Research Article

Simple and Ingenious Mobile Botnet Covert Network Based on Adjustable Unit (SIMBAIDU)

Min-Hao Wu,1 Chia-Hao Lee,2 Fu-Hau Hsu,2 Kai-Wei Chang,2 Tsung-Huang Huang,3 Ting-Cheng Chang,1 and Li-Min Yi1

1College of Information Engineering, Guangzhou Panyu Polytechnic, Guangzhou, Guangdong 511483, China
2Department of Computer Science and Information Engineering, National Central University, Taoyuan 32001, Taiwan
3Green Energy and Environment Research Laboratories (GEL), Industrial Technology Research Institute (ITRI), Chutung, Taiwan

Correspondence should be addressed to Kai-Wei Chang; popdata520@gmail.com

Received 19 March 2021; Revised 9 July 2021; Accepted 26 July 2021; Published 4 August 2021

Academic Editor: Teen-Hang Meen

Copyright © 2021 Min-Hao Wu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Various services through smartphones or personal computers have become common nowadays. Accordingly, embedded malware is rapidly increasing. The malware is infiltrated by using short message service (SMS), wireless networks, and random calling and makes smartphones bots in botnets. Therefore, in a system without an appropriate deterrent, smartphones are infiltrated easily. In the security threats by malware, random calling has become serious nowadays. To develop the defensive system against random calling and prevent the infiltration of the malware through random calling, it is required to understand the exact process of how to make bots in the botnet. Thus, this research develops a simple and ingenious mobile botnet covert network based on adjustable ID units (SIMBAIDU) to investigate how a botnet network is established by using phone numbers. Perfect octave coding (P8 coding) turns out to be effective in infiltrating smartphones and executing commands, which is used for botnets. The results provide the basic process of P8 coding which is useful for developing defensive systems of smartphones.

1. Introduction

The information era has brought many conveniences into people’s lives. The internet allows fast information transmission that makes the way of communication significantly different from the past. However, the increasing use of smartphones or personal computers for such purposes also causes malware embedding increasingly. As the smartphone combines the properties of computers and telephones and is always connected to the network, personal and financial information in them is vulnerable. Thus, a botnet easily threatens security as involving substantial economic loss [1]. It remotely controls the operating system (OS) by back-dooring the system and embedding malware, which allows cyberattackers to steal any information from the system. The botnet is different from traditional computer viruses as it is always hidden. Therefore, users do not recognize that their systems are included in the botnet. Malware such as Trojan horse backdoors on the device and configures it to autostart without the user’s acknowledgment. The botnet is easily infiltrated into smartphones as making them the bots for control. The obscurity and complexity of the botnet are critical issues as it bypasses the phone’s commands in their OSs.

Many researchers have tried to find a way to protect from botnet infiltration and its cyberattacks with the aims of reinforcing security and preventing malicious attacks [2]. Earlier, the research focused on how to detect the botnet [3] using a system such as a honeypot. Later on, an additional passive system was added for web traffic diagnostics [4]. These technologies were based on several different types such as signature-, anomaly-, DNS-, and mining-based approaches.

In the honeypot, the higher the ability to attract malicious attacks, the more efficient to collect information and find new episodes. Whether low or high interactivity, the honeypot increases the probability of being attacked as much as possible and then collects the expected information.
Several honeypots comprise a whole honeynet that spreads the honeynet spots and integrates the honeypot data for malicious behavior analysis. The honeypot helps understand the techniques and features of the botnet. However, it usually takes a long time. The web traffic surveillance pin-points the botnet's existence in the passive monitoring and the analysis of network traffic. However, the mining-based diagnostic [4] only spots well-known botnets; signature-, anomaly-, and DNS-based diagnostics detect the bots or botnets that have not been discovered before. When an attacker modifies the network architecture or protocol into a botnet, the users can use DNS and mining-based diagnostic detection technology to detect botnets. These diagnostics make it easy to identify botnets, regardless of whether the attacker is adjusting its process or designing [4] the potential for information security in both methods.

