computers

MDPI

Article

Design and Implementation of Automated Steganography Image-Detection System for the KakaoTalk Instant Messenger

Jun Park and Youngho Cho *

Department of Defense Science (Computer Engineering), Graduate School of Defense Management, Korean National Defense University, Nonsan 33021, Korea; cucujun12@kndu.ac.kr

* Correspondence: youngho@kndu.ac.kr

Received: 29 October 2020; Accepted: 15 December 2020; Published: 18 December 2020

Abstract: As the popularity of social network service (SNS) messengers (such as Telegram, WeChat or KakaoTalk) grows rapidly, cyberattackers and cybercriminals start targeting them, and from various media, we can see numerous cyber incidents that have occurred in the SNS messenger platforms. Especially, according to existing studies, a novel type of botnet, which is the so-called steganography-based botnet (stego-botnet), can be constructed and implemented in SNS chat messengers. In the stego-botnet, by using various steganography techniques, every botnet communication and control (C&C) messages are secretly embedded into multimedia files (such as image or video files) frequently shared in the SNS messenger. As a result, the stego-botnet can hide its malicious messages between a bot master and bots much better than existing botnets by avoiding traditional botnet-detection methods without steganography-detection functions. Meanwhile, existing studies have focused on devising and improving steganography-detection algorithms but no studies conducted automated steganography image-detection system although there are a large amount of SNS chatrooms on the Internet and thus may exist many potential steganography images on those chatrooms which need to be inspected for security. Consequently, in this paper, we propose an automated system that detects steganography image files by collecting and inspecting all image files shared in an SNS chatroom based on open image steganography tools. In addition, we implement our proposed system based on two open steganography tools (Stegano and Cryptosteganography) in the KakaoTalk SNS messenger and show our experimental results that validate our proposed automated detection system work successfully according to our design purposes.

Keywords: image steganography; steganography-based botnet; SNS security; KakaoTalk messenger security; automated steganography image detection

1. Introduction

Recently, the usage of social network service (SNS) applications is growing rapidly owing to the rapid advancement of mobile smartphones and 4G/5G wireless networks technologies. Meanwhile, cyberattackers start targeting smartphones with SNS applications [1–3]. In particular, many recent studies [4–7] report that cyberattackers can construct a stealthy botnet using steganography techniques in SNS instant messengers (SNS IMs) such as WeChat or KakaoTalk, and such novel type of the botnet is known as *steganography-based botnet* or *stego-botnet* [8,9].

According to our extensive survey, most stego-botnets use image steganography techniques [10–13]. In the image stego-botnet constructed in an SNS IM, a bot master sends its command and control (C&C) messages to bots in a stealthy way as follows [14,15]. First, the bot master hides a secret message containing its commands into a plain image file (cover image) by using an image steganography method or tool such as Steghide or Openstego, and shares the image file (stego-image) in an SNS

chatroom with many participants [16,17]. Next, when chatroom participants read and click the shared image file, it gets downloaded to their smartphones. After that, the secret message hidden in the image file is automatically extracted by a bot software (malware) and then cyberattacks are performed according to the extracted secret message (bot command). Stego-botnets are serious, emerging cyber threats in that they can hide their botnet command and control messages into image files and thus can avoid existing botnet monitoring and detection systems since image files look normal to those defense systems [18,19]. The 2018 Fortinet Threat Landscape Report reported that malwares using image steganography to hide malicious payloads in memes were propagated over SNS IMs and that an image stego-botnet (Vawtrak) is included in the list of explosive growth in botnets [20].

Meanwhile, according to our survey, existing studies have focused on devising and improving steganography-detection algorithms but no studies conducted automated steganography image-detection system although there are a large amount of SNS chatrooms on Internet and thus may exist many potential steganography images on those chatrooms which need to be inspected for security. By this motivation, in this study, we propose and devise an automated detection system of steganography image shared in an SNS IM, which has two major components such as automated collection component (ACC) and automated detection component (ADC). Thus, our proposed system automatically collects the entire image files shared in an SNS IM, examines whether each image file hides a secrete steganography message, and displays the examination results. To the best of our knowledge, this is the first study according to the establishment of a steganography detection system in IM. Thus, we hope this study will contribute to lowering security threats in the KakaoTalk environment and further be expended in other IM platforms through advanced studies.

The main contributions of this study can be summarized as the following:

- We proposed an automated detection model that can automatically collect and detect steganography image files shared in SNS IMs.
- We implemented and constructed our proposed model in the KakaoTalk SNS IM platform; for automated detection, we used two open steganography tools (Stegano [21] and Cryptosteganography [22]) to examine whether collected image files from a KakaoTalk Chatroom contains secret hidden messages.
- We show experimental results that validate our proposed automated detection system work successfully according to our design purpose.

The remainder of this paper is organized as follows. In Section 2, we overview traditional botnets, steganography-based botnets, and existing related studies. In Section 3, we propose and design an automated detection system of steganography images shared in the KakaoTalk chatroom. In Section 4, we implement our proposed system in the KakaoTalk SNS messenger and conduct experiments to show our proposed system work properly according to our design purpose. Finally, we conclude with our future research directions in Section 5.

2. Background and Related Works

Before proceeding with the above research, it is necessary to have a good understanding of existing botnets and steganography-based botnets. Therefore, this section overviews the existing bonnets and the response system, then introduces the steganography-based botnets, and then the steganography-based botnet in the KakaoTalk environment.

2.1. Overveiw of Traditional Botnets

Botnet refers to a large number of networked devices that are infected by malware and are under the control of the bot master [23]. Botnet is one of the most serious network threats in which the bot master with the authority to control the robots remotely controls infected hosts to carry out various cyberattacks, including DDoS, Ad-ware, Spyware, spam transmission, and illegal information gathering [23,24]. The existing botnet type is shown in Figure 1 [25]. The early botnets were mainly IRC

botnets using the characteristics of the Internet Relay Chat (IRC) as shown in Figure 1a, whose structure is flexible and widely used. However, the disadvantage of IRC botnets is that they are easy to detect, which led to the appearance of HTTP-based botnets as shown in Figure 1b. Detecting botnets using HTTP is even more difficult because botnet traffic is hidden in a large amount of normal HTTP traffic [26,27]. However, since the traffic generated by botnets is different from normal HTTP traffic, it is possible to detect them by using filters that distinguish them [26–28]. In addition, HTTP botnets have a centralized structure in which C&C servers are responsible for both command and control, such as IRC botnets. Thus, they have the disadvantage of being neutralized when blocking C&C servers. To compensate for these shortcomings, P2P-evolved botnets have emerged, each of which becomes a C&C server. The structure of the P2P botnet is shown in Figure 1c, and all bots act as C&C servers [29,30]. This is a method of performing commands and controls in a distributed rather than centralized way, so even if one P2P botnet server is discovered, a botnet can be operated with other servers without being neutralized. However, for P2P botnets, the size of the supported groups (hosts) is much smaller than the existing centralized botnets. Centralized has thousands of hosts, but only a few dozen in the P2P model [29,30]. In addition, studies have suggested that P2P-based botnets can be detected through action-based or machine-learning-based detection methods [30], leading to the emergence of more advanced botnets to respond to this. Since then, among the more advanced botnets, botnets using SNS have appeared, and studies have been reported that botnets can be built on SNS messengers [31]. Some of these botnets use images to build botnets. When using images, there are botnets that use steganography technology, which are called steganography base botnets [32].

