Towards social botnet behavior detecting in the end host

Article · April 2015

DOI: 10.1109/PADSW.2014.7097824

CITATIONS
6

READS
120

4 authors, including:

Yuede Ji
University of North Texas
22 PUBLICATIONS **191** CITATIONS

SEE PROFILE

Towards Social Botnet Behavior Detecting in the End Host

Yuede Ji, Yukun He, Xinyang Jiang, and Qiang Li
College of Computer Science and Technology, Jilin University, Changchun, China
{jiyd12, heyk12, jiangxy14}@mails.jlu.edu.cn, li_qiang@jlu.edu.cn

***Abstract*—Social botnet utilizing online social network (OSN) as Command and Control channel (C&C) has caused enormous threats to Internet security. Server-side detection approaches mainly target on suspicious accounts, which cannot identify the specific bot hosts or processes. Host-side approaches target on suspicious process behaviors which are not robust enough to face the challenges of frequent variants and novel social bots. In this paper, we propose a novel social bot behavior detecting approach in the end host. Because social bot binaries or source codes are not easy to collect, we first design a novel social botnet, named wbbot, based on Sina Weibo. We analyze it from two aspects, wbbot architecture and wbbot behaviors. Second, we analyze the host behaviors of existing social botnets which come from public websites, other researchers, and our implementations. We identify six critical phases: infection, pre-defined host behaviors, establishment of C&C, receive the commands of botmaster, execution of social bot commands, and return the results. Third, we present our detection system which consists of three components: host behavior monitor, host behavior analyzer, and detection approach. We present behavior tree-based approach to detect social bot. After constructing the suspicious behavior tree, we match it with the template library to generate detection result. Finally, we collect real-world social botnet traces to evaluate the performance. We would like to share them for academic research. The results indicate that our system has an acceptable false positive rate of 29.6% and remarkable false negative rate of 4.5%. However, compared with other detection tools, our detection result is still remarkable.**

Keywords*—*online social network, social botnet, host behavior, botnet traces, tree similarity

I. Introduction

Online social network (OSN) owns a large number of users, facebook has more than 1 billion active users and other 15 OSNs have more than 100 million active users [1]. Many malicious behaviors are hidden in the enormous network flows. Many conventional bots regained their youth through propagating on OSN sites. Zeus botnet which is first detected in 2007 has risen steadily in 2013 through propagating on facebook [2]. A new kind of botnet, named social botnet, utilizes OSN as C&C channel. Social bot runs on user hosts stealthily [3]. It controls user account on a specific OSN site and communicates with the botmaster. In this way, the botmaster can control social bots to conduct various malicious activities, such as spamming, phishing, and privacy theft. Social botnet has great impact on users and OSNs [4].

Social botnet is rather new with its first discovery in 2009 [5]. It is called koobface and targets on most OSN sites, such as Facebook, Twitter, and MySpace. Its infection starts with spams spreading on these OSN sites containing provocative messages with a hyperlink. This link directs the user to a phishing website, then requires the user to install a Flash plugin. This plugin is a downloader of koobface components which will check the Internet cookies and download other appropriate components. With these components, koobface can perform many malicious activities. Another social bot, named Naz bot, is discovered on Twitter in 2009 [6]. Koobface and Naz bot are first examples of social bot, and after that researchers propose other prototypes [3], [7], [8], [9].

According to detection location, existing social bot detection approaches are mainly divided into two categories: server-side and host-side. Server-side detection approaches mainly use classification methods to identify bot accounts [10], [11], [12]. However, they cannot find the specific bot hosts or processes. In host-side detection, Pieter *et al.* propose an approach detecting social botnet communication through monitoring user activity [13]. They suppose that the communication with social media is suspicious if it is not caused by human activity. They measure the causal relationship between network traffic and human activity to filter out social bot processes. Natarajan *et al.* present a detection scheme to detect StegoBot [14]. They analyze different entropies of images to show that images are generally sensitive to embedding. Based on the analysis, they select efficient features to construct the feature set. They further propose an ensemble of classifiers to classify the vulnerable images from social network. Erhan *et al.* analyze Naz bot to unveil the features of social bot and the evolution of social bot C&C mechanisms. Then they propose server-side and host-side detection mechanisms.

Host-side detection approaches can capture some intrinsic and "expensive" features of social bots [15]. We can eliminate social bot processes if we can successfully detect them on end host. However, existing host-side detection approaches still face many challenges. For the causal relationship detection approach, human activities and network traffic is not easy to synchronize. On one hand, the time interval is not easy to quantify because there are many dynamic changing factors, such as network delay, operating system delay, performance of different computers, etc. On the other hand, many advanced social bots will not perform malicious activities until they have monitored human activities. In this way, malicious activities are mixed with benign human activities. These approaches also face the problem of low detection accuracy with new samples or variants. Therefore, designing effective and efficient host-side social bot detection approach has become one of the urgent tasks of researchers.

In this paper, we propose a novel social bot behavior detection approach in the end host. Because social bot binaries

978-1-4799-7615-7/14/$31.00 ©2014 IEEE

or source codes are not easy to collect, we first design a novel social botnet, named wbbot, based on Sina Weibo. We analyze it from two aspects, wbbot architecture and wbbot behaviors. Second, we analyze the host behaviors from existing social bots, and laboratory works of possible social bots. We identify six critical phases based on life cycle: infection, pre-defined host behaviors, establishment of C&C, receive the commands of botmaster, execution of social bot commands, and return the results. Third, we present our detection system, including three components: host behavior monitor, host behavior analyzer, and detection approaches. In behavior tree-based detection approach, after constructing the suspicious behavior tree, we match it with the templates in the library to calculate the similarity and generate the final detection result. Finally, we collect real-world social botnet traces to evaluate the performance of our detection system. We would like to share them for academic research[1]. We enumerate the threshold from 0.05 to 0.95 with an incremental step of 0.05 to find the best value. The detection results indicate that our system has an acceptable FP rate of 29.6% and remarkable FN rate of 4.5%. However, compared with other detection tools, our detection result is still remarkable.

