# Stegobot: a covert social network botnet

Shishir Nagaraja[1], Amir Houmansadr[2], Vijit Kumar[1], Pragya Agarwal[1], and Nikita Borisov[2]

[1] Indraprastha Institute of Information Technology, New Delhi
[2] University of Illinois at Urbana-Champaign

**Abstract.** We propose Stegobot, a new generation botnet that communicates over probabilistically unobservable communication channels. It is designed to spread via social malware attacks and steal information from its victims. Unlike conventional botnets, Stegobot traffic does not introduce new communication endpoints between bots. Instead, it is based on a model of covert communication over a social-network overlay – bot to botmaster communication takes place along the edges of a social network. Further, bots use image steganography to hide the presence of communication within image sharing behavior of user interaction. We show that it is possible to design such a botnet even with a less than optimal routing mechanism such as restricted flooding. We analyzed a real-world dataset of image sharing between members of an online social network. Analysis of Stegobot's network throughput indicates that stealthy as it is, it is also functionally powerful – capable of channeling fair quantities of sensitive data from its victims to the botmaster at tens of megabytes every month

## 1 Introduction

Malware is an extremely serious threat to modern networks. In recent years, a new form of general-purpose malware known as <u>bots</u> has arisen. Bots are unique in that they collectively maintain communication structures across nodes to resiliently distribute commands and data through a <u>command and control</u> (C&C) channel. The ability to coordinate and upload new commands to bots gives the botnet owner vast power when performing criminal activities, including the ability to orchestrate surveillance attacks, perform DDoS extortion, sending spam for pay, and phishing. This problem has worsened to a point where modern botnets control hundreds of thousands of hosts and generate revenues of millions of dollars per year for their owners [7, 12].
The evolution of botnets has primarily been driven by botnet responses (based on the principle of "whatever-works". Early botnets followed a centralized architecture however the growing size of botnets led to scalability problems. Additionally, the development of mechanisms that detect centralized command-and-control servers further accelerated their demise [6, 13, 10]. This led to the development of a second generation of decentralized botnets. Examples are Storm and Conficker [28, 22, 23] that are significantly more scalable and robust to churn.

We believe that one of the main design challenges for future botnets will be covertness — the ability to evade discovery will be crucial to a botnet's survival as organizations step up defense efforts. While there are several covertness considerations involved, one of the more important ones is hiding communication traces. This is the topic of the present paper. We hope to initiate a study in the direction of defenses against covert botnets by designing one in the first place.

We discuss the design of a decentralized botnet based on a model of covert communication where the nodes of the network only communicate along the edges of a social network. This is made possible by recent advances in malware strategies. Social malware refers to the class of malware that propagate through the social network of its victims by hijacking social trust. Instances include targeted surveillance attacks on the Tibetan Movement [17] and the non-targeted attack by the Koobface [4] worm on a number of online social networks including Facebook [1].

By adopting such a communication model, a malicious network such as a botnet can make its traffic significantly more difficult to be differentiated from legitimate traffic solely on the basis of communication end-points. Additionally, to frustrate defense efforts based on traffic flow classification, we explore the use of covert channels based on information hiding techniques. What if criminals used steganographic information hiding techniques that exploit human social habits in designing botnets? Would it be possible to design such a botnet? Would it be weaker or stronger than existing botnets? These are some of the questions we hope to answer in this paper.

The rest of this paper is organized as follows: in the next subsection we provide an overview on JPEG steganography, which is essential in the design of the social botnet introduced in this paper, *Stegobot*. In Section 3 we describe the design and construction of various components. We evaluate the use of of image steganography in designing the command and control channel of Stegobot using a real world dataset in Section 4.1; and the routing mechanism in Section 4.2. This is followed by related work in Section 5 and conclusions in Section 6.

### 1.1 JPEG steganography

The practical steganography schemes are either based on heuristic methods or provide some provable security based on some specific model. One of the first practical steganography schemes for the JPEG images is the JSteg scheme [3, 24]. It is based on using the Least Significant Bit (LSB) components of the quantized DCT coefficients. More specifically, the message bits are simply replaced with the LSBs of the DCT coefficients of an image, considering some exclusions for preserving the image quality. The embedding path for the LSBs was originally sequential while the use of pseudo-random path was suggested in later implementations. Even with pseudo-random path the LSB steganography techniques are shown to be detectable through different kind of attacks [30, 31, 16]. More specifically, the generalized category attack of [15] is able to detect embedding rates as low as 0.05 bits per non-zero non-one coefficients.

