computers

MDPI

Article

Construction and Performance Analysis of Image Steganography-Based Botnet in KakaoTalk Openchat

Jaewoo Jeon and Youngho Cho *

Department of Computer Engineering, Graduate School of Defense Management, Korea National Defense University, Nonsan 33021, Korea

* Correspondence: youngho@kndu.ac.kr

Received: 30 June 2019; Accepted: 19 August 2019; Published: 21 August 2019

Abstract: Once a botnet is constructed over the network, a bot master and bots start communicating by periodically exchanging messages, which is known as botnet C&C communication, in order to send botnet commands to bots, collect critical information stored in bots, upgrade software functions of malwares installed in bots, and so on. For this reason, most existing botnet detection techniques focus on monitoring and capturing suspicious communications between the bot master and bots. Meanwhile, botnets continue to evolve to hide their C&C communication. Recently, a novel type of botnet using image steganography techniques and SNS (Social Network Service) platforms, which is known as image steganography-based botnet or stegobotnet, has emerged to make its C&C communications undetectable by existing botnet detection systems. In stegobotnets, image files used in SNSs carry messages (between the bot master and bots) which are hidden in them by using image steganography techniques. In this paper, we first investigate whether major SNS platforms such as KakaoTalk, Facebook, and Twitter can be suitable for constructing image steganography-based botnets. Next, we construct a part of stegobotnet based on KakaoTalk, and conduct extensive experiments including digital forensic analysis (1) to validate stegobotnet C&C communication can be successful in KakaoTalk and (2) to examine its performance in terms of C&C communication reliability.

Keywords: botnet; Steganography; Social Network Service; stegobotnet; KakaoTalk

1. Introduction

A botnet consists of huge number of bots, which are computing devices (such as PCs or smartphones) with network functions infected by malwares, and bots are under the control of a cyber-attacker (i.e., a bot master) [1,2]. After the botnet is constructed, the bot master periodically sends its commands (such as attack orders or malware upgrades) to bots or collects information from bots. This periodic, persistent communication between the bot master and bots is known as botnet command and control (C&C) communication [2]. For this reason, fast, reliable, and stealthy botnet C&C communications are of critical importance for the bot master to construct and maintain a powerful botnet. Especially, stealthy botnet C&C communication is essential to avoid being detected by botnet detection systems.

Existing botnets exploited the structural and design features of HTTP (Hyper Text Transfer Protocol) or P2P (Peer to Peer) network to hide their C&C communications [3,4]. However, HTTP-based botnets can be captured by monitoring suspicious traffics that persistently access a C&C server due to their centralized structure [5–7]. For P2P-based botnets, various detection mechanisms using behavior-based detection or machine learning techniques have been proposed [8,9].

To improve the stealthiness of botnet C&C communication, a novel type of stealthy botnet using image steganography techniques was introduced, which is known as steganography-based botnet (or stegobotnet in short) [10–12]. The stegobotnet hides its botnet C&C communication messages

into cover mediums such as image files by using steganography techniques, and thus botnet C&C communications look innocuous and unsuspicious to traditional botnet detection systems that do not have defense mechanism against steganography techniques.

Recently, as the popularity of SNS (Social Network Service) and smartphones increases, researchers have studied on SNS-based stegobotnets which are constructed in popular SNS platforms such as Facebook and Twitter [10,11,13]. The SNS-based stegobotnet has a couple of advantages over existing botnets. First, it hides the existence of its C&C communications by the favor of sophisticated steganography techniques. Second, it separates direct connections between a bot master and bots by locating a SNS server in the middle of the bot master and bots. Third, it can attract and infect numerous smartphone users who actively use SNSs by downloading image files (pictures) in which malicious messages may be hidden by the bot master.

Meanwhile, according to our survey, we observed that no researchers have studied on stegobotnets based on mobile SNS messengers although they are one of popular SNS platforms in these days. By this motivation, in this paper, we construct a part of (innocuous) stegobotnet based on KakaoTalk [14], which is the most popular mobile SNS messenger in Republic of Korea, conduct extensive experiments on it, and report performance analysis results in terms of stegobotnet C&C communication reliability. Based on our experiments, we validated that stegobotnets can be constructed in KakaoTalk Openchat, and C&C message duplication methods can highly improve the reliability of stegobotnet C&C communications. To the best of our knowledge, this paper is the first study on the SNS messenger-based stegobotnet, and we hope our study can serve as a good starter to motivate more research in the future to defend against potential cyber threats by stegobotnets.

The rest of this paper is organized as follows. In Section 2, we introduce the general structure and entities of botnet and stegobotnet, and overview existing studies. In Section 3, we investigate seven popular SNSs to examine whether they can be suitable for stegobotnet platforms. In Section 4, we conduct practical experiments based on the KakaoTalk mobile messenger and report our experiment results. Finally, we conclude in Section 5.

2. Background and Related Works

In this section, we first overview the basic entities, structure, and representative C&C communication methods of a typical botnet. Next, we introduce a novel steganography-based botnet (stegobotnet) and review existing studies on the stegobotnet.

2.1. Overview of a Typical Botnet

2.1.1. Brief Description of Basic Entities and Structure of Botnet

As shown in Figure 1, the basic entities of a typical botnet are bot master, C&C server, and bot, and persistent communications among botnet entities are called botnet C&C communication (C&C communication or botnet communication in short) [1,2,15]. The bot master is an attacker who controls all entities of the botnet and launch cyber-attacks by sending malicious commands to the bots through the C&C Server to which a large number of bots are directly connected. In general, bots are directly connected to the C&C server, not the bot master, and C&C server propagates a bot master's commands to bots; this botnet structure can be simplified such that the bot master and the C&C server are combined at the same location, and in this paper we consider such botnet structure for simplicity. Especially, to construct powerful botnets, fast, reliable, and stealthy botnet C&C communication is of critical importance, since when it is not guaranteed, botnet entities cannot cooperate each other and thus achieving its goals becomes impossible.

