 12 seems to be more interested in those unpopular destinations (Q1). In addition, the volumes of web traffic are also different from profile 8 to 12. However, it is unlikely that the current models capture this difference at the node level.

Table 5.1 Traffic information (log)

<table>
<thead>
<tr>
<th>subnet008</th>
<th>Profile id</th>
<th>HTTP</th>
<th>Imaps</th>
<th>sent</th>
<th>rcvd</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.64</td>
<td>0.18</td>
<td>2.83</td>
<td>3.86</td>
<td>0.22</td>
<td>0.12</td>
<td>0</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2.64</td>
<td>0</td>
<td>3.55</td>
<td>5.90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.64</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>subnet021</th>
<th>Profile id</th>
<th>HTTP</th>
<th>Imaps</th>
<th>sent</th>
<th>rcvd</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3.07</td>
<td>0.19</td>
<td>3.85</td>
<td>6.34</td>
<td>0.1</td>
<td>0</td>
<td>0.77</td>
<td>2.63</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2.46</td>
<td>0.96</td>
<td>1.54</td>
<td>5.68</td>
<td>2.33</td>
<td>0.75</td>
<td>0.36</td>
<td>1.23</td>
<td></td>
</tr>
</tbody>
</table>

5.4.2 Traffic Homogeneity

From Table 5.1, we find that it is possible that the same application is included in two different behavior profiles. Then, the following question is: does the traffic of the same application but different behavior types share the same property? To test this hypothesis, we consider the parameters including request size, response size, session starting time, connection inter-arrival, # of packets within connection, # of sessions within application. For each application in the behavior profile, we extract the empirical distributions of these parameters.

As an example, we consider the web traffic from both subnet008 and subnet021 respectively. During the traffic parameter extraction, although each trace is of two-hour long, by profiling
Figure 5.10  Request size subnet021  Figure 5.11  Response size subnet021

Figure 5.12  Starting time subnet021

Figure 5.13  Request size subnet008

Figure 5.14  Response size subnet008  Figure 5.15  Starting time subnet008
the node of similar behaviors, each behavior profile listed in Table 5.1 consists from 40 to 300 behavior records, indicating about 7 (40 * 10/60 = 6.67) to 50 hours (300 * 10/60 = 50) of data for modeling. As the length (hours) of data is comparable to CSL data (1.67 hours or 100 minutes) used in Trident [59] and the Auckland data considered in [81] which is two days, we believe the data can be considered enough to build the CDFs for the corresponding application and statistical tests.

Fig. 5.10 through 5.12 illustrate the comparison of web traffic from behavior profile 8 and 12 in subnet021 on request size, response size, session starting time. Fig. 5.16 through 5.18 illustrate the comparison of web traffic from behavior profile 6 and 12 in subnet008 on the same parameters. We can find from Table 5.1 that, the behavior profile 8 and 12 in subnet021 represent similar usage in HTTP, they both generate large amount of HTTP traffic. As noted in the previous discussion, they are different in destination ranges. Moreover, the shape of the empirical CDFs looks similar but different in positions. To be specific, in Fig. 5.10, about 80% (from 10% to 90% in CDF) of requests in behavior 8 have size ranging from 2.25 to 2.75, while this range is from 2.5 to 3 in behavior 12, indicating big difference of web traffic request. Similarly, in Fig. 5.11, about 80% of responses have size ranging from 2.5 to 4.1 and from 2.5 to 6 in behavior 8 and behavior 12 respectively.

In contrast, profile 6 and 12 in subnet008 represent two different usages of HTTP: frequent and occasional. Accordingly, the empirical CDFs differ in both shape and position. To be specific, as shown in Fig. 5.13, although 90% of request share the same range of size, behavior 12 has about 10% request of smaller size that is not covered by behavior 6. Similarly, in Fig. 5.14, the response size in behavior 12 covers a much wider range in the CDF than that of behavior 6. This also indicates that the traffic under different behaviors may have different characteristics.

