2010 Fourth International Conference on Network and System Security

# Overcoming Reputation and Proof-of-Work Systems in Botnets

Andrew White, Alan Tickle, Andrew Clark
Information Security Institute
Queensland University of Technology
Brisbane, Australia
{a13.white, ab.tickle, a.clark}@qut.edu.au

***Abstract*—Reputation and proof-of-work systems have been outlined as methods bot masters will soon use to defend their peer-to-peer botnets. These techniques are designed to prevent sybil attacks, such as those that led to the downfall of the Storm botnet. To evaluate the effectiveness of these techniques, a botnet that employed these techniques was simulated, and the amount of resources required to stage a successful sybil attack against it measured. While the proof-of-work system was found to increase the resources required for a successful sybil attack, the reputation system was found to lower the amount of resources required to disable the botnet.**

***Index Terms*—botnet; sybil; reputation; proof-of-work**

## I. Introduction

The rise in botnets is a reflection of a shift towards writing malware for profit, rather than out of curiosity. These botnets are networks of infected machines, controlled by a single entity, that provide their owner all the functionality required to generate a profit. This profit is typically generated by using the infected machines for malicious purposes such as sending spam, stealing personal and financial details, and performing denial of service attacks. For the bot masters, those who create and operate these botnets, their livelihood depends on the capability of the botnet to continue these malicious activities and as such, it is in the best interests of the bot master to ensure that their botnet operates for as long as possible. To achieve this, the bot masters have been continually evolving the defence mechanisms their botnets employ, to avoid having their botnets disabled by new countermeasures.

One threat to the existence of peer-to-peer botnets is the *sybil attack* [1], whereby thousands of fake peers are introduced onto the botnet. These fake peers act as a collective entity in order to manipulate botnet communications, or in this case, disrupt them, such that the botnet can no longer function. Reputation and proof-of-work systems have been highlighted as the best methods available to botnets for preventing this sort of attack in theory [2], but have yet to be evaluated empirically.

In this paper we have evaluated the impact that reputation and proof-of-work systems will have on the capability of the security community to disable botnets, by measuring the amount of resources required to disable botnets employing such techniques. This was achieved through the creation of a botnet simulation framework, which was used to simulate the hypothetical botnet described by Hund et al. [2], and then perform sybil attacks against it. We found that the reputation system made the botnet more vulnerable to the sybil attack, contrary to its intended effect, making it unlikely to be implemented in botnets in the near future. The use of the proof-of-work system however was found to require significantly more resources to disable, indicating that should this system be implemented it would severely impact the security community's capability to disable peer-to-peer botnets.

The remainder of this paper is structured as follows. Section II gives an overview of the current state of botnets and the techniques available to disable them. Section III describes the design of the simulated botnet, and how the techniques it employs help to prevent the sybil attack. Section IV introduces the simulation framework we employed, and describes the methodology of our experiments. Section V provides the results of these experiments, before concluding the paper in Section VI.

## II. Background

In order for a botnet to function, the bot master and the bots must be able to communicate. The way in which this is achieved is known as the command and control structure. Early botnets distributed commands by having infected machines join a particular room on an Internet Relay Chat server [3]. The use of this chat room introduced a central point of failure, that if shut down, would completely disable the botnet. This weakness was heavily exploited by researchers, who studied botnet infections to see where the bots connected to for commands, and then had the servers shut down [4].

Bot masters soon learned their lesson, and began to implement more sophisticated command and control structures, such as peer-to-peer (P2P) networks. By employing P2P techniques, similar to those used in file-sharing networks, the single point of failure that had previously been targeted no longer existed [4]. However, while one vulnerability was removed, another one was introduced, in that for the P2P network to function, each peer must blindly trust other peers and the information they send [5].

One method of exploiting this flaw is known as the *sybil attack*, whereby the lack of identity verification is exploited to create multiple identities [1]. This technique can be used to create thousands of fake identities on the network, known as *sybils*, to allow an individual to have a larger influence

978-0-7695-4159-4/10 $26.00 © 2010 IEEE
DOI 10.1109/NSS.2010.65

IEEE computer society

over the network than they would otherwise be entitled. In P2P botnets, this technique can be used to disrupt the flow of commands through the network, to the point where the botnet is no longer able to function.

This kind of attack was successfully performed against the Storm botnet, where sybils were used to publish benign commands onto the botnet, preventing bots from retrieving the malicious commands sent by the bot master [4]. This particular method is known as the *pollution attack*, and is only effective against botnets that are built on file-sharing P2P protocols. Botnets that employ P2P techniques which are not built using such file-sharing protocols, such as Conficker [6] and Waledac [7], feature no publishing mechanism, and therefore are unaffected by this technique.

