Hidden Bot Detection by Tracing Non-human Generated Traffic at the Zombie Host

Jonghoon Kwon, Jehyun Lee, and Heejo Lee

Division of Computer and Communication Engineering, Korea University
{signalnine, arondit, heejo}@korea.ac.kr

Abstract. Defeating botnet is the key to secure Internet. A lot of cyber attacks are launched by botnets including DDoS, spamming, click frauds and information thefts. Despite of numerous methods have been proposed to detect botnets, botnet detection is still a challenging issue, as adversaries are constantly improving bots to write them stealthier. Existing anomaly-based detection mechanisms, particularly network-based approaches, are not sufficient to defend sophisticated botnets since they are too heavy or generate non-negligible amount of false alarms. As well, tracing attack sources is hardly achieved by existing mechanisms due to the pervasive use of source concealment techniques, such as an IP spoofing and a malicious proxy. In this paper, we propose a host-based mechanism to detect bots at the attack source. We monitor non-human generated attack traffics and trace their corresponding processes. The proposed mechanism effectively detects malicious bots irrespective of their structural characteristics. It can protect networks and system resources by shutting down attack traffics at the attack source. We evaluate our mechanism with eight real-life bot codes that have distinctive architectures, protocols and attack modules. In experimental results, our mechanism effectively detects bot processes in around one second after launching flood attacks or sending spam mails, while no false alarm is generated.

1 Introduction

A botnet is a network of infected machines intended to commit malicious activities. They have been used for many kinds of cyber crimes, such as distributed denial-of-service attacks, mass spamming, click frauds and sensitive information thefts [1] [2]. Once a machine is infected, it participates in a botnet to launch malicious attacks and to create other victims without users' awareness: It may lead legitimate users to potential attackers.

Given the importance of the problem, many network-based botnet detection mechanisms have been proposed to thwart botnet threats. Even though the mechanisms are helpful, botnet still run rampant. While the detection mechanisms are improving, the bot writers are constantly developing their bots stealthier to hide abnormal behaviors. Early botnets normally construct centralized structures using an IRC protocol to receive commands from an owner of the botnet, i.e., botmaster. Recent botnets, however, are becoming more intelligent with various structural features: various protocols (HTTP [3], peer-to-peer [4]

and customized protocol [5]) and architectures (centralized, decentralized, and hybrid structures). Moreover, dividing a botnet into several small groups with multiple C&C (Command and Control) servers and encrypting their communications make it difficult to detect the botnets as well. Particularly, the mechanisms are hard to trace an origin of compromised machines, since the botnets can hide their locations by utilizing IP spoofing, malicious proxy server and NAT-boxes.

In this paper, we propose a host-based bot detection mechanism. Our mechanism focuses on bot attacks (e.g., DDoS attacks and spam mailing) since they are the common behaviors of botnets irrespective of bot types, sizes, etc. Thus we detect the origin of attacks, particularly, the bot processes; The proposed mechanism provides network and system resource protection immediately by shutting down malicious traffics at the attack source. The contribution of our mechanism is threefold: 1) high detection accuracy, 2) attack origin trace, and 3) early detection.

We first define user interactions and bot attacks to distinguish bots and benign processes. The user interactions are communications that include requests by human and progress reports by computer. The interactions decide whether a task is performed by human requests. Then, we monitor the user interactions and network traffics to detect bot attacks in real time. If attack traffics occur in a host without any user interaction, we regard it as a bot attack and trace an involved process. Finally, we find the bot process by analyzing correlations between API calls and attack traffic.

In experiments, we evaluate the efficiency of our basic concept, attack analysis, and bot detection with the prototype of our detection mechanism. The experiments perform with respect to eight real-life bot codes that have distinct architectures, protocols and attack modules. As the results, the bots are accurately detected in around one second after launching attacks. Also, our mechanism detects real bots irrespective of their structural characteristics in a timely manner.

2 Background

In this section, we will introduce previous studies to detect botnets based on network-based, and host-based approaches. The motivation of our research will be discussed in regard with botnet improvements.

2.1 Related Work

Network-based Approaches Several researchers have proposed network-based mechanisms with different approaches. Binkley *et al* . [6] propose an anomaly-based botnet detection algorithm, which combines IRC statistics and TCP work weight. Rishi [7] uses a signature-based botnet detection scheme with the similarity of IRC bot nickname patterns. Even these works are useful for IRC-based botnet detection, it is difficult to adopt other protocols. Zhuang *et al.* [8] develop a mechanism to gain botnet membership using traces of spam emails. They investigate bots participating from Hotmail services using spam campaigns. This

study is also useful to detect and estimate bots. However, The mechanisms cannot decide whether attack sources are correct due to the host concealments.