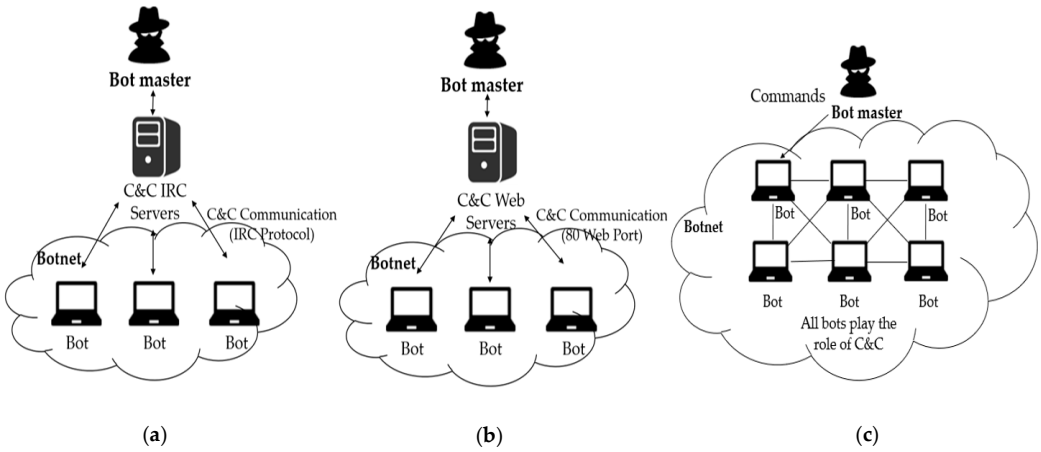

Figure 1. Types of existing botnets. (**a**) IRC Botnet; (**b**) HTTP Botnet; (**c**) P2P Botnet.

2.2. Steganography-Based Botnet (Stego-Botnet)

Unlike traditional botnets introduced in Section 2.1, a novel type of steganography-based botnet was proposed to improve the confidentiality of botnet C&C communication by hiding C&C communication messages to overcome the weaknesses of existing botnets. Steganography-based botnet (or Stego-botnet) uses steganography technology to hide the communication itself between the bot master and the bot so that it is not detected [31]. Figure 2 represents Stego-botnet, which uses IM/SNS as a relay server. The existing botnet communicates directly with the C&C server, bot master, and bots. However, Stego-botnet, which uses SNS, makes it more difficult to detect botnet by separating the bot master and bots [33]. The Bot master initially builds a botnet by utilizing known vulnerabilities held by PCs, smartphones, and IoT devices or by attacking them through social engineering. Subsequently, the attack command message posts or shares the hidden image on SNS, accesses the image posted by the bots, downloads it, and receives the attack order to perform it [34,35]. Because it hides messages with steganography technology during C&C communication, it is no different from normal messages, so it can further avoid detection. Because it hides malicious messages in images and multimedia files that are distributed naturally on SNS, it is difficult to detect them with existing defense

systems. Nagaraja et al. [32] studied image steganography-based hidden communication model in SNS environment and Hiney et al. [36] focused on Facebook during the process of compressing image files that occur during communication to identify conditions where hidden messages are not destroyed.

Figure 2. Covert communication based on steganography-based Botnet in the social network service (SNS) environment.

2.3. Existing Studies Related to Stego-Botnets

First, there are a couple of studies on steganography-based botnets or covert channels in SNS services. Nagaraja et al. [32] first studied the possibility of establishing a steganography-based botnet in 2011. In 2019, Jeon and Cho [37] constructed and evaluated the performance of an image steganography-based botnet at the Kakaotalk SNS messenger which is the most popular in South Korea and has around 50 million users worldwide. KakaoTalk offers three chat modes: one-on-one chat, group chat, and open chat. In the case of open chat, up to 1500 users can participate anonymously in one chatroom. Since the KakaoTalk messenger provides the original file upload option in which a stego-image file can be shared safely without being damaged during the upload process [38], authors showed a possibility of constructing a stego-botnet based on the KakaoTalk open chat. Recently, Gasimove et al. [39] implemented covert channels to transfer hidden information over WhatsApp, which is the most popular IM in the world. While some researches on IMs have been conducted to point out the dangers of hacking IMs by using steganography, no corresponding studies have been identified.

Next, there are a couple of studies on countermeasures against steganography-based secret communication in SNS services as follows. Konstantinos et al. [40] extensively reviewed image steganalysis techniques for digital forensics. Natarajan et al. [41] conducted a research on detecting covert communication or botnets using steganography in SNS environments in 2012. In this study, host-based detection methods were proposed for steganography-based botnet detection. Specifically, assuming that stego images are uploaded to profiles on Facebook, the entropy of these images is trained by using machine learning techniques and detected stego-images by using an ensemble classifier. The same authors extended their work by adding the process of categorizing malicious profiles on SNS (Flickr) prior to the detection of stego images [42].

According to our survey, we observed that there are no studies that conduct automated system or techniques that detect steganography image files shared in SNS instant messengers. By this motivation, we in this study, propose an automated detection model and system that can automatically collect and detect steganography image files shared in SNS IMs.

3. Design of Automated Steganography Image-Detection System

In this section, we describe our automated steganography image-detection procedures in a KakaoTalk chatroom and then design the structure of our proposed system.

3.1. Automated Detection Procedure of Steganography Images Shared in a KakaoTalk Chatroom

Before we describe our automated detection procedure, we assume that a stego-botnet is already constructed in a KakaoTalk chatroom by an attacker (bot master) as shown in Figure 3. Thus, in this situation, the bot master periodically uploads stego-image files containing bot commands at the

chatroom and victims (bots) read and download those stego-image files from the chatroom because the image files look normal and interesting to them.

