JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 31, 1097-1111 (2015)

Bypass Cell-phone-verification Through a Smartphone-based Botnet

FU-HAU HSU, CHI-HSIEN HSU, CHUAN-SHENG WANG, PEI-HSUN LEE, RUEI-MIN JIANG AND JIA-SIAN JHANG
Advanced Defense Lab
Department of Computer Science and Information Engineering
National Central University
Taoyuan County, 320 Taiwan

Due to the trend that more and more web services, such as Google, Facebook, and many auction websites, require users to open their new accounts or to login to their accounts through cell-phone-verification, cell-phone-verification has become an important function of cellular phones. However, our research shows that cell-phone-verification is not always reliable. This study proposes a new attack method named MAC-YURI (My ACcount, YoUr ResponsIbility) against cell-phone-verification to show people one possible abuse of smartphones. Through MAC-YURI, an attacker can utilize a compromised smartphone as a steppingstone to accept and forward account verification code to finish cell-phone-verification when applying a new account or logging in to an account. We have implemented MAC-YURI on an Android smartphone. Experimental results show that MAC-YURI can successfully assist an attacker in obtaining the verification code of an account without the awareness of a steppingstone smartphone owner. Besides, MAC-YURI also develops an SMS-based mechanism to create a smartphone-based botnet. After such a botnet is created, it is difficult to locate the bot master or the machine a bot will contact in the future. Finally, this paper proposes some recommendations to protect a smartphone against MAC-YURI.

Keywords: cell-phone-verification, smartphone-based botnet, cell-phone security

1. INTRODUCTION

Nowadays cell-phone-verification is widely used by many popular network service providers, such as Google, Facebook, and many auction websites, to confirm the identity of a new account applicant or to further verify the identity of a logging-in user. However, our research shows that cell-phone-verification is not always reliable. This study proposes a new attack method named MAC-YURI against cell-phone-verification to show people one possible abuse of smartphones. Experimental results show that MAC-YURI can successfully assist an attacker in obtaining the verification code of an account without the awareness of a steppingstone smartphone owner. In other words, attackers or spammers still can obtain plenty of accounts using innocent persons' identities. And a keylogger attacker still can use an account protected by cell-phone-verification after using keylogger to steal the login name and password of the account [1]. Besides, MAC-YURI also develops an SMS-based mechanism to create a smartphone-based botnet. After installing MAC-YURI code in a compromised smartphone, the smartphone be-

Received October 1, 2013; revised November 26, 2013; accepted December 20, 2013.
Communicated by Hung-Min Sun.
* This work was supported by the National Science Committee of Taiwan under Project NSC 101-2221-E-008-028-MY2 and Project NSC 100-2218-E-008-013-MY3.

comes a MAC-YURI bot with a bot ID. Then its bot master, who knows the bot ID of the MAC-YURI bot, can command it through any machine that can issue a short message, such as a desktop or a mobile device. Hence it becomes more difficult to find the bot master and to destroy the botnet. The target a MAC-YURI bot commanded to contact is also dynamically assigned in a MAC-YURI command; therefore, the target can be changed to any machine that can be accessed by a bot master and receive short messages. As a result, it is difficult to prohibit a MAC-YURI to communicate with other bots by just blocking some hosts. Finally, this paper proposes some recommendations to protect a smartphone against MAC-YURI attacks.

Nowadays the most popular method to spread malware to smartphones is to bundle them with seemingly benign and useful applications, and then to put them in application stores, such as Android Market [2] or Apple Store, for unwitting users to download. But how to cheat a user to install malware into a smartphone is out of the scope of this paper. Hence, in this paper we assume that MAC-YURI code has been bundled with a popular APP. Hence, MAC-YURI code is executed once the Trojan APP is executed.

This paper is organized as follows. Section 2 discusses related work. Section 3 introduces cell-phone-verification. Section 4 describes the MAC-YURI attack models and MAC-YURI commands. Section 5 describes the components, a flowchart, and a workflow of MAC-YURI. This section also proposes some recommendations to protect a smartphone against MAC-YURI attacks. Section 6 analyzes MAC-YURI from various aspects. Section 7 concludes this paper.

