Abstract: A botnet is a collection of computers controlled by a botmaster, often used for malicious activity. Social media provides an ideal platform for controlling a botnet, and also an avenue for botnets to spread their reach. In this research, we develop a botnet, SocioBot, that uses Twitter for its command and control (C&C) system. We conduct a variety of simulations based on this botnet. Epidemic models are used to validate and analyse our botnet simulations.

Keywords: malware; epidemic models; Twitter; stochastic simulation.


Biographical notes: Ismeet Kaur Makkar is a Security Engineer working at a leading software company. She completed her Bachelor of Technology in Computer Science and Engineering in India and Master of Science in Computer Science at San Jose State University with interest in information security. Her work in security focuses on web application security and operational security.

Fabio Di Troia is currently a Lecturer in the Department of Computer Science at San Jose University. His research interests are focused on malware detection, cryptography, and information security.

Corrado Aaron Visaggio is an Assistant Professor of Software Security of the MSc in Computer Engineering at the University of Sannio, Italy. He obtained the PhD in Computer Engineering at University of Sannio (Italy). His research interests include empirical software engineering, software security, and data privacy. He is author of more than 70 papers published in international journals, international and national conference proceedings, and books. He serves on many program committees and editorial boards of international conferences and journals. He is the responsible of the CINI CyberSecurity Lab at University of Sannio.

Thomas H. Austin is an Assistant Professor at San Jose State University, where he teaches classes in information security and programming language paradigms. He earned his PhD in Computer Science from UC Santa Cruz. He has previously worked with Mozilla’s research group, ESIEA Ouest’s Cryptology and Operational Virology lab, and CloudFlare, Inc. His interests are in web security, language-based security, and malware analysis.
Mark Stamp can neither confirm nor deny that he spent seven years as a National Security Agency cryptanalyst. However, he can confirm that he spent two years as Chief Cryptologic Scientist at a small Silicon Valley startup. He is currently a Professor of Computer Science at San Jose State University, where he teaches courses in information security and supervises large numbers of student projects. He has published numerous research papers in security and a popular textbook, Information Security: Principles and Practice, 2nd edition (Wiley 2011), and he recently completed a textbook on machine learning.

1 Introduction

Devices that connect to the internet are often not well protected against malware. Such vulnerable devices can be compromised and used in various attacks, including distributed denial of service (DDoS) and spamming. In one such incident, WordPress was attacked by a large botnet (http://www.zdnet.com/article/wordpress-hit-by-massive-botnet-worse-to-come-experts-warn/). In another recent case, Bitcoin users were advised to change passwords and encrypt their Bitcoin wallets after a successful attack by the Pony botnet (http://www.modernreaders.com/novel-version-of-pony-botnet-attacks-bitcoin-users/1885/lorenzo-tanos).

A bot can be defined as an application program installed on a user’s machine and executing automated tasks (http://us.norton.com/cybercrime-bots/). A botnet is a collection of bot-infected computers controlled by a so-called botmaster. As indicated above, botnets have been implicated in many attacks.

Users collaborate and post information on social networks in the form of multimedia and text. A portion of these posts come from users who try to exploit social network connections for malicious activities (Social News, 2012; O’Malley, n.d.). Thus, social media would seem to provide an ideal means for botnets to spread their reach.

As part of this research, we developed a botnet, SocioBot, that uses Twitter for its command and control (C&C) system. This work represents an extensive enhancement of a research botnet that was previously considered in Singh et al. (2013). The focus of this paper is simulation results based on our SocioBot botnet. We employ epidemic models to validate and analyse our simulation results.

The remainder of this paper is organised as follows. Section 2 provides background information on social networks and botnets, and we also discuss various epidemic models that are relevant to the work presented in this paper. Then in Section 3, we discuss our simulations and provide our results. Section 4 gives our conclusions and suggestions for future work.

2 Background

2.1 Social networks

Social media has become the dominant method of communication on the internet, with more than 70% of internet users relying on social networking (Bullas, 2014; Social Info, n.d.). Due to social media, we have more access – and more timely access – to news and opinions than ever before. It has become common to receive breaking news concerning events anywhere in the world via Twitter. People also share images with each other on Instagram, Flickr, and other photo sharing sites, and people use LinkedIn for professional networking and job searching. Facebook has more than a billion user accounts, while Twitter has more than 500 million users.