Our work makes the following contributions:

(1) We design a social botnet, named wbbot, based on Sina Weibo. We analyze it from two aspects, wbbot architecture and wbbot behaviors. After that, we analyze host behaviors of existing social botnets which come from public websites, other researchers, and our implementations. We identify six critical phases based on life cycle: infection, pre-defined host behaviors, establishment of C&C, receive the commands of botmaster, execution of social bot commands, and return the results.

(2) We present our detection system which consists of three components: host behavior monitor, host behavior analyzer, and detection approach. In behavior tree-based detection approach, after constructing the suspicious behavior tree, we match it with the templates in the library to calculate the similarity and generate the final detection result.

(3) We collect real-world social botnet traces to evaluate the performance of our detection system. We would like to share them for academic research. The detection results indicate that our system has an acceptable FP rate of 29.6% and remarkable FN rate of 4.5%. However, compared with other detection tools, our detection result is still remarkable.

The rest of this paper is organized as follows. Section II presents related works. Section III presents the design and analysis of wbbot. Section IV presents host behaviors of social bot. Section V presents the methodology of our detection mechanism. Section VI presents experiment details and results. Section VII presents limitation and future works. Section VIII summarizes this paper.

II. Related Work

According to detection location, existing social bot detection are mainly divided into two categories: server-side and host-side. In server-side detection, Tan *et al.* propose an approach detecting spams in user generated contents on social networks [10]. They identify that spammers exhibit unique non-textual patterns, such as posting activities, advertised spam link metrics, and spam hosting behaviors. Based on these nontextual features, they propose an offline detection approach utilizing several classification methods. Then they further propose a runtime spam posts detection approach BARS. Chu *et al.* propose an approach on the classification of human, bot, and cyborg accounts on Twitter [11]. They analyze the collection data of over 500,000 accounts to observe the difference among human, bot, and cyborg in terms of tweeting behavior, tweet content, and account properties. Based on the measurement results, they propose a classification system including four components: an entropy-based component, a spam detection component, an account properties component, and a decision maker. They differentiate a human, bot, or cyborg through combining the four components. Francisco *et al.* propose an approach combining multiple scales of users' interactions within a social network to discriminate humans and bots [12]. NEIGHBORWATCHER[16] takes advantage of spammers based configuration information to infer spam. VoteTrust[17] detects a false Social Network accounts using trust-based polling and voting graph model. It has two graph models which are inviting diagram (directed graph) and accept graph (directed weighted graph).

Pieter *et al.* propose an approach detecting social botnet communication through monitoring user activity [13]. They suppose that the communication with social media is suspicious if it is not caused by human activity. Thus they measure the causal relationship between network traffic and human activity to detect social bot. Many social bots can mimic user activity to evade detection mechanisms like this, in order to get a high detection accuracy the approach has to correlate more behaviors to eliminate the bias. Natarajan *et al.* present a detection scheme to detect StegoBot [14]. They analyze different entropies of images to show that images are generally sensitive to embedding. Based on the analysis, they select efficient features to construct the feature set. They further propose an ensemble of classifiers to classify the vulnerable images from social network. Unlike this approach, our approach focus on the whole set of social bots. Erhan *et al.* analyze Naz bot to unveil the features of social bot and the evolution of social bot C&C mechanisms. Then they propose server-side and host-side detection mechanisms.

Conventional C&C detection approaches are not effective enough because social bot mimics human activity on social network, and their domain name and IP are in the white list. In online detection methods, they are not able to solve the source problem and help users to clean bot clients. Unlike existing host-side detection approaches, our approach can balance many bias with a great number of accurate behaviors.

III. Design and Analysis of Wbbot

To better understand the behaviors of social bots on host, we designed wbbot which acts on Sina Weibo. We will introduce wbbot from two aspects: wbbot architecture and wbbot behaviors.

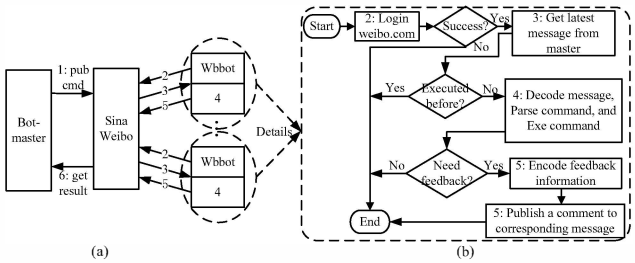
A. Wbbot Architecture

Wbbot master uses Sina Weibo to control wbbots and collect information. Figure 1 represents wbbot architecture, (a)

[1] http://pan.baidu.com/s/1hqvHoSO

represents the botnet architecture, and (b) details wbbot control flow on host. Wbbot first try to use IE to login weibo.com with local user cookies. If failed, it will suspend for a random time. Otherwise, it will try to get the latest messages posted by botmaster. Then, it will check whether the messages have been received before. If they are new messages, it will try to decode them as commands and execute them. If the command needs feedback information, it will encode the feedback information and publish them as a comment to corresponding botmaster message. If the command does not need feedback information, it will wait for another command.

