# Modeling Peer-to-Peer Botnets

Elizabeth Van Ruitenbeek and William H. Sanders
Electrical and Computer Engineering Department,
Information Trust Institute, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL, USA
{evanrui2,whs}@uiuc.edu

## Abstract

*Peer-to-peer botnets are a relatively new yet rapidly growing Internet threat. In the year since its introduction in January 2007, the Storm Worm peer-to-peer botnet has become the largest botnet on the Internet. Unlike previous botnets operating over IRC channels, the Storm Worm botnet uses a decentralized peer-to-peer network to communicate among the bots and to control their computing power. While a centralized control structure can be toppled relatively easily by finding and disconnecting the head, a decentralized control structure is much harder to dismantle. Given this reality, security researchers must find new ways to defend against peer-to-peer botnets. Toward that aim, we have developed a stochastic model of peer-to-peer botnet formation to provide insight on possible defense tactics. We use the stochastic model to examine how different factors impact the growth of the botnet. Simulation results from the model evaluate the effectiveness both of prevention measures and of detection and disinfection methods. In this way, the simulation results from our peer-to-peer botnet model provide guidance for the design of future anti-malware systems.*

## 1. Introduction

Botnets are "networks of compromised machines under the control of an attacker" [11], and such networks are a growing threat on the Internet. After the attacker gains control of compromised computers, the attacker is able to harness the computing power of those compromised machines.

Attackers who operate botnets use the distributed computing power for a variety of nefarious purposes. The computers that comprise a botnet can be used to send and distribute spam email, to extort money from businesses wishing to avoid distributed denial of service attacks against their websites [14], and to commit click fraud [20]. The collective computational power of a botnet could even have the potential to shut down government agencies, utilities, or financial centers [9]. In short, there are mounting economic incentives for creating and maintaining botnets, and the threats posed by botnets cannot be ignored.

Most previous botnets have had a centralized command-and-control structure using Internet Relay Chat (IRC) channels. In contrast, peer-to-peer botnets utilize a decentralized command-and-control structure. While a centralized control structure can be toppled relatively easily by finding and disconnecting the head, a decentralized control structure is much harder to dismantle. Thus, peer-to-peer botnets represent an escalation in the increasingly sophisticated arms race between attackers and defenders.

Therefore, security researchers must develop new methods to mitigate the threat posed by peer-to-peer botnets. Currently, security experts simply recommend "basic computer hygiene" [22] to minimize the spread of computer malware including botnet infections. Such measures include avoiding infection from email attachments, using rootkit-detection systems, and blocking all peer-to-peer connections. However, blocking all peer-to-peer traffic may be too restrictive, and other mitigation techniques may be ineffective. The growth in botnets cannot go unchallenged, so more research on cost-efficient and effective anti-malware systems is necessary.

This paper presents a stochastic model of peer-to-peer botnet formation to provide insight on possible defense tactics. Section 2 discusses the evolution of the Storm Worm peer-to-peer botnet, and Section 3 describes the peer-to-peer functionality of that botnet. Then Section 4 describes our peer-to-peer botnet model. Section 5 analyzes simulation results from the model, and Section 6 discusses the implications of those results. The paper concludes with a discussion on future uses of the peer-to-peer botnet model.

The contributions of this work include the following:

- A stochastic model of peer-to-peer botnet formation
- Insight on how anti-malware companies can target their efforts to fight botnet formation efficiently
- Insight on the necessary level of anti-malware performance to fight botnet formation effectively

## 2. History of the Storm Worm

The first widespread peer-to-peer botnet appeared in January 2007 and is most commonly known as the Storm Worm, even though the malware is more accurately labeled a Trojan horse, not a worm. Antivirus firms have also given the malware other names including Small.DAM, Peacomm, Nuware, and Zhelatin [7]. The probable attackers are thought to be a well-organized group in Russia [17].

The Storm Worm has evolved significantly in the first year after its initial appearance. The attack process relies heavily on social engineering tactics. When the botnet was originally created, the main attack vector was through infected email attachments. Victims would receive an email with a subject line such as "A killer at 11, he's free at 21 and kill again," "U.S. Secretary of State Condolezza Rice has kicked German Chancellor Angela Merkel," "Naked teens attack home director," "230 dead as storm batters Europe," and others [12]. The Storm Worm name refers to the weather-related subject in some of the emails. The body of the email was empty.

The Storm Worm email contained an attachment with names such as "FullVideo.exe," "Full Story.exe," "Video.exe," "Read More.exe," or "FullClip.exe"; however, the attachment was a Trojan horse program, not a video clip [12]. When victims attempted to open the attached file, the executable inserted a kernel mode driver component called *wincom32.sys* and an initialization file component called *peers.ini* [4]. The malware also inserted itself into the services.exe process [12]. After the initial Trojan had been installed, it attempted to connect with peers in the Storm Worm botnet and, subsequently, to download the full payload and begin executing code under the control of the botnet. Section 3 will further describe the role of peer-to-peer networking in the Storm Worm.