Twitter is among the most popular and fastest growing social networking sites (Bullas, 2014). Users on Twitter can post messages, known as ‘tweets’, that are currently limited to 140 characters. And users can follow other users – tweets from anyone a user is following will appear on the user’s home page. Twitter is different from many social networks in that the relationship between two users need not be mutual, that is, if user A follows user B, then user B has no obligation to follow A.

Twitter is a tempting target for the aspiring malware writer. Due to its large number of users, Twitter provides a vast number of potential targets. Also, with Twitter it is relatively easy to target groups of users who share common interests or views. By doing so, an attacker may be able to develop an online relationship with a group of users, or exploit existing relationships among users. Such relationships could be used as part of an attack, or as a means to further spread malware, such as a botnet.

2.2 Botnets

A botnet can be viewed as an application that can connect a large number of computers to a botmaster who controls the bots (). The bots can then execute commands on behalf of the botmaster.

Bots often spread across the internet by actively searching for vulnerable computers to infect. When a bot finds a susceptible computer, it can attempt to infect the machine and then report back to the botmaster. Unlike some other forms of malware, a bot will try to stay hidden until it is instructed to carry out a task (Barford and Yegneswaran, 2006).

After a computer is taken over by a bot, it can be used to carry out a variety of automated tasks. Some of the tasks may be relatively harmless, such as sending spam emails or posting to social networks. On the other hand, a bot might be used to perform malicious activities such as sending spyware, trojan, or virus files (Singh et al., 2013). A bot might also be used to steal information and communicate it to the botmaster. This stolen information could consist of credit card numbers, bank credentials, and other sensitive financial or personal information. Botnets can also be employed for launching DDoS attacks against a specified target. Cybercriminals can
The widespread use of social networks makes such networks an ideal medium to spread a botnet, and as a platform for malicious activities.

Several proof of concept social network bots have been developed. For example, botnets that use Twitter for C&C are constructed and analysed in The Mac Security Blog (http://www.intego.com/mac-security-blog/flashback-mac-malware-uses-twitter-as-command-and-control-center) and Nazario (2009), while Athanasopoulos et al. (2008) uses Facebook to spread a botnet.

2.2.2 Spreading the bots

According to Barroso (2007), botnet infections are spread primarily through browser exploits, malicious email attachments, and operating system vulnerabilities. A powerful infection method consists of sending a malicious link to a victim through email, or embedding a link in social networking sites. The malicious link redirects the victim to a malicious website that exploits a vulnerability in the browser to install a bot.

2.2.3 Trigger event

A trigger event is something that causes the bot to become active and exhibit its malicious functionality. The trigger could be the time of day, or a particular date at which time a spamming or DDoS attack might begin. Another trigger mechanism can be some user action, such as opening a banking site or financial software, which could, for example, activate keylogging software.

2.2.4 Functionality

A botnet’s functionality is the set of activities it has been programmed to perform when so commanded by the botmaster. Botnet functionality could include spamming, stealing financial data, encrypting the system drive, and DDoS attacks, among many other possibilities. A botnet might also be programmed to include functionality like click fraud or spamdexing, where bots post spam messages to blogs and other sites that allow commenting. Real-world examples include Agobot and Spybot (Barford and Yegneswaran, 2006) both have scanning capabilities that can control victim machines and perform DDoS attacks, while Storm (Barroso, 2007) is mainly used for spreading spam.

2.3 Network simulators

Unfortunately, we cannot perform large-scale botnet experiments on an actual social network, so in this research we rely on simulation. Network simulators are software applications that model the behaviour of a network. There are several network simulators that might serve as the basis for a social media-based botnet simulation. Next, we discuss two such simulators.

2.3.1 Spamulator

The Spamulator (Aycock, 2008) was developed as a teaching tool for a course on spam and spyware. It is a lightweight...
network simulator that runs on a single machine, and only simulates the parts of the internet that are relevant for sending spam. The simulator implements a limited set of features that include a network routing daemon for TCP and DNS on simulated servers.

