The ISC Int'l Journal of Information Security

January 2018, Volume 10, Number 1 (pp. 55–61)

http://www.isecure-journal.org

BotRevealer: Behavioral Detection of Botnets based on Botnet Life-cycle

Ehsan Khoshhalpour [1], and Hamid Reza Shahriari [1,*]

[1] *Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 7 April 2017
First Revised: 23 July 2017
Last Revised: 8 October 2017
Accepted: 2 August 2017
Published Online: 5 August 2017

Keywords:
Botnet Detection, Botnet Life-cycle, Host-based Intrusion Detection, Heuristic Algorithm.

Abstract

Nowadays, botnets are considered as essential tools for planning serious cyber attacks. Botnets are used to perform various malicious activities such as DDoS attacks and sending spam emails. Different approaches are presented to detect botnets; however most of them may be ineffective when there are only a few infected hosts in monitored network, as they rely on similarity in bots activities to detect the botnet. In this paper, we present a host-based method that can detect individual bot-infected hosts. This approach is based on botnet life-cycle, which includes common symptoms of almost all types of botnet despite their differences. We analyze network activities of each process running on the host and propose some heuristics to distinguish behavioral patterns of bot process from legitimate ones based on statistical features of packet sequences and evaluating an overall security risk for it. To show the effectiveness of the approach, a tool named BotRevealer has been implemented and evaluated using real botnets and several popular applications. The results show that in spite of diversity of botnets, BotRevealer can effectively detect the bot process among other active processes.

1 Introduction

The majority of attacks and malicious activities in the Internet are made by malware. Botnet is a network of malware-infected computers, which is considered as a basic tool to conduct cyber attacks. In fact, botnet is a network of coordinated compromised computers (bots) which are controlled remotely by an attacker (botmaster) through a command and control channel without the knowledge of their owners. Botnets are used to perform various malicious activities such as distributed denial of service attacks, spam, click fraud, scams and hosting phishing sites. That is why today botnets are identified as one of the largest threats to Internet security [1]. Since botmasters use popular protocols such as IRC, HTTP and P2P as their C&C channel, botnet traffic is usually permitted by firewalls. On the other hand, there is a text-based traffic between botmaster and their bots, and it is sometimes encrypted to evade detection. Furthermore, a bot often remains silent until receiving a command from its botmaster to do malicious activities.

Botnet developers are constantly changing their methods to avoid detection and to make the existing detection methods ineffective. Using P2P protocols rather than using IRC protocol and lately leveraging HTTP protocol for C&C channel is an example of this trend.

Various approaches are proposed to detect botnets.

* Corresponding author.

Email address: `shahriari@aut.ac.ir`(H. R. Shahriari)

ISSN: 2008-2045 © 2018 ISC. All rights reserved.

Some methods adapt bot signatures with previously known bad signatures [2], and [3]. These approaches often depend upon the command and control protocol and need access to the content of packet payload or binary package. The approaches may be bypassed if there is encrypted traffic between bot and botmaster or if some changes applied to the bot signature through hiding techniques such as polymorphism techniques. Some other methods detect botnets based on similar or coordinated network activities [4–7]. These methods can be ineffective if there is only a single individual bot in the monitored network. Another set of methods detect botnets relying on one or more bot particular behaviors [8–11]. These methods, however, realize a more effective approach than the two previous set of methods, but with small changes in bot specific behaviors they can be completely inefficient. Additionally, due to the fact that most of the bot specific behaviors belong to attack phase of botnet life-cycle, bot detection is delayed until observation of final malicious activities.

To overcome the limitations of each of the three aforementioned categories, we focus on botnet life-cycle and present a host-based method to detect a single individual bot-infected host based on the bot process activities in different phases of life-cycle. Our method is based on the idea that botnet life-cycle can be considered as a general signature of almost all types of botnets and a bot distinction from other malware. Relying on this general signature, we will be able to detect botnets whether they are known or unknown. We examine our method by a new bot instance to assess its capability in detection of unknown botnets, and get very successful results.

