RatBot: Anti-Enumeration Peer-to-Peer Botnets

Guanhua Yan[1], Songqing Chen[2⋆], and Stephan Eidenbenz[1]

[1] Information Sciences (CCS-3)[⋆⋆]
Los Alamos National Laboratory
[2] Department of Computer Science
George Mason University

Abstract. As evidenced by the recent botnet turf war between SpyEye and Zeus, the cyber space has been witnessing an increasing number of battles or wars involving botnets among different groups, organizations, or even countries. One important aspect of a cyber war is accurately estimating the attack capacity of the enemy. Particularly, each party in a botnet war would be interested in knowing how many compromised machines his adversaries possess. Towards this end, a technique often adopted is to infiltrate into an adversary's botnet and enumerate observed bots through active crawling or passive monitoring methods.
In this work, we study potential tactics that a botnet can deploy to protect itself from being enumerated. More specifically, we are interested in how a botnet owner can bluff the botnet size in order to intimidate the adversary, gain media attention, or win a contract. We introduce RatBot, a P2P botnet that is able to defeat existing botnet enumeration methods. The key idea of RatBot is the existence of a fraction of bots that are indistinguishable from their fake identities. RatBot prevents adversaries from inferring its size even after its executables are fully exposed. To study the practical feasibility of RatBot, we implement it based on KAD, and use large-scale high-fidelity simulation to quantify the estimation errors under diverse settings. The results show that a naive enumeration technique can significantly overestimate the sizes of P2P botnets. We further present a few countermeasures that can potentially defeat RatBot's anti-enumeration scheme.

1 Introduction

Due to its open nature, the cyber space has been witnessing a growing number of battles or wars among different groups, organizations, or even countries. The recent botnet turf war between SpyEye and Zeus fighting for bots [7] suggests that botnets can play an important role in cyber warfare. In a real battle or war, it is crucial for each party to know the attack capacities of his adversaries. Similarly, in a cyber war involving botnets, a party would be interested in estimating accurately how many compromised machines his opponents possess.

Currently, a commonly adopted approach to estimating botnet sizes is to infiltrate into an adversary's botnet and enumerate observed bots through either active crawling or passive monitoring methods [13, 12]. Different techniques

⋆ Songqing Chen [illegible]This work] is partially supported by NSF grant CNS-0746649 and AFOSR grant FA9550-09-1-0071.

⋆⋆ Los Alamos National Laboratory Publication No. LA-UR 10-03929 have been used to enumerate existing botnets, such as the Storm botnet, and they sometimes led to inconsistent results, spanning from 500,000 [13] to 50 million [25]. Despite technical challenges such as NAT and DHCP that render it difficult to estimate botnet sizes accurately, advanced techniques can be applied to sift out these effects. For instance, the passive enumeration approach proposed by Kang *et al.* can enumerate bots sitting behind a firewall or a NAT [13], and the UDmap algorithm developed by Xie *et al.* [29] helps mitigating the effects of dynamic IP addresses when enumerating bots based on their IP addresses.

In this work, however, we aim to address a more fundamental question: *can a botnet be intelligently designed so that accurately estimating its size is inherently difficult?* Particularly, we are interested in exploring potential tactics that a botmaster can use to *bluff* his botnet size. In a cyber battle, overestimating the size of the adversary's botnet can lead to the effect of intimidation: a party at a disadvantageous position can deploy this tactic to scare off a stronger opponent. In another example, a party can use this tactic to trick his adversary into using an overly high amount of resources to defend against an attack launched from one botnet so that he would hold advantage over his adversary in a different cyber battle that takes place simultaneously. Furthermore, when two botnet owners compete for the same customer who wants to use the larger botnet for, say, spamming or DDoS attacks, one botnet owner may apply the bluffing tactics to get the bid. Sometimes, a botnet owner may want his botnet size to be overestimated so that he can draw some media attention.

To study the power of such bluffing tactics, we design a hypothetical botnet called *RatBot*, which protects itself from being enumerated. RatBot employs the peer-to-peer (P2P) structure to improve its resilience against a single point of failure. The key idea of RatBot is the existence of a fraction of bots that are indistinguishable from their fake identities, which are spoofing IP addresses they use to hide themselves. RatBot prevents adversaries from inferring its size even after its executables are fully exposed. This is done with heavy-tailed distributions to generate the number of fake identities for each bot so that the sum of observed fake identities converges only slowly and thus has high variation.

Due to its anti-enumeration mechanism by design, RatBot distinguishes itself from those technical challenges (e.g., NAT and DHCP) making it difficult to enumerate bots accurately and is thus immune to existing solutions that aim to address these challeges. The wide deployment of NAT actually leads to underestimation of botnet sizes, which is contrary to the design goal of RatBot. Another distinguishing feature is that the degree to which RatBot can bluff about its size is controllable by the attacker. This is ideal in some situations (e.g., cyber war) where the attacker wants to adjust his bluffing tactics dynamically.

To study the practical feasibility of RatBot, we implement it using the actual development code of aMule, a P2P client software that uses KAD for its P2P communications [2]. We further develop a distributed simulation testbed to evaluate the effectiveness of RatBot in misleading botnet size estimation. We perform a variety of tests with different settings and the results show that a naive botnet enumeration approach by counting the IP addresses observed from the P2P botnets could significantly overestimate their sizes.

The remainder of this paper is organized as follows. Section 2 presents related work and Section 3 gives the threat model. In Section 4, we discuss the design of RatBot, and provide the rationale of such design in Section 5. We introduce the implementation of RatBot in Section 6 and use large-scale simulation to evaluate its performance in Section 7. In Section 8, we further discuss potential countermeasures against RatBot and draw concluding remarks in Section 9.