The Spamulator effectively simulates the internet over millions of domains, it works with simple internet applications, it can simulate heavy use, and it can be extended for use in other areas of research. The Spamulator is a loopback network simulator, all components of which run on a single computer.

The Spamulator works as follows. Network packets that originate from a client program and are destined to a simulated server are redirected to a locally maintained queue. These packets on the queue are read by the Spamulator’s core, the loopback network simulator (LNS), which reroutes the packet to the local simulated server. Traffic from the simulated server that is destined for the client program is handled in a similar manner. The Spamulator does not touch network traffic that is not destined for a simulated IP address.

In addition to rerouting packets, the Spamulator searches for and launches simulated servers. It also keeps track of all the open connections, it obtains domain name information from a local simulated DNS, it forwards all packets between a server and a client, and it performs cleanup when a server finishes and is no longer needed. The network interface card (NIC) on the system must be available and active because the Spamulator uses the NIC to send and receive packets.

2.3.2 Docker

Docker is a container based virtualisation framework that is fast, lightweight, and easy to use, which puts it in stark contrast to many traditional virtualisation frameworks (https://www.docker.com/). A Docker container can hold all of the dependencies for an application, and every container is independent of – and therefore insulated from – all other containers. Docker has powerful APIs, which make it a lightweight framework with the capability to deploy applications into its containers.

Docker provides a means to execute and test applications securely within detached containers. This security and execution isolation permits the user to run multiple containers concurrently on a host. Docker also includes a software-as-a-service platform that is used to manage Docker containers. The Docker architecture is illustrated in Figure 1.

A Docker daemon is responsible for the work of making, running, and distributing containers. The user interacts with the Docker daemon through the Docker client, that is, the Docker client is the user-facing interface. The Docker daemon and client transmits messages by using sockets or RESTful APIs (https://www.docker.com/).

Docker has three internal major parts, namely, Docker containers, Docker images, and Docker registries. Next, we briefly discuss each of these parts. Then, we turn our attention to the use of Docker in our simulations.

Figure 1  Docker architecture (see online version for colours)

Source: https://www.docker.com/

Docker containers

Docker containers resemble a directory containing all the components needed for an application to run and function properly. This includes operating system, metadata, and user files. Docker containers can be executed, initiated, terminated, moved, or deleted. Every container is a quarantined and secure application platform.

Docker images

Templates, known as Docker images, are used to create Docker containers. For example, a template might include the Ubuntu operating system, with Apache, and various other user applications. A user can build new Docker images or update images that have already been created.

Docker registries

Docker registries are folders that hold images – these folders can be private or public. The images in public storage registries are searchable and can be accessed and downloaded by other users, whereas images in private storage are excluded from search results.

2.3.3 SocioBot simulations using docker

In Section 3.3, we present results for a Docker-based simulations of SocioBot. In these botnet simulations, the Docker client commands the Docker daemon to initiate a container for each user. The container simulates an independent machine infected with the bot application. This approach allows for a precise simulation of SocioBot. We can concurrently run more than 1000 instances of Docker container on the host machine.

2.4 Epidemic models

Mathematical models of infectious diseases are used to study the ways that diseases spread and to devise strategies to control outbreaks (Daley and Gani, 1999; Diekmann and Heesterbeek, 2000). Researchers have applied such mathematical models
in the fields of network security (Ajelli et al., 2010; Faghani and Saidi, 2009) and to model the propagation of malicious software over the internet (Garetto et al., 2003; Internet Role, n.d.; Yan et al., 2011). Here, we apply such models to analyse the spread of a botnet via Twitter.

There are two broad categories of epidemic models, namely, stochastic and deterministic models. Stochastic is a synonym for random and, not surprisingly, stochastic models use randomisation in one or more input values as a way to model probabilistic effects. Stochastic models can be formulated using Markov chains, or stochastic differential equations, for example. Deterministic models, such as those discussed in the next paragraph, are sometimes used as the basis for formulating stochastic models (Allen, 2008).