The reason behind high detectability of the JSteg scheme is the artifacts it makes into the first order statistics of the DCT coefficients. This led the next generation of the JPEG steganography schemes, namely statistical restoration-based schemes, to consider preserving statistical behavior of the cover images [26]. The main idea is to divide the cover image into two disjoint parts, which one part is used to embed the message and the other part is used to make corrections in order to preserve the selected statistical behavior of the image. A related approach for preserving the statistical behavior is used in the Model Based Steganography [25], where some specific *model* is preserved for the DCT coefficients.

As an example of the heuristic steganography schemes we can mention the F5 scheme [29], which was developed to address the detectability of the LSB-based embedding schemes. By decreasing the absolute value of the coefficients by 1 and using some other tricks the F5 scheme avoids the obvious artifacts on different features of the image. To increase the embedding efficiency F5 uses a coding scheme, namely Matrix Embedding.

Another approach for steganography, recently attracting more attention, is the minimal distortion embedding [8, 14]. These schemes focus on increasing the embedding efficiency by decreasing the embedding distortion. Newman et al. in [20] propose JPEG-compatibility-steganalysis resistant method, which embeds the message into the spatial domain of the image before performing the JPEG compression. YASS [27] is a more recent similar scheme that has been shown to be undetectable with payload of approximately 0.05 bits per non-zero DCT coefficient.

The steganographic *capacity* of a JPEG image is the largest payload that can be undetectably embedded. Fridrich et al. show through experiment that the steganographic capacity of grayscale JPEG images with quality factor of 70 can be approximated to be 0.05 bits per non-zero AC DCT coefficient, on average [9].

## 2 Threat model

We assume the threat model of a global passive adversary. A botnet is a distributed network of compromised machines acting cooperatively. Therefore it is fair to assume that the defenders will also cooperate (ISPs and enterprises) and hence have a global view of communication traffic.

We also assume that infections are not detected. While Stegobot will not withstand hundred-percent deployment of an anti-virus patch targeting the bot malware, it can easily withstand random losses of bots arising (say) due to restoration from a clean disk image. This assumption is due to the fact that online social networks are often scale-free graphs [21]. In a seminal paper [5], Albert and Baraba'si showed that scale-free graphs are highly robust to the removal of randomly selected nodes. Indeed the real world social graph considered in this paper (see dataset description in section 4.2) has a power-law degree-distribution with a slope of $\gamma = 2.3$.

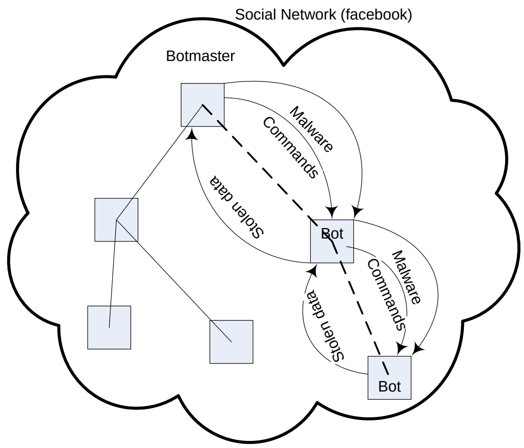



Fig. 1: The topology of the Stegobot botnet

## 3 Stegobot construction

A botnet is a distributed network of a number of infected computers. It is owned by a human controller called the **botherder** and consists of three essential components: the botmaster(s), the bots, and the Command and Control (C&C) channel. **Bots** are compromised machines running a piece of software that implement commands received from one or more **botmasters**; they also send **botcargo** – information acquired by the bot such as the result of executing botherder commands – to the botmaster. Botmasters refer to compromised machines that the botherder interacts with in order to send commands via a **C&C** channel. The botmaster sends instructions to the bots to carry out tasks and receives botcargo sent back to it by the bots.

### 3.1 Design goals

A distinguishing feature of Stegobot is the design of the communication channel between the bots and the botmaster. Stegobot is designed for stealth, therefore we do not wish to include any C&C communication links between computers that do not already communicate.

A further goal is to design probabilistically unobservable communication channels connecting the botmaster and the bots. If the C&C communication is unobservable then botnet detection can be significantly more difficult than where communication is not hidden. This is because in the latter case, traffic-flow signatures and the changes in the structure of traffic connectivity induced by the presence of the botnet can lead to easier detection and removal of the botnet [11, 18].

### 3.2 Malware propagation and bots

The first step in botnet creation is malware deployment. The malware is an executable which infects the machine and performs the activities necessary of a bot. Stegobot is designed to infect users connected to each other via social links such as an email communication network or an online social network that allows friends to exchange emails. The propagation of malware binaries takes place via social-malware attacks [17]. Social-malware attacks refer to the use of carefully written email lures to deliver botnet malware combined with the use of email communication networks to propagate malware. This works when the attackers take the trouble to write emails that appear to come from the co-workers or friends of the victim (social phish). A more effective attack is to replay a stolen email containing an attachment that was genuinely composed by a friend back to the victim after embedding a malicious payload within the attachment.

