BotInfer: A Bot Inference Approach by Correlating Host and Network Information

Yukun He, Qiang Li, Yuede Ji, Dong Guo

BotInfer: A Bot Inference Approach by Correlating Host and Network Information *

Yukun He, Qiang Li, Yuede Ji, and Dong Guo

College of Computer Science and Technology, Jilin University,
Changchun 130012, China
{heyk12@mails.jlu.edu.cn, li_qiang@jlu.edu.cn, jiyd12@mails.jlu.edu.cn,
guodong@jlu.edu.cn}

Abstract. Botnet is widely used in cyber-attacks and becomes a serious threat to network security. Existing approaches can detect botnet effectively in certain environments, however problems still exist in using host or network detection approaches respectively, such as robustness in detection tools, difficulties in global deployment and low precision rate. To solve the above problems, a novel detection approach called BotInfer is proposed. In BotInfer approach, host-based bot detection tools are deployed on some of the hosts; network flow of all the hosts is captured and analyzed; host detection result and flow information are correlated by the bot inference engine. Through the experiments, BotInfer can effectively detect the hosts in the network. When the deployment rate of bot detection tools in the network reaches 80%, the precision rate of the hosts with detection tools is about 99%, and the precision rate of the hosts without detection tools is about 86%.

Keywords: bot detection, cluster, flow analysis, inference algorithm

1 Introduction

In order to achieve malicious purposes, attackers would inject particular malicious codes in a large number of hosts by various means and remotely control these hosts through command and control channel (C&C). The network composed of these controlled hosts is known as botnet. The host controlling these compromised hosts is known as botmaster. The malicious code is known as bot. Botnet has become a serious threat to the Internet, which can cause various cybercrimes, such as spreading attack codes and commands, spamming, information theft, phishing and DDoS attacks.

In recent years, a large number of researches have been conducted to detect and prevent botnet. According to the detection location, existing bot and botnet detection approaches can be divided into two categories: (1) Host-based bot detection approaches utilize the abnormal behaviors on hosts to detect bots,

* Supported by the National Natural Science Foundation of China under Grant No.61170265; Fundamental Research Fund of Jilin University under Grant No. 201003035, No. 201103253; Corresponding author: Qiang Li

including abnormal behaviors in registry modification, file system information, system calls, etc. For example, Stinson *et al.* proposed BotSwat approach [1], Lei Liu *et al.* proposed BotTracer approach [2] and Young Park *et al.* proposed BotTee approach [3]. (2) Network-based botnet detection approaches utilize the flow information captured in the network. Such as S.Nagaraja *et al.* proposed BotGrep[4] which utilizes structure graphs. B. Coskun *et al.* proposed a approach utilizing friends of an enemy [5].

However, there are still some defects either in host-based detection or network-based detection approaches. Host-based detection approaches need to deploy detection tools on each host, which will bring direct impact on the performance of hosts. And once the host detection tool is damaged, the detection result of the host would be inaccurate. While the network-based detection approaches need to collect the users' network flow information, which may invade the users' privacy. Due to the limitations of the approaches purely based on hosts or network, Zeng *et al.* [6] proposed a botnet detection approach, which is the first to combine host and network information. The approach of Zeng can effectively detect the botnets which are based on IRC, HTTP and P2P. However global deployment is still needed, because the approach can be effective only when all hosts in the network have been installed host detection tools.

During the procedures of bot detection, detection tools may have the problem of robustness in host detection, the need of global deployment and low detection rate. Robustness which means when bot detection tools were damaged on hosts, detection tools would get the error detection result. Global deployment means detection tools should be deployed on all the hosts in the local network, or there wouldn not be any detection results about the hosts. While in this paper, we propose a novel bot inference approach (BotInfer) which can solve the above problem to a certain extent. Our works make the following three contributions:

1. We propose BotInfer approach. BotInfer has the higher robustness compared with purely host-based bot detection approach. If bot detection tools on some hosts have been damaged by malwares, BotInfer can still obtain the detection results of other hosts and the flow similarity information between hosts in the network to generate the final detection results. BotInfer doesn not need global deployment. Not all hosts need to be deployed with bot detection tools in the entire local network. When the deployment rate reaches a certain point, reliable detection results can be generated through the existing hosts detection information and network flow analysis.

