# A SMS-Based Mobile Botnet Using Flooding Algorithm

Jingyu Hua [*] and Kouichi Sakurai

Department of Informatics, Kyushu University,
{huajingyu, sakurai}@itslab.csce.kyushu-u.ac.jp

**Abstract.** As a lot of sophisticated duties are being migrated to mobile phones, they are gradually becoming hot targets of hackers. Actually, during the past few years, It has appeared many malware targeting mobile phones and the situation is getting worse. Under this circumstance, we may ask a serious question: whether can those infected phones be organized to a botnet? In this paper, we present a design of such a botnet using Short Message Service (SMS) as its Command and Control (C&C) medium. We cover all the aspects of the botnet design including the stealthiness protection, the topology selecting and the botnet maintaining. Our simulations show that in our proposed SMS-based botnet a newly issued C&C message can be covertly propagated to over 90% of the total 20000 bots within 20 minutes based on a simple flooding algorithm. Moreover, in this process each bot sends no more than four SMS messages and the botnet is robust to both random and selective node failures. Thereby, we demonstrate that the proposed mobile botnet is indeed a serious threat on the security of the mobile computing environment. For this reason, we further explore several effective defense strategies against such a botnet. In doing so, we hope to be one step ahead of the hackers to discover and prevent this upcoming threat.

## 1 Introduction

### 1.1 Background and Motivation

During the past few years, a significant evolution has taken place in the field of smart phones: firstly, their computing power is growing rapidly: some smart phones like iPhone have already outperformed the early desktop. Secondly, as the popularization of 3G and near future's 4G, they are also getting closer to the desktop in the communication capability. For these reasons, more and more sophisticated applications like financial markets, online banking, etc. are being migrated to the smart phones from the traditional PCs. Then, smart phones are inevitably becoming the next hot targets of the hackers with the constant rising of their business values.

In fact, since 2004 many malware targeting mobile phones have already emerged. These worms or viruses can propagate and infect vulnerable smart

[*] supported by the governmental scholarship from China Scholarship Council

phones through all kinds of mediums including Internet [1, 2], Storage Cards [3], SMS [4], MMS [5], and even some local wireless protocols like Bluetooth [6, 7]. Based on this, people may ask a serious question: can those infected mobile phones be carefully organized into a botnet by the hackers as they did in the PC world? For this question, although by now there's no major outbreak of mobile botnets, most researchers believe that the answer is positive: mobile botnets will appear sooner or later [8–11] and it's just a matter of time.

If the mobile botnet is unavoidable, it becomes quite meaningful for us to investigate the potential technologies that can be used to construct them from the perspective of the hackers. By doing so, we can get our defense strategies ready before the real outbreaks of mobile botnets and avoid being left behind again by the hackers as we really did in the desktop battlefield. In this paper, we follow this motivation to evaluate whether the Short Message Service (SMS) can be used to construct an effective mobile botnet. We choose to study SMS because of two reasons: firstly, as a mandatory function SMS is supported nearly by all the existing phones. It is text-based and system-independent. So the hackers can utilize this service to propagate commands among heterogeneous platforms. Secondly, SMS is quite simple and reliable: all you need is a phone number, and you can immediately send the corresponding phone a message with a very small error rate. Therefore, SMS may provide the hackers an ideal C&C medium.

### 1.2 Related Works

In 2009, Traynor et al.[11] propose using a mobile botnet to launch a DDoS attack against the core infrastructure of the cellular network. Their simulation and analysis demonstrate that their attack can cause nation-wide outages with even a single-digit infection rate, which teaches us a good lesson about the astonishing destructive power of mobile botnets. However, their work does not discuss how to construct a mobile botnet in details, especially how to construct an efficient C&C medium.

The first detailed work in the mobile botnet construction is done by Kapil et al [12]. They investigate the possibility to construct and maintain a mobile botnet via Bluetooth. In their design, botnet commands are propagated via Bluetooth when those infected mobile phones move into each other's radio range. Through several large-scale simulations based on some publicly available Bluetooth traces, they demonstrate this malicious infrastructure is possible. However, since this botnet is highly relied on the human mobility, its real performance is hard to guarantee especially when the density of hijacked phones is not high: according to their simulations, a command can only reach 2/3 of the bots even after 24 hour in a botnet with 100 bots.

Zeng et al. [13] later propose another mobile botnet using SMS to implement a Kademlia-like P2P network as its C&C channel. However, because Kademlia is quite complex, too many SMS messages are required to send for a bot to locate and get a command. According to their simulations, even in a small botnet with 200 nodes, on average 20 messages are needed to send for a single command lookup. In addition, since the nodes in this botnet don't know when a new command will be issued, they have to probe continually, which also wastes a lot of messages. As we known, all the SMS messages are under the monitoring of telecom operators and they may also cost money. Sending too many abnormal messages will make the botnet prone to being detected both on the service provider side and the user side.