In our work, we collect a sequence of TCP/UDP activities of each process in the monitored host. The traffic is splited into multiple slices based on the arrival time of packets, and then a profile from each slice is extracted which are then used to distinguish bot process from legitimate ones using some heuristics.

The main contributions of this paper are as follows:

(1) A host-based method is proposed to detect individual bot-infected host in its early steps of activation, e.g., in a few hours. This method can also detect encrypted channels.
(2) We propose a protocol/structure independent method which is robust against botnet evading techniques.
(3) We devise five general behavioral patterns of bot traffic and new heuristics to detect these patterns.

The rest of the paper is organized as follows. In Section 2 we study botnet life-cycle as a basic concept for the proposed method. Section 3 describes our proposed approach, named BotRevealer, in detail. The implementation issues and experiment results are discussed in Section 4. Finally, conclusion and future work are presented in Section 5.

2 Botnet Life-cycle

Different botnets despite their differences often do similar steps and actions during their lifetime, which is called the botnet life-cycle. Having better understanding of these steps and bot behaviors in each phase of the botnet life-cycle, we will be able to improve detection accuracy and response to botnet threats.

Studying previous researches [12–14], we describe botnet life-cycle as following seven steps:

(1) Spread and infection,
(2) Secondary injection,
(3) Hiding and securing,
(4) Rallying/bootstrapping,
(5) Command and control,
(6) Attack,
(7) Remove and release.

In spread and infection phase, botmaster tries to maximize his bot army via infecting new hosts using a variety of methods such as propagation in the local network through shared folder, trick the user to run an infected program or to visit malicious web pages. After a successful infection, bot binary often needs to be downloaded and run on the infected host to turning it into a bot. Then bot tries to hide its presence by some actions such as disabling firewall or preventing anti-virus software from being updated. Now bot process tries to connect to its command and control server or peers address, which is hard-coded in bot binary or found through an alternative method. When bot successfully connect to its server or peer, it will be a new member of the botnet . After this the command and control phase will be started. In this phase, bot is ready to receive commands from its botmaster and perform specified actions. Botmaster may have some conversations with his bot to obtain required information about it e.g., OS version. Furthermore, botmaster may command their bots to update their binary to prevent them from being detected or improving their functionality. Botmaster may command his bots to do any malicious activities such as participating in a DDoS attack, sending spam emails, harvesting sensitive information which is known as attack phase. In some situations, botmaster may decide to leave his bot and remove any footprint on the infected host. These operations are known as remove and release phase.

Our proposed approach discovers bot symptoms in

Table 1. Behavioral patterns of bot traffic

Phase	Behavioral Pattern
Rallying	Sending SYN packets periodically to one or more specific IP
	Numerous opened local ports used to connect to one or more specific IP
Command and control	A permanent connection every time the infected host connects to Internet
	Begin the conversation from outside of the host
	Large size or numerous response packets against a small and little number of incoming packet
	A permanent and often idle connection
	Continuous operation without user interaction
	Fast response to incoming packet
	Frequent and similar response packets
	One-way connection drop and existing un-responded packets
Attack	Port scanning
	Packet flooding
	A lot of connection attempts and failures

three different phases: rallying, command and control and the attack phases. The main focus is in two former phases, and therefore, bot presence will be unfolded in earlier stage of the bot life-cycle.

3 Proposed Botnet Detection Method

In Section 1, we mentioned some previous researches that state some particular behavioral patterns of bot traffic. We studied most of these behavioral patterns in rallying, command and control and attack phases of botnet life-cycle and summarized them as mentioned in Table 1. In order to identify these behavioral patterns, we define some general behavioral patterns and their traffic symptoms according to Table 2.

In order to detect defined general behavioral patterns, we collect transition layer traffic of all processes in term of TCP or UDP packets and divide the whole traffic of each process into smaller slices based on inter-arrival time of each two successive packets. In other words, we aim to identify groups of packets, which are related to a specific communicative operation of the process.