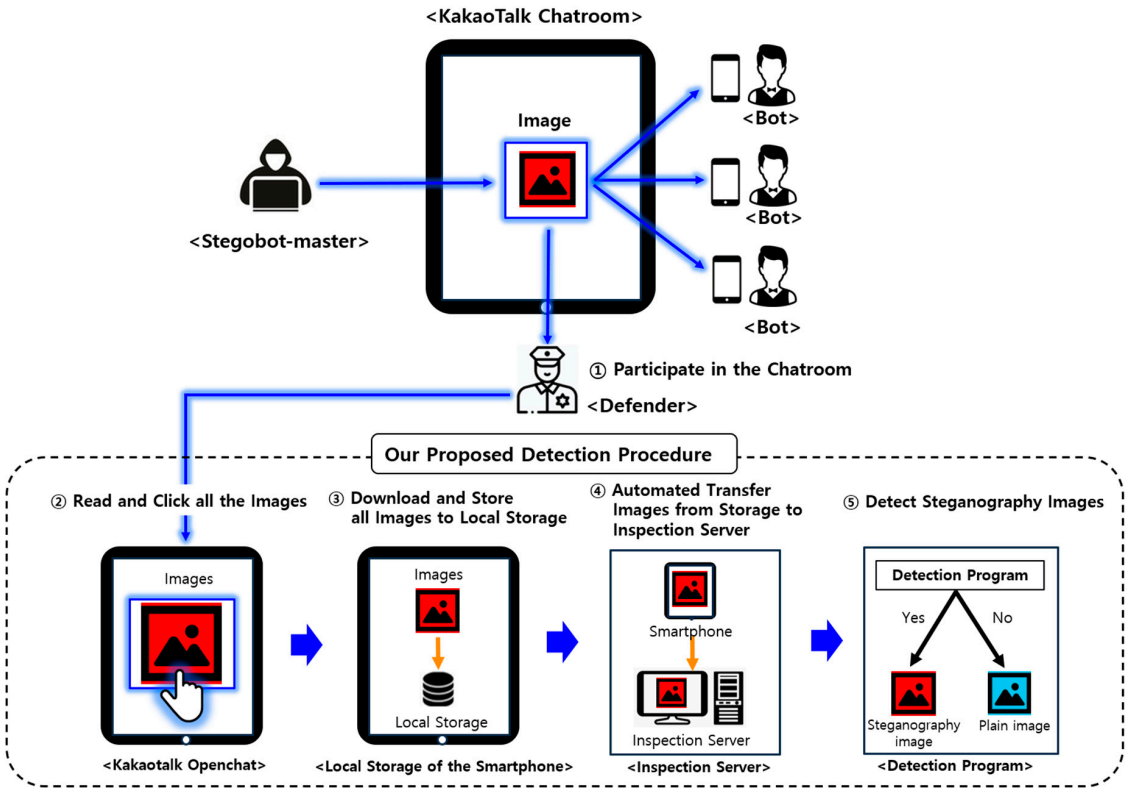

Figure 3. Proposed steganography image detection procedure in a KakaoTalk chatroom.

We now describe our automated detection procedure to capture steganography image files shared by the bot master at the chatroom. The detection procedure consists of the following five steps S1–S5 (see Figure 3). We note that S4 is implemented semi-automatically in Section 4.

S1. Defender participates in a KakaoTalk chatroom that he/she wants to monitor by using his/her device (smartphone or PC).

S2. Defender reads and clicks all shared image files in the chatroom.

S3. Then, image files are downloaded and saved at Defender's device (local storage).

S4. Stored image copies are automatically and periodically transferred from Defender's device (local storage) to Defender's inspection server (this stage is called automated collection).

S5. All collected image files are examined by our detection system and report if there are steganography image files (this stage is called automated detection).

3.2. Design of Our Proposed System Model

To develop our proposed system that works as the steganography image-detection procedure as described in Section 3.1, we design our system that consists of two major components such as *automated collection component (ACC)* and *automated detection component (ADC)* as shown in Figure 4.

First, the automated collection component will automatically collect all image files shared at KakaoTalk chatrooms. We design the automated collection component as follows. When Defender reads and clicks image files shared at a chatroom, those files are stored in the local storage of the Defender's device (e.g., smartphone or PC). To move them from the Defender's smartphone to the inspection server, we used a smartphone-to-PC synchronization app (Foldersync [43]). The reason to use such method is as follows. Initially, we tried to transfer image files from a smartphone to a server by connecting them through a USB cable, but we failed because our testing smartphone

(Samsung Galaxy S10 5G) uses media transfer protocol (MTP) method when it transfers data using a USB cable but unfortunately, it was restricted for our inspection server (PC) to access the storage of the smartphone. On the other hand, we confirmed that it is feasible to use a smartphone-to-PC synchronization application for periodic file transfer from a smartphone to the inspection server. Moreover, we can select a synchronization cycle through the scheduled synchronization option which allows you to periodically transfer image files.

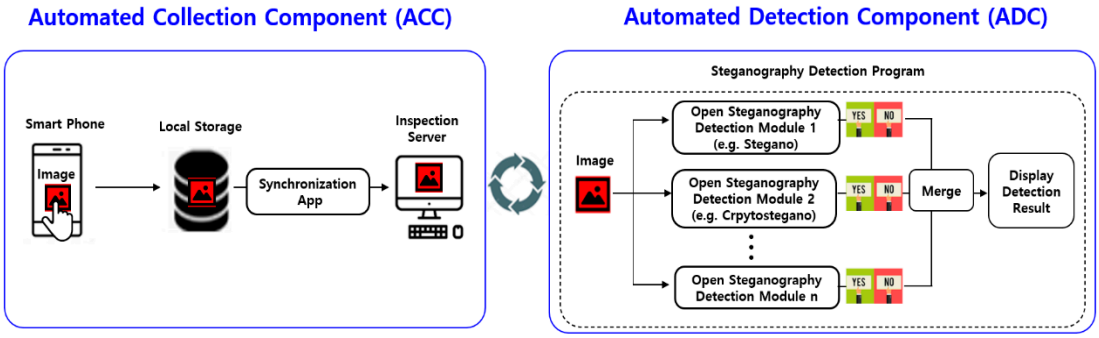

Figure 4. Design of our proposed system with two main components (automated collection component and automated detection component).

Second, the automated detection component will automatically examine whether collected image files contain hidden steganography messages. As shown in Figure 4 (the right part), we design our automated detection component such that it can adopt more than one open steganography-detection software that provides API so that we can develop our steganography detection program based on it. There are numerous image steganography tools and methods which are available in the Internet [44,45] and we do not know what kind of tools will be used by the attacker. Thus, no single steganographic-detection method can detect steganography images perfectly. Consequently, this generic and scalable architecture of our proposed system will overcome the limited detection scope of a single steganography-detection tool, and thus it will extend the detection scope of our proposed system by integrating multiple open steganography software or tools. There are many available steganography tools that can be considered in our ADC such as Stegano, Cryptosteganography, Stegstamp, Stegonography, Stego, Stegbrute, Steganographer, and so on [21,22,46].

