Study of the Honeypot-Aware Peer-to-Peer Botnet and Its Feasibility

Ping Wang

Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362 USA

October 5, 2010

1 Project Overview

The research objective of this project is to investigate one possible advanced botnet – honeypot-aware peer-to-peer (P2P) botnet: verifying our analysis of the propagation of a honeypot-aware P2P botnet, and then showing the feasibility of developing a such botnet in terms of its propagation effectiveness.

A “botnet” is a network composed of compromised computers (“bots”) on the Internet, that are under control of a remote attacker (“botmaster”). Botnets have become one of the most significant threats to today’s Internet. Most of the current botnets are centralized botnets, such as IRC-based botnets. All the bots are connecting to a central authority to get botmaster’s commands. However, in recent years, P2P botnets have emerged. They leverage existing P2P protocols and networks for either bootstrapping onto a hierarchical command and control (C&C) network or command and control directly. With the good feature of P2P networks, that is the resilience to “churn” (the network dynamics caused by node leaving and joining), P2P botnets are more robust than centralized botnets, because the losing a couple of bots will not bring much impact on a P2P botnets, while the C&C communication will be disrupted if a central server in a centralized botnet is captured and shutdown by defenders. Therefore P2P botnets have made the situation even worse.

On the other hand, honeypots and honeynets are effective detection and defense techniques, and hence there has been much recent research in this area [20, 12, 15, 7, 24]. And in the meantime, attacker have started to develop anti-honeypot approaches, such as [17, 9]. It is highly possible that attackers would integrate honeypot detection techniques with P2P botnets, and come up with honeypot-aware P2P botnets, in order to prevent their botnets from being infiltrated and monitored.

So to better prepare for the future, we propose a general hardware- and software-independent honeypot detection, and its implementation in P2P botnets. To show the feasibility of developing a such botnet, we model and analyze the botnet propagation time delay caused by honeypot detection, and compare it with a P2P botnet without the capability of honeypot detection. Our conclusion drew from the numerical results is positive. But to be more convincing and further our research, we would like to perform simulation experiments on real machines in a distributed environment, which is exactly what PlanetLab [3] provides.
In the next section, we will detailed describe our proposed project. Once again, we’ll briefly talk about our motivation in Section 2.1. Our honeypot detection methodology and its implementation in P2P botnets are presented in Section 2.2 and Section 2.3 respectively. Finally Section 2.4 is our experiment propose of how we would like to carry out simulation experiment on PlanetLab.

2 Project Description

2.1 Motivation

A honeypot is a special constructed computer or network trap designed to attract and detect malicious attacks [14]. In recent years, honeypot techniques have become popular, and security researchers have generated many successful honeypot-based attack analysis and detection systems. Especially, more people start leveraging honeypot techniques to monitor and mitigate botnets. Because of this, attackers constructing and maintaining botnets will sooner or later try to find ways to avoid honeypot traps. Therefore, we should study honeypot avoidance techniques that could be deployed by botmasters in their next generation botnets in order for us to design better and long-lasting honeypot-based defense systems.

Therefore, we study how botmasters might attempt to remove honeypot traps when constructing and maintaining their botnets. This knowledge is useful for security professionals to be better prepared for more advanced botnet attacks in the near future. Unlike hardware or software specific honeypot detection methods [21, 11, 5], the honeypot detection methodology presented here is based on a general principle that is hardware and software independent: security defenders who set up honeypots have liability constraint, either legally or ethnically, such that they cannot allow their honeypots to participate in real attacks that could cause damage to others, while botmasters do not need to follow this constraint. Based on this principle, botmasters could detect honeypots in their botnets by checking whether compromised machines in a botnet can successfully send out unmodified malicious traffic.

2.2 Honeypot Detection Methodology

First, we introduce a method to detect honeypots that are infected and acting as bots in a botnet. The general principle is to have an infected computer send out certain malicious or “counterfeit” malicious traffic to one or several remote computers that are actually controlled by the botmaster. These remote computers behave as “sensors” for the botmaster. If the sensors receive the “complete” and “correct” traffic from the infected host, then the host is considered “trusted” and is treated as a normal bot instead of a honeypot. Since honeypot administrators do not know which remote computers contacted are the botmaster’s sensors and which ones might be innocent computers, they cannot defend against this honeypot detection technique without incurring the risk of attacking innocent computers.

