Hindawi
Security and Communication Networks
Volume 2022, Article ID 4196945, 10 pages
https://doi.org/10.1155/2022/4196945

Research Article

Automating Group Management of Large-Scale IoT Botnets for Antitracking

Pengyu Pan,[1,2] **Xiaobo Ma**,[1,2] **Yingjie Fu**,[1,2] **and Feitong Chen**[1,2]

[1]*MOE Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China*
[2]*Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China*

Correspondence should be addressed to Xiaobo Ma; xma.cs@xjtu.edu.cn

Received 23 December 2021; Revised 13 March 2022; Accepted 31 March 2022; Published 14 April 2022

Academic Editor: Weizhi Meng

With the popularity of Internet of Things (IoT) devices, IoT botnets like Mirai have been infecting as many devices as possible such as IP cameras and home routers. Because of the sheer volume and continual operation of many vulnerabilities (many users do not pay much attention to IoT update alerts and leave the configurations by default) of IoT devices, the population of an IoT botnet becomes increasingly tremendous. The growing population, though making a botnet powerful, results in an increased risk of exposure. Specifically, once a bot is captured, the command and control (C&C) channel may be cracked and then tracked, potentially rendering more bots being discovered. To solve this problem, this paper proposes an automated approach to group management of large-scale IoT bots. The basic idea of the proposed approach is to establish a reliable and unsuspicious social network-based C&C channel capable of automatically grouping bots, wherein a group of bots have a unique ID that is against cross-group tracking. The Diffie–Hellman key exchange method is leveraged for efficiently generating the unique group ID, thereby scaling up automatic bot grouping. We refer to the botnet proposed in this paper as a multichannel automatic grouping botnet (MCG botnet) and conduct verification experiments using social networks and more than 2,000 docker nodes. The experimental results show that the MCG botnet has the ability of automatic grouping and antitracking.

1. Introduction

A botnet consisting of a large number of computers is controlled by the botmaster. The botmaster can remotely control bots to initiate DDoS attacks, send spam, collect user privacy, and conduct other malicious activities [1]. With the popularity of Internet of Things (IoT) devices, IoT botnets like Mirai have been infecting as many devices as possible such as IP cameras and home routers. Because of the sheer volume and continual operation of many vulnerabilities (many users do not pay much attention to IoT update alerts and leave the configurations by default) of IoT devices, the population of an IoT botnet becomes increasingly tremendous. The growing population, though making a botnet powerful, results in an increased risk of exposure.

Although there has been quite a lot of research on the security of IoT devices [2–4], Mirai and its variant botnets still pose a huge threat to increasingly more IoT devices [5]. On the other side, when the botmaster faces a large-scale IoT botnet, how to effectively manage the group is a big challenge. Specifically, once a bot is captured, the command and control (C&C) channel may be cracked and then tracked, potentially rendering more bots being discovered. The disclosure of the bot program will lead to the cracking of key information such as communication patterns.

When designing a botnet, how the botmaster communicates to its bots is a problem of top priority [6]. Centralized communication is the simplest and most common method. The salient feature of this type of communication is that it has a command and control (C&C) server. The botmaster and its bots communicate through the server using protocols like IRC and HTTP. Centralized communication has the advantages of low delay, scalability, and large communication capacity. However, its disadvantages are also obvious. Once the C&C server is shut down or taken over, the entire botnet will most likely be destroyed. To alleviate this

problem, the DGA algorithm is applied to the design of botnets. Through such an algorithm, the botmaster and bots can generate the same domain names hosting C&C servers.

However, if one obtains the bot program, it is feasible to reverse-engineer the DGA algorithm. Therefore, the DGA-based botnet could be blocked by predicting and registering domain names in advance. Moreover, the use of the DGA algorithm will cause bots to generate a large number of DNS (failure) queries, which is a distinguishable detection feature. With the aid of machine learning, defenders can easily identify this type of botnet through traffic characteristics [7].

To improve the stealthiness and survivability of botnet communication, more and more botnets use P2P to communicate. Compared with centralized botnets, P2P botnets effectively avoid the problem of single point of failure. However, P2P botnets are not absolutely reliable. Firstly, if a P2P botnet uses a proprietary P2P communication protocol, it is easy to be detected and shut down. At the same time, Sybil attacks and index poisoning will also affect the reliability of P2P botnets. In addition, the delay of the P2P network leads the bot to be not able to receive the commands from the botmaster in time.

To implement real-time, stealthy, and robust botnet communication, the new generation of botnets began to gradually use social networks and other public services as their C&C channels, such as Twitter, Facebook, and Cloud storage. However, no matter what communication channels are used, once the botnet is captured and reverse-engineered, the botnet faces two problems.

P1. How to ensure the survivability of botnets.

P2. How to manage the large number of bots without being tracked.

To solve these problems, this paper proposes an automated approach to group management of large-scale IoT bots. The basic idea of the proposed approach is to establish a reliable and unsuspicious social network-based C&C channel capable of automatically grouping bots, wherein a group of bots have a unique ID that is against cross-group tracking. The Diffie–Hellman key exchange method is leveraged for efficiently generating such a unique group ID, thereby scaling up automatic bot grouping.