2. RELATED WORK

In this section, we describe other work that launching attacks through smartphones. Various variants of Zitmo, which represents a group of mobile variants of Zeus Trojan, have been found gradually recently [3-6]. Similar to MAC-YURI, these variants intercept and forward short messages to launch their attacks. However, unlike MAC-YURI, which only forwards short messages containing verification code, Zitmo intercepts all incoming short messages of a compromised smartphone and forward them to a remote server; hence, the power consumption of Zitmo is much higher than MAC-YURI's, and the phone bill of the owner of a compromised smartphones increases greatly. Besides, MAC-YURI uses a short message to dynamically specify the forwarding destination. The address of the remote server used by Zitmo is hard-coded in the code. On the contrary, because any machine that a MAC-YURI bot master can access could be the machine to receive forwarded short messages, it is difficult to prohibit MAC-YURI bots from forwarding verification short messages and it is difficult to detect MAC-YURI. Furthermore, MAC-YURI can use multiple-hop attacks to increase the difficulty to find the final machine that receives verification short messages.

Zeng *et al.* utilized SMS as a C&C channel to build a P2P-structured mobile botnet [7]. Botnet commands are transmitted through short messages which are created based on some spam templates to prevent normal users from discovering the commands. Traynor *et al.* demonstrated that a botnet composed of as few as 11,750 compromised mobile phones can degrade service to area-code sized regions by 93%, which shows the potent destructive power of mobile phone-based botnets [8].

Soundcomber is Android-based malware [9]. It masquerades as an application with legitimate needs about mircophone and light-weight speech/tone recognition. Soundcomber can locally identify the digits which a user speaks or types to the IVR through a smartphone. This method allows Soundcomber to steal high value information, such as the credit card number of a user, from a phone transaction. Soundcomber developed several covert channels to transmit stolen data to a malware master in a stealthy way. The covert channels facilitate two independent applications with seemingly innocuous permissions to accomplish a single malicious behavior. However, it spends extra time and power when launching such attacks because local tone/speech recognition is required. Besides, background noise may influence the accuracy of tone/speech recognition.

Weidman built a Command and Control server (C&C) over SMS to create a smartphone-based botnet whose bot master utilizes SMS to control smartphone bots [10]. The major task of the bots is to send spam mails. In order to make botnet-related messages transparent to users, Weidmans code is placed below the application layer; hence, root privilege is required to install the code in a smartphone. The above requirement not only makes it more difficult to create a bot but also makes the code only valid on Android.

Bot masters of AndBot use micro-blog web sites as a channel to dispatch commands to their bots [11]. Through TCP/IP connections which introduce more power consumption comparing with SMS, a bot in an AndBot connects to several hard-coded micro-blog web sites to decode and obtain commands from its bot master(s). Hence, once in a while, an AndBot bot needs to actively connect to the above web sites to check for new commands. As a result, commands issued by bot masters may not be executed in real-time.

3. CELL-PHONE-VERIFICATION

When using cell-phone-verification mechanism in a website to register a new account or log in to an existing account, a user is asked to provide his cell phone number so that a short message containing a verification code, called verification short message, can be sent to that number. A new account is created or a login is permitted only after a user provides the correct verification code.

3.1 Google Account

Google provided 2-step verification for accounts contained in their various services [12], such as Gmail, Blogger, YouTube, *etc.* This optional functionality provides account owners a more secure way to login, even though an account owner’s password has been stolen.

3.2 Facebook

Facebook uses similar mechanism to Google’s; however, it is an optional function to Facebook users. According to the help center of Facebook, Facebook uses cell-phone-verification to make sure that Facebook remains a community of people who use their real identities [13]. Fig. 1 illustrates the interfaces used to finish the process of verification.

