BotGAD: Detecting Botnets by Capturing Group Activities in Network Traffic

Hyunsang Choi, Heejo Lee, and Hyogon Kim
Div. of Computer & Communication Engineering
Korea University
Seoul, South KOREA
{realchs, heejo, hyogon}@korea.ac.kr

ABSTRACT

Recent malicious attempts are intended to obtain financial benefits using a botnet which has become one of the major Internet security problems. Botnets can cause severe Internet threats such as DDoS attacks, identity theft, spamming, click fraud. In this paper, we define a group activity as an inherent property of the botnet. Based on the group activity model and metric, we develop a botnet detection mechanism, called BotGAD (Botnet Group Activity Detector). BotGAD enables to detect unknown botnets from large scale networks in real-time. Botnets frequently use DNS to rally infected hosts, launch attacks and update their codes. We implemented BotGAD using DNS traffic and showed the effectiveness by experiments on real-life network traces. BotGAD captured 20 unknown and 10 known botnets from two day campus network traces.

1. INTRODUCTION

A Botnet is a network of compromised machines controlled by an attacker to carry out online criminal activities including identity theft, e-mail spam, click fraud and DDoS attack. All of these infected PCs are unwilling victims, doing malicious tasks unbeknown to their owners. Numerous automated botnet detection studies have been proposed. Even the studies are helpful, they have difficulties which fall into three categories.

1. Botnet traffic is hard to detect because it is similar to normal traffic. What is worse, it may contain encrypted communication. Since many detection approaches need to analyze contents of botnet traffic (e.g., signature matching to capture IRC [24] traffic), they fail to detect the botnet.

2. Botnets evolve quickly as more users fail to protect their computers, helping the attackers evade existing protection mechanisms. For example, hackers adopt rootkit and packing techniques as a means for circumventing anti-virus softwares. Hackers also use fast-flux hosting [15] to hide origins of their botnets and stay active for longer.

3. Even botnet detections method can capture botnets which use the evasion techniques, most usually need huge amount of data which cannot be analyzed in real-time. For example, if a detection method requires whole TCP/UDP traffic, it may not be practical to detect botnet in a large scale of network.

COMSWARE '09, June 16-19, Dublin, Ireland

We find a common property of botnets, called group activity. Most botnets, irrespective of type and protocol, act as a group. Bots receive/send control traffic, download new codes, migrate the communication channel, and perform malicious behaviors. Most of these behaviors can be regarded as the group activity.

We define a group activity as a key feature of botnets and provide a metric to measure the feature. Using the feature, we develop a botnet detection mechanism, BotGAD. BotGAD focuses on the behavior property of the botnet, not their traffic content or signature. Therefore, BotGAD is robust against the channel encryption technique. We evaluate BotGAD with network traffic data including real-world botnet traces as well as normal traces. BotGAD can detect botnets in large scale networks efficiently, since it works in real-time with a small amount of data.

We developed a primitive botnet detection algorithm in our previous work [3]. In this study, unlike our previous work, we devise a generic metric model to measure group activities. We also analyze related parameters using a stochastic method to improve the detection algorithm in respect of detection accuracy and false positive rates. In addition, we demonstrate a feasibility of BotGAD with real-life network traces.

The rest of this paper is organized as follows. Section 2 reviews related works. We describe a botnet group activity and a BotGAD framework in Section 3. We develop group activity model, observed from different stages of the botnet life cycle and propose a metric to detect the group activity. In Section 4, we introduce DNS-based BotGAD as a case study to verify the research effectiveness. Evaluation results and meaningful analysis are discussed as well. We also illustrate possible evasion techniques.

2. RELATED WORK

In spite of botnet's relatively long presence in the Internet, few botnet detection researches have been proposed. In this section, we review several network based approaches among the researches.

Dagon presents a botnet detection and response approach [6]. It measures canonical DNS request rate and DNS density comparison of botnet rallying DNS traffic. However, the approach is inefficient since it generates many false alarms.

Jones [18] describes the botnet background. He made recommendations so that network and system security administrators can recognize and defend against botnet activity.

Cooke *et al.* [4] outline the origins and structure of botnets. They study the effectiveness of detecting botnets by directly monitoring IRC communication or other command and control activity.

Barford *et al.* [1] present an in-depth analysis of bot software

source code. They reveal the complexity of botnet software, and discuss implications for defense strategies based on the analysis.

BotTracer [21] detects three phases of botnets with the assistance of virtual machine techniques. Three phases include the automatic startup of a bot without requiring any user actions, a command and control channel establish with its botmaster, and local or remote attacks.

Binkley *et al.* [2] propose an anomaly-based algorithm for detecting IRC-based botnet meshes. Using an algorithm, which combines an IRC mesh detection component with a TCP scan detection, they can detect IRC botnet channel with high work weight hosts.

Ramachandran *et al.* [25] develop techniques and heuristics for detecting DNSBL reconnaissance activity, whereby botmasters perform lookups against the DNSBL to determine whether their spamming bots have been blacklisted. This approach is derived from an idea that detects DNSBL reconnaissance activity of the botmaster but it is easy to design evasion strategies.

Husna *et al.* [17] investigate the behavior patterns of spammers based on their underlying similarities in spamming. Principal Component Analysis (PCA) is set to identify the features which account for the maximum variance in the spamming patterns. They calculate the proximity between different spammers and classify them into various groups which represent similar proximity.

