ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

SECURING SMS AUTHORIZATION CODES USING CODE TRACKER

D. Saritha Reddy

Assistant Professor, Dept. of Master of Computer Applications, Narayana Engineering College, Gudur, AP, India.

K. Sravani

PG Scholar, Dept. of Master of Computer Applications, Narayana Engineering College, Gudur, AP, India.

Abstract – Short Message Service (SMS) authorization codes play an important role in the application ecosystem, as a number of transactions (e.g., personal identification and online banking) require users to provide a code for authorization purposes. However, authorization codes in SMS messages can be stolen and forwarded by attackers, which introduces serious security concerns. In this paper, we propose a lightweight approach to track and protect SMS authorization codes. Specially, we leverage the taint tracking 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, inter-process communication mechanism, and le operations for secondary storage of SMS authorization codes to ensure that the taint tags cannot be removed. When the authorization code is sent out via either SMS messages or network connections (taint sinks), we extract the taint tag of the data and enforce pre-denied security policies to prevent the code from being leaked.

Keywords– Authorization Code, Short Message Service, Taint Tracking.

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. Leveraging SMS codes for authorization is convenient; however, it may present security concerns. If the code is stolen by attackers, it can cause financial losses to users. On the other hand, SMS-stealing malware is emerging [1], [2]. A research report from the Qihoo 360 company [3] revealed that 6.1% of mobile malware is stealing information. Among these information-stealing malware samples, 67.4% of them are targeting SMS messages. A research paper [4] noted that among the 49 malware families, 27 of them are harvesting user information, including user accounts and short messages. To this end, there is an urgent need to protect the SMS authorization codes in smartphonesthe READ_SMS permission to retrieve SMS messages from thedatabase.

We noted that a number of systems have been proposedto protect SMS authorization codes. For instance, TISSA [6]can provide null or bogus values instead of real data, whichavoids data leakage (including SMS authorization codes).However, TISSA is currently implemented on legacyAndroid's Dalvik runtime and not the

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

newly designed ARTruntime. Secure SMS [7] is another system used to protectSMS messages by changing the Android framework. In particular, when an SMS message arrives, SecureSMS searches the message text.

From another perspective, because SMS authorizationcodes are a type of sensitive data in smartphones, they canbe protected with the well-known taint tracking technique. Taint Droid [8] is such a system for real-time privacy monitoring that can be used to protect authorization codes. However, Taint Droid is implemented on the Dalvik virtual machineunder Android version 4.4 and has not been applicable forthe newly introduced ART runtime since Android 4.4.

In this paper, we propose Code Tracker, a lightweight approach to track and protect SMS authorization codes inAndroid SMS messages. Specifically, Code Tracker adds taint tags to mark the authorization code at the very beginning ofthe incoming SMS messages, and it modifies the related arraystructure, array operations, string operations, IPC (Inter Process Communication) mechanism, and file operations forthe secondary storage of SMS authorization codes to ensurethat the tags cannot be removed. Finally, when the authorization code is sent out (via either SMS or the network),Code Tracker extracts the tag of the data and checks with predefined security policies. By doing so, it prevents authorization codes from being stolen by attackers.

II. BACKGROUND WORK

Most SMS-stealing malware induces users to install it withphishing web links. Some malware masquerades as onlinebanking login interfaces to obtain a user's account information and password and then monitor SMS messages tosteal the SMS authorization code. In the following, we willdescribe several well-known malware families that steal SMSmessages.

A) MazarBOT

MazarBot [12] requests a number of permissions, including SEND_SMS, WRITE_SMS, INTERNET, RECEIVE_SMS, and READ_SMS, which are relevant to the network and SMS. Then,it is injected into the infected user's browser application andobtains the user's bank account and password. MaZarBOTalso registers the SMS receiving broadcast receiver(Android.provider.Teleph-ony.SMS_RECEIVED). When receiving an SMS message, MazarBOT first determines whetherthe message is a special instruction. If not, it will put thecontent of the message and the address in an Intent; then,the Intent will be submitted to an IntentService. In theIntentService, it packs the SMS message and SMS addressinto a JSON object, and it sends the message content to a special remote server, through an HTTP request. After obtaining the user's bankaccount and password and the SMS authorization code,it can steal the assets in the user's account, e.g., theirmoney

B) ANDROID.BANKBOT

Android.BankBot [13] is a malicious app that is remotelycontrolled with a C&C server. When a user opens his bankapplication, Android.BankBot loads a disguised bankingpage to cover up the original page. When the user logsinto their bank account, the attacker can intercept the user'sbank account number, password, and SMS

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

authorizationcode to bypass the two-factor authentication. This malwarefamily can forge a variety of bank login interfaces andconstantly update these fake bank pages through a remoteserver.