4. Implementation and Experiments

In this section, we describe how we implement our proposed system based on the system design explained in the previous section and then conduct experiments to show our proposed system accurately detects test steganography image samples and displays detection results.

4.1. Implementation

4.1.1. Automated Collection Component (ACC)

To implement automated collection component (ACC), we used one smartphone (Defender's device) and one PC (inspection server). As we explained before, when Defender clicks an image file shared at a KakaoTalk chatroom (see Figure 5a), that image file is downloaded and stored at the Defender's device (smartphone). To find the location of the image file, by using various digital forensic methods in [47], we examined the local storage area of the smartphone and found the location as "Internal storage/Android/data/com.kakao.talk/contents/Mg==" (see Figure 5b). When we examine the folder, there exists a file whose name consists of 64 hexadecimals without any file extension. To analyze the file, we used Hex Editor (see Figure 5c) [37]. By adding an image file extension (.jpg or .png) after the name of the file, we could convert it to an image file (see Figure 5d).

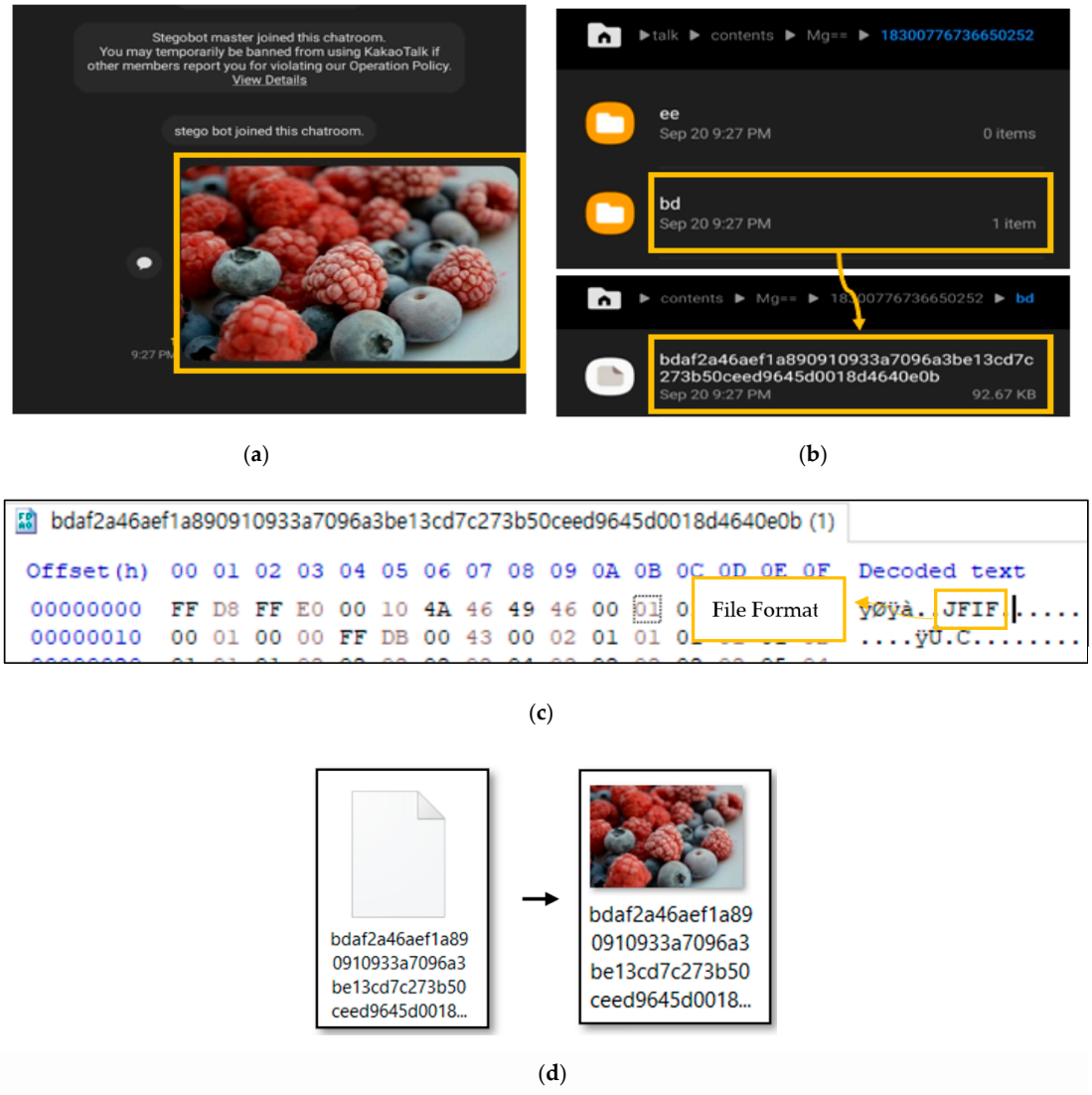

Figure 5. Locating an image file downloaded from a KakaoTalk chatroom at the smartphone. (**a**) Defender clicks on the uploaded image. (**b**) Location of downloaded image files. (**c**) Hex view of Stegano image file. (**d**) An example of converting a file without an extension into an image file.

Next, after locating all image files, we moved them to the inspection server. As we explained in Section 3.2, we used a synchronization app for android smartphones, which is a freeware Folder Sync version 3.0.17 [43] (see Figure 6a). Folder Sync supports various synchronization methods for Cloud, FTB, SMB, etc., and the collection period and schedule can be determined (see Figure 6b,c); we used the SMB option to implement our proposed system. If a server (PC) and a smartphone are located at the same Wi-Fi zone, all files in the specified folder of the smartphone are periodically moved to the folder specified in the server (PC) according to the pre-determined time interval.

4.1.2. Automated Detection Component (ADC)

Once image files are collected by ACC, automatic detection component (ADC) examines whether the collected image files contain hidden steganography messages. As we explained in Section 3.2, we designed ADC which has a flexible architecture that can adopt multiple open steganography-detection software libraries in order to extend its detection scope easily.

To this end, we implemented ADC by using Python Programming Language (version 3.8) [48] according to its design as follows.