The botnet diagnosticians are based on the peer-to-peer (P2P) or hybrid system [5]. As it processes Internet relay chat (IRC) and hypertext transfer protocol (HTTP), P2P is regarded to be relatively good for the detection of botnet infiltration [6]. However, due to the limitation of the power and memory of the smartphone, antivirus software is not appropriate for botnet diagnostics of the smartphone. As smartphones are using various applications, the botnet causes severer security problems for the smartphone than any other operating system [7]. Therefore, there has been much research on the mobile botnet. How to detect and prevent the related cyberattacks, the new covert channel, and the botnet control mode are critical issues nowadays. However, as the mobile botnet is originated from the traditional botnet of personal computers, its details have not been discussed considerably [8–11].

Zeng et al. [12] showed that the botnet is more vigorous on personal computers than the mobile devices. For the mobile botnet, technologies related to BlueTooth, short message service (SMS), and commands and control (C&C) are required to consider [8, 10, 13–15]. Recent research has been focused on the Android botnet such as DroidDream [16] as the use of SMS as a covert channel in the mobile botnet was found. To prevent such incidents, modifying caller ID numbers is recommended, which solves the encountered problems on many mobile OSs. As the caller ID numbers keep changing, the attacker cannot decide which data or command to transmit. They also need to start networking services whenever trying to manipulate the callee. As long as the victim's smartphone is turned on and infected by malware, the attacker uses caller ID numbers for including them as bots without the limits of cost and location. However, when the caller ID numbers keep changing, the smartphone becomes self-protecting and concealment. Despite the easiness and effectiveness, there are not many research results on the use of the caller ID number for preventing mobile botnet infiltration. Thus, this paper focuses on hidden communication and the control model on the smartphone with the following research objectives.

We aim to propose a hidden communication model by using caller IDs to transmit the binary file, a simple and ingenious mobile botnet covert network based on adjustable ID units (SIMBAIDU). This applies to any platform regardless of its operating system. Through the test of the proposed model on Windows Mobile 6 (WM6), we also intend to use caller IDs to send binary executable files, which proves that using caller IDs is used for preventing covert channel controls. As the maximum number of digits of caller IDs is 15 according to the E.164 standard, the experiment results provide a compression method in encoding transferring files and the permutations and combinations of the caller IDs. The model enables sending commands at different times by using changing caller IDs. The proposed model prevents the infiltration of the mobile botnet, which is applied to the networks such as general packet radio service (GPRS), 4G, 5G, and Wi-Fi for the concealed communication by modifying the caller IDs.

2. Proposed System and Simulation

2.1. System Structure. The structure of the proposed system is shown in Figure 1. The botmaster or botherer is the master and manipulator of a botnet. The victim is the person with an infiltrated mobile device by malware. Generally, the attacker makes the bot program that runs automatically after each reboot and is hidden by the bot program. The proposed system uses P8 coding for changing the original caller IDs. The caller mechanism here assumes the attacker's infiltration and spoofing by using the caller IDs. Any command or binary execution file is sent through different phone numbers (caller IDs) and caller ID modification. In the proposed model, we cooperated with a telecommunication operator to change the caller IDs.

2.2. Operation of Malware. In hidden communication, the attacker downloads and sets up malware through social engineering as unnoticed by the user. The attacker uses malicious code for calling which sends binary files and then issues a command to execute calling. This is a typical process of making a botnet. If the malware gives an instruction, it executes the corresponding operation. With a part of a binary file, it is hidden in the device. By executing the files, the malware operates the mobile device as the attacker wants. Without knowing, the victim may pay extra charges and start specific programs that delete the data remotely. The attacker turns the victim's device to be a springboard to call others and tries to infiltrate other devices that are connected to the victim's device. This approach has three essential features, so-called, three zeros: zero cost to spend, zero packets to use, and zero chance to be revealed.