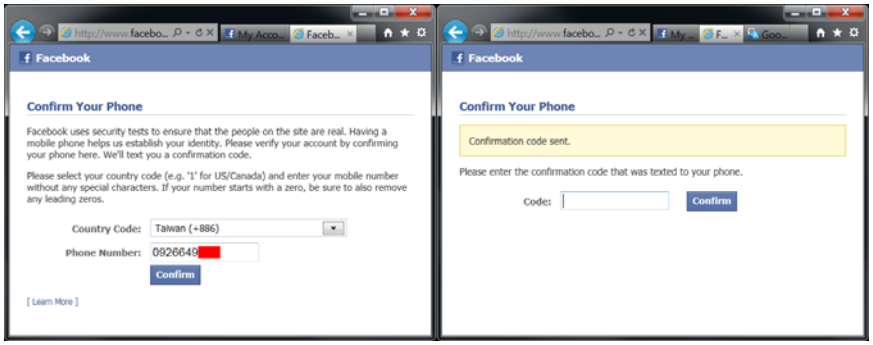


Fig. 1. Cellular verification of Facebook.

3.3 Auction Website

It cannot be too careful to verify the sellers in an auction website, especially when trading activities are involved. Auction websites, such as Yahoo and PChome eBay, provide the platforms for people to purchase goods on line. When signing up for Yahoo Auction, the service will ask those applicants to verify their phone numbers.

4. MAC-YURI ATTACK

This section describes two MAC-YURI attack models, single-hop attack and multiple-hop attack. Both MAC-YURI attack models use victims' smartphones as stepping-stones to receive and forward verification-related short messages.

In both attack models, an attacker uses special short messages, called *MAC-YURI messages* hereafter, to issue commands to MAC-YURI bots. MAC-YURI messages do not generate any ring tone or any notifications in the receiver smartphones. And MAC-YURI messages are processed by MAC-YURI code only. Hence, when MAC-YURI bots handle MAC-YURI messages, these messages are completely stealthy to the owners of compromised smartphones and could be used to create communication channels for smartphone botnets. A MAC-YURI bot has two mode, *normal mode* and *forwarding mode*. When in normal mode, a MAC-YURI bot intercepts any incoming short message and analyzes the message to determine whether it is a MAC-YURI short message or it is just a normal short message. If the short message is a normal one, the MAC-YURI bot passes it to the original smartphone code and lets the code handle it. If the short message is a MAC-YURI message, the bot switches to forwarding mode. When in forwarding mode, a MAC-YURI bot prepares to forward the first incoming short message received by the bot during the time interval specified by the latest MAC-YURI message. The short message needed to be forwarded to other device is supposed to be a verification short message.

4.1 Single-Hop Attack

A single-hop MAC-YURI attack only utilizes one smartphone bot to obtain the verification code. A single-hop MAC-YURI attack consists of three phases, *command delivery phase*, *verification code delivery phase*, and *account grab phase*.

As shown in Fig. 2, at the command delivery phase, an attacker sends a MAC-YURI message to a MAC-YURI bot. The command in the message asks the bot to prepare to receive and forward the first incoming short message during the time interval specified by the command. The device to which a short message is forwarded is also indicated in the command. After handling the MAC-YURI message, the bot deletes it immediately and switches to forwarding mode.

Fig. 2. Command delivery phase of a MAC-YURI attack.

Fig. 3. Verification code delivery phase of a MAC-YURI attack.

As shown in Fig. 3, at the verification code delivery phase, an attacker signs up a web service, such as Gmail or Facebook, or logs in a web service first. Meanwhile, the attacker fills the cell-phone-verification field on a web page of the web service with the phone number of a MAC-YURI bot. The bot has been in forwarding mode. After the above steps, the web service sends a verification short message to the specified bot.

As shown in Fig. 4, after receiving a verification short message from a web service provider, a MAC-YURI bot in forwarding mode forwards the short message to a device, such as a desktop or another smartphone, controlled by the attacker. Then the bot deletes the verification short message and switches back to normal mode. Finally, the attacker obtains the verification code which he can use to create a new account or log in to an existing account.

Fig. 4. Account grab phase of a MAC-YURI attack.

4.2 Multiple-Hop Attack

Multiple-hop attacks are mutants of single-hop attacks. Multiple-hop attacks greatly increase the difficulty to find attackers' real identities. In a multiple-hop attack, attackers utilize a Skype account with fake identity or a stolen smartphone to issue commands to multiple MAC-YURI bots. Then a verification short message issued by a web service provider flows through these MAC-YURI bots. Finally, the verification short message is sent to the attackers.