Any two successive packets, such that the time difference between their arrival times is less than a threshold value τ, are placed in a single group. Thus, upon observing a new packet, it belongs to previous group, if inter-arrival time between this packet and previous one is less than τ, otherwise it will start a new group.

To get the more accurate results, we need to take a suitable value for τ by which all packets associated with a particular command and control activity, be grouped together. We choose τ value based on some experiences on botnet traffic.

3.1 Profile Extraction

We have to collect information about each group to be analyzed afterwards. Collected information about each group is saved in a profile named P_g. In fact, P_g is a statistical profile of the corresponding group of packets in terms of desired parameters that have been seen in Table 3.

Table 3. Profile parameters for a group of packets (P_g)

Parameter	Description
Index	Group number
Duration	Group duration time
StartPacket	Start packet of the group
ReceiveCount	Number of Receive packets
SendCount	Number of Send packets
Distance	Distance from previous group

3.2 Profile Analysis

In order to analyze the extracted profiles from traffic of a particular process, we need to look for certain statistical evidences in its corresponding data structures. Thus, we are able to identify each general behavioral pattern mentioned in Table 2. To detect these behavioral patterns we define five new heuristics and related threshold variables namely δ_1, δ_2, δ_3, δ_4 and δ_5 to analyze data structures.

Pseudocode 1 depicts simple descriptions of detection rules of patterns. Detection of any of mentioned behaviors causes 20 percent increase in risk of the related process for being bot process.

4 Implementation and Experiments

To evaluate the effectiveness of our approach, we used real bots in a local area networks consisting of some virtual machines. We run three virtual machines as botmaster, C&C server and victim machine. To have a rigorous evaluation, some real bots with some common legitimate applications were run in hosts. We used Spybot as an IRC-based bot, Zeus as an HTTP-based bot and a remote administration tool called NuclearRAT as three malicious processes to evaluate our method using false negative value. We also

Table 2. General behavioral patterns and their traffic symptoms

Phase	General behavioral pattern	Behavior description	Traffic symptoms
Rallying	Try to connect	Try to connect to somewhere outside of the host periodically	Observation of periodic occurrence of "Open" and "Connect" TCP packets without any "Send" or "Receive" packet
Command and Control, Attack	Keep alive Connection (Heart Pulse)	Try to determine connection status and keep it alive periodically	Observation of periodic exchange of "Send"/"Receive" packets
	Remote start	Wait for receiving a command to do some operations	Observation of a "Receive" packet before occurrence of any "Send" packet in a group of packets
	Mostly sending	Generally try to send some information or do some attack	Observation of "Send" packets more than "Receive" packets
	Lightweight connection	Connection is mostly idle until receiving command	Observation of low rate of "Send" and "Receive" packets in a connection

used some popular programs such as Yahoo messenger, Google Talk and Mozilla Firefox to inspect our method for false positive rate. We collected approximately one-hour traffic of each application for our experiment.

We used an off-the-shelf tool called DiamondCS Port Explorer to collect and log TCP/UDP activities of processes. We wrote approximately 500 lines of Perl code to analyze outputs of this tool to categorize TCP/UDP packets, extract profiles and discover general behavioral patterns to detect possible bot processes.

4.1 Threshold Variable Analysis

To achieve the best values for threshold variables, we examined a range of values for these variables. To find the best value, we study the effect of different values on false positive (FP) and false negative (FN) rates. FP value shows the number of detections of legitimate programs as a malicious ones, whereas FN value shows the number of bot processes that were not detected.

As an example, FN and FP values are calculated for different values of δ_1 and are shown in Figure 1. FN and FP values are also calculated for different possible values of δ_2, δ_3, δ_4 and δ_5 variables in the same way. Best value for threshold variables δ_1 to δ_5 are 5, 4, 0.8, 0.8 and 0.2, respectively.