The main contributions of this paper are as follows:

(i) We establish a reliable C&C channel through social networks for our MCG botnet. To verify the timeliness and reliability of our established C&C channel, we deploy a real-world testbed consisting of Twitter and GitHub.

(ii) We dissect the C&C channel in consideration of its fine-grained functionalities, namely, control channel, command channel, return channel, and registration channel. We analyze the functions and characteristics of these four channels in detail in comparison to the characteristics of existing social networks and public network services.

(iii) We leverage the Diffie–Hellman key exchange method for automatic bot grouping of the MCG botnet. To evaluate the grouping performance, we use more than 2,000 docker nodes to conduct experiments. The results show that as the number of bots increases, the bots could be effectively grouped in real time without the botmaster's intervention, enabling the botmaster to control each group in an isolated fashion with built-in antitracking capability.

The structure of this paper is as follows: Section 2 discusses related work. Section 3 elaborates the design and workflow of the MCG botnet. Section 4 conducts performance evaluation. Section 5 discusses countermeasures against the MCG botnet. We finally conclude the paper in Section 6.

2. Related Work

Botnets have been a hot research area. With the wide application of machine learning-based detection techniques, traditional centralized botnets (e.g., based on IRC and HTTP protocols) and P2P botnets could be detected under certain conditions [8–12].

To evade detection, more sophisticated C&C structures like hybrid P2P and more dynamic C&C resource management like URL Flux are used for improving botnet survivability [13]. Meanwhile, increasingly more botnets adopt social networks, such as Twitter, as C&C channels [14]. For example, Pantic et al. designed a botnet that exploits Twitter accounts as C&C channels, wherein the account names are automatically generated based on Markov chains. The botnet is of strong stealthiness due to its social network-based and automatically generated channels [15]. Yin et al. proposed a social network-based botnet with antidisruption capability [16]. The proposed botnet introduces a divide-and-conquer and automatic reconstruction mechanism to improve survivability. In addition to directly leveraging social networks as C&C channels, botmasters are incorporating steganography techniques, such as image steganography and Unicode steganography [17, 18], to further improve the stealthiness of botnets. Besides social networks, many Internet services are also used by botnets. For example, Google's C2DM service provides C&C channels for mobile botnets [19]; SLBot is a serverless botnet based on Service Flux [20]; the URL shortening services are used to evade network-level detection and blacklisting [21].

Social network-based botnets have been paid much attention to in the literature. Chew et al. proposed a social network-based hybrid botnet (RSHB) [22]. When RSHB uses social networks for communication, it also adds a reputation mechanism and a resurrection mechanism, which further improves the robustness of the botnet. In addition to social networks, blockchain has also become a focus for botnet designers. The op_return field of Bitcoin provides a certain storage space, which can transmit information simply and efficiently [23]. The ECDSA algorithm used by Bitcoin has also been used to build subliminal channels and has been proven feasible [24]. To improve the efficiency of ciphertext transmission and reduce the number of interactions, Luo et al. proposed a covert communication

method based on Bitcoin transactions [25]. The method of communication using Bitcoin inevitably comes at the cost of transaction fees. To address this limitation, the Whisper protocol used in Ethereum allows botmasters to control bots at almost zero cost [26].

Existing studies provide covert communication between the botmaster and the bot. However, they do not take into account the situation that a bot using their covert communication is hacked or captured by the honeynet. The proposed MCG botnet focuses on how to effectively and automatically group large-scale botnets so to improve the antitracking capability. That is, capturing and containing some bots do not threaten the stealthiness of other bots and the survivability of the entire botnet.

3. Design

In this section, we will detail the MCG botnet architecture and its workflow.

We refer to the proposed botnet as a multichannel automatic grouping botnet (MCG botnet). The multichannel property means that the C&C channels are categorized into different types according to their functionalities in terms of botnet communication. Different types of channels are separate yet cooperative, beneficial to the robustness of the botnet. The MCG botnet has four types of channels, namely, registration, command, control, and return channels. The registration channel is used to collect the information of newly infected bots. Having a number of registered bots, the botmaster stores the command in the command channel, and then a bot could find the address of the command channel in the control channel so as to retrieve the command. After executing the retrieved command, the bot transmits the results to the botmaster through the return channel. The automatic grouping property enables bot group management hierarchically. That is, the parent group is in charge of the management of the child group. The taking down of one group does not affect its parent or child group (if the child group has already been generated).

3.1. Architecture. Designing a botnet architecture needs to consider the communication requirements of different parties like the botmaster, the bot, and the C&C channels. However, the communication requirements of different parties may differ in various scenarios. For example, when launching DDoS attacks, the botmaster propagates commands to its bots in a timely manner due to the need of all bots simultaneously participating in the attack. In contrast, when bots transmit information like user privacy, the timeliness is not a top priority. According to the communication requirements of the MCG botnet (see Table 1), the MCG botnet is architected, comprising four channels, namely, command channel, control channel, return channel, and registration channel.

3.1.1. Command Channel and Control Channel. Command and control (C&C) channel is a general term involving almost all aspects concerning how a botmaster controls his/her bots (e.g., transmitting commands, sending botnet configuration information). Existing studies do not consider command and control channels separately. Since the command channel and the control channel have distinct communication requirements, we design the two channels separately to fulfill their distinct communication requirements.