To have a thorough comparison, we further apply the two-sample Kolmogorov-Smirnov Test (KS-test) [115] to compare in a statistical way if the two CDFs of each feature are from the same distribution. The two-sample KS-test is one of the most useful and general nonparametric methods for comparing two samples, as it is sensitive to differences in both location and shape.
Figure 5.16  Inter arrival subnet021

Figure 5.17  # of packets subnet021

Figure 5.18  # of sessions subnet021

Figure 5.19  # of packets subnet008
of the empirical cumulative distribution functions of the two samples [115]. The basic idea of the two-sample KS-test is that, given any two random samples X1 and X2 with size \(n_1\) and \(n_2\), for each value \(x\), the KS-test compares the proportion of X1 values less than \(x\) with proportion of X2 values less than \(x\). This is given by [115]:

\[
D_{n_1,n_2} = \max_x |F_{n_1}(x) - G_{n_2}(x)|
\]

and the null hypothesis is rejected at level \(\alpha\) if

\[
\sqrt{\frac{n_1 n_2}{n_1 + n_2}} D_{n_1,n_2} > K_\alpha
\]

The results of KS-test are shown in Table 5.2, where 1 refers to the two CDFs are not from the same distribution and 0 refers to they are from the same distribution. As we can find, most of the tests indicate the two CDFs under consideration are not from the same distribution. As a consequence, in our evaluations, although the above parameters does not fully covered the traffic characteristics, the results obtained on these parameters are enough to show the difference in traffic from same application but different behaviors and it should be better to model such traffic separately to generate realistic traffic. On the other hand, it is worth mentioning that, in some special cases such as traffic from automatic services (e.g. automatically check emails very 60 seconds), the traffic could be same in most parameters.

<table>
<thead>
<tr>
<th>trace</th>
<th>pkt #</th>
<th>request</th>
<th>response</th>
<th>session #</th>
<th>inter-arrival</th>
<th>start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnet008</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>subnet021</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

5.4.3 Realistic Traffic Generation

We also partially evaluate the performance of traffic modeling under the framework. We simulate the traffic according to the empirical distributions of the above parameters. As an example, the LBNL trace considered are the web traffic from the behavior 12. Fig. 5.21 to 5.26 illustrate the empirical and simulated distributions of request size, response size, session starting time, connection inter-arrival, # of packets within each connection and # of sessions
within each node, we can see the simulated the traffic matches the origin trace quite well. The KS-test also confirms that the distributions of each figure are the same.

5.4.4 Discussions and Future Work

In general, network traffic is mainly generated by real people using the corresponding network. However, most major current traffic models only consider application as the real source of the traffic, which is not real in general unless nodes are assumed to be homogeneous. In this Chapter, we use the node behavior to capture the difference between real users and build a new framework for traffic modeling without the assumption of node homogeneity. As the framework captures the real source of the traffic, the traffic generated under it can be considered more realistic.

Moreover, based on the evaluations using real user data, we obtain several important observations that may help the understanding and generation of better realistic background traffic, including:

1. Even if nodes are different from each other, there are in general similar behaviors shared among nodes due to the usage of the Internet, and traffic can be modeled from node behavior profiles at the node level;

2. Normal traffic is of different types in general, from the behavior perspective. The precision of traffic modeling can be adjusted by the clustering threshold during behavior extraction;

3. It is not necessary that traffic from the same application but different behavior profiles shares all the properties.

In addition, the framework has the following advantages:

1. Full coverage and compatibility: it covers and is compatible with most traffic models in the literature. In other words, when used under our framework, there is no requirement to modify the current traffic modeling schemes, and the framework can further help them to generate better realistic traffic.
2. Behavior oriented: the framework captures different combinations of node behaviors in traffic generation. Therefore, it is easy to generate the traffic from any combination of nodes in the network, which is important for evaluations in distributed environments.

3. Scalable, adjustable and repeatable: the framework is able to be applied to different environments and model traffic under different precision requirements. In addition, as shown in the experiments, no matter what the threshold value is used in behavior extraction, the number of behaviors identified is generally much fewer than the number of nodes in the network, which makes the framework scalable. Further, by increasing or decreasing the number of given behavior profile(s) in the network, we can also scale up and down the traffic generation under different realistic scenarios.

However, although the LBNL trace is the latest enterprise trace publically available, it was captured during 2004 and 2005 and might be different from the current enterprise or Internet traffic. Therefore, more traces including different network scenarios are needed for further evaluations on this framework and traffic models within it. Additional parameters like packet inter-arrival time can be further considered.