4.2 Detection Accuracy

Evaluation results are shown in Table 7. These results are obtained when the parameters are set as $\tau = 3$, and δ_1 to δ_5 are set to 5, 3, 0.7, 1 and 0.2, respectively. We can see that BotRevealer is successfully able to detect bot processes among other benign applications. Unlike [15], our method shows that it is possible to detect general behaviors of bots in a few hours during

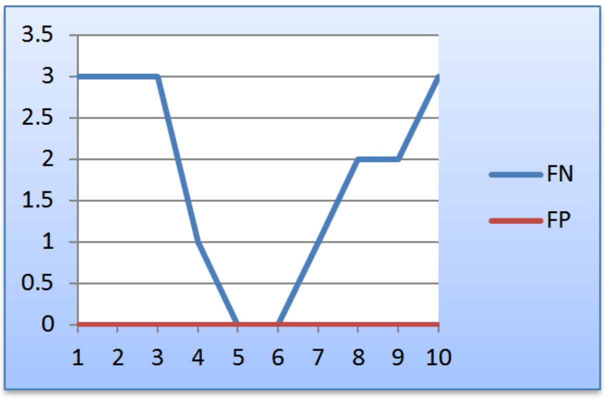

Figure 1. δ_1 value analysis

Table 4. Accuracy Metric

Accuracy Metric	Formulation
Precision	$\frac{N_{tp}}{N_{tp}+N_{fp}}$
Recall	$\frac{N_{tp}}{N_{tp}+N_{fn}}$
F-measure	$\frac{2*Precision*Recall}{Precision+Recall}$

botnet activities.

To determine the accuracy in detection of each general behavior pattern, we calculate some accuracy metrics as shown in Table 4. In this table, N_{TP}, N_{FP} and N_{FN} are the number of true positive, false positive and false negative, respectively. The parameters are calculated based on the best values of each threshold variable and are shown in Table 5. The results of calculation of accuracy parameters for each general behavioral pattern are shown in Table 6. This table shows that BotRevealer is able to distinguish bot processes among other legitimate processes in the operating system.

We test BotRevealer with a new botnet namely Kelihos botnet to evaluate its capability in detection of new bots. This bot is available in a dataset which

Pseudocode 1 General Behavioral Patterns Detection Rules

```
1: function DetectTryToConnect(profiles)
2:     $TC \leftarrow$ all profiles that $SendCount = 0$ and $ReceiveCount = 0$
3:     $D[i] \leftarrow$ number of profiles in $TC$ where $Distance = i$
4:     $M \leftarrow$ maximum value in array $D$
5:     if $M \geq \delta_1$ then return TRY_TO_CONNECT
6:     end if
7: end function
8: function DetectHeartPulse(profiles)
9:     $TC \leftarrow$ all profiles that $SendCount > 0$ and $ReceiveCount > 0$ and $Distance \geq \tau.\delta_2$
10:    $D[i] \leftarrow$ number of profiles in TC where $Distance = i$
11:    $M \leftarrow$ maximum value in array D
12:    if $M \geq \delta_1$ then return HEART_PULSE
13:    end if
14: end function
15: function DetectRemoteStart(profiles)
16:    $TC \leftarrow$ all profiles such that $SendCount > 0$ and $ReceiveCount > 0$ and $Distance \geq \tau.\delta_2$
17:    $P \leftarrow$ number of profiles in $TC$ where $StartPacket =$ "Receive"
18:    $Q \leftarrow$ number of profiles in $TC$ where $StartPacket =$ "Send"
19:    if $P/Q \geq \delta_3$ then return REMOTE_START
20:    end if
21: end function
22: function DetectMostlySending(profiles)
23:    $TC \leftarrow$ all profiles
24:    $S \leftarrow$ sum of $SendCount$ values in $TC$
25:    $R \leftarrow$ sum of $ReceiveCount$ values in $TC$
26:    if $S/R \geq \delta_4$ then return MOSTLY_SENDING
27:    end if
28: end function
29: function DetectLightweightConnection(profiles)
30:    $TC \leftarrow$ all profiles
31:    $S \leftarrow$ sum of $SendCount$ values in $TC$
32:    $R \leftarrow$ sum of $ReceiveCount$ values in $TC$
33:    $T \leftarrow$ sum of $Distance + Duration$ values in $TC$
34:    if $\frac{S+R}{T} \leq \delta_5$ then return LIGHTWEIGHT_CONNECTION
35:    end if
36: end function
```