Command Channel. We use the command channel to store the commands released by the botmaster. When selecting a command channel, one should focus on the storage capacity as well as security.

Control Channel. We use the control channel to store the address of command channel. The bots obtain the address of the command channel through the control channel and finally obtain the commands stored by the botmaster in the command channel. Control channel should be considered from a reliability perspective.

The whole process is shown in Figure 1, which is divided into four stages.

Phase I **(1–3)**: The botmaster stores ciphertext and signature in command channel.

Phase II **(4)**: Convert the address of the command channel into a short URL and publish it to the control channel.

Phase III **(5)**: The bots regularly obtain the short URL from the control channel.

Phase IV **(6–8)**: The bots find the command channel through the short URL to extract information.

In Phase II, we are facing an issue with synchronizing the control channel address between the botmaster and its bots. Based on the previous experience of designing botnets [6], a botmaster generally uses DGA algorithm or similar algorithms, which brings new problems while completing synchronization. If the bot program is obtained by defenders, the seeds used will be obtained. It is fatal to botnets because defenders can predict subsequent addresses in the same way and block addresses that may be C&C channels in advance.

The fundamental problem is that no matter how the botmaster generates new addresses, they will be considered malicious. Therefore, this paper adopts social network as the C&C channel to avoid this problem. The botmaster and its bots have agreed on the designated social network user in advance. The botmaster will convert the address of the command channel into a short URL and send it to the comments section of the designated user's blog post. The bots only need to periodically read the comments section of the designated social network user to obtain the command channel address. Then the bots obtain the commands stored in advance by the botmaster from the command channel. Neither social network administrators nor defenders can block a normal user with just a malicious comment. Therefore, the MCG botnet does not need to consider the synchronization between botmaster and bot. For the MCG botnet, it is necessary to find a social network that can retain malicious comments from users for a long time as a control channel.

Table 1: Channel features.

Channel	Timeliness	Security	Anonymity	Reliability	Capacity
Control	*Y*	—	—	*Y*	—
Command	—	—	—	—	*Y*
Return	—	—	*Y*	*Y*	*Y*
Registration	—	*Y*	—	—	—

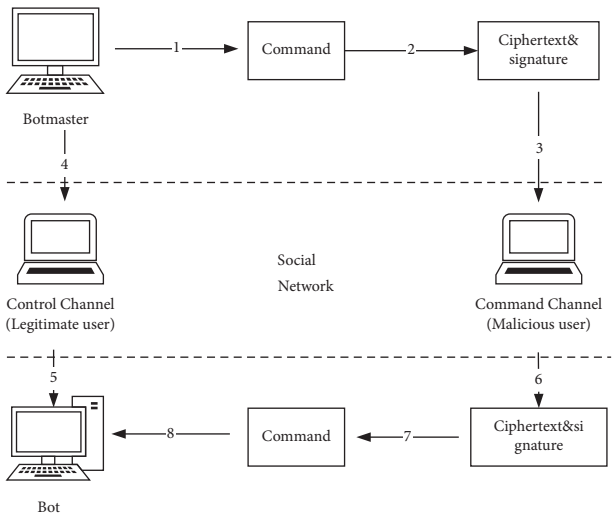


Figure 1: Command and control channel design.

3.1.2. Return Channel. Return channel is used by bots to transmit information to the botmaster. When the botmaster uses its bots to steal user privacy, the bots need to send information to the botmaster frequently, and the amount of information may be relatively large. Therefore, the return channel should be easy to register and use. As the receiver, the botmaster should focus on the issue of anonymity to avoid being traced by defenders.

3.1.3. Registration Channel. The bot that infects normal devices is called the parent bot, and the new infected bot is called the child bot. In the MCG botnet, the child bot needs to rely on the parent bot to complete the registration so as to formally join the botnet. The botmaster obtains the scale of the entire botnet through the registration channel.

In recent years, defenders have generally adopted technologies such as honeypots and honeynets to monitor, detect, and analyze the entire botnet. Therefore, the registration channel should ensure real time and reliability and avoid the leakage of some key information as much as possible. This is important for grouping of botnets.

So far, we detailed the four channels of the MCG botnet. As shown in Table 1, we summarize the four channels from the aspects of timeliness, security, anonymity, reliability, and large capacity requirements (Y: there is a clear demand for this indicator; —: there is no clear demand for this indicator).

3.2. Bot Grouping. The ever-expanding botnet also brings two new problems to the botmaster:

P1. How the botmaster can control part of the botnet more accurately when the botnet scale is large.

P2. Once a bot is captured, the information such as the C&C channel used by the botnet will be leaked, potentially making the entire botnet taken down/over.

To address the above problems, we propose the following techniques:

(1) Bot grouping: Child bots will be divided into new groups by parent bots without intervention by the botmaster. The botmaster can also adjust the scale of each group of bots in real time according to the size of the botnet.

(2) Botnet group management: To avoid the harm caused by the bot being captured, this paper generates exclusive configuration information for each group of the MCG botnet. Once a group of bots are mixed with honeypots, the configuration information leakage will not affect other groups.