Recently, Mulliner et al.[23] also investigate several methods to construct a mobile botnet. They first introduce a SMS-only C&C which uses the tree as the underlying topology. This topology suffers an obvious drawback that when one node fails, all its subnodes are isolated from the botnet and can no longer receive any commands. As a result, the botmaster has to continually broadcast ping messages to locate the failed nodes and then repair the tree, which brings great side effects to the stealth and the feasibility. They then introduce another improved SMS-HTTP hybrid C&C, which first hangs command SMS messages on some website and then informs several random selected bots to download and send them. The weak point is that although those SMS messages are encrypted, they may still disclose the destination bots because the decryption keys are embedded in the URLs. In addition, those random selected nodes are also prone to being captured because they may send unusual high number of SMS messages if the botnet is very large.

### 1.3 Challenging Issues

Generally, the hackers have to overcome the following challenges to design an effective SMS-based mobile botnet:

(1) The proposed botnet should have an efficient C&C architecture, in which a command issued by the botmaster can reach most of the bots in a short time. What's more, for security reason each bot should only send a small number of SMS messages in this process.

(2) Because all the SMS messages are under the monitoring of the telecom operators, we need special measures to disguise the botnet messages as the legal ones to evade being filtered out.

(3) Once a botnet is constructed, we need special mechanisms to maintain it. This mainly involves two issues: Firstly, the botmaster usually wants to master the runtime statuses of their controlled bots. Thereby, besides a command propagation channel, we need another reporting channel from the bots to the botmaster. Secondly, the botmaster has the requirement to update the malware distributed on the bots regularly. The sizes of these updates usually greatly exceed the maximum payload of a single SMS message, so we need an extra updating mechanism.

### 1.4 Our Works and Contributions

In this study, we mainly focus on investigating the above challenges to develop a proof-of-concept SMS-based mobile botnet in advance of the hackers. In particular, we make the following contributions:

(1) We propose a proof-of-concept SMS-based mobile botnet. It uses SMS to propagate C&C messages based on a simple flooding algorithm. We discuss how to guarantee the stealth of this botnet both on the user side and the service provider side by combining the data-encryption and the data-hiding.

(2) We do simulations to evaluate the performance of our proposed botnet with different topologies. We find that in the Erdos-Renyi random graphs the propagation of commands can be very fast and robust to both random and selective node failures. In particular, a message can reach over 90% of the total 20000 bots within 20 minutes from any node. And each node sends no more than 4 messages in this process.

(3) We then present a method to construct the desired random graph topology for our botnet under the help of an internet server. We also introduce a mechanism to maintain the constructed botnet by associating our mobile botnet with a traditional PC botnet.

(4) Based on the above, we demonstrate that it's entirely possible to utilize SMS for Mobile Botnet Command and Control. Therefore, in the end of this paper we explore several potential defense strategies against this mobile botnet.

## 2 The Overview of the Proposed SMS-based Botnet

In this section, we present the overview of our proposed SMS-based mobile botnet with the sample in Fig. 1. For simplicity, we neglect technical details here and just show our basic idea.

In our proposed botnet, each compromised smart phone, namely bot is made to maintain a partial list of the other peers as its virtual neighbors. For instance, in the sample botnet in Fig.1, the bot $P1$ maintains a peer list $(A2, A6)$, where $A2$ and $A6$ are the phone numbers of $P2$ and $P6$, respectively. By doing so, $P1$ gets to know that $P2$ and $P6$ also belong to the botnet and it is enabled to directly send them SMS messages. Then, the botnet commands are propagated in this special architecture based on a modified flooding algorithm: The botmaster can first send his command to any node via SMS. And then, for each node, whenever it receives the new command, it is made to continually select uncommunicated neighbors as the destinations to forward this command via SMS until the forwarding count reaches a pre-defined upper bound.

Here, two things we have to point out. Firstly, the bots do not use the group messaging to forward their received commands. Instead, we make them forward a command to their neighbors one by one with random time intervals. We make this decision because the use of group messaging will provide the service provider extra detection signatures. What's more, with this mechanism the number of forwardings may be also reduced: while a bot is waiting for the next forwarding window, it may receive the same command from a bot it is planning to forward the command to. As a result, these redundant forwardings can be avoided. Secondly, since sending SMS messages may cost money, we have to limit the number of forwardings made by each bot. Otherwise, if too many messages are required to send to propagate one command, a bot will face a high risk of being noticed while its user receives an unexpected high bill. That's why in the above we introduce a upper bound for the forwarding times of each bot.

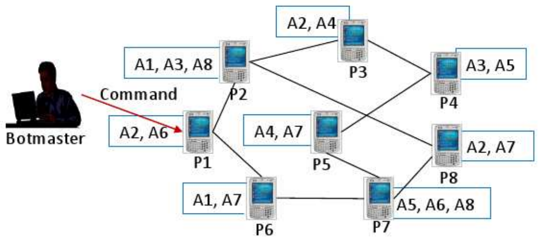


**Fig. 1.** An example of our proposed SMS-based mobile botnet

If this simple botnet is made true, we will get a complete P2P botnet: each node is equivalent and there's no centralized infrastructure. In addition, because each bot only knows a limited number of others, the resistance of this botnet to the bot capture is high: de-infection of one bot only discloses its neighbors, which brings limited effects to the whole botnet. Now, let's discuss the details of this SMS-based mobile botnet in the following sections.