For the antivirus firms, one of the most challenging aspects of the Storm Worm is the sheer number of variants. Titillating new email subject lines and attachment file names are added sometimes daily [16]. In addition, the back-end servers for Storm Worm re-encode the malware binary twice per hour using polymorphic techniques to minimize effectiveness of signature-based detection [16].

Over the course of the year, Storm Worm has also expanded its infection mechanisms from email attachments to email with links to infected sites and e-card spam that installs rootkits in the infected computer [10]. The links to infected sites are sometimes disguised as fake log-in links [15].

In response to the Storm Worm, antivirus firms have worked diligently on detection and removal. The biggest improvement occurred when the Microsoft Malicious Software Removal tool issued a patch in September 2007 that was correlated with a significant 20% drop in the Storm Worm botnet size [19]. This shows that efforts to remove machines from a botnet can make progress in reducing the size of the botnet.

Estimates of the size of the Storm Worm botnet vary widely. Reports in March 2007 claimed that the size of the Storm Worm botnet was between 20,000 and 100,000 computers [16]. In September 2007, others claimed the size of the Storm Worm botnet was between one and two million machines, although only 10% of the compromised machines were active at one time [9]. In November 2007, Storm was considered the world's largest botnet with 230,000 active members per 24-hour period [13]. The highest estimates say that the Storm Worm botnet could contain as many as 50 million compromised machines [9].

Regardless of its exact size, researchers agree that the Storm Worm botnet is currently the world's largest botnet. Due to its size, the Storm Worm botnet contributes significantly to the malware traffic on the Internet. On August 22, 2007, 57 million virus-infected messages crossed the Internet, and 99% were from the Storm Worm [9].

The most recent reports show that the Storm Worm botnet is being divided up because it may be more economically viable for the operators of the Storm Worm botnet to run several smaller botnets [13]. Larger botnets are more conspicuous and easier to detect. The Storm Worm botnet has begun the use of 40-bit encryption keys, which could be used to segment the botnet [23]. According to security researcher Joe Stewart, "This could be a precursor to selling Storm to other spammers, as an end-to-end spam botnet system" [23].

The threat of peer-to-peer botnets does not stop with the Storm Worm. Researchers indicate that a new peer-to-peer botnet is emerging that could some day surpass the size and sophistication of the Storm Worm botnet [13].

## 3. Peer-to-Peer Functionality in the Storm Worm Botnet

The Storm Worm botnet is the first major botnet to use peer-to-peer networking for command and control of the bots. As described earlier in Section 2, the initial infection involves a Trojan installing some files onto a vulnerable machine. However, a compromised machine is not part of the botnet until it joins the peer-to-peer network that controls the botnet.

The Storm Worm botnet uses the Overnet protocol, based on the Kademlia algorithm, for peer-to-peer networking. The Kademlia algorithm uses a system of distributed hash tables to support the decentralized architecture of peer-to-peer botnets [18].

After the Trojan installs the initial infection files and inserts the malware into the services process, the malware begins its bootstrap process. The infection files include a list of 146 nodes that are already part of the Storm Worm botnet, so the newly compromised computer attempts to connect to those nodes and become a peer. After successfully joining the network, the new bot updates its list of peers to find other close peers. The new bot also searches the network to find an encrypted URL that points to the secondary injection payload. The bot decrypts this URL and downloads the secondary injection code. This secondary injection code then executes on the bot. The secondary injection code is the code that sends the spam emails, participates in DDoS attacks, or performs other malicious botnet activities. The bot can also be programmed to periodically search for updates using the peer-to-peer network [11].

The computational power of the Storm Worm botnet has been used in a variety of activities. According to Symantec's research on the spam-generating machines in the Storm Worm botnet, "about half of those machines were only sending spam and the remainder were acting as HTTP servers, hosting the exploits and binaries used to infect new machines, and as SMTP servers for relaying spam" [8]. Some of this spam is used to recruit new bots into the botnet, but other spam is used for pump-and-dump stock scams [9].

The Storm Worm botnet is also notorious for DDoS attacks on anti-spam websites [9]. Automated DDoS attacks have been launched against security researchers probing the Storm Worm botnet.

The peer-to-peer communication of the Storm Worm botnet makes it difficult to measure the size of the botnet, but some researchers, including Brandon Enright, have tried [7]. Enright developed code called Stormdrain that crawls the Storm peer-to-peer network to probe the state of the bots in the network. His work examined how Storm bots transitioned between states. Our Möbius model builds on this concept of bot state transitions over time.

## 4. Modeling Peer-to-Peer Botnets

Security researchers continually strive to better understand the malware against which they must protect computers. Since peer-to-peer botnets will likely become more common in the future, the propagation and communication mechanisms of peer-to-peer botnets should be carefully studied. To that end, this paper presents a stochastic model of the creation of a peer-to-peer botnet. By varying input parameters, this model provides insight on the factors that affect how quickly a botnet is able to increase its size. These insights can help guide the development of anti-malware systems that are effective against peer-to-peer botnets. The botnet model is an abstraction of the creation of a peer-to-peer botnet based loosely on the Storm Worm botnet. Many of the input parameter values are based on information provided by Michael Bailey [1] of the University of Michigan concerning the behavior of the Storm Worm botnet.