Few studies have been performed on the effectiveness of sybil attacks on peer-to-peer botnets. Two such studies, one by Davis et al. [8], and one by Wang et al. [9], both focus on disabling the Storm botnet, but provide conflicting results as to which is the best strategy to use. While perhaps this is the result of the different methods with which they went about their studies, the end result is that there is some confusion as to the best method of conducting sybil attacks against P2P botnets. Unfortunately the differences in design between the botnet we simulated and the Storm botnet are significant enough to cause us to be unable to explore this area further.

Hund et al. [2] proposed that reputation and proof-of-work systems were possible methods that bot masters will employ in order to defend their botnets against the sybil attack. Their paper outlined a theoretical botnet design, named *Rambot*, that employed these techniques, which we have used as the specification for the botnet simulated in our experiments. While the two proposed techniques employed different methods, they both attempted to provide a method of distinguishing legitimate bots from fake bots.

Various methods of creating artificial botnets have been attempted. Barfod and Blodgett [10] created an isolated network in which real botnet infections were run on virtual machines. Only five instances per machine were achieved, limiting this method to small simulations before significant expenses are incurred. In addition, no current botnets implement reputation or proof-of-work systems, making acquiring the botnet infections for this method impossible without developing a real botnet ourselves.

Another option is to synthetically simulate a botnet, as was done by Davis et al. [8] in their examination of the sybil attack against the Storm botnet. To achieve this, the functionality of the botnet relevant to the experiments, such as the command and control structure, was abstracted out and simulated using graph theory. This approach allowed the simulation of tens of thousands of nodes on a single machine, making it a far superior method for experiments on large populations than that of Barfod and Blodgett [10]. While Davis et al. [8] simulated their botnet using graph theory, with the inclusion of the reputation and proof-of-work systems, this method was deemed too complex and prone to error for our purposes.

The concept of abstracting out relevant behaviour is similar to that used in agent-based models, which have been shown to be a useful tool in modeling peer-to-peer networks [11]. However, their focus on agents reacting to the environment, rather than other agents, made this an undesirable option. In addition, other complex behaviour, such as the collaboration and shared resources between sybils would have been difficult to implement in these tools.

Since no botnet code existed, it was decided that developing a synthetic botnet simulation would not only be faster than developing a real botnet, but much more efficient for the desired experiments. While we decided to utilise the concept of abstracting out relevant bot behaviour for our experiments, instead of using available tools we created our own botnet simulation framework for this purpose.

## III. Botnet Design

The botnet to be simulated features a peer-to-peer command and control structure, based on the ideal botnet suggested by Hund et al. [2], the paper that introduced the concept of using reputation and proof-of-work systems in a botnet. This section details the functionality of the botnet that was implemented, and how the reputation and proof-of-work systems are designed to hinder the sybil attack.

### A. Peer-to-Peer Protocol

In this botnet's peer-to-peer protocol, bots connect directly to each other in order to form a distribution network for bot master commands. When a bot receives a command, it forwards the command to all the bots it is connected to. These bots then forward the command to all the bots they are connected to, and this process continues until all bots have received the command. Since a bot only forwards a command the first time it is received, once all bots have seen the command the distribution process will automatically stop. This allows the bot master to spread commands throughout the botnet by simply selecting a bot at random and sending it a command.

The command distribution process for this P2P protocol is shown in Figure 1. In the first step, the bot master randomly selects a bot from the botnet and connects directly to it. This allows the bot master to send a cryptographically signed command directly to the selected bot, as is indicated in step 2. Upon receiving this command, the bot verifies the command, and then forwards it to all the bots it is connected to in step 3. The distribution process then continues in step 4, with all bots that have received the command forwarding it to the bots they are connected to, until all bots within the botnet have seen the command.

All commands sent by the bot master are cryptographically signed, in order to prevent the possibility of others injecting commands onto the botnet. During the simulations, this process was considered cryptographically secure, and therefore the commands themselves were considered immune to exploitation. Instead of focusing on exploiting the commands, the focus was on exploiting the way in which these commands were distributed through the botnet using the P2P protocol.

As part of the P2P protocol, each bot within the botnet is responsible for maintaining a list of all other bots that it knows. This list is used to randomly select peers to connect to, both when the bot comes online, and when the number of active connections does not exceed a set threshold. In this peer-to-peer protocol, connections made between two bots persist until one of the participating bots goes offline. This allows commands to quickly propagate throughout the botnet, as each bot is permanently connected to numerous other bots.

The implication of this design however is that these connections change only as a result of bots within the botnet turning on and off, meaning that the structure of the botnet changes very slowly over time. This allows the sybils the time to position themselves in the network to block the propagation of bot master commands. An alternative strategy, such as forming the connections on demand when a command is received, and closing them immediately after the command is sent, could have a significant impact on the effectiveness and strategies used to perform the sybil attack.