Fig. 1: Architecture of Wbbot ((a) represents the botnet architecture, (b) details wbbot control flow on host)



B. Wbbot Behaviors

TABLE I: Host Commands List

	Command	Description
Host	getNetInfo	get host information (MAC, IP, username, etc.)
	getVersion	get the windows system Version
	exeCmd	execute a DOS command
	timeExeCmd	execute a DOS command at a specific time
	visit	force the IE browser to open an URL
	redirect	rebind the domain and IP
Social Network	pubWeiboText	order wbbot to publish a message
	postComment	order wbbot to comment a message on a user
	addFollowing	order wbbo to follow an account
	autoAddFollowing	order wbbot to automatically follow others

Wbbot behaviors can be classified into host behavior, and social network behavior. Wbbot has similar host behaviors like other bots, it can steal private information or redirect users to malicious web site when they surf on the Internet. Wbbot has six basic commands as shown in Table I. Wbbot can perform some essential social network activities automatically, such as the four basic social commands as shown in Table I.

IV. Host Behaviors of Social Bots

In order to get common host behaviors, we analyze existing social bots, including two samples: koobface [5], and Naz bot [6], [7]; three laboratory works: stegobot [8], the bot designed by Boshmaf [3], and facebot [9]. We divide their behaviors into six different phases based on life cycle as shown in Table II. Then we analyze each phase in the following subsections.

A. Phase 1: Infection

Conventional botnet has many proven infection mechanisms, such as malicious urls with an email, unwanted malware downloading, installing cracked softwares, and infected removable disks [18], [19]. Besides these, social bots also rely on

TABLE II: Host Behaviors of Social Botnet

Social bot	Koobface	Naz bot	stegobot	bot designed by Boshmaf	facebot
1 Infection	provocative spam message with a hyperlink on OSN	existing mechanisms	malicious urls in normal emails	existing mechanisms	existing mechanisms
2 Pre-defined host behaviors	check the Internet cookies	-	collect email address and passwords, or credit card number or log all keystrokes	-	steal sensitive information, like password
3 Establishment of C&C	connect to C&C through HTTP server	visit RSS of some specific user accounts (upd43's RSS)	use image steganography to share images	connect to botmaster server through HTTP	image steganography
4 Receive commands of botmaster	botmaster sends encrypted messages to bots through HTTP	RSS returns Base64-encoded messages	download several recent images of botmaster	-	-
5 Execution of social bot commands	decrypt the messages, download other components and execute them	decrypt the text, visit urls in the text, downloads malicious file and execute it	several commands, such as receive search keywords from the botmaster, respond with matches from the filesystem	most commands target on OSN sites, including read, write, connect, disconnect, etc.	-
6 Return the results	different components connect with different C&C server and execute different commands, including spreading koobface url	steal user information and send them to the server controlled by botmaster through HTTP	upload images with hidden information	return the information to botmaster using HTTP	bot and botmaster join the same facebook group, botmaster scans the profile image of every new member

OSN sites to propagate. As shown in Table II, koobface controls user accounts to send spams containing a catchy message with a malicious url [5]. Because of the specific feature of OSN sites, all the friends of the user will receive this message. In this way, koobface can reach a very high propagation speed. Stegobot utilizes a more advanced mechanism [8]. The first way is writing email lures to deliver bots combined with the use of email communication networks. A more effective way is to replay a stolen email containing an attachment of a malicious payload to reply to a friend. Other three bots utilize existing mechanisms.

We summarize the infection behaviors in Table III. We use A to denote the set of infection ways and $A = \{A[1], A[2], A[3]\}$. $A[1]$ denotes the browser behavior of downloading suspicious binaries, for example, the fake flash plugin of koobface. $A[2]$ denotes downloading binary attachment of emails, for example, the propagation mechanism of stegobot. $A[3]$ denotes new binaries coming from other ways, for example, the removable disks.

B. Phase 2: Pre-defined Host Behaviors

Social bots mainly target on OSN sites, thus they perform different host behaviors with conventional bots. As shown in Table II, koobface checks the Internet cookies to find out the OSN sites that the user used [5]. Stegobot scans system files to collect email address and passwords, or credit card number, or log all keystrokes [8]. Facebot steals sensitive information, like user accounts, passwords [9]. Besides these OSN related operations, they also have some conventional operations to ensure them working well.

We summarize possible pre-defined host behaviors in Table III. We use B to denote the set and $B = \{B[1], B[2], B[3], B[4], B[5], B[6]\}$. $B[1-4]$ denotes behaviors similar with conventional bots and $B[5-6]$ denotes specific social bot behaviors. $B[1]$ denotes modifying the bootstrap list of system, for example, modifying bootstrap Registry keys including Run, Runonce and other Registry keys. $B[2]$ denotes modifying the bootstrap list of browser, for example, using Browser Helper Objects (BHO). $B[3]$ denotes logging all the keystrokes, like the techniques used by anti keylogger. $B[4]$

denotes stealing sensitive information, such as the user files browsing history. $B[5]$ denotes checking Internet cookies. $B[6]$ denotes monitoring OSN operations, email operations, etc.

C. Phase 3: Establishment of C&C Channel

Social bots have many different mechanisms of establishing C&C channel as shown in Table II. Koobface connects to C&C server through HTTP to establish the C&C channel [5], and the bot designed by Boshmaf [3] also uses this mechanism. Naz bot visits Really Simple Syndication (RSS) of some specific user accounts to establish [7]. Stegobot share images to establish C&C channel using image steganography to hide the sensitive information [8]. Facebot also utilizes image steganography to establish C&C by hiding sensitive information in user profile picture [9].