Deterministic models are based on mathematical representations, where each variable changes according to a given mathematical formula, not due to random variations. They are also called compartmental mathematical models because every entity in the population is put into a different compartment or subgroup. Each compartment represents a specific stage of the disease, including susceptible, infectious, and recovered, which are denoted as \(S\), \(I\), and \(R\), respectively (Daley and Gani, 1999). Individuals in the population can transition from one compartment to another over time. Deterministic models are specified in terms of differential or difference equations.

Table 1 gives the notation commonly used with epidemic models. Some of these terms are discussed in more detail below.

2.4.1 SIR model

The SIR model assumes a fixed population with three compartments (Kermack and McKendric, 1927). In this model, \(S(t)\) represents the number of individuals in the population at time \(t\) who are susceptible to infection, \(I(t)\) is the number of individuals at time \(t\) who are infected and can spread the disease to individuals in the susceptible subgroup, while \(R(t)\) represents the individuals who were infected and have recovered or are otherwise ‘removed’ from the simulation (e.g., due to death from the disease, or acquired immunity). The group \(R(t)\) is usually known as the recovered group, which is the term we use. In many cases, individuals in the recovered group cannot be infected again, that is, recovery imparts immunity from subsequent infection. The transitions in the SIR model are illustrated in Figure 2.

![Figure 2 SIR model (see online version for colours)](image)

In the SIR model, we have \(N = S(t) + I(t) + R(t)\), where \(N\) is the total population in the simulation. Using the notation in Table 1 and Figure 2, the SIR model can be specified by the system of differential equations

\[
\begin{align*}
\frac{dS}{dt} &= -\frac{\beta SI}{N} \\
\frac{dI}{dt} &= \frac{\beta SI}{N} - \gamma I \\
\frac{dR}{dt} &= \gamma I.
\end{align*}
\]

There are multiple variants of the basic SIR model. For example, in the SIS model, individuals become susceptible again after recovering from the disease (Allen, 2008). Next, we discuss the SEIR model, which is another variant of the SIR model. We will consider the SEIR model again in Section 3.

2.4.2 SEIR model

In the SIR model, we assume that a person is infectious immediately after infection. This is not always the case – there are many diseases (e.g., SARS and Ebola) where an individual is not infectious for an extended period of time after being infected. This time-lag between infection and becoming infectious is known as the latent period, and is referred to as the exposed state \(E\) in the SEIR model (Li and Fang, 2009).

In the SEIR model we have \(N = S(t) + E(t) + I(t) + R(t)\), where \(N\), \(S(t)\), \(I(t)\), and \(R(t)\) are the same as in the SIR model, while \(E(t)\) is the number of individuals at time \(t\) who have been exposed but are not yet infectious. This model is illustrated in Figure 3.

![Figure 3 SEIR model (see online version for colours)](image)

The SEIR model is specified by the system of differential equations

\[
\begin{align*}
\frac{dS}{dt} &= -\frac{\beta SI}{N} \\
\frac{dE}{dt} &= \frac{\beta SI}{N} - \epsilon E \\
\frac{dI}{dt} &= \frac{\beta SI}{N} - \gamma I \\
\frac{dR}{dt} &= \gamma I.
\end{align*}
\]
\[
\frac{dI}{dt} = \frac{\epsilon E}{N} - \gamma I \\
\frac{dR}{dt} = \gamma I.
\]

2.4.3 Basic reproduction number

The basic reproduction number, which is denoted as \( R_0 \), is defined as the number of secondary infections generated, on average, by a single infectious individual. From the definitions in Table 1, it is apparent that

\[
R_0 = \frac{\beta N}{\gamma}.
\]  

The number \( R_0 \) tells us whether a disease will cause an epidemic, or whether it will die out without intervention (Heffernan et al., 2005). Specifically, if \( R_0 < 1 \), then individuals recover from the disease at a faster rate than new infections occur, causing the disease to die out. Furthermore, the smaller \( R_0 \), the faster the disease dies out. On the other hand, if \( R_0 > 1 \), then the transmission rate is greater than the recovery rate, and hence the infection will become a self-sustaining epidemic in the population. Also, the larger the value of \( R_0 \), the faster the epidemic will spread.