Bots initially only know the few bots that they are provided with as part of the infection process, but learn about other bots by exchanging a fragment of their peer lists upon successful connection. Only a fragment is exchanged in order to prevent crawling of the botnet, as was done against the Storm botnet by Holz et al. [4]. In the event a connection initiated by a bot is unsuccessful, either because the destination bot is offline, or has reached the specified connection limit, the bot is removed from the peer list. A new bot is then selected randomly from the list, and the process continues until sufficient connections to the botnet have been made.

Should a bot's peer list be empty as a result of all previously known bots being unreachable, a backup service is contacted to retrieve a random list of peers. This backup service could take the form of a web server on a certain address, or searching for content on an existing P2P networking using specific keywords. For the purposes of our simulations however, we assumed that this service was easily reachable, and simply produced a random selection of all known peers. Given the focus on utilising the sybil attack, the use of this service was considered immune to abuse.

At present, this P2P network design features no methods in which to deal with a sybil attack. By introducing a number of sybils onto the botnet, which behave similarly to regular bots, except that they do not forward commands, it is possible to isolate the botnet into numerous subgroups which cannot communicate. This particular variant of the sybil attack is known as the *eclipse attack* [12].

An example of the eclipse attack in action is shown in Figure 2. Similar to the process shown in Figure 1, the bot master connects to a bot at random in step 1, and sends the bot a command in step 2. In step 3, the bot sends the command to the bots it is connected to, to continue the command distribution process. The difference however is shown in step 4, where the sybils intentionally do not forward the command, resulting in bots within the botnet not receiving the command. In all the botnets we simulated, this was how the sybils behaved, they would connect to multiple legitimate peers and not forward botnet commands.

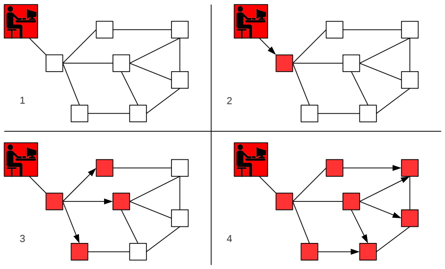

Fig. 1. Command Distribution Process

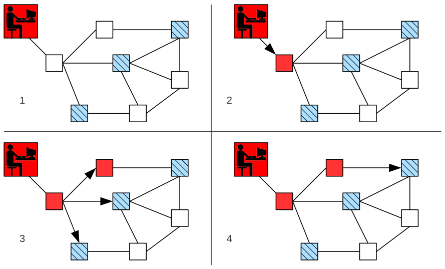

Fig. 2. Eclipsing the Command Distribution Process

Since botnets rely on the bots receiving the commands in order to function, by restricting the flow of commands throughout the botnet, the capability of the botnet to function can be severely crippled. While ideally it would be desirable to perform the eclipse attack such that every legitimate botnet node only connects to sybils, the size of modern botnets makes this infeasible. Instead, the goal of eclipse attacks on the botnet would be to minimize the number of bots that are able to receive bot master commands, by approaching as close to the ideal attack as possible.

It is in the best interests of the sybils to exploit the botnet protocol wherever possible, in order to increase the impact of each sybil node. This can be achieved by ensuring that when sybils exchange peer lists only other sybils are exchanged, and that the sybils maintain a shared peer list so that all sybils know about as many bots as possible. Heavily coordinating the sybil actions allows this to be taken further, as by centrally controlling the sybils listed in peer exchanges and the connections made by sybils, the knowledge of the sybils by the botnet can be maximized.

### *B. Reputation System*

One identified method to prevent such eclipse attacks is to employ a reputation system [2]. Reputation systems introduce a metric of trust between peers in the botnet, allowing peers that do not follow the rules to be excluded. This is achieved by prioritizing the peers that are connected to based on their reputation, such that the peers with the highest reputation are connected to first, thereby avoiding connections to peers with lower reputations.

In the reputation system implemented in the simulated botnet, each bot is responsible for maintaining a list with the reputations of all bots it has come into contact with. This list is kept in addition to the peer list previously described for forming connections, and is never shared, meaning that the reputation system is not distributed. As such, reputation earned by a bot is only relevant to the bot it has earned the reputation with. While this makes the implementation of the reputation system simple, it also prevents the possibility of protocol flaws which allow reputation to be fraudulently earned.

Since no information in the reputation system is shared, only information on positive contributions are available, meaning that the reputation will be solely based on the amount of good work done [13]. As such, points in the reputation system are earned by correctly forwarding bot master commands. By introducing a time frame into each command, which indicates the period of time in which the command is valid, the failure to receive a command from a bot within this time frame can be used as an indicator that this good work was not done. Consequently, should a bot not correctly forward a command, it will lose points in the reputation system. Additionally, this time frame can be used to prevent bots artificially inflating their reputation by continually forwarding the same command.