2 Related Work

Behaviors of real-world botnets have been analyzed to provide insights into how botnets operate in reality [4, 12, 13]. Complementary to these efforts, our work sheds light on the potential challenges regarding enumerating zombie machines in P2P botnets accurately. In spirit, our work is similar to that of Rajab *et al.* [18] as both explore the challenges of estimating botnet sizes, but ours focuses on P2P botnets rather than IRC botnets. Some previous work has shown that multiple factors contribute to inaccurate botnet size estimation, including DHCP and NAT effects [24]. Our results show that even if advanced techniques are deployed to sift out these effects [13, 29], the botnet can still adopt sophisticated obfuscation techniques to make it a difficult task to estimate its size accurately.

A plethora of botnet detection techniques have been developed recently. Gu *et al.* have proposed a series of bot detection methods exploiting spatial-temporal correlation inherent in bot activities [11, 10]. Other botnet detection techniques include DNS-based methods [19], ISP-level analysis [14], signature-based approaches [9], and flow-level aggregation and mining [31]. Our work is orthogonal to these efforts and focuses on anti-enumeration tactics.

Hypothetical botnets proposed previously include Super-Botnet [27], Overbot [20], AntBot [30], and hybrid P2P botnets [28]. Our work differs from these work on two aspects. First, our work focuses specifically on hypothetic P2P botnets that aim to inflate the adversary's estimation of botnet sizes. Second, we have used large-scale high-fidelity simulation to quantify the estimation errors under diverse settings rather than present the design from a conceptual level.

The design and implementation of RatBot presented in this work is based on the Storm botnet, which used the KAD protocol. Besides the Storm botnet, a few other botnets also applied the P2P protocol to organize their bots, such as Nugache [26], Waledac [23], and Conficker [17]. Although none of these botnets have applied anti-enumeration techniques to inflate the number of bots they have, some methods developed for RatBot can be borrowed to enhance their resilience against enumeration by the adversaries. However, as we shall discuss later, there is a tradeoff among operational flexibility, local detectability, and resilience against enumeration in the design space of P2P botnets.

3 Threat Model

In this work, we consider two families of P2P botnets: *immersive P2P botnets* and *exclusive P2P botnets*. For an immersive P2P botnet, the botmaster delivers

C&C information through a P2P network that has normal P2P nodes in addition to bots. The original Storm botnet, for instance, was an immersive P2P botnet because the C&C information was delivered to the Storm bots through the Overnet network. An exclusive P2P botnet, by contrast, has bots exclusively as its peers and thus does not have any normal P2P user traffic in it. Since the Overnet network was shut down, the Storm botnet became an exclusive P2P botnet dubbed Stormnet because only bots can participate in the botnet.

The two primitive operations in a P2P network are *publish* and *search*. The *publish* primitive is used to publish a data item either on the machine used by the caller itself (*e.g.*, in an unstructured P2P network) or on a machine with an identifier that is close to that of the data object (e.g., in a structured P2P network). The *search* primitive is used by a peer node to search for data items that satisfy some specific conditions, such as containing certain keywords or producing a certain hash digest. In this work, we assume that in the P2P network *search* operations are *spoofable*, that is to say, a peer node can request a peer to find a data item using a spoofed source IP address. This holds for many P2P networks, which use UDP to implement the request/response mechanism in a search operation. For instance, the widely deployed KAD protocol uses UDP for signaling and TCP for data transfers [16].

It will be seen later that spoofable search operations play a key role in the design of RatBot for hiding authentic search operations. It is, however, noted that these constraints limit the design of RatBot only when it is implemented as an immersive P2P botnet. For an exclusive P2P botnet, as bots do not require an existing P2P network for their C&C communications, the botmaster has more freedom on the implementation of spoofable search operations.

In this work, we assume a reasonable adversarial model from the attacker's standpoint. First, we do not assume that the P2P botnet deploys a strong authentication scheme. As evidenced by previous efforts of successfully reverse-engineering the Storm bot executable, it is possible for white-hat security analysts to reveal secret keys used for bot communications through static or dynamic malware analysis, and create fake bots to infiltrate into the P2P botnet [12, 13]. Second, we also assume that the white-hat security analyst, through thorough static code analysis, possesses full knowledge about the functionalities of an authentic bot, including its communication protocol and anti-enumeration techniques. Third, we assume that the behaviors of a fake bot and an authentic bot are indistinguishable to the bots. A fake bot can intercept any message that passes through it, thus obtaining the source IP address it has used. Fourth, a fake bot may stay in the P2P botnet for a long time so that for some P2P protocols (e.g., KAD) a large number of peer nodes would add it to their contact lists, or actively crawl the P2P network to obtain a list of observed P2P nodes.

In the paper, we use the *adversary* and the *white-hat security analyst* interchangeably. Next, we shall present the design of RatBot.

4 RatBot Design

The key idea of RatBot is the existence of an army of *obscure bots*, each of which creates a list of fake identities to hide itself. In this work, we assume that the identity of a bot is manifested as the IP address that it uses to communicate with other peers in the network. Although the P2P identifier (e.g., KAD ID) of a bot can also be used for enumeration purpose, these identifiers sometimes can be changed by bots, thus leading to inaccurate estimate of the botnet size. Moreover, a compromised machine can run multiple instances of bot executable and counting each instance as a bot overestimates the size of a botnet.

As opposed to obscure bots, we say the remaining bots are *explicit bots*. By their nature, explicit bots can be enumerated. In Figure 1, we present the architecture of RatBot in the form of an immersive P2P botnet. If RatBot is an exclusive P2P botnet, no normal peers would exist.