This honeypot detection procedure is illustrated in Fig. 1. A newly infected computer cannot join a botnet before it is verified. This potential bot machine must first send out malicious traffic to many targets, including the botmaster’s secret sensor (unknown to the newly infected machine). When the botmaster’s sensor receives the traffic and verifies the correctness of the traffic (ensuring that it was not modified by a honeypot), the sensor informs the bot controller (e.g. the botmaster or a C&C server) of the bot’s IP address. The bot controller then authorizes the checked bot so
that the bot can join the botnet. To prevent the possibility of a single point of failure, a botmaster could set up multiple sensors for this test.

This honeypot detection procedure can be performed on a newly infected computer before it is allowed to join a botnet. Such a botnet has a built-in authorization mechanism. The botmaster (or the C&C server) uploads the authorization key to the host and allows it to join the botnet only after the host passes the honeypot detection test. In addition, botmasters may perform the honeypot detection periodically on botnets to discover additional honeypot bots. This could be done whenever botmasters renew their bots’ authorization keys or encryption keys, or update the botnet software.

This honeypot detection scheme relies on the report of sensors deployed by botmasters. Therefore, botmasters must first ensure that sensor machines themselves are not honeypots. This is not hard to be done since only a few sensor machines are needed—botmasters can manually investigate these machines thoroughly beforehand.

2.3 Honeypot Detection in Peer-to-Peer Botnets

Most current botnets in the Internet use the centralized structure, such as IRC-based botnets. To increase the availability of the C&C channels in a centralized botnet, a botmaster has to increase the number of C&C servers in the botnet. This will increase the financial cost of maintaining the botnet, since the botmaster will need to purchase more Dynamic DNS domain names. In addition, the botnet is susceptible to C&C server hijacking, which exposes the identity of the entire botnet to security professionals, as was illustrated in [13].

To botmasters, changing a botnet’s control architecture to be peer-to-peer is a natural way to make a botnet harder to be shut down by defenders. In recent years, botmasters have tested and implemented different kinds of preliminary P2P botnets such as Slapper [6], Sinit [4], Phatbot [2] and Nugache [18]. Some researchers have studied P2P botnet designs [23, 25]. Therefore, we believe more P2P botnets will be created in the near future.

Botmasters will need to come up with a new honeypot detection technique for a P2P botnet. In a P2P botnet, each bot contains a list of IP addresses of other bots that it can connect with, which is called “peer list” [25]. Because there are no central authorities to provide authentication in a P2P botnet, each bot must make its own decision, or collaborate with its peers, to decide whether
its hosted machine is a honeypot or not. Here we present a simple but effective advanced worm, called “two-stage reconnaissance worm”, that can be used to distributively detect honeypots as it propagates.

### 2.3.1 Two-Stage Reconnaissance Worm

A two-stage reconnaissance worm is designed to have two parts: the first part compromises a vulnerable computer and then decides whether this newly infected machine is a honeypot or not; the second part contains the major payload and also the authorization component allowing the infected host to join the constructed P2P botnet. Due to the different roles in a worm propagation, we call the first part the “spearhead”, the second part the “main-force” of the worm.

![Illustration of the propagation procedure of a two-stage reconnaissance worm](image)

Figure 2: Illustration of the propagation procedure of a two-stage reconnaissance worm

A simple way to verify whether a newly compromised host is a honeypot or not is to check whether or not the worm on it can infect other hosts in the Internet. Fig. 2 illustrates the propagation procedure of a two-stage reconnaissance worm in infecting host B and checking whether it is a honeypot or not. First, the vulnerable host B is infected by the spearhead of the worm, which contains the exploiting code and the peer list. Second, the spearhead on host B keeps scanning the Internet to find targets (such as host C) to infect with the spearhead code. Third, after the spearhead on host B successfully compromises m hosts (include both vulnerable and already-infected ones), it tries to download the main-force of the worm from any host in its peer list that has the main-force component. The main-force code lets the worm join the constructed botnet via the authorization key contained in the main-force (e.g., the authorization key could be a private public key).