In addition to providing a mechanism to gauge the trustworthiness of a peer, the way in which reputation is earned allows it to also function as an indication of the bot's availability. It is for this reason that bots to connect to are selected from the reputation list based on their reputation, rather than randomly as in the regular peer list. Rather than replace the regular peer list entirely, the reputation peer list simply augments the connection process, allowing bots from both lists to be selected to connect to.

Bots are removed from the reputation list if their reputation falls below a threshold value. This can be the result of bots not forwarding commands correctly, or being unavailable for extended periods of time. Unlike the regular peer list, bots that are unable to be connected to are not immediately removed from the list, but rather remain on the list until their reputation drops enough to fall below the threshold.

The addition of the reputation system forms another hurdle sybils must overcome in order to successfully cripple the botnet's capabilities. For the sybils to dominate all of a bot's connections, they must not only infiltrate the bot's regular peer list, but also the reputation list. Without rising to the top of the reputation list, the bot will continue to form connections to the same legitimate bots, preventing the sybils from isolating the bots into subgroups.

In order to rise to the top of this list however, the sybils must forward bot master commands, thereby contributing to the structure and operation of the botnet. Not only does this assist the botnet, but participating in such activities raises ethical and legal issues [14]. While various sybil strategies could be employed here, such as forwarding commands for a limited period of time, we decided to have the sybils simply not forward any bot master commands. This allows the effect of the reputation system to be accurately measured and directly compared to only employing the basic P2P protocol.

### C. *Proof-of-Work System*

Proof-of-work puzzles are cryptographic puzzles that are time consuming for a client to solve, but are able to be easily and quickly verified by the server. These proof-of-work puzzles can be used to validate the intentions of a machine wishing to connect to a server, by forcing them to spend time and resources solving the problem. For a botnet, these proof-of-work puzzles can be used during the connection process between bots, in an attempt to verify that the connecting machine is a legitimate bot.

In the simulated botnet, the proof-of-work scheme takes place during the connection process, before the exchange of peer lists. The bot accepting the connection functions as the server, generating the puzzle, and the bot that initiated the connection is required to solve the proof-of-work task in order to verify their intentions and complete the connection process. Hund et al. [2] suggested a proof-of-work average solution time of 5 minutes, so this is what was used in our simulations unless specified otherwise.

Due to the length of the solution time, it is possible that machines will not be able to solve the proof of work task before going offline. In this case, after waiting for a sufficient period of time, the bot functioning as the server simply drops the connection. It is possible that machines could get infected that are never online long enough to solve a single proof-of-work task, and therefore can never connect to the botnet. However, given their limited online time, such machines would be of little use to the botnet.

While the addition of the proof-of-work puzzle in the connection process will slow the rate at which bots are able to form connections, it has significant impacts on attempting to perform the eclipse attack on the botnet. As noted by Singh et al. [12], nodes mounting an eclipse attack must have a higher number of connections than a regular node. This means that in order to be successful, sybils must solve more proof-of-work tasks than regular bots.

This turns these proof-of-work tasks into a significant hurdle, as typically sybils are created by simulating huge numbers of fake peers on a single machine. The addition of these proof-of-work tasks means that each of these sybils will require the same amount of computing power as a stand-alone machine, if not more. This will impose a huge limitation on the rate at which sybils can form connections, unless a prohibitively expensive amount of computational resources can be acquired to run the sybils on.

## IV. Simulation Design

As mentioned in Section II, it was decided that we would synthetically simulate the botnet, on a framework of our own creation. This would allow the simulation to place the emphasis on the botnet defending itself against attack, rather than simulating the attacks produced by botnets. This section details the design of the simulation framework, and how it was used in the experiments.

### A. Framework

The focus of this simulation framework was to evaluate how the use of the reputation and proof-of-work systems in a botnet impacted the effectiveness of the sybil attack. All of this behaviour takes place at the application level of the TCP/IP model, making the simulation of any lower-level aspects superfluous. By abstracting out only the functionality of the botnet relevant to the outcome of the experiment, the process of creating and simulating the botnet was able to be simplified.

Since only the application layer was relevant to the experiments, the method in which packets passed between bots was arbitrary, allowing bots to be modeled as objects under the object-orientated programming paradigm. These objects would recreate a bot's behaviour by passing messages to other bot objects, and react accordingly to receiving such messages. As such, the simulation was effectively event driven, with bots only acting as a result of the actions of others, similarly to how the bots in a real version of this botnet would behave.

To coordinate the passing of messages between bots, a queue to hold these messages was implemented. The use of this queue would then allow the simulation to be run by simply attempting to empty the queue. This would involve emptying a message from the queue, passing it to the relevant bot, allowing the bot to act, which could possibly involve queuing a new message, and continuing this process until the queue was empty. By using a queue, it ensured that all bots received an equal chance to act, rather than have the simulation's execution gravitate around a small subset of the bots.