Once the attacker secures an initial foothold (deploy the malware on at least one victim's machine), the attacker can expand the list of compromised machines with little additional effort. Whenever one of the initial set of victims sends an email containing an attachment to one of their colleagues, the bot quickly embeds a malicious payload to the attachment. Upon opening the attachment, the receiver's computer also gets infected and the process continues. Therefore once a single user is compromised (and the compromised machine continues to be operated for sending emails), the attacker can recruit additional bots in an automated fashion. Indeed this was the modus operandi behind the Ghostnet surveillance attacks on both Google and the Tibetan administration in 2009 [17].

Of course the attacker's attempts at composing email lures can fail with non-zero probability. However this exercise needs to succeed only once (as explained in the previous paragraph ) to generate a botnet containing thousands of nodes, and the risk of failure is offset by targeting multiple people within a social group.

### 3.3 Bots

In Stegobot, bots possess a pre-programmed list of activities such as harvesting email addresses and passwords, or credit card numbers or simply to log all keystrokes. Alternatively, in a more flexible design the bots execute commands received from the botmaster. For instance, bots receive search keywords from the botmaster and respond with matches from the filesystem, as in the case of the Tibetan attacks [17].

As explained in the previous paragraph, Stegobot spreads along the social network of its victims. While we have used emails to explain social-malware attacks, the attacks are by no means restricted to email communication networks alone; online social networks are equally good targets. For instance, Koobface [4] is a worm the propagates on Facebook over social links. Further, it is noteworthy that Facebook is adding email extensions to its existing service; and Google added a social networking service — Google Buzz — to its popular email service in 2010. This allows bots to communicate with eachother and the botmaster over the social network as explained in the next section.

### 3.4 Message types

Stegobot uses two types of message constructions. First, **Bot-commands** are broadcast messages from the botmaster. Examples include search strings for file contents or within keylogged data.
Second, *botcargo* messages return information requested by the botmaster such as files matching search strings. Botcargo messages can be divided further into two types: locally generated (*botcargo-local*) or forwarded messages (*botcargo-fwd*) on a multi-hop route to the botmaster.

### 3.5 Communication channel

In Stegobot, we use the images shared by the social network users as a media for building up the C&C channel. Specifically, we use image steganography techniques to set up a communication channel within the social network, and use it as the botnet's C&C channel.
A bot running on a computer can communicate with a bot running on a different computer if both the computers are being used by people connected by an edge in the social network. The social network acts as a peer-to-peer overlay over which the information is transferred from each bot to the botmaster. In Stegobot, information between bots must only be transferred using steganographic channels. In our case, this channel is constructed by hiding the botcargo within images using standard techniques reviewed in earlier sections. By keeping the size of the botcargo within certain limits, it is possible to make the presence of bot communication difficult to discover by examining the communication channel alone (section 4.1).

*One-hop communication:* takes place according to a push-pull model. Consider the example of Facebook (see figure 2). When a user (**pushes**) uploads an image to Facebook from an infected host, the bot intercepts the image and inserts the botcargo into the image using an image steganography technique as previously discussed. Upon completion of image upload, all the neighbors of the user are notified (by Facebook). When a neighbor of the publisher logs into Facebook from an infected machine and views the picture, the bot (**pulls**) intercepts the image and removes the steganographically embedded botcargo from the image. All botcargo is finally destined for the botmaster who downloads the cargo by viewing newly posted pictures from her neighbors. When the botmaster intends to issue a command, she does so by preparing a botcargo message and uploading it to her Facebook account.

*Multi-hop communication:* In Stegobot, routing is based on a very simple algorithm: **restricted flooding**.

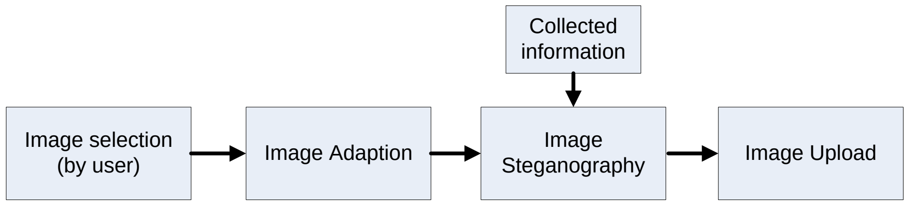



Fig. 2: Process of sending a one-hop message

*Congestion control:* Each bot maintains a bandwidth *throttle* which is used to control the ratio of *botcargo-local* to *botcargo-fwd* messages.