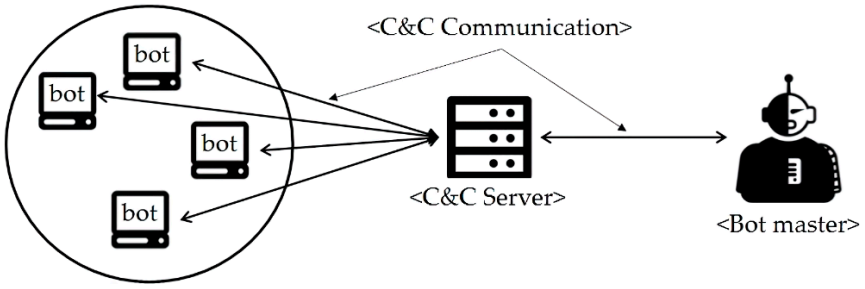


Figure 1. Basic entities of botnet: bots, command and control (C&C) server, and bot master.

2.1.2. Existing Botnet C&C Communication Methods

Stealthy botnet C&C communication is necessary to avoid being detected by botnet detection systems. We explain how two representative type of botnets (HTTP-based botnet and P2P-based botnet) try to hide their botnet C&C communications as follows.

First, in HTTP-based botnets, botnet C&C communication messages are encapsulated into HTTP (Hyper Text Transfer Protocol) packets which look normal in WWW and thus look unsuspicious to security systems [1,16–18]. In this approach, due to its centralized botnet structure, all bots persistently access C&C server to receive the bot master's commands and report the result of the command execution to the bot master. Consequently, HTTP-based botnets can be captured by monitoring and analyzing such persistent HTTP packets to a certain server [6,7]. Examples of HTTP-based botnets include Clickbot [16], Rustock [17], and BlackEnergy [18].

Next, P2P-based botnets with a decentralized botnet structure are proposed to overcome the above-mentioned weakness of HTTP-based botnets due to its centralized botnet structure [4,15]. In this approach, botnet C&C communication messages are routed in somewhat random ways among the botnet entities in P2P network services such as BitTorrent [19] and Gnutella [20]; for example, when an intermediate node in a routing path (from a bot mater to a certain bot) becomes unavailable, alternate routes can be easily set up thanks to the P2P network structure. Thus, such random routes make it difficult for existing HTTP-botnet detection systems to capture them. To detect P2P-based botnets, behavior-based detection or machine learning-based detection are proposed [8,9]. Examples of P2P-based botnets are Slapper [21], Sinit [22], and Nugache [23].

2.2. A Novel Stealthy Botnet: Steganography-Based Botnet (Stegobotnet)

As we briefly mentioned in Section 1, a novel steganography-based botnet (stegobotnet) has been proposed to improve the stealthiness of botnet C&C communication by hiding C&C communication messeges. In stegobotnets, botnet C&C communications are hidden into cover mediums such as image files or video files used in the networks [10,12].

Especially, as the popularity of SNS (Social Network Service) and smartphones increase, researchers studied on *SNS-based stegobotnets* which are constructed in popular SNS platforms such as Facebook and Twitter [11,13]. The SNS-based stegobotnet has a couple of advantages over existing botnets. First, it hides the existence of its C&C communications by the favor of sophisticated steganography techniques. Second, it separates direct connections between a bot master and bots by locating a SNS server between a bot master and bots. Third, it can attract and infect numerous smartphone users who actively use SNSs by downloading images and video clips in which malicious messages may be hidden by a bot master.

Figure 2 shows a general structure of stegobotnet that is constructed in a SNS platform (e.g., KakaoTalk or Facebook) and Table 1 shows new terms (stegobotnet, stegobot-master, stegobot, stego-image, and stego-message) in the stegobotnet with corresponding terms used in the traditional botnet.

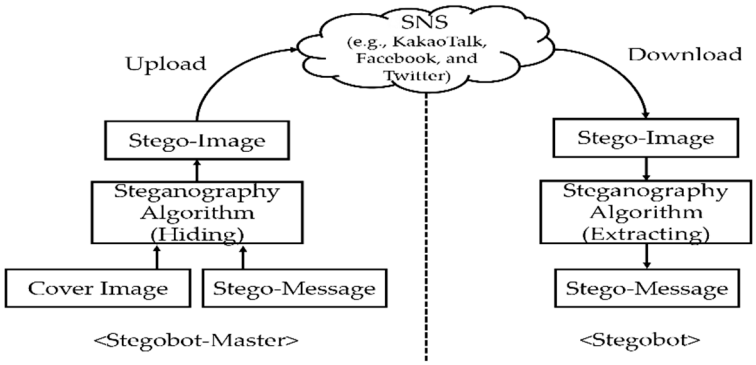


Figure 2. A general structure of stegobotnet based on a SNS (Social Network Service) platform.

Table 1. Basic terms used in stegobotnet.

Terms	Description	Terms in Botnet
stegobotnet	A steganography-based botnet	botnet
stegobot-master	An attacker who controls stegobotnet	bot master
stegobot	A victim device controlled by stegobot-master	bot
stego-image	image file that contains a stego-message stealthily	none
stego-message	messages hidden in stego-image by stegobot-master	C&C communication messages

We now explain how the stegobotnet works in SNSs by using Figure 2. In the stegobotnet, stegobot-master (bot master) first creates a stego-image which stego-messages (C&C communication messages) are hidden by using steganography algorithms. Next, the stegobot-master posts (uploads) the stego-image to an SNS platform which has a lot of SNS members. When stegobots (bots) access the SNS and download the stego-image which look normal to them, malwares in the stegobots extract botnet commands from the stego-image and then conduct malicious behaviors based on the botnet commands.

2.3. Existing Studies on Stegobotnets

Singh et al. [12] presented a botnet that uses spam emails for botnet C&C communication. Most email provider use spam email extraction functions that automatically extract spam emails from entire emails in their email servers. Specifically, a bot master first hides malicious commands in spam emails by using steganography techniques send them to bots. When spam emails are stored in bots' spam email boxes by the spam email extraction functions, malwares installed in bots can search spam emails in the spam email boxes and then extract commands hidden in the spam emails.

Nagaraja et al. [10] firstly proposed the concept of image steganography-based botnet for stealthy botnet C&C communications. They constructed image steganography-based botnet on Flicker which is a well-known social network service, and conducted experiments on how well image files in which command messages are hidden by image steganography techniques are routed from a C&C server to bots.

Compagno et al. [11] proposed a steganography-based botnet in Facebook and Google Plus. They used Unicode steganography technique in web pages. For example, this technique uses the fact that messages after non-readable character like separator "|" are not read in web browsers, and thus can be hidden in web pages. Based on their experiment results, when their steganography-based botnet propagates hidden command messages on Facebook and Google Plus, around 75% of the bots received those messages after 72 h.