was created by the CVUT Malware Capture Facility Project [16] and can be downloaded with the name CTU-Malware-Capture-Botnet-25. BotRevealer successfully detected it as a bot and it shows its effectiveness in detecting new bots.

Table 5. NTP, NFP and NFN values

Variable	Value	NFP	NFN	NTP
δ1	5	0	0	3
δ2	4	2	1	2
δ3	0.8	1	0	3
δ4	0.8	0	1	2
δ5	0.2	2	1	2

Table 6. Experimental results for behavioral patterns

Behavioral pattern	Precision	Recall	F-measure
Try to connect	1	1	1
Keep alive connection	0.77	0.87	0.82
Remote start	0.89	1	0.94
Mostly send	1	0.90	0.95
Idle connection	0.77	0.87	0.82

4.3 Comparison with Previous Work

Table 8 provides different capabilities of BotRevealer in comparison with other botnet detection methods. In [17] some detection criteria for comparative analysis of botnet detection techniques are presented, but these criteria are general for all IDS techniques. However, we define new criteria more specific for botnet detection. As demonstrated in this table, our method has some significant features in detecting botnets. The symbol (*) indicates the feature is supported by the method. As seen, BotRevealer discovers botnet via life-cycle as a general signature, detects botnet activities in early stages and finds individual bot-infected host independent of its C&C protocol and content of packets. BotRevealer does not rely on known signatures of bots, but it discovers a general signature of almost all kinds of botnets; therefore, it will be able to detect unknown new bots.

5 Conclusion and Future Work

In this paper, we described the botnet life-cycle as a general signature of almost all types of botnets despite their differences and the bot distinction from other malware. BotRevealer discovers botnet leveraging life-cycle as a general signature, mostly in early stages and detect individual bot-infected host independent of its C&C protocol and content of packets. We show that our method is able to detect general behaviors of bots in a few hours during botnet activities. However, BotRevealer requires running in inspected hosts and collecting network traffic on the host. Thus, it may cause processing and storage overhead in the hosts.

In future, we will try to identify general behavioral

Table 7. Experimental results

Application	Network general behavioral pattern analysis						Result
	pattern 1	pattern 2	pattern 3	pattern 4	pattern 5	Risk Value	
Spybot	*	*	*	*	*	100%	Bot
NuclearRAT	*	*	*	*	-	80%	Bot
Zeus	*	-	*	-	*	60%	Bot
Yahoo Messenger	-	*	-	-	-	20%	-
Google Talk	-	-	*	-	*	40%	-
Skype	-	*	-	-	-	20%	-
Emule	-	-	-	-	-	0%	-
BitTorrent	-	*	*	-	-	40%	-
CuteFTP	-	-	-	-	-	0%	-
Firefox	-	-	-	-	-	0%	-
Opera	-	-	-	-	*	20%	-

Table 8. Detection capability comparison

Detection methods	General signature detection	Individual bot detection	Unknown bot detection	Detection in earlier stage	Encrypted C&C	Protocol/ Structure-Independent
Rishi [2]	-	*	-	*	-	-
BotHunter [3]	*	-	*	*	*	-
BotMiner [4]	-	-	*	-	*	*
BotRevealer	*	*	*	*	*	*

patterns of bots in system calls level and extract system call profiles to analyze these profiles along with network traffic profiles. Furthermore, we will try to take advantage of machine learning techniques to find best threshold values in our method.