*Metrics:* We measure the effectiveness of the routing strategy using a set of metrics:

- Channel efficiency: the percentage of *botcargo-fwd* messages that reach the botmaster averaged over all bots.
- Channel bandwidth: similar to efficiency, but it is the absolute number of *botcargo-fwd* messages that reach the botmaster averaged over all bots.
- Duplication: the number of duplicate *botcargo-fwd* messages received by the botmaster.
- Botnet bandwidth: the total number of botcargo-fwd reaching the botmaster every month excluding duplicates.

## 4 Experiments

### 4.1 Steganography experiments

We use YASS [27] as the image steganography scheme of the C&C channel over the Facebook social network. As mentioned before, in order to reduce the effect of the unknown uploading process performed by Facebook social network the bot performs an image adaption process before performing the steganographic process. Based on our experiments, we perform two adaptations on every image: 1) each image is converted to the JPEG format, 2) images are resized to meet the maximum resolution limits performed by Facebook, i.e., $720 \times 720$ [1]. This is performed keeping the aspect ratio of the images.
We use a database of 116 different images to perform our experiments. In each experiment an image is adapted to Facebook constraints, as mentioned before, and then the hidden information is embedded into that image using YASS scheme. The stego image is then uploaded into Facebook through a Facebook user account, and then downloaded from the

[1] More recently, Facebook is allowing uploading of higher-resolution images that increase the steganographic capacity at least 10 times based on our preliminary experiments

Facebook using another Facebook account. Finally, the downloaded image is evaluated by the YASS detector described in [27] in order the extract the hidden message. To evaluate the robustness of our steganographic process we calculate the bit error rate (BER) metric which is defined as the ratio of error message bits to the total number of message bits for each image.
Table 1 summarizes the average of the BER parameter (over all of the images) for different metrics of YASS scheme. $Q$ is the quality factor of YASS scheme and represents the amount of compression performed by YASS during the steganography process. $Q$ has a range of $[0, 100]$ and directly impacts the quality of the stego image, i.e., higher $Q$ results in images with higher quality/size. Based on our experiments we can model the Facebook uploading process as a JPEG compression with a quality factor of $Q_f$. For $Q > Q_f$ Facebook applies extra compression on the image which results in loosing some of hidden information bits. On the other hand decreasing $Q$ results in lower number of bits being inserted by the YASS scheme. So, there should be an optimum value for $Q$ within the range of $[0, 100]$ which minimizes the BER rate, i.e., maximizes the robustness to Facebook perturbations. As Table 1 shows the BER values are minimized for a $Q = 75$, hence we approximate the quality factor of the Facebook compression to be $Q_f \approx 75$.
We also investigate the effect of the redundancy parameter of YASS, $q$, on the BER. The parameter $q$ represents the number of times an information bit is repeated inside an image by the YASS scheme. Intuitively, we expect that larger $q$ results in reducing the BER, since more redundant bits can help better in reconstructing a noisy message; this is confirmed through our experiments as Table 1 shows. In fact, the $q$ parameter makes a tradeoff between robustness and steganographic capacity: increasing $q$ improves robustness by reducing BER while it also reduces the number of data bits inserted by the YASS scheme. Table 2 shows the number of bits inserted by YASS for different values of $q$.
Our experiments show that a small number of image, namely bad images, are responsible for the most of the error in the average BER. Excluding these images in the steganography process can significantly improve the BER performance of in our experiments. We define and use a metric, SelfCorr, in order to decide whether an image is a "bad" or "good" image. The SelfCorr metric evaluates the cross correlation of an image by a noise-filtered version of itself. We declare images with $SelfCorr > 0.9964$ as "bad" images. Table 3 illustrates the BER results after excluding the small number of "bad" images determined by the SelfCorr metric. As can be seen, the average BER is significantly improved, e.g, the average BER is 0 for $q \geq 12$.

### 4.2 Routing results:

Combining social-malware with steganographic channels yields a covert botnet where new bots are recruited as infections spread along the edges of the social network, while existing bots communicate using the well understood image based steganographic channels. In this section, we study the routing capabilities of such a botnet using a real-world example.