Beside, meaningful approaches have proposed such as BotHunter, BotSniffer and BotMiner. BotHunter [9] models a botnet infection dialog model with high level abstract. It then detects botnets using IDS-driven dialog correlation according to the bot infection dialog model. BotSniffer [10] utilizes a detection method referred to spatial-temporal correlation. It assumes that all botnets tend to communicate with a highly synchronized fashion. BotMiner [11] presents a botnet detection mechanism which clusters botnet communication and activity traffic. It applies clustering algorithms and performs cross-plane correlation to detect botnets. In spite of the improvements of detection mechanisms, they have limitations. The mechanisms need a long monitoring time and unforged large-scale data to detect abnormal behaviors; however, real botnets communicates silently, divides into several small groups, and forges their information.

Host-based approaches BotSwat [12] is one of the impressive researches. It traces all input data through networks and user inputs using a taint propagation trace technique to uncover botnet commands. This work is designed to detect botnets irrespective of their architectures. Nevertheless, it has limitations such as false alarms and high system overheads due to the taint propagation trace. BotTracer [13] detects three phases of botnets with the assistance of the virtual machine techniques. It also has false alarms, since the three phases of bot-like activities can be occurred by benign processes as well.

KolBitsch *et al.* [14] propose a malware detection method that uses behavior graphs. They redefine a malware API call sequence as a behavior graph and detect metamorphic malwares with high accuracy. However, it cannot work when malwares change their entire behavior sequences.

Not-a-bot [15] guarantees user availability from bot attacks. It distinguishes user requests from automatically generated request using Trusted Platform Module (TPM) and offers attesters to user requests. Although it can certify user availabilities, we cannot assume that other requests without attesters are malicious behaviors. Also, the attesters can be abused another malicious attack such as DDoS attacks with fake attesters.

2.2 Motivation

In spite of the numerous research efforts, botnet problems are most significant security issues, because bot authors constantly improve their bot codes to evade detection mechanisms. Major improvements of botnet technologies consists of following features.

- ***Protocol changes*** - HTTP and P2P-based botnets take possessions of traditional IRC botnets [3]. Moreover, customized protocols and hybrid types are adopted as alternative protocols. These changes offer the chance to evade botnet detection methods depending on specific protocol characteristics.
- ***Communication encryption*** - Current botnets basically adopt encryption techniques to their communications. Encryption can keep the botnets safe by evading communication analysis.

- ***Intermittent communications*** - According to Botlab [5], recent botnets communicate with their C&C once for two days on average. Especially, Rustock contacts to C&C only once for 164 days. Such intermittent communication patterns also make the botnets difficult to be revealed. In the worst case, adversaries can exclude their communications by hardcoding the commands in a bot binary.
- ***Botnet subgrouping*** - Adversaries can divide an entire botnet into small groups using multiple C&C servers. Even defenders reveal the subset of botnet, other groups of a botnet are still available.
- ***Source concealment*** - A recent report represents many of botnets spoof IP source addresses to hide their actual locations. Not only the IP spoofing techniques, but also malicious proxy servers and NAT-boxes can evade backtracking of their real locations.

The botnet phenomenon is getting harder to detect. Especially, the previous researches are difficult to provide expeditious responses against botnet threats, since they commonly need a long monitoring time, wide monitoring area and unforged source information. We thus propose a host-based botnet detection scheme to thwart the intelligent botnets with next three basic considerations.

- ***High detection accuracy*** - Botnets are continuously changing their behavior features. We thus have to use constant and general features to improve detection accuracy.
- ***Attack origin trace*** - In current network infrastructures, botnets can easily hide their actual locations. It may lead not only inefficient responses, but also wrong responses. We have to trace real locations of the bots.
- ***Early detection*** - Botnets can bring huge damages just in few seconds. Therefore, the early detection is one of the major considerations in botnet detection.

In the next section, we analyze and arrange the characteristics of bots for detecting bots in a host machine.

3 Basic concept

Two properties can distinguish between bots and benign processes in a host machine. The first property is whether or not a behavior is intended by a user, and the second is whether or not the behavior attempts malicious attacks. We address more details of the properties in this section.

3.1 User Interaction

A bot is designed to serve its master. Once infected with a bot, the infected host works according to the masters' commands and tries to hide the infection from legitimate users; Most bots do not need any interactive functionality with a user to avert the user suspicion. Bayer *et al.* [16] describe that common malicious code operates without the user intention. About 33.26% of codes has

a Graphical User Interface, and 97.8% of this consist of simple message boxes to deflect users' doubt. This analysis shows that bots operate without any user intention; they can be classified as "non-human generated". We define such an interactive relationship between human and host as "User Interaction" that can be classified as follows:

- ***Request Interaction*** ***(UI_{RQ})*** : When humans use a computer, they execute various tasks using physical input devices such as a keyboard, a mouse, a touch screen and a microphone. The physical input devices deliver user requests to each programs by generating predefined input interrupt events. Such request activities by human can be classified as request interaction. Our mechanism considers keyboard and mouse events as request interactions.
- ***Report Interaction*** ***(UI_{RP})*** : Any event that informs the program status to the user are classified as a report interaction. Common programs periodically report their information such as a task progress which is caused, even resolved, by a user request. In this paper, we define several Windows events that are delivered to GUI as report interactions.

