A Tracker To Protect Authorization Codes In SMS Using Code inject Method

Rubina Ariff[1], Sabaria. S[2]
[1]Dept of Computer Applications
[2]Assistant Professor, Dept of Computer Applications
[1, 2] B. S. Abdur Rahman Crescent Institute of Science & Technology, Vandalur,Chennai-600 048, India

Abstract- *Now a days, Mobile applications have brought tremendous impact to businesses, social, and lifestyle in recent years. Various mobile applications markets offer a wide range of apps from entertainment, business, health care and social life. Many online banking services are developed in worldwide and mostly using the otpsms to transact the payment from the user bank, but that otp is traced out by the unauthorised apps. More researches are carried out to secure the otpsms from the unauthorised app, then also can't protect the otp from anonymous user apps. To solve this problem and protect the sms, we proposed the CodeInject method, which inject the code into the otp. The otp sending through the sms is not the original otp, it is modified otp by CodeInject method. The authorised apps only have the technique to get the original otp from the modified otp. With the modified otp, the app cant transact the amount from bank. CodeInject method consists of Operation function choose and the critical number generate, this will generate the modified otp for the original otp.*

Keywords- Code Inject, Mobile Applications, Authorized Apps, Otp.

I. INTRODUCTION

Smartphones are widely used in our daily life. Increasingly more users leverage smartphones for online transactions, bank transfers and other operations. Simultaneously, increasingly more websites and applications (apps for short) leverage codes delivered via short message service (SMS) messages to authorize users. We call this type of code an authorization code in this paper. For instance, an SMS authorization code can be required when users log into a banking application or reset their passwords. SMS authorization codes play a crucial role within the application ecosystem, as variety of transactions (e.g., personal identification and online banking) require users to supply a code for authorization purposes. However, authorization codes in SMS messages are often stolen and forwarded by attackers, which introduces serious security concerns.In this project, we propose Code Tracker, a lightweight approach to track and protect SMS authorization codes. In Specific, we leverage the taint tag technique to mark the authorization code with taint tags at the origin of the incoming SMS messages (taint sources), and then, we propagate the tags in the system. To this end, we modify the related array structure, array operations, string operations, IPC mechanism, and file operations for auxiliary storage of SMS authorization codes to make sure that the taint tags cannot be removed. The authorised apps only have the technique to get the original otp from the modified otp. With the modified otp, the app cant transact the amount from bank. CodeInject method consists of Operation function choose and the critical number generate, this will generate the modified otp for the original otp. More researches are carried out to secure the otp sms from the unauthorised app, then also can't protect the otp from anonymous user apps. To solve this problem and protect the SMS.

II. RELATED WORKS

A. SECURITY OF SMS MESSAGES

An assortment of frameworks have been intended to forestall SMS messages from being spilled in cell phones. For instance, Secure SMS and other comparable frameworks influence cryptographic calculations to scramble the SMS messages for confidentiality, honesty and verification administrations, which is an alternate objective contrasted with CodeTracker. SecureSMS endeavors to secure SMS messages by changing the applications getting grouping of instant messages in the framework so that the default SMS application can get the instant message. Then, at that point, it blocks the SMS broadcasting to forestall noxious applications from getting the message. Notwithstanding, Secure SMS just works in Android variants before 4.4.Different frameworks have likewise been proposed to forestall phishing messages. Specifically, these frameworks search the substance of SMS messages to URLs that may connection to malignant applications for establishment and afterward block clients perilous activities. Rather than these applications, CodeTracker plans to give security to approval codes in SMS messages.

B. STATIC AND DYNAMIC ANALYSIS SYSTEMS

To comprehend the chance of protection spillages, a number of data flow examination frameworks have been proposed by scientists. These frameworks can be classified into two primary types. One sort incorporates static examination frameworks that perform examination on the dismantled codes of applications, including FlowDroid, ComDroid, AmanDroid , Droid- Power, CHEX , and so on In any case, the constraint of static examination frameworks is that they can't recognize runtime data divulgence .Accordingly ,dynamic investigation frameworks have been proposed to follow the data flows at runtime in applications. For instance, TaintDroid and a few expanded frameworks (counting DataChest, NDroid, DroidBox , and so on can uphold corrupt following for continuous protection observing on inheritance Dalvik (however not ART) runtime in Android. TaintART and ART are two powerful frameworks planned for the recently presented ART runtime in Android and can be utilized to follow and secure touchy information (counting approval codes) in cell phones. In any case, as referenced in Section I, TaintART experiences the issue of extensibility, what's more, ART doesn't perform well for between application following, which block their utilization for following and securing SMS approval codes. A few other unique examination frameworks focusing on malware have likewise been proposed. For instance, Droid Ranger incorporates an authorization based conduct foot printing plan to identify new examples of known Android malware families; it applies a heuristics-based filtering plan to distinguish certain characteristic practices of obscure noxious families. Malton gives an exhaustive perspective on malware practices by directing multi-facet observing furthermore, data ow following, just as efficient way investigation. Conversely, CodeTracker has an alternate objective of giving following to and insurance of approval codes in SMS messages.