Pantic et al. [13] demonstrated the feasibility of botnet C&C communication by using their steganography system on Twitter. Their botnet creates stego-messages and distribute them on Twitter.

In addition, they developed a user account generation system that can automatically create user accounts on Twitter. If an attacker account is not available, this system creates another account so that the bot master consistently propagates bot commands to the bots.

According to our survey, we observed that no researchers have studied on stegobotnets based on mobile SNS messengers although they are one of popular SNS platforms in these days. By this motivation, in this paper, we will study a stegobotnet based on KakaoTalk, which is the most popular mobile SNS messenger in Republic of Korea, conduct extensive experiments on it, and report performance analysis results in terms of stegobotnet C&C communication reliability.

3. Investigation of the Suitability of Popular SNSs for Stegobotnet Platform

In this section, we first investigate a couple of popular SNSs to examine whether they are suitable for Stegobotnet platform, provide our investigation results, and describe an attack scenario based on KakaoTalk Openchat which looks very suitable for Stegobotnet platform.

3.1. Suitability Investigation Procedures and Results

To construct an effective image-based Stegobotnet in an SNS, the SNS must have some basic and essential features as the followings:

1. The number of active members in the SNS is very high.
2. Users can upload image files and the image files can be downloaded by many users.
3. Hidden message in an image file should not be modified when the image file is uploaded and downloaded in the SNS

Our initial suitability investigation proceeded as follows:

First, we selected seven popular SNSs, which satisfy the first two features, such as KakaoTalk, Naver Blog, Naver band, Daum Tistory, Facebook, Twitter, and Instagram, as shown in Table 1. First four SNSs are very popular in Republic of Korea, and the rest of them are well-known globally.

Next, we examine whether the feature (3) is satisfied in part for the seven SNSs as follows. If an image file is modified in part when it is uploaded and then downloaded, we are not sure if the hidden message of the image file by steganography technique will be modified. Meanwhile, if an image file is not modified at all (i.e., even one bit of information is not changed), we can claim that the image file will have the same hidden message. Based on this rationale, we investigated whether the image file is changed when it is uploaded and downloaded as the following steps:

- We prepared three sample image files (file size: 34 KB and file format: JPEG, BMP, and PNG).
- We posted those sample images on the seven SNSs by using a smartphone A; for KakaoTalk, Naver Blog, Naver Band, Twitter, and Instagram, we used their smartphone apps, and for the remaining SNSs, we posted image files by using their websites' upload functions.
- We downloaded the sample images by using a smartphone B.
- We examined whether the sample image files are changed, and reported details if changed in terms of hash value, file size, and file format (extension).

The investigation results are shown in Table 2. First, the hash values of image files in KakaoTalk and Naver Blog did not change, which means that image files in KakaoTalk and Naver Blog maintain as their original forms and thus hidden messages in the image files will not change. Meanwhile, for other SNSs, the hash values of image files changed, which means that hidden messages may get damaged during upload and download operations. Specifically, the size of all image files decreased slightly, and image format (extension) changed from PNG to JPEG in Naver Band, Facebook, Twitter, and Instagram except Daum Tistory, while BMP and JPEG are handled correctly in all SNSs.

Table 2. Suitability investigation results for seven popular SNSs.

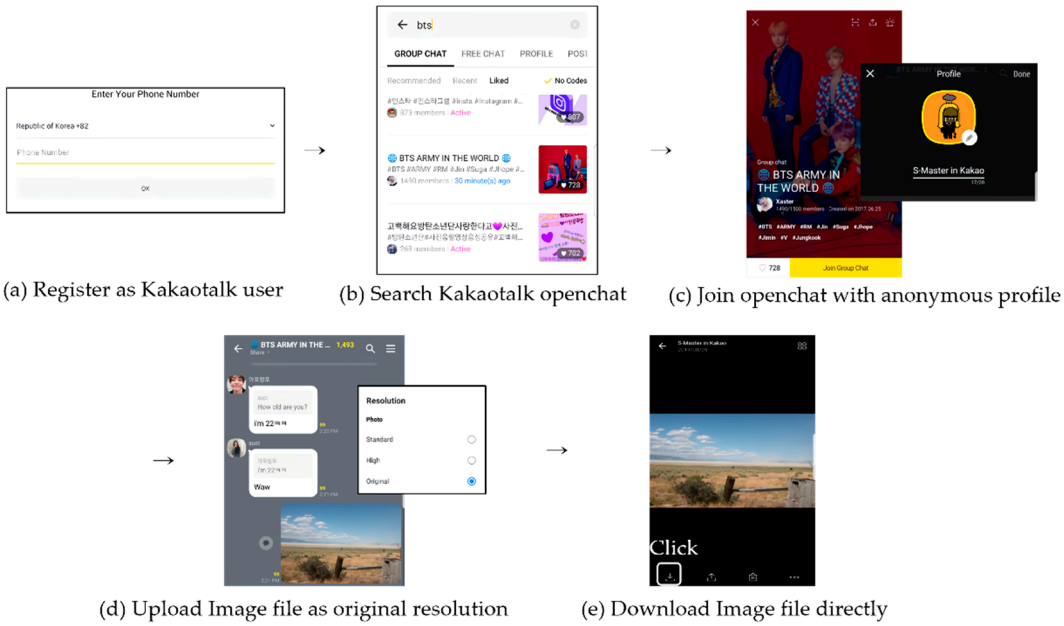
SNS	File Size	File Format	Hash Value	Suitability
KakaoTalk	Unchanged	Unchanged	Unchanged	Suitable
Naver Blog	Unchanged	Unchanged	Unchanged	Suitable
Naver Band	34 KB → 20 KB	PNG →JPEG	Changed	Not suitable
Daum Tistory	34 KB → 30 KB	Unchanged	Changed	Not suitable
Facebook	34 KB → 27 KB	PNG →JPEG	Changed	Not suitable
Twitter	34 KB → 24 KB	PNG →JPEG	Changed	Not suitable
Instagram	34 KB → 33K B	PNG →JPEG	Changed	Not suitable

Based on our investigation results, we found that KakaoTalk provides the most attractive environment for Stegobotnet deployment among the seven SNSs in that it satisfies all three basic features we described above. For the rest of our paper, we will use KakaoTalk for further investigations including its performance analysis in Section 4.