Table 1. Program classification by user interaction

UI_{RQ}	UI_{RP}	Classification	Related Activity	Dangerous
O	O	User Interactive Service	Web surfing Document work Sending mail Playing game Multimedia	Low
O	X	Triggered Service	History logging	Medium
X	O	Report Service	Auto update Reserved work	Low
X	X	Background Service	Launching DDoS Spamming Information theft	High

We classify the system behaviors based on user interaction. Table 1 shows our classification results. Specific interrelation between a user interaction and a program will be discussed in section 5.1. We use the user interaction methods as a first criterion to trace the bots.

3.2 Bot Attacks

The second property to distinguish between bots and normal is whether its task lies on maliciousness. We define bot attacks and find bot processes that launch the attacks. Several researches report major bot attacks; TrendMicro [17], CISCO [18] and Liu *et al.* [19] present that the major threats involving bots are performing DDoS and mass spamming. Symantec [3] and ArborNetworks [20]

report that over 90% of DDoS attacks and spam mail are generated by botnets. Hence, we focus on the attacks as major bot attacks. We discuss each attack characteristic and detection approach as follows:

- ***DDoS attacks*** : A DDoS attack is an attempt to make a target system or network resource unavailable to legitimate users. DDoS attacks can be performed with various protocols such as TCP, UDP, ICMP, ARP flood. ArborNetwork [20] present that DDoS attacks pose the largest operational problems amongst the threats on the Internet. Such DDoS attacks have a distinctive feature that can be observed in a host machine. It is the destruction of packet symmetry between request and reply packets by flooding or IP spoofing. Nevertheless, common legitimate network traffic keeps packet symmetry. Therefore, we decide to use the analysis of packet symmetry as a second detection criterion.
- ***Spam emails*** : According to some estimates [3] [21], botnets are responsible for the distribution of approximately 90% of all spam emails. Moreover, a considerable number of recent bots has been propagated using spam relay. Symantec shows that about 30% of malicious codes is propagated by spam email attachments, and it ranks as the second popular propagation method. Botlab [5] also estimates spamming statistics with the vast quantities of spam emails that bots transmit. It shows that bots can generate spam emails continuously. Srizbi and MegaD bots send out more than 1,500 messages per minute, Grum, Kraken and Pushdo bots also generate about 300 spam emails per minute. Despite the slow and constant rate of spam emails sending by Rustock and Storm bots, the quantity of their spam email seems to be massive, since the average rate of a legitimate user is only three per day. We monitor the quantity and periodicity of the mail traffic, which is generated without user interactions, to detect spam relay.

4 Design Overview

In this section, we explain the details of our detection mechanism. Our detection system implemented as a prototype can be divided as two parts; information gathering and bot detection. Information gathering consist of three modules: *Network Monitor (M1), API Monitor (M2)* and *User Interaction Monitor (M3)*. Bot detection part that organized with two modules: *Malicious Network Activity Analyzer (M4)* and *Related Host Activity Analyzer (M5)*, analyzes monitoring results for detecting bot attacks and processes. Fig. 1 presents our algorithm with the five modules.

- ***M1 (Network Monitor)*** : This module records all communication tendencies in accordance with the features of a packet P, occurrence time T and process ID PID to a packet information table PT. When a new packet is observed in the host, we first extract features of the packet and occurrence time. The features consist of five tuples: source address, source port, destination address, destination port and protocol (according to the incoming or outgoing, P can be presented as $\check{P}$ and $\hat{P}$). After that, we find P from PT. If P already exists in PT, the packet count in time slot T_n is incremented by one. If not, we