The Diffie–Hellman key exchange method is applied for automatic bot grouping, as is shown in Figure 2. First, the botmaster and its bots generate random numbers a and b, respectively. P and G are two prime numbers shared between the botmaster and its bots. They are used to generate A and B. Finally, the botmaster and each bot exchange A and B to generate K, the group ID of the bot. Bots in the same group generate the same random number b.

If any one of a, b, P, and G changes, the shared key K will change accordingly. In the MCG botnet, as long as different groups use different random numbers b, the botmaster and its bots can obtain different K.

It has been explained in Section 3.1 that the child bots can only rely on the parent bots to register. The parent bots will generate exclusive configuration information for the child bots, including encryption and decryption keys and random number b. Subsequently, the parent bots send this information to the botmaster to complete the registration of the child bots. Parent bots and child bots belong to different groups. Parent bots update the random number b regularly to achieve the effect of automatic grouping. The botmaster can also actively request the parent bots to generate a new random number b for the child bots according to the scale of the botnet to complete the scale control of different groups.

Regardless of whether the botmaster intervenes, the random number b is only generated by the parent bots. Then it is sent to botmaster to avoid security risks caused by synchronization. Assuming that a child bot is a honeypot since the bots of different groups use different encryption and decryption keys, defenders can only crack the communication content between the bots of the current group and the

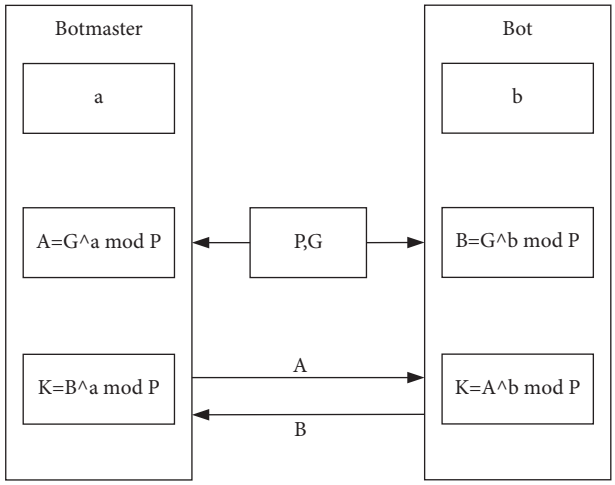


Figure 2: Group ID generation.

botmaster through this honeypot. For a large-scale botnet, the impact is very limited. A large-scale botnet refers to a botnet that consists of a large number of bots, and the bots usually are geographically dispersed in various networks.

3.3. Workflow. When the MCG botnet is initialized, the botmaster will first generate two prime numbers P and G, random number a, and public key A required by the Diffie–Hellman key exchange method. Then the botmaster generates the private key Priv$K$ for signing the ciphertext and the corresponding public key Pub$k$. To reduce the harm caused by key leakage, each group uses a different key for encryption and decryption. Both symmetric and asymmetric encryption algorithms could be used, for example, AES algorithms and RSA algorithms. When the amount of transmitted information is small, asymmetric encryption algorithms could be used. Otherwise, symmetric encryption algorithms could be used.

P, G, A, and Pub$k$ are hardcoded into the bot program. Pub$k$ is used to verify the botmaster's signature. It is different from the RSA key used for encryption and decryption in the bot group. When the MCG botnet is first deployed, the botmaster generates different random numbers b and RSA keys for them according to different groups. The same group of bots uses the same random number and key. The working flow chart of the MCG botnet is shown in Figure 3.

(1) The botmaster uses the RSA public key corresponding to the bot group to encrypt the information. Then Pri$K$ is used to sign and store to command channel.

(2) The botmaster converts the address of the command channel into a short URL and sends it to the control channel, which is the comments section of the designated social network user.

(3) The bots regularly querie the control channel to obtain the address of the command channel.

(4) The bots find the command channel according to the address obtained in the previous step and obtain the information transmitted by the botmaster.

(5) The execution result is transmitted to the botmaster through the return channel.

(6) After a bot (parent bot) infects a new bot (child bot), it copies its own program to the new bot and passes it to the random number b and RSA key (not the random number b and RSA key used by the current bot group).

(7) After step (6), the parent bot will complete the registration of the child bot through the registration channel.

Different bots may originate from different groups. Consequently, when communicating with a bot group, the botmaster should use the public key of the corresponding bot group to encrypt the commands.

For example, Group-A, Group-B, and Group-C are three bot groups that use different public/private key pairs. When the botmaster commands bots in Group-A to perform tasks, it encrypts commands using Group-A's public key. The botmaster sends the encrypted commands to bots in Group-A through C&C channels (e.g., the channel proposed in Sec. 3.1.1). Bots in Group-A, Group-B, and Group-C will receive the encrypted commands, but only bots in Group-A can decrypt them. After bots in Group-A execute the commands, the execution result is transmitted to the botmaster through the return channel.