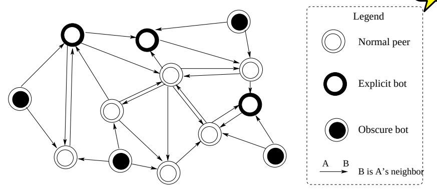


Fig. 1. RatBot Architecture

4.1 Obscure Bot Selection

When a machine is infected and becomes a bot, it decides whether it should be an obscure bot. As an obscure bot uses spoofed IP packets to hide its true identity, an obscure bot must be able to spoof IP packets. Not every end host in the Internet, however, possesses such a capability due to reasons such as NAT deployment and blocking of spoofed packets by firewalls or the host operating systems [5]. We thus let each bot contact a dedicated server during its bootstrapping phase. The server is hardcoded in the bot executable code[3]. When a bot contacts a server, it generates a UDP *query* packet with an arbitrary spoofed source; the payload of the packet carries the authentic IP address of the bot. If the packet arrives at the server, it means that the bot is capable of spoofing. The server decides whether the bot should become an obscure bot and if so, sends back a *response* packet to the bot using its authentic IP address carried in the query packet. If the bot receives the response packet within a certain period of time, it becomes an obscure bot; otherwise, it is an explicit bot.

How does the server decide whether a bot should be an obscure bot? Suppose that it knows the size of the current botnet; this can be done by simply letting each newly infected bot report to it using their authentic IP addresses. The server then makes its decision by aiming to have a fraction ξ of the entire botnet as obscure bots. ξ is *not* hardcoded in the bot executable and it is thus not known to the adversary. Hence, the adversary cannot estimate the botnet size as $m/(1-\xi)$, where m is the number of explicit bots that he has observed.

4.2 Identity Obfuscation

Once a bot decides that it is an obscure bot, it randomly generates a list of spoofing IP addresses that it will use to obfuscate its own IP address later in P2P communications. The spoofing IP addresses should be chosen to be difficult for the adversary to verify their validity, even if the adversary is able to reverse-engineer the bot code. For example, these spoofing IP addresses should avoid

[3] To improve the resilience of the botnet, multiple servers can be specified in the executable code. Also, fast flux techniques can be used to prevent easy disruption.

using those from the dark IP address space, and being too concentrated in a small IP address subspace. The detail of such algorithm is beyond the scope of this work. For a given obscure bot, how many spoofing IP addresses does it create? The answer provides a key role in the level of difficulty for the adversary to infer the correct botnet size. Consider a simple scheme in which each obscure bot generates a constant number k of spoofing IP addresses. As explained later, a distinguishing feature of an obscure bot is that it does not respond to any request by another peer. Suppose that the adversary can enumerate the entire list of IP addresses S that do not respond to any normal P2P requests. Then, the number of obscure bots can be estimated at $|S|/(k+1)$ if it is assumed that spoofing IP addresses do not overlap.

Two observations are worth noting here. First, as obscure bots generate spoofing IP addresses independently, these spoofing IP addresses may overlap in practice. But given that the large IP address space to spoof, such overlapping likelihood should be low. Second, due to the P2P structure of the botnet and independent generation of spoofing IP addresses by individual bots, compromising a small number of bots, although helping the adversary rule out the spoofing addresses used by these bots, does not prevent the overall size of the botnet from still being overestimated.

We now discuss how RatBot chooses the number of spoofing IP addresses per bot. Consider a botnet with n obscure bots. Let X_i denote the number of spoofing IP addresses obscure bot i generates. RatBot uses two levels of obfuscation. For the first level (**distribution-level obfuscation**), RatBot uses a distribution with high variation to generate X_i, such as the Pareto distribution with PDF:

$$f(x) = \begin{cases} \frac{\alpha x_m^\alpha}{x^{\alpha+1}} & \text{for } x \geq x_m, \\ 0 & \text{for } x < x_m, \end{cases}$$

where x_m and α are the *cutoff* and *scale* parameters, respectively. The mean of the Pareto distribution is $\alpha x_m/(\alpha-1)$ and its variance is $(x_m/(\alpha-1))^2 \cdot \alpha/(\alpha-2)$. It is noted that when $\alpha \leq 2$, the variance becomes infinite. If we set $\alpha \leq 2$, then we cannot apply the central limit theorem on $\sum_{i=1}^{n} X_i$ due to the infinite variance. It is noted that X_i drawn from the Pareto distribution is a float number. In practice, we generate $\lfloor X_i \rfloor$ spoofing IP addresses for sure, where $\lfloor x \rfloor$ denotes the largest integer no greater than x, and an extra one with probability $X_i - \lfloor X_i \rfloor$.

In Section 5, we shall present the rationale behind using the Pareto distribution for generating X_i and also its limitation. To make size estimation even more difficult, RatBot employs another level of obfuscation in generating X_i (**parameter-level obfuscation**). Instead of using a fixed mean for X_i, the mean of X_i on the i-th obscure bot actually depends on certain attributes of the bot itself. Measurements from the Storm botnet suggest that bot infection is not uniformly distributed either over different ASes or geographically [6]. Hence, we let the mean number of spoofing IP addresses generated by an obscure bot be a function of the time zone where the bot is located. In previous works, security analysts used the observed IP addresses to derive their geographic locations using IP geolocation tools [1] and thus their corresponding time zones. Now that spoofed IP addresses are used, it is difficult to accurately infer the time zone of each bot, which renders it hard to estimate the mean of each X_i.

An obscure bot may use a dynamic IP address to communicate with other peers. Whenever the obscure bot observes that the IP address of the hosting machine has changed, it regenerates its spoofing IP addresses as above.