Fig. 5 illustrates how a verification short message issued by a web service provider is transmitted to an attacker. When obtaining the verification code, the attacker can activate an account with the identity of an innocent user.

Fig. 5. Multiple-hop attack.

4.3 Join a Botnet

The attacker controlling a MAC-YURI bot is called its bot master. A bot master controls a group of MAC-YURI bots and several Command and Control (C&C) server hosts. Each bot has a bot ID whose initial value is randomly set by its MAC-YURI code when the code is first executed on the bot. After being installed, a new bot contacts a C&C server to get the phone number of a compromised bot, called introducer. Then the new bot sends a SMS message to the introducer to tell the introducer its bot ID. Through the SMS message, the introducer can get the phone number and bot ID of the new bot. After the introducer sends the phone number and bot ID of the new bot to the C&C server, the new bot joins the botnet.

4.4 MAC-YURI Command Format

This subsection discusses the format of MAC-YURI commands. Attackers use short messages to delivery MAC-YURI commands to MAC-YURI bots to control them.

As shown in Fig. 6, a MAC-YURI command consists of some of the following fields, command type field (1 byte), bot ID field (4 bytes), wait time field (2 bytes), standby time field (2 bytes), and destination field (less than or equal to 32 bytes). The command type field indicates the operation to perform. Currently, MACYURI has 4 types of commands, "?", "#", "*" and "!". Command "?", called SMS command, requests a command receiver to forward an incoming short message to a smartphone through SMS.

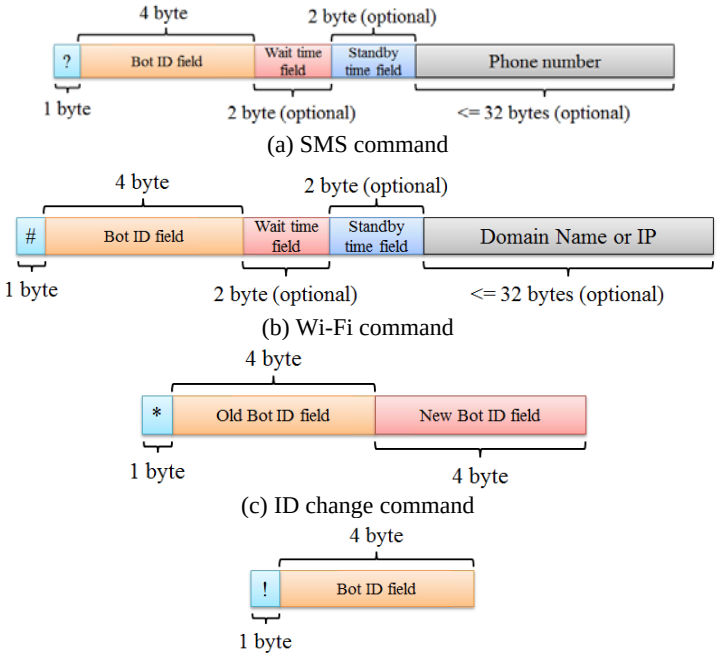


Fig. 6. Format of a MAC-YURI command.

Command "#", called Wi-Fi command, requests a command receiver to forward an incoming short message to a device through Wi-Fi or 3G. Command "*" changes the bot ID of a MAC-YURI bot. Command "!" tells a MAC-YURI bot to abandon the latest SMS or Wi-Fi command.

After receiving a MAC-YURI command, a MAC-YURI bot uses the bot ID field of the command to confirm that the command comes from its bot master. A MAC-YURI command with invalid bot ID is ignored.

The wait time field specifies the time interval a command receiver needs to wait before it can start forwarding an incoming short message to other device. The standby time field specifies the time interval during which a command receiver must forward the first incoming short message. Fig. 7 depicts the above time intervals in a MAC-YURI SMS command or Wi-Fi command.