Zhuang *et al.* [31] develop techniques to map botnet membership using traces of spam email. To group bots into botnets they look for multiple bots participating in the same spam email campaign. They apply the technique against a trace of spam email from Hotmail web mail services.

Karasaridis *et al.* [19] propose an approach using IDS-driven dialog correlation according to a defined bot infection dialog model. They combine heuristics that assume the network flow of IRC communication, scanning behavior, and known models of botnet communication for backbone networks.

Rishi [9] uses the similarity of nicknames to detect botnet channels.

BotHunter [12] models the botnet infection life cycle as sharing common steps: target scanning, infection exploit, binary download and execution, command-and-control channel establishment, and outbound scanning. It then detects botnets employing IDS-driven dialog correlation according to the bot infection life-cycle model. Malwares not conforming to this model would seemingly go undetected.

BotSniffer [13] is designed to detect botnets using either IRC or HTTP protocols. BotSniffer uses a detection method referred to as spatial-temporal correlation. It relies on the assumption that all botnets, unlike humans, tend to communicate in a highly synchronized fashion. BotSniffer has a similar concept with BotGAD in respect of capturing the synchronized botnet communication. Different from BotGAD, BotSniffer performs string matching to detect similar responses from botnets. Botnet can encrypt their communication traffic or inject random noise packets to evade.

BotMiner [11] presents a botnet detection method which clusters botnet's communication traffic and activity traffic. Communication traffic flow contains all of the flows over a given epoch including flows per hour (fph), packets per flow (ppf), bytes per packet (bpp), and bytes per second (bps). The activity traffic identifies hosts which are scanning, spamming, and downloading any Portable Executable binary. Clustering algorithms are applied and performed cross-plane correlation to detect botnets.

As we mentioned above, several network-based approaches have been proposed to detect the botnets. Even though they are useful, they may suffer from several tactics to evade the detection methods. Stinson *et al.* [26] propose a systematic framework for evaluating an evadability of a detection method to assess the fitness of a detection method. We discuss possible evasion tactics of our mechanism in respect of implementation complexity and effects on a botnet utility in Section 4.3.

3. GROUP ACTIVITY OF BOTNET AND DETECTION SCHEME

3.1 Botnet Group Activity

An inherent property of the botnet, we define a group activity. If bots act as independent hosts, the botnet becomes meaningless from the definition of the botnet. Figure 1 shows an example of group activity from a centralized botnet architecture. We desire

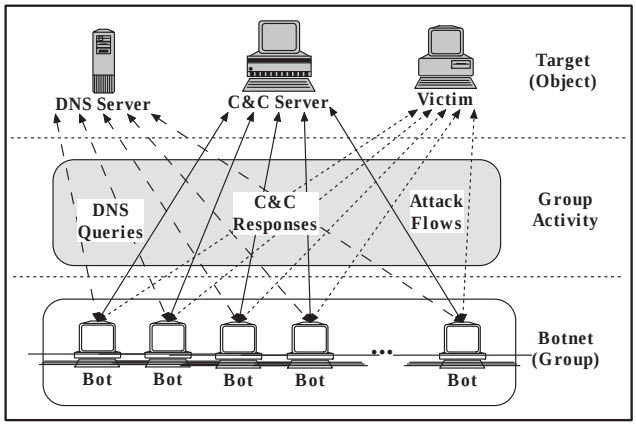

Figure 1: Group activity of centralized botnet

the group activity representation to be generic and intuitive so that models can be understood and explained. Suppose that we monitor incoming and outgoing traffic at a network gateway. Then, two cases of group activities will be observed.

- *Outgoing group activity*: clients in a network construct an internal group and the group performs an activity to a remote target.
- *Incoming group activity*: clients out of a network construct an external group and the group performs an activity to a target in the network.

Let t_i and t_e denote an internal target and an external target, respectively. Then, an internal and an external group (G_i, G_e) which perform activity a to external/internal target within a time window w_n, represented as follows.

$$G_i = \{a, t_e, w_n\}$$

$$G_e = \{a, t_i, w_n\}$$

Consequently, the group activity consist of four factors: group, activity, target, and time.

IRC, HTTP, and P2P are widely used as botnet protocols. The botnet performs group activities irrespective of its protocol. Especially, centralized (IRC/HTTP) botnets frequently perform group communication with a control server. Therefore, group activities can be shown in the centralized botnet more often than decentralized (P2P) botnets. However, group activities can be observed in P2P botnets during upgrading/synchronizing [10]. For example, Storm P2P botnet frequently synchronizes the infected machines

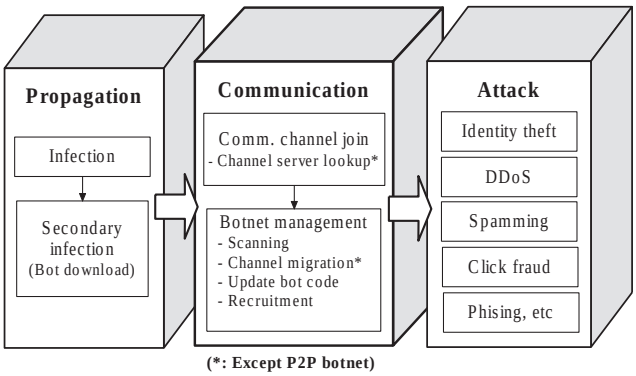

Figure 2: Botnet life cycle

with the Network Time Protocol (NTP) server. This synchronization behavior will be shown as a group activity.