The group size during automatic grouping of our proposed MCG botnet can be accomplished in a predefined manner. Consider that bots in Group-A infect new bots to constitute Group-B. Given the size of Group-A S_a, the botmaster just needs to define the maximum number of bots that each bot in Group-A could infect (e.g., n) to control the upper bound of the size of Group-B (i.e., $S_a n$). We control the upper bound because in most cases a large group size would lead to the taking down of many bots. However, since bots in Group-A may infect overlapping sets of bots, the actual size of Group-B may be much smaller than the upper bound. In such a case, the botmaster could assign a larger value to n to further increase the upper bound. Note that if one keeps n constantly exceeding 1, the child group size may grow larger than the parent group size. To stop the group size growing unlimitedly, the botmaster normally needs to make n decrease from generation to generation.

Consider that a bot in Group-A infects a set of new bots. Such a set of new bots, along with those infected by all the remaining bots in Group-A, constitute a new group named Group-B. Group A is the parent group, while Group-B is its child.

When the bots in Group-A infect a new bot, they generate a grouping random number, which is the same across all the bots in Group-A due to the same seeding mechanism, as well as a symmetric key (or a private/public key pair), for Group-B. The grouping random number and the key will be sent to all the bots in Group-B directly, and to the botmaster through the registration server, by bots in Group-A. The random number is used to generate the ID of Group-B, and the key is for encrypting the communication message between the botmaster and the bots in Group-B.

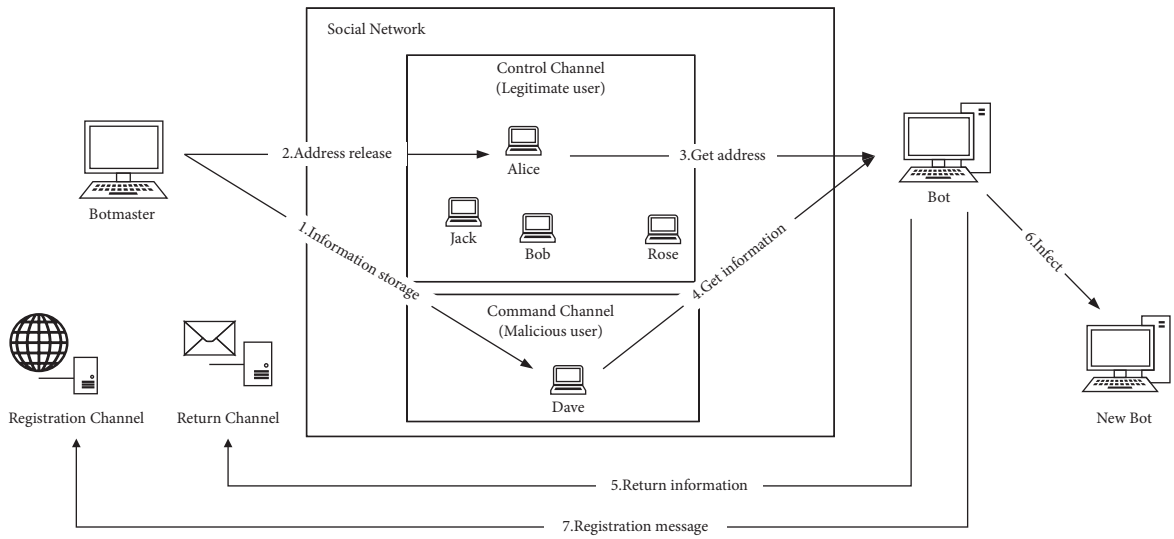


Figure 3: The overall workflow of the MCG botnet.

The exposure of the key would affect the confidentiality of the communication between the botmaster and the bots in Group-B. However, the affection is limited because the communication between the botmaster and the bots in other groups remains confidential.

4. Experiment

To evaluate the MCG botnet, we first selected and verified the four channels and finally conducted a simulation experiment on the botnet automatic grouping.

4.1. Channel Selection. In Section 3.1, we have described the functions and requirements of the four channels in detail. As shown in Table 2, to select the appropriate channel, we investigated popular social networks and e-mail services. In Table 2, the "Register" column represents whether using the service is conditioned on registration. The "High Capacity" means whether uploading programs, pictures, and other large-sized objects is allowed.

The control channel uses the comments section of legitimate users in the social network to transmit the address of the command channel. The two following issues should be considered: (1) The social network allows the botmaster to comment freely in the comments section of other people's accounts. (2) Social networks do not strictly control comments.

The bots obtain the address of the command channel from the control channel and obtain the commands and so forth. Therefore, the command channel should have a larger capacity. As shown in Figure 4, we use GitHub as the command channel and Twitter as the control channel.

Among all channels, the command channel and the control channel are crucial to the robustness of the MCG botnet. To verify the robustness, we measure the survival times of the command channel over GitHub and the control channel over Twitter.

As Figure 5 shows, we register one GitHub account to store the ciphertext of the command "DDoS www.test.com" along with the signature of the bomaster and one Twitter account to post the address of the command at the comments sections of those Twitter accounts that our registered account follows. The address of the command is a short URL (compressed by TinyURL [27]) locating "DDoS https://www.test.com" at GitHub. Note that there are 20 Twitter accounts that our registered account follows, and these followed accounts are all news accounts due to their long time stability that is beneficial to short URL propagation. We post one short URL at the comments section of each of the 20 news Twitter accounts and measure the survival times of the 20 posted short URLs (i.e., 20 control channels). Also, we measure the survival time of the control channel, the time that the posted short URLs are accessible. Figure 5 shows the results. We see that all the 20 control channels and one command channel survive for the entire 72 hours, indicating the robustness of the MCG botnet.