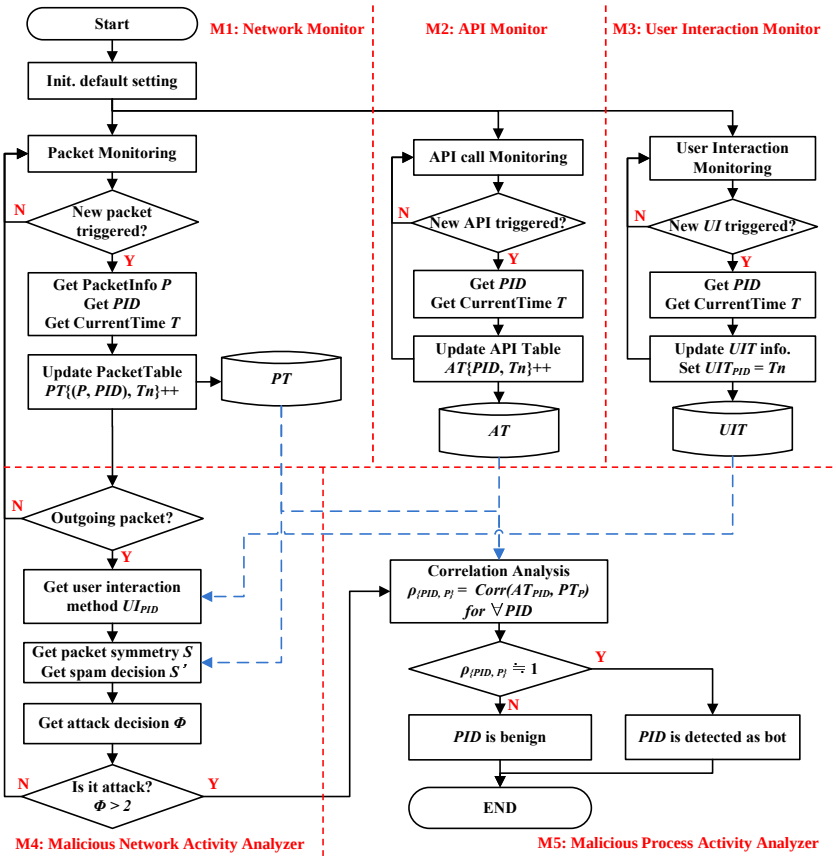


Fig. 1. Bot attack and process detection algorithm.

arrange a new slot for P with the time slot and set the packet count to one. PID is optional information for user interaction check. A sample constitution of PT is illustrated as follows.

$$PT = \begin{array}{c} \\ \check{P}_1, (PID_1) \\ \hat{P}_1, (PID_1) \\ \check{P}_2, (PID_2) \\ \dots \\ \hat{P}_N, (PID_N) \end{array} \begin{array}{c} T_1\ T_2\ \dots\ T_i \\ \left[\begin{array}{c} 3,\ 4, \dots, 1 \\ 4,\ 5, \dots, 1 \\ 17, 6, \dots, 9 \\ \dots \\ 4,\ 4, \dots,\ 5 \end{array}\right. \end{array}$$

- ***M2 (API Monitor)*** : We monitor network API function calls by means of an API hooking technique to extract internal activities related with attack traffic. Foremost, we inject a hook DLL to all running processes in the host, using a System-wide Windows Hook technique. Meanwhile, API addresses on the Import Address Table (IAT) are overwritten; thus, the API calls are redirected to our hook DLL. The hook DLL logs API types, times and PID, and redirects

to original API calls. Monitored API functions which have responsibility for network traffic are chosen by extracting the MSDN library. As a results, 37 API calls are selected, such as *send()*, *sendto()*, *InternetConnction()*, *InternetWriteFile()* and *HttpSendRequestEx()*. When the API monitors are triggered, API call counts in the API information table AT are updated with PID and T. AT is referred by *M5* to reveal bot processes which generate attack traffic.

- ***M3 (User Interaction Monitor)*** : User interactions for each process are monitored. We hook the I/O events for the request interaction and Windows events for the report interaction. Furthermore, we logged the latest time of the event occurrence to the user interaction table UIT with PID. UIT information provide a decision whether or not a packet is generated by a user in *M4*.

- ***M4 (Malicious Network Activity Analyzer)*** : This provides an analysis to detect bot attacks. When an outgoing packet is generated, it runs first analysis to determine whether the packet was generated by human requests. At this point, we have to answer that; how long time interval between user interaction and packet generation can be regarded as a valid time interval for human intention? When a task starts after 10 seconds from a user request, we cannot be sure that the task is intended by a user. We define $\Delta t = 1sec.$ as the valid time interval for two reasons: 1) experience shows that current computing performance completes all user requests within a few micro seconds. 2) common programs are designed to report their states every second. Therefore, we define the user interaction coefficient, as follows:

$$UI_P = \begin{cases} 1 & \text{if CurrentTime-LastUITime} < \Delta t \\ 0 & \text{Otherwise.} \end{cases}$$

The second analysis is an attack decision. Since we focused on the major bot attacks such as DDoS and spam, we first define a packet symmetry test to analyze packet flooding. Kreibich *et al.* [22] introduced the packet symmetry metric with $\ln(\frac{Outgoing\ packets+1}{Incoming\ packets+1})$. However, the metric can exaggerate the packet symmetry for lower-rate transmission by +1 operation. Moreover, if the monitoring time is too short, the metric leads miscalculation due to the variation of response time. To solve the problems, we construct a new packet symmetry metric with simple modification. A following equation represents the packet symmetry metric S.