2. We extract 13 features from network flow to calculate the host-flow similarity. Through correlating host-flow similarity and host detection result, bot inference algorithm can infer whether all the hosts in the local network infected bot or not.

3. We implement a detection prototype based on BotInfer approach. And our approach was evaluated by using mixed network flow which is from captured lab flow in multiple time windows and the CERNET network during a day. Our experimental results show that the proposed approach can detect different types of bots with high robustness and property deployment rate.

The remainder of this paper is outlined as follows: Section 2 is related works. Section 3 introduces the design of BotInfer, including the problem statement and assumptions, and the overall architecture of BotInfer. Section 4 implements a model based on BotInfer, including host detection, network flow analysis and bot inference engine. Section 5 experimentally analyzes this approach in terms of accuracy and the rate of deployment. Section 6 is discussion about limitation of BotInfer and the conclusion.

2 Related Works

Currently, primary host-based bot detection approaches include: (1) BotSwat [2], proposed by Stinson *et al.*, which can distinguish between bot behaviors and benign programs. (2) BotTracer [7], proposed by Lei Liu *et al.*, which is to judge bot infection from the three indispensable stages in the process of bot attacking. (3) BotTee [1], proposed by Younghee Park *et al.*, which extracts the suspicious system call sequences to match with the bot command patterns. (4) JACKSTRAWS [8], proposed by Jacob *et al.*, uses machine learning to identify C&C connection accurately. (5) Konrad Rieck et al. [9] used the machine learning algorithm to automatically analyze malware behavior. (6) Fatemeh Karbalaie et al. [10] used data miner approache to detect host malwares.

Network detection approaches include flow graph analysis, flow features clustering, machine learning, the analysis of activities of network flow. (1) BotFinder proposed by Florian Tegeler *et al.* [11], used the machine learning to divide the captured network flow into two types: benign and malicious and the final model generated will decide whether the flow generated by hosts is malicious or not. (2) Leyla Bilge *et al.* proposed DISCLOSURE approach [12], using large-scale network flow data, extracting flow features to detect C&C server in botnet. (3) Francois *et al.* proposed Bottrack approach [13], analyzing bots' communication patterns via NetFlow data analysis and PageRank algorithm.

In host-based approaches, any damage on bot detection tools will completely fail the detection, so detection tools must be deployed on all hosts. Besides, bot detection tools need to monitor system information of the user hosts, and it will decrease the performance of user hosts. In network detection approaches, they rely only on network flow and do not consider the hosts detection information, so the detection accuracy is low. In our BotInfer approach, hosts detection information and network flow analysis information are effectively correlated, the above problems are solved to a certain extend.

3 Bot Inference Approach

3.1 Problem Statement and Assumptions

Botmaster spreads commands to bots via C&C channel. After receiving commands, the bots on hosts will perform malicious behaviors, in such areas as, the allocation of file resources, generation of registry, network flow on hosts and so

on. As a result, we can monitor the information on hosts to analyze whether the host has been infected. For all the flows of bots are generated automatically in the background, rather than through artificial operations, so there are great similarities in communication flows between botmaster and bots. For example, in the centralized architecture, the bots receive commands almost simultaneously from the centralized server, and their communication flows are very similar with each other in the aspects of the number of packets, the size of packets. While, in the distributed architecture, the commands from botmaster need to be spread among hosts, so the flows between hosts infected with the same bot also have great similarity. Using the results detected on hosts and the information of flow similarity obtained by flow analysis, we can finally infer whether the hosts are bots or not through inference algorithm. BotInfer is a bot detection approach used in a local network based on the above assumptions. BotInfer mainly targets at a large local network to effectively detect bots, which use IRC, HTTP and P2P as their C&C channels, when the deployment rate of host detection tools reaches a certain point. The detection accuracy for unknown bot mainly depends on the accuracy of the host detection tools.

