5-2 A Holistic Perspective on Understanding and Breaking Botnets: Challenges and Countermeasures

ZHANG Zonghua and KADOBAYASHI Youki

Malware has gained the most prevalence in today's cyber- attacks that threaten our network assets. More seriously, their attack consequence can be significantly enlarged when a huge amount of bots (hosts compromised by malware) coordinate each other with particular intents by constructing botnets. While various prevention, detection, and response techniques have been developed for defending against botnets, attackers constructing and maintaining botnets always manage to evade defense systems. Instead of limiting our attention to the technical design of specific detection techniques, this paper rather gives a comprehensive review on the features and security-evasion techniques that can be possessed by the botnets, with the objective to obtain a fundamental understanding on sophisticated attacker behavior, which is believed to be the preliminary yet essential step towards the design and development of effective anti-botnet techniques. We then develop a top-down analytical framework as a basis for critical evaluation on the existing countermeasures. The framework not only allows us to envision a holistic methodology for achieving in-depth defense boundary of computer networks in the presence of bots, but also suggests a number of practical ways for detecting bots at different system levels with certain degree of sophistication.

Keywords
Network security, Intrusion detection, Botnet, Malware

1 Introduction

Among various cyber-attacks, the most destructive and difficult ones to cope with are those that occur in stages over time and cooperated by a group of attack parties, which is called multi-stage coordinated attacks, or MSCA[58]. To accomplish such an attack, an attacker needs to undergo a process of reconnaissance, penetration, attack, and exploit. Meanwhile, attacker members need to plan and cooperate with another for their common goals by resource/tool sharing, task allocation, information communication and synchronization. As one of the MSCA variants, botnets gain increasing prevalence in today's cyber-attacks that significantly threaten our network assets[19]. As reported, there were less than one thousand bots observed in 2003, while the magnitude has already reached to millions in 2007.

A botnet is consisted of a large amount of bots, i.e., the programs that operate in an automated way for a user or the other programs, which also generally refers to the hosts that are compromised and controlled by malware. While malware may intrude a system in many forms, e.g., Trojan horses, spyware, keystroke loggers, rootkits, their ultimate goal is to gain privileged access to a system and conduct malicious activities without the owner's awareness or informed consent. In particular, Botnets distinguish themselves from other intrusion forms in two salient features: Firstly, they have clear intents, e.g., financial profits[18]; Secondly, a botmaster (an attacker controlling the bots) can interact with its bots via particular command and control (C&C) mechanism[44]. A typical life-cycle of botnets basically contains four stages: probing vulnerable hosts, breaking in the host, establishing C&C channel and interacting with botmaster, and infecting the next target. However, a successful propagation, as well as the interactive operations between botmaster and bots, must rely on particular architecture and protocols[15]. One of the most popular protocols used by botnets, for example, is IRC (Internet Relay Chat)[6] , which is specifically designed for large social chat rooms. Once the botnet is constructed, the botmaster may launch a variety of attacks, such as spamming, DDoS, traffc sniffng, by utilizing those bots spreading over the Internet.

It is obvious that the multi-stage coordinated nature reveals both temporal and spatial characteristics of botnet. More specifically, the bot infection process must rely on a number of stepping-stones, each of which only occurs when the previous one has been achieved, so a complete intrusion sequence needs a certain time. Moreover, a botmaster usually controls a large amount of bots, which may act simultaneously in a similar manner at a particular moment. On the one hand, the spatio-temporal characteristic enables a network defender to monitor botnets[14][21], by integrating both host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS). On the other hand, from the perspective of botmasters, they may intend to make their enclaves obfuscated to evade detections. For instance, a powerful botnet may adopt encryption techniques to secure its C&C channel and avoid honey-pot traps[60], and a more sophisticated botmaster can use peer-to-peer communications to enhance the robustness of his botnet in case of the failure of the singular C&C server[53]. In addition to the coarse-grained properties, an individual bot may leave behavioral traces with fine-grained observable subjects, e.g., traffic packets, system log files, files systems, memory, as the consequence of attacks. A perfect profile of a bot is thus expected to contain all those traits[56]. To that end, virtual machines (VMs) serves as a useful tool to construct honeynets for detecting and tracking bonets by conducting fine-grained malware behavior analysis[2]. Taking the essential characteristics of botnets as a starting point, this paper presents a comprehensive review on the botnet features, especially those of the next-generation botnets, which may significantly deepen our understanding on the sophisticated botmaster behavior and thus facilitate the design and development of effective countermeasures. We also develop a top-down framework, from the Internet to hosts, as a basis for critical evaluation on the existing botnet detection techniques. The framework allows us to envision a holistic methodology for achieving in-depth defense mechanisms by developing and integrating different anti-bot techniques and tools, which also suggests us a number of practical ways for monitoring, detecting, and tracking bots at different system levels with certain degree of threat and sophistication.

The rest of this paper is organized as follows. Section **2** gives a general botnet analysis with emphasis on its architectures and essential characteristics. Section **3** addresses the next generation botnet features and security-evasion techniques. We then investigate the existing techniques for botnet detection and propose a holistic methodology for analyzing and detecting botnets in Section **4**. Section **5** concludes this paper.

2 An overview on Bontnets

The construction and maintenance procedure of botnet is illustrated in Fig. 1, where the edge numbers show the sequence of botnet behavior, which is specifically described as follows,

1. Botnet probes a target network (with a certain address range) and exploits the vulnerability of the victims, by means

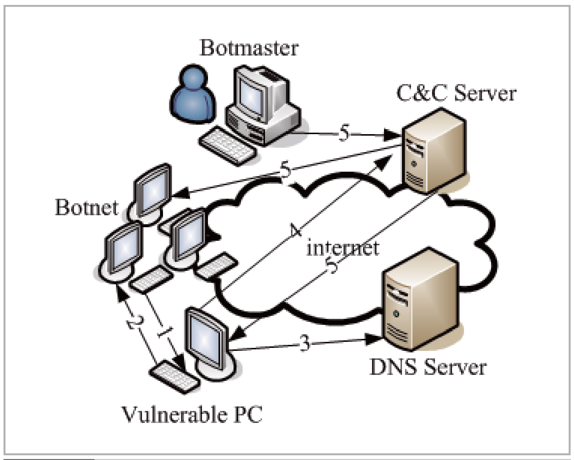

Fig.1 Botnet construction and maintenance

of self-replicating worms, viruses, and so on,

2. the victim executes a shell script (stealthily) and downloads the bot binary from the infection source, and then the bot installs and runs automatically by embedding itself into the system boot process,
3. the bot contact C&C server(s) by looking up their IP address through DNS servers (which can be ignored and is dependent on the network structure and communication protocols),
4. the bot establishes a connection with the C&C server (via particular communication channel) and joins in the botnet,
5. the bot receives commands from the botmaster for fetching binary updates and launching further attacks, infecting other victims in the meantime.