Since the botnet's design caused connections to change only as a result of bots coming online and going offline, this was an important factor that required simulation. Population statistics from the Torpig botnet showed that the number of bots online at any one time followed a diurnal pattern, with the maximum and minimum occurring regularly at specific times of the day [15]. This pattern was caused by the physical machines the bots had infected turning on and off, in accordance with when people in different parts of the world were awake. Using this information, as well as other statistics extracted from Torpig such as the number of new machines infected and infected machines disinfected over time, allowed the simulation to have a realistic botnet population.

One aspect of the population for which no data was available was the proportion of bots that are permanently online. This is a particularly important consideration in the simulated botnet, as connections only change as a result of machines turning on and off. This means that if two machines that are permanently online connect to each other, they will never disconnect. As no information on what percentage of machines vulnerable to botnets were permanently online was available, this value was assumed to be 1% of the botnet population, however the effect of this assumption was then later examined in more detail in Section V-B.

However, for this population information to be included, the simulation must incorporate a temporal aspect such that the population can change as a function of the time of day. Thus far in the simulation, the only factor which takes a known period of time is the time taken to solve a proof-of-work task. For this reason, the act of emptying the queue once was considered equivalent to the period of time taken to solve a proof-of-work task, and other factors such as the number of connections a bot could form in this period of time scaled to suit. Given the lack of information available for what number of connections a bot would form in a specific period of time, the study of an early P2P botnet by Grizzard et al. [16] was used as a baseline for this value. For further information on parameters please consult the full work [17].

As was discussed in Section III-A, it is in the best interests of the sybils to maximize their exposure to the botnet. For this reason, the sybils that were simulated acted as one collective entity, utilising a shared set of knowledge to coordinate their actions accordingly. This also facilitated the monitoring and consequent limiting of the resources available to the sybils, such that it could be varied during the experiments.

Although the botnet implemented in the simulation framework was the hypothetical perfect botnet described by Hund et al. [2], the framework was implemented such that other P2P botnets could also be simulated. This would allow the resilience of botnets to various attacks to be examined, rather than the attack capabilities of the botnet as is common for other simulation environments. Given the prototypal nature of this framework however, it was developed in a high-level language (Ruby) for rapid development. Preparing the framework for widespread use would require a reimplementation of the simulation in a more efficient language, to facilitate better simulation performance.

### B. Methodology

To evaluate the effects the reputation and proof-of-work systems had on the resilience of the botnet to the sybil attack, three distinct sets of experiments were performed, all of which followed the same basic methodology. This involved the simulation of four different botnets; a control botnet which featured neither the reputation or proof-of-work systems but only the basic P2P protocol, two botnets employing only one of the systems on top of the P2P protocol, and a botnet employing both systems on top of the P2P protocol. Comparing the results of the experiments on these different botnets allowed the effect of the reputation and proof-of-work systems to be determined.

Each of these botnets were randomly generated with the same random seed, and allowed to run for three days of simulation time without the presence of any sybils. This was to allow the botnet to stabilize and represent a well established botnet. Although only an hour of simulation time was enough to have the botnet operate at full capacity, three days were used to allow the peer lists of the bots to stabilize. The major reason for this is that different bots are online at different parts of the day, meaning that a few days were required for bots to successfully establish reputations with each other. Throughout this period, bot master commands were sent at regular intervals to allow reputation building to occur.

After the botnet had been successfully established, the sybil nodes were switched on, in order to degrade the botnet's performance as much as possible within one simulation day. The performance metric of the botnet was the *command penetration*, which was measured as the percentage of bots that received each of the bot master's commands. Since the bots must receive commands in order to perform their malicious actions, this was regarded as an accurate measure of the effect the sybils were having on the botnet.

The sybil attack was deemed a success if it could reduce the command penetration of the botnet to below 10%, at which point the botnet was considered disabled for the purposes of our experiments. If the botnet was disabled, the simulation would move onto testing the next botnet. If this level of command penetration was not reached, the amount of resources allocated to the sybil would be increased and the simulation run again. This process would then be repeated multiple times with different random seeds, to minimise the effects of the randomisation present in the botnet protocol. In all experiments the number of sybils was set to 10% of the botnet's total population, as this is what Davis et al. [8] found to be the most efficient ratio of sybils to bots.

## V. Results

Three different sets of experiments were performed, each focusing on a different aspect of the botnet. The first experiment analysed the effect the use of the different techniques had on the botnet's resilience to the sybil attack. In the second experiment, the impact of how many bots were permanently online was examined. For the last experiment, the relationship between the proof of work solution time and the resilience to sybils was examined.