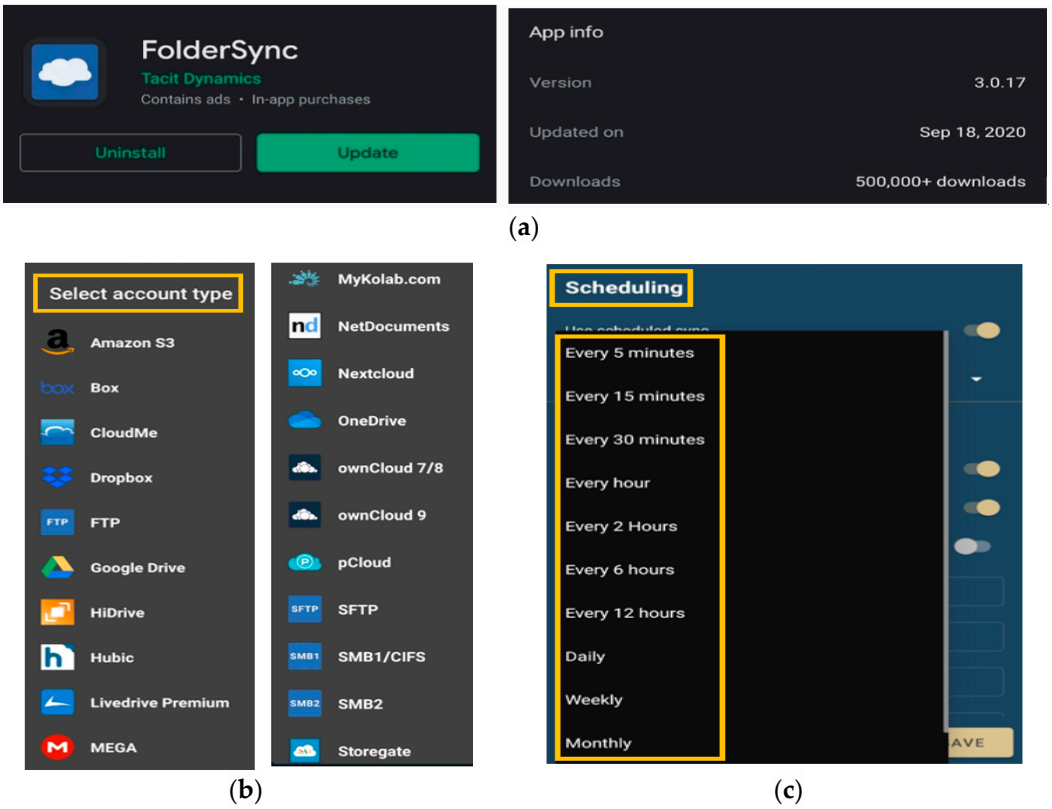

Figure 6. Implementation of automated collection component. (**a**) Freeware sync applications FolderSync. (**b**) Synchronization method selection function. (**c**) Periodic file transfer function.

First, ADC finds steganography image files from the collected files. Next, for each image file, ADC checks whether a hidden message can be extracted from the image file. For this, as shown in Figure 7, we integrated the detection function of an open steganography tool (Stegano version 0.9.8, Cryptosteganography version 0.8.3) into our ADC [21,22]; these steganography tools provides a source library of its steganography detection so that it can be easily integrated into our ADC. We note that our ADC can easily extend its detection capability by employing an open source steganography tools by this manner.

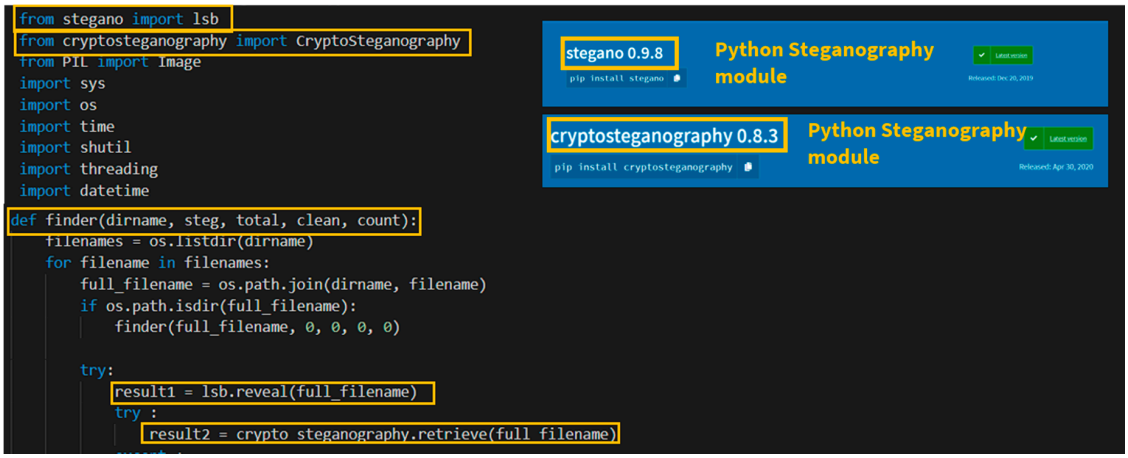

Figure 7. Steganography image detection of automated detection component (ADC).

Second, our ADC periodically conducts the above detection procedure because image files are uploaded frequently at the chatroom. As shown in Figure 8, ADC can adjust its detection cycle by

setting the Thread timer to a certain value (e.g., every 300 s). This function enables ADC to periodically check and examine recently shared image files.

```
from stegano import lsb
import sys
import os
import time
import shutil
import threading
import datetime

def Auto_find():

    start = time.localtime()
    print("== Inspection Start ==")
    print("- Inspection Start Time : ", start.tm_year,"Year", start.tm_mon,"Month", start.tm_mday, "D", start.tm_hour,"H" , start.tm_min,"M", start.tm_sec, "S")
    print("File Number  Inspecion Result(Y or N)   Hidden Message   File name")
    search("D://Jun/Kakaotalk")
    finder("D:\Jun\inspect", 0, 0, 0, 0)
    end = time.localtime()
    threading.Timer(300, Auto_find).start()
    print ("Inspection End Time: ", end.tm_year,"Year", end.tm_mon,"Month", end.tm_mday, "D", end.tm_hour,"H" , end.tm_min,"M", end.tm_sec, "S" )

Auto_find()
```

Figure 8. Implementation of periodic examination of ADC.