We describe a botnet life cycle in Figure 2. Exploiting a vulnerability (infection) is the first step in the botnet life cycle. Once a vulnerability has been exploited, an infected machine usually downloads a malicious binary from the Internet and executes it. Then, the machine connects to C&C channel(s) to receive orders from a botmaster. Centralized botnets use DNS to look up the channel server. Bots send periodic DNS queries [7] and join the channel. The DNS lookups can be regarded as a group activity. The infected hosts (group) send queries (activity) to the DNS server (target).

After bots are initialized, IRC bots usually keep their connection with the channel using PING/PONG messages of IRC protocol. HTTP bots periodically/sporadically perform channel access by sending HTTP request to take orders from a botmaster [21]. Sending PONG message or HTTP request can be considered as a group activity. Bots (group) send PONG message or HTTP request (activity) to the C&C server (target). We found an actual botnet C&C server in our monitored network. The server sends/receives PING/PONG messages with 104 infected hosts every 205 seconds.

The botnets also perform an array of malicious behaviors including DDoS attack, spamming and click fraud. Especially, centralized botnets move their C&C repeatedly. The botnet malicious behaviors and the C&C migration will be also observed as botnet group activities.

3.2 Characteristics of Botnet Group Activity

As mentioned, group activities can be observed in a botnet life cycle. However, group activities can be appeared in normal communication as well (e.g., flash crowds). Therefore, we find characteristics of botnet group activity to distinguish them from normal group activities as shown in Table 1. When a botnet group performs activity within a period of time, the group is conditionally unchangeable. Here, 'conditionally' means that there could be trivial changes in groups because of temporally deviated member(s), removed member(s), and recruited member(s). Although these changes have effects on the botnet group, they can be ignored within a short time period [7]. The dynamics of IP addresses can also affect the consistency of the botnet group, but the dynamics are ignorable within a short period, as well. The changes in a botnet group and the IP address dynamics will be discussed in Section 4.2. Using the uniformity feature, we can differentiate botnet groups from legitimate groups. We also find temporal properties of botnet group activities. The botnet group activities usually show periodic/sporadic and intensive occurrence.

Table 1: Differences between botnet and normal group activities

	Group Uniformity	Activity Periodicity	Activity Intensity
Botnet group	Consistent (conditionally)	Periodic/ Sporadic	Intensive
Normal group	Fluctuate (random)	Random	Moderate

We measure a similarity coefficient to check a uniformity of groups to detect botnet groups. Assume that a group is observed G within w_n and G' within w_{n+1}. To measure the group uniformity, we compute a similarity between G and G', a real number in [0, 1]. The similarity tends to 1 when the group is a botnet.

Numerous similarity measures in use, differ primarily in the way they normalize the intersection value. We describe three of them: Kulczynski, Cosine and Jaccard similarity coefficients. Kulczynski similarity is represented to be:

$$S_{Kulc}(G, G') = \frac{1}{2}\left(\frac{|G \cap G'|}{|G|} + \frac{|G \cap G'|}{|G'|}\right)$$

and Cosine similarity is represented to be:

$$S_{Cos}(G, G') = \frac{2|G \cap G'|}{\sqrt{|G|}\sqrt{|G'|}}$$

Jaccard similarity is represented to be:

$$S_{Jacc}(G, G') = \frac{|G \cap G'|}{|G \cup G'|}.$$

A group is represented by a vector, with components along exactly those dimensions corresponding to the elements in the group. The similarity measures are based on the binary term vectors and normalize the similarity value between 0 and 1. If both vectors are all zeros, we declare the similarity measure to be one. If only one of the vectors is all zeros, the similarity measure is declared to be zero. Kulczynski similarity yields the average conditional probability that a characteristic is present in one group given that the characteristic is present in the other group. Different from Kulczynski similarity, cosine similarity is length invariant, that is only the direction of the vectors is compared and the magnitude of the vectors is ignored.

3.3 BotGAD Detection Framework

In this section, we illustrate the framework of BotGAD. BotGAD consists of four parts: (1) Data collector, (2) Group classifier, (3) Similarity analyzer and (4) Botnet reporter (Figure 3). The sensors gather network traffic of monitored networks. Incoming/outgoing traffic is aggregated to the data collector. The group classifier makes groups from the traffic using a predefined group size threshold. The similarity analyzer estimates group similarities and the botnet reporter summarizes/reports results.

As shown in Figure 4, groups are generated within each time window w_n. To make the group as a vector and compute the similarity among vectors, we employ a matrix of the groups. The rows of the matrix represent IP addresses of group members and the columns correspond to time windows. We mark 1 when a member perform the group activity within w_n and 0 if it does not. After marking all elements in the matrix, we compute the similarity of each neighbor column vector. The similarity coefficient between column vector w_1 and w_2 indicates measured similarity of