Furthermore, traffic modeling is not an end to itself. To achieve a complete evaluation, the performance of the modeling schemes under our framework should be further evaluated by the current well-known NIDSs. More metrics should be considered for the evaluation, including false positive rates generated from different current NIDSs on the emulated background traffic.

This is our ongoing work on the top layer in the NetBottle Project [137] [138], where our ultimate goal is to emulate real-world network traffic in a variety of testbed environments. As illustrated in Fig. 5.20, the NetBottle consists of three layers: data collection, model construction and emulation, with the lower layers providing the data to the layer above. Currently, we plan to use Xen [113] or Xen World [114] to simulate the case of large scale deployment.

5.5 Conclusion

In this Chapter, we discussed the problem of background traffic modeling for NIDS evaluations. Based on the observation of correlations of node behaviors at different times; we propose
Figure 5.20 NetBottle project

a novel framework for traffic modeling. The framework makes no assumption on node behavior homogeneity and captures the network traffic from the node behaviors instead of individual application or application mix. It is shown to be compatible with the major traffic models in the literature. We also evaluate the framework in our initial evaluations and obtain several important observations which may be helpful for further studies in this area.
Figure 5.21  Request size subnet021  Figure 5.22  Response size subnet021

Figure 5.23  Starting time subnet021  Figure 5.24  Inter arrival subnet021

Figure 5.25  # of packets subnet021  Figure 5.26  # of sessions subnet021
CHAPTER 6. SUMMARY AND DISCUSSION

6.1 Conclusion

In this dissertation, we study and design efficient and effective techniques to provide security guarantee for enterprise networks. Two kinds of fundamental areas are considered: malware detection and realistic background traffic generation. We hope that the proposed schemes can serve as fundamental components for network security research, and the key observations made in this research can provide further insight to the difficult problems in network security. Our contributions in this research are summarized in the following:

1. A new node behavior profiling approach which captures time and node correlations to study the behavior of normal nodes. We show that it is easy to understand the node behavior using the profiles and the normal behaviors are generally of different types. We also show in our evaluations that the behavior profiles are of different popularity and part of them can be considered statistically stable from the network perspective. In addition, we give an example on worm detection to demonstrate that the profiles are capable of being used for further analysis and detection when coupled with different statistical testing methods;

2. A framework for botnets, which is able to capture botnet structures of all known kinds and predict new botnets;

3. Based on the proposed framework, we predict a new botnet that we call the Loosely Coupled P2P Botnet (lcbot). It is stealthy and can be considered as an extension of existing P2P botnet structures. We also design several new metrics and conduct experiments to
compare the performances between lcbot and other P2P botnets in the literature to gain insight understanding of P2P botnets;

4. Two behavior based anomaly detection approaches on P2P botnets by coupling the statistical tests with the node behavior profiles identified. In brief, the key difference between our work and previous work in botnet detection is that, instead of trying to filter out the botnet behaviors in a network, we detect them by measuring their impacts on one or more normal behaviors in a statistical way;

5. A fast and optimized anomaly detection approach on P2P botnets. We make use of SPRT (Sequential Probability Ratio Test) to measure the impacts of the C&C behavior from P2P botnets ($H_1$) on normal behaviors ($H_0$) in a fast manner. We propose an approach that simplifies the behavior of each node in a stable way suitable for fast detection via SPRT. Then, under the worst case attack model, we formulate the C&C detection problem as an optimization problem and derive the optimal values for $H_1$ mathematically. An SPRT based fast and optimized detection approach is further evaluated under different traces;

6. A new traffic modeling structure for NIDS evaluations and several important observations that are helpful to improve the realistic level of current modeling schemes. By taking into account that the node behavior is the real source of the network traffic, we take a step further to propose a novel traffic modeling structure that captures the traffic characteristics by jointly considering the node heterogeneity with other known features. We also apply empirical CDFs to model the common traffic features under this structure. In our initial evaluations, the empirical CDF and the modeling structure work well under the real user data from enterprise networks. We also obtain several important new observations that can be further used to improve the current traffic models.

7. Validation of the assumptions made in this research. Simulations and evaluations of the above approaches are further conducted under real user data from enterprise networks.
To the best of our knowledge, we are the first to propose the botnet framework, consider the worst case attack model and propose corresponding fast and optimized solution in botnet related research. We are also the first to propose efficient solutions in traffic modeling without the assumption of node homogeneity.