## 3 Stealthiness Study

As we known, the long live of a botnet is relied on its stealthiness, i.e., the ability to work without being noticed by the defenders. Therefore, at first we discuss the stealthiness of our proposed SMS-based botnet. This mainly involves two aspects: the stealth on the user side and the stealth on the service provider side.

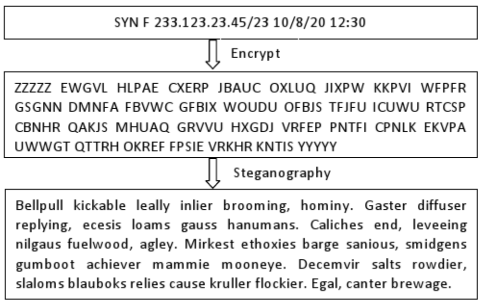
On the user side, the most important thing is to enable the bot code to send and receive SMS messages carrying botnet commands without being noticed by the users. Besides, the bot code itself as well as the related processes and resources should be also hidden from the users. This requires us to implement a rootkit on the compromised mobile OSs. This is hard but not impossible. Recently, Papathanasiou and Percoco [14] introduced a method to implement a Linux KLM-based rootkit on Google's Android platform. Based on their work, we implement a prototype rootkit that can hijack the system call table to cover our malicious behaviors including secret message sending and receiving. The only problem is that we haven't thought of any elegant ways to persist this rootkit to make it survive from the reboots. This is quite important because mobile phones subject to frequent reboots. In addition, as we said before, because SMS message may cost money each bot should only send a limited number of messages in the process of command propagation. Otherwise, they face a high risk of being perceived when their users have to pay an abnormally high bill. We will show in the next section that by carefully selecting a topology for our botnet a command can quickly reach over 90% of the total 20000 bots in 20 minutes with each bot sending no more than four messages, which is quite stealthy.

On the service side, since all the SMS messages go through the gateways of telecom operators, the stealth is much more difficult to guarantee. In the traditional networks, since different subnets are administrated by different organizations, it's difficult to uniformly deploy a defense system against some security events even if the system is proven effective. However, in the cellular network, the situation is totally different. All the communications in this network are usually monitored by only one telecom operator. It's easy for them to quickly popularize a defense system within the whole network. For this reason, the C&C messages of our mobile botnets should never be broadcasted in their original forms. Otherwise, once one message forwarded by a bot is recognized as a botnet command, it will be easy for the telecom operators to compose a signature to filter out all these messages and further detect all the infected mobile phones. Hence, more complicate technologies are needed to guarantee the stealth of our bontet on the server side.

Our solution is to encrypt the C&C messages before they are transmited. We make each pair of neighbors share a unique secret key. And then commands sent from one node to another are encrypted and decrypted with their unique key. By doing so, the same command forwarded by different nodes to different destinations will appear in different forms. As a result, even if a command is captured by the service provider, it's still impossible for him to create a uniform signature to filter out all the command messages sent by different bots. Hence the stealth is greatly improved. However, this solution suffers a critical drawback: after the encryption, the obtained SMSs become random texts that are greatly different from those normal messages. So we further utilize some text steganography algorithms[15, 16] to convert those ugly cipertexts to texts more closer to the natural language. Fig. 2 shows such an example. After being encrypted with the Rijndael cipher [17], the command "SYN F 233.123.23.45/23 10/8/20 12:30", which informs the bots to launch a SYN attack against 233.123.23.45/23 at 12:30 on August 20, 2010, will become unreadable random texts. However, if we further convert the ciphertext with the Stego! steganographic algorithm [16], we can obtain a stegotext that at first glance looks like a English text. Therefore, it will become more difficult for the service provider to determine the presence of a botnet command within a SMS message in a short time.

## 4 Topology Study Based on Simulation

As shown in Fig. 1 the proposed SMS-based botnet can be regarded as a graph $G = (V, E)$, where $V$ is the set of nodes and each node is corresponding to a bot; $E$ is the set of edges and for arbitrary two nodes $v_1, v_2 \in V$, they are linked $(\exists e_{ij} = (v_i, v_j) \wedge e_{ij} \in E)$ if and only if their phone numbers are contained in each other's neighbor list. Obviously, the efficiency of the command propagation is closely related to the topology of the botnet graph. Therefore, in this section, we construct different topologies to study their different performances based on



```
SYN F 233.123.23.45/23 10/8/20 12:30
```

```
ZZZZZ EWGVL HLPAE CXERP JBAUC OXLUQ JIXPW KKPVI WFPFR
GSGNN DMNFA FBVWC GFBIX WOUDU OFBJS TFJFU ICUWU RTCSP
CBNHR QAKJS MHUAQ GRVVU HXGDJ VRFEP PNTFI CPNLK EKVPA
UWWGT QTTRH OKREF FPSIE VRKHR KNTIS YYYYY
```

Bellpull kickable leally inlier brooming, hominy. Gaster diffuser replying, ecesis loams gauss hanumans. Caliches end, leveeing nilgaus fuelwood, agley. Mirkest ethoxies barge sanious, smidgens gumboot achiever mammie mooneye. Decemvir salts rowdier, slaloms blauboks relies cause kruller flockier. Egal, canter brewage.