3.2 Architecture of BotInfer

In Figure 1, the approach of BotInfer is mainly divided into three sections. S1 is the host detection, which deploys detection tools on some of the hosts in the network. When detection tools find out bot activities on hosts, they will immediately generate detection results and suspicious degrees, which will be sent to the bot inference engine. S2 is the network analysis, which obtains the communication flows of all hosts in the network and filters safe IP address got from the known safe URLs (such as, `www.microsoft.com`, `www.google.com`). We believe that the communication activities between hosts and these URLs are benign behaviors. Flow features are extracted from filtered flows, such as flow duration, packet size and packet quantity. Then according to the features of host flows, the hosts with similar flow in a certain period are put into the same cluster and the similarity degree is calculated. S3 is the most important part of the BotInfer approach , which is used to correlate the detection results generated in S1 and the data sets got by S2 to obtain the final results.

4 Implementation

A prototype is implemented based on BotInfer approach. Existing bot detection tools are used to get host detection result. Network flow is filtered and clustered. Algorithms are implemented in network flow similarity calculation and bot inference engine.

4.1 Host Detection

Host-based detection approaches are in large numbers, which mainly analyze the abnormal behaviors of hosts [14] [15] [16] [17]. Instead of doing in-depth

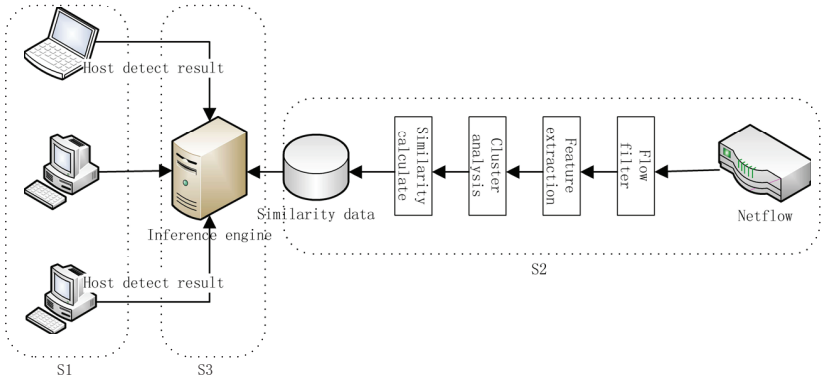

Fig. 1. BotInfer architecture

researches on host-based detection approaches, we pay more attention to analyse the network flow and the inference engine. In the experiments, traditional antivirus tools are used to get the detection results on hosts.

4.2 Network Analysis

In order to obtain the similarity degree of the hosts' communication flows, the communication flows of hosts in entire network need to be captured in a time window, and then do the work of flow filtering, extracting flow features, clustering analysis and calculating flow similarity degree.

Flow Capture and Filter When extracting flow features, we only care about the overall statistics of hosts' communication flows and do not research the specific content of communication packets, so this will not involve users' privacy data. For the acquired data of flows, it can be filtered by the white list of the IP addresses. It is believed that it is secure for user hosts to communicate with the hosts in white list, such as `www.facebook.com` and `www.microsoft.com`. The flows generated from the IP on white list can be filtered, then the data quantity is decreased greatly when analyzing network flow, which is sure to cut the calculation overhead to a great extend. We won not filter the flows of internal hosts in a local network, because in botnet, which uses P2P as C&C channel, the internal hosts also communicate with each other and their flows are similar.

Flow Feature Extraction and Cluster For the data of network flow after filtering process, the feature information of flows can be extracted according to the IP addresses in flows. Due to the fact that the bots on hosts usually generate less flows, if we simply collect the flows between a host and other hosts, the flows generated by bots and botmaster will be covered by other processes, and the features of flows generated by bots will not be obvious. As a result, it is not

Table 1. Host flow feature

IP_IP	IP of two hosts (no distinction between source and target host)
totalFlows	total flows between two hosts
totalPackets	total packets of the flows between two hosts
totalbytes	total bytes of the flows between two hosts
totalDuration	total durations of the flows between the two hosts
packetsVariance	the variance of packets number in each flow
bytesVariance	the variance of bytes number in each flow
durationVariance	the variance of durations number in each flow
packetsPerFlow	the number of packets per flow
bytesPerFlow	the number of bytes per flow
durationPerFlow	the duration per flow
numberOfPort	total number of ports used in communication
numberOfTcp	total number of TCP flows in communication
numberOfUdp	total number of UDP flows in communication

conducive to obtaining flow similarity of different hosts in a same time window. Therefore, we do not consider the direction of flows, that is, do not distinguish the IP between the source hosts and the target hosts of flows. We analyze all communication information of two hosts within a time window to generate a vector composed of 13 features. Table 1 shows all the information contained in a feature vector. According to the feature, the similar flows can be put into the same cluster by using the approach of hierarchical clustering, Davies-Bouldin (DB) [18], which chooses an appropriate height to split the dendrogram.