6.2 Future Work

In this research, we design the cooperative detection techniques that jointly consider host and network characteristics. As shown in the evaluations, the combination of these two complementary parts can provide good detection results and is promising for realistic traffic generation. In our future work, we plan to improve and design additional behavior based schemes and couple them with new network based methods to develop new cooperative techniques. In particular, we summarize our future work including those discussed in each individual Chapter as follows:

6.2.1 Evaluations of Anonymization Schemes

One of our future work is to apply the node behavior profiling approach to new research areas such as anonymization. As shown in our evaluations, the proportion value of the popular behaviors in CTD/ITD and $p_0 = Pr(Y_i = 1|H_0)$ used in SPRT-FOD can be considered as an inherent characteristic for the normal traffic in a network. Therefore, they can be used as metrics for evaluating anonymization schemes on different trace data. In our future work, we plan to evaluate different anonymization schemes under different metrics, including those mentioned above.

6.2.2 Improvement of the Existing Approach

We plan to improve the efficiency and coverage of the proposed P2P botnets detection approach. To achieve this goal, in our future work, we plan to study new metrics that are stable for normal nodes and can be used for anomaly detection to improve the efficiency. Additional trace data including normal P2P traffic and P2P botnet traffic will be further
considered to increase the coverage of the existing approach (e.g., detection of P2P-protocol based botnets and Icbot).

In addition, visualization is also one of the important parts in our future work. As each behavior profile can be represented by the cluster centroid and a deviation, with the use of visualization, it can help the administrators to identify the network anomaly in a fast manner, without going into too much details of the network traffic.

6.2.3 Coupling with Multi-Variable Analysis

In this research, the node behavior consists of multiple variables. Therefore, it is possible to apply different multi-variable analysis to study the normal node behavior and additional statistical tests on multi-variable can be used to design corresponding detection schemes.

6.2.4 Mitigation & Traceback

Once bots/botnets are detected, a followed question is how to find the real person controlling them. This is because detection alone can not stop the use of botnets entirely. For example, even if the botnet is detected and terminated, it is still possible that the cleaned machine being infected and used for constructing botnet a second time. On the other hand, mitigation is also an important question after detection, as we want to minimize the loss by botnets as much as possible if the detected botnet is launching attacks (e.g., DDoS). These issues are also quite important to network security and we plan to investigate and develop effective and efficient techniques in our future work.

6.2.5 High-speed & Large-scale Detection

We plan to develop a new real-time detection system combining non-P2P and P2P protocol based P2P botnet detection and existing botnet detection techniques seamlessly. The system should be characterized in layered design with a good sampling strategy, which make it work in very high-speed (e.g., 1-100G bps) and very large (e.g., ISP level) network environments.
In addition, all the techniques proposed in this research are passive detection. As the botnet evolves to more serious types, we plan to investigate active techniques in botnet detection.

6.2.6 Testbed Construction

One of our ultimate goals is to generate different scenarios of realistic background traffic in the testbed. To achieve it, we plan to use Xen [113] or Xen World [114] to simulate the case of large scale deployment. Xen is developed by the University of Cambridge, and it is an open source virtual machine monitor. It is designed to run over 100 full-featured operating system instances on a single typical computer [114], which is quite suitable for testbed development. Therefore, by incorporating our traffic modeling framework with Xen Worlds, it is possible to simulate traffic from hundreds of nodes by using only a couple of actual machines.
## APPENDIX A. Detailed Information of Each Behavior Profile