**Fig. 2.** Steganography Example

some simulations. By doing so, we hope to find an efficient topology for our SMS-based mobile botnet.

## 4.1 Simulation Setup

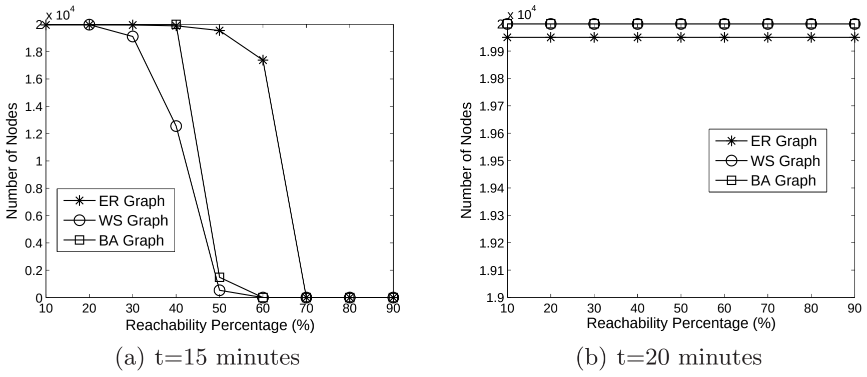
We mainly study three common topologies: Erdos-Renyi random graphs [18], Barabasi-Albert scale-free graphs [19] and Watts-Strogatz small world graphs [20]. We choose to study these three graphs because they are the most common complex networks in our world and also hot candidates in the design of traditional PC botnet [21].

All the topologies were implemented in the C programming language with the igraph C library [22]. We generated twenty graphs for each topology. And all their average node degrees are set to 6. We choose this value because it makes our simulated graphs become approximated connected graphs (The largest connected components covers more than 99% of the nodes). This is a necessary condition for a command to be eventually propagated to most of the nodes in the botnet. The simulations were driven by discrete events in seconds. When a node receives a command in an event, it will register new forwarding events in a central event queue one by one until the forwarding times reach a pre-set bound $b$. In each event, the node randomly selects an un-communicated neighbor as the target to forward this command. The time interval between two continuous forwarding events is a random value between $[60s, 120s]$. All the results for each topology presented in the following are the averages of the data obtained from the simulations on their own twenty graphs.

## 4.2 Simulation Results

We select three key measures to compare the effectiveness of the three topologies: Reachabilities from nodes, Influence of the forwarding Bound and Resistance to node failures. We present the details and the comparison results below:

**Reachabilities from nodes** : with our decentralized C&C infrastructure, the botmaster can issue his command to the botnet from any node within the botnet. Therefore, a key performance measure is the distribution of the reachabilities from different nodes in our botnet by an appointed time. Here, the reachability from a node at time $t$ refers to the percentage of nodes that a command can reach by $t$ from that node. In our study we use $I(r, t)$, the inverse cumulative distribution of the reachability to quantify this measure. It represents the number of nodes whose reachabilities at time $t$ are larger than $r$.



(a) t=15 minutes (b) t=20 minutes

**Fig. 3.** Inverse cumulative distributions of the reachabilities after 15 and 20 minutes

Fig. 3 presents the simulation results of $I(r, t)$ in the three topologies when $t = 15$ and 20 minutes. These simulations assume no forwarding bound and no node failure. We can find that the efficiency of our botnet is very high in this case: no matter which of the three topologies is selected, a command can reach over 90% of the total nodes in 20 minutes from almost any node. In particular, the ER random model outperforms the other two: nearly 100% of the nodes in the ER model can make a 50% reachability within 15 minutes, while in the WS graphs and BA graphs no more than 2000 nodes can achieve this. In our opinion, this is quite reasonable: firstly, compared with the ER graph, the nodes in the WS graph show a property of high local clustering. As a result, the command forwardings made by local neighbors in this topology are more prone to conflict, which greatly reduces the propagation speed of commands. Secondly, the node degrees in the BA graph follow a power law distribution which means that some nodes in this topology having much higher degrees compared with others. As a result, these nodes attract many redundant forwardings because they are connected by more nodes. Thus, the command propagation in this topology is also slowed down.