References

[1] W. Lee, C. Wang, and D. Dagon, Botnet detection: countering the largest security threat. Springer, 2008.

[2] J. Goebel and T. Holz, “Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation,” in First Workshop on Hot Topics in Understanding Botnets (HotBots’07), 2007.

[3] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, and M. Park, “BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation,” in Proceedings of the 16th USENIX Security Symposium (Security’07), 2007.

[4] 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), 2008.

[5] M. Eslahi, M. Yousefi, M. V. Naseri, Y. Yussof, N. Tahir, and H. Hashim, “Cooperative Network Behaviour Analysis Model for Mobile Botnet Detection,” in IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2016, pp. 107–112.

[6] C. Dietz, A. Sperotto, G. Dreo, and A. Pras, “How to Achieve Early Botnet Detection at the Provider Level?,” in IFIP International Conference on Autonomous Infrastructure, Management and Security, 2016, pp. 142–146.

[7] R. Aryan, and H.R. Shahriari, “Botnet detection based on network behavioral and anomaly detection,” 18th National CSI Computer Conference, Iran (Islamic Republic of), 12-15 March 2013

[8] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, “Exploiting Temporal Persistence to Detect Covert Botnet Channels,” in 12th International Symposium on Recent Advances in Intrusion Detection (RAID’09), Springer Berlin Heidelberg, 2009, pp. 326–345.

[9] G. Fedynyshyn, M. C. Chuah, and G. Tan, “De-

tection and Classification of Different Botnet C&C Channels," in Autonomic and Trusted Computing, Springer Berlin Heidelberg, 2011, pp. 228–242.

[10] L. Cavallaro, C. Kruegel, and G. Vigna, "Mining the Network Behavior of Bots," Tech. Rep. 2009-12, Department of Computer Science, University of California at Santa Barbara (UCSB), CA, USA, 2009. .

[11] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," Computers & Electrical Engineering, vol. 50, pp. 91–101, 2016.

[12] R. A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Analysis of Botnets throuth Life-Cycle," in Proceedings of International Conference on Security and Cryptography (SECRYPT), 2011, pp. 257–262.

[13] N. Hachem, Y. Ben Mustapha, G. G. Granadillo, and H. Debar, "Botnets: Lifecycle and Taxonomy," in Conference on Network and Information Systems Security (SAR-SSI), IEEE, 2011, pp. 1–8.

[14] F. Naseem, U. Sabir, M. Shafqat, and A. Shahzad, "A Survey of Botnet Technology and Detection," International Journal of Video & Image Processing and Network Security (IJVIPNS-IJENS), vol. 10, no. 1, pp. 13–17, 2010.

[15] S. García, V. Uhlír, and M. Rehak, "Identifying and Modeling Botnet C & C Behaviors," in In Proceedings of the 1st International Workshop on Agents and CyberSecurity, ACM, 2014.

[16] "CVUT Malware Capture Facility Project," https://agents.fel.cvut.cz/malware-capture-facility, [Accessed: 10- Oct- 2016] .

[17] R. S. Abdullah, M. F. Abdollah, Z. Azri, M. Noh, M. Zaki, and S. R. Selamat, "Revealing the Criterion on Botnet Detection Technique," IJCSI International Journal of Computer Science, vol. 10, no. 2, pp. 208–215, 2013.

Ehsan Khoshhalpour received the M.S. degree in Information Security in 2013. He received his B.S. degree in Computer Engineering from Shahid Chamran University of Ahvaz in 2009. His research interests include malware recognition and intrusion detection systems.

Hamid Reza Shahriari is currently an assistant professor at the department of computer engineering and information technology at Amirkabir University of Technology. He received his Ph.D. in Computer Engineering from Sharif University of Technology in 2007. His research interests include information security, especially software vulnerability analysis, security in e-commerce, trust and reputation models, and database security.