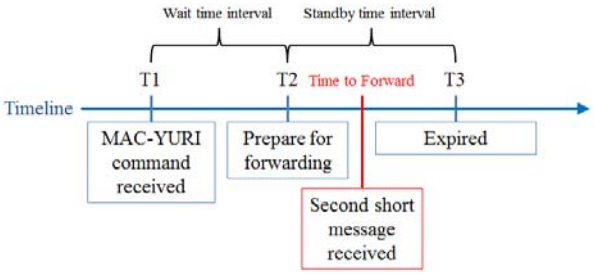


Fig. 7. Time intervals of MAC-YURI attack.

5. EXPERIEMTNAL SETUP

This section describes the components, and future extension of MAC-YURI. This section also proposes some recommendations to protect a smartphone against MAC-YURI attacks.

5.1 MAC-YURI Components

After being installed in a smartphone, MAC-YURI is executed in the background as a service. Fig. 8 shows the major MAC-YURI components. This subsection describes these components and the relationships between these components.

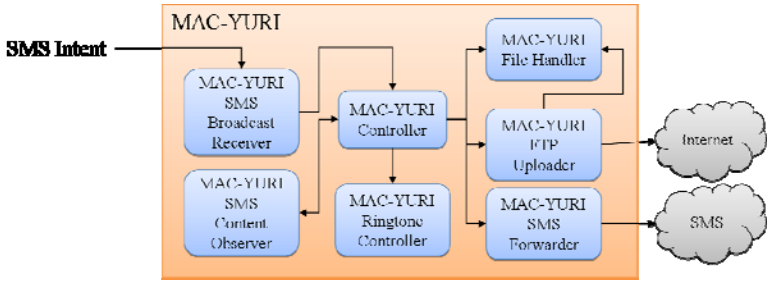


Fig. 8. MAC-YURI components.

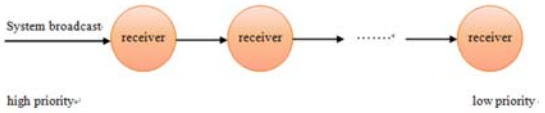


Fig. 9. Android broadcast receiver.

5.1.1 MAC-YURI SMS broadcast receiver

This component is responsible for executing/awaking MAC-YURI Controller whenever MAC-YURI Controller is not running. When a MAC-YURI bot receives an incoming short message that contains a MAC-YURI command, this component stores the short message to the MAC-YURI Command Job Queue. Moreover, an incoming short message which is the first one arrives at the bot at the standby time interval specified by the latest MACYURI command will also be stored in the MAC-YURI Command Job Queue. At compile-time, an intent-filter, "android.provider.Telephony.SMS_RECEIVED," is added to the manifest file of MAC-YURI; hence, when a MAC-YURI bot receives a short message, the related intent will notify this component of the event. MAC-YURI SMS Broadcast Receiver is an Android Broadcast Receiver. Android uses ordered broadcast to handle SMS broadcast. Fig. 9 shows Android ordered broadcast. In ordered broadcast, it is possible to intercept a short message before the original Android code handles it, if we give the MAC-YURI SMS Broadcast Receiver component the highest priority (<intent-filter android:priority="1000">) and use API abortBroadcast() to stop other Android applications to handle MAC-YURI related short messages, such as a MAC-

YURI message or a verification short message. In this way, incoming MAC-YURI related short messages will not trigger any ringtone or vibration and short messages will not be stored in the SMS content provider which makes it more difficult to detect MAC-YURI.

5.1.2 MAC-YURI command job queue

MAC-YURI Controller and MAC-YURI SMS Broadcast Receiver use this component to handle incoming MAC-YURI related short messages. MAC-YURI uses Command Job Queue to store incoming MAC-YURI related short messages. Each stored short message represents an unfinished job, called *MAC-YURI command job*. Whenever the MAC-YURI SMS Broadcast Receiver gets a MAC-YURI related short message, the MAC-YURI SMS Broadcast Receiver pushes the short message into the Command Job Queue Table 1 shows the fields of a MAC-YURI command job.

Table 1. Fields of a MAC-YURI command job.

Field	Description
Content	The content of a MAC-YURI command job.
Address	The sender cell-phone number.
Time	SMS arrival time.