Flow Similarity Calculation This part mainly calculate the similarity of host communication flows in the same time window and the same cluster through the use of similarity information of flows between different hosts. P and F are used to record the flows information between two hosts within a time window. P is composed of the IP of two hosts and F is the vector composed of the 13 features of flows between two hosts. For instance, the communication flows between host A and B can be expressed as (P_{AB}, F_{AB}). Their similarity is calculated through the distance of features, for features j and k, their similarity is:

$$S_{jk} = \left| \frac{\sum_{i=1}^{13}(x_{ij} - \overline{x_j})(x_{ik} - \overline{x_k})}{\sqrt{\sum_{i=1}^{13}(x_{ij} - \overline{x_j})^2}\sqrt{\sum_{i=1}^{13}(x_{ik} - \overline{x_k})^2}} \right| \tag{1}$$

x_{ij} denotes the ith feature of j, x_{ik} denotes the ith feature of k, $\overline{x_j} = \frac{1}{13}\sum_{i=1}^{13} x_{ij}$, $\overline{x_k} = \frac{1}{13}\sum_{i=1}^{13} x_{ik}$.

For the two flow-features in a cluster (P_{AB}, F_{AB}) and (P_{CD}, F_{CD}), the similarity S_{ABCD} between two flow features can be obtained by calculating F_{AB} and F_{CD}. When analyzing the IP of hosts, the similar flows can be divided into two types: (a) and (b) in Figure 2. The similar flows in (a) are generated by four different hosts and this type is regular in distributed botnet using P2P as

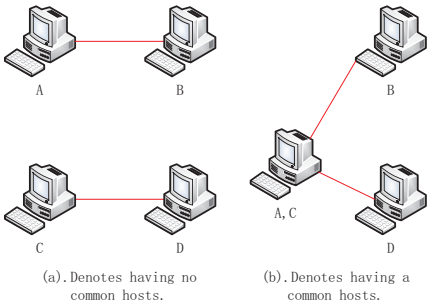

Fig. 2. Flow similarity.

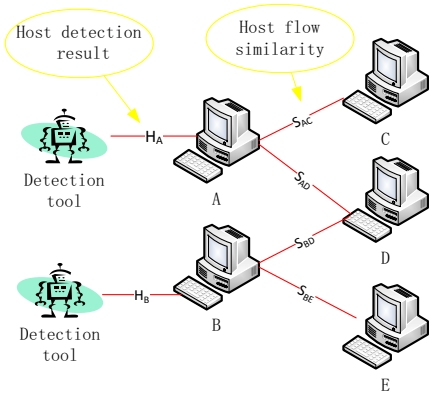

Fig. 3. Inference algorithm

C&C protocol. HS is used to represent the host similarity, so we can get the HS between hosts A and C, A and D, B and C, and B and D, for example, in Figure 3 $HS_{AC} = HS_{AD} = HS_{BC} = HS_{BD} = S_{ABCD}$. The similar flows in (b) are generated by the same host. As shown in (b), A and C is the same host and this host may be a botmaster. This situation is regular in centralized botnets using IRC or HTTP as C&C protocol. So, we can get the host similarity of hosts B and D: $HS_{BD} = S_{ABCD}$. FS is used to represent the final similarity between hosts, that is $FS_{AB} = max(HS_{AB})$.