### Table A.1 Detailed behavior information of trace 1526(0107) (logscale)

<table>
<thead>
<tr>
<th>id</th>
<th>http</th>
<th>app</th>
<th>shlp</th>
<th>https</th>
<th>netbios</th>
<th>sh</th>
<th>maps</th>
<th>byte</th>
<th>byte</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.1269</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4224</td>
<td>4.3524</td>
<td>4.8986</td>
<td>0.0172</td>
<td>0</td>
<td>3.129</td>
<td>0.2642</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.716</td>
<td>0</td>
<td>5.0958</td>
<td>5.8046</td>
<td>1.1099</td>
<td>1.757</td>
<td>0</td>
<td>2.6045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.0645</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6788</td>
<td>2.003</td>
<td>2.2049</td>
<td>0.3513</td>
<td>0.3609</td>
<td>0.0645</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.237</td>
<td>0</td>
<td>0</td>
<td>0.2839</td>
<td>0.0215</td>
<td>1.1928</td>
<td>4.5819</td>
<td>5.1808</td>
<td>2.3456</td>
<td>2.4502</td>
<td>1.5521</td>
<td>3.2853</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2.5194</td>
<td>0</td>
<td>0</td>
<td>0.5565</td>
<td>0</td>
<td>1.3696</td>
<td>4.6856</td>
<td>5.7147</td>
<td>2.6351</td>
<td>1.9429</td>
<td>1.1843</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0699</td>
<td>0.0711</td>
<td>2.5184</td>
<td>3.1006</td>
<td>0.8681</td>
<td>0.7766</td>
<td>1.0179</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.4910</td>
<td>3.2015</td>
<td>0</td>
<td>1.7243</td>
<td>0</td>
<td>0.9522</td>
<td>4.8989</td>
<td>5.043</td>
<td>1.835</td>
<td>3.208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2.125</td>
<td>0</td>
<td>0</td>
<td>0.0753</td>
<td>2.8</td>
<td>1.437</td>
<td>4.977</td>
<td>5.6514</td>
<td>0.0892</td>
<td>0.7787</td>
<td>2.3922</td>
<td>1.0262</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1083</td>
<td>2.474</td>
<td>3.966</td>
<td>0.301</td>
<td>1.175</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.2679</td>
<td>2.659</td>
<td>3.353</td>
<td>0</td>
<td>1.234</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.1026</td>
<td>0</td>
<td>1.5343</td>
<td>2.8523</td>
<td>0</td>
<td>1.8386</td>
<td>5.3617</td>
<td>5.929</td>
<td>2.943</td>
<td>2.112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.426</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.1082</td>
<td>2.534</td>
<td>0</td>
<td>0.301</td>
<td>1.575</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.942</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1083</td>
<td>2.474</td>
<td>3.966</td>
<td>0.301</td>
<td>1.175</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.1297</td>
<td>0</td>
<td>0</td>
<td>0.1165</td>
<td>0</td>
<td>1.7496</td>
<td>4.1738</td>
<td>4.361</td>
<td>0.4318</td>
<td>0.9971</td>
<td>1.7863</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.0538</td>
<td>2.7129</td>
<td>0</td>
<td>0</td>
<td>0.7585</td>
<td>3.959</td>
<td>4.6987</td>
<td>0.5204</td>
<td>1.9416</td>
<td>2.7206</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2.7726</td>
<td>0</td>
<td>1.0185</td>
<td>0.0602</td>
<td>0</td>
<td>1.7892</td>
<td>4.7461</td>
<td>6.2821</td>
<td>2.968</td>
<td>0.0409</td>
<td>1.6185</td>
<td>1.6198</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.2158</td>
<td>0</td>
<td>0</td>
<td>0.4352</td>
<td>1.151</td>
<td>3.999</td>
<td>3.8669</td>
<td>0.6511</td>
<td>0</td>
<td>1.3266</td>
<td>0.4034</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>2.6591</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.741</td>
<td>4.782</td>
<td>0</td>
<td>2.6633</td>
<td>0.3596</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2.6702</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.9422</td>
<td>5.5135</td>
<td>0</td>
<td>2.6702</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3.5188</td>
<td>0</td>
<td>0</td>
<td>1.2523</td>
<td>0</td>
<td>2.9944</td>
<td>3.5289</td>
<td>1.1795</td>
<td>0</td>
<td>0.5604</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0.625</td>
<td>0.2007</td>
<td>1.9816</td>
<td>4.6222</td>
<td>4.5431</td>
<td>2.1194</td>
<td>0</td>
<td>1.2412</td>
<td>1.0863</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>1.6549</td>
<td>0</td>
<td>0</td>
<td>4.3588</td>
<td>3.3499</td>
<td>1.6549</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0.9488</td>
<td>3.6008</td>
<td>0</td>
<td>0</td>
<td>3.1483</td>
<td>6.2519</td>
<td>6.2235</td>
<td>3.298</td>
<td>1.6099</td>
<td>1.4806</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>2.5919</td>
<td>2.7876</td>
<td>0</td>
<td>0.8495</td>
<td>1.4542</td>
<td>5.0925</td>
<td>5.56</td>
<td>2.6073</td>
<td>1.5118</td>
<td>2.7876</td>
<td>1.4978</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.2386</td>
<td>0</td>
<td>0</td>
<td>0.4226</td>
<td>0.4515</td>
<td>3.9433</td>
<td>4.5173</td>
<td>0.9495</td>
<td>0.4828</td>
<td>1.8998</td>
<td>0.6021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>1.9368</td>
<td>0</td>
<td>0</td>
<td>4.8776</td>
<td>3.7942</td>
<td>1.9308</td>
<td>1.7482</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>1.145</td>
<td>0</td>
<td>0</td>
<td>1.6958</td>
<td>4.0396</td>
<td>6.6823</td>
<td>3.7645</td>
<td>1</td>
<td>1.145</td>
<td>1.738</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9031</td>
<td>2.2529</td>
<td>1.3617</td>
<td>0.4771</td>
<td>0</td>
<td>0.9031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1.0792</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.3852</td>
<td>3.9105</td>
<td>2.2148</td>
<td>0</td>
<td>1.1761</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.9206</td>
<td>6.1813</td>
<td>6.1906</td>
<td>1.4314</td>
<td>0</td>
<td>3.1495</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B. Main Functions in Agobot