III. PROPOSED WORK

SYSTEM MODEL

The goal of our work is to track and protect authorizationcodes in SMS messages. To achieve this, there are severalchallenges that need to be addressed. First, we must determine whether a text message contains an authorization codeand then mark it with the taint tag. Second, the SMS authorization code could be processed in many locations, e.g.,it might be copied or passed to a new variable or be savedto the SMS database. The taint tags need to be reversed andpropagated in these scenarios. Third, we need to determinethe correct place to enforce pre-defined security policies toensure that the SMS authorization code cannot be stolen.

To overcome these challenges, we propose a lightweightapproach to track and protect SMS authorization codes, called Code Tracker. The overall design of Code Tracker is shown in Fig-1.

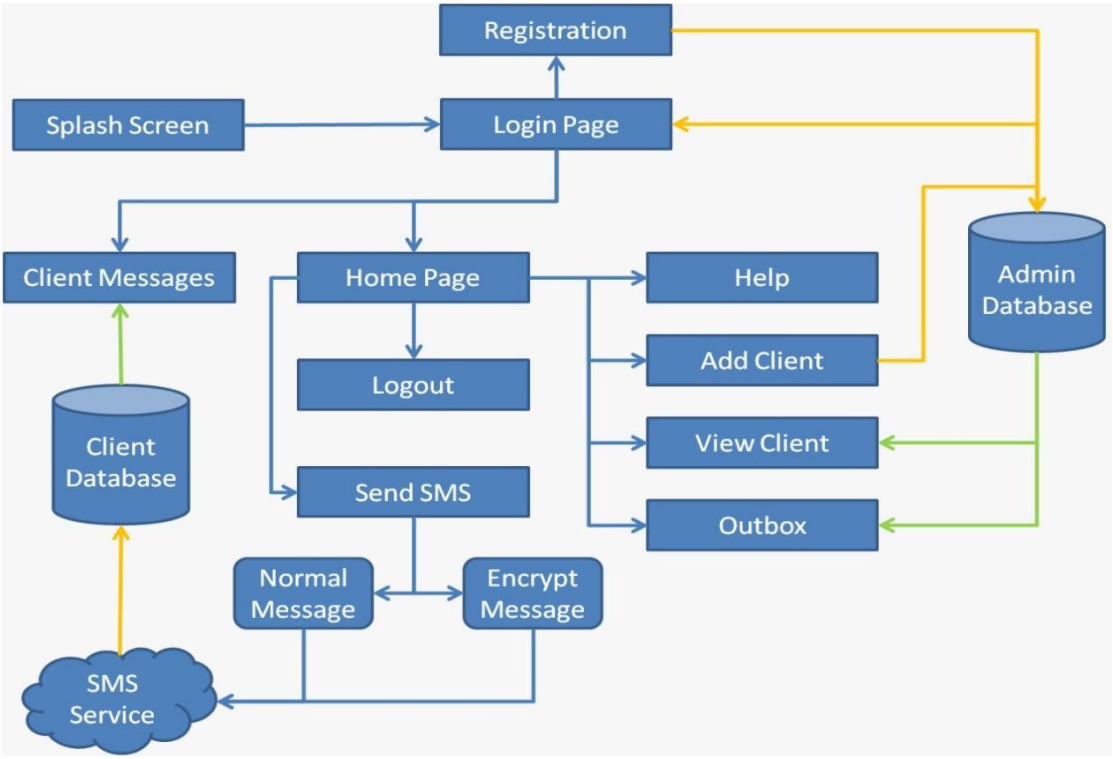

Figure. 1: System Overview

The modules of the system are as follows:

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

Admin Module

- ✓ In this module, admin login to the system. He able to perform sending SMS to the customer whose data is available. In this module admin chooses the contact information and send sms as either normal or secure type. Once he chooses the secure type the sms will be automatically encrypts and send it to the customer.