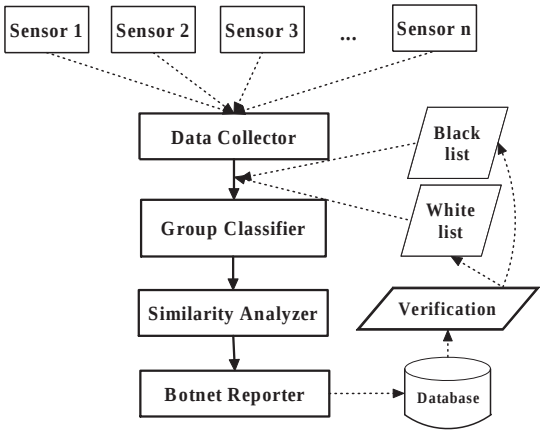

Figure 3: Design of the detection mechanism

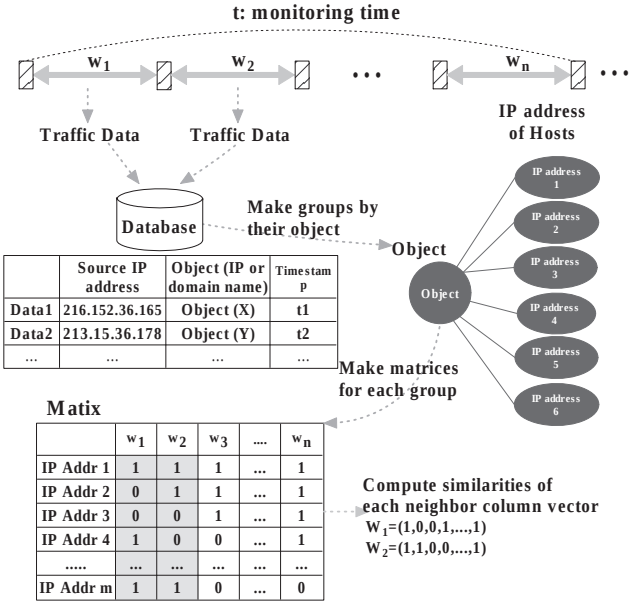

Figure 4: Similarity estimation methods

the group between w_1 and w_2. We estimate the Kulczynski, Cosine, and extended Jaccard similarity(Tanimoto) for a vector calculation, represented S_{Kulc}, S_{Cos}, and S_{Jacc}, respectively.

$$S_{Kulc}(\vec{w}_i, \vec{w}_{i+1}) = \frac{1}{2}\left(\frac{\vec{w}_i \cdot \vec{w}_{i+1}}{\|\vec{w}_i\|^2} + \frac{\vec{w}_i \cdot \vec{w}_{i+1}}{\|\vec{w}_i\|^2}\right),$$

$$S_{Cos}(\vec{w}_i, \vec{w}_{i+1}) = \frac{\vec{w}_i \cdot \vec{w}_{i+1}}{\|\vec{w}_i\|\|\vec{w}_{i+1}\|},$$

$$S_{Jacc}(\vec{w}_i, \vec{w}_{i+1}) = \frac{\vec{w}_i \cdot \vec{w}_{i+1}}{\|\vec{w}_i\|^2\|\vec{w}_{i+1}\|^2 - \vec{w}_i \cdot \vec{w}_{i+1}}.$$

We calculate an average similarity value within a given monitoring time t considering relative errors. Assume that there are n columns ($t = nw$). Some botnet groups can be seen in w_i, not in w_{i+1} due to the relatively small value of w choice. Therefore, we delete deficient column vectors which satisfy $\|\vec{w}_i\|^2 < 0.1 \cdot m$ to reduce relative errors. (m is the number of hosts in the group.)

$$\overline{S} = \frac{1}{n}\sum_{i=1}^{n-1} S(\vec{w}_i, \vec{w}_{i+1}), \overline{I} = \frac{1}{n}\sum_{i=1}^{n} \|\vec{w}_i\|^2,$$

$$\overline{P} = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\sum_{i=1}^{n-1}(t_{i,j} - t_{i+1,j})}.$$

We also calculate the periodicity $\overline{P}$ and intensity $\overline{I}$ for every groups. The periodicity and intensity imply how periodically and intensively the group appears at every time windows. $t_{i,j}$ is a timestamp of entry in row i and column j. We use Euclidean distance to measure the periodicity. If the periodicity $\overline{P}$ is equal to zero, the group entries occurred periodically at each time window. Using the periodicity, we can detect periodic bots. We measure an occupancy rate of contributor columns out of all columns (delete deficient columns) and calculate the intensity of the group. If the intensity is equal to one, the group entries appear intensively. A lot of groups founded in normal communication patterns, do not appear intensively. Thus, the intensity is useful for deleting false alarms.

With the combination of average similarity, periodicity and intensity, BotGAD decides whether a groups is a botnet or not.

1. If an average similarity of a certain group exceeds λ_D, the group is listed on a suspicious group list. Using the similarity estimation, suspicious group including periodic/sporadic botnets and some normal groups are collected.

2. We delete false positives which have $\overline{I}$ lower than λ_I(a threshold for intensity)

3. Among remainder groups, we detect periodic bots using $\overline{P}$. If $\overline{P}$ is smaller than λ_P(a threshold for periodicity), we judge the groups are periodic bots.

As a result, BotGAD reports periodic/sporadic botnet groups, suspicious groups (which need further investigation), and false positives such as update related group activities.

4. CASE STUDY: BOTGAD USING DNS

4.1 Experiment and Evaluation Result

This section describes how we implement a prototype of BotGAD. Generally, botnets utilize DNS for many cases. Centralized botnets perform DNS queries [7] and join to the communication channels. It is possible to use a hard-coded IP address of a C&C server, but it can be perilous to reverse engineering techniques. Therefore, a botmaster arranges several C&C servers in the bot binary. Botnets adopt a dynamic DNS (DDNS) [28] which is a resolution service that automatically perceives a change of the IP address of a server and substitutes the DNS record by frequent updates and changes. By using the DDNS service the bots can always keep there communication channel opened to the controller. The following six cases show the DNS used in botnets:

- *Rally*: If a host infection succeeds, the host send DNS query to know the name of a C&C server.