### A. *Effect of Reputation and Proof-of-Work Systems*

Figure 3 shows the effect the use of the reputation and proof-of-work systems had on the resources required for the sybils to successfully disable the botnet. The Y axis represents the command penetration of the botnet, that is what percentage of the online bots received the bot master's command. The X axis indicates the amount of resources consumed by the sybils in order to achieve this command penetration, represented as a percentage of the botnet's total resources. This means that for a botnet of 1000 bots, the sybils consuming 10% of the botnet's resources would mean they used the bandwidth and computational resources equivalent to 100 bots in the botnet. Measuring the sybil resources in terms of the bot resources was chosen as while no information on previous sybil attacks was available, the amount of resources available to each bot within the botnet was known. Additionally, since early results indicated that the amount of resources required to disable the botnet increased linearly with size, this method also allows the results to be applied to a botnet of any size.

When the proof-of-work system is not used, the limiting resource requirement for the sybils is the amount of available bandwidth. This is because simulating the numerous sybils required takes an insignificant amount of computational resources, and only the bandwidth limits the rate at which sybils form connections. However, when the proof-of-work scheme is employed, the limiting factor is the amount of computational power available to the sybils, as the introduction of the proof-of-work task limits the rate at which connections are formed.

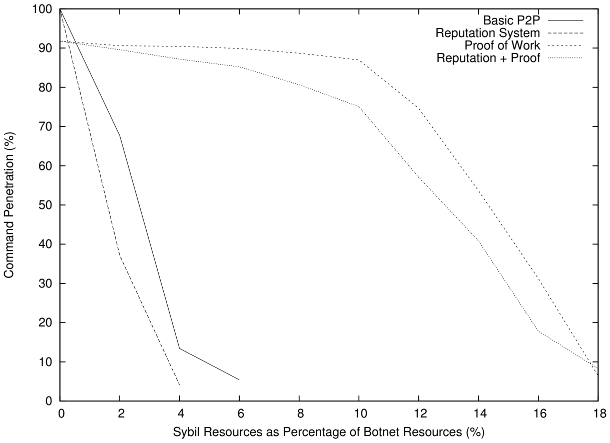

Fig. 3. Command Penetration 24 hours after Sybil Activation

As can been seen from Figure 3, the introduction of the proof-of-work system had the most significant impact on the botnet's resilience to the sybil attack. The reputation system on the other hand appears to make the botnet more vulnerable to the sybil attack. This is the case when comparing the reputation system to the basic P2P protocol, and when comparing the reputation and proof-of-work system combined to only the proof-of-work system.

However, the proof-of-work system not only impacted on the sybils' ability to form connections, but legitimate bots as well. Without the presence of sybils, the use of the proof-of-work scheme alone lowered the command penetration of the botnet by approximately 10%. This can be explained by the fact that for bots within the botnet, it now takes 5 minutes to form each connection rather than being able to form them almost instantaneously. Consequently, each bot takes significantly longer to make a sufficient number of connections to the botnet after it first comes online. With this less efficient use of a bot master's resources however comes with significantly higher resilience to the sybil attack.

### B. *Effect of the Percentage of Permanently Online Peers*

Due to the way in which the P2P protocol is designed, if two bots that are permanently online connect to each other, they will never disconnect. Should a sufficient amount of such machines connect together, there is the possibility of these machines forming a command distribution backbone for the network. Due to the fact that these machines will always be online, and should always be forwarding bot master commands, the reputation system should prioritize connections to these peers, making the botnet more resilient to sybils.

Figure 4 shows however that this is in fact not the case.

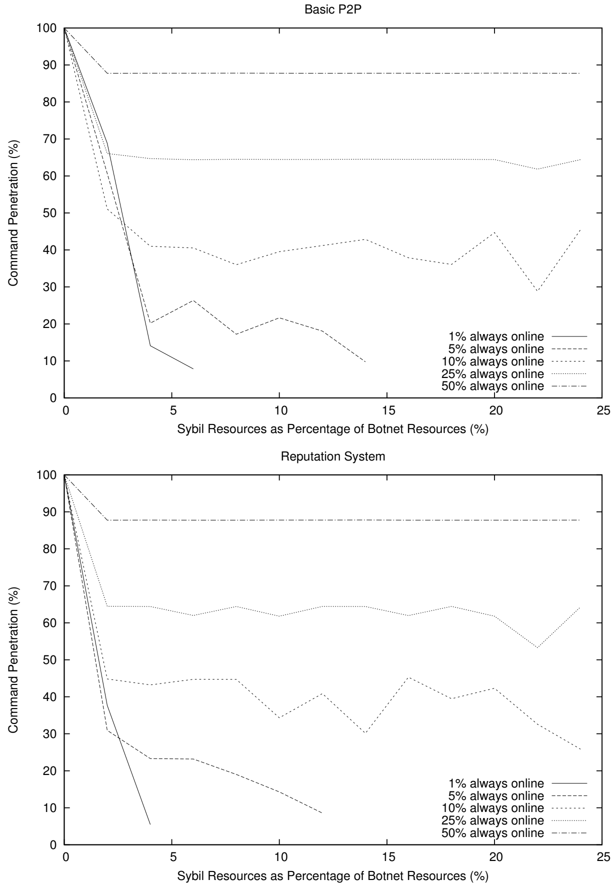

Fig. 4. Effect of the Percentage of Permanently Online Machines on Command Penetration

