SECURE AND EFFICIENT DATA TRANSMISSION AND STEALTHY P2P-BOTNET DETECTION FOR BUILDING A SCALABLE NETWORK

Bhanupriya T M[1] Sujatha S R[2] M Siddappa[3]
[1]M.Tech , [2]Associate Professor, [3]Professor and Head
Computer Science and Engg,Dept. Sri Siddhartha Institute of Technology,
Tumkur, Karnataka
Email: [1]bhanupriya.tm91@gmail.com, [2]sujathassit@gmail.com, [3]siddapa.p@gmail.com

***Abstract* - Peer-to-peer (P2P) botnets have recently been adopted by botmasters for their resiliency against take-down efforts. Besides being harder to take down, modern botnets tend to be stealthier in the way they perform malicious activities, making current detection approaches ineffective. In addition, the rapidly growing volume of network traffic calls for high scalability of detection systems. This scheme, propose a novel scalable botnet detection system capable of detecting stealthy P2P botnets. The system first identifies all hosts that are likely engaged in P2P communications. It then derives statistical fingerprints to profile P2P traffic and further distinguish between P2P botnet traffic and legitimate P2P traffic. The parallelized computation with bounded complexity makes scalability a built-in feature of this system. Extensive evaluation has demonstrated both high detection accuracy and great scalability of the proposed system.**

***Index Terms*— Botnet, P2P, intrusion detection, network security, botmaster.**

I. INTRODUCTION

A BOTNET is a collection of compromised hosts that are remotely controlled by an attacker 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. In a centralized architecture, all bots in a botnet contact one or a few C&C servers *owned* by the botmaster. However, a fundamental disadvantage of centralized C&C servers is that they represent a *single point of failure*. In order to overcome this problem, botmasters have recently started to build botnets with a more resilient C&C architecture, using a peer-to-peer (P2P) structure [1] or hybrid P2P/centralized C&C structures [4].Bots belonging to a P2P botnet form an overlay network in which any of the nodes can be used by the botmaster to distribute commands to the other peers or collect information from them. Notable examples of P2P botnets are represented by Nugache [5], Storm [2], Waledac [4], and even Confiker, which has been shown to embed P2P capabilities [3], storm and Waledac are of particular interest because they use particular C&C structures as the primary way to organize their bots. While more complex, and perhaps more costly to manage compared to Centralized botnets, P2P botnet offers higher resiliency against take-down efforts, since even if a significant portion of bots in a P2P botnet are disrupted the remaining bots may still be able to communicate with each other and with the botmaster.

Detecting botnets is of great importance. However, designing an effective P2P-botnet

detection system is faced with several challenges. First, the P2P file-sharing and communication applications, such as Bittorrent, emule, and skype, are very popular and hence C&C traffic of P2P botnets can easily blend into the background P2P traffic. This challenge is further compounded by the fact that a bot-compromised host may exhibit mixed patterns of both legitimate and botnet P2P traffic (e.g., due to the coexistence of a file-sharing P2P application and a P2P bot on the same host). Second, modern botnets tend to use increasingly stealthy ways to perform malicious activities that are extremely hard to be observed in the network traffic. For example, some botnets send spam through large popular webmail services such as Hotmail [6], which is likely transparent to network detectors due to encryption and overlap with legitimate email use patterns. Third, as the volume of network traffic grows rapidly, the deployed detection System is required to process a huge amount of information efficiently.

II. RELATED WORK

In [7], there is a general detection framework that is independent of botnet C&C protocol and structure, and requires no *a priori* knowledge of botnets (such as captured bot binaries and hence the botnet signatures, and C&C server names/addresses).
Like any intrusion/anomaly detection system, BotMiner is not perfect or complete. It is likely that once adversaries know the detection framework and implementation, they might find some ways to evade detection.

In [9], there is a methodology to analyze and mitigate P2P botnets. In a case study, this scheme examines in detail the Storm Worm botnet, the most wide-spread P2P botnet currently propagating in the wild.
This made possible to infiltrate and analyze in-depth the botnet, which allows to estimate the total number of compromised machines. In this there is lack of method to analyze the second-tier computers in detail as well as the ways to identify the operators of the Storm Worm.
In [6], the design and implementation of a novel system called *BotGraph* to detect a new type of botnet spamming attacks targeting major Web email providers.
In [8], there are devise techniques to localize botnet members based on the unique communication patterns arising from their overlay topologies used for command and control.
It do not achieve perfect accuracy, they achieve a low enough false positive rate to be of substantial use, especially when combined with complementary techniques

In [10], it provides an efficient approach for identifying the P2P application traffic through application level signatures.
It not yet implemented how to adapt signatures if new protocol versions are introduced and exploiting other characteristics of data transfers such as communication patterns, timings and traffic volumes to perform application classification.