We summarize possible C&C mechanisms in Table III. We use C to denote the set of ways establishing C&C mechanisms and $C = \{C[1], C[2], C[3], C[4]\}$. $C[1]$ denotes automatically connecting some specific HTTP servers to establish C&C mechanism, such as koobface and the bot designed by Boshmaf. $C[2]$ denotes automatically uploading messages. A bot can use encrypted messages as C&C mechanism. $C[3]$ denotes automatically uploading pictures to establish C&C channel, such as stegobot and facebot. $C[4]$ denotes automatically visit some specific users, such as the RSS, user profile, etc.

D. Phase 4: Receive the Commands of Botmaster

As shown in Table II, the botmaster of koobface sends encrypted messages to bots through HTTP. Naz bot visits the RSS of a specific twitter user to get a Base64-encoded message. Stegobot visits a specific user account and downloads its several recent images. However, social bot is different with conventional bot in spreading commands. In conventional bot, it is a "push" model, that means the botmaster sends the commands to the bots. For example, in HTTP-based Zeus bot, the bots passively waiting for the commands of botmaster. However, in social bots, it is more like a "pull" model, that means the bot proactive gain the commands of botmaster.

As shown in Table III, D denotes the set of ways receiving commands from botmaster and $D = \{D[1], D[2], D[3]\}$. $D[1]$ denotes the bot automatically downloading some specific user messages. It is related with the C&C mechanism of automatically uploading messages. $D[2]$ denotes the bot automatically downloading some specific user pictures, such as stegobot. $D[3]$ denotes the bot automatically downloading user profiles. It is related with the C&C mechanism of automatically visiting some specific users. $D[4]$ denotes the bot automatically listening on a port and receive messages. This is a conventional HTTP-based mechanism.

E. Phase 5: Execution of Social Bot Commands

Social bots mainly perform malicious behaviors related with OSN sites. As shown in Table II, koobface decrypts the message, downloads related components, and executes them. Then they will perform many different malicious behaviors. In Naz bot, it will decrypt the text message, visit the urls in the message, download malicious file and execute it [7]. In stegobot, it has several OSN related commands, such as receiving search keywords from the botmaster, and responding with matches from the filesystem [8]. In the bot designed by Boshmaf, most commands focus on OSN sites, such as read, write, connect, and disconnect [3].

TABLE III: Behaviors of Different Phases

Phase	Notation	Description
1	A[1]	browser download suspicious binaries
	A[2]	download the binary attachment of emails
	A[3]	other suspicious binaries coming from outside
2	B[1]	modifying bootstrap list of system
	B[2]	modifying bootstrap list of browser
	B[3]	log all the keystrokes
	B[4]	stealing sensitive information
	B[5]	checking Internet cookies
	B[6]	monitoring OSN operations, email operations, etc.
3	C[1]	automatically connect some specific HTTP servers
	C[2]	automatically upload messages
	C[3]	automatically upload pictures
	C[4]	automatically visit some specific users
4	D[1]	automatically download some specific user messages
	D[2]	automatically download some specific user pictures
	D[3]	automatically download user profiles
	D[4]	automatically listen on a port and receive messages
5	E[1]	commands executing in the host
	E[2]	commands executing on OSN sites
	E[3]	commands related with HTTP
6	F[1]	Return the encrypted information to HTTP server
	F[2]	Find the botmaster account and review the information
	F[3]	Automatically join a specific chat group
	F[4]	Automatically publish suspicious messages
	F[5]	Automatically upload suspicious pictures

We divide the commands into three categories based on the operating location, host, OSN, and HTTP server as shown in Table III. We use E to denote the set of execution of social bot commands and $E = \{E[1], E[2], E[3]\}$. $E[1]$ denotes the commands executing in the host, such as the data stealer component of koobface. $E[2]$ denotes the commands executing on OSN sites, such as connect, read, write, and disconnect commands of stegobot. $E[3]$ denotes the commands related with HTTP, such as rogue DNS changer of koobface.

F. Phase 6: Return the Results

After the commands are successfully executed, the bots will return the results. As shown in Table II, different components of koobface return results to specific servers through OSN and HTTP. Naz bot steals user information and sends them to the server through HTTP, and the bot designed by Boshmaf also returns the results through HTTP. Stegobot uploads images with hidden information to return the results. Facebot also uploads images with hidden information, while it targets on the profile picture. At the same time, the bot and botmaster join the same Facebook group. Then the botmaster scans the profile image of every new member and decrpyts the picture to gain the sensitive information.

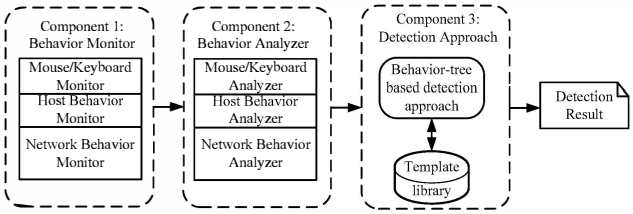
We summarize the possible mechanisms in Table III. We utilize F to denote the set of returning results and $F = \{F[1], F[2], F[3], F[4], F[5]\}$. $F[1]$ denotes returning the encrypted information to HTTP server like the mechanism of Naz bot. $F[2]$ denotes finding the botmaster account and reviewing the information. $F[3]$ denotes automatically joining a specific chat group like facebot. $F[4]$ denotes automatically publishing suspicious messages. $F[5]$ denotes automatically uploading suspicious pictures like stegobot and facebot.