$$S = \ln[\frac{max(\sum_{i=n}^{m} PT\{\hat{P}, T_i\} \times UI_P, 1)}{max(\sum_{i=n}^{m} PT\{\check{P}, T_i\}, 1)}]$$

From S, if flood packets are generated, $PT\{\hat{P}, T_i\}$ will be rapidly increased and $PT\{\check{P}, T_i\}$ will be stable. S therefore will increase with logarithmic scale. Conversely, S will become zero in the case of perfectly balanced traffic. Based on this difference, we determine the threshold of packet symmetry as 2. It corresponds to an nearly 8:1 (outgoing:incoming) ratio of packet symmetry within the time unit n to m. Such a liberal ratio prevents the false positive caused by huge data transmission.

Another attack decision test is for spam packets. In order to send a single mail to a single victim, a system has to transmit at least eight packets: *SYN, EHLO, AUTH LOGIN, MAIL FROM, RCPT TO, DATA, DATA Fragment*,

and *QUIT*. If the packets are transmitted without user interaction, it means that unrequested mail is delivered to someone. We define a spam decision metric S' as follows:

$$S' = \ln[max(\sum_{i=m}^{n} PT\{\hat{P}, T_i\} \times UI_P, 1)]$$

Finally, based on the two attack decision tests, we derive Φ for bot attack decision. If Φ is greater than 2, the packet will be classified as a bot attack.

$$\Phi = \begin{cases} S' & \text{if } P.dport = 25 \\ S & \text{Otherwise.} \end{cases}$$

- ***M5 (Related Process Activity Analyzer)*** : When a packet is identified as an attack in *M4*, we have to determine its responsible bot process. *M5* conducts bot process detection through analysis of host activities. An easy and simple way to trace a process generating packets is the use of the port binding table. Unfortunately, malicious processes are easily able to hide their port binding information in many cases. Bayer *et al.* [16] state that only 1.88% of malwares binds ports, even though 45.74% of malwares uses TCP traffic and 27.34% uses UDP traffic. Consequently, we have to find another way for process tracing. In our work, an analysis of correlation between the malicious traffic and the APIs performs bot process detection. We use the Pearson Correlation Coefficient. Pearson's equation is represented as:

$$\rho_{X,Y} = corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

X, Y are described to be:

$$X = AT\{PID, T_n\}, \quad Y = PT\{(P, PID), T_n\}$$

The correlation result is a real number in [-1, 1], and it means as follows:

$$\rho_{X,Y} = \begin{cases} 1 & \text{positive correlated} \\ 0 & \text{uncorrelated} \\ -1 & \text{negative correlated} \end{cases}$$

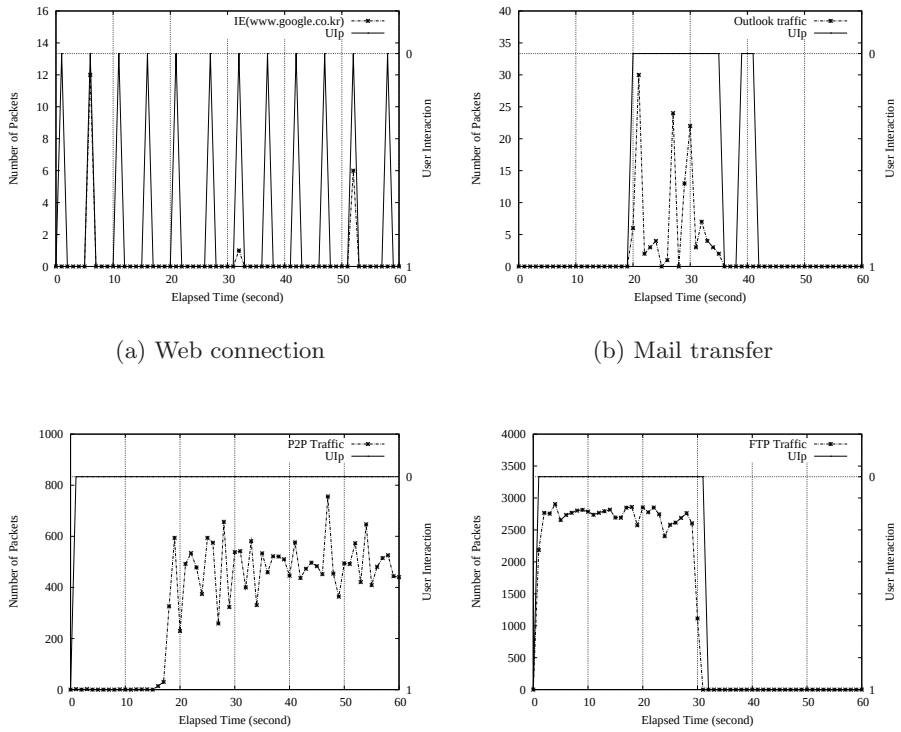
The correlation result tends towards one, when X and Y are intensely related.