While the general procedure is similar (usually the first four steps are automated and the last step is command-controlled), bots do not necessarily use the same technique to implement each stage. In particular, they may use different infiltration schemes, C&C server architectures and protocols (IRC, P2P, HTTP, etc.), channel obfuscation techniques, bot-botmaster rallying mechanisms and authentication schemes, and so on. A fundamental understanding on botnets, therefore, relies on the exploration of both coarse-grained and fine-grained properties.

2.1 Structures of command and control servers

The botnet construction and maintenance, as well as the communication between bots and botmaster, rely on particular structures and protocols, which also determine the ways that bots fetches binary updates from botmaster and botmaster controls bots via commands. As shown in Fig. 1, the communication is initialized by the newly infected victim, which attempts to contact C&C server for joining the botnet. In general, botnet structure can be classified in terms of the models of C&C server into two categories, centralized and distributed. The distributed structure can be further implemented by P2P-based and random model. For better illustration, the three C&C models are shown in Fig. 2.

A botnet with centralized C&C model usually owns one or a few C&C server, so the communications between the bots and botmaster are all directed to the single C&C server, as shown in Fig. 2(a). This centralized C&C model is often used by the existing botnets due to its easy implementation and maintenance. Since a botmaster is always goal-directed and profit-driven, a botnet with centralized C&C model can be easily constructed by simply compromising only one computer as C&C server for controlling thousands of bots. Another advantage of this kind of model is that it is easy for a botmaster to control and coordinate the bots with less latency, the botnet maintenance is also easier than the other models. However, one drawback of centralized C&C model is that the C&C server plays as a single point of failure of the botnet, and the whole botnet would be disrupted once the C&C server is detected and removed. A majority of today's botnets are using centralized C&C model, such as AgoBot, SDBot, BRbot, SpyBot.

A more robust C&C model is P2P-based, where a pool of C&C servers are distributed in botnets. This structure may avoid the failure point in centralized models, as the communi-

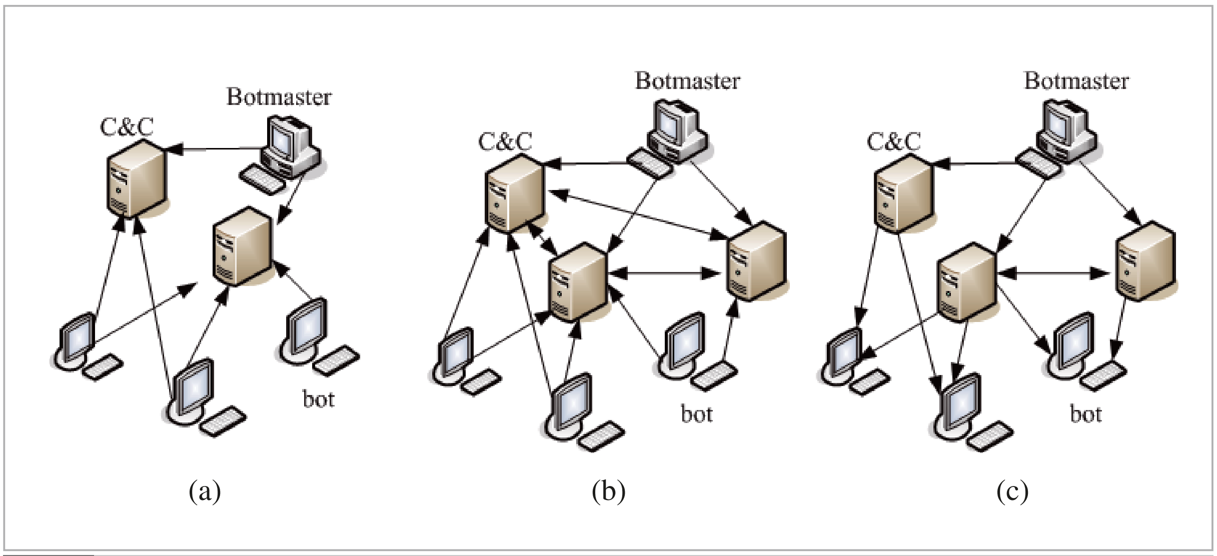

Fig.2 *C&C Structures*

(a) Centralized C&C servers (b) P2P-based C&C servers (c) Random C&C servers

cation between bots and botmaster relies on multiple C&C servers rather than a single one, as shown in Fig. 2(b). With such a structure, the botmaster can still control the botnet through those alive servers even though a fraction of C&C servers have been detected. Obviously, however, the construction and maintenance need much more effort. The scalability of this structure is limited due to the intrinsic characteristic of P2P, where only a small number of clients can be supported. Also, since botmaster needs to maintain C&C servers, along with the communication between bots and C&C server, the coordination among bots is harder to achieve, and the response latency of bots is considerable. The typical botnets using this structure includes Phabot, Storm, Slapper, Nugache, and Sinit. Another more complex C&C model is random in essence[9], which is illustrated in Fig. 2(c). In such a model, a bot does not actively initialize the connection with botmaster, it rather waits for the botmaster's call and then act further as the commands from botmaster. As such, the botmaster has to spend time hunting the bots it may own before instructing the bots and launching attacks. While the random model is feasible in theoretical sense, it has not yet been widely used in practical botnets. Like P2P-based C&C model, the random model suffers from low scalability and high response latency, even though it is easy to be implemented and is robust to detection and tracking.

In order to evaluate the utility of different botnet structures, Dagon et al.[15] proposed three key metrics, i.e., effiectiveness, efficiency, and robustness. While the metrics provide us insights into the bots' networks (mainly bots coordination), they are insufficient for evaluating the three C&C models discussed above. To make it more complete, we propose another five metrics for comparing the characteristics of the introduced C&C models, namely, easiness of construction and maintenance (C&M), effectiveness, robustness, latency, and scalability. A brief comparison is given in Table 1, and the metrics are explained as follows,

- Easiness of C&M measures the easiness of botnet construction and maintenance with respect to the network size, propagation scale, bot update speed, etc.,
- Effectiveness is used to estimate an overall utility of a botnet given a particular intent like email spamming and DDoS,
- Robustness evaluates the botnet vulnerabilities and its resilience to the failure of C&C servers,

Table 1 *A comparison of three typical C&C models*

Structure	Easiness of C&M	Effectiveness	Robustness	Latency	Scalability
Centralized	High	High	Low	Low	High
P2P-based	Low	Medium	High	Medium	Medium
Random	Medium	Medium	Medium	High	Low

- Latency measures the efficiency of bots behavior in terms of command-message receiving and forwarding, bots coordination, and so on,
- Scalability determines whether the botnet can be easily extended or shrunk.