Table 1: Average BER (over 116 images) without removing "bad images"

| q | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| Q=65 | 0.3073 | 0.1320 | 0.0520 | 0.0227 | 0.0097 | 0.0047 | 0.0022 | 0.0010 | 0.0006 | 0.0003 |
| Q=70 | 0.2966 | 0.1318 | 0.0529 | 0.0219 | 0.0096 | 0.0049 | 0.0025 | 0.0010 | 0.0005 | 0.0002 |
| Q=75 | 0.3015 | 0.1557 | 0.0680 | 0.0283 | 0.0101 | 0.0067 | 0.0027 | 0.0010 | 0.0004 | 0.0000 |
| Q=80 | 0.3086 | 0.1839 | 0.0846 | 0.0347 | 0.0143 | 0.0089 | 0.0034 | 0.0015 | 0.0008 | 0.0000 |
| Q=85 | 0.3512 | 0.2618 | 0.1777 | 0.0854 | 0.0372 | 0.0183 | 0.0127 | 0.0053 | 0.0024 | 0.0013 |
| Q=90 | 0.4287 | 0.3917 | 0.3639 | 0.3390 | 0.3146 | 0.2906 | 0.2567 | 0.2122 | 0.1591 | 0.1262 |

Table 2: Number of bits inserted in each image for different values of $q$

| q | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| Data bits | 40280 | 20140 | 13426 | 10070 | 8056 | 6173 | 5754 | 5035 | 4475 | 4028 |

*Dataset:* We chose to study the Flickr[2] social network [2], an online friendship network that facilitates image sharing. We crawled the Flickr website and downloaded on a fraction of the Flickr social network. Specifically, our dataset contains 7200 nodes (people), the social network edges (online friendship relations) between them, and the number of images posted per person per month. The dataset corresponds to user activity on Flickr over a period of 40 months. The Flickr dataset will be made available on our website for the research community.

In our simulation, each bot node generates $K$ *botcargo-local* (see section 3.4) messages per month. $K = 10$ corresponds to say ten files that the bot plans to route to the botmaster across the social overlay network. $ttl$ is fixed at $log(N = 7000) \approxeq 3$ hops. Each bot reserves a minimum of 5% of node bandwidth to forward *botcargo-fwd* messages received from neighbors. Further, we assume *bot-command* messages broadcast from the botmaster at a rate of one message per month. This means that the botmaster can instruct her bots to change operation no more than once a month.

Stegobot's infection strategy is based on social malware attacks. In our experiments, we have assumed an infection rate of 50%. While this num-

[2] Unfortunately, we did not have access to the Facebook topology or the upload patterns of users.

Table 3: Average BER after removing "bad images"

| q | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| Q=65 | 0.2945 | 0.1088 | 0.0311 | 0.0092 | 0.0022 | 0.0002 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Q=70 | 0.2836 | 0.1105 | 0.0340 | 0.0095 | 0.0016 | 0.0002 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Q=75 | 0.2892 | 0.1372 | 0.0492 | 0.0136 | 0.0011 | 0.0001 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Q=80 | 0.2977 | 0.1686 | 0.0662 | 0.0175 | 0.0020 | 0.0003 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Q=85 | 0.3436 | 0.2512 | 0.1631 | 0.0646 | 0.0165 | 0.0029 | 0.0012 | 0.0000 | 0.0000 | 0.0000 |
| Q=90 | 0.4255 | 0.3877 | 0.3589 | 0.3331 | 0.3074 | 0.2823 | 0.2464 | 0.1978 | 0.1396 | 0.1035 |