Because botnets are a relatively new phenomenon, the body of scholarly research on botnets is still quite small. Since peer-to-peer botnets are a recent development, most of the existing scholarly research has focused on IRC-based botnets. Barford and Yegneswaran [2] classify and codify the capabilities of four common botnet codebases, but the four example codebases rely on IRC communication, not peer-to-peer communication. Cooke, Jahanian, and McPherson [3] also provide insight on IRC-based botnets using measurements from a bot honeypot network. They briefly mention the possibility of peer-to-peer botnet communication. Rajab et al. [21] discuss the difficulties in estimating the size of botnets. They focus on tracking IRC botnets and admit that peer-to-peer botnets would bring a whole new set of challenges to botnet tracking.

In the area of botnet modeling, Dagon, Zou, and Lee [5] have monitored botnets and detected diurnal patterns that affect the botnet propagation rate and attack strength. They created a diurnal propagation model to reflect this pattern. Their work also focuses on botnets with centralized command-and-control structures. By exploring the creation process for decentralized peer-to-peer botnets, our botnet model represents a step beyond the existing scholarly work on IRC botnets.

The botnet stochastic model was constructed in the Möbius software tool [6], which was designed to perform discrete event simulation and compute analytical/numerical solutions of models represented in a variety of modeling formalisms. The Möbius botnet model represents the system of computers infected as bots, and Möbius solvers run the botnet model and measure the size of the botnet over time.

In the model, computers joining the peer-to-peer botnet progress through a series of stages. These stages are modeled in the stochastic activity network (SAN) model as extended places (light-colored circles, as shown in Figure 1). Each extended place maintains a count of how many computers (represented as tokens) are in that stage[1].

Computers represented in the botnet model move through many different stages. When a compromised machine has first been infected by the initial Trojan, the computer is added to stage *InitialBotInfection* (i.e., a token is

[1] In Möbius, a place is stored as a short, but an extended place can be stored as a double. Extended places are used in this model to enable storing large numbers.

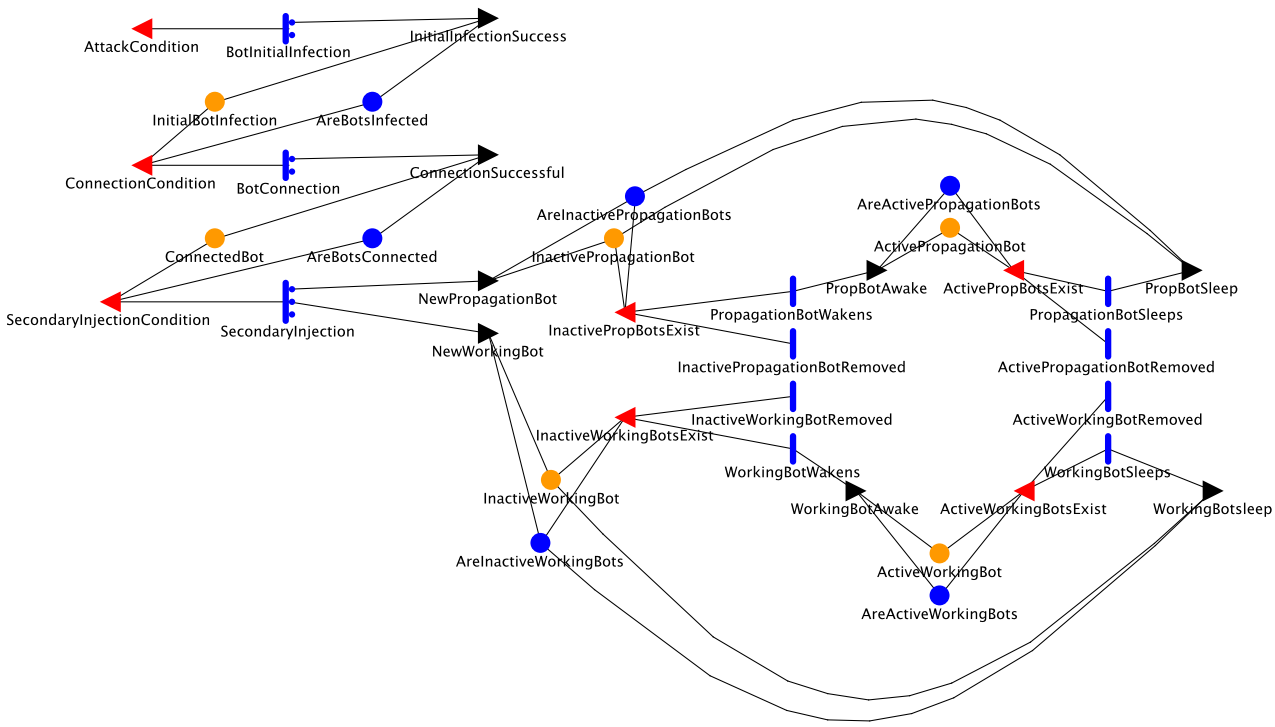


**Figure 1. Botnet SAN Model**

added to extended place *InitialBotInfection*). After the compromised machine successfully connects with peers in the network, the computer is moved to stage *ConnectedBot* (i.e., a token is removed from extended place *InitialBotInfection*, and a token is added to extended place *ConnectedBot*). Then the computer downloads the secondary malware injection, joins the botnet, and is moved to either stage *InactivePropagationBot* or stage *InactiveWorkingBot*.