2.2 Communication protocols

C&C servers construct the backbone of a botnet, enabling bots to contact their botmaster, as well as further communication with particular network protocols. Usually the C&C server addresses are included in the bot binary, a bot thus can directly connect its C&C server when parses and executes its binary codes. However, the hard-coded IP address puts C&C server at the risk of being tracked if one of bots is captured and intercepted during its bootstrap process. A more secure and reliable rallying mechanism may rely on Dynamic DNS services, allowing a botmaster to arbitrarily change its C&C server by simply updating the IP address (of the old C&C sever) in the dynamic DNS entry. As such, the bots can be directed to their C&C server automatically by querying DNS servers. While a single DNS server, from the perspective of botmaster, may introduce a new vulnerable point (a large amount of query messages from bots may occur simultaneously), a distributed DNS service may reduce this anomaly and thus significantly enhance the survivability of C&C servers.

Another fundamental issue naturally arises is that how bots communicate with their botmaster. In practice, there is no such a compelling need for a bot to create new protocols. Since bots must exploit the vulnerabilities of the system and application programs for breaking in a computer, they may utilize the same protocols as the compromised software. A majority of today's botnets are using the Internet Relay Chat (IRC) protocol, which allows multiple communication modes (e.g., unicast, broadcast, multicast) and data sharing among a large amount of members. In an IRC-based botnet, a new bot should firstly contact IRC server for joining the botnet. To that end, the bot needs to authenticate itself to the IRC server as well as the C&C channel (username and password). If the authentication is successful, the bot is then involved in the botnet and gets privilege to take further actions, e.g., parsing and executing the cannel topic that contains botmaster commands. However, while IRC provides a means of communication for bots and botmaster, the botnets may vary in command sets for distinguishing themselves from the others. A detailed description on IRC-based botnets can be found in[6], and a comprehensive comparison between different versions of IRC-based botnet is reported in[3].

Another popular protocol used by botnets is HTTP. Since HTTP is one of the predominant protocols for today's Internet, botmaster can hide his/her communication with bots into the huge amount of normal traffics, thereby evading most detection systems which are focused on the examination of traffic patterns. In another word, as the application scenarios of HTTP are much more diverse and complex than those of IRC protocol, botnets relying on HTTP may have more chance to survive. For instance, due to the prevalence of IRC-based botnets, most of firewall policies filter out IRC-related traffic, while HTTP traffic can hardly be blocked, though header and payload analysis[52] may reveal some abnormal packets.

While IRC and HTTP protocols gain the most popularity in centralized-based botnets, there are also some other application protocols that can be utilized by distributed botnets, such as IM, P2P, and VoIP protocols. For example, by exploiting the vulnerabilities of P2P file sharing protocols, Phatbot can construct a botnet in any P2P network; the vulnerabilities of MSN messenger and SKYPE may also lend their hosts to bots for constructing random botnets.

2.3 Observation-centric botnet behavior

Since botnets span over the Internet, the first-level observations should be collected from Internet infrastructures, and the behavior metrics should be specified from a global viewpoint. A diurnal model is created in[14] for measuring the botnet propagation, which enables one to compare propagation rates for different botnets and prioritize response. Rajab et al.[46] argued that the effective sizes of botnets rarely exceed a few thousands rather than 350,000 bots as estimated in[14]. They further showed that the botnet-size estimate remains challenging, and a single metric is insufficient for estimating the size of a botnet. Rather, integrating the results from multiple concurrent and independent views of a botnet's behavior may provide more reliable estimation[47]. However, a botnet behavior may vary with its objective, and thus requires different metrics. In[15], three metrics, i.e., effectiveness, efficiency, and robustness, are proposed for measuring the botnet utility for various activities. Another three general metrics, relationship, response, and synchronization are proposed in[1] for detecting botnet, while the metrics are limited to IRC-based botnets that behave in cooperative manner.

Furthermore, in network-based anomaly detection, observation-centric analysis is always the preliminary and fundamental step for designing effective and efficient detection systems[59]. Botnets can be essentially regarded as malware-driven network anomalies, so the observation-centric bot behavior regardless of particular structures and protocols may facilitate our understanding. In particular, the characterization of bot behavioral traces in terms of attack consequence may cover a large class of specific malware variants and strategies. While there is no fundamental difference between the behavior of a bot and any other malware, a system-wide collection and extraction of information flows in terms of particular observable subjects (e.g., network packets, system logs, file systems, memory, etc.), especially those generated during the communication between bot and botmaster, may disclose some special features of botnets. Following the observation categories for measuring system normality introduced in[7], the traces resulted by botnets can be correspondingly characterized as follows,

- Macroscopic level, concerns the long-term average behavior of the network, mainly the coordinated global behaviors of botnets in the Internet.
- Mesoscopic level, examines intranet-level events like traffic patterns and network packets, and mainly explores spatio-temopral characteristics.
- Microscopic level, focuses onexact mechanisms at the kernel-level of atomic operations in the operating system, such as the software programs, individual processes, and system calls.

However, insisting on the isolation of the observations at different levels is neither necessary nor meaningful for the understanding of botnet behavior, so the classification is only from the point of view of system scales for monitoring. More specifically, macro-level botnet behavior can be observed at Internet infrastructures, such as DNS servers[46]. Since most of botmasters employ DNS servers for their bots to resolve the IP addresses of their C&C servers, a huge cluster of queries may occur at a particular DNS server during a certain period. This phenomenon becomes obvious when a botmaster intends to change its C&C server, where all the bots will disconnect from an old C&C server and switch to a new one (with different IP address) by sending DNS queries with a dynamic DNS domain name. That can be observed through the correlation of distributed monitors in the Internet. In addition to the collective DNS querying behavior, botnets with centralized C&C server may trigger anomalous network flows that significantly deviate from normal patterns.

The meso-level observations are usually resulted from the communications between bots, botmaster, attack target, which usually reveals special spatiotemporal properties. Firstly, bots must contact C&C server to fetch binary, a centralized C&C server may attract a large amount of traffic due to the release of binary updates (similar to DNS server which attracts IP address queries). Secondly, since bots are coordinated and controlled by the botmaster, their response to the commands should be simultaneous and with very similar latency (not as diverse as human response). The network traffic then reveals abnormal patterns (e.g., more bursty than normal ones) with respect to statistical properties. Thirdly, in response to the botmaster command, a bot may initiate some connections to a target, the host of this bot then observes suspicious outgoing links that is not triggered by a legitimate system operation. More generally, a group of coordinated bots must have similar communication traffic (uplinks to the botmaster) and attack traffic (downlinks to the target). The two clusters of traffic patterns may share much similarity as well.