3.2. Brief Introduction to KakaoTalk Openchat: Registration, Chatroom Participation, and Image File Upload/Download

KakaoTalk is the most famous SNS messenger in Republic of Korea and started its service in March 2010. KakaoTalk has around 50 million members globally, although most of them are Korean. KakaoTalk provides three chat modes such as one-to-one chat, group chat, and openchat. Unlike one-to-one and group chat, a user can participate in an openchat room of interest anonymously without disclosing his/her identity, and up to 1500 members can participate in one openchat room.

We now explain how to register to KakaoTalk, search and participate in an openchat participation, and upload/download image files in an openchat room in turn (see Figure 3). First, to register to KakaoTalk (or to create a KakaoTalk ID), a user must provide his/her mobile phone number because KakaoTalk uses it for user authentication; he/she may get multiple KakaoTalk IDs if he/she has multiple phone numbers (Figure 3a). Next, once a user creates a KakaoTalk ID, the user can search currently available openchat rooms by using topic search function, and then participate in an openchat room of interest if there is no participation password which can be set up by the openchat room owner; Figure 3c shows an openchat room that looks like a fan club of a worldwide famous boyband group "BTS", and we can see the number of participants is around 1500 and a participant ID "S-Master" which means "Stegobot-Master" that we created intentionally (see Figure 3b,c). Last, participants in an openchat room can freely upload their image files and download image files which are someone posted. When uploading an image file, three upload options are available such as Standard, High, and Original. Among them, when the original mode is selected, an image file will be uploaded as its original form without compression; for this reason, attackers will use this original option to avoid damages of stego-image files during stegobotnet communication (see Figure 3d,e).



(a) Register as Kakaotalk user (b) Search Kakaotalk openchat (c) Join openchat with anonymous profile

(d) Upload Image file as original resolution (e) Download Image file directly

Figure 3. Procedures of Kakaotalk registration, Kakaotalk Openchat participation, and Image File Upload and Download in a KakaoTalk openchat room.

3.3. An Attack Scenario Using Stegobotnet in KakaoTalk Openchat

To help you understand how an attacker (stegobot-master) can construct stegobotnet and launch cyberattacks based on the stegobotnet, we describe a simple but possible attack scenario based on KakaoTalk Openchat as the following steps (S1–S8) as shown in Figure 4. In this scenario, stegobotnet is constructed by S1–S4, stegobotnet C&C communication is performed by S5–S7, and various cyberattacks can be launched in S8.

S1. Attacker creates a legitimate account (ID) in KakaoTalk by using a valid mobile phone number; the attacker may use a stolen mobile phone number to hide his identity.
S2. Attacker logs in to KakaoTalk with ID
S3. Attacker searches and participate in an openchat room; attractive openchat rooms will have a huge number of participants who can be stegobots (victims) by attacks; as we explained earlier, the maximum number of openchat participants is 1500.
S4. Attacker starts infecting participants' smartphones with malwares as much as possible by exploiting vulnerabilities existing in participants' smartphones; for this purpose, the attacker may use hacking tools to discover such vulnerabilities or post some malicious files or ULR links in the chatroom to force participants to install malwares to their smartphones [24].
S5. Participants (stegobots) are infected by malwares that includes steganography embedding and extraction functions.
S6. Attacker (stegobot-master) posts stego-image files in which malicious messages are hidden to the openchat room.
S7. When participants (stegobots) read or click the posted stego-image files, the files will be downloaded to their smartphones and then malwares installed in participants' smartphones will extract hidden messages (stego-messages) from the downloaded stego-images.
S8. Malwares will for participants' smartphones (stegobots) to perform cyberattacks based on the extracted hidden messages such as DDoS (Distributed Denial of Service) attacks.

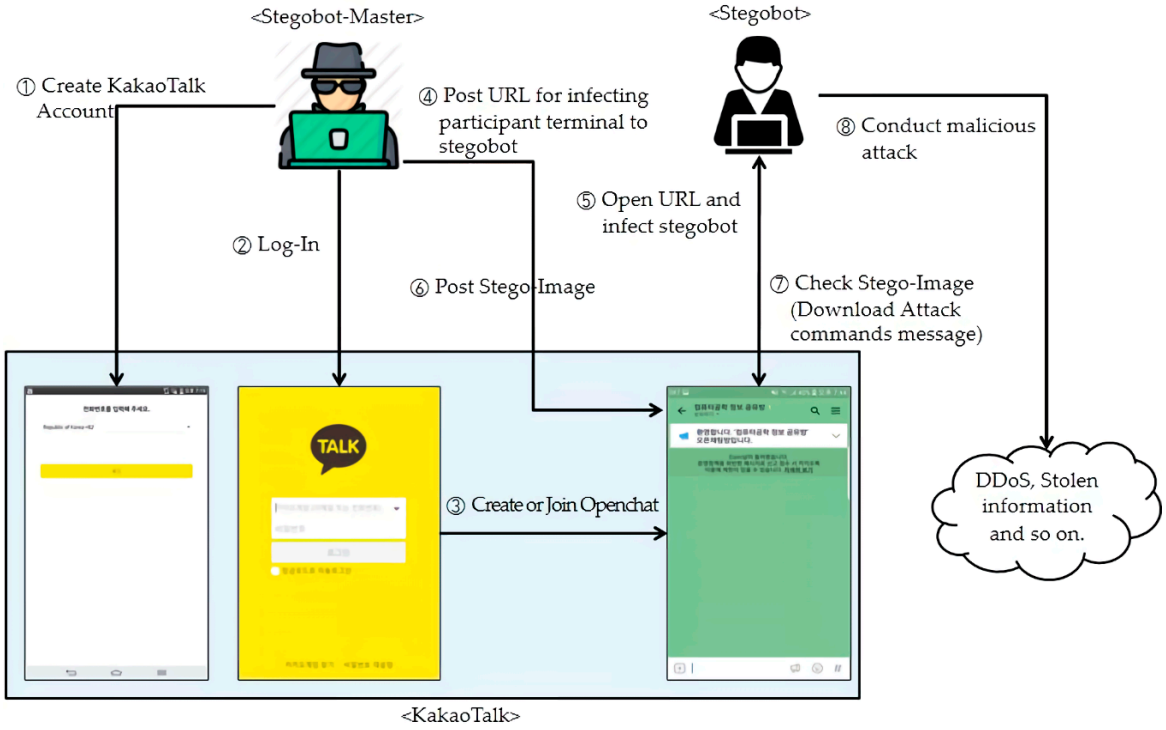


Figure 4. Stegobotnet construction in KakaoTalk Openchat and attack scenario. DDOS: Distributed Denial of Service.