As shown in Table 3, we checked the command channel and the control channel every 24 hours and found that the survival rate was 100%, and comments on Twitter can be accessed immediately. This experiment demonstrates the real time and robustness of using the Twitter user comments section as the control channel and GitHub as the command channel.

The return channel is used by the bots to actively send information to the botmaster so the botmaster should be more careful to avoid leaking too much personal information. The botmaster can use e-mail as the receiver and configure different sending mailboxes for different bot groups through the command channel. The registration channel is used for the registration of new bots, and the communication volume is small. Due to the needs of botnet grouping, it is necessary to ensure high security and real-time performance. Third-party services are usually difficult to achieve. Therefore, we use a private server as a registration channel.

Table 2: Network service properties.

Type	Name	Register	High Capacity
Social network	Twitter	*Y*	*N*
	Facebook	*Y*	*N*
	Instagram	*Y*	*N*
	Weibo	*Y*	*N*
	GitHub	*Y*	*Y*
E-mail service	Gmail	*Y*	*Y*
	Outlook	*Y*	*Y*
	YOPmail	*N*	*N*
Cloud storage	OneDrive	*Y*	*Y*
	Dropbox	*Y*	*Y*
	Pan.baidu	*Y*	*Y*

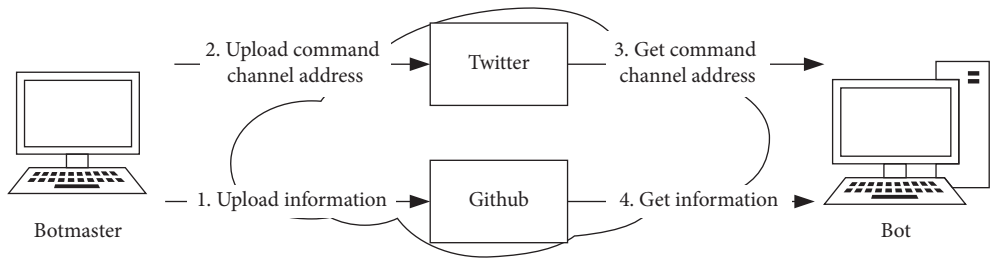


Figure 4: Real-world control channel and command channel testbed.

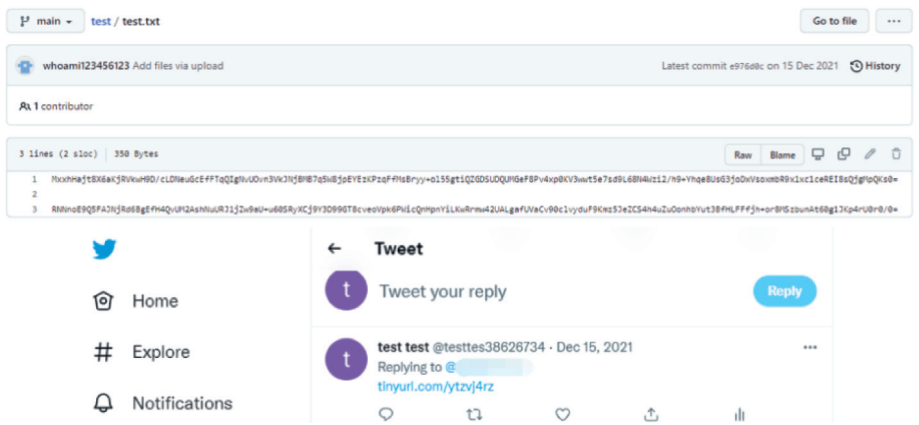


Figure 5: The GitHub account to store the ciphertext of the command and the Twitter accounts where we post the address of the command (i.e., short URL).

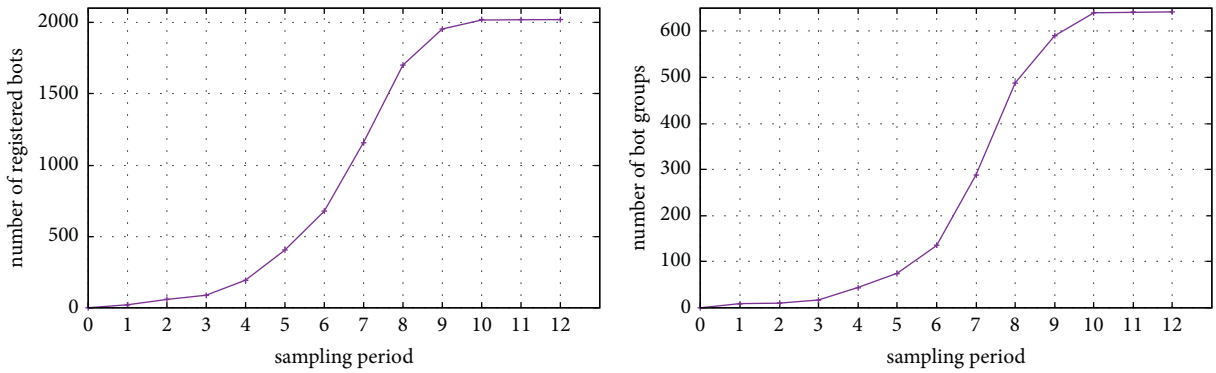
Table 3: Survival times versus the number of command and control channels.