5 Experimental Results

In order to evaluate an effectiveness of our approach, we perform a series of experiments with respect to real bot samples. In the first experiment, we show an efficiency of our basic concept. The second experiment shows detection results for various bot attacks. At last, we evaluate an efficiency of our mechanism with eight real-world bot codes which have distinct structures, protocols and attack modules. All experiments are based on the Windows XP machines with a 2.8GHz i5 CPU, 4GB main memory and 100Mbps bandwidth.

5.1 Evaluation for User Interaction

We first observe user interactions by operating bots and benign processes to analyze relationships between user and processes. In particular, benign processes that offered automated network services are evaluated in comparison to the bots. After execution, generated traffic and user interactions are traced and logged without any external interferences. Fig. 2 represents the results of benign activities generated by normal processes. Fig. 2(a) shows a Web page loading through IE (Internet Explorer). The IE periodically generates reconnect messages to gain new information such as real-time topics top 10, and the user interaction as well. Fig. 2(b) is the observation results for the mail traffic automatically generated by the Outlook Express. The traffic is transmitted by the resolved send/receive function. As we can see, the Outlook Express also generates the user interactions to report task progresses, despite the user not being interested in. Fig. 2(c) and 2(d) illustrate mass size file transfers based on FTP and P2P. They also generate user interactions during the tasks.



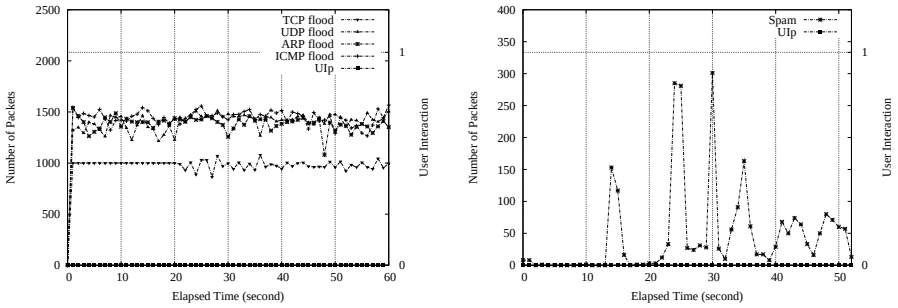
(a) Web connection

(b) Mail transfer

(c) FTP file transfer

(d) P2P file transfer

Fig. 2. User interaction related to benign activities



(a) DDoS attacks (b) Spam attack

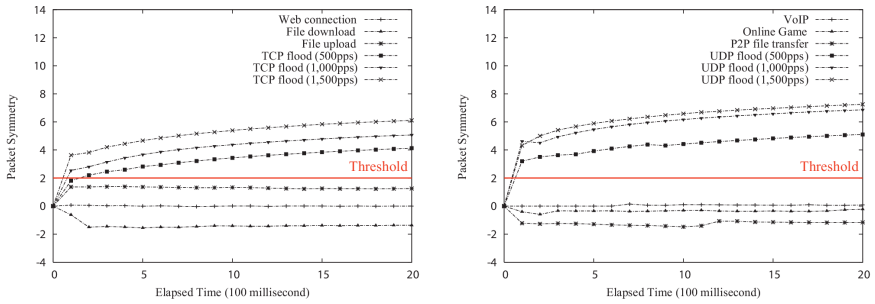
Fig. 3. User interaction related to bot activities

Fig. 3 indicates user interactions and traffic when specific attacks are launched by bots. The first one shows results of various flood attack cases, and second one shows spam email attack. When the bots launch the attacks, we cannot observed any user interactions.

5.2 Bot Attack Detection

We decide to verify a performance of the attack decision method. We perform TCP, UDP, ICMP and ARP flooding three times each, and also generate various benign traffic to compare with the attacks. Fig. 4 indicates the distribution of the packet symmetry S for the each experiments during 20 time units.

– ***TCP flood*** : In case of normal traffic, all packets show reasonable symmetry rates in their communications. Although the packets for file transfer show



(a) TCP symmetry (b) UDP symmetry

Fig. 4. Cumulative distribution of packet symmetry S

apprehensive rates, their distribution constantly reside in a normal range. Nevertheless, all S of the TCP flood packets exceed our threshold 2 within brief time, and continuously increase.

- ***UDP flood*** : Despite UDP not guaranteeing packet symmetry basically, common network services preserve UDP packet equilibrium. We can confirm this for VoIP services, online-games, P2P file sharing services and DNS queries. In experience, all UDP based services have S converging to zero. However, in the case of UDP flood packets are all exceed the threshold.
- ***ICMP and ARP flood*** : ICMP and ARP flood cases also clearly confirm our assumption. All normal request packets can receive a corresponding reply packet, whereas flood packets cannot.