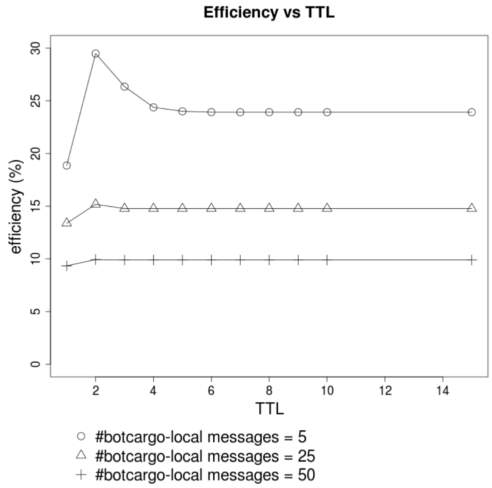



Fig. 3: Average channel efficiency against ttl

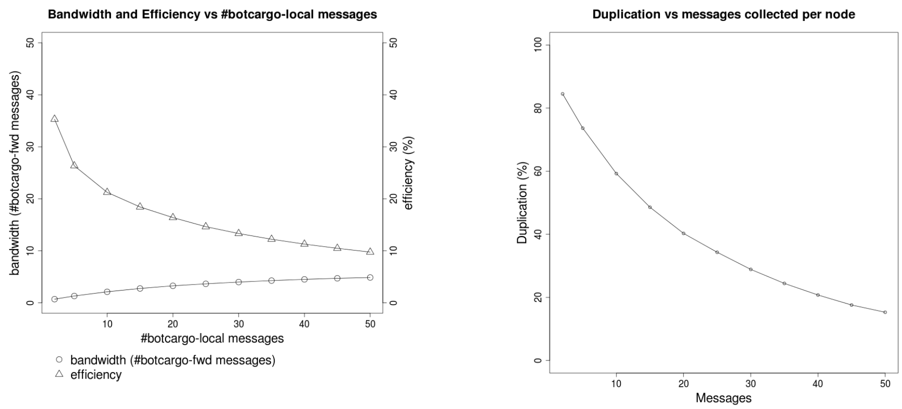
ber might appear high to some readers, it is actually a conservative estimate; social-malware has been known to have infection rates approaching 90-95% in real-world attacks [17].

*Botcargo preparation:* Each bot gathers botcargo (both from the host as well as from its neighbors). It then encodes as much of the botcargo in a single image as allowable according to a detection threshold $\ell$ bits. The practically possible values for the number of bits is given in table 2 and a discussion in section 4.1.

*Routing:* In Stegobot, routing is carried out by restricted flooding. Each bot publishes (floods) botcargo to all neighbors (joined the botnet) within *ttl* hops in the social network. Finally, the botmaster receives botcargo through the one of its infected neighbors. We assume that the botmaster is a randomly chosen node in the network. For each of the graphs below, we averaged the results over fifty different botmaster nodes.

Figure 3 shows the efficiency of botcargo transmission for increasing amounts of *ttl* and various numbers of *botcargo-local* messages. For $K = 5$ *botcargo-local* messages, the efficiency peaks at 30% and decreases and then stabilizes for higher *ttl* values as the resulting increase in the number of *botcargo-fwd* messages begins to cause congestion. Congestion effects are also felt when the number of *botcargo-local* messages increase even at a smaller *ttl*. This justifies our intuition for using $ttl = log(N)$ where $N$ is the number of infected nodes in the botnet.

In restricted flooding, high-degree nodes in the topology play the role of hubs and are able to pull and collect large amounts of botcargo. As such



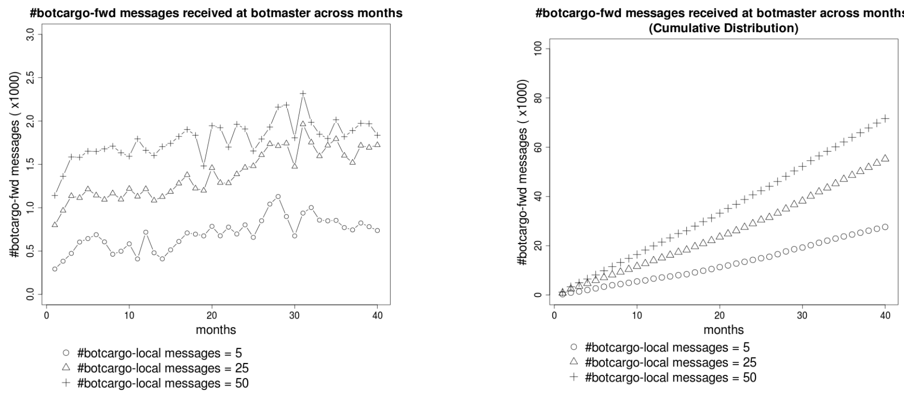

(a) Normalized Bandwidth and Efficiency

(b) Duplication

Fig. 4: Communication channel bandwidth and efficiency

they become a natural point where stolen information is collected and can then be siphoned off to the botmaster.

*Channel Bandwidth and Efficiency:* Figure 4 shows the bandwidth and efficiency of the communication channel in the average case. Figure 4.a shows the monthly average number of *botcargo-fwd* messages received by the botmaster (normalized by the size of the botnet) for various amounts of *botcargo-local* messages collected per bot (constant across bots). Figure 4.a also shows the average efficiency of the communication channel from a bot to the botmaster as the size of the botcargo changes. The network seems to operate at an average efficiency of 30% of collected botcargo reaching the botmaster when $K = 2$ (#botcargo per bot per month). This decreases with increase in $K$ although the absolute number of messages delivered at the botmaster increases marginally from .75 per bot for $K = 2$ to 2.5 per bot for $K = 10$. Further increases result in even more marginal increases as the effects of congestion result in decreasing routing efficiency. A positive effect of increasing per node botcargo collection sizes ($K$) is the reduction in duplicate messages reaching the botmaster. This is shown in figure 4.b, the proportion of duplicate messages rapidly decreases until $K = 10$ and further reduces to 40% at $K = 20$. We observe that the positive effects of duplication reduction correspond with an increase in normalized bandwidth as the number of *botcargo-local* messages collected per node increase.
The main result of our experiments is shown in figure 5. Figure 5.a shows the average number of botcargo messages delivered to the botmaster. This shows an increasing trend. This can be traced to the increasing number of users and the number of average number of photo updates per user increase over the months in our dataset. The sharp drops and