Table 2. Format of a MAC-YURI status file.

Line	Format
1	Current status
2	Forward method (SMS / Wi-Fi / 3G network)
3	Destination field (Phone number / DN or IP)
4	Wait time field
5	Standby time field
6+	Content of the incoming short message.

5.1.3 MAC-YURI controller

MAC-YURI Controller initializes the bot ID of a new MAC-YURI bot and notifies the bot master of the bot ID and phone number of the new bot. MAC-YURI Controller retrieves MAC-YURI command jobs from MAC-YURI Command Job Queue.

MAC-YURI Controller executes its tasks according to the current mode of a MAC-YURI bot and the latest command the bot receives. After retrieving a MAC-YURI command job from the MAC-YURI Command Job Queue, MAC-YURI Controller parses it first. If the MAC-YURI command job contains a MAC-YURI SMS or Wi-Fi command with a valid bot ID, MAC-YURI stores command-related information into a status file, switches the bot to forwarding mode. Command-related information includes the data stored in the wait time field, standby time field, and destination field. Table 2 shows the format of a MAC-YURI status file. Since MAC-YURI needs to read, modify, and send

Table 3. Permissions embedded in the manifest file of MAC-YURI malware.

Permission
`android.permission.RECEIVE_SMS`
`android.permission.READ_SMS`
`android.permission.WRITE_SMS`
`android.permission.SEND_SMS`
`android.permission.INTERNET`

short messages, Table 3 lists the permissions embedded in the manifest file of MAC-YURI malware.

5.1.4 MAC-YURI file handler

This component is invoked whenever MAC-YURI Controller needs to modify or read the content of a file. Android's file system is a bit different from that of personal computers; hence, to facilitate MAC-YURI programming, we pack commonly used file operation APIs of Android into a customized class called "*MAC-YURI File Handler*," which uses the API – getFileStreamPath(path) to convert a relative path to the absolute path that includes an Android application's home directory.

5.1.5 MAC-YURI SMS forwarder

This component uses SMS to forward the first short message received by a MAC-YURI bot in the standby time interval indicated in the latest MAC-YURI command. MAC-YURI Controller calls this component to forward a short message. Table 4 shows the API used. For different Android platform versions, MAC-YURI imports different Android framework libraries at compile-time to satisfy the requirements of different Android versions.

Table 4. API for forwarding short messages.

Android Platform	API Level	Android Framework	API Name
1.6-2.2	4-8	`android.telephony.SmsManager`	`sendTextMessage`
1.5	3	`android.telephony.gsm.SmsManager`	

5.1.6 MAC-YURI FTP uploader

If the SIM card of a MAC-YURI bot supports 3G or Wi-Fi, then after the bot receives a MAC-YURI command which asks the bot to forward a short message to a host, MAC-YURI FTP Uploader will use FTP [14] to forward the short message that satisfies the condition specified by the MAC-YURI command to the destination host. This component uses MAC-YURI File Handler to read the content of the related status file created by MAC-YURI Controller. Uploading the status file over Internet can be done, if either the SIM card supports 3G or a Wi-Fi connection is available.

5.2 Weakness and Future Extension

The current MAC-YURI version assumes that a verification short message arrives

at a MAC-YURI bot soon after the bot receives the corresponding MAC-YURI command. During the short standby time interval specified by the MAC-YRUI command, if another normal short message arrives at the bot first, then the MAC-YURI bot will forward the normal short message, instead of the verification short message, and leave the verification short message in the MAC-YURI bot.

Instead of current design, MAC-YURI can adopt the following approaches to increase the difficulty to detect it. First, a MAC-YURI command can disguise as an advertisement. Hence, even if the owner of a compromised smartphone sees a MAC-YURI command, the command is not likely to cause any suspicion, because it just looks like an advertisement. Second, the functionality of MAC-YURI can be implemented by a group of distinct applications which communicate with each other through a covert channel proposed by Soundcomber [9]. As a result, each application only needs a small subset of the permissions required by MAC-YURI, which can reduce the vigilance of a smartphone owner when installing these applications. Besides, except forwarding verification short messages, attackers can extend the functionality of MAC-YURI to allow them to remotely control MAC-YURI bots. Attackers can even expand the function of MAC-YURI commands and use MAC-YURI messages as a communication channel to create a smart-phone-based botnet, which can greatly increase the destructive power of MAC-YURI.