III. BOTNET DETECTION TECHNIQUES

The botnet detection techniques can be classified into three, namely,

1) Honey pot
2) Passive anomaly analysis and
3) Based on traffic application.

A honey pot [11] is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers.

The passive anomaly based detection is done by monitoring system activity and classifying it as either normal or anomalous. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out of normal system operation. This is as opposed to signature based detection which can only detect attacks for which a signature has previously been created. In order to determine what traffic attack is, the system must be taught to recognize normal system activity PAYL [12] and MCPAD [13] are two anomaly based intrusion detection techniques that reduces the high false positive rate.

Botnet detection techniques based on traffic application classification are usually guided by botnet C&C control protocol e.g. if one is only interested in IRC-based botnets then traffic will be classified into IRC and non-IRC groups.

IV. PROPOSED SYSTEM

The proposed system, presents a novel scalable botnet detection system capable of detecting stealthyP2P botnets. This scheme refers to a stealthy P2P botnet as a P2P botnet whose malicious activities may not be observable in the network traffic. Particularly, it aims to detect stealthy P2P botnet even if P2P botnet traffic is overlapped with traffic generated by legitimate P2P applications running on the same compromised host and achieve high scalability. This system identifies P2P bots within a monitored network by detecting the C&C communication patterns that characterize P2P botnets, regardless of how they perform malicious activities in response to the botmaster's commands. Specifically, it derives *statistical fingerprints* of the P2P communications generated by P2P hosts and leverages them to distinguish between hosts that are part of legitimate P2P networks and P2P bots.

V. SYSTEM DESIGN

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 the systems into two phases. In the first phase, it aims 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 applications. We then analyze the remaining traffic and extract number of statistical features to identify flows generated by P2P clients. 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. It contains following components.

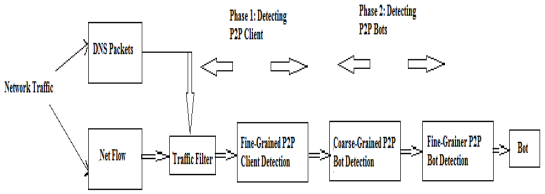

Fig 1: System Overview

A. Identifying P2P Clients

Traffic Filter: The Traffic Filter component aims at filtering out network traffic that is unlikely to be related to P2Pcommunications. This is accomplished by passively analyzing DNS traffic, and identifying network flows whose destination IP addresses were previously *resolved* in DNS responses.

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. In order to detect P2P clients, we first consider the fact that each P2P client frequently exchanges control messages (e.g., ping/pong messages) with other peers. Besides, we notice that the characteristics of these messages, such as the size and frequency of the exchanged packets, are *similar* for nodes in the same P2P network, and *vary* depending on the P2Pprotocol and network in use. As a consequence, if two network flows are generated by the same P2P application and they carry the same type of P2P control messages, they tend to share similar flow size. In addition, a P2P client will exchange control messages with a large number of peers distributed in many different networks. Consequently, the destination IP addresses of network flows that carry these control messages will spread across a large number of networks where each network can be represented by its BGP prefix.

To identify flows corresponding to P2P control messages, we first apply a flow clustering

process intended to group together similar flows for each candidate P2P node *h*. Given sets of flows $S_{tcp}(h)$ and $S_{udp}(h)$, we characterize each flow using a vector of statistical features $v(h) = [Pkt_s , Pkt_r , Byte_s , Byte_r]$, in which Pkt_s and Pkt_r represent the number of packets sent and received, and $Byte_s$ and *Byte* represent the number of bytes sent and received, respectively. 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, $C_j(h)$, represents a group of flows with similar size. For each $C_j(h)$, we consider the set of destination IP addresses related to the flows in the clusters, and for each of this IPs we consider its BGP prefix.

B. Detecting P2P Bots

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.

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 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.

Fine-Grained Detection of P2P Bots: The objective of this component is to identify P2P bots from all persistent P2P Clients. We leverage one feature: the overlap of peers contacted by two P2P bots belonging to the same P2P botnet is much larger than that contacted by two clients in the same legitimate P2P network. Assume that two hosts in the monitored network say h_A and h_B, are running the same legitimate P2P file-sharing application (e.g., Emule). Users of these two P2P clients will most likely have uncorrelated usage patterns. It is reasonable to assume that in the general case the two users will search for and download different content (e.g., different media files or documents) from the P2P network. This translates into a *divergence* between the set of IP addresses contacted by hosts h_A and h_B. The reason is that the two P2P clients will tend to exchange P2P control messages (e.g., ping/pong and search requests) with different sets of peers which "own" the content requested by their users, or peers that are *along the path* towards the content. On the contrary, if h_A and h_B are compromised with P2P bots, one common characteristic of bots is that they need to periodically *search for commands* published by the botmaster. This typically translates into a *convergence* between the set of IPs contacted by h_A and h_B.