**Influence of the forwarding Bound** : as we said before, to guarantee the stealth of the botnet, we introduce an upper bound for the forwarding times of one command on each node. Although this may greatly affect the command

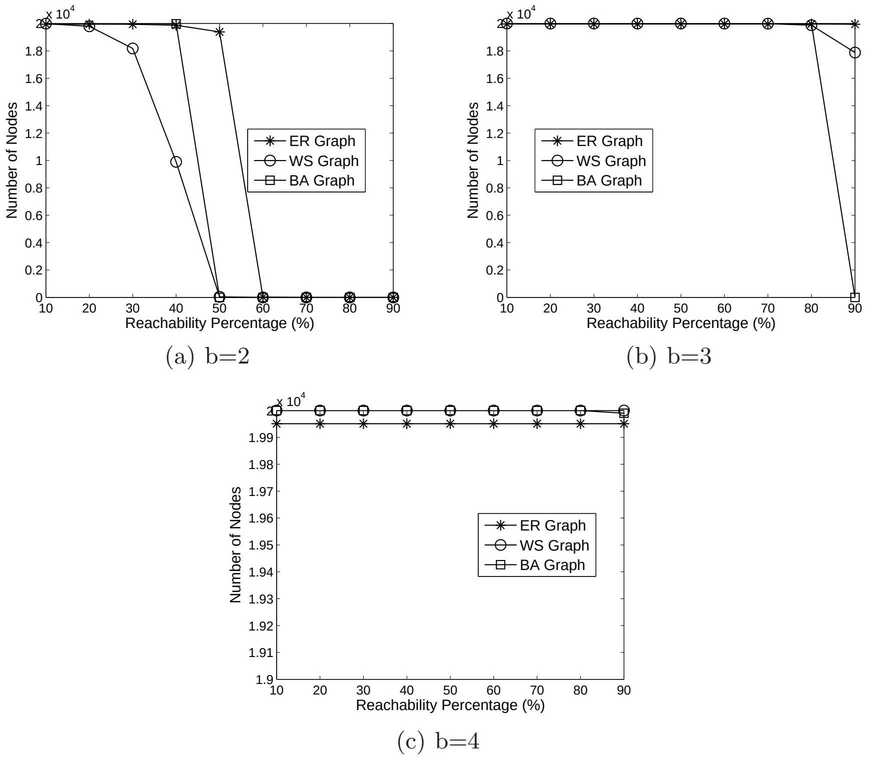


**Fig. 4.** Effects of the forwarding bound

propagation in our botnet, it is necessary at some time. For instance, in our first simulation without forwarding bound, we found some nodes in the BA model sent more than 20 messages for one command, which have been enough to cause the attention of their users. Hence the success of our botnet is based on the premise that commands can be well disseminated even if the forwarding times are constrained to a small value on individual bots. So we did more simulations to study the impacts of the forwarding bound in the three topologies. Fig. 4 presents the simulation results of $I(r,t)$ at the time of 20 minutes while the forwarding bound is set to 2, 3, 4, respectively. We can find that the ER random model is least affected by the forwarding bound: the reachabilities of most nodes in this topology can still exceed 90% after 20 minutes so long as the forwarding bound is no less than 3. For the other two topologies, although they are more affected, the effects of the forwarding bound can be also removed when $b \geqslant 4$. Since the cost of four SMS messages are too small to draw the attention of the users, we conclude that the stealth of our mobile botnet is high.

**Resistance to Node Failures** :by now, all our simulations make an assumption that all the nodes in the botnet are active and work correctly while the botmaster issues a command. However in reality, hijacked smart phones may be turned off by their users, be out of energy or even be disinfected and removed from the botnet forever. In all these cases, the bots will become inactive and lose the ability to receive and forward messages temporarily or perpetually. This is so called node failures. Obviously, the resistance to these node failures is critical to the success of a mobile botnet. Therefore, we do further simulations to test the performance of the three topologies when node failures happen. In the first test,

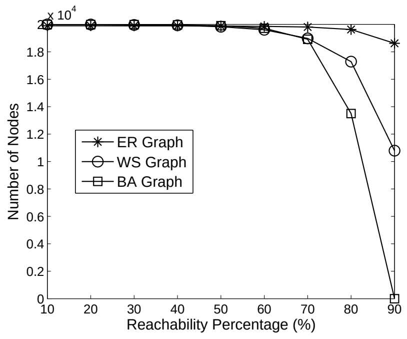
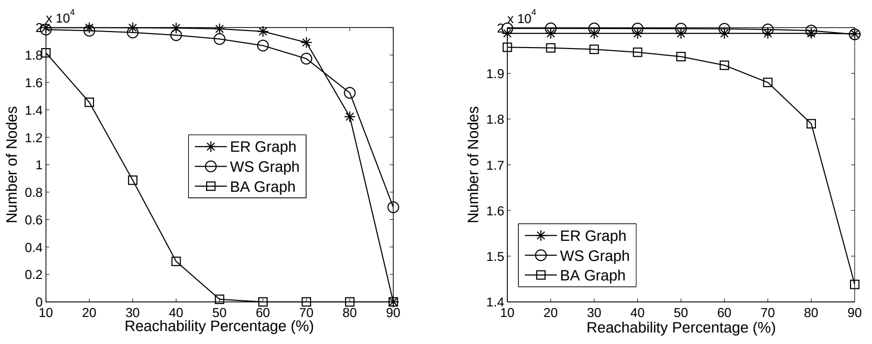