By comparing the effect the percentage of permanently online machines has on both the basic P2P botnet, and the botnet employing only the reputation system, it can be seen that the reputation system's command penetration is consistently lower when 1% and 5% of the machines are permanently online, and takes less resources to disable. As the percentage of permanently online peers increases, the reputation system's negative impact on the command penetration decreases, to the point where it appears to have no effect when 50% of the peers are permanently online.

These results suggest that there is a significant flaw in the design of the reputation system used in these simulations. The use of the reputation system increases the chances that a bot will reconnect to a bot it has already connected to before, and prioritizes those with high reputations, rather than the very low reputations the lack of command forwarding would have earned the sybils. However, should these trusted nodes be unavailable, possibly because sybils are occupying their connections, the use of the reputation system makes bots more likely to connect to sybils they have connected to before, even if their reputation is low.

Another possible factor is that attempting to connect to the same nodes every time the machine turns on could cause a bot to learn about less peers than if no reputation system was used, which would increase the chances of connecting to sybils once they are known. Additionally, the information from the reputation system is only used in the selection of peers to initiate connections with, meaning that although peers which fall below the threshold reputation are no longer chosen from the reputation list to connect to, current or incoming connections from these peers are not proactively blocked. The fact that these peers are also not removed from the regular peer list also means that there is still the possibility that these same peers will be chosen to connect to again.

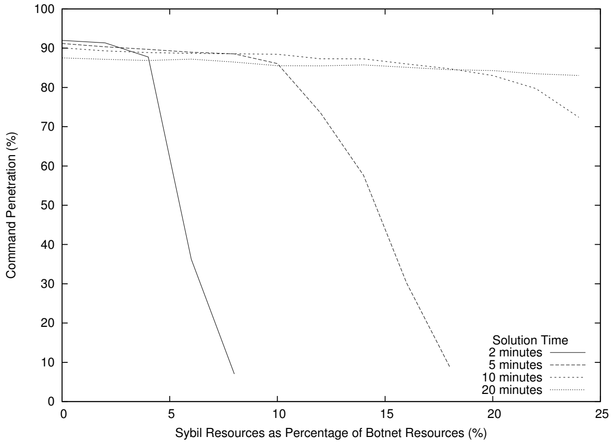

Fig. 5. Effect of the Proof-of-Work Solution Time on Command Penetration

While the actual cause of the reputation system's poor performance was unable to be determined, these ideas speculate on some of the possible causes. Regardless of the reason however, the reputation system still makes the botnet more vulnerable to the sybil attack than no reputation system at all. Therefore, the reputation system is in fact a technique unlikely to be implemented by bot masters as was first anticipated.

### C. *Effect of Proof-of-Work Solution Time*

Figure 5 shows the effect that variations in the proof of work solution time had on the command penetration of the botnet. Intuitively, increasing the proof-of-work solution time also increased the amount of resources required for sybils to disable the botnet. Since the sybils rely on rapidly forming a large number of connections to the botnet, increasing the solution time means that the sybils must increase their computational resources to be able to continue forming connections at the same rate.

While increasing the solution time increased the botnet's resilience to sybils, it also decreased the botnet's command penetration in their absence. This is due to the fact that increasing the solution time decreases the rate at which bots are able to form connection, causing them to require more time to form a sufficient number of connections to the botnet. Although at a solution time of 20 minutes the command penetration of the botnet was still approximately 90% without sybils, it can be seen that increasing this time too far would result in the botnet being unable to function, particularly if a large portion of machines were not online long enough to solve the proof of work tasks.

As a result, the value of the proof-of-work solution time is essentially a trade-off for the bot master. Increasing the solution time results in an improved resilience to the sybil attack, at the cost of less efficient use of the bot master's resources. To the bot master, the best value for this parameter therefore relies on how long they believe that increasing the solution time will extend the lifespan of the botnet. With the solution time set at a reasonable value, the use of the proof-of-work system could significantly increase the amount of time the botnet is able to operate before being disabled.

## VI. Conclusions

While previous works have identified reputation and proof-of-work systems as candidates for techniques botnets will employ in the near future, it would seem that only the proof-of-work system is up to the task. Overcoming the use of the reputation system in the simulated botnet required less resources than not using a reputation system at all, effectively making the implementation of such a system pointless. This however is only indicative of the reputation system utilized in this botnet, as perhaps a different method would be more successful.