4.3 Bot Behavior Description

In a typical P2P protocol, a packet between two peers can be classified into three categories: request, response, and data transfer. TCP makes spoofing difficult because it requires handshaking between peers. In many normal P2P networks, request and response signaling packets are delivered through UDP and data transfer uses TCP. We consider the two cases in the following. (1) If the P2P botnet is an exclusive P2P botnet, UDP can be chosen by design for delivering all request, response and data transfer packets. (2) If the P2P botnet is an immersive one, the botmaster does not have the freedom to choose the transport layer protocol. In this study, we assume that request and response signaling packets use UDP. If bot communications do not involve any data transfer packets, spoofing becomes much easier; however, if the P2P protocol uses TCP for data transfer *and* bots need data transfer for command & control, it leaves a door for more accurate bot size estimation by the adversary, as will be explained in Section 8.

For an explicit bot, its behavior conforms to the standard P2P protocol. For an obscure bot b, let $\mathcal{I}(b)$ denote the set of spoofing IP addresses associated with it. The behaviors of an obscure bot are given as follows.

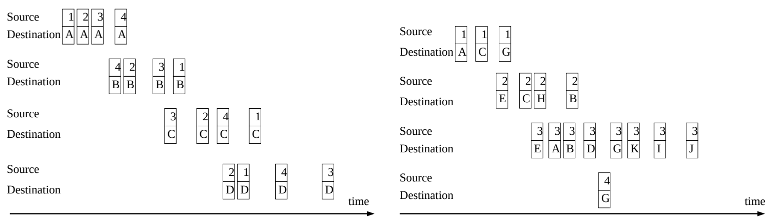
Response packets. An obscure bot does not respond to any request by another peer. On the arrival of a request packet, it silently drops the packet. As the packet is delivered through UDP, which is connectionless, the origin of the request packet does not know whether the recipient receives the packet or not.

Request packets. We first consider a naive *packet-level* obfuscation scheme for request packets. When an obscure bot b needs to send out a request packet to peer A at time t, it replicates the packet for $|\mathcal{I}(b)|$ times and each of these packets uses a distinct source IP address from set $\mathcal{I}(b)$. Including the original request packet, there are in total $|\mathcal{I}(b)|+1$ packets to be sent to peer A. For each obscure bot, we define its obfuscation window as w time units. We *randomly* reorder the $|\mathcal{I}(b)|+1$ packets as p_0, p_1, ..., and $p_{|\mathcal{I}(b)|}$. Packet p_0 is sent out at time t. The interval between the sending times of packet p_i and p_{i+1} where $i = 0, 1, ..., |\mathcal{I}(b)|$ is drawn from an exponential distribution with mean $w/|\mathcal{I}(b)|$.

As the order of the packets is random, the recipient peer, if a monitoring node by the adversary, cannot determine which packet carries the authentic source IP address. However, every time a request packet with an authentic source IP is sent, packets with all associated spoofing IP addresses are also sent to the recipient. Hence, if the recipient is a monitoring node deployed by the adversary, she can cluster IP addresses with the same (or approximately the same) number of appearances within w time units. It is highly unlikely that source IP addresses in normal request packets would show such strong correlation as in the naive obfuscation scheme. As such, even though the adversary does not know exactly which source IP address is authentic, he can still infer the actual size of the botnet by assuming that IP addresses frequently appearing in the same interval of w time units would come from the same obscure bot.

It is noted that request packets are usually used by a bot to search for C&C messages from the botmaster. Hence, to prevent correlation-based analysis, RatBot uses a *session-level* obfuscation scheme for each search operation. Figure 2 illustrates the difference between packet-level and session-level obfuscation. Suppose that an obscure bot needs to find a data item with key $\mathcal{K}$. We call it an *authentic session*, which contains the whole sequence of the peer nodes this bot has contacted in order to accomplish this search operation.

For each of its spoofing IP addresses, the obscure bot will create a *spoofing session*, which contains a sequence of peer nodes that are randomly drawn from a local peer node repository. This repository, denoted $\mathcal{R}$, contains peers that were observed in the past authentic sessions and also the current neighbors that the obscure bot knows. It is noted that peers in an authentic session may appear with a certain order. For instance, when a bot searches a data item with key $\mathcal{K}$ in a DHT P2P network, peers in the authentic session are ordered (or partially ordered) in their distances from key ID $\mathcal{K}$. Hence, when constructing the sequence of peers in a spoofing session, such orders are also mimicked.



(1) Packet-level obfuscation (2) Session-level obfuscation

Fig. 2. Obfuscation comparison (*In packet-level obfuscation, each authentic packet is mixed with a number of packets with spoofed sources but the same destination; in session-level obfuscation, each authentic session is mixed with a number of sessions with spoofed sources and previously observed peers as destinations.*)

The intervals between the starting times of sessions, including both authentic and spoofing ones, are randomly drawn from an exponential distribution with mean γ time units. The order of the starting times of spoofing sessions is randomized. The authentic session is inserted among the top ϕ spoofing sessions, if there are so many, and its place is also randomly chosen. The decision on ϕ should make it difficult to tell which session is authentic but meanwhile ensure that the start of the authentic session would not be postponed significantly due to obfuscation. In our implementation, we let ϕ be 5.

Let Ψ denote the empirical distribution of the number of request packets sent in an authentic session. For each spoofing session, we use Ψ to generate the number of request packets. Each of these request packet carries the spoofing IP address as its source IP and search key $\mathcal{K}$, and is sent to every peer node in the corresponding spoofing session. The interval between two request packets is randomly drawn from the empirical distribution of the intervals between request packets in the past authentic sessions. We use Γ to denote this distribution.