3 Experiments

In this section, we present our Twitter-based botnet, SocioBot, which is an enhanced version of the research botnet in Singh et al. (2013). Then we analyse the spread of such a botnet over Twitter, using a Docker-based simulation. We compare the results from the Docker simulations to simple stochastic simulations based on epidemic models. These latter simulations serve to validate the Docker-based results.

3.1 SocioBot

In SocioBot, we use Twitter for the C&C of the botnet. Since we are using Twitter, we do not have to create a dedicated IRC server or any other network overlay control structure.

Each individual bot is a Java application. We assume that the application has been installed on each infected host. One possible mechanism for the spread of the bots will be discussed in the next section – for now, we simply assume that the hosts have already been infected.

The bot application monitors the botmaster Twitter account for any commands and performs operations based on the commands. The commands are encoded so that they appear to be innocuous tweets. We use the Twitter4j Java library to access the Twitter APIs (http://twitter4j.org/en/index.html).

We created a twitter4j.properties file which contains all the configuration information for the bot application. This includes parameters such as consumer key, secret key, access token, user name, and the request interval. This information can be updated at any time without affecting the operation of the botnet. The update process is achieved by uploading a copy of a new configuration file to Dropbox (https://www.dropbox.com/). The bot application monitors the Dropbox link for any changes to the properties file. Whenever the properties file changes, the latest copy of the properties file is downloaded and used for subsequent operations.

Twitter limits the number of API requests from an application to 200 per hour (https://dev.twitter.com/overview/documentation). Consequently, our application polls the botmaster Twitter account every four hours for any new tweets. If there are multiple tweets, each tweet is analysed in chronological order. The bot application is added to the startup programs folder so that the bot launches every time the system restarts.

For the sake of brevity, we omit additional details on the specifics of our implementation of SocioBot. For additional details, see the paper (Singh et al., 2013), which considers a similar, although somewhat less sophisticated, botnet.

3.2 Spreading the infection

In this section, we briefly consider one plausible scenario by which bots could infect computers via Twitter. Of course, many other approaches could be used.

Suppose that the botmaster uploads the bot software onto a server and creates a tiny URL, using a service such as Bitly (https://bitly.com/), which effectively disguises the actual URL. The botmaster posts this tiny URL as a tweet on his twitter account. This would not be indicative of an attack, since such shortened URLs are often used in tweets to reduce the number of letters in a message.

Users who click on the shortened URL will have the bot software downloaded and installed (assuming the system does not block it), thereby spreading the bot infection. Detailed examples of such drive-by download attacks – which by some accounts are the most common method for spreading malware today – can be found in Daswani (2010).

Once the bot has been installed on a victim machine, it can then receive commands from the botmaster. When a bot receives the appropriate command from the botmaster, it can use the Twitter account of the infected host to further spread the infection by using the same drive-by download attack discussed above.

Below, we use two different methods to simulate the spreading of the bot infection. First, we present results based on simulations using Docker. Then we consider stochastic simulations, that is, simulations based on generating random numbers under the constraints of standard epidemic models. These latter simulations serve to validate important aspects of the more realistic Docker simulation. In all simulations, we use the SIR or SEIR model to update the populations (susceptible, infectious, etc.). We present results to demonstrate the relationship between the average number of followers, and the probability that users click on the malicious link.

3.3 Docker-based simulations

In this section, we analyse the spread of SocioBot over Twitter, by combine epidemic models with our Docker simulation. For the all experiments in the section, we assume a fixed population of \( N \) Twitter users.
In reality, the number of infected users would depend on a variety of factors, including the density of followers of the infected users, time of day, geographic location, and the reach of the infection into the follower circle (Faghani and Saidi, 2009). For our models, we simplify somewhat so that only the density of followers and the ‘reach’ (i.e., the probability that a follower clicks on the malicious link) are considered. The standard epidemic models discussed above can then be applied. Specifically, we consider two different variants, which correspond to the SIR and SEIR epidemic models.

