© 2015 IJSRSET | Volume 1 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Building a Scalable System for Stealthy P2P-Botnet Detection

B. Anitha, Avinash Sivan, V. Hari Prasath, S. Selvaraj
Computer Science and Engineering, Dhanalakshmi College of Engineering, Chennai, Tamilnadu, India

ABSTRACT

In this paper we discussed about Peer-to-peer (P2P) because botnets have recently been taken by botmasters for their attack against take-down efforts. Inside being harder to take down, modern bot nets tend to be attack in the way they perform malicious activities, making current detection approaches ineffective. In addition, the rapidly growing volume of network traffic calls for high measurable of detection systems. We propose a new measurable botnet detection system capable of detecting attack P2P botnets. ABOTNET is a collection of compromised hosts that are remotely controlled by an attacker (the botmaster) through a command and control (C&C) channel. Botnets serve as the infrastructures responsible for a variety of cyber-crimes, such as spamming, distributed denial of-service (DDoS) attacks, identity theft, click fraud, etc. The C&C channel is an essential component of a botnet because botmasters rely on the C&C channel to issue commands to their bots and receive information from the compromised machines. Botnets may structure their C&C channels in different ways.

Keywords: Botnet Detection, Software Architecture, Signature Based Etection, Data Mining, Click Fraud, Search Log Analysis

I. INTRODUCTION

Recent malicious attempts are intended to get financial benefits through a large pool of compromised hosts, which are called software robots or simply "bots." A group of bots, referred to as a botnet, is remotely controllable by a server and can be used for sending spam mails, stealing personal information, and launching DDoS attacks. Growing popularity of botnets compels to find proper countermeasures but existing defense mechanisms hardly catch up with the speed of botnet technologies. [1] In this paper, we propose a botnet detection mechanism by monitoring DNS traffic to detect botnets, which form a group activity in DNS queries simultaneously sent by distributed bots. A few works have been proposed based on particular DNS information generated by a botnet, but they are easily evaded by changing. ABOTNET is a collection of compromised hosts that are remotely controlled by an attacker (the botmaster) through a command and control (C&C) channel. Botnets serve as the infrastructures responsible for a variety of cyber-crimes, such as spamming, distributed denialof-service (DDoS) attacks, identity theft, click fraud, etc. The C&C channel is an essential component of a botnet because botmasters rely on the C&C channel to issue commands to their bots and receive information from the compromised machines. Botnets may structure their C&C channels in different ways.