The bot-driven observation at micro-level is essentially the same as that of worms, viruses, and other malware variants. A fundamental trait that distinguishes malware from benign software is that its behavior with respect to information accessing and processing are exhibited without the user's acknowledge and consent. While different bots may vary in their impacts on the system elements, they always tend to trigger the fluctuations of system normality, from hardware states to kernel modules. For example, the execution of a bot may trigger abnormal audit events and system log files that are different from the normal profiles. It may also activate application ports and disable anti-virus programs. The execution of a piece of malicious code may generate anomalous process and system call sequences. A bot targeting on web browser can call for a set of API applications, system process, BHO objects whereas a legitimate process never does so. A salient feature for bots, considering their objective, is that they always attempt to initiate outgoing network connections, thereby resulting in anomalous events at network interfaces.

3 Hardening botnets

Botnet is still in its blooming stage, and lots of techniques can be further explored by attackers for hardening botnets. A careful examination shows that most of the existing botnets suffer from three vulnerabilities: C&C server structure, communication protocol, and observable bot-driven trace. So the next-generation of botnets may focus on the improvement of those three aspects.

3.1 Enhancing robustness of C&C model

Obviously, a botnet with only one C&C server can be easily detected and tracked. While the survivability of a botnet with centralized C&C model can be increased by using more C&C servers, the failure curse is not essentially ruled out. Some botnets turn to use P2P-based and random C&C models, as introduced in Section **2.1**, making C&C servers harder to be detected and tracked. However, their construction and maintenance become non-trivial. It then comes to obvious that a better C&C model is the one which can achieve the best trade-off between the robustness and the feasibility of construction and maintenance, if the effectiveness, scalability and latency are guaranteed to some extent.

A hybrid P2P botnet is proposed in[53],

which removes the C&C mechanism by using a set of sensor host (servant bots) that are included in the bots' peer list (size-constant). The communications between botmaster and bots are relayed by servant bots, which are also in charge of network maintenance and bot updates. To make the communication secure, each servant bot generates its own symmetric encryption keys for incoming connections from other bots. The servant bots in this scheme play the role of C&C severs, with a strong assumption that such bots must posses hard-coded IP addresses that are accessible from the Internet, which is not essentially different from the traditional C&C severs. One promising feature of this scheme is that the C&C server list maintained by each bot can be periodically updated and exchanged with other bots, enabling the botnet to be tolerant to the failure of a fraction of servant bots. Also, since the peer list contains partial information about C&C servers and only a small portion are exchanged each time, the botnet can be resilient to be tracked even some bots are captured. It is true that such botnets are more robust than the botnets employing obvious C&C mechanism. However, like other P2P-based botnets, they suffer from low scalability and high latency issues. In practice, the availability of servant bots also dramatically impedes the growth of a botnet, as reported in [29], the number of C&C servers are only hundreds per day.

A tree-structured algorithm is developed in[51], which is used to construct a super-botnet by generating and combining a collection of small-scaled botnets controlled by independent C&C severs. In particular, three parameters are set in advance by the botmaster for creating his/her botnets, i.e., the number of sub-botnet, the size of sub-bot, and the number of new bots a bot need to create. The algorithm essentially contains two stages: creates new C&C severs (each controls a sub-botnet) and then populates each sub-botnet. To enhance the robustness of the super-botnet, C&C severs are intentionally isolated and placed as far away from each another as the balanced tree structure allows. Theoretically, the algorithm works well provided that the bot compromising rate is high and C&C severs are readily available. Since a bostmaster has no way to obtain a whole picture about the botnet, his/her commands are routed between the sub-botnets (then further to the bots in each sub-botnet) instead of traversing among the entire botnet directly. To make this process secure, asymmetric encryption keys are used for inter-sub-botnet communications and symmetric keys are used for intra-sub-botnet communications. Because of this, the maintenance and update of a botnet are not as easy as its construction. Also, it is obvious that the latency is always high and an attack could not be effective only when the sub-botnets are coordinated well.

The two designs illustrate us the clear idea about the intent of the next generation botnets, that is, removing the weakness point of botnet cause by C&C mechanisms. Thus, the more advanced C&C design will provide robust network connectivity, control traffic dispersion, and resilient to tracking, meanwhile facilitates the monitoring and maintenance by the botmaster. However, due to the easy implementation and maintenance, as well as its effectiveness (as shown in Table 1), IRC-based centralized botnet is still a popular variant, while its integration with the upcoming C&C mechanisms will bring more threat.

3.2 Obfuscating communication channels

Regardless of the C&C model that a botnet uses, the communication channel between botmaster and bot is another vulnerable spot for botnets, since it may reveal the bot traces and expose the command messages to be intercepted, interpreted, and manipulated. To evade that, a sophisticated botmaster will obfuscate the communication channels and hide communications. In most cases, communication obfuscation employes authentication, authorization, and encryption techniques, varying with the C&C structure. A very simple case is in an IRC-based botnet, where a bot firstly needs to authenticate itself to the IRC server to initiate a session, and then a password is required to join in a communication channel. In some cases, the authorization is also necessary for a botmaster to issue his/her commands[46] in order to prevent other botmasters from overtaking his/her botnet.

Generally speaking, the goal of communication obfuscation is to make botnet traffic have normal appearance and communications hard to be interpreted[23]. One straightforward way is to disperse botnet traffic by randomizing service ports[53]. By doing so, botnet traffic directed to a C&C server can be distributed to a range of ports (usually standard ports, like SSH, HTTPS) rather than a single one that can be easily detected by a port-specific detector. This trick has high requirement on bot capability, since usually a bot variant targets on a particular port. A botmaster can even use convert channels like TCP and ICMP tunneling and IPV6 tunneling[28] to hide their communication traces.

Another family of powerful botnets use the state-of-the-art encryption techniques to obfuscate the communications between bot, C&C server, and bot-master. This category of obfuscation techniques heavily depends on the C&C structure as well as the botmaster's intents. For the centralized C&C mechanism, asymmetric encryption techniques are more suitable and efficient: the key pairs $< K^+, K^- >$ are generated by a botmaster, where public key K^+ is embedded into a bot's program during its propagation and private key K^- is used as signature to send the subsequent commands. For a botnet with distributed C&C servers, symmetric encryption schemes have more advantage in that each bot needs to include a group of, rather than a single, C&C servers in its list $Server\_list = \{(Server_1, k_1), (Server_2, k_2), \ldots, (Server_n, k_n)\}$ (where $Server_i$ is the identity of C&C server i, and k_i is the key issued by i). The keys stored in the bot therefore allow it to contact any server it prefers in the list and resilient to the reverse-engineering, because the keys are specialized for it and their leaking does not necessarily disclose the identity of other C&C servers that are not included in the list. A more sophisticated example was discussed in[51], both public-key cryptography and secret key cryptography are applied for securing tree-structured C&C servers. In particular, public-key cryptography is used for securing the conversations between the botmaster and all the C&C servers, and secret key cryptography is used for securing the communications between C&C server and the sub-botnet it controls. To ensure the integrity of command messages, secret splitting scheme is used among C&C servers, with which an encrypted command can be executed if and only if each piece of it have been successfully decrypted by the corresponding sub-botnet (each sub-bot can only decrypt one piece of the command using its public-key piece). A pre-defined time bomb, traversing the entire botnet as a command message, is used to coordinate the actions of all the bots.