Compromised computers in the botnet exist in one of four stages: *ActivePropagationBot*, *ActiveWorkingBot*, *InactivePropagationBot*, or *InactiveWorkingBot*. The distinction is made between propagation bots and working bots. The propagation bots are programmed to propagate the botnet infection and recruit new computers into the botnet, while the working bots are programmed to send spam, run "pump and dump" stock scams, and conduct other nefarious business. When computers first join the botnet, they are assigned to be either propagation bots or working bots.

The distinction is also made between active and inactive bots. To minimize the probability of detection, the malware installed on any given bot is only active a small fraction of the time. The inactive bots have a much lower probability of detection and, thus, a much lower probability of being disinfected and removed from the botnet. Over time, the propagation bots cycle between stages *InactivePropagationBot* and *ActivePropagationBot*. The working bots cycle between stages *InactiveWorkingBot* and *ActiveWorkingBot*.

The transitions of computers between botnet stages are indicated in the SAN model by the vertical bars (see Figure 1). The transition of computers into the *InitialBotInfection* stage occurs with an exponential distribution whose rate depends on the current number of computers in the *ActivePropagationBot* stage. When the botnet is larger, there are more active propagation bots, and computers can be recruited into the botnet at a higher rate. However, the number of computers entering the *InitialBotInfection* stage also depends on *ProbInstallInitialInfection*, the probability that users who receive the infected spam will activate the infection by opening the attachment or clicking on the link to an infected website. In the simulations of this model, we vary *ProbInstallInitialInfection* to examine the effectiveness of anti-malware measures that minimize this probability.

Next, the transition of computers from the *InitialBotInfection* stage to the *ConnectedBot* stage occurs with an exponential distribution whose rate depends on the number of computers in stage *ConnectedBot*. Similar to the previous transition, the model specifies that infected computers successfully connect to their peers with probability *ProbConnectToPeers*.

Computers in the stage *ConnectedBot* have established a connection with the peer-to-peer network but await the secondary injection of malware code from their peers. The transition from the *ConnectedBot* stage to the *InactivePropagationBot* or *InactiveWorkingBot* stage has an exponential

distribution whose rate depends on the number of computers in the *ConnectedBot* stage. The probability *ProbPropagationBot* determines the fraction of new bots that are assigned as propagation bots. The remainder of the new bots are assigned as working bots. As before, computers successfully complete the final step of joining the botnet with the probability *ProbSecondaryInjectionSuccessful*. In all three of the transitions described above, those computers that do not successfully move to the next stage in the infection process are removed from the model.

Each computer that joins the botnet must travel once through the above three transitions. As a member of the botnet, the computer then cycles between inactive and active stages. The transition of computers from stage *InactivePropagationBot* to *ActivePropagationBot* occurs with an exponential distribution whose rate depends on the number of computers in stage *InactivePropagationBot*. Similarly, the transition of computers from stage *InactiveWorkingBot* to *ActiveWorkingBot* occurs with an exponential distribution whose rate depends on the number of computers in stage *InactiveWorkingBot*. In the same way, the transition of computers from an active stage to the respective inactive stage occurs with an exponential distribution whose rate depends on the number of computers in that active stage.

The final set of transitions in the model represent the removal of active or inactive bots from the botnet. These removal transitions are represented as competing exponentials. For example, when a bot is in the stage *ActivePropagationBot*, it can transition into stage *InactivePropagationBot*, or it can be detected and removed from the botnet. The relative rates of these competing exponentials determine the probability that the bot is detected and removed from the botnet before it transitions to an inactive stage.

The removal transitions represent when the compromised machines are completely removed from the botnet whether due to effective anti-malware actions or physical disconnection from the Internet. In the simulations of this model, we vary the rate at which compromised machines are removed from the botnet to examine the effectiveness of detection and disinfection techniques.