Data transfer packets. If botnet C&C information is stored as a file, each bot needs to fetch the file from the host machine. If RatBot is designed to be an exclusive P2P botnet, UDP can be chosen for data transfer. Otherwise, if it is an immersive P2P botnet, RatBot makes its decisions in the following order: (1) If the C&C information can be spread without involving data transfer, RatBot will not use data transfer. For instance, C&C information can be stored as metadata tags in a KAD-based P2P network. (2) If the P2P network allows UDP for data transfer, RatBot will use UDP instead of TCP for data transfer. (3) Only if the P2P network uses only TCP for data transfer, RatBot would use TCP. It is noted that the third option exposes the identity of obscure bots if the peer hosting the C&C information is actually a monitoring node deployed by the adversary. This is because TCP requires a three-way handshake between the obscure bot and thus the host machine and the connection cannot be spoofed.

5 Rationale

In this section, we explain why a high variance distribution such as the Pareto distribution is used to generate X_i in Section 4.2. As we assume an adversarial model in which the adversary knows the distribution used to generated X_i, we must ensure that the adversary's knowledge does not lead to a good estimation of the botnet size. The adversary also knows that an observed IP address cannot be from an explicit bot if it is used in response packets. Let M be the number of IP addresses observed by the adversary that never respond to any requests. The challenge is: can the adversary infer the number of obscure bots provided that he knows the distribution used to generate X_i?

If only the distribution-level obfuscation is used, all X_i are independent and identically-distributed random variables. According to the *law of large numbers*, $\sum_{i=1}^{n} X_i$ always approaches $n\mu$, where μ is the mean of X_i, **when n is large**. As the adversary knows the distribution and thus μ, he can estimate the botnet size as $M/(\mu + 1)$. To defeat this type of inference, it is necessary to use a distribution that converges so slowly that $\sum_{i=1}^{n} X_i$ can still be far away from $n\mu$ at reasonable scales of botnet sizes.

The *Chebyshev's inequality* tells us that $\mathbb{P}\{|Y - \bar{Y}| \geq t\} \leq t^{-2}Var(Y)$, where $\bar{Y}$ and $Var(Y)$ are the mean and variance of random variable Y, respectively. Hence, the convergence speed of $\sum_{i=1}^{n} X_i$ is affected by the variation of X_i. That explains our choice of the Pareto distribution: for $\alpha < 2$, its variation is infinite and thus slows down the convergence of $\sum_{i=1}^{n} X_i$.

Suppose that there are $100,000$ obscure bots and the average number of spoofing IP addresses an obscure bot generates is 20. We consider four different settings for the scale parameter: $\alpha = 1.01$, 1.1, 1.5, and 1.8. We set the cutoff parameter accordingly to obtain the same mean for X_i. We simulate 1000 cases with different random number generation seeds. In each case, we assume that the adversary sees all the obscure and spoofed IP addresses. Let the observed total number be M. The adversary estimates the number of actual obscure IP addresses as $M/21$ as each obscure IP address has 20 spoofed ones. The following table shows the mean and the standard deviation of the adversary's estimation:

α	1.01	1.1	1.5	1.8
mean	23596.80	81758.83	99854.08	99962.19
standard deviation	83014.82	91258.15	4553.54	1262.98

From the table, it is clear that when α is close to 1, the variability of the estimated bot size becomes more significant. For instance, when $\alpha = 1.01$, even after 1000 sample runs, the derived mean is still far away from the actual one, which is 100000. In reality, the adversary witnesses the result of only one sample; hence, if α is small and thus the variability is very high, the adversary will get an estimate on the botnet size with high variation.

Using heavy tailed distributions such as the Pareto distribution to generate X_i does have its limitation, even though they can produce highly variable results. The high variation of these distributions actually results from their high skewness in their probability density functions. Figure 3 depicts the probability density function of the Pareto distribution when $\alpha = 1.01$ and the mean is 20. Clearly, it is highly skewed as $\mathbb{P}(X_i \leq 1) = 0.805$, which means that around 80% of the data points, if drawn from this distribution, would stay below 1.

To see how this would help the adversary's estimation, we simulate the observed number of spoofing IPs when there are 1000, 10000, 100000, and 1000000 obscure bots. Each obscure bot uses the Pareto distribution with mean 20 and scale parameter 1.01 to generate the number of spoofing IP addresses. For each scenario, we simulate 1000 times. The results are shown in Figure 4, where each data point represents the number of observed spoofing IPs. Note that for each scenario, the number of observed spoofing IPs is highly clustered among the 1000 sample runs. Suppose that the adversary has observed 3000 spoofing IP addresses. Then, he can infer that the real size of the botnet is likely to lie between 10000 and 100000. Hence, RatBot uses another level of obfuscation (i.e., parameter-level obfuscation) to defeat such kind of statistical inferences.

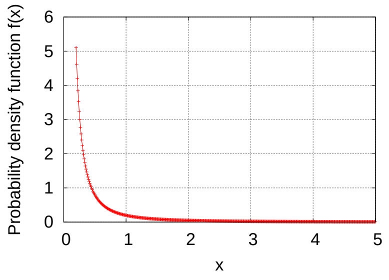


Fig. 3. PDF of Pareto distribution

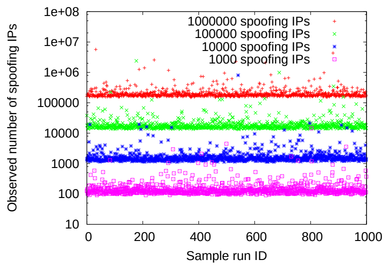


Fig. 4. Observed spoofing IPs

6 Kad-Based RatBot Implementation

In this section, we discuss how to implement RatBot based on KAD, which extends from the Kademlia protocol proposed by Maymounkov and Mazieres [15]. Our implementation of RatBot is based on a popular KAD client, *aMule*[4]. UDP is used in aMule for searching and publishing data objects. If it is an explicit bot,

[4] The version we used in our study is aMule 2.1.3.

we keep the original implementation intact. Otherwise if it is an obscure bot, we make the following modifications. First, when the bot receives a request message, it drops the message immediately. A request message in KAD carries some special operation codes, such as `KADEMLIA_HELLO_REQ`, `KADEMLIA_SEARCH_REQ`, `KADEMLIA_REQ`, `KADEMLIA_PUBLISH_REQ`, etc.