V. METHODOLOGY

A. System Architecture

Figure 2 shows the architecture of our social bot detection system, which primarily consists of three components: behavior monitor, behavior analyzer, and detection approach.

Fig. 2: Detection System Architecture



Behavior monitor contains three modules: mouse/keyboard monitor, host behavior monitor, and network behavior monitor. Mouse/Keyboard monitor records human mouse and keyboard operation. Host behavior monitor records run-time system events of Registry and file system. Network behavior monitor records run-time network inflows and outflows. We utilize pyhook to monitor human mouse/keyboard operation, the similar techniques as Process Monitor to monitor host behaviors, and Microsoft Network Monitor to monitor network behaviors. They all generate corresponding log files at a constant time window which is set to 20 minutes in our experiment. Behavior analyzer component also contains three modules in correspondence with the former component: mouse/keyboard analyzer, host behavior analyzer, and network behavior analyzer. The analyzers extract corresponding behaviors and generate analysis report for component 3. Behavior tree-based approach first construct the behavior tree, then match it with template library to generate the final detection result.

B. Behavior Tree-based Approach

1) Behavior Tree Representation: We formally define behavior tree as: $T = \langle V, E \rangle$, where V is the set of vertices, E is the set of edges. Each vertice represents a behavior and the depth is three. We utilize $L1$ to denote the root layer and $L4$ to denote the leaf layer. In behavior tree architecture, $L1$ represents the categories of our detection result, including social bot, and benign. Suppose $L1$ layer represents social bot, then $L2$ layer is composed of the six phases described in Section IV. For every node in $L2$, the children nodes in $L3$ are the specific behaviors of the phase set. The nodes in $L4$ represents the different implementations of each behavior. We take an example of $B[1]$ as shown in Figure 3.

Fig. 3: Behavior Tree Architecture

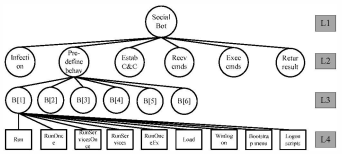
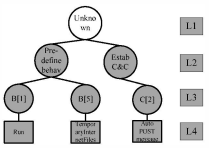


Fig. 4: Flagged Behavior Tree



2) Behavior Tree Construction: After received the analysis reports of behaviors in $L4$ from Component 2, we will construct the behavior tree from bottom to top. Once the behavior in $L4$ layer is identified, we will flag the node, then its parent node in $L3$ layer will also be flagged, and the same to $L2$ layer. After the construction process, we will get its behavior tree. For example, the suspicious process has the following behaviors: modifying the Registry value of Run, check Internet cookies, and automatically upload messages using POST function to OSN sites. Thus we will construct its behavior tree as shown in Figure 4. After the construction procedure, we will match it with the template library.

Algorithm 1 Behavior Tree Construction Algorithm

Input:
The node set in $L4$ layer of behavior tree
Output:
The flagged behavior tree
Empty all the child pointers of each node in the whole tree
for b in behavior node set **do**
  $k = b$
  while $k \rightarrow parent \neq NULL$ **do**
    flag k
    add k as a child of $k \rightarrow parent$
    if $k \rightarrow parent$ is flagged **then**
      break
    else
      $k = k \rightarrow parent$
    end if
  end while
end for
return root

Behavior tree construction algorithm is shown in Algorithm 1, we initialize every node has the parent node pointer. From our analysis reports of Component 2, we can extract the specific behaviors in $L4$ layer. We empty all the child pointers of each node in the whole tree, then we flag the tree from bottom to top. After the construction algorithm, only the root node is not flagged and we can visit all the flagged nodes from the root node.

3) Template Library Construction: Template library construction is an off-line process that is based on three aspects: existing social bot samples, possible social bots of laboratory works, possible implementations from our analysis. If we only extract the templates of existing social bots, we cannot detect novel social bots. Thus, we add the templates of possible laboratory works, such as stegobot, bot designed by Boshmaf, and facebot. In order to deal with the problem of frequent variants, we manually analyze the specific behaviors to find out the "expensive" ones [15]. In this way, we build the social bot behavior tree template library.

4) Behavior Tree Match: After constructing the suspicious behavior tree, we will match it with the template library. We calculate the similarity between the suspicious behavior tree with every template to find out the highest similarity value. Through comparing the similarity with a threshold to determine whether it is social bot or benign.

We utilize tree edit distance to calculate the similarity. The tree edit distance between ordered labeled trees is the minimal-cost sequence of node edit operations that transforms one tree into another. There are three edit operations on labeled ordered trees: (1) Delete a node and connect its children to its parent with the original order; (2) Insert a node between an existing node and a subsequence of consecutive children of this node;

(3) Rename the label of a node. In tree edit distance algorithms, Demaine *et al.* propose an algorithm with $O(n^2 m(1+\log \frac{m}{n}))$ time complexity and $O(mn)$ space complexity [20], m, n denote the number of tree nodes. After that Pawlik and Augsten propose a robust tree edit distance algorithm (RTED) with $O(n^3)$ time complexity and $O(mn)$ space complexity [21].