4. Experiments and Result Analysis

In this section, we conduct two kinds of experiments to validate that stegobotnet communication can be possible in the KakaoTalk (Experiment 1) and to analyze the performance of stegobonet communication in the KakaoTalk Openchat by using various performance metrics (Experiment 2).

4.1. Experiment 1: Validating that Stegobotnet Communication Can Be Possible in KakaoTalk

The main goal of conducting Experiment 1 is to validate that stegobotnet C&C communication can be successful in KakaoTalk by examining the following:

- Whether a stego-image file is successfully delivered from a stegobot-master to stegobots
- Whether a hidden message (stego-message) of the stego-image file is extracted successfully when stegobots receive the stego-image file

Based on the experimental environment in Table 3, we conducted Experiment 1 as follows. Two Samsung Galaxy S7 smartphones (S_{MASTER} and S_{BOT}) are used to run KakaoTalk mobile messenger application; S_{MASTER} and S_{BOT} are used for a stegobotmater and a stegobot, respectively. For a cover image file, we used one 1.5 MB JPEG file. With the cover image file, we generated a stego-image in which a secret message "stegobotnettest" was hidden by OpenStego ver. 0.73 [25], which is a well-known and free steganography tool; the file extension of the created stego-image file is .png, because OpenStego produces a PNG image file as output file after processing the input cover image file.

Table 3. Experimental environment of Experiment 1

Elements	Value
Smartphone Model	Two Samsung Galaxy S7 (S_{MASTER}: stegobot-master, S_{BOT}: stegobot)
Android OS version	8.0.0 (Android Oreo)
KakaoTalk version	8.1.5. (released in Dec. 8, 2018)
Cover image file	1.5 MB JPEG file
Steganography tool	OpenStego ver. 0.7.3 (Freeware)

Figure 5 shows the whole steps (a–c) of conducting Experiment 1. First, by using S_{MASTER}, we created a chatroom and then invited a KakaoTalk user (S_{BOT}) (a). After that, we (S_{MASTER}) posted the stego-image file to the chatroom and then downloaded it by using S_{BOT} (b). Finally, we checked if the hidden message "*stegobotnettest*" is extracted successfully from the downloaded image file by using OpenStego (c). As we can see at c in Figure 5, the extracted message was the exactly same with the original hidden message. Consequently, based on this experiment result, we validated that the stegobotnet C&C communication between a stegobot-mater (S_{MASTER}) and stegobots (S_{BOT}) can be successful in the KakaoTalk SNS messenger.

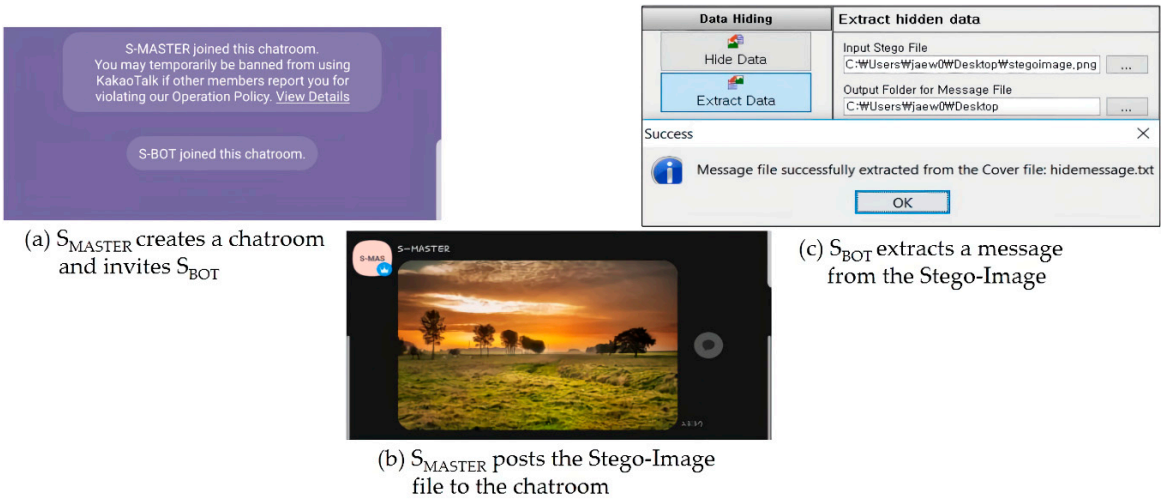
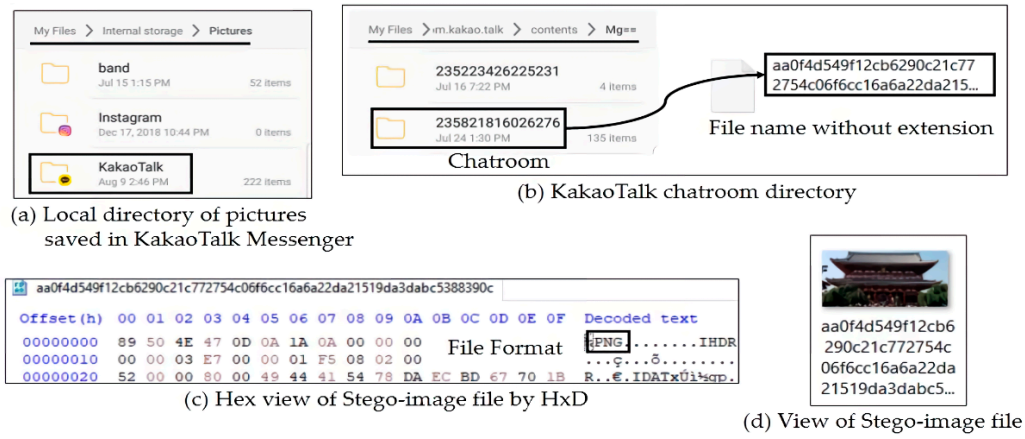


Figure 5. Whole experiment steps of Experiment 1 and validation of successful stegobotnet C&C communication.