- *Update*: Botnets usually update their codes with the latest one by downloading it from their web repository. the botnets find the repository using DNS.

- *Synchronization*: Some botnets synchronize the system time of infected machines with the Network Time Protocol (NTP) using time server DNS (e.g., Storm worm botnet [16]).
- *Cloning and Reconnection*: Bots frequently do cloning and reconnecting to be undetectable. At the moment, bots find their new/old channel servers using DNS.
- *Migration*: Botnets migrate C&C servers using DNS.
- *Attack*: Spamming, DDoS attack and click fraud attacks may use DNS to find victims.

We also implement BotGAD using group activities observed in TCP/UDP traffic. However, they generate many false positives and it is a complicated work to inspect the results automatically. Therefore, we evaluate only DNS-based BotGAD in this paper. The DNS-based BotGAD cannot detect botnets which do not use DNS.

We obtained DNS traces tapped from the gateway router of /16 campus network on May 19th, 2008 (Experiment #1). From 1Gb/s line, 6.28GB of DNS traffic was captured (0.58Mbps) 2. 19.52 million DNS queries were captured and 81% were A and MX query types (qType) with which BotGAD only deals. We observed average 640,000 domain groups. Among them, about 80% of the groups have a single host. Only 8% of the groups (51,200) have more than 3 hosts. The largest group was Microsoft Windows update (*www.update.microsoft.com*) which has 8001 hosts. In the experiment, we decided a group size threshold, λ_S to be 3. Therefore, BotGAD delete groups which have smaller size than 3. A relationship between the group size and a similarity is shown in Figure 5. When the group size is small, the similarity of the group tends to be high. Because, as the group size gets smaller, hosts of the group have high probability of behaving similar actions, by accident.

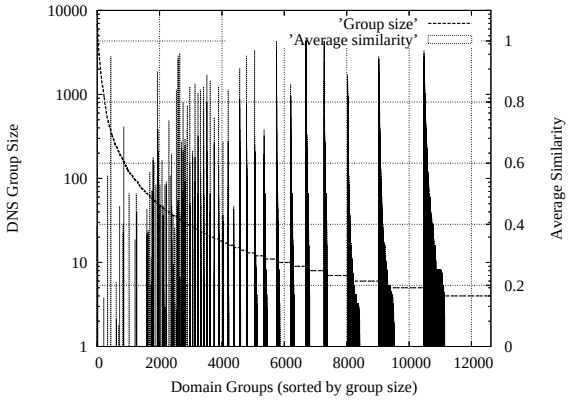

Figure 5: Measured cosine similarity sorted by group size

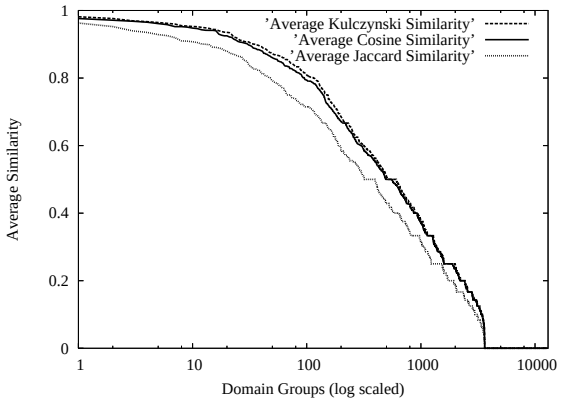

Figure 6: Measured 3 different similarities

We estimated the Kulczynski, Cosine and Jaccard similarities (w: 10 minute, t: 1 hour). Figure 6 shows measured average similarities. 72% of groups' average similarity are estimated to 0. The three methods always follow the order, $\overline{S}_{Jacc} \leq \overline{S}_{Cos} \leq \overline{S}_{Kulc}$. When detection threshold λ_D is 0.8, S_{Kulc}, S_{Cos} and S_{Jacc} report 121, 109 and 51 suspicious domains respectively. However, only 22 domains are truly botnet domains. With the measured intensity of each group, we delete 71, 66 and 33 false positives domains. However, 28, 21 and 7 false positives still remains. Most are update related domains, such as an antivirus software update, installshield update and toolbar update. The update related domain groups occur frequently and periodically, similar with botnets. These domains can be removed using whitelist (we made whitelist using known popular web, software domains).

We tested BotGAD again, with the same campus network traffic on Dec 24th, 2008 (Experiment #2). Captured traces include 1.48GB of DNS traffic estimated at 0.14Mbps and 4.6 million DNS queries (Table 2). DNS queries are decreased remarkably because the NAC (Network Access Control) [29] solution was installed in the network during June, 2008. 10,850 domain groups were captured and BotGAD reports 30 domains as suspicious groups including 14 false positives and 16 botnets. The DNS queries and false positives are decreased because normal traffic is decreased by the NAC solution. The comparison infer that the NAC solution affects positively to BotGAD.

Table 2: Comparison of experiment results

	Experiment #1	Experiment #2 (NAC)
DNS Queries	19.52M	4.6M
Domain Groups	51,200	10,850
Suspicious Groups	43	30
Detected Botnets	22	16
Unknown Botnets	14	12
Known Botnets	8	4
False Positives	21	14

Table 3: Detected botnet domains and false positives (time window w=10 min).