C. IMPRISONMENT OF SMARTPHONE APPS

Various frameworks have been carried out to restrict applications admittance to touchy information. For instance, Kirin applications by forestalling outsider applications from getting to private information. FlaskDroid accomplishes this objective by snaring Android framework administrations. AppCage use two free client level sandboxes to mediate and manage an application's admittance to delicate APIs. To forestall potential security spillage, Aurasium, AppGuard, TISSA, furthermore, RetroSkeleton have been proposed to uphold gained access control on delicate information. This load of frameworks might have the option to be utilized to give assurance to touchy information (counting SMS approval codes) on heritage runtimes (i.e., Dalvik) in Android, however not on the ART runtime. Conversely, CodeTracker works well on Android's ART runtime and can give security just as following for approval codes in SMS messages.

III. EXISTINGMETHODOLOGY

Code an approval code in this paper. For example, a SMS approval code can be required when clients sign into a financial application or reset their passwords. Utilizing SMS codes for approval is helpful; nonetheless, it might introduce security concerns. On the off chance that the code is taken by assailants, it can cause monetary misfortunes to clients.

Another framework used to ensure SMS messages by changing the Android system. Specifically, when a SMS message shows up, Secures sms look through the message text.

DISADVANTAGES

- Malicious applications could capture SMS messages to recover approval codes and afterward block the SMS broadcasting covertly without educating clients.
- Malicious applications can't impede SMS broadcasting, and the framework SMS application will get the SMS messages.

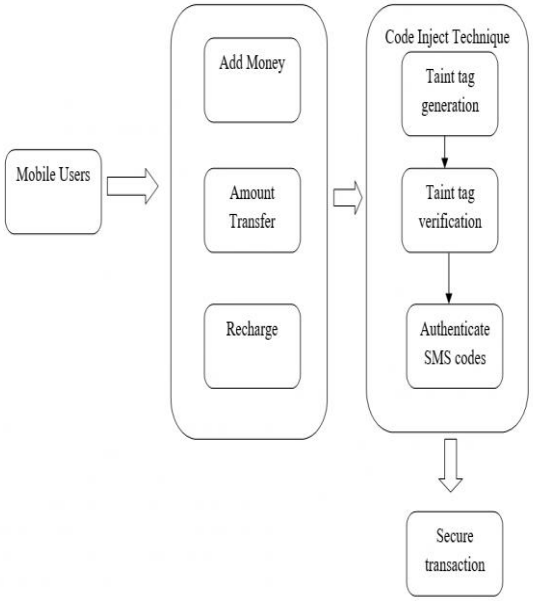
IV. PROPOSEDMETHOD

First, Code Tracker is designed for the protection of SMS authorization codes, not for the protection of general text messages. However, in our DE compilation process, we found that many malware apps steal general messages.

We can easily extend Code Tracker into a prototype system to protect all text messages by applying the taint tags to SMS messages and changing the security policies accordingly. Second, Code Tracker requires changes to the underlying framework; it cannot be transparently supported as a user-level solution.

ADVANTAGES

- Dynamic lightweight approach for tracking and protecting authorization codes in Android.
- Specifically, we leverage the taint tracking technique and mark authorization codes with taint tags at the origin of the incoming SMS messages and propagate the tags through the system. Then, we apply security policies at the endpoints where the tainted authorization code is being sent out.