4.3 Bot Inference Engine

According to the results of host detection and flow similarity analysis, bot inference engine can calculate the suspicion degree of the hosts without reporting detection results. As shown in the Figure 3, there are five hosts in a cluster, A, B, C, D and E. The detection results of A and B are H_A and H_B, hosts C, D and E report nothing. S_{AC} represents the calculation results of similarity between A and C. Host D has no detection result, due to the fact that it has similar communication flows with hosts A and B, what's more, bots have been found out on hosts A and B, then the final detection result of host D is: $F_D = (H_A S_{AD} + H_B S_{BD})/(S_{AD} + S_{BD})$. So a reliable detection result for the hosts without detection tools or with invalid detection tools is infered. If the host D has reported its detection result H_D and the result shows that host D has been infected with bots, however the degree of suspicion is not very high, we could not accurately judge whether host D has really been infected with bots, and there may be the possibility of activities of benign programs leading to the inaccurate report by detection tools. We can calculate the final detection result of host D more accurately through bot inference engine to combine flows similarity information of other hosts in network, $F_D = (H_D + (H_A S_{AD} + H_B S_{BD})/(S_{AD} + S_{BD}))/2$. Meanwhile, if there are

a large number of hosts in the network, which have similar flows with host D and send their detection reports, the detection results could be more accurate through bot inference engine.

For more general situations, X represents any host in network and it is in cluster $N(X)$. The similarities between all hosts in $N(X)$ are in a certain range. H_X represents the detection results generated by host X, and S_{KX} means the similarity between host K and host X. Finally, the detection result of host X through inference engine is

$$F_X = \begin{cases} \sum_{K \in N(X)} H_K S_{KX} / \sum_{K \in N(X)} S_{KX} & , H_X = 0 \\ (H_X + \sum_{k \in N(X)} H_K S_{KX} / \sum_{K \in N(X)} S_{KX})/2 & , H_X \neq 0 \end{cases} \tag{2}$$

The range of H_X is between 0 and 1, and the larger the value, the greater the suspicion degree of whether the host has been infected with bots. The range of host flows similarity S_{KX} is also between 0 and 1 and the larger the value, the greater the flows similarity between two hosts. Analyzing the above inference algorithm, it is easy to obtain the final detection result F_X, which also ranges from 0 to 1, and the larger the value, the greater the probability to be infected with bots.

5 Experiment

According to the algorithm proposed, we design and implement the prototype based on BotInfer and analyze its accuracy and deployment rate through experiments. For host detection, we use the average detection result of those acquired through several existing tools as the suspicion degree of whether the host is infected by bots. In network analysis, the fprobe [1] [19] and flow-tools [20] are used to capture the flows of the whole network. According to the IP information, the captured host flows are filtered by safe-browling [21] proposed by google and top 1,000,000 URLs [22] proposed by alexa. Then the data is processed and analyzed by hclust [23] package and Python language. The final detection result is obtained through inference engine.

5.1 Environment Setup

In the experiment, a controllable local network is built. User hosts are deployed in VMware virtual machines with Windows XP Professional. BotInfer is deployed in a Windows XP Professional host which has been equipped with quad core 2.40GHz CPU and 2G RAM. The network information collector and analyzer are deployed in a Ubuntu 10.10 host equipped with 2.40GHz CPU and 2G RAM, and this host is used as the gateway of the entire lab network. The topology of experiment environment is shown in Figure 4.

[1] NetFlow probes: fprobe and fprobe-ulog, `http://fprobe.sourceforge.net/`

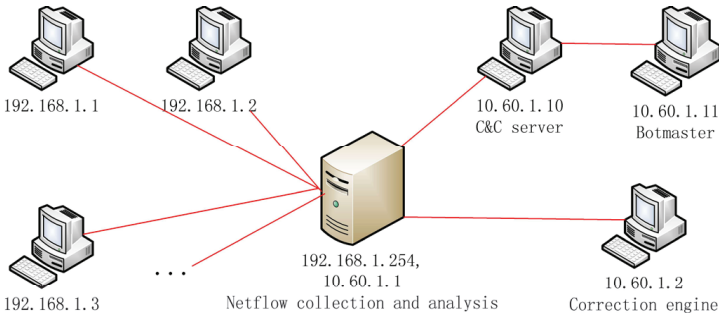

Fig. 4. Experiment architecture

To evaluate the effectiveness of the algorithm in detecting bot host, the bot programs are installed in only some of user hosts and the botmaster is installed in a host in a public network. Because of the uncontrollability of P2P botnet, we only use HTTP and IRC bots, including SdBot, AgoBot, RBot and Nugache. In order to better simulate the real environment of user hosts, we install and run softwares like mIRC, pcAnywhere, Firefox, eMule and uTorrent, etc. in user hosts and let them use the network as usual. For the network flow analysis, we capture the network flow of a certain backbone at the CERNET network during a day as the background data. There are 755,255 flow records in a time window of 10 minutes. After filtration, we get the information of 63,589 hosts. We integrate the filtered information of features in the public network with the flow features captured in local network. This will better evaluate the effectiveness of cluster analysis when distinguishing bot flows.