Figure B.1 Main() function in Agobot3-0.2.1-pre4-priv
Figure B.2  Agomainctrl() function in Agobot3-0.2.1-pre4-priv
APPENDIX C. Main Functions in SDbot05

Figure C.1 Main() function in sdbot5b
APPENDIX D. Formulation of SPRT-FOD using Cost

D.1 System Model Considered in SPRT-FOD

For all the nodes in a network, classify them into two types: vulnerable and secure nodes. The vulnerable nodes refer to those having a higher probability to be infected (e.g. nodes poorly maintained or having many infection/attack records in the past), while the secure nodes refer to those having higher security guarantee (e.g. well maintained nodes and used by advanced users with good security knowledge). Therefore, if infected by bots, all vulnerable nodes can be controlled by the botmaster.

In contrast, even some secure nodes might be infected by bots, given the higher security level, the bots are expected to be removed in a much more immediate way. Hence, taking into account the fact that botnets will be used multiple times, secure nodes are not sustainable to use for the botmaster. Therefore, we consider those secure nodes not available to the botmaster during C&C in this system model. That is to say, in this system, only vulnerable nodes can be controlled by the botmaster, and need to be checked during detection.

D.2 Problem Formulation

Let $n$ be the total number of vulnerable nodes in a network, then $n$ can also be considered as the nodes needed for SPRT to reach $H_0$ in the normal case. Therefore, from Eq. 4.1 in Chapter 4, we know that when all other parameters are fixed, $p_1$ is only determined by $n$ and vice versa. Moreover, from Eq. 4.1, the larger/smaller the $n$, the smaller/larger the $p_1$. That is to say, choosing larger $p_1$ (smaller $n$) means that we should take more effort to ensure the large portion of nodes to be secure nodes.

Accordingly, if $p_1$ is small ($n$ is large), this means we can take less effort such that only a
relatively small portion of nodes need to be secure. Therefore, $p_1 - p_0$ is a measure on how much effort we need to take to provide corresponding security level under $p_0$. Since this effort can be represented in terms of time or money, $p_1 - p_0$ can be further considered as a cost needed to prevent against bots in the corresponding network. Of course, the smaller the cost, the better the detection scheme from the administrator’s or defender’s perspective.

In contrast, given the provided security level, as discussed in both Chapter 4 and Appendix 5, the portion of bots that can be used by the botmaster without being detected is given as $np'/N$ ($N$ is the total number of nodes in the testing data, $p'$ is derived from Eq. 4.7 or E.1). That is, $np'/N$ is the gain that can be obtained by the botmaster under given $p_1$.

In other words, $np'/N$ is the loss from the defender’s perspective. This loss can also be considered as a cost in terms of time or money (e.g. damages from the botnet in the corresponding network). Therefore, we also want this loss to be small.

In summary, the security effort and portion of nodes available to the botmaster can be considered as costs in terms of time or money. Therefore, we want the overall cost to be minimized. Without loss of generality, we want the sum of these two losses to be minimized, which has been discussed in Chapter 4.