Considering the evolution trend of today's botnets, it is not surprising that more complex encryption techniques would be used with the advent of more sophisticated C&C structures. However, the intrinsic instability of botnets impels the botmaster to adopt more practical schemes rather than the ones requiring strong assumptions for implementation. In order to improve the scheme in[51], we propose a harder betnet called bot-enclave (the idea is similar with intrusion tolerant enclaves[16]), which is resilient to failures of C&C servers that are with Byzantine behavior. Bot-enclave combines Byzantine fault tolerant protocols with secret sharing techniques, and enables a botmaster to keep his/her botnet survive if the number of the failed C&C servers meets the condition $f < [\frac{n-1}{3}]$, where n is the total number of C&C servers that the botmaster possesses. It is also possible to construct hierarchical bot-enclave by combining a group of independent bot-enclave together. However, the response latency of bots and enclave maintenance will turn to a considerable issue.

While encryption helps botnets to significantly obfuscate their communications, it is actually a two-sided sword considering the other metrics in Table 1. On the one hand, the more complex of the encryption schemes the more secure of a botnet. On the other hand, a complex encryption scheme requires the botmaster to carefully design an appropriate key management method, requiring sophisticated time scheme to synchronize the coordinated bots' behavior, therefore impedes the botnet's effectiveness and maintenance. As a means of obfuscation, encryption technique is important, but not all. For hardening botnet, a sophisticated botmaster would never solely focuses his/her attention on obfuscating communications. Another important factor determining the application of encryption techniques in a botnet is the botmaster's intent. For example, if the objective of a botmaster is to launch DDoS attacks or spamming, simple obfuscation techniques are sufficient, as response and coordination are the most important concerns in this case. However, for a botmaster who intends to harvest information for financial gains, since he/she wants his/her botnet to live as long (and silent) as possible for collecting more information instead of dismissing shortly after attack like a DDoS, the better obfuscation the more profit he/she probably gets.

3.3 Other evasion techniques

The introduced evasion techniques are proactive, since the botmaster intends to ensure his/her botnet to be dependable and secure before he/she really launch attacks. Another category of evasion technique is reactive in essence. Rather than actively harden his/her botnet beforehand, a botmaster is forced to take measures to avoid and cope with the anti-botnet techniques.

Honeynet[26] is one of the most popular and effective techniques for analyzing botnet (we will give more discussion in the Section **4.1**), and has attracted not only security professionals' efforts but also attackers' attention. In case of being caught and analyzed, a sophisticated attacker is motivated to design a family of bots that are aware of the existence of honeynet or other Internet security sensors. A series of anti-honeypot techniques have been extensively discussed in[24][41]. In general, the detection target, from the perspective of an attacker, can be either hardware like VMware and other emulated virtual environment[10], or software like the faulty response of honeypot program. For instance, a commercial anti-honeypot spamming tool was proposed in[36], which aims at detecting honeypot open proxies by testing whether the remote open proxy can send email back to the spammer. A probe response attack technique was proposed in[4] for locating the Internet sensors using their published data. Another fact exploited by the attacker is that the application of a honeypot may trigger legal issues[49], so the honeypot can not really send out attack traffic due to privacy and liability constraints, and they are usually blocked from out-going or directed to a particular target instead. Supported by this observation, an advanced honeypot-aware botnet was designed in[60]. In particular, a botmaster can intentionally command his/her bots (especially those suspected to be honeypots) to actively send out malicious traffic to other compromised hosts (botnet sensors), and then determine whether a bot is a honeypot or an actual victim by observing the outgoing traffic originated from the suspected bot.

Another category of evasion techniques targets at detection systems, and are frequently used at host-level by bot (or malware for more generality) writers. A simple variant is to turn off the anti-virus software installed in the computer once its execution encounters particular environment or instructions. The more sophisticated malware aims at manipulating the operational algorithms of the detector. In particular, malware detectors can be either signature-based or anomaly-based, while both of them can be evaded by the attackers: program obfuscation enables malware to be polymorphic or metamorphic[11][40], and even allows the attacker to deliberately mislead the detector's signature generation[42]; anomaly-based detectors are susceptible to mimicry attacks[5][35], which blend attack traces with normal ones or emulate normal user behaviors. By using evasive bots, a botmaster can improve success rate of compromising new host and extend the botnet's life.

4 Breaking botnets

It is a mission impossible for our computer networks to be totally immune to botnets due to the increasing connectivity and complexity of today's computer systems, as well as the rapid emergence of malware variants. While lots of anti-botnets techniques have been developed, none of them is a silver bullet. A more practical alternative, therefore, is to integrate the appropriate techniques together for constituting an in-depth defense boundary. According to the role in defending against botnets, the techniques are basically classified as prevention, detection, and tracking techniques. According to the data source of observations, the detection technique falls into two categories, i.e., host-based detection systems and network-based detection systems. Furthermore, the design scheme classifies detection systems as misuse-based (signature-based) and anomaly-based (profile-based) systems. With the challenges in mind, this section aims at developing a top-down analytical framework as a basis for comparative studies and critical evaluation on the existing anti-botnet techniques. Based on that, we then propose a holistic methodology for breaking botnet.

4.1 Botnet prevention

Like firewalls and IDS, Honeypot has been taken as one of security counterparts in today's computer network by intentionally set as a trap by the security administrator[2][13][17][32]. It generally has three purposes[45]: distracting adversaries from more valuable machines on a network, providing early warning about new attack and exploitation trends, and allowing in-depth examination of adversaries during and after exploitation of a honeypot. Two or more honeypots then constitute a honeynet. While authentication, authorization, and encryption techniques traditionally serve as security prevention measures, in our holistic methodology defending again botnets, honeypot is positioned in the first defense line due to its capability of attracting and capturing bots.

Honeypot has been successfully applied to worm detection[13][50], and now its role slightly changes with the prevalence of botnets and other malware variants. There is a variety of honeynet demos and tools have been developed, such as honeyd[45], mwcollect & nepenthes[37] (the two merged already), and honey-trap[27]. In particular, honeyd provides a basis for simulating virtual computer systems at the network level and thus provide large-scale honeynet monitoring on the network with OS-various hosts and arbitrary routing topologies. Nepenthes are mainly used to collect malware by extracting the malware binaries from the exploit payload using known patterns and downloading samples. Honeytrap examines TCP connection attempts by dynamically creating port listeners, and aims at dealing with novel malware.