5.2 Experimental Result and Analysis

Table 2. Host configurations

Host IP	Bot Host	Detection Tools	Common Software
192.168.1.1-192.168.1.24	yes	yes	yes
192.168.1.25-192.168.1.30	yes	no	yes
192.168.1.31-192.168.1.40	no	no	yes

The Accuracy We use 40 VMware machines as user hosts which have been installed with Windows XP Professional. The hosts are deployed in the same network which is 192.168.1.1-192.168.1.40/24. Bot instances in the hosts, host detection tools and the running of benign softwares are shown in Table 2. We guarantee that the deployment rate of host detection tools deployed on bot

Table 3. The accuracy of BotInfer

Bot name	Average FP	Average FN	Average TP	Average TN	Duration
SdBot	0	0.02	0.98	1	24h
AgoBot	0	0.04	0.96	1	24h
RBot	0	0.04	0.96	1	24h
Nugache	0	0.05	0.95	1	24h

infected hosts comes to 80%, that the benign softwares on hosts can access the network and the botmaster can communicate with bots in C&C channel as usual. The commands we use include dns, open, download, redirect, etc. On the network flow collector and analyzer, feature filtering and feature extraction processes are both in a time window of 10 minutes. Then mixed flow-features which are extracted from lab network flows and background are used to do cluster analysis and similarity calculation. Finally we get the detection result of the entire network through BotInfer. Table 3 shows the average detection results of the 4 bot instances being detected respectively in individual time windows during 24 hours.

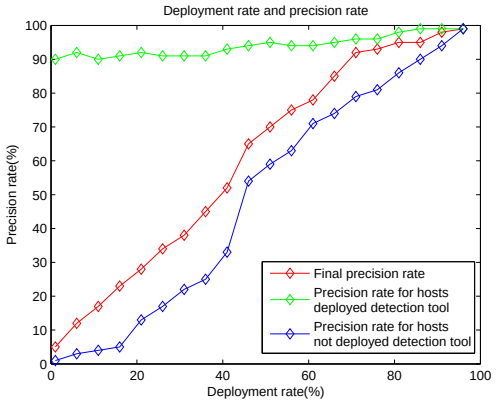

Fig. 5. Deployment rate and precision rate

Deployment Rate BotInfer has improved bot detection accuracy to a certain extent. Detection tools need not to be deployed on all the hosts in the entire network. It can get all hosts detection results through bot inference engine, avoid the failure caused by bot detection tools' failure on some of the hosts in the network, so as to improve the robustness. In normal user hosts, we regularly adjust the number of hosts infected with bots, the number of benign hosts and the number of hosts being installed with detection tools, and we analyze multiple bot

instances in IRC and HTTP botnets. Figure 5 shows the influence of deployment rate of host detection tools on the overall test results. When the deployment rate is over 50%, the accuracy of the test results of the entire network will be significantly improved; when deployment rate is over 80%, the detection accuracy rate of hosts without detection tools increases to more than 86%.

6 Conclusion

Although there are a lot of ways to analyze network flows, there are still many challenges. When it comes to flow analysis, the following questions have been solved in this paper: how to select the flow-features to distinguish benign flows from malicious flow efficiently; how many clusters needed is reasonable when partition flow-features; the storage of the final results. At the same time, we also have figured out the differences between the final results generating from the inference engine of hosts with detection tools and those of hosts without detection tools.

Through analyzing the architecture of BotInfer, the results acquired by inference engine still depend on the results of host detection tools to a large extent. Pure network analysis is unable to get the degree of suspicion for hosts in the network directly. What's more, the detection of unknown botnet also mainly depends on the efficiency of host detection tools. In order to reduce the influence of host detection results on correlation results, a possible way is to deploy more than one detection tools in the networks to improve the efficiency of host detection, another way is to get the network detection result using machine learning or other graph algorithm to analyse the captured network flow.