By deploying such a two-stage reconnaissance worm, the botnet is constructed with a certain time delay as the worm spreads. This means that some infected hosts will not be able to join the botnet, since they could be cleaned before the main-force is downloaded. However, this does not affect the botnet, since it makes no difference to the botmaster whether or not the botnet contains bots that will be quickly removed by security defenders.

In fact, it is not a new idea to spread a worm in two stages. Blaster worm and Sasser worm used a basic FTP service to transfer the main code of the worm after compromising a remote vulnerable host [10]. The two-stage reconnaissance worm presented here can be treated as an advanced two-stage worm by adding the honeypot detection functionality into the first-stage exploit code.

The reconnaissance worm described above needs a separate procedure (Procedure 3 as shown in Fig. 2) to obtain the complete bot code. This could be a problem for a botnet since the original Host A might be unaccessible from others, or Host A has changed its IP address when Host B tries to get the main-force worm code. To deal with this issue, the worm could combine the main-force code together with the spearhead code, but first deactivate and possibly encrypt the main-force code at the beginning. After the spearhead code verifies that a hosted machine is not honeypot,
it will unpack and execute the main-force code. One drawback of this approach is that honeypot defenders can easily obtain the main-force code even when their honeypots are not able to join the botnet.

2.3.2 Modeling and Analysis of Honeypot Detection Time Delay

As described above, an infected host joins in a botnet only after it has executed the main-force code of the reconnaissance worm. Thus the botnet grows a step behind the propagation of the worm’s spearhead. This time delay affects when the botmaster can use the botnet to conduct attacks, or when he/she can upgrade the botnet code. Thus botmasters may be interested in knowing this time delay. In addition, the time delay also affects the attack strength by a newborn botnet (some compromised machines have not joined in the botnet yet due to the honeypot detection procedure), thus security defenders may also be interested in knowing the delay time. In this section, we study the time delay caused by honeypot detection procedure. We present an analytical model for modeling the growth of a botnet as the two-stage reconnaissance worm spreads.

The modeling presented here tries to show that a two-stage worm will not slow down botnet construction, even though it adds a delay. The modeling results, as presented below, show that all infected computers (not including detected honeypots) will join the botnet in the end, and the machines will join the botnet shortly after the initial infection.

The spearhead of a two-stage reconnaissance worm propagates in way similar to that of a traditional worm, thus it can be modeled by the popular epidemic model as used in [19, 22, 26], etc. Now we present a simple model, where the two-stage reconnaissance worm uniformly scans the IP space. Papers such as [16, 27] have presented modeling of local preference scanning, bandwidth-limited spread, and other worm scanning strategies. The model presented here can be extended based on the models in those papers for various non-uniform scanning strategies.

Let $I(t)$ denote the total number of infected hosts at time $t$ — whether a host is infected only by the spearhead or by the full worm; $\bar{I}(t)$ denotes the number of infected hosts that have joined in the botnet by time $t$, i.e., they have the main-force of the worm. The propagation of the spearhead can be modeled as [22, 26, 27]:

$$\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t)[N - I(t)]$$ (1)

where $N$ is the total vulnerable population, $\eta$ is the worm’s average scan rate per infected host, $\Omega$ is the size of the IP space scanned by the worm.

First, we derive the propagation model of $\bar{I}(t)$ via “infinitesimal analysis” for the two-stage reconnaissance worm with $m = 1$, i.e., a spearhead-infected host downloads the main-force right after it sends out the spearhead and compromises another host. At time $t$, there are $I(t)$ infected hosts, among them $[I(t) - \bar{I}(t)]$ are infected only by the spearhead — they have not infected others yet. At the next small time interval $\delta$, each spearhead-only infected host will have the probability $p = \eta \delta N / \Omega$ to infect another host since there are $N$ targets to infect (a target host that has already been infected still counts). Therefore, on average $[I(t) - \bar{I}(t)]p$ spearhead-only infected hosts will infect others and download the main-force of the worm during the small time interval $\delta$. Thus we have,