However, as discussed in Section **3.3**, honeynet has become one of the targets of attackers, and can be possibly evaded by sophisticated malware. As a remedy, honeypot should be integrated with automated malware detectors which take outputs of honeypots for generating and updating attack signatures. Also, the automatic data patch generation for unknown vulnerabilities following malware detectors may enable a network to be immune to novel attacks. In doing so, a defense boundary thus can be built for preventing bots from propagation and infection, hopefully restraining botnet construction and growth.

4.2 Network-based botnet detection and tracking

NIDS has been evolving more than two decades, aiming to detect attacks by monitoring network traffic patterns and examining network packets. However, the special characteristics of observations resulted by botnets, as introduced in Section **2.3**, call for some redesigns of existing NIDS.

Since botnet is an Internet-scale problem, Internet infrastructures naturally lies at the highest level for botnet detection and tracking.

However, the global behavioral characteristics of botnets basically rely on their C&C structures and mechanisms, and they would keep unchanged unless the infrastructure changes. For example, DNS service serves an important role for tracking C&C centralized botnet, since bots switch C&C address and domains frequently, resulting in anomalous address queries and traffic patterns. Some interesting findings are reported in[33][46]. Generally speaking, it is a compelling need to distribute security monitors through the interested spots, and to further develop distributed algorithms and techniques for correlating the monitoring results. While the design principle sounds feasible, there is no practical tools, architecture, and methodology have been enforced, which are impeded by two factors: firstly, large-scale data collection and analysis is time intensive and human effort consuming; secondly, some ISPs are reluctant to be involved in the tracking process which may bring them unnecessary maintenance cost and impact their quality of service.

A lower level botnet detection focuses on enterprise-like networks, and most of existing research efforts concentrate at this level. The most significant observation supporting the design of botnet detection and tracking techniques is the special spatio-temporal property of botnets, i.e., multi-stage and coordinated, revealed from the traffic patterns between bots, C&C server, and attack targets. In[21], a set of statistical algorithms was used to explore the spatio-temporal correlation in network traffic by observing that bots controlled by the same bot-master engage in coordinated communication, propagation, attack, and fraudulent activities. To break the limits of the method to IRC-based botnet, the same authors developed another detection framework that is independent of botnet C&C structure and protocol for clustering similar communication traffic (bot to C&C server) and malicious traffic (bot to target system), and then further identify the bots lying in the intersections of the two clusters[22]. In addition to botnet detection, honeynet serves as a useful tool for botnet tracking[2][17], which can be used to conduct root-cause analysis of DDoS attacks launched by botnets.

The introduced detection systems, however, can be only effective when the traffic patterns are observable and deviate significantly from the normal ones. If the botnet traffic patterns appear normal, e.g., in P2P networks, and the traveling packets are encrypted, more fine-grained analysis and detection schemes like the ones introduced in[43][57] are needed.

4.3 Host-based botnet detection

Hosts are the end of a botnet, so a direct approach to disrupting botnets is to make hosts immune to infection by using HIDS. Similar to NIDS, the observations caused by bots have distinct features from the other attack variants, e.g., bot and botmaster has interactive communications, so traditional HIDS can be no longer effective and have evolved to specific malware/bot detectors.

The design principe of malware detector is to capture system-wide information flow within a host triggered by malware[56]. To that end, the most useful tool is Virtual Machine, which can be used to analyze the malare samples caught by a honeypot and generate malware signatures[5][11][42]. More specifically, malware code analysis can be either static or dynamic depending on whether the malware is actually executed or not[38]. Obviously, a malware detector is expected to detect not only known malware, but also the novel ones and evasive ones.

However, although some of the malware detectors are useful for bot detection, their efficiency will be undermined if the distinct features of bots are out of concern. The basic idea can be illustrated as the design in[12], even though the detector is not specifically designed for bots. It captures user unintended malicious outbound connections (extrusions) by correlating outbound connections with user-driven input at the process level. The user-driven input can be further extended to system-driven, as in[34], the spyware-like behavior is characterized by exploring the relationship between BHO and toolbar interface events with user browsing behavior. With the similar design rationale, and by examining the observations emitted in the process of malware infection, Gu et al.[20] developed a BotHunter for correlating a number of IDS sensors working at different stages of the malware by tieing together the conversation trail of inbound intrusion alarms with the outbound communication patterns indicating host infection.

It then comes to obvious that an effective host-based bot detector should not only limit its observations within a host. Rather, it needs to solve a fundamental problem, i.e., whether what observed inside a host and outgoing links are really triggered by the incoming links or legitimate user-and system-driven behaviors.

4.4 A holistic methodology for breaking botnets

Intuitively, an effective detection and tracking scheme of botnets needs a perfect behavioral profile capturing and characterizing all the information flows caused by botnets. An ideal approach is the one which is able to monitor, extract, and correlate all the related observations at different system levels, and explore all the essential features, such as spatial, temporal, sequential, and frequency-based properties, etc. With the botnet characteristic in mind, on the basis of observations introduced in Section **2.3**, and from an architectural viewpoint, it is straightforward to envision such a methodology for correlating and integrating the observations and the reports of anti-botnet tools at different layers, i.e., Internet, intranet, and host, for achieving a whole snapshot of the botnet. The ultimate goal is to conduct botnet detection and tracking, as well as the root-cause analysis of botnet-driven anomalies.

An integrated framework for botnet detection is shown in Fig. 3, where the tools and components are only for easier illustration and does not necessarily mean their application and implementation in particular environment. Following the top-down analytical framework, a holistic methodology is given as follows,

- Developing and deploying distributed Internet threat monitors and security sensors for global monitoring as the work reported in[4][25][31][48], such as and then designing distributed algorithms for analyzing cross-site suspicious observa-

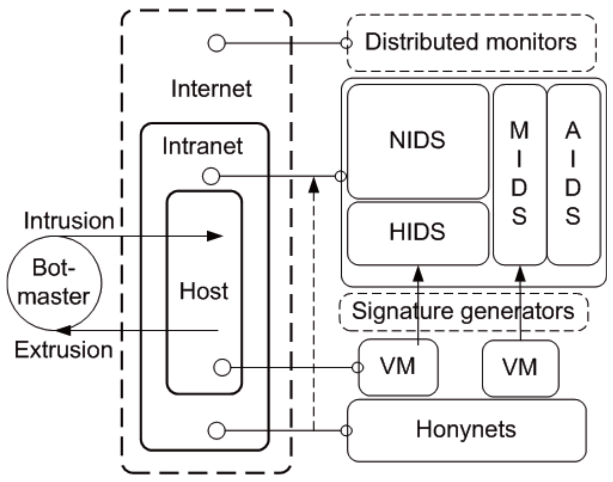

Fig.3 An integrated framework for Botnet detection. shown are behavior trails (arrow lines), observation monitoring and analysis (arrow lines with circle end), and security components (solid blocks for suggested delopyment, and dashed ones for optional deployment). Bots usually have an intrusion process and an extrusion process, leaving traces at three layers: host, intranet, and Internet.