As we can see, benign packets and flood packets show completely polarized results. According to the results, we can classify the flood attacks irrespective of volumes and types without false alarm. The attack detection results prove that our packet symmetry threshold 2 works well.

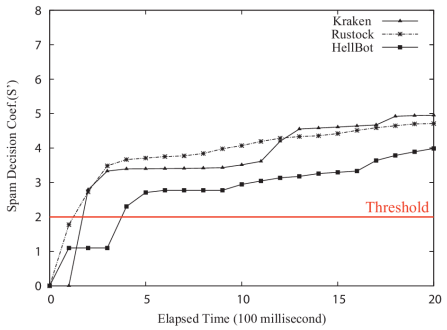
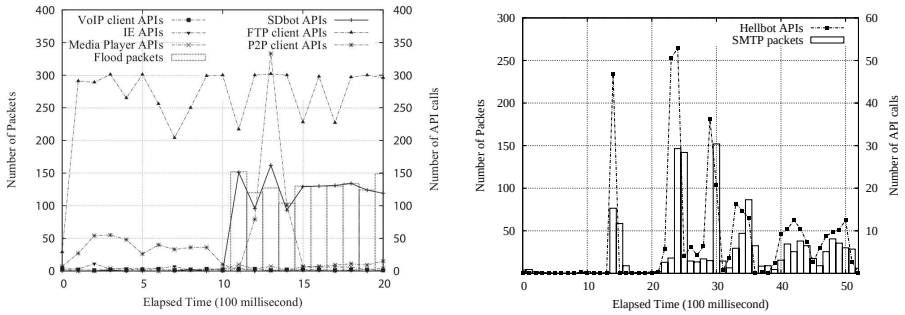


Fig. 5. Cumulative distribution of spam attack decision S'

Fig. 5. represents the attack decision results S' for spam email packets. Bots continuously connect to a large number of mail servers and tries to transfer a huge number of spam emails. From this, the bots generate the hundred of SMTP packets every second. There is no human intention, of course. Consequently, S' immediately exceed the attack decision threshold 2, and we can determine the packets as spam email attacks.

5.3 Bot Process Detection

Our detection scheme for bot process detection is based on the correlation between API calls and attack packets pattern. This approach can lead incorrect results according to the point of view of analysis time. If we only considers time duration after launching attacks, there exist potential false positives such as processes that generate mass traffic in same time period. Therefore, we decide to analyze not only a time period during the attack, but also a period before



(a) Flood packets (b) SMTP packet

Fig. 6. Correlation between attack packets and API calls

the attack commenced. Fig. 6(a) shows that the number of flood packets for two seconds (before and after one second for each) and the number of API calls generated from various processes at the same time.

The SDbot exhibits API call patterns that highly correspond to flood packets, while other processes do not. The datasets for detection are the number of API calls and packets recorded every 100 ms: The range of time unit can be one second, 100 ms and 10 ms. We determine 100 ms to be the optimal value based on experience, since 10 ms is too short to guarantee the relationship between a API call and packet due to processing time. One second is also not an appropriate time unit, because we need a long observation period for correlation analysis. From this, we use total 20 time units (two seconds) for correlation analysis and trigger the analysis in one second after the attack perception. Table 2 shows that the results of correlation analysis. The SDbot is detected with high correlation and the other processes are accurately classified as benign processes. Fig. 6(b) shows that SMTP traffic and the spamming bots' APIs. The spam attack also shows high correspondence, despite the number of API calls being much greater than the number of packets. The Hellbot is detected with a correlation result 0.990812.

Table 3 shows that the final detection results. In our experience, the bots are accurately detected in around one second after launching attacks. The results show that our mechanism can detect bots irrespective of their architecture, protocols, and attack types in reasonable time.

Table 2. Correlation analysis results

Process	$\rho_{PID,P}$	Result	Process	$\rho_{PID,P}$	Result
SDbot	0.926548	Malicious	VoIP client	-0.06426	Normal
FTP client	0.191231	Normal	Internet Explorer	-0.16704	Normal
P2P client	0.022686	Normal	Media player	0.315773	Normal

Table 3. Bot process detection results

Bot type	Structure	Protocol	Attack Type	$\rho_{PID,P}$	Result
Kraken	Centralize	HTTP	Spamming	0.837	Malicious
Rustock	Centralize	HTTP	Spamming	0.772	Malicious
Storm	Decentralize	P2P	UDP flood	0.992	Malicious
SDbot	Centralize	IRC	TCP flood	0.926	Malicious
			UDP flood	0.981	Malicious
			ICMP flood	0.901	Malicious
Agobot	Centralize	IRC	TCP flood	0.897	Malicious
			UDP flood	0.933	Malicious
			ICMP flood	0.912	Malicious
Hellbot	Centralize	IRC	TCP flood	0.943	Malicious
			Spamming	0.991	Malicious
Rbot	Centralize	IRC	TCP flood	0.860	Malicious
			UDP flood	0.911	Malicious
			ICMP flood	0.948	Malicious
Xbot	Centralize	IRC	ARP flood	0.812	Malicious