The remaining components in the botnet model are the input gates and output gates. The input gates are the left-pointing triangles connected to the transition activities. The input gates contain an expression that must be true to enable the activity to occur. In each of the input gates in the botnet model, the expression checks that the value used to calculate the rate of the exponential distribution is nonzero. The input gates also contain function code that executes each time the transition activity occurs. In most of the input gates in the model, this code removes a token from the previous stage. Figure 2 provides an example of the input gate code for *ConnectionCondition*. The output gates are right-pointing triangles that contain function code to execute after the activity fires. In the output gates in the model, this code adds a token to the next stage to represent the transition that occurred. Figure 3 provides an example of the output gate code for *ConnectionSuccessful*.

```
Input Gate: ConnectionCondition
Predicate: (AreBotsInfected->Mark() > 0)
Function:
 InitialBotInfection->Count->Mark()--;
 if
 (InitialBotInfection->Count->Mark()==0)
  {AreBotsInfected->Mark() = 0;}
```

**Figure 2. ConnectionCondition code**

```
Output Gate: ConnectionSuccessful
Function:
 ConnectedBot->Count->Mark()++;
 AreBotsConnected->Mark()=1;
```

**Figure 3. ConnectionSuccessful code**

## 5 Simulation Results

After constructing the botnet stochastic model using the Möbius tool, we assigned values to the input parameters and generated the simulation results. In the following simulation experiments, we analyzed the significance of the order of magnitude of the differences between key input parameters, so exact input values were not necessary.

### 5.1 Simulation Set-up

The results were generated from the peer-to-peer botnet model using the terminating simulator in Möbius. For each combination of input parameters, 100 replications were performed, and the results were averaged. To track the expansion of the botnet, the measures of interest were defined and evaluated every 60 minutes of simulated time over a period of one week of simulated time. The measures of interest include the instantaneous number of propagation and working bots—both active and inactive. Other measures include the number of machines in the *InitialBotInfection* stage and in the *ConnectedBot* stage, as well as the number of bots removed from the botnet during each hour. This paper will display results in terms of the number of propagation bots because this number is critical when studying the rapid formation of a large botnet.

At the beginning of each simulation replication, the extended place and place values are initialized to zero—with the exception of the *ActivePropagationBot* stage and its flag

**Table 1. Variable Values for Baseline Experiment (Rates are per minute)**

| Variable | Value |
|---|---|
| ProbConnectToPeers | 100% |
| ProbInstallInitialInfection | 10% |
| ProbPropagationBot | 10% |
| ProbSecondaryInjectionSuccessful | 100% |
| RateActivePropBotRemoved | 0.01 |
| RateActiveWorkingBotRemoved | 0.01 |
| RateConnectBotToPeers | 12.0 |
| RateInactivePropBotRemoved | 1.0e-4 |
| RateInactiveWorkingBotRemoved | 1.0e-4 |
| RateOfAttack | 10.0 |
| RatePropagationBotSleeps | 0.1 |
| RatePropagationBotWakens | 0.001 |
| RateSecondaryInjection | 14.0 |
| RateWorkingBotSleeps | 0.1 |
| RateWorkingBotWakens | 0.001 |

*AreActivePropagationBots*. To jumpstart the botnet infection process, the *ActivePropagationBot* stage begins with 200 machines, and the flag *AreActivePropagationBots* is set to one.

The first simulation was the baseline experiment, in which the input variables were assigned the values shown in Table 1 based on information provided by Michael Bailey [1] of the University of Michigan. All rates in the model are expressed per minute. In addition to the baseline botnet experiment, 14 more experiments were simulated. Six experiments examined the impact of reductions in the probabilities that a machine successfully completes the three stages of the infection process. Eight experiments examined the impact of changes in the rate at which bots are detected and removed from the botnet.

## 5.2 Probability Reduction Experiments

The aim of the probability reduction experiments was to gage how drastically the anti-malware system must lower the probability of machines joining botnets in order to significantly impact the growth of the botnet. Since the peer-to-peer botnet infection process occurs in several stages, one might ask if it matters where in the infection process the malware spread is stopped. To test this, we varied the probabilities at two different infection stages.

In these experiments, two parameters were varied separately. The parameter *ProbConnectToPeers* is 100% in the baseline experiment, and three experiments measured the impact of reducing this parameter value to 80%, 60%, and 40%. The parameter *ProbInstallInitialInfection* is 10% in the baseline experiment, and three additional experiments measured the impact of reducing this parameter value to 8%, 6%, and 4% (which would be 80%, 60%, and 40%, respectively, of the baseline value).