We select RTED algorithm to calculate the edit distance of behavior trees. We utilize d to denote the edit distance of two behavior trees and s to denote the similarity. In the worst situation of calculating edit distance is that the labels are totally different, then the cost is $max(m, n)$ with delete and rename operation. Thus the value of d is between $[0, max(m, n)]$. We can utilize Equation 1 to calculate the similarity of two trees. We utilize θ to denote the threshold and if $s \geq \theta$ then the root node will be flagged as social bot, otherwise benign. The behavior tree match algorithm is shown in Algorithm 2, after the algorithm we can flag the process as social bot or benign.

$$s = 1 - \frac{d}{max(m, n)} \tag{1}$$

Algorithm 2 Behavior Tree Match Algorithm

Input:
Suspicious behavior tree t
Output:
The result of root node
set $max\_s = 0$
for T in Template **do**
  $d = RTED(t, T)$
  $s = 1 - \frac{d}{max(t.length, T.length)}$
  if $s \geq max\_s$ **then**
    $max\_s = s$
  end if
end for
if $max\_s \geq \theta$ **then**
  flag the root node as social bot
else
  flag the root node as benign
end if

VI. Experiment

A. Data Collection

We have evaluated the performance of our system in detecting seven types of social bots with real-world traces. We obtained the binaries of koobface from public web sites. The author of Twitterbot shared their source code with us [22]. Pieter *et al.* shared TWebot builders and binaries with us [23]. Based on the description provided by Yazan Boshmaf *et al.* [3], we re-implemented it and named it yazanbot. Based on the analysis of Nazbot in [7], we re-implemented it and named it fixNazbot. Besides wbbot, we also design fbbot based on facebook using similar techniques. We set up VMWare virtual machines running Windows XP. While running these bots, we request users to operate the host normally as usual. Both the benign and malicious behaviors were captured by our system. Table IV shows the details of these traces.

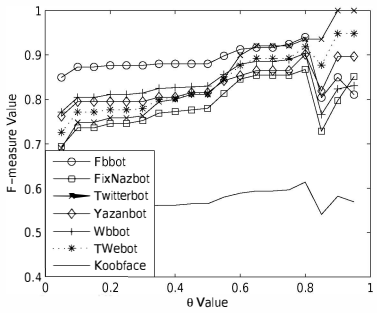
TABLE IV: Social Bot Traces

Trace	Duration	Size
Koobface	24 h	5.32 GB
Twitterbot	24 h	8.36 GB
TWebot	18 h	2.77 GB
Yazanbot	24 h	7.36 GB
FixNazbot	24 h	4.99 GB
Wbbot	18 h	11.5 GB
Fbbot	5 h	4.65 GB

B. Experiment Results

We use all the traces to evaluate the performance of our system. In our experiment, if a benign process is misclassified as malicious, it will be regarded as a false positive (FP). If a malicious process is misclassified as benign, it will be regarded as a false negative (FN). In the same way, we define true positive (TP) and true negative (TN). We first take an experiment to find the best values for θ. Parameter θ is the final threshold for determining whether a suspicious process is suspicious or benign. We use F-measure to evaluate it, which provides comprehensive and accurate result [24]. We enumerate θ value from 0.05 to 0.95 with an incremental step of 0.05. As shown in Figure 5, except for koobface, others all reach a F-measure value greater than 0.8. Almost all of them get robust and high F-measure values when θ is set to 0.8. With these observations, we select 0.8 as the final θ value.

Fig. 5: F-measure Value of Different θ



For each trace, we get the detection result of every time window, and present the average results in Tabel V. We use FP rate and FN rate to represent detection results. We also present the total detection results. The FP rate of all the traces are from 15% to 36%, and the total FP rate is 29.6%. The FP rate is a little high, which lies in that many processes perform similar behaviors with social bots, especially the automatic OSN softwares, such as weibo desktop, renren desktop. Besides that, most social bots mimic user activities or benign software activities. They can also generate many benign behaviors to reduce their suspicious level.

TABLE V: Detection Result

Trace	Avg FP Rate	Avg FN Rate
Koobface	35.9%	31.8%
FixNazbot	34.7%	0%
Yazanbot	35.0%	0%
Twitterbot	15.4%	0%
Fbbot	25.6%	0%
Wbbot	35.3%	0%
TWebot	25.2%	0%
Total	29.6%	4.5%

For the FN rate, koobface is 31.8%, others are 0%, and the total is 4.5%. Koobface has a such high FN rate because we only have their binaries and cannot configure them. In their binaries, the server addresses have already been shut

down. Therefore, in koobface traces, the malicious processes do not have any successful C&C connections. Under this circumstance, koobface behaves more like a malware which only has a little suspicious host behaviors. Thus, our detection system has a high FN rate for koobface. However, the total FN rate is only 4.5%, which is still remarkable. In summary, the results indicate that our detection system has an acceptable FP rate and a remarkable FN rate.

In order to compare with other detection tools, we analyze the binaries of all the social bots using VirusTotal, which provides detection results of popular antivirus engines and website scanners [25]. The results are shown in Table VI, the URL in the table is the ID number and the real url is https://www.virustotal.com/en/file/URL/analysis. Koobface has a very high detection ratio, which is definitely detected as malicious. We should note that koobface is a famous malware which is already in signature databased of most antivirus engines. While others all have a very low detection ratio, which is detected by less than 3 detection tools. Compared with them, our detection result is more remarkable.