In addition, we conducted digital forensic analysis to the smartphone (to $\mathbf{S_{BOT}}$) and report a couple of things that we found as the following:

- First, when a receiver (S_{BOT}) clicks a posted image file and downloads it, the image file is stored at the S_{BOT}'s local directory ".../Phone/Pictures/KakaoTalk/" as shown in Figure 6a.
- Second, the stego-image file delivered to S_{BOT} was also stored at the S_{BOT}'s local directory ".../Phone/AndroidData/com.kakao.talk/contents/Mg==/" as well as ".../Phone/Pictures/KakaoTalk/" as shown in Figure 6b. Interestingly, even if we did not click and save the image file in a chatroom, the image file is automatically downloaded to the former directory ".../Phone/AndroidData/com.kakao.talk/contents/Mg==/". In the figure, the folder 23582186026276 is corresponding to the KakaoTalk chatroom that we created, and the file aa0f4 is the stego-image file that we generated. In addition, as shown in Figure 6c, we can see that it is a PNG image file when we view it by hex editor HxD [26]. When we add .png file extension after its filename, we can see that it is the same stego-image file (see Figure 6d).

- Last, for KakoTalk version 8.4.7 released in July 10, 2019, image files are not automatically downloaded unless they are not clicked and saved by KakaoTalk users. We will further explain about this in Experiment 2.



(a) Local directory of pictures saved in KakaoTalk Messenger

(b) KakaoTalk chatroom directory

(c) Hex view of Stego-image file by HxD

(d) View of Stego-image file

Figure 6. Results of digital forensic analysis to S_{BOT}; a stego-image file without file extension is stored in $\mathbf{S_{BOT}}$.

4.2. Experiment 2: Analyzing the Performance of Stegobotnet Communication in KakaoTalk Openchat

4.2.1. Experimental Goal, Performance Metrics, and Methods

The goal of conducting Experiment 2 is to analyze the performance of the stegobotnet constructed in the KakaoTalk Openchat in terms of botnet communication reliability. To this end, we use the following two performance metrics:

- ***Stego-Image Delivery Success Rate ($DSR_{Stego\text{-}Image}$)*:** This metric indicates how successfully stego-images are delivered from a stegobot-master to stegobots, and is obtained as

$$DSR_{Stego-Image}(\%) = \frac{Num.\ of\ Stegobots\ who\ successfully\ recieved\ Stego-Images}{Num.\ of\ Stego-Images\ posted\ \times\ Num.\ of\ Stegobots} \times 100. \quad (1)$$

- ***Stego-Message Delivery Success Rate ($DSR_{Stego\text{-}Message}$)*:** This metric indicates how successfully (reliably) hidden messages in stego-images are delivered from a stegobot-master to stegobots, and is calculated by

$$DSR_{Stego-Messege}(\%) = \frac{Num.\ of\ Stegobots\ who\ successfully\ recieved\ Stego-Messeges}{Num.\ of\ Stego-Messege\ posted\ \times\ Num.\ of\ Stegobots} \times 100. \quad (2)$$

In our experiments, we assume that stegobot-master wants to send a certain number of hidden messages (stego-messages) and one stego-message can be hidden into more than one stego-images by message duplication approach for reliable delivery. Consequently, $\boldsymbol{DSR_{Stego\text{-}Message}}$ can be different with $\boldsymbol{DSR_{Stego\text{-}Image}}$.

Based on the experimental environment in Table 4, we conduct Experiment 2 to examine (1) how reliably stego-messages can be delivered to stegobots and (2) how duplicating stego-messages can improve the performance of stegobotnet C&C communication in terms of stego-message delivery reliability. Unlike Experiment 1, we (S_{MASTER}) joined an existing KakaoTalk chatroom where 11 members including us (S_{MASTER}) are participating. All participants are our colleagues, but we did not inform them in advance about our experiments. We prepared 20 cover image files to attract participants' interests and thus make them click and download those image files to their smartphones (see Figure 7).

For each experiment, we will explain experimental methods, report experiment results, and provide our analysis on the results in detail.

Table 4. Experimental environment of Experiment 2.

Elements	Value
KakaoTalk version	8.4.7. (released in July 10, 2019)
Numumber of participants in KakaoTalk chatroom	11 ($\mathbf{S_{MASTER}}$ and 10 stegobots $\mathbf{S_{BOT1}}$, $\mathbf{S_{BOT2}}$, ... , $\mathbf{S_{BOT10}}$)
Experiment duration time	48 h
Cover image files	20 JPEG files (various file size in [581 KB, 2.8 MB])
Steganography tool	OpenStego ver. 0.7.3 (Freeware)

(**a**) Tokyo Tower (**b**) Nagoya Castle (**c**) Hommaru Palace

Figure 7. Sample cover image files used to generate stego-images in Experiment 2.

In this experiment, we assume that the stegobot-master wants to send 10 stego-messages to stegobots. To this end, by using 20 cover image files and OpenStego, we made two group of sample stego-image files as the following:

- Sample Image Group 1 (ND: Not Duplicated): We made 10 stego-images with different hidden messages such as such as "stegobomessage1", "stegobomessage2", ... , "stegobomessage10". That is, total number of stego-messages in this group is 10.
- Sample Image Group 2 (D: Duplicated): We made 10 pair of stego-images and each pair of stego-images has the same hidden message; that is, total number of stego-messages in this group is 10.

For each sample image group, we (S_{MASTER}) intermittently uploaded stego-images to the KakaoTalk chatroom for 5 h, and then waited for 48 h. After that, we collected data from all participants' smartphones, and conducted performance analysis by using our performance metrics; we discussed in Experiment 1 about the exact location where stego-images are stored in Android smartphones.