The proof-of-work system however proved to be much more successful in defending the botnet against the sybil attack. By implementing such a system, it introduced a computational resource requirement in addition to a bandwidth requirement for participation in the botnet. For sybils, this meant that an immense amount of computational resources were required, a resource much more expensive to obtain than bandwidth, to be able to disable the botnet successfully.

As a result, the use of such proof-of-work systems in botnets is cause for serious concern. While overcoming the reputation system is quite possible, overcoming the proof-of-work system is much more infeasible. Should bot masters actually begin to implement such systems, it would have significant impacts on the capability of the security community to disable botnets.

### A. Future work

While the simulation framework used proved to be suitable for these experiments, the inefficiency of the implementation language became a problem at larger botnet populations, limiting the number of experiments that were able to be performed. While a simulation on botnets of 1000 bots would complete in seconds, 10000 bots would take hours, and 20000 bots more than a day. Additionally, the simulation framework was single threaded, meaning it was not able to take advantage of the multiple cores that have become commonplace on computer hardware. A more efficient, perhaps even multithreaded implementation in a different programming language, would not only allow the simulation of these larger botnets, but also allow experiments to be performed much more rapidly.

Further work on the reputation system is required to determine the exact cause for its poor performance. While in theory it should have prevented the sybils from being able to overrun the botnet, in practice this was not the case. Perhaps removing the regular list entirely, or actively avoiding peers with bad reputations could provide the tweak necessary for the system to work as intended.

The proof-of-work system provides the most avenue for future research. Should a botnet actually implement such a system, searching for implementation flaws would be the first priority. Other possibilities however include intentionally sending weak proof-of-work tasks, or attempting to hold all of a bot's possible connections without ever actually solving such a task.

## References

[1] J. Douceur, "The sybil attack," *Peer-to-Peer Systems*, pp. 251–260, 2002.

[2] R. Hund, M. Hamann, and T. Holz, "Towards Next-Generation Botnets," in *Proceedings of the European Conference on Computer Network Defense, December 11-12, 2008, Dublin, Ireland*. IEEE Computer Society, 2008, pp. 33–40.

[3] N. Ianelli and A. Hackworth, "Botnets as a Vehicle for Online Crime," *The International Journal of Forensic Computer Science*, vol. 2, pp. 19–39, 2007.

[4] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. C. Freiling, "Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm," in *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 15, 2008, San Francisco, California, USA*. USENIX Association, 2008.

[5] F. Leder, T. Werner, and P. Martini, "Proactive Botnet Countermeasures–An Offensive Approach," *Cooperative Cyber Defence Centre of Excellence Tallinn, Estonia*, 2009.

[6] F. Leder and T. Werner, "Know Your Enemy: Containing Conficker," *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[7] L. Borup, "Peer-to-peer botnets: A case study on Waledac," Master's thesis, Technical University of Denmark, 2009.

[8] C. Davis, J. Fernandez, S. Neville, and B. Victoria, "Optimising Sybil Attacks against P2P-based Botnets," in *Proceedings of the 4th International Conference on Malicious and Unwanted Software, October 13-14, 2009, Montreal, Quebec, Canada*. MALWARE, 2009.

[9] P. Wang, L. Wu, B. Aslam, and C. Zou, "A Systematic Study on Peer-to-Peer Botnets," in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, August 36, 2009, San Francisco, California, USA*. ICCCN, 2009.

[10] P. Barford and M. Blodgett, "Toward Botnet Mesocosms," in *Proceedings of the First Workshop on Hot Topics in Understanding Botnets, April 10, 2007, Cambridge, Massachusetts, USA*. USENIX Association, 2007.

[11] M. Niazi and A. Hussain, "Agent based Tools for Modeling and Simulation of Self-Organization in Peer-to-Peer, Ad-Hoc and other Complex Networks," *IEEE Communications Magazine*, vol. 47, no. 3, pp. 163–173, 2009.

[12] A. Singh, T. Ngan, P. Druschel, and D. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *IEEE INFOCOM*, 2006, pp. 1–12.

[13] S. Marti and H. Garcia-Molina, "Taxonomy of trust: Categorizing P2P reputation systems," *Computer Networks*, vol. 50, no. 4, pp. 472–484, 2006.

[14] S. Mansfield-Devine, "Hacking the hackers," *Computer Fraud & Security*, vol. 2009, no. 6, pp. 10–13, 2009.

[15] 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 16th ACM Conference on Computer and Communications Security, 2009, Chicago, Illinois, USA*. ACM, 2009.

[16] J. Grizzard, V. Sharma, C. Nunnery, B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study," in *Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets*. USENIX Association, 2007.

[17] A. White, "An Evaluation of Current and Future Botnet Defences," Honours Thesis, Queensland University of Technology, 2010, Available http://eprints.qut.edu.au/32595/.