**Fig. 5.** Resistance to random node failures

we randomly removed 10% of the nodes from the botnet before each simulation. And then we observe the command propagation within the remained botnet. The simulation results are shown in Fig. 5. We can find that in the ER Graph the reachability can still exceed 90% from most of the nodes in 20 minutes; however, in the WS Graph only half of the nodes can make this and the situation in the BA Graph is even worse: the maximal reachability percentage can only exceed 80% if the start node is carefully selected. Therefore, we know that the ER random topology is more robust to the random node failures compared with the scale free and the small world topologies. Beside random node failures, we also test the resistance of the three topologies to the selective node failures. In this test, instead of random removing, we removed 10% of the nodes that are the most connected (i.e., nodes with the highest degrees). The simulation results for this case are shown in Fig. 6. Obviously, the influence is enlarged. In the ER Graph no node can reach 90% of the other nodes in the botnet within 20 minutes now and only about 70% of the nodes can achieve a reachability of 80%. The situation in the WS Graph is a little better: there are still around 7000 nodes that can make a reachability percentage over 90%. Compared with them, the situation in the BA Graph is much worse: less than 4000 nodes (20%) can achieve a reachability of 40%. We then extended the simulation time to 25 minutes. At this time, almost all the nodes in the ER Graph and the WS Graph can achieve a reachability larger than 90%. However, in the BA Graph the situation is still



(a) t=20 minutes (b) t=25minutes

**Fig. 6.** Resistance to selective node failures

worse: only less than 4000 nodes can bypass a reachability of 80%. Thereby, the ER Graph and the WS Graph are less affected by the selective node failures.

**Conclusion** : based on the simulations and analysis above, we can conclude that in general the ER model is appropriate to build our SMS botnet. Under this topology with 20000 nodes, if each node knows 6 other peers on average, a command can be propagated to over 90% of the nodes in 20 minutes from any node. And in this process no more than 4 messages are required to send on each node. It's also robust to both random and selective node failures.

## 5 Botnet Construction

In the last section, we know that the ER random graph is a suitable topology for our SMS-based mobile botnet. So in this section we introduce a mechanism to construct this topology. We do not discuss the ways to infect mobile phones but simply assume that a number of mobile phones have already been compromised. What we concern is how these mobile phones get to know each other and finally form a botnet with the random graph topology.

Our proposal utilizes an internet server to help the construction. In this scheme, we first set up a helper server on the internet. And then when a mobile phone is infected by the malware of the botnet, it is made to actively connect to the helper server through internet to register itself and get a list of neighbors assigned by the server. Of course such kind of communication should be also hidden from the users under the help of our rootkits. The details of the above construction process are shown in Fig. 7, which can be divided in to four steps:

1. For a specific node (mobile phone) $n_i$, after being infected, it registers itself on the helper server by informing its phone number.

2. The helper server replied to $n_i$ with a unique handshake sign $s_i$ and a neighbor set $L$ containing $\frac{dm}{N}$ nodes randomly selected from the infected phones

that have already registered on the server. The handshake sign $s_i$ will be used later by other bots to start a neighbor making process with this node. We denote by $d, m, N$ the expected average degree of the botnet, the current count of registered phones and the expected population of the botnet, respectively. For each element $j \in L$, it consists of two parts: its phone number and its handshake sign $s_j$. The server will refuse further registration once the total number of the registered infected phones reaches $N$;

3. After receiving the reply, $n_i$ begins to make neighbors with those nodes appeared in $L$ one by one. This is done by sending them their specific handshake signs via SMS.

4. When a node $n_j$ receives its handshake message from $n_i$, it knows that it has been selected as a neighbor of $n_i$ by the helper server. It then replies to $n_i$ with a random selected normal message from its inbox or outbox. And several bytes at a fixed location in this message will be used as the secret key for the future communication between $n_i$ and $n_j$. Based on this scheme, we can obtain

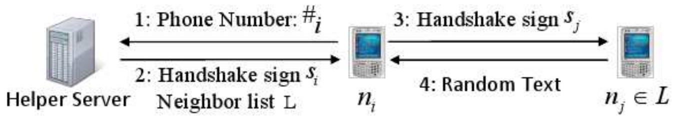


**Fig. 7.** Construction Protocol

a SMS botnet with the ER random graph topology, which is strongly desired. The proof for this assertion is given as follows:

*Proof.* Assume $n_i$ is the $i$-th node that registers on the helper server. Then, for each earlier registered node $n_j$ that $j < i$, it is selected as a neighbor of $n_i$ with the probability $\frac{1}{i-1} \cdot \frac{d(i-1)}{N} = \frac{d}{N}$; i.e., in the generated topology graph the probability that $n_i$ is connected with $n_j$ $(j < i)$ is $\frac{d}{N}$. On the other hand, for each latter registered nodes $n_k, k > i$, $n_i$ will be selected as its neighbor with the probability $\frac{1}{k-1} \cdot \frac{d(k-1)}{N} = \frac{d}{N}$; i.e., in the generated topology graph the probability that $n_i$ is connected with $n_k$ $(k > i)$ is also $\frac{d}{N}$. Thereby, we obtain the conclusion that the generated topology is our desired random graph. And the average degree is $d$.