In sending binary executable files, Win32 API and the registry keys are used to capture caller IDs. Caller IDs, in general, have phone numbers with up to nine digits, ten different names, and decimal separators. We used phone numbers as caller IDs in this study. As computers commonly use hexadecimal from zero to F, using binary files requires changing caller IDs based on P8 coding. Sending commands and files demands changing phone numbers as the malware uses different arrangements and frequencies of the numbers
based on P8 coding. The implementation of the arrangement is a permutation with or without repetition as follows:

\[
\begin{align*}
\text{permutations with repetition: } & \ n^r \text{ commands,} \\
\text{permutations without repetition: } & \ P(n, r) = \frac{n!}{(n-r)!} \text{ commands,}
\end{align*}
\]  

where \( n \) is the total number of phone numbers and \( r \) is the number of the phone numbers for finding victims. \( r \) is usually determined when the attacker programs the bot malware. Equations (1) and (2) present the total number of commands that the attacker issues. To reduce the complexity of communications when controlling and avoid annoyance, permutations without repetition are usually used. With permutations without repetition, the attacker uses one command for one device or one command for several devices according to the designed order. For instance, six IDs create 30 available commands (\( 6!/4! = 30 \)). Therefore, it is enough to use 2 or 3 numbers to issue commands to control simple malicious behavior.

The history of calls is recorded by the telecommunication carrier as long as the callee answers in Taiwan. Thus, the carrier does not have any record of when the attacker infiltrates without the callee’s answer. Therefore, the attacker then hides the identity, and the infiltrated device as a bot executes any given commands.

2.3. Compression and Encoding

2.3.1. Compression and Decompression. There are a huge number of applications that implement on the victim’s smartphone through the attacker’s call. The binary executable files are sent in general. Thus, octal is used to code the related program and confirm the loss-less data compression.

In the Perfect Octave Coding (P8 Coding), octal couples with clefs (eight and nine) and a run-length approach. This method is based on encoding numbers into musical notes. In sending, the sender needs to convert the hexadecimal data into a phone number. As the first step, three bytes of hexadecimal data are read as shown in Figure 2. At the second step, hexadecimal codes are converted into binary codes (Figure 3). Then, the binary codes are packaged into octal codes (Figure 4). Finally, the codes are compressed to obtain the second sequence. To packet the second sequence and to group 15 digits of a phone number, phone numbers are used as caller IDs for calling the victim’s device.

2.3.2. P8 Coding. When the malware of the victim’s device receives the decompressed command, its binary executable file does not appear on the device. Its conversion is carried out in the reverse order of the sender’s process. The data compression uses a run-length encoding method that combines octal numbers and the number of conversions to decompress them without error.

A sample of the syntax of P8 coding is similar to music scores. A clef is used as a flag, and key signatures present the number of repetitions. Time signatures represent the name of exact repeating times for coding patterns. Notes are the numbers from zero to seven. This system uses the clef, time signature, notes, and key signatures depending on actual compression requirements. The brackets in Figure 5 showing the syntax of the compression represent the option fields that are optional in compression.

According to the syntax, Figures 6 and 7 show examples of P8 codings. When a clef is eight, the number of repetitions is defined from 5 (one digit) to 99 (two digits). When a clef is 9, the number of repetitions is greater than 99. It continuously squeezes adjacent repeating fingers into six or more numeric characters. As previously described, the number of
repetitions is greater than 100, a clef digit is 9. When a key signature digit indicates the number of repetitions to be 3, then repeating times are in three digits. For example, in case of “931342,” “9” is for the number of repetitions greater than 100, “3” indicates the number in three digits, “134” is the number, and “2” is for repeating the process twice. In the case of “9410003,” “4” indicates the number in four digits and the repetition occurs 1000 times. As we use octal codes, the used numbers 8 and 9 are the signs for information. If a time signature has only one digit, we need “8” as an end code.

If the location of $b_{i+1}$ is 8, then $b_i$ must be 0 to 7, indicating that the number of exact times ($b_{i-1}$, time signature) is five to nine. If $b_{i+1}$ is zero to seven, then $b_i$ must be zero to nine. $b_{i-1}$ and $b_i$ represent ten and nine digits of the exact times. $b_{i+1}$ represents the number of digits (in Figure 8).