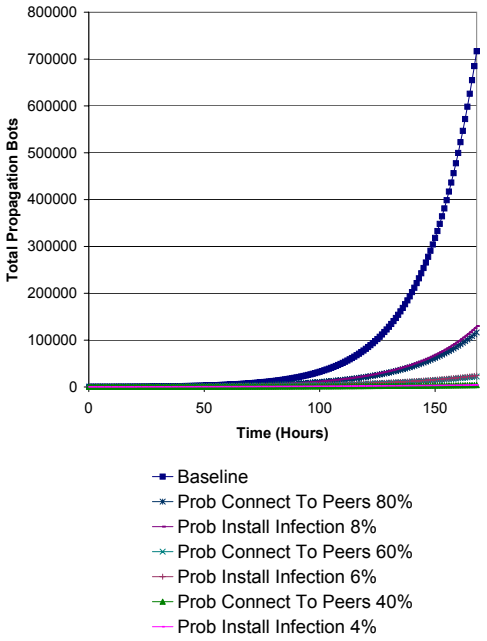


**Figure 4. Propagation Bot Growth Over One Week: Varying Infection Probabilities**

The full-scale graph of the simulation results is shown in Figure 4. This view of the results shows the reduction in botnet size in comparison to the baseline botnet size. At the end of one week of simulated time, the baseline botnet contains over 700,000 propagation bots. In contrast, for the two experiments in which the probability (*ProbConnectToPeers* or *ProbInstallInitialInfection*) was reduced to 80% of the baseline value, the botnet contains less than 150,000 propagation bots. Figure 5 provides a closer look at the data in Figure 4. In Figure 5, the topmost curve is still the baseline curve, and the next two curves are the two 80% experiments. However, it is now evident that the two 60% probability experiments are squeezed together, and the two 40% curves are also nearly identical. Thus, it appears that anti-malware can be designed to stop botnet infection at any stage in the infection process. The important consideration is that the malware be able to minimize as much as possible the probability of victims progressing through the complete infection process. For example, if one of the infection process probabilities is reduced to 20% of its baseline value, the botnet contains less than 5000 active propagation bots

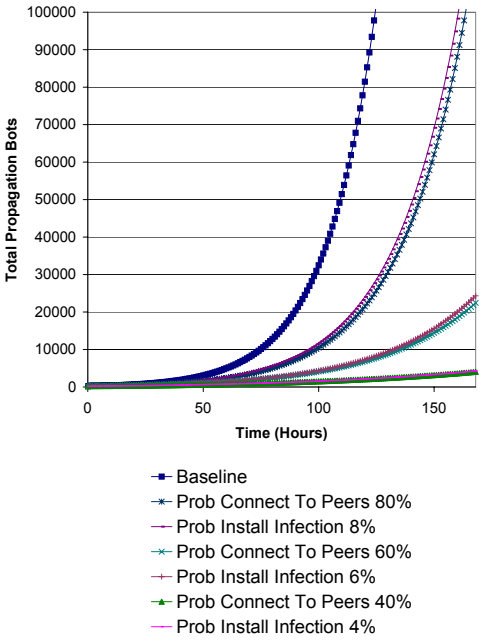


**Figure 5. Propagation Bot Growth Over One Week: Varying Infection Probabilities (A Closer Look)**

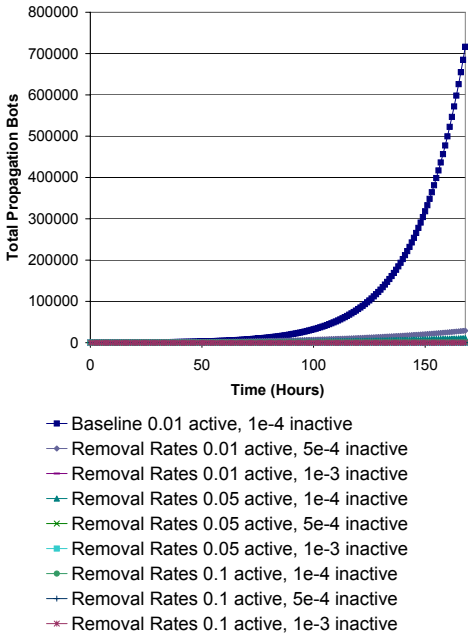


**Figure 6. Propagation Bot Growth Over One Week: Varying Bot Removal Rates**

at the end of one simulated week. This is a vast reduction from the baseline infection of 700,000.

### 5.3 Increased Removal Rate Experiments

The goal of the removal rate experiments was to discern how effective bot detection and removal must be in order for the anti-malware system to be able to fight botnet formation. The two removal rate parameters were each varied an order of magnitude in these experiments. While the baseline removal rate for active bots is 0.01 per active bot per minute, these experiments varied the active bot removal rate from 0.01 to 0.05 to 0.1. And while the baseline removal rate for inactive bots is 1.0e-4, that rate was varied from 1.0e-4 to 5.0e-4 to 1.0e-3. Because detection of an inactive bot is much more difficult than detection of an active bot, the removal rate is much lower.

Including the baseline experiment, we ran nine different experiments to test every combination of active and inactive bot removal rate listed above. The simulation results are shown in Figure 6. In comparing Figure 4 and Figure 6 (they have the same scale on the vertical axis), we can compare the relative effectiveness of the two approaches. The probability reduction experiments simply slow the growth rate of the botnet, while the removal rate experiments focus on whittling away at the size of the botnet. If an anti-malware system can remove bots from the botnet at a fast enough rate, then the botnet can actually decrease in size. When the simulation results are examined in more detail, then the results of the experiments can be grouped into three categories. The first category contains the results curves with low removal rates and positive botnet growth. In this case, the exponential growth of the botnet continues, albeit at a lower rate. Figure 7 is a closer look at the data in Figure 6, and Figure 8 provides an even closer look. The baseline experiment produces the topmost curve. The second topmost curve is produced by removal rates of 0.01 for active and 5.0e-4 for inactive propagation bots. The third curve is produced by removal rates of 0.05 for active and 1.0e-4 for inactive propagation bots. These three curves fit in the first category.