$$\bar{I}(t + \delta) - \bar{I}(t) = [I(t) - \bar{I}(t)] \cdot p = \frac{\eta}{\Omega} [I(t) - \bar{I}(t)] N \cdot \delta$$ (2)
Taking $\delta \to 0$ yields the botnet growth model ($m = 1$):

$$
\frac{dI(t)}{dt} = \frac{\eta}{\Omega} [I(t) - \bar{I}(t)] N \tag{3}
$$

$$
\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t)[N - I(t)]
$$

For a general two-stage reconnaissance worm that has $m > 1$, we can derive the botnet growth model in the similar way. For example, if $m = 2$, then we need to add an intermediate variable $I_1(t)$ to represent the number of spearhead-only infected hosts at time $t$ — each of them has infected exactly one host at time $t$. Using the similar infinitesimal analysis as illustrated above, we can derive the botnet growth model ($m = 2$):

$$
\frac{d\bar{I}(t)}{dt} = \frac{\eta}{\Omega} I_1(t) N
$$

$$
\frac{dI_1(t)}{dt} = \frac{\eta}{\Omega} [I(t) - I_1(t) - \bar{I}(t)] N - \frac{d\bar{I}(t)}{dt}
$$

$$
\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t)[N - I(t)] \tag{4}
$$

The above two models assume that the spearhead in host $A$ can download and execute the main-force immediately after it infects $m$ target hosts, which means we assume that at least one of the hosts in $A$’s peer list contains the main-force when $A$ wants to download the main-force. If the size of the peer list is not too small, this assumption is accurate for modeling purposes.

We use Matlab Simulink [1] to derive the numerical solutions of model (3) and model (4). We use the Code Red worm parameters [28], $N = 360,000$, $\eta = 358/\text{min}$, $\Omega = 2^{32}$ in the calculation and assume one initially infected host. Fig. 3 shows the worm propagation and the botnet growth over time. The propagation speed relationship would be similar for any other set of worm parameters. This figure shows that the botnet is constructed with a certain time delay (depends on $m$) as the

![Figure 3: Worm propagation and the constructed botnet growth](image)
worm spreads, but in the end all infected hosts will join the P2P botnet. This shows that the method described could potentially produce a viable and large botnet capable of avoiding current botnet monitoring techniques quite rapidly.

2.4 Experiment Proposal

2.4.1 Experiment Goal

We would like to leverage PlanetLab to simulate a P2P botnet’s propagation. We are trying to first verify our model and analysis of the propagation time delay of a honeypot-aware P2P botnet presented in Section 2.3.2; then we want to compare it with a similar P2P botnet but without honeypot detection capability, in order to show the feasibility and the tradoff of developing a honeypot-aware P2P botnet.

2.4.2 Experiment Implementation

We will simulate the following two types of P2P botnet, and record the botnet propagation process.

- A P2P botnet without honeypot detection capability, which relies on a random-scanning worm to propagate;
- A honeypot-aware botnet which relies on a two-stage reconnaissance worm to propagate.

We will be using OverSim [8], an open source P2P network simulator as the basis of our botnet implementation. And the SingleHost underlay model provided by OverSim, i.e., each OverSim instance only emulates a single host, which can be connected to other instances over existing networks like the Internet, is the model we are going to employ. This model perfectly fits the real network, like PlanetLab.

The botnet that will be implemented is a P2P botnet that uses a two-stage reconnaissance worm to distributively detect honeypots as it propagates and builds the botnet.

In the P2P botnet we simulate, there is one important parameter worth noting once again (Section 2.3.2), which is \( m \), the number of real vulnerable machines that an infected host needs to compromise, before it can download the main-force, join the botnet and become a real bot. When \( m = 0 \), this P2P botnet is a botnet without the honeypot detection capability, which is the baseline. And when \( m > 0 \), the botnet we simulate is a honeypot-aware botnet. It is obvious that the larger the value of \( m \) is, the longer the time delay of the botnet propagation is.

References


8