For this construction mechanism, we have to notice that the introduced helper server may become the single failure point of the whole botnet especially when the infected mobile phones connect to it via the cellular network. Because once this server is disclosed, the telecom operators can easily create a signature based on the destination IP or URL to capture all the following infected phones trying to register themselves. To deal with this problem, the botmaster can take the following measures:

(1) Instead of using the cellular network like 3G, make the bots first search for available WiFi access points to communicate with the helper server. Compared with the cellular network, WiFi is much more stealthy because all the incoming and outgoing data are out of the monitoring of the telecom operators.

(2) Limit the time period for the construction stage. When the botnet size reaches an expected value or the time exceeds a predetermined threshold, directly refuse the later registration and even remove the server.

(3) Use a popular Internet service like HTTP to communicate between the server and the bots. In addition, all these communications should be encrypted.

(4) Use multiple helper servers instead of just one. This increases the hardness of the filtering.

## 6 Botnet Maintaining

Another major challenge in the design of a botnet is how to maintain a botnet after it is constructed. This involves two aspects: first, the botmaster may want to learn the runtime statuses of his controlled bots, such as how many bots are online, the results of an attack, etc. Therefore, in addition to an effective C&C channel, the botmaster also needs an effective report channel. Second, for 'security' reason, the botmaster has to update bot codes regularly. Therefore, we also have to design an effective updating mechanism.

It's difficult to fulfill these two tasks simply relied on SMS. For the botnet reporting, it's a process of information aggregation. If the botmaster collects bot reports via SMS messages, the used mobile phone will be quickly overwhelmed by a mass of messages and be noticed by the defenders. For the botnet updating, the data required to transfer usually greatly exceed the maximum payload of a single SMS message. So it's also unsuitable to deliver the updates via SMS.

As we known, after a long time developing, the traditional PC botnets usually have implemented some mature channels to collect and distribute data among their controlled bots. Therefore, if we can associate our mobile bots with some PC bots, the traditional PC botnet can help to maintain the mobile botnet. Our maintain mechanism showed in Fig. 8. is based on this idea. First of all, the botmaster has to collect a set of PC zombies that are visible on the internet, which means remote computers can actively connect to them if a malware installed on them listens on some ports. These zombies do exist because many network firewalls allow inbound connections to some special ports like 80. With this set, when the hijacked phones register themselves on the helper server, we assign each of them such a zombie. Then, after being equipped with a special malicious daemon these zombies can be used to collect reports and deliver updates for the botmaster: when the botmaster wants to learn the runtime information of the mobile botnet, he first issues a report command to the moble botnet via the SMS channel. After receiving the command, the smart phones are made to covertly connect to their assigned PC zombies to upload their information. Then, the botmaster can collect these information by commanding the PC botnets. Similarly, when the botmaster wants to update their malware, he first distributes the updates to the PC zombies through the C&C channel of the PC botnet. And then he issues an update command in the SMS botnet to ask the hijacked phones to download the updates from their associated zombies.

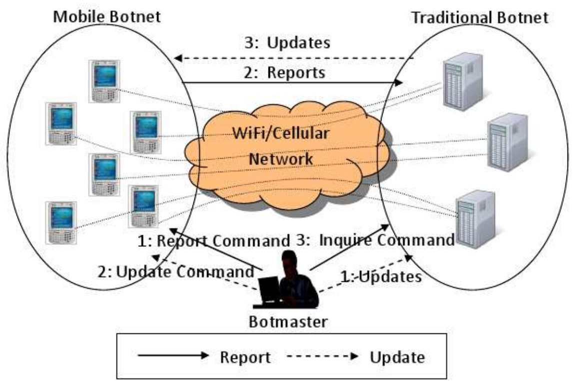


**Fig. 8.** Maintaining of the SMS-based mobile botnet

In the process of maintaining, we may take the following measures to increase the stealth of our mobile botnet:

(1) The same as in the construction process, bot phones should try to use WiFi instead of cellular network to communicate with their internet peers.

(2) The communication between the bot phones and the PC zombies should be encrypted with individualized keys, which means different pairs of phones and zombies use different encryptions.

(3) The mobile bots should change their associated PC zombies regularly.

## 7 Defense Strategies

The modeling and simulation in the previous sections have shown that it's entirely possible to construct an efficient and stealthy mobile botnet using SMS as its C&C channel. We are not smarter than the hackers. They may also have discovered this powerful tool. Therefore, in this section we consider several defense strategies against this threat.

According to our analysis, the most effective defense strategy against out SMS botnet is to disable the ability of malware to send or receive SMS message without the knowledge of the users. The best way to achieve this is to add some non-software-controlled signals (ringing, light flash, vibration, etc.) on the hardware level to inform the users that their mobile phones are sending or receiving SMS messages. Now, when an infected mobile phone attempts to send a botnet command covertly, the user can learn it immediately. This mechanism can be also extended to defend other mobile malware that utilize sensitive resources like WiFi by introducing hardware signals for the uses of these resources.