5.3 Protection Recommendation

A smartphone user can utilize the following methods to reduce the damage level caused by MAC-YURI. First, pay attention to strange permission requests, such as turning off ring-tone, when installing an Android application. Second, examine his SMS log for any unusual short messages, such as a short message that is not sent by the user. Finally, when an Android system receives or sends an SMS message, it records an AT command in the log file. Therefore, we can monitor both the log file and SMS database. If a new AT command is logged but there is no corresponding SMS message, there is a high possibility that the smartphone is infected by some malware.

6. EVALUATION

We have implemented MAC-YURI on Android 1.6 platform. In this section, we use various measurements to evaluate the effectiveness and permission number of MAC-YURI code.

6.1 Effectiveness

We evaluate the effectiveness of MAC-YURI by performing four different experiments to grab the verification code sent by five different web services, Facebook, Gmail, Google, Yahoo Auction, and PChome & eBay JV. For each web service, we first manually created an account and enabled account cell-phone-verification. Second, we installed MAC-YURI in a smartphone and set the phone number of the smartphone into the cell-phone-verification system of the service. Therefore, when we used the account of the

web service, the cell-phone-verification system of the web service will send the verification code to the smartphone. After the verification code arrived at the smartphone, MAC-YURI intercepted it and forwarded it to a server specified by us. We tested both the single-hop and multiple-hop attacks for these web services to evaluate the effectiveness of MAC-YRUI. All experimental results showed that MAC-YURI can successfully obtain verification codes in a few seconds.

6.2 Number of Permission

In this measurement, we compare the number of permissions that MAC-YURI requires with the numbers of permissions that the top 100 free Android applications require. Through this measurement, we want to check whether installing MAC-YURI requires an abnormal number of permissions, which may cause a victim's suspicion. Fig. 10 shows the permission numbers of the top 100 free Android applications from Android Market [2] in the US in March, 2011. On the other hand, researches in [15, 16] showed that from about one-third to one-fourth of Android applications are overprivileged. These applications may make users to allay their suspicion, and cause that malicious attacks success more easily.

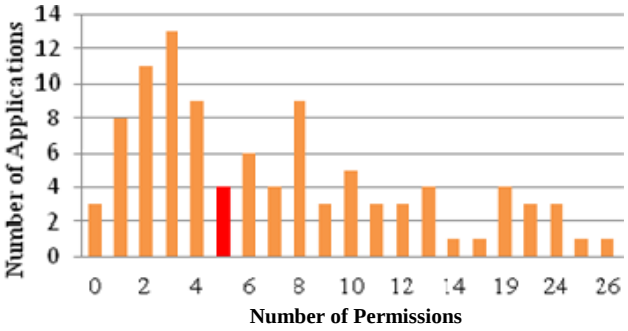


Fig. 10. Permissions used by popular Android applications.

7. CONCLUSION

This paper proposes a novel attack technique, MAC-YURI, which shows cell-phone-verification is not as secure as we think. Attackers can launch a MAC-YURI attack to create a new account or login to an existing account, even though the related web service provider uses cell-phone-verification to secure her accounts. Experimental results show MAC-YURI attacks can assist attackers to obtain the verification code of several famous web service providers through compromised smartphones. Because the number of permissions required by MAC-YURI is low, a smartphone user may unwittingly install MAC-YURI on his smartphone. Besides, as discussed in subsection 5.3, attackers can utilize MAC-YURI to build a smartphone-based botnet, which is as dangerous as a normal botnet. Hence, when enjoying various functions of smartphones, people should not ignore the risks they may encounter.