Survival time (hours)	24	48	72
Number of command channels	1	1	1
Number of control channels	20	20	20

We have proven the robustness of the command channel and the control channel. Even if the return channel and registration channel are blocked, the botmaster can still contact its bots by changing the e-mail and server address through the command channel and the control channel.

4.2. Automatic Grouping. To verify the grouping effect of the MCG botnet, this paper uses 2020 docker nodes to simulate IoT devices to carry out simulation experiments. Using six high-performance services, the CPUs are all Intel(R) Xeon(R) Gold 6248R CPU @ 3.00 GHz, and the RAMs are all DDR4, 503G. The operating systems are Ubuntu20.04, Ubuntu18.04, and Ubuntu16.04.

During initialization, 5 of the docker nodes are initialized as bots, which are 5 groups, respectively. As to the C&C method in the docker node experiment in Section 4.2, our major goal is to verify the automatic bot grouping capability, which is independent of the specific C&C method. Therefore, in the docker node experiment, we use neither the



(a) (b)

Figure 6: Automatically grouping experiments. (a) Changes in the number of bots. (b) Changes in the number of bot groups.

Table 4: Statistics of bot grouping.

Number of bots in a single group	≥15	≥10	≥5	≥1	Total
Number of groups	5	24	107	506	642
Number of bots	92	272	703	953	2020

proposed C&C method nor any other methods. As shown in Figure 6(a), the number of registrations of bots increases exponentially. Due to the limited number of docker nodes, the number of bots will gradually stabilize. Therefore, the number of registrations of bots and the number of groups show an overall S-shape. As shown in Table 4, it can be seen that, with the continuous increase of bots, the number of bot groups has also increased sharply.

In the initial stage (period: 0–3), due to the small number of bots, the speed of new bots joining the botnet is slower and the number of bot groups is small. With the increase in the number of bots, it will soon enter the explosive stage (period: 3–8), and bots show an exponential growth trend. At this stage, due to the large increase in bots, the number of groups also shows an exponential growth trend. However, at this time, most bot groups contain fewer bots. As shown in Table 4, there are a total of 136 groups with bots greater than or equal to 5, as well as a total of 1067 bots, which is more than half of the total bots.

As shown in Figure 6(b), it can be seen from the simulation experiment that the MCG botnet has better automatic grouping ability. Even if devices such as honeypots join the botnet, a honeypot can only affect one group at most and will not interfere with the normal operation of the entire botnet. This simulation experiment shows that the MCG botnet greatly improves survivability through automatic grouping.

5. Countermeasures

Compared with other botnets, the MCG botnet's main features are as follows: Automatic grouping botnet is to reduce the harm caused by bot program leakage. The communication between the botmaster and its bots is disassembled according to function, and the social network is used as the control channel. A reliable C&C channel is built as the basis for botnet communication.

Therefore, the focus of detecting the MCG botnet should be to strengthen the monitoring and censorship of social networks. Identity authentication is also an effective approach when using public services. If social network platforms and users can strengthen the review of the comments section, especially some comments that are not relevant to blog posts, it will pose a greater threat to such botnets. In addition, public services that are free to use and registration-free should use verification codes and other methods to verify the user's identity.

When we use news accounts as control channels, the bot needs to access them regularly to get the address of the command channel. This can lead to a rise in page views of these news accounts, attracting the attention of defenders. If a bot is captured (e.g., by honeynet), the social network accounts used by the MCG botnet will be exposed. Defenders can report these accounts to interrupt the operation of the MCG botnet. However, there are many Twitter accounts and a huge number of many other social network accounts that the botmaster can choose as control channels. More importantly, a parent group is able to dynamically define the social network account that the bots in its child group should use, making the control channels of different bot groups completely isolated. In other words, defenders may interrupt the operation of certain groups, but the remaining groups would be still working.

The MCG botnet has group management and anti-tracking capabilities. However, like most botnets, when the MCG botnet performs malicious actions such as DDoS attacks, a large number of bots will generate very similar traffic behaviors. This is very critical to combat botnets. Defenders can collect enough traffic information. Then they can analyze key information such as botnet communication methods, network topology, botnet size, and types of infected devices through machine learning and data mining. Further, they cooperate with relevant social network service providers or equipment manufacturers to propose targeted measures against such botnets.

6. Conclusion

This paper proposed a multichannel self-grouping social botnet called MCG botnet. According to the communication requirements of the MCG botnet, the MCG botnet is

architected comprising four channels, namely, command channel, control channel, return channel, and registration channel. A botnet grouping strategy is designed based on the Diffie–Hellman key exchange method. It greatly enhanced the MCG botnet's survivability, since containing one group does not affect the survivability of all the remaining groups. We perform experiments on Twitter and GitHub to verify the robustness of the C&C channel in a real environment. We also set up a simulation environment containing 2020 docker nodes to test the grouping ability of the botnet. It has propounded implications on personalized group control and antitracking of large-scale IoT botnets.