tions occurring at Internet infrastructures,
- Designing and applying clustering (cross-clustering) algorithms[22][55] and statistical algorithms like Bayes inference for analyzing NIDS sensors and Honeynet reports (which specifically consider the conversions between botmaster and bots, i.e., intrusion and extrusion process,[20]), with the objective to pinpoint compromised hosts, track botnet and infer root-cause of attacks, such as the spammer of spamming attacks[54], origin of DDoS attacks[17],
- Developing malware detectors for host-based and fine-grained bot analysis, along with the automated signature generators for extending and updating attack signatures of HIDS[8][39].

Although the framework contains three layers, they are tightly integrated in essence, since the bot trail and attack consequence occur at the three layers simultaneously rather than independently. Also, there is not always a compelling need to develop novel detection schemes, the state-of-the-art tools like Snort[30], PAYL[52] may contribute to the detection of anomalous network packets associated with bot traces.

Another important issue is the response to the discovered botnet. As botnet is a growing industry and attracts the increasing number of botnet creators and managers in black markets[18], botnet detection and tracking can never been solely treated as a technical issue by security professionals (e.g., removing bots, updating system patches and bot signatures of AVs), it also turns to a serious social issue on Internet criminals. Thus, the response to botnet is not limited to the removal of bots in a particular victim, it also involves the global cooperation among ISPs (e.g. DNS service) and coordination of law enforcement in different countries, because a botnet can span over the Internet with bots extensively distributed in different spots.

5 Conclusion

Botnet becomes one of the serious threats to our cyberspace due to its persistence, broad coverage, and malicious intents. Taking the characteristic as a basis for analysis and understanding, this paper explored the next-generation botnets with respect to their features and evasion techniques. We then reviewed the existing anti-botnet techniques, and proposed a holistic methodology to integrate the appropriate techniques together for achieving an in-depth defense line for detecting and disrupting botnets.

As the botnet evolves to be more sophisticated, in the next stage, we intend to examine the potential measures and techniques for hardening botnet, as the fundamental understanding on the adversary's behavior and the discoveries in advance are always the essential ways for developing and designing effective countermeasures. Our ultimate goal is to develop a systematic approach, following the methodology presented in this paper, to infer the root-cause of botnets based on observations collected at different system levels and reports of bot detectors deployed at different spots, and eventually provide convincing evidence for botnet ditection, tacking and law enforcement.

References

1 M. Akiyama, T. Kawamoto, M. Shimamura et al., “A proposal of metrics for botnent detection based on its cooperative behavior”, In Proc. of the 2007 International Symposium on Applications and Internet Workshops (SAINTW'07), 2007.

2 P. Bacher, T. Holz, M. Kotter, and G. Wicherski, “Know your enemy: tracing botnest”, http://www.honeynet.org/papers/bots/

3 P. Barford, and V. Yegneswaran, “An inside look at botnets”, In Proc. of Special workshop on malware detection, Advances in Information Security, 2006.

4 J. Bethencourt, J. Franklin, and M. Vernon, “Mapping Internet Sensors With Probe Response Attacks”, In Proc. of USENIX Security Symposium, pp.193-208, Aug. 2005.

5 K. Borders, X. Zhao and A. Prakash, “Siren: catching evasive malware”, In Proc. of IEEE Symposium on Security and Privacy (S&P’06), May 2006.

6 D. Brumley, “Tracking hackers on IRC”, http://www.indonesiajakarta.org/home/

7 M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, “Measuring system normality”, ACM Transactions on Computer Systems, Vol.20, No.2, pp.125-160, May 2002.

8 J. Caballero, S. Venkataraman, P. Poosankam, et al., FiG: Automatic Fingerprint Generation, In Proc. of NDSS, Feb. 2007.

9 E. Cooke, F. Jahanian, and D. McPherson, “The Zombie roundup: understanding, detecting, and disrupting botnets”, In Proc. of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet (SRUTI 05), Jun. 2005.

10 J. Covery, “Advanced Honey Pot Identification And Exploitation”, http://www.phrack.org/fakes/p63/ p63-0x09.txt, Jan. 2004.

11 M. Christodorescu, S. Jha, S. A. Seshia, et al., “ Semantics-aware malware detection”, In Proc. of IEEE Symposium on Security and Privacy (S&P’05), pp.32-6, May 2002.

12 W. Cui, R. H. Katz, and W. Tan, “Design and Implementation of an Extrusion-based Break-In Detector for Personal Computers”, In Proc. of The 21st Annual Computer Security Applications Conference (ACSAC 2005), Dec. 2005.

13 D. Dagon, X. Qin, G. Gu, et al., “Honeystat: Local worm detection using honeypots”, In Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004), pp.39 58, Sep. 2006.

14 D. Dagon, C. C. Zou, and W. Lee, “Modeling botnet propagation using time zones”, In Proc. of the 2006 Network and Distributed System Security Symposium(NDSS 2006), pp.235-249, Feb. 2006.

15 D. Dagon, G. Gu, C. P. Lee, and W. Lee, “ A taxonomy of botnet structures”, In Proc. of the Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp.325-339, Dec. 2007.

16 B. Dutertre, V. Crettaz and V. Stavridou, “Intrusion-Tolerant Enclaves”, In Proc. of IEEE Symposium on Security and Privacy (S&P’02), pp.216-226, May 2002.

17 F. C. Freiling, T. Holz, and G. Wicherski, “Botnet tracking: exploiring a root-cause methodogy to prevent DDoS attacks”, In Proc. of 10th European Symposium On Research In Computer Security (ESORICS 2005), pp.319-335, Sep. 2005.

18 N. Friess, J. Aycock, and and R. Vogt, “Black Market Botnets”, In MIT Spam Conference, 2008, pp.1-8, 2008.

19 D. Geer, “Malicious bots threaten network security”, IEEE Computer, Vol.38, No.1, pp.18-20, Jan. 2005.

20 G. Gu, P. Porras, and V. Yegneswaran, et al., “BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation”, In Proc. of the the USENIX Security Symposium (Security’07), Aug. 2007.