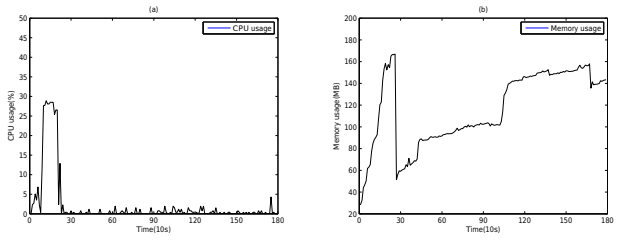
TABLE VI: VirusTotal Detection Result

File / URL	Detection ratio
Koobface (malware.exe) d25f124698ee9af1e234e4be8772223a79d3fea1fe5a6f0b6a39d8650b40bfa2	47 / 54
Twitterbot (TwitterbotClient.jar) 13584184aa4ecaba3c271d72e46d5f4a9f2ea70bc3001b3e82f314e9d6e26c0a	0 / 54
TWebot (test_bot_1.exe) 6f94057e0532bb988f4538b66da641ee5c7c8f52cd30a2bd2fbcfa135b6203ac	1 / 51
Yazanbot (yazanbot.exe) 02a565b257d3d2a65deae70868cd9704c84fd8af0e4f5dddd8fe9ea587634ff7	2 / 54
FixNazbot (naz.exe) a16506bbf918229922ed6040543d456403f6ebd33f7e14ecdcdce19d6a624c38	2 / 54
Wbbot (wbbot.exe) cdc5572928fa4051fae8438d9272ce7f209b0005334c15e5932195c4e86976d4	3 / 53
Fbbot (main.exe) 4e657fa2df5fab3b3b5783a21b13fecdd57fa573c6e56e7cdf98a59b2f155865	2 / 54

C. Performance Overhead

We evaluated the running overhead of our detection system. The CPU and memory usage are shown in Figure 6. The first peak of CPU and memory usage owes to starting the programs. We can see that the CPU usage is low enough while memory usage get steady after a period of rising. As for the three components in our detection system, the behavior monitor consumes most overhead because it monitors and records human mouse and keyboard operation, system runtime Registry, file system and network behaviors. The behavior analyzer component just reads and scans the log files, which only consumes a litter overhead. However, in behavior-tree detection approach, we use T_n to denote the number of templates, its time complexity is $O(n^3T_n)$. The nodes number n is less than 200 and templates number T_n is less than 100, thus $n \leq 200$ and $T_n \leq 100$, that means in the worst situation our approach needs to calculate $800,000,000$ times. It also brings a little delay for real-time detection. We would like to add some heuristic mechanisms to reduce the match with all the trees in template library. For example, when the similarity exceeds the threshold, then the match is finished. We can also optimize the match order with frequency of high similarity values. In summary, our detection system consumes some overhead, however, we believe the remarkable detection result can make up for this.

Fig. 6: CPU and Memory Usage of Our Detection System



VII. Limitation and Future Work

A. Limitation

There are several mechanisms to evasion that we can imagine attackers would adapt against our detection approach. As our approach is based on the analysis of time windows, efforts to divide malicious behaviors into several time windows are an obvious choice. Attackers can set a random time delay between different behaviors. In this way, they can disrupt our approaches. While this evasion mechanism has its own drawbacks, such as cookies are changed, critical processes are terminated, and antivirus tools may identify some suspicious behaviors. In summary, such mechanisms will take an unknown risk to evade our detection. A previous research on conventional bot detection of Zeng *et al.* accumulates some critical behaviors and uses the Exponential Weighted Moving Average (EWMA) algorithm to compute the average suspicion level every time window [26].

Another evasion mechanism is using multiple processes or different instantiations of the same process. In this way, every process only performs one or two malicious behaviors and acts like benign processes. A well-designed multiprocess bot can evade disrupt our approaches. This is a common challenge of detection approaches focusing on single process. Ma *et al.* present such a multiple process mechanism to evade existng behavior-based malware detections by dividing a malware into multiple "shadow processes" [27], [28]. However, multiple processes social bot has to solve the critical problems of bootstrap all the processes, distribution of bot functions, hidden inter process communication (IPC). Thus, it is also a difficult task for attackers.

The FP rate of our detection system is a little higher. Besides social bots have similar behaviors with human or benign softwares, the behaviors in our detection system also need further optimizing. The behaviors from Phase 3 to Phase 6 are difficult to differentiate with benign OSN softwares and human OSN activities. We should filter out more expensive and special behaviors of social bots.

B. Future Work

We are very interested in exploring automated ways of detecting social bots from both host and server side. Our host detection approach faces some limitations as described above, we will try to fix these limitations to pursue a better detection system. Besides detection in host, we will further analyze social bot behaviors in server side and hope that we can find some specific features. And we believe, either host side or server side detection approaches are able to mimic all

social bots, thus a correlation of them may be possible to solve the problems in both of them. How to correlate the malicious behaviors in both sides, how to solve the synchronization problem between them, is it able to detect social bots in real time? These problems are still unknown only if we take steps in them. As in conventional bot detection, the host and network correlation approach of Zeng [26] is able to solve some of these problems and detect different types of botnets with low false-positive and false-negative rates.

VIII. Conclusion

In this paper, we focused on the problem of detecting social bots in the end host. We implement a social botnet, wbbot, based on Sina Weibo. Then, we extract six phases of social bot in the end host: infection, pre-defined host behaviors, establishment of C&C, receive the commands of botmaster, execution of social bot commands, and return the results. Based on the specific behaviors of the six phases, we propose our detection system which consists of three components: host behavior monitor, host behavior analyzer, and detection approach. In behavior tree-based approach, after constructing the suspicious behavior tree, we match it with the templates in the library to calculate the similarity to generate the final detection result. Finally, we evaluate the performance of our approach with real social botnet traces. The results indicate that our system has an acceptable FP rate of 29.6% and remarkable FN rate of 4.5%. However, compared with other detection tools, our detection result is still remarkable.