The role of clefs is shown in Figure 9 that represents a mutually exclusive relation. When the exact time is great than 100, it resolves the digit of the number of actual times. Then, that will use the clef of 9 as shown in Figure 7. “8” on the left is a sign to reveal the meaning of the following digits. The purpose of the design is to reduce the amount of data and for loss-less transmission. “9” as the first digit signals another encoding rule for “key signature” as follows: “9,” “the digits of duplicate times,” “duplicate times,” “octal number.” This is generalized as the following syntax of P8 coding: “clef 9,” “key signature,” “time signature,” and “note.”

Therefore, the syntax needs one more field to indicate the number of digits of exact times (key signature). The exact times from five to nine have a single digit. Then, the file is compressed with a clef of eight or nine. When the adjacent number is seven, and the number of exact duplicate times is nine, as shown in Figure 10. As discussed before, the compressed length by a duplicate time of 8 or 9 is the same. We use a clef of 9 for a duplicate time of over 100 and 8 for that of below 99.

2.4. Simulation. The experiment was simulating the syntax on the desktop computer in the Windows 7 operating system. The computer was equipped with a 3.00 GHz AMD Athlon(TM) II X2 250 processor and 2 GB of random access memory (RAM). The binary files corresponding to P8 coding were compressed and analyzed by using several caller IDs. We created a program called “hello-world binary executable file” whose size was 3500 bytes. The file displays the word “hi” in a message box when it modifies or overrides the function of the smartphone or uses another Windows API. With the file, we created the other binary executable file for calling other smartphones. The size of the original file was 7680 bytes. The files were extracted and converted to a binary file by using a mobile device of HP iPAQ. They used specific phone numbers to execute conversion and data-saving commands. Windows Mobile 6 Professional Emulator displayed the particular phone number which they want to release. The files were programmed to appear in a specific dictionary by the attacker and operate.

The experiment included the following steps: (1) mastering several groups of available phone numbers, (2) calling the victim’s smartphone to make it a bot, (3) infiltrating into other smartphones by calling from the smartphone, and (4) running a simple executable file. As calling and infiltrating into random smartphones is illegal, we simulated the process by using Windows Mobile 6 Professional Emulator.

3. Results and Discussion

In the simulation, sending the file to the victim’s smartphone numbers from calling to executing the file took about 7.8 s on average. Sending the file to 212 smartphones took 27.5 m to complete the operation. When the file was octal coded, the size of the file was 9559 bytes which was 2.7 times larger than the original file. The compressed file with P8 coding had a reduced size of 3135 bytes in the second sequence. The file compression ratio of the octal coding was 32.8%, while that of the P8 coding was 87.5%. This result reveals the compression ability of P8 coding.
In the process, the data need to be expanded to octal numbers first. Since P8 coding compresses the file, it is important to consider the compression ratio in the following process: another customized binary executable of the file function, for example, is calling with some of the related operations. This original file is 7680 bytes. Octal coding increased the size to 20482 bytes. After compression by P8
coding, it decreased to 13446 bytes. The compression ratio of the P8 coding was 65.6%. When compared to the size of the original file, no compression effect was observed. The file size was increased by 2.7 times by the octal coding when multiplying by eight and dividing by three. The compression process opened the sequence of octal codes rather than the original data in the coding method of this study. Thus, there is an indirect compression of the original file. If an improved compression ratio of the original file is needed, the compression ratio needs to be increased in octal coding. Then, the compression effect is enhanced. Antivirus software such as Airscanner Antivirus for Windows Mobile and Trend Micro Mobile Security Enterprise 5.5 scanning did not detect the files in the mobile device for the experiment.

4. Conclusions
This paper proposes a botnet that regulates caller IDs, especially phone numbers, as a simple and ingenious mobile botnet covert network based on adjustable ID units (SIMBAIDU). SIMBAIDU is the first systematic way to establish malicious attacks. By using perfect octave coding (P8 coding), hexadecimal codes are converted into binary codes and then into octal codes to call and execute a command to control the infiltrated smartphone. After calling the phone numbers stored in the victim’s smartphones, a bot program decompresses the caller IDs of the smartphone and sends binary executable files. This process allows manipulating the victims’ smartphones remotely and establishing covert channels.