Result Type	Domain Name	Group Size	Average Similarity
Unknown Botnets	bosam.gnway.net	13	0.99
	shiyansend.zyns.com	33	0.98
	shiyansend.solaris.nu	33	0.97
	shiyansend.servebbs.org	29	0.87
Known Botnets	tzhen.3322.org	14	0.92
	proxima.ircgalaxy.pl	33	0.87
	proxim.ntkrnlpa.info	4	0.85
	roon.shannen.cc	11	0.83
	time.nist.gov*	50	0.91
False Alarms	updates.installshield.com	22	0.94
	us.update2.toolbar.yahoo.com	33	0.93
	asp.ircdevilz.net**	1	0

We classify the detected result into known botnets, unknown bot-

nets and false positives with our manual inspection steps as follows.

1. *Check blacklist*: Compare with exist blacklists acquired from KISA (Korea Information Security Agency) [20] and CyberTA [5]. If the domain is in the blacklist, we regard the domain as known botnet.

2. *Ascertain resolved IP address of the domain*: Look up the domain whether the resolved IP address is abnormal or unaccessible.

3. *Domain name keyword matching*: Extract all keywords from the second and third level domain name from the blacklists, and find the keywords from the result domain names.

4. *Investigate the domain information from the domain crawler [8]*: Look up the domain whether it acts as a name server, mail server, or web server and check the domain is listed on RBLs such as Spamhaus SBL.

5. *Google based inspection*: Use approach similar with Google-based profiling method [27] to verify the result domain.

6. *Nmap portscans*: Access the domain and explore what services (application name and version) are offered by the domain using an nmap [23] portscans.

The known botnets are revealed at step 1. We distinguish unknown botnets from step 2 to 6. If a domain satisfies more than three suspicious condition among step 2∼6, we regard the domain as a unknown botnet. For example, if a domain is resolved abnormal IP address listed on RBL and have third level domain listed in the blacklist, the domain is classified into the unknown botnet. Remainders are considered as the false positives.

From experiments, 20 unknown botnets and 10 known botnets are detected. We list some detected botnet domains and false positives in Table 3. BotGAD cannot detect a single infected client since they do not conform to a group such as *asp.ircdevilz.net*** in Table 3. Therefore, a single client C&C remains as inevitable false negatives.

BotGAD reports *time.nist.gov** used by the NIST Internet Time Service (ITS) as a botnet domain. We find that 20 hosts among overall 89 hosts who send queries of NIST domain, generate excessive queries estimated 4.2 queries/sec. Obviously, they are abnormal because NIST does not allow to send queries more frequently than once every 4 seconds. Moreover, normal system queries the NIST domain at every polling interval set in the Windows registry (usually couple of minutes or hours). Digging deeper inspection to the 20 abnormal hosts, we find that they are infected by Storm botnet, the largest known P2P botnet. Storm synchronizes the system time of the infected machine with the help of the Network Time Protocol (NTP) using time server domains (*time.nist.gov* and *time.windows.com*) [16]. Consequently, BotGAD captures not only IRC and HTTP botnets but also P2P botnet (Storm) synchronization activity.

Figure 7 shows false positive rates (false positives out of all measured groups). We estimate the false positive rates within a given monitoring time t from 1 to 6 hours. Jaccard produces lower false positives than the others.

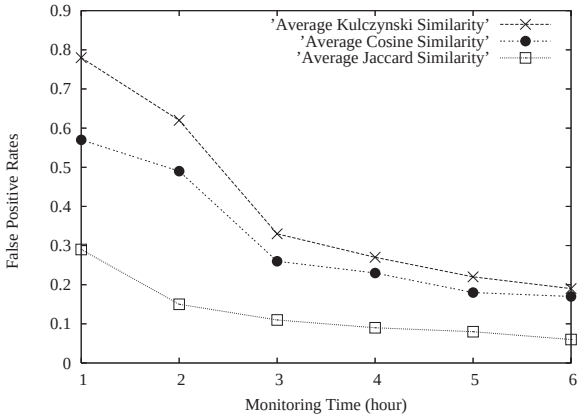

Figure 7: False positive rate

4.2 Parameters and Analysis

We now describe parameters of BotGAD. We divide the parameters into two types, parameters from the group property and parameters from the detection mechanism (DNS based BotGAD) (Table 4). If $\Delta n = 0$, it is clear that we can detect a botnet domain group when we choose t that satisfies $\alpha \cdot t > 0$. Suppose the worst case that the botnet are groups appeared randomly. We derive a similarity equation using Poisson distribution.

Table 4: Parameters related with BotGAD

	Parameter Description	Variable
Parameters from The Group Property	TTL in DNS resource record	T_L
	DNS querying ratio	α
	Botnet querying delay	T_D
	Group size and change in the group size	$n, \Delta n$
Parameters from The Detection Mechanism	Monitoring time	t
	Time window	w
	Group size threshold	λ_S
	Detection threshold	λ_D

$$S = \frac{n}{n + |\Delta n t|}\left(1 - \frac{1}{e^{\alpha(t-x)}}\right),\ x = \begin{cases} w, & w \geq T_L \\ T_L, & w < T_L \end{cases}$$

The derived equation consists of four parts: TTL in DNS resource record T_L, group size parameter n and Δnt, parameters from the detection mechanism t and w, and DNS querying ratio α.