Acknowledgements

We gratefully acknowledge the funding from the National Natural Science Foundation of China under Grant No. 61170265, Fundamental Research Fund of Jilin University under Grant No. 201103253. Thank Mark Stamp and Pieter Burghouwt for sharing social bots with us.

Corresponding author: Qiang Li, Email: li_qiang@jlu.edu.cn

References

[1] List of virtual communities with more than 100 million active users (accessed June 2014).
URL http://en.wikipedia.org/wiki/List_of_virtual_communities_with_more_than_100_million_users

[2] Malware that drains your bank account thriving on facebook (accessed June 2014).
URL http://bits.blogs.nytimes.com/2013/06/03/malware-that-drains-your-bank-account-thriving-on-facebook/

[3] Y. Boshmaf, I. Muslukhov, K. Beznosov, M. Ripeanu, Design and analysis of a social botnet, Computer Networks 57 (2) (2013) 556–578.

[4] Twitter bots create surprising new social connections (accessed June 2014).
URL http://www.technologyreview.com/news/426668/twitter-bots-create-surprising-new-social-connections/

[5] J. Baltazar, J. Costoya, R. Flores, The real face of koobface: The largest web 2.0 botnet explained, Trend Micro Research 5 (9) (2009) 10.

[6] Twitter based botnet command and control (accessed June 2014).
URL http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel

[7] E. J. Kartaltepe, J. A. Morales, S. Xu, R. Sandhu, Social network-based botnet command-and-control: Emerging threats and countermeasures, in: Proceedings of the 8th International Conference on Applied Cryptography and Network Security, ACNS'10, 2010, pp. 511–528.

[8] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, N. Borisov, Stegobot: A covert social network botnet, in: Proceedings of the 13th International Conference on Information Hiding, IH'11, 2011, pp. 299–313.

[9] J.-P. Verkamp, P. Malshe, M. Gupta, C. W. Dunn, Facebot: An undiscoverable botnet based on treasure hunting social networks.

[10] E. Tan, L. Guo, S. Chen, X. Zhang, Y. Zhao, Spammer behavior analysis and detection in user generated content on social networks, in: Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on, 2012, pp. 305–314.

[11] Z. Chu, S. Gianvecchio, H. Wang, S. Jajodia, Detecting automation of twitter accounts: Are you a human, bot, or cyborg?, IEEE Trans. Dependable Secur. Comput. 9 (6) (2012) 811–824.

[12] F. Brito, I. Petiz, P. Salvador, A. Nogueira, E. Rocha, Detecting social-network bots based on multiscale behavioral analysis, in: SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies, 2013, pp. 81–85.

[13] P. Burghouwt, M. Spruit, H. Sips, Towards detection of botnet communication through social media by monitoring user activity, in: Proceedings of the 7th International Conference on Information Systems Security, ICISS'11, 2011, pp. 131–143.

[14] V. Natarajan, S. Sheen, R. Anitha, Detection of stegobot: A covert social network botnet, in: Proceedings of the First International Conference on Security of Internet of Things, SecurIT '12, 2012, pp. 36–41.

[15] T. Stein, E. Chen, K. Mangla, Facebook immune system, in: Proceedings of the 4th Workshop on Social Network Systems, ACM, 2011, p. 8.

[16] J. Zhang, G. Gu, Neighborwatcher: A content-agnostic comment spam inference system, in: In Proceeding of the Network and Distributed System Security Symposium (NDSS13), 2013.

[17] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, Y. Dai, Votetrust: Leveraging friend invitation graph to defend against social network sybils, in: INFOCOM, 2013 Proceedings IEEE, IEEE, 2013, pp. 2400–2408.

[18] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, R. M. Salles, Botnets: A survey, Comput. Netw. 57 (2) (2013) 378–403.

[19] M. Kammerstetter, C. Platzer, G. Wondracek, Vanity, cracks and malware: Insights into the anti-copy protection ecosystem, in: Proceedings of the 2012 ACM conference on Computer and communications security, ACM, 2012, pp. 809–820.

[20] E. D. Demaine, S. Mozes, B. Rossman, O. Weimann, An optimal decomposition algorithm for tree edit distance, in: Automata, languages and programming, Springer, 2007, pp. 146–157.

[21] M. Pawlik, N. Augsten, Rted: a robust algorithm for the tree edit distance, Proceedings of the VLDB Endowment 5 (4) (2011) 334–345.

[22] A. Singh, Social networking for botnet command and control.

[23] P. Burghouwt, M. Spruit, H. Sips, Detection of covert botnet command and control channels by causal analysis of traffic flows, in: Cyberspace Safety and Security, Springer, 2013, pp. 117–131.

[24] F1 score - wikipedia, the free encyclopedia (accessed June 2014).
URL http://en.wikipedia.org/wiki/F1_score

[25] Antivirus scan - virustotal (accessed June 2014).
URL https://http://www.symantec.com/security_response/writeup.jsp?docid=2010-011016-3514-99

[26] Y. Zeng, On detection of current and next-generation botnets, Ph.D. thesis, The University of Michigan (2012).

[27] W. Ma, P. Duan, S. Liu, G. Gu, J.-C. Liu, Shadow attacks: automatically evading system-call-behavior based malware detection, Journal in Computer Virology 8 (1-2) (2012) 1–13.

[28] Y. Ji, Y. He, D. Zhu, Q. Li, D. Guo, A multiprocess mechanism of evading behavior-based bot detection approaches, in: 10th Information Security Practice & Experience Conference (ISPEC 2014), 2014.