Data Availability

The original data in the research process can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest in the manuscript.

Acknowledgments

This work was supported in part by National Natural Science Foundation (61972313), Postdoctoral Science Foundation (2019M663725 and 2021T140543), and the Fundamental Research Funds for the Central Universities, China. Xiaobo Ma is an XJTU Tang Scholar supported by Cyrus Tang Foundation.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pp. 41–52, Rio de Janeriro, Brazil, October 2006.

[2] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on IoT security: challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of Things*, vol. 11, Article ID 100227, 2020.

[3] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–782 743, 2019.

[4] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the internet of things networks of smart cities," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020.

[5] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: iot botnet of high wattage devices can disrupt the power grid," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 15–32, Princeton University, Princeton, NJ, USA, 2018.

[6] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[7] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang, "Identifying suspicious activities through dns failure graph analysis," in *Proceedings of the 18th IEEE International Conference on Network Protocols*, pp. 144–153, Kyoto, Japan, October 2010.

[8] A. H. R. A. Awadi and B. Belaton, "Multi-phase irc botnet and botnet behavior detection model," *International Journal of Computer Applications*, vol. 66, 2015.

[9] G. Sagirlar, B. Carminati, and E. Ferrari, "Autobotcatcher: blockchain-based p2p botnet detection for the internet of things," in *Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pp. 1–8, IEEE, Philadelphia, PA, USA, October 2018.

[10] T. Sengupta, S. De, and I. Banerjee, "A closeness centrality based p2p botnet detection approach using deep learning," in *Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, Kharagpur, India, July 2021.

[11] S. Y. Yerima and M. K. Alzaylaee, "Mobile botnet detection: a deep learning approach using convolutional neural networks," in *Proceedings of the 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1–8, IEEE, Dublin, Ireland, June 2020.

[12] S. Almutairi, S. Mahfoudh, S. Almutairi, and J. S. Alowibdi, "Hybrid botnet detection based on host and network analysis," *Journal of Computer Networks and Communications*, vol. 2020, Article ID 9024726, 16 pages, 2020.

[13] C. Liu, W. Lu, Z. Zhang, P. Liao, and X. Cui, "A recoverable hybrid c&c botnet," in *Proceedings of the 2011 6th International Conference on Malicious and Unwanted Software*, pp. 110–118, IEEE, Fajardo, PR, USA, October 2011.

[14] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The rise of social botnets: attacks and countermeasures," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1068–1082, 2016.

[15] N. Pantic and M. I. Husain, "Covert botnet command and control using twitter," in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 171–180, Los Angeles, CA, USA, December 2015.

[16] T. Yin, Y. Zhang, and S. Li, "Dr-snbot: a social network-based botnet with strong destroy-resistance," in *Proceedings of the 2014 9th IEEE International Conference on Networking, Architecture, and Storage*, pp. 191–199, IEEE, Tianjin, China, August 2014.

[17] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: a covert social network botnet," in *International Workshop on Information Hiding*-Springer, Berlin, Heidelberg, 2011.

[18] A. Compagno, M. Conti, D. Lain, G. Lovisotto, and L. V. Mancini, "Boten elisa: a novel approach for botnet c&c in online social networks," in *Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 74–82, IEEE, Florence, Italy, September 2015.

[19] S. Zhao, P. P. Lee, J. C. Lui, X. Guan, X. Ma, and J. Tao, "Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 119–128, Orlando, Florida, USA, December 2012.

[20] D. Wu, B. Fang, J. Yin, F. Zhang, and X. Cui, "Slbot: a serverless botnet based on service flux," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 181–188, IEEE, Guangzhou, China, June 2018.

[21] S. Lee and J. Kim, "Fluxing botnet command and control channels with url shortening services," *Computer Communications*, vol. 36, no. 3, pp. 320–332, 2013.

[22] C.-J. Chew, Y.-C. Chen, J.-S. Lee, C.-L. Chen, and K.-Y. Tsai, "Preserving indomitable ddos vitality through resurrection social hybrid botnet," *Computers & Security*, vol. 106, Article ID 102284, 2021.

[23] S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao, "Zombiecoin: powering next-generation botnets with bitcoin," in *International Conference on Financial Cryptography and Data Security*, pp. 34–48, Springer, Berlin, Heidelberg, 2015.

[24] D. Frkat, R. Annessi, and T. Zseby, "Chainchannels: private botnet communication over public blockchains," in *Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1244–1252, IEEE, Halifax, NS, Canada, July 2018.

[25] X. Luo, P. Zhang, M. Zhang, H. Li, and Q. Cheng, "A novel covert communication method based on bitcoin transaction," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2830–2839, 2021.

[26] M. Baden, C. F. Torres, B. B. F. Pontiveros, and R. State, "Whispering botnet command and control instructions," in *Proceedings of the 2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 77–81, IEEE, Rotkreuz, Switzerland, June 2019.

[27] Tiny, "Tinyurl.com - shorten that long url into a tiny url," 2020, https://tinyurl.com/.