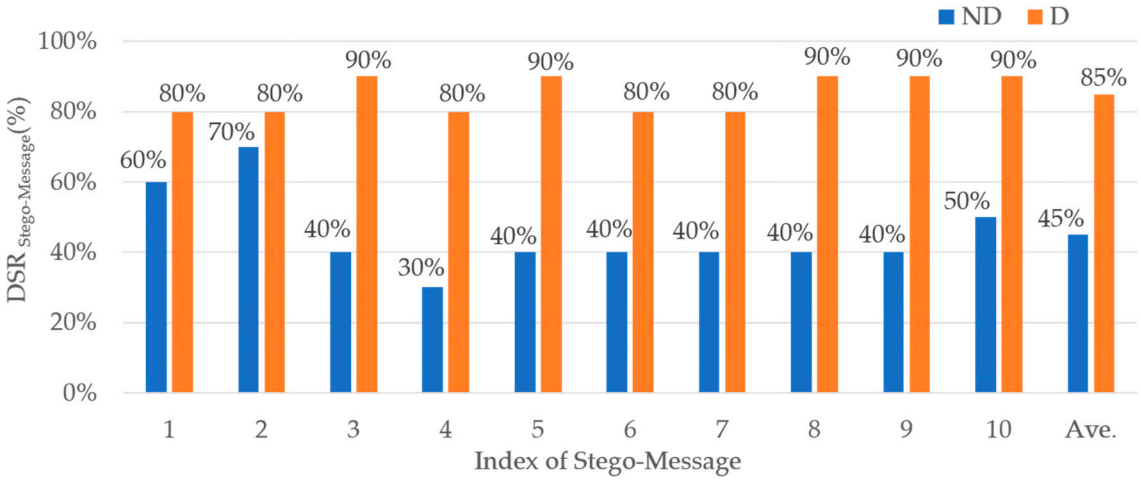
4.2.2. Experiment Result and Analysis

We explain our experiment results as follows (see Figure 8).

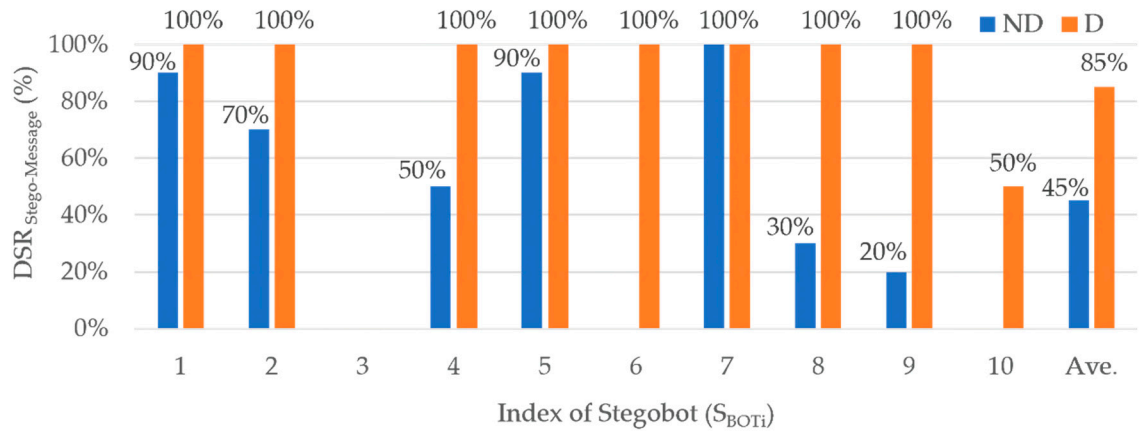
First, we report experiment results for Sample Image Group 1 (Not Duplication—blur-colored graphs in the figure). As shown in Figure 8a, only 45% of stego-image files were delivered successfully from the stegobot-master to 10 stegobots. That is, $DSR_{Stego\text{-}Image}$ (Sample Image Group 1) = 45%. In other words, 55% of stego-image files were not delivered to stegobots for 48 h since the stegobot master posted the stego-image files to the KakaoTalk chatroom. Specifically, as we can see in Figure 8b, three stegobots (S_{BOT3}, S_{BOT6}, and S_{BOT10}) did not click and download stego-image files at all (i.e., $DSR_{Stego\text{-}Image}$ = 0%). Except the three stegobots, other stegobots (S_{BOT1}, S_{BOT2}, S_{BOT4}, S_{BOT5}, S_{BOT7}, S_{BOT8}, S_{BOT9}, and S_{BOT10}) downloaded around 64% of stego-image files in average. In this experiment of using Sample Image Group 1, $DSR_{Stego\text{-}Image} = DSR_{Stego\text{-}Message}$ because all stego-images have different hidden messages.

Next, for Sample Image Group 2 (Duplicated—orange-colored graphs in the figure), 85% of stego-image files were delivered successfully from the stegobot-master to 10 stegobots (see Figure 8a). That is, $DSR_{Stego\text{-}Message}$ (Sample Image Group 1) = 85%; Unlike Sample Image Group 2, we use $DSR_{Stego\text{-}Message}$ because $DSR_{Stego\text{-}Image}$ can be different with $DSR_{Stego\text{-}Message}$ due to the duplication method and we want to see how stego-message duplication method can improve $DSR_{Stego\text{-}Message}$ compared with the case when the duplication method is not used. By duplicating stego-messages and sending them to stegobots, the stegobot-master could improve the stego-message delivery success rate $DSR_{Stego\text{-}Messgage}$ by 89% compared when the duplication method is not used. This is because the successful stego-message delivery from a bot master to a bot is guaranteed when at least one of two duplicated stego-messages is successfully delivered to the stegobot. Interestingly, it is very impressive that 8 stegobots received 100% of all stego-messages in Sample Image Group 2, while only one stegobot received all stego-messages (see Figure 8b).

Consequently, based on our experiment results of Experiment 2, we validated that stego-message duplication methods can highly improve the reliability of stegobotnet C&C communications by increasing successful stego-message delivery rate.



(**a**) $DSR_{Stego\text{-}Message}$ for each Stego-Message



(**b**) $DSR_{Stego\text{-}Message}$ for each stegobot

Figure 8. Results of Experiment 2 (ND: Not duplicated, D: Duplicated).

5. Conclusions and Future Works

As the popularity of SNS and smartphones grows, novel SNS-based stegobotnets have emerged in order to better conceal their C&C communications by the favor of sophisticated steganography

techniques and infect mobile smartphone devices whose owners actively use SNSs by downloading image files and video clips in which malicious messages may be hidden by a bot master.

In this paper, to examine the seriousness of stegobotnets that are constructed over SNS messengers, we constructed a part of stegobotnets based on the KakaoTalk mobile messenger, conducted extensive experiments based on it, and analyzed experiment results in terms of stegobotnet C&C communication reliability. According to our investigation and experiment results, we reported the following. First, stegobotnets can be constructed in SNSs where hidden messages in stego-image files are not damaged, and the KakaoTalk mobile messenger (version 8.4.7) is one of such SNSs. Second, stego-message duplication methods can highly improve the stego-message delivery success rate from a stegobot-master to stegobots.