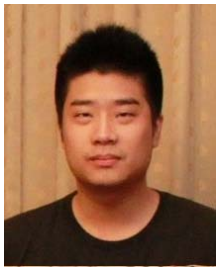
REFERENCES

1. Symantec, Symantec Internet Security Threat Report: Trends for 2010, Internet Security Threat Report, Vol. 16, 2011.
2. Android market, https://market.android.com, 2011.
3. F. Y. Rashid, "New Android Trojans go after SMS messages," posted on eWeek.com, http://www.eweek.com/c/a/Mobile-and-Wireless/New-Android-Trojans-Go-After-SMS-Messages-268345/, 2011.
4. B. Krebs, "ZeuS Trojan for Google Android spotted, posted on Krebs on Security," 2011, http://krebsonsecurity.com/2011/07/zeus-trojan-for-google-android-spotted/.
5. Headlines, "Zeus Trojan migrates to Blackberry OS," posted on Infosec Island, http://www.infosecisland.com/blogview/12381-Zeus-Trojan-Migrates-to-Blackberry-OS.html, 2011.
6. D. Goodin, "ZeuS trojan attacks bank's 2-factor authentication," posted on The Register, http://www.theregister.co.uk/2011/02/22/zeus_2_factor_authentication_attack/, 2011.
7. Y. Zeng, K. G. Shin, and X. Hu, "Design of SMS commanded-and-controlled and P2P-structured mobile botnets," in *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2012, pp. 137-148.
8. P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. L. Porta, "On cellular botnets: Measuring the impact of malicious devices on a cellular network core," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009, pp. 223-234.
9. R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound Trojan for smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium*, 2009, pp. 223-234.
10. G. Weidman, "Transparent botnet command and control for smartphones over SMS," in *Proceedings of International Conference on ShmooCon*, 2011.
11. C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, and Z. Tianning, "Andbot: Towards advanced mobile botnets," in *Proceedings of the 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats*, pp. 11.
12. Google, "Getting started with 2-step verification," Google, https://support.google.com/accounts/bin/answer.py?hl=en&answer=180744&topic=1056283&rd=1, 2012.
13. Facebook Help Center, "Verifying your account, Facebook," http://www.facebook.com/help/?page=116296378454689, 2012.
14. J. Postel and J. Reynolds, File Trensfer Prorocol (FTP), RFC 959, 1985.
15. A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011, pp. 627-638.
16. X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the Android ecosystem," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 31-40.

Fu-Hau Hsu (許富皓) received his Ph.D. degree in the Department of Computer Science from Stony Brook University, New York, USA in 2004. He is an Associate Professor at National Central University and has had an appointment in the Department of Computer Science and Information Engineering since August 2005. He is affiliated with the Advanced Defense Lab and the Wireless and Multimedia Lab.

Chi-Hsien Hsu (許齊顯) is a Ph.D. student in the Department of Computer Science and Information Engineering of National Central University. He received his M.S. degree in Computer Science and information engineering from National Central University, Taoyuan, Taiwan, in 2009, and his B.S. degree in Computer Science and Information Engineering from National Central University. His research areas include mobile security, operating system, and network security.

Chuan-Sheng Wang (王傳陞) received the B.S degree in mathematics from National Central University, in 2008, and the M.S degree in Computer Science and Information Engineering from National Central University, in 2010. He is currently working toward the Ph.D. degree in Department of Computer Science and Information Engineering, National Central University with Prof. Fu-Hau Hsu. His research interests include network security, operating system and malware analysis.



Pei-Hsun Lee (李珮瑄) received the B.S degree in Computer Science and Information Engineering from National Central University, in 2009, and the M.S degree in Computer Science and Information Engineering from National Central University, in 2011. His research areas include Android security and system security.

Ruei-Min Jiang (江瑞敏) is a Master student in the Department of Computer Science and Information Engineering, National Central University. He received the college degree in Computer Science and Information Engineering from National Central University, Taoyuan, Taiwan, in 2012. His research areas include security issues about OS design, mobile devices, especially Linux, Android, and network security.

Jia-Sian Jhang (張嘉顯) is an M.S. student in the Department of Computer Science and Information Engineering, National Central University. He received the B.S. degree in Computer Science and Information Engineering from National Central University, Taoyuan. His research interests include mobile security, operating systems and network security.