Last, ADC displays its examination results periodically. As shown in Figure 9b, the examination results include inspection number, inspection results by two open steganography tools (Y (if detected) or N (if not detected)), hidden message (if each tool can extract it), and inspected filename. In addition, ADC displays inspection start time and we use this information to calculate inspection processing time later in our experiments.

```
if result1 and result2 != None :
    count = count + 1
    print("    ", count, "        ", "Y", "            ", "Y", "            ", result1, "    ", result2, "    ", full_filename)
    steg = steg + 1
elif result1 != None:
    count = count + 1
    print("    ", count, "        ", "Y", "            ", "N", "            ", result1, "    ", result2, "    ", full_filename)
    steg = steg + 1
elif result2 != None :
    count = count + 1
    print("    ", count, "        ", "N", "            ", "Y", "            ", result1, "    ", result2, "    ", full_filename)
    steg = steg + 1
else :
    count = count + 1
    print("    ", count, "        ", "N", "            ", "N", "            ", result1, "    ", result2, "    ", full_filename)
    clean = clean + 1
```

(**a**)

```
== Inspection Start ==
- Inspection Start Time :  2020 Year 12 Month 1 D 20 H 43 M 50 S
File Number   Inspection Result(Y or N)          Hidden Message          File name
              Stegano    |     Crypto          Stegano    |    Crypto
    1           N              N                None           None     D://Jun/inspect\Ex_image (2).jpg
    2           N              N                None           None     D://Jun/inspect\Ex_image (6).bmp
    3           Y              Y                Secret         Unknown    D://Jun/inspect\Ex_image (6)_stegano.bmp
==Inspection Statistics==
Number of Inspection Files :  3 Steganography Images  : 1 Normal Images :  2
Inspection End Time:  2020 Year 12 Month 1 D 20 H 44 M 3 Second
```

(**b**)

Figure 9. Implementation of displaying detection result of ADC. (**a**) Codes for displaying detection results; (**b**) An example of actual detection result display by our ADC.

4.2. Experiments

4.2.1. Experimental Purpose and Methods

In this experiment, we demonstrate that our implemented system can work properly according to our design by automatically and periodically collecting image files from a KakaoTalk chatroom, detecting sample steganography image files from the collected files, and displaying inspection results.

Table 1 shows our experimental environment. For the Defender's smartphone and inspection server, we used one Samsung Galaxy S10 smartphone and one laptop (Lenovo Ideapad), respectively. For the SNS chatroom, we used the KakaoTalk IM mobile application. We implemented our ACC and ADC by using the Python Programming Language ver. 3.8, Folder sync ver. 3.0.16, and two open steganography modules (Stegano ver. 0.9.8. and Cryptosteganography ver. 0.8.3). In addition, we prepared 40 sample images (BMP and PNG format), and we used 20% of sample images (8 images) as stego-images by embedding a hidden message "Secret" by using Stegano and Cryptosteganography. Figure 10 shows our sample images (32 normal images and 8 stego-images). All these images have the same resolution (640 × 420 pixel).

Table 1. Experimental environment.

Element	Value
Defender's smartphone	Samsung Galaxy S10 5G (Android 10 Q); ACC is installed
Inspection server	Lenovo Ideapad L3 (Window 10); ADC is installed
Synchronize application	Folder sync ver. 3.0.16 (Freeware) [43]
KakaoTalk version	8.9.6 (released on 24 July 2020) [9]
Steganography detection module	Stegano ver. 0.9.8 (coded by Python) [21] Cryptosteganography ver 0.8.3 (coded by Python) [22]
Sample cover images	40 (20 PNG, 20 BMP/Resolution: 640 × 420 pixel)
Stego-images	8 (4 PNG (by Stego), 4 BMP (by Cryptosteganography))

We conducted our experiment as follows. First, we created a KakaoTalk chatroom. Next, the Defender (smartphone) with our proposed system (ACC) joined the chatroom. Then we uploaded 40 sample images randomly for two hours (120 min) to the KakaoTalk chatroom; to ease our analysis, we uploaded four stego-images by Stegano between 1st and 20th turn and four stego-images by Stegano between 21st and 40th turn. The ACC and ADC were set to collect and inspect sample images every 15 min, respectively. Thus, ACC and ADC operate 8 times for two hours to collect and inspect uploaded images in the chatroom, and we observed the upload turns of stego-images were 1st, 4th, 13rd, 18th, 23rd, 29th, 33rd, and 38th; the first four images were made by Stegano and the remaining four images were made by Cryptosteganography. We will confirm these stego-images were correctly detected by our ADC.

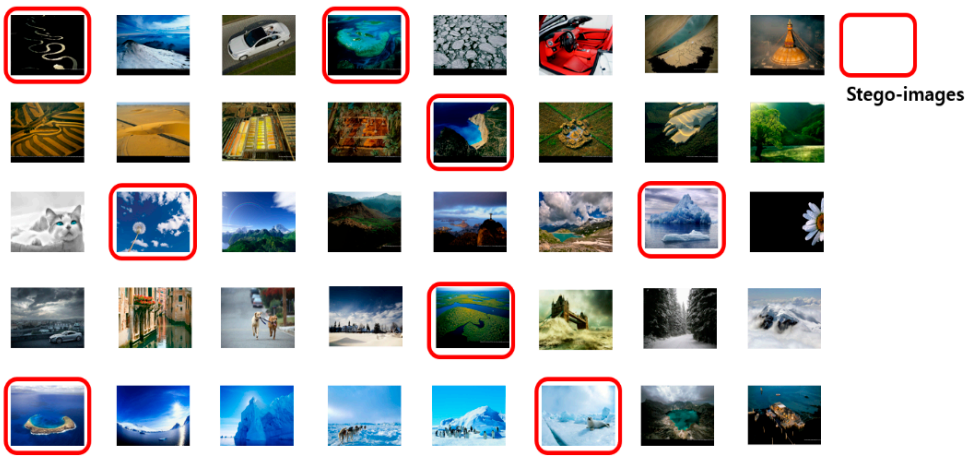

Figure 10. 40 Sample images used in our experiments.

4.2.2. Experimental Results and Analysis

We now explain the results of our experiment. When 40 images were uploaded, ACC copied and transferred them directly to the inspection server every 15 min as we set. Figure 11 shows the inspection result by our ADC. For each inspected file, our ADC displayed the inspection result such that "N" for normal image and "Y" for stego-image. Among the 40 sample images, 8 stego-images (1st, 4th, 13rd, 18th, 23rd, 29th, 33rd, and 38th) were correctly detected with the existence of a steganographic hidden message in image files by both steganography modules, and the remaining normal images were not detected. However, Cryptosteganography failed in extracting the hidden message from the first four stego-images files. On the other hand, Stegano extracted a message which is incomprehensible from the remaining four stego-images files.