6 Conclusion

Despite the significant research efforts invested to detect bots, it is still challenging, since bots rapidly evolve with new techniques. In this paper, we propose a new effective scheme to detect running bots based on bot attacks and their corresponding activities in the attack source. Our algorithm cannot be easily evaded, because the attack activities are the only way to gain profits using bots. The evaluation shows effectiveness of our mechanism against real bots. Even if some bots bind itself to benign programs and operate according to the programs interaction, attackers are limited on the timing critical functionalities. Future work will focus on applying our approach for other attacks. Our approach can be adapted to detection of various threats based on their attack definition. We also plan to experiment on numerous bots and other malwares.

7 Acknowledgments

This research was sponsored in part by the Seoul R&BD Program(WR080951) and the Mid-career Researcher Program through NRF grant funded by the MEST[2010-0027793]. Additionally, this research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA(NIPA-2010-C1090-1031-0005).

References

1. E. Cooke, F. Jahanian, and D. McPherson : The zombie roundup: Understanding, detecting, and disrupting botnets, in Proceedings of USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet(SRUTI'05), pp. 170-179, Jul. 2005.
2. The Honeynet Project : Know your enemy: Tracking botnets, 2005. http://www.honeynet.org/papers/bots.

3. Symantec : Symantec global internet security threat report, Apr. 2010.
4. J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon : Peer-to-peer botnets: Overview and case study, in Usenix Workshop on Hot Topics in Understanding Botnets(HotBots'07), Apr. 2007.
5. J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy : Studying spamming botnets using botlab, in Proceedings of the 6th USENIX symphsium on Networked System Design and Implementation(NSDI'09), pp. 291-306, Apr. 2009.
6. J. R. Binkley and S. Singh : An algorithm for anomaly-based botnet detection, in The 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet(SRUTI'06), pp. 43-48, Jun. 2006.
7. J. Goebel and T. Holz : Rishi: Identify bot contaminated hosts by irc nickname evaluation, in Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets(HotBots'07), Apr. 2007.
8. L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, and J. D. Tygar : Characterizing botnets from email spam records, in Proceedings of the 6th USENIX symposium on Networked System Design and Implementation(NSDI'09), Apr. 2008.
9. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee :Bothunter: Detecting malware infection through ids-driven dialog correlation, in Proceedings of the 16th USENIX Security Symposium(Security'07), pp. 167-182, Aug. 2007.
10. G. Gu, J. Zhang, andW. Lee :Botsniffer: Detecting botnet command and control channels in network traffic, in Proceedings of the 15th Annual Network and Distributed System Security Symposium(NDSS'08), Feb. 2008.
11. G. Gu, R. Perdisci, J. Zhang, and W. Lee :Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection, in Proceedings of the 17th USENIX Security Symposium(Security'08), pp. 139-154, Jul. 2008.
12. E. Stinson and J. C. Mitchell : Characterizing bots remote control behavior, in Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment(DIMVA'07), pp. 89-108, Jul. 2007.
13. L. Liu, S. Chen, G. Yan, and Z. Zhang :Bottracer: Executionbased bot-like malware detection, in Proceedings of the 11th Information Security Conference(ISC'08), pp. 97-113, Sep. 2008.
14. C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang : Effective and efficient malware detection at the end host, in Proceedings of the 18th USENIX Security Symposium, pp. 351-366, Aug. 2009.
15. R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy :Not-a-bot: Improving service availability in the face of botnet attacks, in Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats(LEET'08), pp. 307-320, Apr. 2008.
16. U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel : A view on current malware behaviors, in USENIX Workshop on Large-Scale Exploits and Emergent Threats(LEET'09), Apr. 2009.
17. Trend Micro : Taxonomy of botnet threats, technical report, Nov. 2006.
18. CISCO : Botnets: The new threat landscape, White Paper, Dec. 2007.
19. J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang : Botnet: Classification, attacks, detection, tracing, and preventive measures, EURASIP Journal on Wireless Communication and Networking, vol. 2009, Article ID 692654, 2009.
20. D. McPherson, R. Dobbins, M. Hollyman, C. Labovitz, and J. Nazario : Worldwide infrastructure security report, 2009.
21. Georgia Tech. Information Security Center :Emerging cyber threats report, 2009.
22. C. Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt. Using packet symmetry to curtail malicious traffic. ACM HotNets: Proceedings from the Fourth Workshop on Hot Topics in Networks, Dec. 2005.