- *TTL in DNS resource record T_L*: Most operating systems including Windows have DNS resolver cache. For example, when the Windows resolver receives a positive or negative response to a query, it adds that positive or negative response to its cache, and as a result, creates a DNS resource record [22]. If a DNS resource record is in the cache, the resolver uses the record from the cache instead of querying a server. After a time period specified in TTL in the DNS resource record, the resolver discards the record from the cache. The cache can decrease botnet DNS queries as well as normal queries. However, botnets commonly use DDNS [14], so we can estimate the value of TTL applied in botnet domains. Errors from the T_L will be reduced when we choose time window w larger than T_L of botnet domains.

- *Change in the group size Δn:* We can estimate Δn from the botnet propagation model [7]. When considering IP address dynamics, the authors in [30] address that more than half (61.4%) of the IP addresses were observed as dynamic IP addresses. However, over 95% of IP addresses have inter-user durations longer than a hour. Errors derived from dynamics of IP addresses can be ignored when we choose the value of w within an hour.

- *Time window w and monitoring time t:* The time window w should be decided with T_L of botnet domains, α based on an existing bot implementation, and T_D. If we choose too small value of t, botnet queries will be decreased by the DNS resolver cache. And asynchronous botnets will not be shown because of a botnet DNS querying delay T_D. Therefore, we should select w which follows $min(T_L, T_D) < w$. A large value of w will increase false positives because many normal groups will be selected as a suspicious group. Therefore, we choose the time window considering the overall related parameters and false positive rates.It is obvious that false positives decreases when the monitoring time t increases (Figure 7).

- *DNS querying ratio α:* We set w to 10 minute ($<$ 1hour) in our experiment, so we can consider $\Delta n = 0$. $T_L < w$. Then, the derived equation can be approximated to

$$S \approx 1 - \frac{1}{e^{\alpha(t-w)}}.$$

If $t = 5w$, similarity S will be approximated as shown in Figure 8. The graph implies that α is the most important parameter of BotGAD.

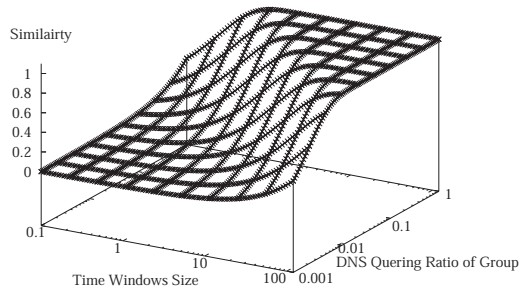

Figure 8: Relationship among similarity S, time window w, and DNS querying ratio α

4.3 Evadability of BotGAD

Even though several automated botnet detection methods have been proposed, botnets can evade them with a high level of attacker power. To evaluate the evadability of BotGAD, we refer Stinson's [26] systematic evaluation which demonstrates an evasion tactic in respect of two associated costs: implementation complexity and effect on botnet utility. An evasion tactic's implementation complexity is based on the ease with which bot writers can incrementally modify current bots to evade detection. If it takes high implementation cost, the evasion tactic results in less useful. An effect on botnet utility is also important cost to the adversaries. If an evasion tactic restricts the botnet utility, the tactic is less effective. Stinson *et al.* list the botnet utility: diversity of attacks, lead time required to launch an attack, botnet size, attack rate, and synchronization level. We now address likely evasion techniques of BotGAD and evaluate the evadability (Table 5).

1. *Evasion by restricting attack targets:* We define a group activity model assuming that the detection system monitors a network gateway (inbound or outbound). Therefore, botmaster can restrict botnet communications or attacks to target hosts on the same internal or external network for the evasion. The evasion tactic has low complexity and results in decreased botnet size for any given attacks.

Table 5: Evasion tactics used to defeat BotGAD, the tactic's implementation complexity and effects on botnet utility

Evasion Tactic	Implementation Complexity	Effects on Botnet Utility
1. Restrict attack targets	Low	↓Attack diversity
2. Induce IP churn	Unknown	None
3. Threshold attacks	High	↓Attack rate
4. Botnet subgrouping	Low	↓Botnet size
5. Minimize the synchronicity	High	↓Attack diversity

2. *Evasion by inducing IP churn:* BotGAD use IP addresses to detect a botnet. Therefore, bots can obtain a different IP address on demand will defeat the BotGAD. An implementation complexity of this technique is unknown.

3. *Evasion by threshold attacks:* BotGAD relies on time related variables, time window and monitoring time. A botmaster can control a botnet behavior frequency to decrease the group similarity. The implementation complexity is high and applying this tactic reduces attack rates of the botnet.

4. *Evasion by the botnet subgrouping:* A botmaster can apply multi-purpose time sharing botnets where a subset of bots are used for one purpose (DDoS) and others for another purpose (spamming). If the botnet subsets are static, BotGAD can detect each subset independently. However, if the botmaster randomly changes the subsets, BotGAD can detect each subsets only when the monitoring time is shorter than the changing frequency.

5. *Evasion by minimizing the synchronicity:* Bots can delay communication time and do not perform any synchronized attacks to evade BotGAD. For example, many new botnets have adopted P2P architecture, where bots are coordinated in distributed fashion. The proposed detection system can not capture the P2P bots if more than $(1 - \lambda_D)n$ bots have communication delay time larger than w, and do not perform any synchronized attacks. In this case, botnet utilities including botnet's synchronization level and diversity of attacks will be decreased.