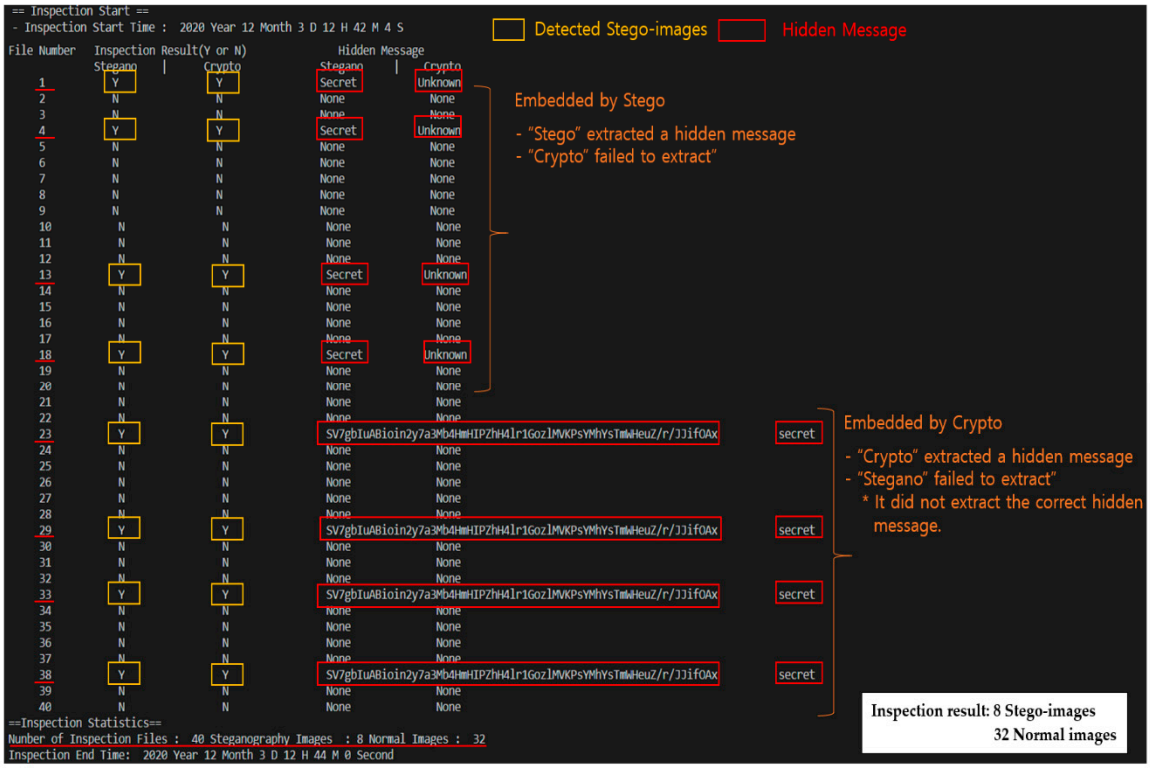

Figure 11. Inspection results displayed by our ADC.

The summary of our experiment is shown in Table 2. For each collection interval (the total number of intervals is 8), we can see the number of collected/inspected files and inspection result (time taken to inspect, the number of normal images, and the number of stego-images). Specifically, during our experiment (two hours), 40 sample image files were collected and inspected every fifteen minutes with ACC and ADC. For example, for the first interval, four images are collected and inspected, and the inspection result shows that two images (1st and 4th images) are detected as steganography images correctly for 7 s (the average time per file is 1.75 s). For the second interval, six new images are collected but 10 images including four images collected in the previous interval are inspected together. Consequently, the time taken to inspect grows as the collection interval increases. When the all 40 files are inspected after eight intervals, 8 stego-images were detected correctly, and the average inspection time per file was 2.73 s. Therefore, we confirm that our system works properly according to our design purposes.

Table 2. The summary of experiment results.

	Collection Interval	Num. of Collected/ Inspected Files	Inspection Result		
			Time Taken to Inspect (Total/Avg. Per File)	Num. of Normal Images	Num. of Stego-Images
	1st (10:45~11:00)	4/4	7″/1.75″	2	2
	2nd (11:00~11:15)	6/10	29″/2.9″	8	2
	3rd (11:15~11:30)	3/13	38″/2.92″	10	3
	4th (11:30~11:45)	7/20	56″/2.8″	16	4
	5th (11:45~12:00)	6/26	1′17″/2.96″	21	5
	6th (12:00~12:15)	4/30	1′25″/2.83″	24	6
	7th (12:15~12:30)	2/32	1′30″/2.81″	26	6
	8th (12:30~12:45)	8/40	1′56″/2.9″	32	8
Total	2 h (00:30~02:30)	40	8′18″/2.73″	32	8

5. Conclusions and Future Works

In this paper, to defend against image steganography-based C&C communication in an SNS chatroom, we proposed, designed, and implemented an automated steganography image-detection system for the KakaoTalk instant messenger. Our proposed system automatically and periodically collects shared image files in a KakaoTalk chatroom to the inspection server, and then examine whether the collected image files contain hidden messages and display the inspection results.

In our future work, we plan to extend our research as follows. First, we will study a method that can trace a bot master in an SNS chatroom, especially in a public chatroom where participants can hide their identities by using nicknames. Tracing a bot master is a very important research issue, but challenging because of the limitation that we can obtain information about the bot master hiding its identity at the chatroom. Second, we will extend our study by considering other SNS IMs such as WeChat or Telegram and other open steganography-detection tools to broaden and strengthen our proposed system's detection capability. Third, our ACC (automated collection component) has a limitation such that it depends on a third party software (FolderSync). We will develop a Python module that automatically locates folders in a smartphone and transfers image files in the folders to our inspection server. Last, we will study a prevention method that can be combined with our detection system to effectively prevent those files from being spread to other SNS chatrooms.

Author Contributions: Conceptualization, Y.C.; methodology, J.P. and Y.C.; software, J.P.; validation, J.P.; formal analysis, J.P. and Y.C.; investigation, J.P. and Y.C.; writing—original draft preparation, J.P. and Y.C.; writing—review and editing, Y.C.; visualization, J.P.; supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Zhou, Y.; Jiang, X. Dissecting Android Malware: Characterization and Evolution. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 24–25 May 2012; pp. 95–109.
2. Geol, D.; Jain, A. Mobile phishing attacks and defence mechanisms: State of art and open research challenges. *Comput. Secur.* **2017**, *73*, 10. [CrossRef]
3. La Polla, M.; Martinelli, F.; Sgandurra, D. A Survey on Security for Mobile Devices. *IEEE Commun. Surv. Tutor.* **2012**, *15*, 446–471. [CrossRef]
4. Yin, T.; Zhang, Y.; Li, S. Dr-SNbot: A social network-based botnet with strong destroy-resistance. In Proceedings of the 9th IEEE International Conference on Networking, Architecture, and Storage (NAS), Tianjin, China, 6–8 August 2014; pp. 191–199.
5. Zhang, J.; Zhang, R.; Zhang, Y.; Yan, G. The rise of social botnets: Attacks and countermeasures. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 1068–1082. [CrossRef]
6. Yang, C.; Harkreader, R.; Gu, G. Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers. In Proceedings of the International Workshop on Recent Advances in Intrusion Detection, Menlo Park, CA, USA, 20–21 September 2011; pp. 318–337.