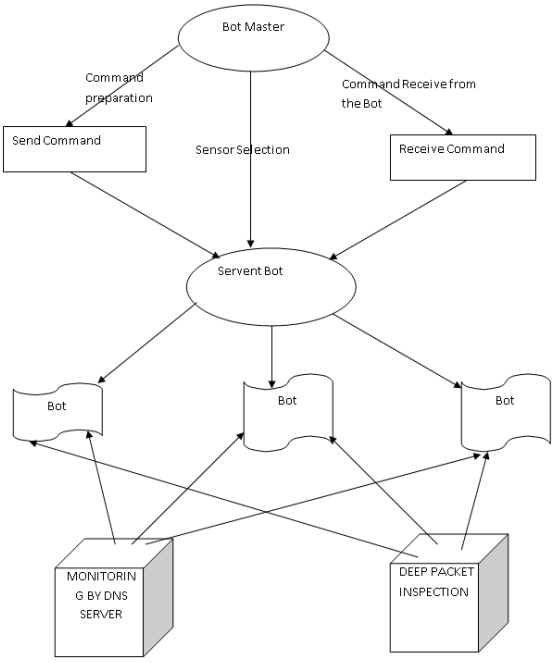


Figure 1: The System

II. METHODS AND MATERIAL

A. User Interface Design

In this module we design the windows for the project. These windows are used to send a message from one peer to another. We use the Swing package available in Java to design the User Interface. Swing is a widget toolkit for Java.[2][3] It is part of Sun Microsystems' Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs System Overview: A P2P botnet relies on a P2P protocol to establish a C&C channel and communicate with the botmaster. Therefore P2P bots exhibit some network traffic patterns that are common to other P2P client applications (either legitimate or malicious). Thus, we divide our systems into two phases. In the first phase, we aim at detecting all hosts within the monitored network that engage in P2P communications. As shown in Figure 1, we analyze raw traffic collected at the edge of the monitored network and apply a pre-filtering step to discard network flows that are unlikely to be generated by P2P applications. We then analyze the remaining traffic and extract a number of statistical features to identify flows generated by P2P clients. [4] In the second phase, our system analyzes the traffic generated by the P2P clients and classifies them into either legitimate P2P clients or P2P bots. Specifically, we investigate the active time of a P2P client and identify it as a candidate P2P bot if it is persistently active on the underlying host. We further analyze the overlap of peers contacted by two candidate P2P bots to finalize detection.

Identifying P2P Clients

Traffic Filter the Traffic Filter component aims at filtering out network traffic that is unlikely to be related to P2P communications. This is accomplished by passively analyzing DNS traffic, and identifying network flows whose destination IP addresses were previously resolved in DNS responses.[7] Specifically, we leverage the following feature: P2P clients usually contact their peers directly by looking up IPs from a routing table for the overlay network, rather than resolving a domain name. This feature is supported by Table II (No-DNS Peers), which illustrates that the vast majority of flows gener- ated by P2P applications do not have destination IPs resolved from domain names. The remaining small fraction of flows are corresponding to a possible exception that a peer bootstraps into a P2P network by looking up domain names that resolve to stable super-nodes) Since most non-P2P applications (e.g., browsers, email clients, etc.) often connect to a destination address resulting from domain name resolution, this simple filter can eliminate a very large percentage of non-P2P traffic, while retaining the vast majority of P2P communications.

Fine-Grained Detection of P2P Clients

This component is responsible for detecting P2P clients by analyzing the remaining network flows after the Traffic Filter component. For each host h within the monitored network we identify two flow sets, denoted as Stcp(h) and Sudp(h), which contain the flows related to successful outgoing TCP and UDP connection, respectively. We consider as successful those TCP connections with a completed SYN, SYN/ACK, ACK handshake, and those UDP (virtual) connections for which there was at least one “request” packet and a consequent response packet.

Coarse-Grained Detection of P2P Bots

Since bots are malicious programs used to perform profitable malicious activities, they represent valuable assets for the botmaster, who will intuitively try to maximize utilization of bots. This is particularly true for P2P bots because in order to have a functional overlay network (the botnet), a sufficient number of peers needs to be always online. In other words, the active time of a bot should be comparable with the active time of the underlying compromised system. If this was not the case, the botnet overlay network would risk degenerating into a number of disconnected sub networks due to the short life time of each single node. In contrast, the active time of legitimate P2P applications is determined by users, which is likely to be transient.[6] For example, some users tend to use their file- sharing P2P clients only to download a limited number of files before shutting down the P2P application [20]. In this case, the active time of the legitimate P2P application may be much shorter compared to the active time of the underlying system. It is worth noting that some users may run certain legitimate P2P applications for as long as their machine is on.[8] For example, Skype is a popular P2P application for instant messaging

and voice-over-IP (VoIP) that is often setup to start after system boot, and that keeps running until the system is turned off. Therefore, such Skype clients (or other "persistent" P2P clients) will not be filtered out at this stage. Hence, the first component in the "Phase II" of our system ("Coarse-Grained Detection of P2P Bots") aims at identifying P2P clients that are active for a time TP2P close to the active time Tsys of the underlying system they are running on.