For preventing this approach of establishing the covert channel, the VoIP carriers need to offer the users the right to change their caller IDs frequently. Potential threats by mobile botnets are regarded to be through SMS and wireless networks yet, but this study proves that caller IDs are also a covert channel. The fraudsters from overseas keep trying to call with modified phone numbers and some websites provide a tool for changing them. In this situation, the results of this study show how caller IDs are used as a channel for cyberattacks using the botnet. As P8 coding is used for making botnets, the proposed process provides the basis for suggesting possible defenses to prevent such threats.

Data Availability
The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
The authors thank Guangzhou Panyu Polytechnic Innovation and Entrepreneurship Education Center and Panyu Polytechnic Innovation Team support. The Consortium was funded by the Guangzhou Panyu Polytechnic Innovation and Entrepreneurship Education Center under grant no. 210113263 and Panyu Polytechnic Innovation Team under grant no. 2020CXTD003.

References
[1] M. Schipka, “Dollars for downloading,” Network Security, vol. 2009, no. 1, pp. 7–11, 2009.
[2] S. Zander, G. Armitage, and P. Branch, “Covert channels and countermeasures in computer network protocols,” IEEE Communications Magazine, vol. 45, no. 12, pp. 136–142, 2007.
[3] W. Lee, C. Wang, and D. Dagon, Botnet Detection: Countering The Largest Security Threat, Springer Science & Business Media, Berlin, Germany, 2007.
[4] M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” in Proceedings of the 2009 Third International Conference On Emerging Security Information, Systems And Technologies, pp. 268–273, IEEE, Athens, Greece, June 2009.
[5] S. Chang, L. Zhang, Y. Guan, and T. E. Daniels, “A Framework for p2p botnets,” vol. 3, pp. 594–599, in Proceedings of the 2009 WRI International Conference On Communications And Mobile Computing, vol. 3, pp. 594–599, IEEE, Kunming, China, January 2009.
[6] D. Dagon, G. Gu, C. P. Lee, and W. Lee, “A taxonomy of botnet structures,” in Proceedings of the Tenth-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 325–339, IEEE, San Juan, PR, USA, December 2007.
[7] A. Karim, S. A. A. Shah, R. B. Salleh, M. Arif, R. M. Noor, and S. Shamshirband, “Mobile botnet attacks – an emerging threat: classification, review and open issues,” KSII transactions on internet and information systems, vol. 9, pp. 1471–1492, 2015.
[8] A. Flo and A. Josang, “Consequences of botnets spreading to mobile devices,” in Short-Paper Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009), pp. 37–43, Oslo, Norway, October 2009.
[9] M. Ahamad, Emerging Cyber Threats Report For 2009, Georgia Tech Information Security Center, Atlanta, GA, USA, 2008.
[10] C. Mulliner, “Smartphone botnets,” in Proceedings of the 5. GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit, p. 10, Nancy, France, October 2010.
[11] A. Aprville, “Symbian worm Yxes: towards mobile botnets?” Journal in Computer Virology, vol. 8, no. 4, pp. 117–131, 2012.
[12] Y. Zeng, K. G. Shin, and X. Hu, “Design of SMS commanded-and-controlled and P2P-structured mobile botnets,” in Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 137–148, Tucson, AZ, USA, April 2012.
[13] K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee, “Evaluating bluetooth as a medium for botnet command and control,” Detection of Intrusions and Malware, and Vulnerability Assessment, vol. 6201, pp. 61–80, 2010.
[14] C. Mulliner and C. Miller, “Injecting SMS messages into smart phones for security analysis,” in Proceedings of the USENIX Workshop on Offensive Technologies (WOOT), vol. 29, Montreal, Canada, August 2009.
[15] C. Mulliner and J.-P. Seifert, “Rise of the iBots: owning a telco network,” in Proceedings of the 2010 5th International Conference On Malicious And Unwanted Software, pp. 71–80, IEEE, Nancy, France, October 2010.
[16] S. Perez, “More droidDream details emerge: it was building a mobile botnet,” 2011, http://www.readwriteweb.com/archives/droiddream-malware-was-going-to-install-more-apps-on-you %20r-phone.php.