The second category contains the results curves with slightly higher removal rates and relatively flat botnet growth. Figure 8 shows that three curves fit in this category. The botnet growth in these three experiments seems to level off at approximately 1000 propagation bots. One curve in this category is produced by removal rates of 0.01

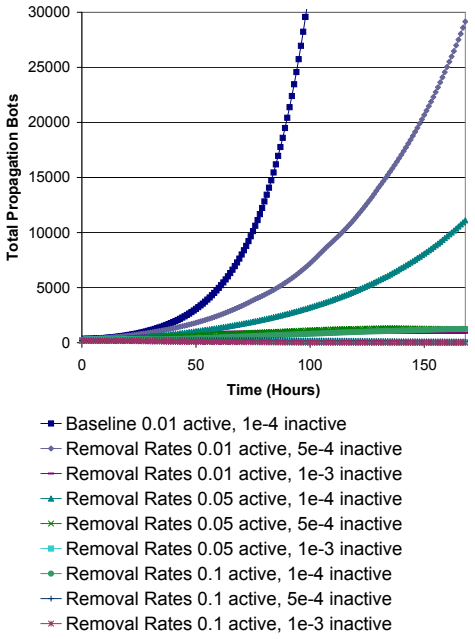


**Figure 7. Propagation Bot Growth Over One Week: Varying Bot Removal Rates (A Closer Look)**

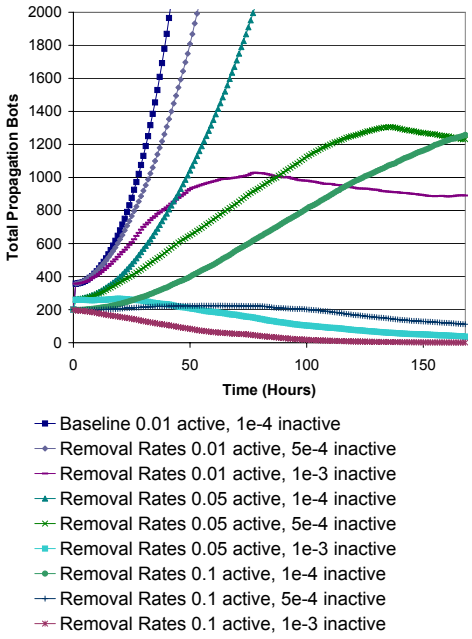


**Figure 8. Propagation Bot Growth Over One Week: Varying Bot Removal Rates (Yet a Closer Look)**

for active and 1.0e-3 for inactive propagation bots. The second curve is produced by removal rates of 0.05 for active and 5.0e-4 for inactive propagation bots. The third curve is produced by removals rates of 0.1 for active and 1.0e-4 for inactive propagation bots. This set of curves represent a balance between low and high removal rates.

The third and final category contains the results curves with the most aggressive removal rates and negative botnet growth. In this category, at least one of the two removal rates for each experiment is of the same magnitude as its competing exponential. The competing exponentials represent a propagation bot switching from active to inactive or vice versa. One curve in this category is produced by removal rates of 0.05 for active and 1.0e-3 for inactive propagation bots. The second curve is produced by removal rates of 0.1 for active and 5.0e-4 for inactive propagation bots. The final curve is produced by removal rates of 0.1 for active and 1.0e-3 for inactive propagation bots. These simulation results demonstrate that aggressive bot removal can minimize or reverse botnet growth. The caveat is that aggressive bot detection and removal can be very difficult, especially when the bot is in an inactive stage. However, these simulation results also show that even with poor detection during the inactive stage, a system with aggressive detection during the short periods of time when the bot is active can still greatly minimize botnet growth. The same balancing principle is evident when the bot detection and removal is poor during the active stage but aggressive during the inactive stage.

These type of results are useful for guiding the development of anti-malware products because the model provides insight on how botnets can be targeted most efficiently. If it is much easier to detect active bots, and the active bot detection rate can be raised high enough, then that path may be the more beneficial to pursue than inactive bot rootkit detection, even if the bot spends only a small fraction of time in the active state. Our peer-to-peer botnet stochastic model could be used to provide more specific guidance if the model parameters were fine-tuned to reflect a desired evaluation environment. Although the botnet model can provide useful insights, the model does contain a limitation that should be acknowledged. That is, the baseline botnet exhibits unbounded exponential growth. Clearly, in the real world there is an upper bound on the number of computers worldwide that can become infected. A real botnet cannot continue to grow indefinitely at an ever-increasing rate.

However, since we are using this model to study the formation of a botnet (namely the first week of the life of a rapidly growing botnet), this approximation is justified in the range in which we study the model.

## 6 Implications

The results from the experiments are discussed in terms of reducing infection process probabilities and increasing bot removal rates. What does this mean in terms of real anti-malware systems?