(a) Botcargo delivered (b) Cumulative botcargo delivered

Fig. 5

increases are related to routing performance under **churn**, when a few large uploaders suddenly stop using uploading for certain periods of time, or dormant users being uploading in larger numbers (say from one-two images to twelve-fifteen images per month). Figure 5.b indicates the cumulative amount of traffic received by the botmaster over the years and gives a sense of the total amount of sensitive material she can steal and the long-term trends. Combining the total number of messages reaching the botmaster (18000 *botcargo-fwd*) with the number of bits embedded in each message, we obtain a monthly bandwidth of between 21.60MB/month in the average case ($q = 8$) to 86.13MB ($q = 2$) for lower interference from the image adaption process.
Overall, it is easy to see that even with a simple and naive routing algorithm such as restrictive flooding, the botmaster is easily able to collect around 10% of the total amount of stolen information. With a slightly more sophisticated algorithm that exploits the presence of medium and high degree hub nodes as super-peers, one could design a better routing algorithm. For instance, in the current implementation all nodes behave the same way, hence hub nodes also locally flood all the botcargo they receive. This is replayed back and forth between hubs and the rest of the network causing severe congestion. By ensuring that super-peers carefully route incoming botcargo only to other super-peers, we believe it should be possible to significantly improve network throughput.

# 5 Related work

Early botnets followed a centralized architecture. However, the growing size of botnets led to scalability problems. Additionally, the development of mechanisms that detect centralized command-and-control servers [6, 13, 10, 11] has motivated the design of decentralized peer-to-peer botnets. Several recently discovered botnets, such as Storm and Conficker, have adopted the use of structured overlay networks [28, 22, 23].

These networks are a product of research into efficient communication structures and offer a number of benefits in robustness and scalability. Their lack of centralization means a botnet herder can join and control at any place, simplifying ability to evade discovery. The topologies themselves provide low delay any-to-any communication and low control overhead to maintain the structure. Further, structured overlay mechanisms are designed to remain robust in the face of churn, an important concern for botnets, where individual machines may be frequently disinfected or simply turned off for the night.

The work closest to ours is the work of Nappa et al. who describe a design of a resilient and stealth botnet using Skype [19]. By hijacking active (logged in) Skype sessions, the botnet is able to bypass firewalls that might otherwise prevent bots from directly communicating with eachother. Our design goes a lot further due to the unobervability properties of our communication channel. Unlike their design, we do not add new connection end-points – no communication between user-accounts (bots) that do not already communicate, and no additional communication is introduced beyond what that users already exchange, resulting in a stealthier design.

## 6 Conclusions

In this paper, we have presented and analyzed the design of a covert botnet that aims to collect sensitive information from victims. The proposed botnet deploys innovative social malware infection strategies to create an overlay network over the communication network of users. This significantly increases the robustness of the botnet as it gains by limiting the amount of damage from a fraction of compromised bot nodes. A critical aspect of our design is the use of image based steganographic techniques to hide bot communication and make it indistinguishable from image noise. While techniques for image steganography are well known, we go one step further to show that it is possible to design an effective covert network by exploiting the social network connecting users and the social habits of individual users.

The essence of communication traffic security lies not merely in protecting content but also unobservability. By exploiting the social network of victims to communicate, a botnet can bring covertness gains in both communication topology and traffic. First, the presence of bot communication does not introduce new communication end points hence yielding a covert topology that is robust to detection methods that use network topology to localize botnets; and second, combining this with image steganography gives (probabilistic) communication unobservability hence making the network robust to detection methods based on classifying traffic flows.