Second, in the KAD protocol peers regularly send `KADEMLIA_HELLO_REQ` messages to each other to exchange liveness information. It is noted that the adversary can use such messages to determine whether a peer is an obscure bot or just a spoofed IP address. There are two solutions to this. One option is that the obscure bot obfuscates these messages as well, using spoofing IP addresses. The flip side of this approach is that peers may inject those spoofed IP addresses into their routing tables, thus affecting normal routing operations. The other solution is that an obscure bot does not send out such messages at all. Even though obscure bots and their spoofed IP addresses may still be inserted into their neighbors' routing tables when their neighbors receive search requests from them, the lack of liveness messages makes them less likely to be chosen in a search process because KAD prefers long-lived nodes when forwarding search requests. Also, when a peer node finds that a neighbor has not been alive for a certain period of time, it removes that neighbor from its routing table. Given these considerations, we adopt the second approach in our implementation.

Third, as obscure bots do not send out `KADEMLIA_HELLO_REQ` messages to their peers, their peers do not send back response messages with type `KADEMLIA_HELLO_RES`. According to the standard KAD protocol, obscure bots' routing tables would shrink faster because neighbors without liveness messages are removed from the routing table after a certain period of time. To avoid this, we increase the longevity of each neighbor without liveness messages in an obscure bot's routing table from the original two minutes to two hours.

Fourth, a KAD node initiates some random searches when it observes that a bucket does not have enough contacts in its routing table. For an obscure bot, it has to use its authentic IP address for such random lookups. It is necessary to obfuscate these searches also, because otherwise the adversary can infer whether an observed IP address is authentic or not by how many unique keys it uses for searching. In our implementation, we obfuscate these random searches as well.

Finally, we let RatBot use the metadata tags in KAD, such as filenames, to hide C&C information. Hence, no data transfer is needed for normal bot operations. Also, obscure bots never publish any information into the P2P network; they only passively search commands given from the botmaster. The botmaster uses only explicit bots to publish his C&C information.

7 Experimental Evaluation

We now evaluate the effectiveness of RatBot in preventing the adversary from obtaining an accurate estimate on the botnet size. Due to the destructive nature of RatBot, we do this in a simulated environment to avoid legal and ethical issues. Our KAD-based implementation of RatBot used the actual implementation code of aMule. We further intercepted all system calls in it, such as time-related and socket functions and replaced them with simulated function calls specific to our local distributed simulation platform. According to the literature, behaviors of both normal P2P users and bots exhibit strong time zone effects [21, 8]. To incorporate these details into our simulation, we model the geographic distribution of normal KAD peers based on previous measurements on the KAD network [21] and that of bots according to the Storm botnet IP distribution [6].

Our model of normal P2P user behaviors is based on the observations on the online patterns of normal KAD users [22]. The starting time of a normal peer being online is modeled with a Gaussian distribution with mean at 7:00pm and standard deviation at 2 hours, and the duration of an online session is generated with a three-parameter Weibull distribution. The online activity model of a bot machine is simply defined as follows: the starting time of it being online is drawn from a Gaussian distribution with mean at 8:00am and the end time is drawn from a Gaussian distribution with mean at 6:00pm; for both distributions, the standard deviation is one hour. This model reflects people's normal work hours.

The number of spoofing IP addresses corresponding to an obscure bot is generated from a Pareto distribution whose parameters are set as follows. Let us number the 24 time zones from 1 to 24. The mean of the Pareto distribution is drawn from a Gaussian distribution with mean and standard deviation set as $2z$ and $4z$, respectively, where z is the time zone number of the obscure bot. The scale parameter of the Pareto distribution is 1.05 and its cutoff parameter can be calculated accordingly from its mean. In each experiment of this study, we use 100 processors from a cluster machine to simulate the behaviors of RatBot.

7.1 Exclusive RatBot

In the first set of experiments, we study the behavior dynamics of exclusive RatBots. We let the botmaster send out a command every day. To improve the reachability of the command to individual bots, the botmaster uses five bots to publish it with 32 keys (as in the Storm botnet) periodically every 100 seconds. Each individual bot, when online, periodically searches the command every 100 seconds with these 32 keys until it gets the command successfully. We simulate 10,000 bots and vary the number of obscure bots among $\{1000 \times i\}_{i=0,1,2,3,4,5}$. Among the 10,000 bots, 10% of them are P2P servers that always stay online. We assume an adversarial model in which the adversary controls 10 servers that can be used to monitor bot traffic. We simulate the botnet for two days: the first day is used as a ramp-up phase for each obscure bot to obtain some empirical distributions, and the second day is used for testing. For each scenario, we simulate it for 20 times with different random number seeds.

We first verify our implementation to ensure that behaviors of spoofing sessions are close to those of authentic sessions. In Figure 5, we depict the frequency histogram of the number of appearances of packets from spoofing and authentic sessions observed by the monitors, respectively, in five runs when there are 1000 obscure bots. There is no obvious systematic difference between authentic and spoofing sessions that can be exploited to differentiate them. From the simulation results, we also note that regardless of the number of obscure bots in the RatBot, almost every individual bot gets the command eventually. Hence, the existence of obscure bots does not affect the utility of the P2P botnet.