3.3.1 SIR model

Recall that in the SIR model, we assume that the infected nodes recover after the infection and cannot be infected again. This would correspond to a vulnerability that was exploited to install the bot and is now patched, which is analogous to the natural immunisation obtained after contracting a disease. In this model, \( S(t) \), \( I(t) \), and \( R(t) \) denote the number of susceptible, infected, and recovered nodes, respectively, at time \( t \). Every member of the population belongs to exactly one of these groups at any given time and hence \( N = S(t) + I(t) + R(t) \) for all \( t \), where \( N \) is the total population of Twitter users.

The infection rate \( \beta \) denotes the probability that an infected user spreads the infection. Such an infection only happens if a follower clicks on the re-tweeted short-URL. Thus, only the followers of an infected user can be infected using this model. However, not all infected users spread the infection at the same rate. The number of infections depends on the user’s connectivity (i.e., the number of followers the user has) and the probability that the retweeted link is clicked.

We experimented with different values for the parameter \( \beta \) and the probability of followers clicking on the retweeted link, which we denote as \( p \). Note that \( p \) represents the probability that any additional defense (e.g., AV software) fails to stop an infection, assuming the user clicks on the malicious link. In each such experiment, we calculate the fraction of users infected at each time step. For simplicity we assume that all users have the same number of followers.

The graph in Figure 4 illustrates the results for different values of \( p \), with \( \beta \) fixed. In this graph, the \( x \) axis represents the time step, while the \( y \) axis represents the fraction of infected users. As the probability of infection increases, the infection spreads faster. That is, as \( p \) increases, the curve moves to the left, as expected.

Note that in Figure 4, the precise meaning of the ‘time step’ is not specified. That is, the time step represents a unit of time, but we have not specified whether it is an hour or a day, or some other unit. For the analysis considered in this paper, we are primarily interested in the qualitative behaviour of the botnet, so the precise time units are not critical. But, for the range of parameters we have chosen, it would be reasonable to consider the time unit to be one day.

Figure 5 shows the spread of the bots over the period of time with a constant population of 1000 users, and with the number of followers of each user fixed at 10. Here, we assume a simplified SI model, where the infected users do not subsequently recover. As the number of infectious users increases, the susceptible user group decreases accordingly, due to the fixed population size. The assumption here is that the botmaster is tweeting continuously, i.e., one tweet per time unit, which is re-tweeted by the bots. This is analogous to a situation where a worm spreads unimpeded on a network.

![Figure 4](image-url) Fraction of users infected (see online version for colours)

![Figure 5](image-url) Infection without recovery (SI model) (see online version for colours)

Figure 6 shows results for the SI model for various values of \( p \). A low probability of infection transmission means it takes longer time for users to be infected and thus the epidemic is slower. For example, we see that all users are infected at time \( t = 4 \) in case of a transmission probability of 1, as compared to a time of \( t = 7 \) in case of a transmission probability of 0.25.

Next we change the experiment to accommodate recovery in the model, that is we use the full SIR model. We perform an experiment with the click probability set to \( p = 0.5 \). Also, we set the rate of recovery to \( \gamma = 0.5 \), which means that 50% of the infected nodes recover from infection at the next time step. Figure 7 shows the results of this simulation. At this recovery rate, the infection eventually dies out, at which point all users are recovered.
that is, the probability that a user clicks on the malicious link that causes the infection to spread. We denote the average number of followers as $n$ and the click probability as $p$.

**Figure 8** SEIR model ($p = 0.5$ and $\gamma = 0.5$) (see online version for colours)

3.4.1 **SEIR model**

We start with a single infected user who in turn will infect some of his followers. The number of followers for each user is determined by selecting at random (uniformly) from 10 to 100.

We compute a random click probability from 0.1 to 0.6, biased slightly towards 0.1.

Figure 9 shows the SEIR model for the spread of a botnet using the parameters in the previous paragraph, assuming an incubation period of one time unit, i.e., a new bot attempts to infect its followers at the next time step. As expected, the infectious plot follows the exposed plot with a delay of one time step, due to the incubation period of one time unit.

**Figure 9** Stochastic simulation (SEIR model) (see online version for colours)

We repeated this experiment with the number of followers fixed at 100 and the click probability set to 0.5; the results are given in Figure 10. The peak number of infected users is higher in this case as compared to Figure 9, since the average number of followers is higher.