With botnets evolving, a large number of hosts are still suffering from bots. In this paper, BotInfer is able to infer hosts infected with bot in the entire network efficiently when the deployment rate of host detection tools reaches a certain point in the local network. We have picked up 13 features of the collected flows and calculate the flow similarity between hosts so as to distinguish the flow of benign programs from that of bots in C&C communication. Finally, through inference engine to combine the results of host detection and that of network flow analysis, the detection report can be acquired for all hosts in the network including the hosts with and without detection tools. In Botinfer, we have conducted experiments in the lab, and analyzed them in multiple aspects.

References

1. Younghee Park and Douglas S. Reeves, Identification of bot commands by run-time execution monitoring, 2009 Annual Computer Security Applications Conference, pages 321C330 (2009)
2. Stinson.E. , Mitchell.J. , Characterizing Bots Remote Control Behavior, 4th DIMVA Conference (July 2007)
3. Lei Liu, Songqing Chen, Guanhua Yan and Zhao Zhang, BotTracer: Execution-Based Bot-Like Malware Detection, international conference on Information Security (2008)

4. Baris Coskun, Sven Dietrich and Nasir Memon, Friends of An Enemy: Identifying Local Members of Peer-to-Peer Botnets Using Mutual Contacts, 2010 ACSAC Conference (2010)
5. S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar and N. Borisov, BotGrep: Finding P2P bots with structured graph analysis. In USENIX Security Conference (August 2010)
6. Yuanyuan Zeng, Xin Hu and Kang G.Shin, Detection of Botnets Using Combined Host- and Network-Level Information, DSN (2010)
7. Lei Liu, Songqing Chen, Guanhua Yan and Zhao Zhang, BotTracer: Execution-Based Bot-Like Malware Detection, international conference on Information Security(ISC) (2008)
8. G. Jacob, R. Hund, C. Kruegel and T. Holz, JACKSTRAWS: Picking Command and Control Connections from Bot Traffic, USENIX Security Symposium (2011)
9. Konrad Rieck and Philipp Trinius, Carsten Willems Automatic analysis of malware behavior using machine learning, Journal of Computer Security, Volume 19, Number 4 (2011)
10. Fatemeh Karbalaie, Ashkan Sami and Mansour Ahmadi, Semantic Malware Detection by Deploying Graph Mining, International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, (2012)
11. Florian Tegeler, Xiaoming Fu, Giovanni Vigna and Christopher Kruegel, BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection, CoNEXT (2012)
12. Leyla Bilge, Davide Balzarotti and William Robertson, DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis, ACM (2012)
13. J. Francois, S. Wang, R. State and T. Engel, Bottrack: Tracking Botnets Using Netflow and Pagerank, In IFIP Networking (2011)
14. Guofei Gu, Correlation-based Botnet Detection in Enterprise Networks, Doctor Thesis, GIT (2008)
15. Young Hee Park, Qinghua Zhang, Douglas S and Reeves, AntiBot: Clustering Common Semantic Patterns for Bot Detection, COMPSAC (2010)
16. Taeho Kwon, Zhendong Su, Modeling High-Level Behavior Patterns for Precise Similarity analysis of Software, Technical Reports, University of California, CSE-2010-16, (2010)
17. Xinyuan Wang and Xuxian Jiang, Artificial Malware Immunization based on Dynamically Assigned Sense of Self, ISC (2010)
18. M. Halkidi, Y. Batistakis and M. Vazirgiannis, On Clustering Validation Techniques, JIIS,17(2-3):107C145 (2001)
19. NetFlow probes: fprobe and fprobe-ulog, `http://fprobe.sourceforge.net/`
20. flow-tools, `http://www.splintered.net/sw/flow-tools/docs/flow-tools.html`
21. Safe Browsing API - Google Developers, `https://developers.google.com/safe-browsing/`
22. Alexa Top 500 Global Sites, `http://www.alexa.com/topsites`
23. R: Hierarchical Clustering, `http://stat.ethz.ch/R-manual/R-patched/library/stats/html/hclust.html`