The second defense strategy has to first develop some special honeypots to capture the malware installed on the bots. This should take the propagation method of the malware into consideration. If they spread via local wireless protocol like Bluetooth or WiFi, the defenders can distribute vulnerable honey phones in crowded places with their Bluetooth or WiFi modular open to wait for being attacked and installed malicious codes. If they spread via social engineering by sending spam mails or hanging horses on the websites, the defenders can actively visit URLs contained in the spam mails or malicious sites to download and analyze the malicious codes. Then, once the malicious codes of our botnet are captured, defenders can take at least two further measures to detect or disable the botnet. Firstly, as we introduced earlier, our botnet is constructed via a helper server on the internet. Therefore, the defenders can perform reverse engineering of the obtained malicious codes to extract the embedded IP address or the URL of this server and then deploy a targeted filtering signature on the internet access points in their controlled cellular network to detect all the bots trying to connect with this server. Secondly, the defenders can also simply run the malicious codes to infiltrate the botnet. Then they can learn all the neighbor bots as well as the associated PC zombies used to upload runtime information and download updates. So the defenders can keep the latest version of the bot codes. In addition, after infiltrating the botnet, the defenders can receive the newest command issued by the botmaster as those ordinary bots. Then if this command is to attack a specific server, phone or send spam, the defenders can quickly deploy corresponding filtering signatures in the appropriate places in the cellular network to detect and block all the infected phones that are launching these attacks.

## 8 Conclusion

In this paper, we aim to investigate the possibility to utilize SMS as a medium for the mobile-botnet command and control. For this purpose we design a proof-of-concept SMS-based botnet based on a simple flooding algorithm. We mainly study the performance of this botnet under three different topologies including the random graph topology, small world topology and the power law topology. According to our simulations, the random graph topology outperforms the other two. Within this topology, a newly issued command can reach over 90% of the total 20000 bots in 20 minutes and no more than 4 messages are required to send for each bot in this process, which is quite efficient. This topology is also robust to both random and selective node failures. We then discuss how to construct and maintain this botnet. Based on these studies, we obtain the conclusion that the SMS-based botnet is indeed a serious threat on the security of the mobile computing environment. Therefore, we further explore several defense strategies against this botnet.

## References

1. Virus News of Kasperksy Lab. Popular Porn Sites Distribute a New Trojan Targeting Android Smartphones (2010), `http://www.kaspersky.com/news?id=207576175`
2. Porras, P., Saidi, H., Yegneswaran, V.: An analysis of the Ikee.B (Duh) iPhone botnet (2009), `http://mtc.sri.com/iPhone/`
3. Lelli, A.: Security Response: A Smart Worm for a Smartphone-WinCE.PmCryptic.A (2008), `http://www.symantec.com/connect/blogs/smart-worm-smartphone-wincepmcryptica`
4. Apvrille, A.: Symbian worm Yxes: Towards mobile botnets? In: 19th Annual EICAR Conference, France (2010)
5. Mulliner, C., Vigna, G.: Vulnerability Analysis of MMS User Agents. In: ACSAC' 06, Miami Beach, USA (2006)
6. F-Secure Corporation. Worm:SymbOS/Mabir.A (2005), `http://www.f-secure.com/v-descs/mabir.shtml`
7. Ferrie, P., Szor, P., Stanev, R.: Security Response: SymbOS.Cabir. Symantec Corporation (2007)
8. Vanhorenbeeck, M.: Mobile botnets: an economic and technological assessment (2008), `http://www.daemon.be/maarten/mobbot.html`
9. Flo, A.R., Josang, A.: Consequences of Botnets Spreading to Mobile Devices, In: 14th Nordic Conference on Secure IT Systems, Oslo (2009)
10. Campbell, M.: Mobile botnets show their disruptive potential. The New Scientist, Vol.204(2734) (2009)
11. Traynor, P., Lin, M., Ongtang, M., et al.: On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core. In: CCS'09, Chicago, USA (2009)
12. Singh, K., Sangal, S., Jain, N., et al.: Evaluating Bluetooth as a Medium for Botnet Command and Control. In: DIMVA'10, Bonn, Germany (2010)
13. Zeng, Y., Hu, X., Shin, K.G.: Design of SMS Commanded-and-Controlled and P2P-Structured Mobile Botnets. University of Michigan Technical Report CSE-TR-562-10 (2010)
14. Papathanasiou, C., Percoco, N.J.: This is not the droid you are looking for. In: DEF CON 18 (2010)
15. Singh, K., Srivastava, A., Giffin, J., et al: Evaluating Email's Feasibility for Botnet Command and Control. In: DSN'08 (2008)
16. Walker, J.: Stego!Text Steganography (2005), `http://www.fourmilab.ch/javascrypt/stego.html`
17. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
18. Erdos, P., Renyi, A.: On random graphs I. Publicationes Mathematicae.15 (1959)
19. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science 286 (1999)
20. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393 (6684) (1998)
21. Davis, C., Neville, S., Fernadez, J.M., et al.: Structured Peer-to-Peer Overaly Networks: Ideal Botnets Command and Control Infrastructures? In: ESORICS'08, Spain (2008)
22. Csardi, G.: The igraph library (2005) `http://igraph.sourceforge.net/index.html`
23. Mulliner, C., SeifertRise, J.P.: Rise of the iBots: 0wning a telco network. In: MALWARE'10, France (2010)