## References


1. Facebook. `http://www.facebook.com`.
2. Flickr. `http://www.flickr.com`.
3. JSteg. `http://zooid.org/~paul/crypto/jsteg/`.
4. Koobface. `http://en.wikipedia.org/wiki/Koobface`.
5. Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Error and attack tolerance of complex networks. Nature, 406(6794):378–382, July 2000.
6. James R. Binkley and Suresh Singh. An algorithm for anomaly-based botnet detection. In SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, pages 7–7, Berkeley, CA, USA, 2006. USENIX Association.
7. J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In ACM conference on Computer and communications security, pages 375–388, New York, NY, USA, 2007. ACM.
8. Jessica J. Fridrich, Miroslav Goljan, and David Soukal. Perturbed quantization steganography. Multimedia Syst, 11(2):98–107, 2005.
9. Jessica J. Fridrich, Tomás Pevný, and Jan Kodovský. Statistically undetectable jpeg steganography: dead ends challenges, and opportunities. In Deepa Kundur, Balakrishnan Prabhakaran, Jana Dittmann, and Jessica J. Fridrich, editors, Proceedings of the 9th workshop on Multimedia & Security, MM&Sec 2007, Dallas, Texas, USA, September 20-21, 2007, pages 3–14. ACM, 2007.
10. Jan Goebel and Thorsten Holz. Rishi: Identify bot contaminated hosts by IRC nickname evaluation. In HotBots, 2007.
11. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In Proceedings of the 17th USENIX Security Symposium (Security'08), 2008.
12. C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In CCS, October 2008.
13. Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In HotBots, 2007.
14. Younhee Kim, Zoran Duric, and Dana Richards. Modified matrix encoding technique for minimal distortion steganography. In Jan Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, Information Hiding, volume 4437 of Lecture Notes in Computer Science, pages 314–327. Springer, 2006.
15. K. Lee and A. Westfeld. Generalized category attack improving histogram-based attack on jpeg lsb embedding. In Information Hiding workshop, pages 11–13. Springer-Verlag, Lecture Notes in Computer Science, 2007.
16. K. Lee, A. Westfeld, and S. Lee. Category attack for lsb embedding of jpeg images. In International Workshop on Digital Watermarking, IWDW, volume 4283, pages 35–48. Springer-Verlag, Lecture Notes in Computer Science, 2006.

17. Shishir Nagaraja and Ross Anderson. The snooping dragon: social-malware surveillance of the tibetan movement. Technical Report UCAM-CL-TR-746, University of Cambridge, March 2009.
18. Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. Botgrep: finding p2p bots with structured graph analysis. In Proceedings of the 19th USENIX conference on Security, USENIX Security'10, pages 7–7, Berkeley, CA, USA, 2010. USENIX Association.
19. Antonio Nappa, Aristide Fattori, Marco Balduzzi, Matteo Dell'Amico, and Lorenzo Cavallaro. Take a Deep Breath: a Stealthy, Resilient and Cost-Effective Botnet Using Skype. In Proceedings of the $7^{th}$ Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), Bonn, Germany, Lecture Notes in Computer Science. Springer, July 2010.
20. Newman, Moskowitz, Chang, and Brahmadesam. A steganographic embedding undetectable by JPEG compatibility steganalysis. In IH: International Workshop on Information Hiding, 2002.
21. M. E. J. Newman. The structure and function of complex networks. SIAM Review, 45(2):167–256, 2003.
22. P. Porras, H. Saidi, and V. Yegneswaran. A multi-perspective analysis of the Storm (Peacomm) worm. In SRI Technical Report 10-01, 2007.
23. Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. A foray into Conficker's logic and rendezvous points. In 2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET '09), 2009.
24. Niels Provos and Peter Honeyman. Hide and seek: An introduction to steganography. In IEEE Security and Privacy, volume 1, pages 32–44, 2003.
25. P. Sallee. Model-based steganography. In IWDW, pages 154–167, 2003.
26. K. Solanki, K. Sullivan, U. Madhow, B.S. Manjunath, and S. Chandrasekaran. Provably secure steganography: Achieving zero k-l divergence using statistical restoration. In ICIP, 2006.
27. Kaushal Solanki, Anindya Sarkar, and B. S. Manjunath. YASS: Yet another steganographic scheme that resists blind steganalysis. In Teddy Furon, François Cayre, Gwenaël J. Doërr, and Patrick Bas, editors, Information Hiding, volume 4567 of Lecture Notes in Computer Science, pages 16–31. Springer, 2007.
28. Sam Stover, Dave Dittrich, John Hernandez, and Sven Dietrich. Analysis of the Storm and Nugache trojans: P2P is here. ;login, 32(6), December 2007.
29. Westfeld. F5–A steganographic algorithm: High capacity despite better steganalysis. In IH: International Workshop on Information Hiding, 2001.
30. A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In A. Pfitzmann, editor, 3rd International Workshop of Information Hiding, volume 1768, pages 61–75. Springer-Verlag, Lecture Notes in Computer Science, 2000.
31. X. Yu, Y. Wang, and T. Tan. On estimation of secret message length in jsteg-like steganography. In International Conference on Pattern Recognition, volume 4, pages 673–676, 2004.