While this behavior is not unique to P2P bots and may be representative of other P2P applications (e.g., Skype clients that run for as long as a machine is on), identifying persistent P2P clients takes us one step closer to identifying P2P bots. To estimate Tsys we proceed as follows. For each host h ∈ H that we identified as P2P clients according to Section IV-B, we consider the timestamp tstart(h) of the first network flow we observed from h and the timestamp tend(h) related to the last flow we have seen from h. Afterwards, we divide the time tend(h)−tstart(h) into w epochs (e.g., of one hour each), denoted as T =[t1,...ti,...,tw]. We further compute a vector A(h,T) =[a1,...ai,...,aw] where ai is equal to 1 if h generated any network traffic between ti−1 and ti. We then estimate the active time of h as Tsys = w i=1 ai. In order to estimate the active time of a P2P application, we can leverage obtained fingerprint clusters. It is because that a P2P application periodically exchanges network con- trol (e.g., ping/pong) messages with other peers as long as the P2P application is active. For each host h (again, we consider only the hosts in H, which we previously identi- fied as P2P clients), we examine the set of its fingerprint clusters FC(h) ={FC1,...FCj ...,FCk} (see Section III).[7] Based on the flows belonging to a fingerprint cluster FCj, we use the same approach of computing Tsys to calcu- late its active time, denoted as T(FCj). Then, we estimate the active time (TP2P) of a P2P application as ˆ TP2P =max (T(FC1),...T(FCj),...T(FCk)).

B. De-Activate Traffic

The Traffic Filter component aims at filtering out network traffic that is unlikely to be related to P2P flows whose destination IP addresses were previously resolved in DNS responses. Specifically, we leverage the following feature: P2P clients usually contact their peers directly by looking up IPs from a routing table for the overlay network, rather than resolving a Domain name.

C. Coarse Grained Peer-To-Peer Detection

This component is responsible for detecting P2P clients by analyzing the remaining network flows after the Traffic Filter component. For each host h within the monitored network we identify two flow sets, denoted as Stcp (h) and Sudp (h), which contain the flows related to successful outgoing TCP and UDP connection, respectively. [5] We consider as successful those TCP connections with a completed SYN, SYN/ACK, ACK handshake, and those UDP (virtual) connections for which there was at least one "request" packet and a consequent response packet.

D. Coarse Grained Bot Detection

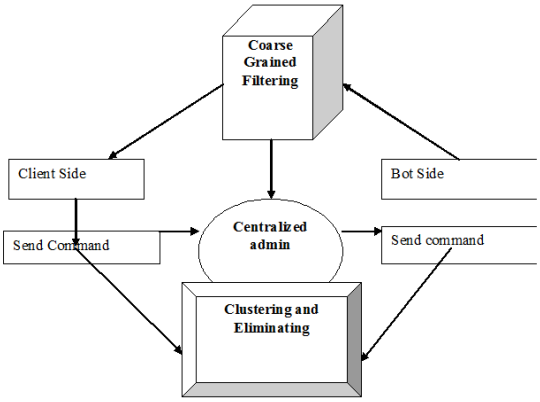
Since bots are malicious programs used to perform profitable malicious activities, they represent valuable assets for the botmaster, who will intuitively try to maximize utilization of bots. This is particularly true for P2P bots because in order to have a functional overlay network (the botnet), a sufficient number of peers needs to be always online. In other words, the active time of a bot should be comparable with the active time of the underlying compromised system.

E. Clustering And Eliminating

The distance between two flows is subsequently defined as the euclidean distance of their two corresponding vectors. We then apply a clustering algorithm to partition the set of flows into a number of clusters. Each of the obtained clusters of flows, Cj (h), represents a group of flows with similar size. For each Cj (h), we consider the set of destination IP addresses related to the flows in the clusters, and for each of these IPs we consider its BGP prefix (using BGP prefix announcements).

III. RESULTS AND DISCUSSION

The implementation objective is to integrate high scal- ability as a built-in feature into our system. To this end, we first identify the performance bottleneck of our sys- tem and then mitigate it using complexity reduction and parallelization.