Figure 11 shows the result of the analogous experiment with the number of followers fixed at 50 and the click probability set to 0.5; the results are given in Figure 10. The peak number of infected users is higher in this case as compared to Figure 9, since the average number of followers is higher.

**Figure 11** shows the result of the analogous experiment with the number of followers fixed at 50 and the click probability set to 0.5; the results are given in Figure 10. The peak number of infected users is higher in this case as compared to Figure 9, since the average number of followers is higher.

3.3.2 **SEIR model**

In the SEIR model, the infected nodes of the SIR model are partitioned into two sets, namely, exposed nodes (those that have the infection but cannot yet spread it) and infected nodes (those that are capable of spreading the infection to other nodes). The exposed class corresponds to bots that have been installed on user machines, but are not yet performing any malicious action.

For the SEIR model, we conducted an experiment with a total population of 1000 users with $p = 0.5$ and $\gamma = 0.5$. The incubation period from exposed to infection is set to one time unit, which means that exposed users can start spreading the infection after one time step. We also assume that nodes recover after two time steps. These results appear in Figure 8.

3.4 **Stochastic simulations**

In this section, we consider simplified stochastic simulations, where we generate random numbers and move individual users between categories (susceptible, infectious, etc.), based on the relevant model parameters. These results are designed to validate the basic characteristics of the more detailed Docker simulation discussed in the previous section.

For the simulations in this section, we consider a population of 5,000,000 users. The simulation results depend on the average number of followers and the click probability,
probability set to 0.3. Since the average number of followers is smaller, the peak number of infected users is smaller, as compared to Figure 10. In addition, since the click probability is smaller, it takes longer to reach the peak number of infected users, as compared to the experiment in Figure 10.

Figure 10  Stochastic simulation \((n = 100\text{ and } p = 0.5)\) (see online version for colours)

![Figure 10](image)

Figure 11  Stochastic simulation \((n = 50\text{ and } p = 0.3)\) (see online version for colours)

![Figure 11](image)

Figure 12 provides a comparison of our Docker simulation with the stochastic simulation for an equivalent set of parameters. From these graphs, we see that the Docker simulation yields comparable result to the stochastic simulation, which confirms that the former provides a useful platform for testing more complex situations than the stochastic simulation can manage. Recall that the Docker simulation creates containers for each user, and these containers can have a rich structure, thereby simulating any aspect of a user’s account or behaviour that we want to model. In contrast, the stochastic simulation can only model interactions at this simplest level. By validating the Docker simulation with the stochastic simulation, we have confidence that tests involving more sophisticated versions of the Docker simulation will be valid. That is, the behaviour that we observe from a more advanced Docker simulation will be due to the more advanced features we are trying to test, rather than being caused by some inherent weakness in underlying structure of the Docker simulation.

3.4.2 Modelling the spread of a tweet

As an aside, we note that epidemic models can be used to analyse the spread of legitimate tweets. Such models could be useful, for example, to determine when (or whether) an given tweet will ‘go viral’, that is, when a tweet is re-tweeted by large numbers of users over a short time interval. For this situation, an SIR model would likely be the sufficient, where susceptible users are potential re-tweeters, infectious users are those who have received the tweet, and users are recovered once they have re-tweeted to some subset of their followers. Figure 13 shows results for two such experiments; we compare \(p = 0.3\) with \(p = 0.5\), and let \(n = 70\) in both cases. The results are amenable to a similar interpretation as the results discussed in the previous section.

Figure 13  Spread of tweets \((n = 70\text{ with } p = 0.3\text{ and } p = 0.5)\) (see online version for colours)

![Figure 13](image)

3.5 Simulations for \(R_0\)

As discussed in Section 2.4.3, the basic reproduction number \(R_0\) is crucial when analysing epidemic models. Recall that if \(R_0\) is greater than 1, then the infection will become epidemic, whereas in cases where \(R_0\) is less than 1, the infection will die out without becoming epidemic.

We calculated the basic reproductive number for all numbers of followers from \(n = 10\) to \(n = 100\) and for selected click probabilities from 0.2 to 0.6. Figure 14 shows the variation of \(R_0\) when the recovery rate was held fixed at 1.
The higher the average number of followers, the greater is $R_0$, and the same relationship holds for the probability to spread the infection.