Architecture Diagram

V. MODULES

User Module

In this module, the user interface design to develop to add user details. Also, the user interface design to develop to add user details. This modules help to create saving bank accounts for costumers .it stores all the basic information of costumers. this feature is characterized by an authentication process that ensures that the user logging into the system is a registered customer. The user has to login with a valid username and password otherwise, access will be denied. This is a security mechanism to ensure that only authorized users can access the system

Money Transfer

In this module money can be transfer from one user account to another user account. This would be the main part of the system where a user inputs the type of transfer to be carried out (it could be payment of utility bills or a transfer of a certain amount as payment for goods bought or a number of other types). This would also include the part where the user would also input the amount to be transferred. From the transaction page, the system would get the input necessary for the transfer to occur.

Taint Tag Generation

The length of OTP is 4 and the set size of all possible characters in the OTP is 62. So the total number of possible sets of the pair of OTPs are 6212. Some of them are – [{456789, 456788}, {456789, 456789}] But the possible sets of equal pair of OTPs are:626. Some of them are – [{456788, 456788}, {456789, 456789}]. Hence the probability of collision of two OTPs is:

$$626 / 6212 = 1 / 626 = 1 / 56800235584 = 1.7605561\text{-}11$$

So the probability of two OTPs colliding are as less probable as the existence of your life on earth (Ratio of the number of years you will live to the number of years from the start of the universe and everything in existence). So yes, OTPs are way more secure than static passwords ! By using the Code Inject method, we are creating a onetime password on the user side (instead of server side) through a smartphone application. This means that users always have access to their one time password. So it prevents the server from sending a text message every time user tries to login. Also, the generated password changes after a certain time interval, so it behaves like a one time password.

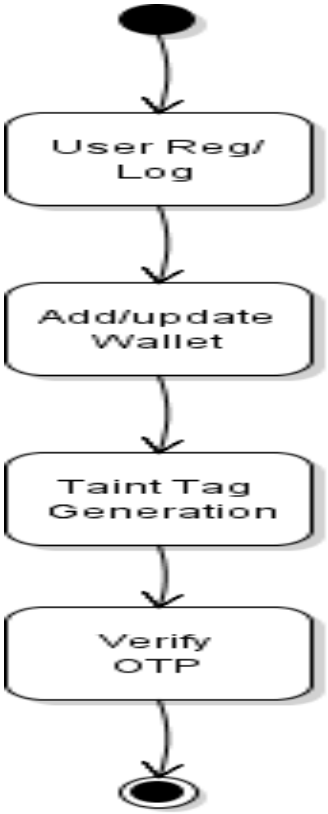
Identify The SMS Authorization Code

To identify an SMS authorization code and then apply the taint tag, our system has to determine whether an SMS message contains an authorization code. First, we need to decide when to identify the authorization code. Note that the Android SMS system mainly obtains SMS messages via SMS broadcasting or by reading from the SMS database. Therefore, we only need to determine whether an SMS message contains an authorization code before the SMS broadcasting and after the message is fetched from the SMS database. However, because the framework layer of Android will not have decoded the message content before the SMS broadcasting, it is difcult for us to recognize the authorization code by searching the content of the message. Therefore, we leverage the sender address of the SMS message to determine whether the message possibly contains an authorization code; if so, we mark it as a potential SMS authorization code. We maintain a list of sender addresses of SMS authorization codes, and we treat all the SMS messages that originate from these addresses as messages potentially containing SMS authorization codes. After the SMS message can be read from the SMS database, we search the content of the message to obtain the string pattern of the authorization code to determine whether the message contains an authorization code. After identifying an SMS message that contains an authorization code (or potentially contains such a code), we mark and track the message by adding a tag (or taint tag) to it (the marked message is called a taint source). It is important to note that if we add tags to all the variables in the system, it can better track the data, but the memory overhead will become a concern. We observe that an SMS message is generally stored

in a character or byte array; therefore, we only need to add tags in character and byte arrays. In addition, we add one tag for each array to reduce the memory overhead.