Figure 6 gives the median, smallest, and largest number of IP addresses observed by the adversary in 20 sample runs eventually and after one day, respectively, under different number of obscure bots. In the eventual results, we show the total number of spoofing IP addresses generated by obscure bots plus the number of actual bots. We notice that after one day, the adversary observes a large fraction of both actual and spoofing IP addresses. This is because we assume the adversary is able to deploy monitors among the core servers of the P2P botnet and the bots search the command frequently.

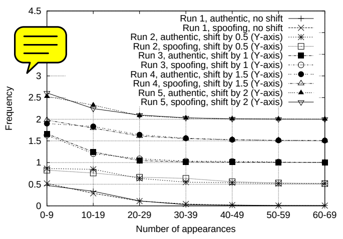
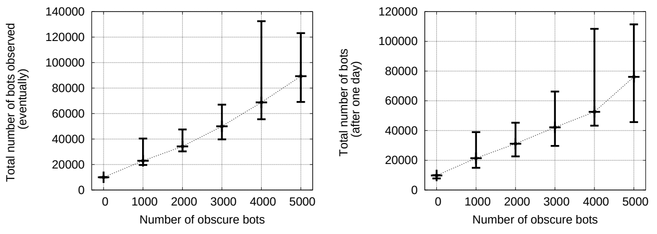


Fig. 5. Frequency histogram of the number of appearances in five runs



(1) Eventual results (2) One-day results

Fig. 6. Total number of bots observed by the monitors, including explicit, obscure, and spoofing bots

Unsurprisingly, if we increase the number of obscure bots, the number of observed IP addresses by the adversary also increases. When there are 4000 or 5000 obscure bots, there are cases where the total number of IP addresses observed by the adversary exceeds 100,000, suggesting that the obfuscation technique of RatBot can lead to an overestimation more than 10 times of its actual size. On the other hand, given the same number of obscure bots, the observed number of IP addresses also varies significantly among different runs. In some scenarios, the largest number of IP addresses observed is twice as much as the smallest number of IP address observed in the 20 sample runs. It is also noted that the median tends to be close to the minimum due to the fact that the Pareto distribution is skewed towards its cutoff parameter at its lower end.

7.2 Immersive RatBot

We now evaluate how immersive RatBot affects the accuracy of botnet size estimation. We simulate a P2P network with 7,000 normal peers and 3,000 bots. The botmaster uses five bots to publish commands with 32 keys periodically every half hour. Each bot uses these 32 keys to search for the current command every half hour until it obtains the command successfully. Here, we let bots perform publish and search operations less frequently than those in exclusive RatBot because normal P2P peers may treat these bots performing frequent operations as abnormal and thus limit interations with them. Among 7,000 normal peers, 990 of them always stay online as servers. We assume the adversary deploys 10 monitors in the network and they appear as servers always online. Each monitor is also a captured bot and can be used to reveal the 32 keys used by the bots to search the current command. The monitor identifies a peer as a bot if it observes that the peer uses any of these keys to search or publish a data item in the P2P network. We vary the number of obscure bots among 0, 1000, 2000, and 3000. For each scenario, we simulate it for four days, the first of which is used as a ramp-up phase for each obscure bot to obtain some empirical distributions and the remaining days are used for testing. We simulate each scenario 20 times.

Figure 7 depicts the number of bots observed by the adversary under different numbers of obscure bots. For visual clarity, we shift the points horizontally slightly to prevent overlapping. For each scenario, we show the median, minimum, and maximum among the 20 sample runs. The results corresponding to "Eventually" show the sum of both the number of authentic bots (including obscure bots) and the total number of spoofing IP addresses generated by all obscure bots.

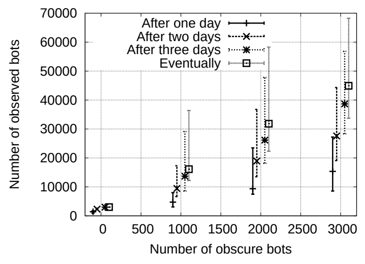


Fig. 7. Number of bots observed by monitors under different numbers of obscure bots (0, 1000, 2000, 3000)

According to the results, we make the following observations. First, the existence of obscure bots produces estimated botnet sizes with high variation. For instance, after three days, if there are no obscure bots, the ratio of the maximum and the minimum of observed bots is 1.016; when we introduce 1000, 2000, and 3000 obscure bots, the ratio becomes 3.405, 2.637, and 2.006, respectively. Such high variation renders it difficult for the adversary to infer the true size of the botnet. Second, it is obvious that increasing the number of obscure bots helps inflate the number of observed bots by the adversary. When there are 1000 obscure bots, the ratio of the median number of observed bots after three days to the true size of the botnet is only 4.5, but when there are 3000 obscure bots, this number becomes 12.8. Hence, the botmater can use the fraction of obscure bots to control the error in the adversary's estimation.

8 Countermeasures

Given the disruptive nature of RatBot, it is important for us to understand its weakness and potential methods to mitigate it. In this section, we present a few countermeasures that can defeat the obfuscation techniques deployed by RatBot. First, RatBot requires each bot to contact a central server initially to decide whether it should work as an obscure bot. The server can easily become a single point of failure. If the adversary manages to monitor traffic from and/or to this server, the identities of true bots can be revealed. With regard to this, it is noted that each bot only needs to contact this server during the bootstrapping phase. As there is little communication for this purpose, it is a difficult task to monitor such traffic. Moreover, existing botnets commonly apply distributed server farms and fast-flux techniques to improve resilience of their services. These techniques can also be applied here to prevent the single failure of the server.