21 G. Gu, J. Zhang, and W. Lee, “BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic”, In Proc. of the 15th Annual Network and Distributed System Security Symposium (NDSS’08), Feb. 2008.

22 G. Gu, R. Perdisci, J. Zhang, and W. Lee, “BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnent Detection”, In Proc. of the the USENIX Security Symposium (Security’08), Aug. 2008.

23 M. Handley, C. Kreibich,, and V. Paxson, “Network Intrusion detection: Evasion, Traffic Normalization, and End-to-End Protocols Semantics”, In Proc. of the the USENIX Security Symposium, Aug. 2008.

24 T. Holz, and F. Raynal, “Defeating Honeypots: System Issues (Part 1,2)” SecurityFocus InFocus Article, Sep. 2004

25 CAIDA Telescope analysis, http://www.caida.org/analysis/security/telescope/

26 http://www.honeynet.org/

27 http://honeytrap.mwcollect.org/

28 http://www.uscert.gov/reading room/IPv6Malware-Tunneling.pdf

29 http://atlas.arbor.net/summary/botnets/

30 http://www.snort.org/

31 Internet Storm Center (ISC), http://isc.sans.org/

32 X. Jiang, and D. Xu, “Collapsar: A VM-Based Architecture for Network Attack Detention Center”, In Proc. of the 13th USENIX Security Symposium, pp.1528, 2004.

33 A. Karasaridis, B. Rexroad, and D. Hoeflin “Wide-scale botnet detection and characterization”, In Proc. of the First Workshop on HotTopics in Understanding Botnets (HotBots07), Apr. 2007.

34 E. Kirda, C. Kruegel, G. Banks, et al., “Behavior-based Spyware Detection”, In Proc. of the 15th USENIX Security Symposium, pp.273-288, 2006.

35 C. Kruegel, E. Kirda, D. Mutz, et al., “Automating mimicry attacks using static binary analysis”, In Proc. of the 14th USENIX Security Symposium, pp.161-176, 2005.

36 N. Krawetz, “Anti-Honeypot Technology”, IEEE Security& Privacy Magazine, Vol.2, No.1, pp.76-79, Jan/Feb. 2004.

37 http://www.mwcollect.org/

38 A. Moser, C. Kruegel, and Engine Kirda, “Exploring multiple exectuion paths for malware analysis”, In Proc. of IEEE Symposium on Security and Privacy (S&P’07), May 2007.

39 J. Newsome, B. Karp, and D. Song, “Polygraph: Automatic Signature Generation for Polymorphic Worms”, In Proc.of IEEE Symposium on Security and Privacy (S&P’05), May 2005.

40 C. Nachenberg, “Computer virus-antivirus coevolution”, Communiations of the ACM, Vol.40, No.1, pp.46-51, Jan. 1997.

41 L. Oudot, and T. Holz, “Defeating Honeypots: Network Issues (Part 1, 2)” SecurityFocus InFocus Article, Sep. 2004

42 R. Perdisci, D. Dagon, W. Lee, et al., “Misleading worm signature generator using deliberate noise ijnection”, In Proc. of IEEE Symposium on Security and Privacy (S&P’06), May 2006.

43 M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos, “NetworkLevel Polymorphic Shellcode Detection Using Emulation”, In Proc. of Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2006), pp.54-73, 2006.

44 R. Puri, “Bots &botnet: an overview”, http://www.sans.org/reading room/ whitepapers/malicious/1299.php, Sep. 2004

45 N. Provos, “A Virtual Honeypot Framework”, In Proc. of the 13th USENIX Security Symposium, Aug. 2004.

46 M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “A Multifaceted Approach to Understanding the Bot-net Phenomenon”, In Proc. of the 6th ACM SIGCOMM conference on Internet measurement (IMC’06), pp.41-52, Oct. 2006.

47 M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging”, In Proc. of the First Workshop on HotTopics in Understanding Botnets (HotBots07), Apr. 2007.

48 Y. Shinoda, K. Ikai, and M. Itoh, “Vulnerabilities of Passive Internet Threat Monitors”, In Proc. of USENIX Security Symposium, pp.209-224, Aug. 2005.

49 Lance Spitzner, “Honeypots: Are They Illegal?” SecurityFocus InFocus Article, Jun. 2003

50 Y. Tang, and S. Chen, “Defending against internet worms: a signature-based approach”, In Proc. of the IEEE INFOCOM, May 2006.

51 R. Vogt, J. Aycock, and M. J. Jacobson, Jr., “Army of Botnets”, In Proc. of the 2007 Network and Distributed System Security Symposium(NDSS 2007), pp.111-123, Feb. 2007.

52 K. Wang, S. J. Stolfo, “Anomalous payload-based network intrusion”, In Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004), pp.203 222, Sep. 2004.

53 P. Wang, S. Sparks, and C. C. Zou, “An advanced hybrid peer-to-peer botnet”, In Proc. of the First Workshop on HotTopics in Understanding Botnets (HotBots07), Apr. 2007.

54 Y. Xie, F. Yu, K. Achan, et al., “Spamming Botnet: Signatures and Characteristics”, In Proc. of SIGCOMM’08, Aug. 2008.

55 T. F. Yen, and M. K. Reiter, “Traffic aggregation for malware detection”, In Proc. of DIMVA 2008, pp.207-227, 2008.

56 H. Yin, D. Song, M. Egele, et al., “Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis”, In Proc. of the 14th ACM Conference on Computer and Communications Security (CCS’07), Oct. 2007.

57 Q. Zhang, D. S. Reeves, P. Ning, and S. P. Iyer, “ Analyzing Network Traffic To Detect Self-Decrypting Exploit Code”, In Proc. of the 2nd ACM symposium on Information, computer and communications security (AsiaCCS’07), pp.4-12, 2007.

58 Z. H. Zhang, P. -H. Ho, X. Lin, and H. Shen, “Janus: A Two-Sided Analytical Model for Multi-Stage Coordinated Attacks”, In Proc. of the 9th International Conference on Information Security and Cryptology (ICISC2006), LNCS 4296, pp.136-154, Dec. 2006.

59 Z. H. Zhang, “Adaptive Observation-Centric Anomaly-based Intrusion Detection: Modeling, Analysis and Evaluation”, Ph.D thesis, JAIST, Mar. 2006.

60 C. C. Zou, and R. Cunnigham, “Honeypot-aware advanced botnet construction and maintenance”, In Proc. of Int. Conf. on Dependable Systems and Networks (DSN2006), pp.199-208, Jun. 2006.

***Zhang Zonghua**, Ph.D.*

Expert Researcher, Traceable Secure Network Group, Information Security Research Center

Networks Security

***KADOBAYASHI Youki**, Ph.D.*

Guest Expert Researcher, Traceable Secure Network Group, Information Security Research Center

Network Security