We apply hierarchical clustering, and group together hosts according to the distance. In practice the hierarchical clustering algorithm will produce a dendrogram (a tree-like data structure). The dendrogram expresses the "relationship" between hosts. The closer two hosts are, the lower they are connected at in the dendrogram. Two P2P bots in the same botnet should have small distance and thus are connected at lower level (forming a *dense* cluster). In contrast, legitimate P2P applications tend to have large distances and consequently are connected at the upper level. We then classify hosts in *dense* clusters as P2P bots, and discard all other clusters and the related hosts, which we classify as legitimate P2P clients.

VI. SYSTEM IMPLEMENTATION

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, we investigate the global distribution of sub-clusters and further group similar sub-clusters into clusters.

In the current design, we employ *K*-means as the first step clustering. The main reason is that *K*-Means can achieve Bounded time complexity $O(n K I)$, where *K* explicitly indicates the number of expected clusters, *n* is the number of Flows for each host, and *I* is the maximum number of iterations. For the second-step clustering, we use *hierarchical clustering* with

DaviesBouldin validation to group sub-clusters into clusters.

It consists of following 5 modules,

A. Service provider:

In this module, the service provider will browse the data file, initialize the router nodes and then send to the particular receivers. Service provider will send their data file to router and router will select smallest distance path and send to particular receiver.

B. Router

The Router manages a multiple networks to provide data storage service. In network n-number of nodes are present (n1, n2, n3, n4, n5…). In a router service provider can view node details and attacked nodes. Service provider will send their data file to router and router will select smallest distance path and send to particular receiver. If any attacker is found in a node then router will connect to another node and send to particular user.

C. P2P Controller

In this module, the P2P Controller consists of two phases. If Bot is occurs in router then P2P controller is activated. In a first phase DNS packets, Net flow, Traffic filter and Fine-grained P2P client detection are present. Aim is that detecting all hosts within the monitored network that engage in P2P communications. 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. In the second phase, Coarse-grained P2P Bot detection, Fine-grained P2P client detection and Bots are present; our system analyzes the traffic generated by the P2P clients and classifies them into either *legitimate* P2P clients or P2P *bots*.

D. Receiver (End User)

In this module, the receiver can receive the data file from the router. Service provider will send data file to router and router will send to particular receiver. The receivers receive the file by without changing the File Contents. Users may receive particular data files within the network only.

E. Attacker

Attacker is one who is injecting malicious data to the corresponding node and also attacker will change the bandwidth of the particular node. The attacker can inject fake bandwidth to the particular node. After attacking the nodes, bandwidth will changed in a router.

VII. CONCLUSION

Many challenges remain open in the detection of botnet. One main issue is bots have become increasingly sophisticated, so evasion techniques have been developed to deceive detection mechanisms allowing botnets to have long operating times. Another challenge for researchers is the difficulty of testing their proposals in a real scenario or using real data.
This paper, presents a novel botnet detection system that is able to identify stealthy P2P botnets, whose malicious activities may not be observable. To accomplish this task, it derive statistical fingerprints of the P2P communications to first detect P2P clients and further distinguish between those that are part of legitimate P2P networks and P2P bots. It also identifies the performance bottleneck of the system and optimizes its scalability. The proposed system accomplishes high accuracy on detecting stealthy p2p bots.

REFERENCE

[1] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, “Analysis of the storm and nugachetrojans: 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.

[3] P. Porras, H. Saidi, and V. Yegneswaran. (2009). *Conficker C Analysis* [Online]. Available: http://mtc.sri.com/Conficker/addendumC/index.html

[4] 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.

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

[6] 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.

[7] 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.

[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] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm," in *Proc. USENIX LEET*, 2008, pp. 1–9.

[10] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th ACM Int. Conf. WWW*, 2004, pp. 512–521.

[11] Honey net webpage http://www.honeynet.org/project

[12] K. Wang, S. Stolfo, "Anomalous payload-based network intrusion detection", in: *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, (Sophia Antipolis, France, 2004).

[13] Perdisci, Roberto; DavideAriu, PrahladFogla, Giorgio Giacinto, and Wenke Lee. "McPAD : A Multiple Classifier System for Accurate Payload-based Anomaly Detection". *Computer Networks, Special Issue on Traffic Classification and Its Applications to Modern Networks* **5** (6) (2009): 864–881.