In order for RatBot to operate, the search operation must be spoofable. Hence, if a P2P network deploys anti-spoofing techniques, RatBot cannot survive in it. For example, the P2P network can simply use TCP for all signaling and data transfers. Even if UDP is used for signaling, the P2P network can add a level of anti-spoofing mechanism in a query: when Peer A receives a query from Peer B, it sends back a confirmation request to Peer B and only answers Peer B's query after receiving a reply from Peer B on its request. It is noted that this countermeasure works only against immersive RatBot because the botnet has to be blended into an existing P2P network. Albeit effective in defeating the anti-enumeration scheme by RatBot, fully deploying anti-spoofing techniques in all enterprise networks and ISPs still has a long way to go [5]. For instance, the recent analysis of 5,000 DDoS attacks suggests that a significant fraction of them still used spoofing techniques to generate large volumes of attack traffic [3].

If the RatBot needs TCP data transfer to fetch the command, the adversary can deploy monitors in the P2P network and place those command data on them. By monitoring which machines fetch the command data, the adversary can obtain a list of authentic bots as the three-way handshaking mechanism in TCP cannot be spoofed with spurious IP addresses.

Another effective approach to defeat RatBot is deploying anti-spoofing techniques in the whole Internet. The degree to which the RatBot can obfuscate its size depends on how many obscure bots it has to perform spoofing operations. If the majority of Internet addresses cannot be spoofed, we can still obtain a good estimate on the size of RatBot by simply ignoring those obscure bots.

RatBot's relying on spoofing packets for obfuscation introduces another weakness: enterprise networks and ISPs can detect the existence of bots in their networks by looking for hosts that send out spoofed packets. RatBot prevents enumeration by the adversary at the global level at the price of increased vulnerability to detection at the local level due to its use of spoofing packets.

9 Conclusions

In recent years, botnets, which emerge as a major cyber threat, have been widely used to send spamming emails and launch DDoS attacks. In a botnet war, a botnet owner may want to bluff his botnet size in order to intimidate the adversary, gain media attention, or win a contract. In this work, we explore the tactics that a botnet may use to achieve this goal. We present the design of a type of P2P botnets called RatBot, which applies obfuscation techniques to defeat standard enumeration techniques, and use large-scale high-fidelity simulation to evaluate its performance. We hope our work will raise the awareness of white-hat cyber-security practitioners on the challenges of estimating the sizes of botnets accurately and adopt effective countermeasures in practice.

References

1. `http://www.ip2location.com/`.
2. `http://www.amule.org`.
3. `http://asert.arbornetworks.com/2010/12/the-internet-goes-to-war/`.
4. P. Barford and V. Yegneswaran. *Malware Detection*, volume 27 of *Advances in Information Security*, chapter An Inside Look at Botnets. Springer US, 2007.
5. R. Beverly, A. Berger, Y. Hyun, and k claffy. Understanding the efficacy of deployed Internet source address validation filtering. In *Proceedings of ACM IMC'09*, 2009.
6. http://isisblogs.poly.edu/2008/05/19/storm-worm-ip-list-and-country-distribution-statistics.

7. `http://www.net-security.org/secworld.php?id=8858`.
8. D. Dagon, C. C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Proceedings of NDSS'06*.
9. J. Goebel and T. Holz. Rishi: identify bot contaminated hosts by IRC nickname evaluation. In *Proceedings of HotBots'07*, 2007.
10. 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 USENIX Security'08*, 2008.
11. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security'07*.
12. T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08*.
13. B. B. Kang, E. Chan-Tin, C. P. Lee, J. Tyra, H. J. Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, and Y. Kim. Towards complete node enumeration in a peer-to-peer botnet. In *Proceedings of ACM ASIACCS'09*, 2009.
14. A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of HotBots'07*, 2007.
15. P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of IPTPS'01*.
16. M. Pietrzyk, G. Urvoy-Keller, and J.-L. Costeux. Digging into kad users' shared folders. In *Posters of ACM SIGCOMM'08*, 2008.
17. P. Porras, H. Saidi, and V. Yegneswaran. Conficker C P2P protocol and implementation. `http://mtc.sri.com/Conficker/P2P/`, September 2009.
18. 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 *HotBots'07*.
19. A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using dnsbl counter-intelligence. In *Proceedings of SRUTI'06*, 2006.
20. G. Starnberger, C. Kruegel, and E. Kirda. Overbot: a botnet protocol based on kademlia. In *Proceedings of SecureComm'08*, 2008.
21. M. Steiner, T. En-Najjary, and E. W. Biersack. A global view of kad. In *IMC'07*.
22. M. Steiner, T. En-Najjary, and E. W. Biersack. Analyzing peer behavior in kad. Technical Report EURECOM+2358, Institut Eurecom, France, October 2007.
23. B. Stock, J. Gobel, M. Engelberth, F. C. Freiling, and T. Holz. Walowdac - analysis of a peer-to-peer botnet. In *Proceedings of the 2009 European Conference on Computer Network Defense*, 2009.
24. B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the ACM CCS'09*, 2009.
25. `http://www.neoseeker.com/news/7103-worm-storm-gathers-strength/`.
26. S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich. Analysis of the storm and nugache trojans: P2p is here. *;Login:*, 32(6), December 2007.
27. R. Vogt, J. Aycock, and M. J. Jacobson. Army of botnets. In *NDSS'07*, 2007.
28. P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. In *Proceedings of HotBots'07*.
29. Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are ip addresses? In *Proceedings of ACM SIGCOMM'07*, 2007.
30. G. Yan, D. T. Ha, and S. Eidenbenz. Antbot: Anti-pollution peer-to-peer botnets. *Computer Networks*, 55(8), June 2011.
31. T.-F. Yen and M. K. Reiter. Traffic aggregation for malware detection. In *Proceedings of DIMVA'08*, 2008.