In our future work, we will extend our study as follows. First, we will study effective defense methods which can detect the existence of stegobotnet and stego-images in both smartphones and a SNS server. Second, we will study backtracking methods which can find the stegobot-master by monitoring the stegobotnet C&C communication. Third, we would like to further study various stego-message duplication methods for reliable botnet C&C communications to better understand the characteristics of stegobotnet C&C communications and devise defense methods against them.

Author Contributions: Formal analysis, J.J. and Y.C.; Investigation, J.J. and Y.C.; Methodology, Y.C.; Software, J.J.; Validation, J.J. and Y.C.; Supervision, Y.C.; Visualization, J.J.; Writing—original draft, J.J.; Writing—review & editing, J.J. and Y.C.

Funding: This article received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khattak, S.; Ramay, N.; Khan, K.; Syed, A.; Khayam, S. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 898–924. [CrossRef]
2. Vormayr, G.; Ramay, K.; Fabini, J. Botnet communication patterns. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2768–2796. [CrossRef]
3. Yang, Z.; Wang, B. A Feature Extraction Method for P2P Botnet Detection Using Graphic Symmetry Concept. *Symmetry* **2019**, *11*, 326. [CrossRef]
4. Dittrich, D.; Dietrich, S. P2P as botnet command and control: A deeper insight. In Proceedings of the IEEE 3rd International Conference on Malicious and Unwanted Software (MALWARE), Fairfax, VI, USA, 7–8 October 2008.
5. Acarali, D.; Rajarajan, M.; Komninos, N.; Herwono, I. Survey of approaches and features for the identification of HTTP-based botnet traffic. *J. Netw. Comput. Appl.* **2016**, *76*, 1–15. [CrossRef]
6. Eslahi, M.; Rohmad, S.; Nilsaz, H.; Naseri, M.; Tahir, N.; Hashim, H. Periodicity Classification of HTTP Traffic to Detect HTTP Botnets. In Proceedings of the IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Langkawi, Malaysia, 12–14 April 2015.
7. Zeidanloo, H.; Manaf, A.; Vahdani, P.; Tabatabaei, F.; Zamani, M. Botnet detection based on traffic monitoring. In Proceedings of the IEEE International Conference on Networking and Information Technology, Manila, Philippines, 11–12 June 2010.
8. Saad, S.; Traore, I.; Ghorbani, A.; Sayed, B.; Zhao, D.; Lu, W.; Felix, J.; Hakimian, P. Detecting P2P Botnets through Network Behavior Analysis and Machine Learning. In Proceedings of the IEEE Ninth Annual International Conference on Privacy, Security and Trust, Montreal, QC, Canada, 19–21 July 2011; pp. 174–180.
9. Garg, S.; Singh, A.; Sarje, A.; Peddoju, S. Behaviour analysis of machine learning algorithms for detecting P2P botnets. In Proceedings of the IEEE 15th International Conference on Advanced Computing Technologies (ICACT), Rajampet, India, 21–22 September 2013; pp. 1–4.
10. Nagaraja, S.; Houmansdr, A.; Piyawongwisai, P.; Singh, V.; Agarwal, P.; Borisov, N. Stegobot: A covert social network botnet. In Proceedings of the Information Hiding Conference, Prague, Czech Republic, 18–20 May 2011; pp. 299–313.

11. Compagno, A.; Conti, M.; Lain, D.; Lovisotto, G.; Mancini, L. Boten ELISA: A new novel approach for Botnet C&C in Online Social Networks. In Proceedings of the IEEE Conference on Communications and Network Security, Florence, Italy, 28–30 September 2015; pp. 74–82.
12. Singh, K.; Srivastava, A.; Giffin, J.; Lee, W. Evaluating Email's Feasibility for Botnet Command and Control. In Proceedings of the 38th Annual IEEE/IFIP International Conference on Defendable Systems and Networks, Anchorage, AK, USA, 24–27 June 2008.
13. Pantic, N.; Husain, M. Covert Botnet Command and Control Using Twitter. In Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, 7–11 December 2015.
14. KakaoTalk. Available online: https://www.kakaocorp.com/service/KakaoTalk?lang=en (accessed on 25 June 2019).
15. Bailey, M.; Cooke, E.; Jahanian, F.; Xu, Y.; Karir, M. A Survey of Botnet Technology and Defenses. In Proceedings of the IEEE Cybersecurity Applications & Technology Conference for Homeland Security, Washington, DC, USA, 3–4 March 2009.
16. Daswani, N.; Stoppelman, M. The anatomy of clickbot.A. In Proceedings of the First Conference on the First Workshop on Hot Topics in Understanding Botnets, Berkeley, CA, USA, 10 April 2007.
17. Chiang, K.; Lloyd, L. A case study of the restock rootkit and spam bot. In Proceedings of the First Conference on the First Workshop on Hot Topics in Understanding Botnets, Berkeley, CA, USA, 10 April 2007.
18. Nazario, J. *Blackenergy DDoS Bot Analysis*; Arbor Networks: Chemsford, MA, USA, 2007.
19. Desimone, J.; Johnson, D.; Yuan, B.; Lutz, P. Covert Channel in the BitTorrent Tracker Protocol. In Proceedings of the 2012 International Conference on Security and Management, Las Vegas, NV, USA, 16–19 July 2012.
20. Davis, C.R.; Neville, S.; Fernandez, J.M.; Robert, J.M.; McHugh, J. Structured Peer-to-Peer Overlay Networks: Ideal Botnets Command and Control Infrastructures? In Proceedings of the 13th European Symposium on Research in Computer Security, Malaga, Spain, 6–8 October 2008; pp. 461–480.
21. Arce, I.; Levy, E. An Analysis of the slapper Worm. *IEEE Secur. Priv.* **2003**, *1*, 82–87. [CrossRef]
22. Falliere, N. *Sality: Story of a Peer-to-Peer Viral Network*; Symantec Security Response: Mountain View, CA, USA, 2011.
23. Stover, S.; Dittrich, D.; Hernandez, J.; Dietrich, S. Analysis of the storm and nugache trojans: P2P is here. *USENIX Login* **2007**, *32*, 18–27.
24. Zhou, Y.; JiangConti, X. Dissecting Android Malware: Characterization and Evolution. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 21–23 May 2012.
25. OpenStego. Available online: http://www.openstego.com (accessed on 28 June 2019).
26. HxD. Available online: http://mh-nexus.de/en/hxd (accessed on 13 August 2019).