7. Wu, D.; Fang, B.; Yin, J.; Zhang, F.; Cui, X. Slbot: A serverless botnet based on service flux. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; pp. 181–188.
8. WeChat. Available online: https://www.wechat.com (accessed on 27 September 2020).
9. KakaoTalk. Available online: https://www.kakaocorp.com/service/KakaoTalk?lang=en (accessed on 21 September 2020).
10. Mokel, T.; Eloff, J.; Olivier, M. An Overview of Image Steganography. In Proceedings of the ISSA 2005 New Knowledge Today Conference, Sandton, South Africa, 29 June–1 July 2005.
11. Kanzariya, K.; Nimavat, V. Comparison of Various Images Steganography Techniques. *Int. J. Comput. Sci. Manag. Res.* **2013**, *2*, 1213–1217.
12. Sharda, S.; Budhiraja, S. Image Steganography: A Review. *Int. J. Emerg. Technol. Adv. Eng.* **2013**, *3*, 707–710.
13. Singh, S.; Singh, A. A Review on the Various Recent Steganography Techniques. *Int. J. Comput. Sci. Netw.* **2013**, *2*, 142–156.
14. Makkar, I.; Troia, F.; Visaggio, C.; Austin, T.; Stamp, M. Sociobot: A twitter-based botnet. *Int. J. Comput. Sci. Netw.* **2017**, *12*, 1–12. [CrossRef]
15. Faghani, M.; Nguyen, U. Socellbot: A new botnet design to infect smartphones via online social networking. In Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Montreal, QC, Canada, 29 April–2 May 2012; pp. 1–5.
16. Steghide. Available online: http://steghide.sourceforge.net (accessed on 27 September 2020).
17. Openstego. Available online: https://www.openstego.com (accessed on 27 September 2020).
18. Gaonkar, S.; Dessai, N.; Costa, J.; Borkar, A.; Aswale, S.; Shetgaonkar, P. A Survey on Botnet Detection Techniques. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 24–25 February 2020; pp. 1–6.
19. Sonawane, S. A Survey of Botnet and Botnet Detection Methods. *Int. J. Eng. Res. Technol.* **2018**, *7*, 57–61.
20. FORTINET Threat Landscape Report Q4 2018. Available online: https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/threat-report-q4-2018.pdf (accessed on 24 October 2020).
21. Stegano·PyPI. Available online: https://pypi.org/project/stegano (accessed on 21 September 2020).
22. Cryptosteganography·PyPI. Available online: https://pypi.org/project/cryptosteganography (accessed on 30 November 2020).
23. 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]
24. Vormayr, G.; Ramay, K.; Fabini, J. Botnet communication patterns. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2768–2796. [CrossRef]
25. Imam, M.; Nir, M.; Matrawy, A. A Survey on Botnet Architectures, Detection and Defences. *Int. J. Netw. Secur.* **2014**, *17*, 264–281.
26. 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]
27. 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.
28. 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.
29. Yang, Z.; Wang, B. A Feature Extraction Method for P2P Botnet Detection Using Graphic Symmetry Concept. *Symmetry* **2019**, *11*, 326. [CrossRef]
30. 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.
31. Ferrara, E.; Varol, O.; Davis, C.; Menczer, F.; Flammini, A. The rise of social bots. *Commun. ACM* **2014**, *59*, 96–104. [CrossRef]
32. 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.

33. Yuk, S.; Cho, Y. A Study on Steganography-based Botnet C & C Covert Communication Model using Thumbnail Images in SNS Instant Messengers. In Proceedings of the KSII Spring/Fall Conference, Jeju, Korea, 22–23 May 2020; pp. 197–198.
34. Kwak, M.; Cho, Y. Video Steganography-based Botnet Communication in Telegram Messenger. In Proceedings of the 15th Asia Pacific International Conference on Information Science and Technology (APIC-IST 2020), Seoul, Korea, 5–7 July 2020; pp. 50–73.
35. Kartaltepe, E.; Morales, J.; Xu, S.; Sandhu, R. Social Network-Based Botnet Command-and-Control: Emerging Threats and Countermeasures. In *Applied Cryptography and Network Security, Lecture Notes in Computer Science*; Zhou, J., Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6123, pp. 511–528.
36. Hiney, J.; Dakve, T.; Szczypiorski, K.; Gaj, K. Using Facebook for Image Steganography. In Proceedings of the 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–27 August 2015; pp. 442–447.
37. Jeon, J.; Cho, Y. Construction and Performance Analysis of Image Steganography-Based Botnet in KakaoTalk Openchat. *Computers* **2019**, *8*, 61. [CrossRef]
38. Number of Global Monthly Active Kakaotalk Users from 1st Quarter 2013 to 2nd Quarter 2020. Available online: https://www.statista.com/statistics/278846/kakaotalk-monthly-active-users-mau (accessed on 28 September 2020).
39. Gasimov, V.; Mustafayeva, E. Implementing Covert Channels to Transfer Hidden Information over WhatsApp on Mobile Phones. *Am. J. Eng. Appl. Sci.* **2020**, *6*, 32–35.
40. Konstantinos, K.; Ergina, K.; Giorgos, P. A Review of Image Steganalysis Techniques for Digital Forensics. *J. Inf. Secur. Appl.* **2018**, *40*, 217–235.
41. Natarajan, V.; Sheen, S.; Anitha, R. Detection of StegoBot: A covert social network botnet. In Proceedings of the 1st ACM International Conference on Security of Internet of Things, Kollam, India, 17–19 August 2012; pp. 36–41.
42. Natarajan, V.; Sheen, S.; Anitha, R. Multilevel Analysis to Detect Covert Social Botnet in Multimedia Social Networks. *Comput. J.* **2015**, *58*, 679–687. [CrossRef]
43. Foldersync. Available online: https://www.tacit.dk (accessed on 21 September 2020).
44. Xiao Steganography. Available online: https://xiao-steganography.en.softonic.com (accessed on 27 September 2020).
45. Stealthencrypt. Available online: http://www.stealthencrypt.com/ (accessed on 27 September 2020).
46. The Top 29 Steganography Open Source Projects. Available online: https://awesomeopensource.com/projects/steganography (accessed on 4 December 2020).
47. Barghuthi, A.; Nedaa, B.; Huwida, S. Social networks IM forensics: Encryption analysis. *J. Commun.* **2013**, *8*, 708–715. [CrossRef]
48. Python. Available online: https://www.python.org (accessed on 21 September 2020).