The implemented BotGAD use DNS traffic to detect botnet, evasion tactics using DNS is possible. If bots avoid using DNS, BotGAD cannot detect the bots. However, using the IP address instead of DNS can be perilous to reverse engineering techniques The bots will lose their mobility and robustness. If bots intentionally generate fake DNS queries using source address spoofing, the fake queries can poison BotGAD. We can check follow-up TCP connections of DNS queries to delete the fake queries.

From the analysis of the evasion tactics, we recognize that the botnet random subgrouping would be the most effective tactic to evade the BotGAD. We have a plan to improve BotGAD to be robust against the botnet subgrouping.

5. CONCLUSION

It is necessary to provide appropriate countermeasures to botnets which represent the major threats to network security and major contributors to unwanted network traffic. Therefore, we proposed BotGAD to reveal both unknown domain names of C&C server and IP addresses of hidden infected hosts. We define an inherent property of botnets, called group activity. We develop metric model to measure the property and detection mechanism which can detect

botnets from large scale networks in real-time. Botnets frequently use DNS to rally infected hosts, launch attacks, update their codes. Therefore, we implemented BotGAD using DNS traffic as a case study. We showed the effectiveness of the implemented system by the experiments on real-life campus network trace. Consequently, BotGAD captured 20 unknown and 10 known botnets from two day campus network trace.

6. ACKNOWLEDGEMENTS

This research was supported by the Ministry of Knowledge Economy, Korea, under the ITRC support program supervised by the IITA (IITA-2009-(C1090-0902-0016)) and the IT R&D program of MKE/IITA (2009-S-026-01, The Development of Active Detection and Response Technology against Botnet).

7. REFERENCES

[1] P. Barford and V. Yegneswaran. An inside look at botnets, 2006. Special Workshop on Malware Detection, Advances in Information Security, Springer Verlag.

[2] 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)*, 2006.

[3] H. Choi, H. Lee, H. Lee, and H. Kim. Botnet Detection by Monitoring Group Activities in DNS Traffic. In *Proceedings of IEEE Int'l Conf. Computer and Information Technology (CIT'07)*, Oct 2007.

[4] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disturbing botnets. In *The 1st Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '05)*, July 2005.

[5] Cyber-TA. SRI Honeynet and BotHunter Malware Analysis Automatic Summary Analysis Table. http://www.cyber-ta.org/releases/malware-analysis/public/.

[6] D. Dagon. Botnet detection and response. In *OARC Workshop, 2005*, 2005.

[7] D. Dagon, G. Gu, C. Lee, and W. Lee. A taxonomy of botnet structures. In *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, Dec 2007.

[8] Domaincrawler. Domain Information Services. http://www.domaincrawler.com/.

[9] 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.

[10] J. Grizzard, V.Sharma, C. Nunnery, B. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, Apr 2007.

[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)*, July 2008.

[12] 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)*, August 2007.

[13] G. Gu, J. Zhang, and W. 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)*, February 2008.

[14] S. Herona. Working the botnet: how dynamic DNS is revitalising the zombie army. *Network Security*, pages 9–11, Jan 2007.

[15] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Detection and mitigation of fast-flux service networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, Feb 2008.

[16] T. Holz, M. Steiner, F. Dahl, E. Biersacky, and F. Freiling. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In *Proceedings of the Firts workshop on Large-scale Exploits and Emergent Threats (LEET'08)*, Apr 2008.

[17] H. Husna, S. Phithakkitnukoon, S. Palla, and R. Dantu. Behavior analysis of spam botnets. In *Proceedings of The 3rd Intl. Conf. on COMmunication System softWAre and MiddlewaRE (COMSWARE'08)*, Jan 2008.

[18] J. Jones. Botnets: Detection and mitigation, Feb 2003. FEDCIRC.

[19] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, Apr 2007.

[20] Korea Information Security Agency (KISA). Botnet C&C server domain list. http://www.knsp.org/sink_dns/total.uniq.dns.rr.txt.

[21] L. Liu, S. Chen, G. Yan, and Z. Zhang. BotTracer: Execution-based bot-like malware detection. In *Proceedings of the 11th Information Security Conference (ISC 2008)*, Sep 2008.

[22] Microsoft Help and Support. http://support.microsoft.com/kb/318803.

[23] Nmap, Network Mapper. Free Security Scanner. http://nmap.org/.

[24] J. Oikarinen and D. Reed. Internet Relay Chat Protocol. RFC 1459, 1993.

[25] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using dnsbl counter-intelligence. In *The 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '06)*, 2006.

[26] E. Stinson and J. C. Mitchell. Towards systematic evaluation of the evadability of bot/botnet detection methods. In *Proceedings of the 2nd USENIX Workshop on Offensive Technologies (WOOT'08)*, July 2008.

[27] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci. Unconstrained endpoint profiling (googling the internet). In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication (SIGCOMM'08)*, Aug 2008.

[28] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic updates in the domain name system (DNS update), 1997. http://www.faqs.org/rfcs/rfc2136.html/.

[29] Wikipedia. Network Access Control. http://en.wikipedia.org/wiki/Network_Access_Control.

[30] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are ip addresses? In *Proceedings of the ACM SIGCOMM 2007 conference on Data communication (SIGCOMM'07)*, Aug 2007.

[31] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, I. Osipkov, G. Hulten, and J. D. Tygar. Characterizing botnets from email spam records. In *Proceedings of the Firts workshop on Large-scale Exploits and Emergent Threats (LEET'08)*, Apr 2008.