**Figure 14** Basic reproductive number (see online version for colours)

We also calculated the basic reproductive number for different values of the recovery rate. Figure 15 shows the value of $R_0$ for two different values of the recovery rate. Of course, the longer it takes to recover from infection, the higher the basic reproduction number, all else being equal.

**Figure 15** Effect of recovery rate on $R_0$ (see online version for colours)

We performed experiments to estimate $R_0$ using our Docker simulation. That is, we ran simulations over a range of values for the number of followers and the click probability, and observed whether the number of infected users became epidemic or not. Specifically, the number of followers of a user varied from 5 to 15 with the click probability varying from 0.1 to 0.5.

Figure 16 shows the results for this experiment. The red area on the heat map indicates the values for which an epidemic occurs while the blue indicates non-epidemic parameter values.

We also calculated the values of the basic reproduction number using the mathematical formula for $R_0$ in equation (1). These results, which appear in Figure 17, are similar to those obtained from the Docker experiments, as given in Figure 16. It appears that the Docker simulation slightly underestimates the chance of an epidemic at lower click probabilities with lower number of followers. Regardless, the results are sufficiently close so as to provide additional evidence for the validity of our Docker simulation.

**Figure 16** Docker simulation for $R_0$ (see online version for colours)

**Figure 17** Direct calculation of $R_0$ (see online version for colours)

3.6 Defensive strategies

To this point, we have conducted several experiments to simulate various SocioBot scenarios. We now briefly consider defensive mechanisms that might reduce the effectiveness of the approach used in SocioBot.

The results presented above show that by reducing the click probability $p$, we can affect the spread of the botnet. Thus we could reduce the spread of SocioBot by, for example, educating users to avoid clicking on suspect links. While this is clear, the value of simulation is that it enables us to quantify the effect that could be achieved. That is, if we assume that increased education can reduce the probability $p$ of clicking on a link by some specified amount, then we can quantify the expected reduction in the severity of the SocioBot infection, thus enabling a rational cost-benefit analysis. For example, from Figure 16 we see that in case the average number of followers is 10, when $p = 0.2$ then the basic reproduction number $R_0$ is less than 1, and hence the infection will not become a self-sustaining threat. On the other hand, if the average number of followers is 10 and $p = 0.3$, then $R_0 > 1$, which implies that the infection will spread. Note that both of these statements are also true with respect to a direct calculation based on the model parameters, as given in Figure 17. These results indicate that realistic changes in user behaviour (as modelled by $p$) can have a large impact on botnet behaviour (as indicated by $R_0$).
4 Conclusion and future work

In this paper, we discussed our design for SocioBot, a botnet that uses Twitter for its C&C system, and could also use Twitter to directly spread its reach. We then analysed a SocioBot simulation that uses Docker to simulate users and can therefore model virtually any aspect of user behaviour. We compared a basic version of the Docker simulator to results obtained using a straightforward stochastic simulation based on epidemic models. These results show that the Docker simulator is consistent with standard epidemic models, with respect to modelling the relevant interaction dynamics.

Enhancements to our current Docker simulator can be used to study SocioBot and other botnets in a controlled environment. For example, suppose that we have a proposed strategy for testing for malicious URLs. We could implement this strategy in each Docker host, and then test it within the simulation framework presented in this paper. We could then precisely quantify the effectiveness of the strategy. Other IDS-like techniques could also be tested in a similar manner.

Another category of future work would be to simulate various defensive strategies for Twitter itself. For example, if Twitter chooses to limit the number of re-tweets during a time interval, we could quantify the effect on the spread of a botnet like SocioBot. Many other such defensive strategies could be tested. Clearly, Twitter already uses some defensive mechanisms, but they do not provide such information to the public. A challenging and ambitious project would be to attempt to reverse engineer Twitter’s defenses by comparing simulated results to those observed on Twitter.

References


**Websites**

Bitly, https://bitly.com/


Dropbox – Keep Your Files Safe, Synced, and Easy to Share, https://www.dropbox.com/


Twitter Dev Documentation, https://dev.twitter.com/overview/documentation