A. Performance Bottleneck

Out of four components in our system, "Traffic Filter" and "Coarse-Grained Detection of P2P Bots" have linear complexity since they need to scan flows only once to identify flows with destination addresses resolved from DNS queries or calculate the active time. Other two components, "Fine-Grained Detection of P2P Clients" and "Fine-Grained P2P Detection of P2P Bots", require pairwise comparison for distance calculation. Specifically, if we denote the number of flows generated by a host as n and the number of hosts as S, the time complexity of Fine-Grained Detection of P2P Clients approximates O(S∗n2). Comparably, if we denote the number of persistent P2P clients as l, the time complexity of Fine- Grained P2P Bot Detection approximates O(l2). Since the number of flows generated by network applications (i.e., n) could be enormous (e.g., more than hundreds of thousands of flows are generated by a single P2P client in our experiments), the computation overhead of Fine-Grained Detection of P2P Clients may become prohibitive. On contrary, the precent age of P2P clients in the ISP network is relatively small (e.g., 3%-13% as reported in [22]). Consequently, Fine- Grained P2P Bot Detection is unlikely to introduce huge performance overhead. For instance, given a typical ISP net- work or a large enterprise network that has 65,536 hosts (/16 subnet), if we assume that 8% hosts run P2P applications and conservatively assume that half of them are persistent, the number of persistent P2P clients (i.e., l) subject to analysis by Fine-Grained P2P Bot Detection is 2,221, incurring negligible overhead. To summarize, "Fine-Grained P2P Client Detection" is the performance bottleneck.

B. Two-Step Flow Clustering

We use a two-step clustering approach to reduce the time complexity of "Fine-Grained P2P Client Detection". For the first-step clustering, we use an efficient clustering algorithm to aggregate network flows into K sub-clusters, and each sub- cluster contains flows that are very similar to each other. For the second-step clustering, [10] we investigate the global distribution of sub-clusters and further group similar sub-clusters into clusters.

C. System Parallelization

Since the two-step clustering analyses network flows for each single host, we can parallelize the computation for all hosts.[9] We formulate the problem as follows: given S hosts denoted as H ={ h1,h2,...hS} and M computation nodes denoted as C ={ c1,c2,...cM}, we partition H into Mexclusive subsets HT 1, HT 2..HTM and assign HT i to ci for analysis, whose processing time is denoted as exc(ci, HT i). Our target is to design a partition algorithm so that the overall processing time, denoted as T = max(exc(ci, HT i)), is minimized. If we assume each computation node has the same capacity, T will be minimized when the analysis workload is evenly distributed across all computation nodes.

IV. CONCLUSION

We conducted a systematic study on the feasibility of solely using DNS queries for massive-scale stealthy communications among entities on the Internet. Our work shows that DNS—in particular the code word mode combined with advanced querying strategies—can be used as an extremely effective stealthy C&C channel. To address the open problem raised in on how to algorithmically generate short-lived and realistic-looking domain names, we found that using MC produces realistic-looking domain names. Our work points out the potential severity of DNS abuse for massive-scale communications and the challenges associated with its detection. Understanding the capacity of botnets communication power helps identify and eliminate nefarious attacks launched from them. DNS based botnet C&C is stealthier than application based C&C (e.g., e-mail or social network, and such a C&C system also benefits from the decentralization of DNS. Some of our

anomaly detection analysis is useful beyond the specific DNS tunnelling problem studied.

V. REFERENCES

[1] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2P is here," in Proc. USENIX, vol. 32. 2007, pp. 18–27.

[2] P. Porras, H. Saidi, and V. Yegneswaran, "A multi-perspective analysis of the storm (peacomm) worm," Comput. Sci. Lab., SRI Int., Menlo Park, CA, USA, Tech. Rep., 2007. P. Porras, H. Saidi, and V. Yegneswaran. (2009). Conficker C Analysis [Online]. Available: http://mtc.sri.com/Conficker/addendumC/index.html

[3] G. Sinclair, C. Nunnery, and B. B. Kang, "The waledac protocol: The how and why," in Proc. 4th Int. Conf. Malicious Unwanted Softw., Oct. 2009, pp. 69–77.

[4] R. Lemos. (2006). Bot Software Looks to Improve Peerage [Online]. Available: http://www.securityfocus.com/news/11390

[5] Y. Zhao, Y. Xie, F. Yu, Q. Ke, and Y. Yu, "Botgraph: Large scale spamming botnet detection," in Proc. 6th USENIX NSDI, 2009, pp. 1–14.

[6] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in Proc. USENIX Security, 2008, pp. 139–154.

[7] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? Telling P2P file-sharing and bots apart," in Proc. ICDCS, Jun. 2010, pp. 241–252.

[8] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "BotGrep: Finding P2P bots with structured graph analysis," in Proc. USENIX Security, 2010, pp. 1–16.

[9] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in Proc. 6th ACM Symp. Inf., Comput. Commun. Security.