In addition, we use Fig. D.1 to better illustrate the idea discussed in Chapter 4 and this part. In Fig. D.1, although all the parameters including $p_1$ are known to the botmasters, $p_1$ is not fixed and considered as a variable. To derive the value of $p_1$, different coefficients can also be assigned to each cost depending on the relative importance of these costs in the corresponding network.
Figure D.1  Formulation of SPRT-FOD
APPENDIX E. Another Solution to SPRT Optimization

The optimization problem of SPRT discussed in Chapter 4 can be also considered in another much trickier way. That is, in order to be detected under rate $\gamma$, the detection scheme needs to conclude $p'$ from $p_0$ at least $1 - \gamma$ of time. In other words, this is equivalent to the case where $p_0 = p'$ and the false positive rate $\alpha' = \gamma$. Considering the average number $n'$ caused by $p'$ should also be equal to $n$, we can formulate the above problem in another way. As shown in the follows, both ways reach the same results in terms of optimal $n$, hence $p_1$.

\begin{align}
    n' &= E_0(n') = \frac{(1 - \alpha') \ln \frac{\beta}{1-\alpha'} + \alpha' \ln \frac{1-\beta}{\alpha'}} {p' \ln \frac{p_1}{p'} + (1-p') \ln \frac{1-p_1}{1-p'}} = n \quad \text{(E.1)} \\
    p' \ln \frac{p_1}{p'} + (1-p') \ln \frac{1-p_1}{1-p'} &= \frac{(1 - \alpha') \ln \frac{\beta}{1-\alpha'} + \alpha' \ln \frac{1-\beta}{\alpha'}} {n} \quad \text{(E.2)}
\end{align}

Similarly, we formulate the detection problem as the following optimization problem:

$$\min_{1<n<N, n=f(p_1)} c(n) = l_u(n) + l_p(n)$$

where

$$l_u(n) = p_1(n) - p_0, l_p(n) = n * p'(n)/N$$

And the comparison of the previous scheme ($p_1$ based) and this scheme ($p_0$ based) are illustrated in Fig. E.1 E.2 E.3 E.4. The other parameters are the same as used in Chapter 4. As we can find from these figures, although the combined cost are slightly different, the minimum values are achieved under the same $n$ (hence $(p_1)$) for both schemes.
Figure E.1  Optimization on $p_1$ ($p_0 = 0.6$)  

Figure E.2  Optimization on $p_0$ ($p_0 = 0.6$)  

Figure E.3  Optimization on $p_1$ ($p_0 = 0.5$)  

Figure E.4  Optimization on $p_0$ ($p_0 = 0.5$)
Bibliography


[40] Y. Chen, IRC-Based Botnet Detection on High-Speed Routers, ARO-DARPA-DHS Special Workshops on Botnets, June 2006.


[67] S. Chang, T. Daniels, Node Behavior Profiling for Enterprise Network Security, in Pro-
ceedings of Third International Conference on Emerging Security Information, Systems
and Technologies (Securware), June, 2009.


[69] M. Al-kofahi, S. Chang and T. Daniels, SCWIM an Integrity Model for SOA Networks,
in Proceedings of IEEE International Conference on Web Services (ICWS), September,
2008.

[70] S. Chang, T. Daniels, SPRT based Fast and Optimized (SPRT-FOD) Detection of C&C

[71] S. Chang, T. Daniels, A Novel Background Traffic Modeling Structure for NIDS Evalua-


[73] S. Chang, T. Daniels, Realistic Background Modeling using Node Behavior Profiling, in

[74] A. Dainotti, A. Pescape, P. Salvo Rossi, F. Palmieri and G. Ventre. Internet Traffic Model-
ing by means of Hidden Markov Models, International Journal of Computer and Telecom-
communications Networking Volume 52 , Issue 14, 2008

Workloads, Ph.D. Dissertation, Dept. of Computer Science, 2006, University of North
Carolina at Chapel Hill.

[76] S. Avallone, D. Emma, A. Pescap and G. Ventre. A Distributed Multiplatform Architec-


[113] Xen-virtual machine monitor, online at http://www.cl.cam.ac.uk/Research/SRG/netos/xen/


[125] List of Known ports, http://www.bekkoame.ne.jp/s_ita/port/

[126] Trojan Port List, http://www.misec.net/trojanhunter/portlist/


http://home.eng.iastate.edu/ hawklan/nb-index.html.