Anti-malware measures that could reduce the probability of botnet infection include signature- or anomaly-based antivirus products as well as measures to educate users on how to avoid computer malware. The simulation results indicate that even incomplete efforts to deploy anti-virus products could still be relatively effective in reducing the rate of botnet growth. Education can also be an effective prevention measure, even when education efforts are not able to convince the entire user population to avoid activating malware. Thus, security professionals should strive to educate computer users on how to avoid infection by malware.

Because botnet code can hide its presence on a computer, anti-malware measures to increase bot removal rates typically involve rootkit detection and removal products. The simulation results demonstrated that strong bot removal products could significantly reduce the rate of growth and even shrink the size of a botnet. The results show that if the removal rate is quick and aggressive enough, anti-malware systems could potentially bring down even a decentralized, peer-to-peer network.

Since the rate at which infected machines are removed from a botnet can significantly constrain the growth of the botnet, security researchers have good motivation for developing mechanisms to detect when machines become compromised so that those machines can be quarantined and disinfected. Ideally, the botnet would never be able to compromise machines; realistically, at least a few users will continue to fall prey to social engineering tactics that permit botnet rootkits to compromise their machines. Given this context, botnet rootkit detection and removal are important research areas for anti-malware.

## 7. Conclusions

Peer-to-peer botnets are a relatively new Internet threat that the Storm Worm brought to the attention of the popular media in 2007. Because peer-to-peer botnets have a decentralized command-and-control structure, they are more difficult to dismantle than previous IRC botnets with centralized control. Thus, more research is necessary to fully understand the threat and to prepare to defend against it.

In this paper, we presented a stochastic model of the creation of a peer-to-peer botnet. The model was inspired by the Storm Worm botnet but is a more general model of peer-to-peer botnets. The simulations of the botnet model tracked the growth of a botnet over a period of one week. The simulation results provided insight on the effectiveness of various anti-malware approaches. Simulations with this model demonstrated the effectiveness both of prevention measures (e.g., antivirus products and user education) and of detection and disinfection methods (e.g., rootkit detection and removal products).

Because peer-to-peer botnets are a new threat, there is much to study. The stochastic model we present here could be extended in future work to study the growth possibilities of future peer-to-peer botnets. The model could also be used to evaluate the effectiveness of other potential anti-malware techniques. While we have chosen to focus this paper on the insights valuable to anti-malware developers, it would also be possible to use our model to predict how botnet operators could create more potent and aggressive botnets. Such predictions could ultimately be useful to anti-malware developers as well.

## Acknowledgments

The authors would like to thank Michael Bailey of the University of Michigan for providing information on the Storm Worm botnet. The authors would also like to thank Brad Smith for providing additional sources of information on the Storm Worm botnet.

## References

[1] M. Bailey. Private communication, March 21, 2008.

[2] P. Barford and V. Yegneswaran. *Malware Detection*, volume 27 of *Advances in Information Security*, chapter An Inside Look at Botnets, pages 171–191. Springer US, 2007.

[3] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, pages 39–44, June 2005.

[4] F.-S. Corporation. F-secure malware information pages: Small.dam. *www.f-secure.com*, January 19, 2007.

[5] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *13th Annual Network and Distributed System Security Symposium (NDSS'06)*, 2006.

[6] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Trans. on Software Engineering*, 28(10):956–969, Oct. 2002.

[7] B. Enright. Exposing stormworm (powerpoint presentation). *ToorCon 9 Computer Security Conference*, October 2007.

[8] D. Fisher. Experts predict storm trojan's reign to continue. *searchsecurity.techtarget.com*, October 22, 2007.

[9] S. Gaudin. Storm worm botnet more powerful than top supercomputers. *Information Week*, September 6, 2007.

[10] S. Gaudin. Storm worm erupts into worst virus attack in 2 years. *Information Week*, July 24, 2007.

[11] J. B. Grizzard, V. Sharma, C. Nunnery, B. Byung, H. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[12] A. Hidalgo. Trojan.peacomm: Building a peer-to-peer botnet. *www.symantec.com*, January 19, 2007.

[13] K. J. Higgins. The world's biggest botnets. *www.darkreading.com*, November 9, 2007.

[14] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Second Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[15] R. Lemos. Group behind 'storm worm' changes tactics. *www.symantec.com*, August 24, 2007.

[16] R. Lemos. How 'storm worm' wreaked havoc with malware defenses. *www.symantec.com*, March 6, 2007.

[17] J. Leyden. Storm worm linked to spam surge. *www.channelregister.co.uk*, September 14, 2007.

[18] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer-Verlag, 2002.

[19] R. McMillan. Storm worm now just a squall. *www.pcworld.com*, October 21, 2007.

[20] C. T. News. Expert: Botnets no. 1 emerging internet threat. *www.cnn.com*, January 31, 2006.

[21] 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: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 5–5, Berkeley, CA, USA, 2007. USENIX Association.

[22] B. Smith. A storm (worm) is brewing. *Computer*, 41(2):20–22, February 2008.

[23] D. Utter. Storm botnets using encrypted traffic. *www.securitypronews.com*, October 16, 2007.