Verification of SMS Authorization code

The malicious apps could forward the stolen SMS authorization code through Sms Manager or network interface (taint sinks). Therefore, to catch such behaviors, we need to modify the corresponding interfaces in Android's framework layer and apply corresponding security policies. When it forwards the message through the Sms Manager, we extract the tag of the data to be sent. When it forwards the data through the network interface, it could be in several ways, e.g., by email, with HTTP request, and with TCP/UDP sockets. However, in any way, the network data will eventually be submitted to the system call of the kernel, which is performed through the Posix class. Therefore, we could detect and protect the SMS authorization data by monitoring the network-related operations in the Posix class. If we get a taint tag from a byte or character array, we may possibly get several values. Among these values, 0x00000000 (i.e., t_n) represents that the data do not contain any taint tags; 0x00000001 (i.e., t_p) represents that the data potentially contain an authorization code and that the data are directly obtained through SMS broadcasting; 0x00000002 (i.e., t_d) and 0x00000003 (i.e., t_d|p) represent that the data are fetched from the SMS database; and 0x000000007 (i.e., t_a|d|p) represents that the data are fetched from the SMS database and contain an authorization code. When the value is 0x00000001 or 0x00000007, we manipulate the data according to our pre-defined rules (e.g., prohibit sending, warn the user, or send a bogus value). It is important to note that if an app sends out data with a tag of 0x00000001 (i.e., t_p), we think that it is a dangerous operation. This is because the data are directly obtained through SMS broadcasting, and then, the app is attempting to send it out. This is a malicious action, as a benign app always fetches an SMS message from the SMS database and then sends it out. Code Tracker. Secure SMS endeavors to secure SMS messages by changing the applications getting grouping of instant messages in the framework so that the default SMS application can get the instant message. Then, at that point, it blocks the SMS broadcasting to forestall noxious applications from getting the message. Notwithstanding, Secure SMS just works in Android variants before 4.4.Different frameworks have likewise been proposed to forestall phishing messages.



Activity Diagram

VI. RESULT ANDDISCUSSION

The result that we are getting from our implemented technique as reflected in figure our implemented technique provides better results than previous technique term of accuracy**.** Also, the implemented technique provides better results when compared with other existing technique.

VII. CONCLUSION AND FUTURE WORK

We design a dynamic lightweight approach for tracking and protecting authorization codes in Android, called Code Tracker. Specifically, we leverage the taint tracking technique and mark authorization codes with taint tags at the origin of the incoming SMS messages and propagate the tags through the system. Then, we apply security policies at the endpoints where the tainted authorization code is being sent out. The total computation cost of the proposed scheme is reasonable and user-acceptable for online transactions, in that 2.82 ms at most are required for generating (and examining) each verification message. Furthermore, the security robustness against super-level malicious adversaries is guaranteed with the derived formal analysis. In brief, according to the analysis and evaluation results, we prove that the proposed transaction scheme is practical for common intelligent mobile devices (and mobile networks). In the future, the system performance may be further improved with

the enhancement of the security components adopted in the proposed scheme.

REFERENCES

[1] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security, 2003, pp. 452–473.

[2] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signature revisited," in Proc. 12th Australasian Conf. Inf. Security Privacy, 2007, pp. 308–322.

[3] K.-H. Yeh, "Cryptanalysis of Wang et al's certificateless signature scheme without bilinear pairings," Nat. Dong Hwa Univ., Hualien, Taiwan, Tech. Rep. NDHUIM-IS-2017-001, 2016.

[4] I. Lacmanovi, B. Radulovi, and D. Lacmanovi, "Contactless payment systems based on RFID technology," in Proc. 33rd Int. Conv. MIPRO, 2010, pp. 1114–1119.

[5] L. Mainetti, L. Patrono, and R. Vergallo, "IDA-Pay: An innovative micropayment system based on NFC technology for Android mobile devices," in Proc. 20th Int. Conf. Softw., Telecommun. Comput. Netw., 2012,

[6] B. Cha and J. W. Kim, "Design of NFC based micro-payment to support MD authentication and privacy for trade safety in NFC applica-tions," in Proc. 7th Int. Conf. Complex, Intell. Softw. Intensive Syst., 2013, pp. 710–713.

[7] E. Kazan and J. Damsgaard, "A framework for analyzing digital payment as a multi-sided platform: A study of three european NFC solutions," in Proc. Eur. Conf. Inf. Syst., 2013, Paper 155.

[8] W.-D. Chen, K. E. Mayes, Y.-H. Lien, and J.-H. Chiu, "NFC mobile payment with citizen digital certificate," in Proc. 2nd Int. Conf. Next Gener. Inf. Technol., 2011, pp. 120–126.

[9] E.-O. Blassa, A. Kurmusb, R. Molvac, and T. Strufed, "PSP: Private and secure payment with RFID," Comput. Commun., vol. 36, no. 4, pp. 468– 480, 2013.

[10] T. Ali and M. A. Awal, "Secure mobile communication in m-payment system using NFC technology," in Proc. 2012 Int. Conf. Informat., Electron. Visi on, 2012, pp. 133–136.