Client Module

- ✓ In this module, the client register and login to the system. In this module, client receives the secure messages in readable form. The secure sms automatically decrypted and stored in the system internal memory.

SMS Module

- ✓ This module is useful to both admin and clients. This module provides facility to send and receive the sms through mobile network.

Encryption Module

- ✓ In this module, sending sms will be checking if the sending sms is secure type then the sms will be automatically encrypted and send to intended client.

Decryption Module

- ✓ In this module, decryption process performed at client side. Once the secure sms is received in the client module then the app automatically decrypts the sms.

To process the Secure the SMS we use AES algorithm as follows

Algorithm used: AES (Advanced Encryption Standard)

Input: 128 bits block data, plain text.

Output: 128 cipher text Procedure: AES takes 10 rounds for 128 bits block size. This algorithm has four transition rounds they are: Substitute bytes, ShiftRows, MixColumns and AddRoundKey. For 128 bits it will take 16 bytes and performs transition rounds from 0-9 and last 10th round doesn't have [7][8]

MixColumns. The 16 bytes are arranged as two-dimensional array in hexa-decimal format which is called as state array. Each state array has 4 words as (w0, w1, w2, w3) which are used for rounds from 0-9. i.e. 44 words are used for 10 rounds. The next step is Substitute bytes where it uses an S-box to perform a byte-to-byte substitution of the block. Further it uses ShiftRows as a simple permutation. The output of ShiftRows is multiplied with actual plain input box this round is called MixColumns method. In AddRoundKey a simple bitwise XOR of the current block with a portion of the expanded key. The 1st round output is taken as 2nd round input and repeats the process until 10th round. The output of 10th round after AddRoundKey without MixColumns is the desired 128 bits Cipher Text.[12]

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

IV. RESULTS AND DISCUSSION

In this system we developed Secure SMS Authorization Code using Code Tracker to improve the user experience and Security. The following screens show that our system is more users friendly and efficient.

Fig-1Showing the use interface of the application when open it, afterinstalling into a mobile.

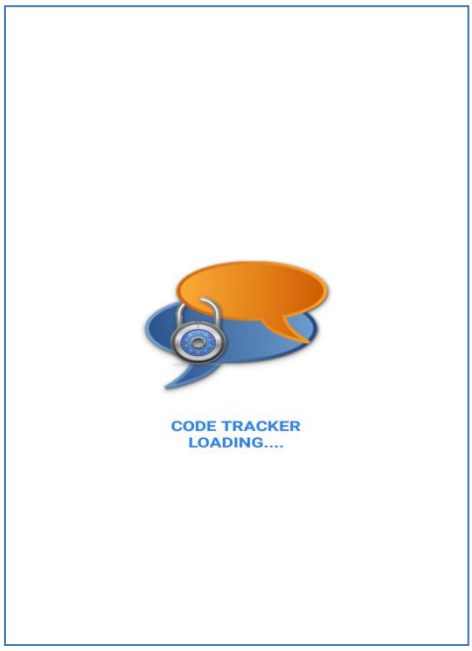

Figure. 2: Registration Page

Figure -3: Shows the login screen for this application,having two types of users, admin and client.

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

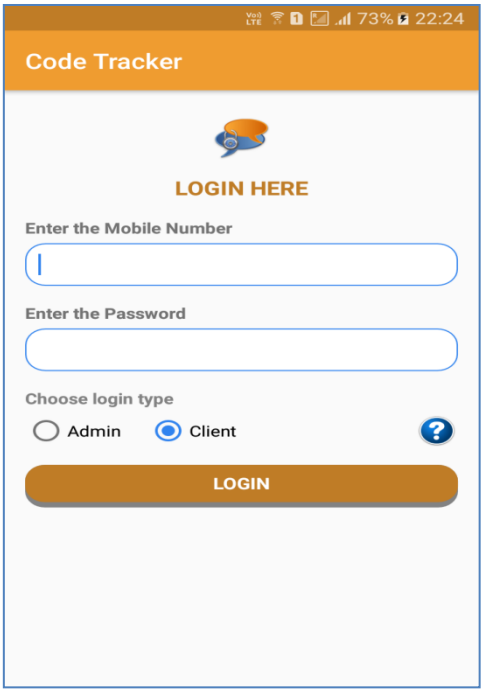

Figure 3: Login Page

This is the admin menu page.And there are some operations which are done by the admin shown in Fig-4.

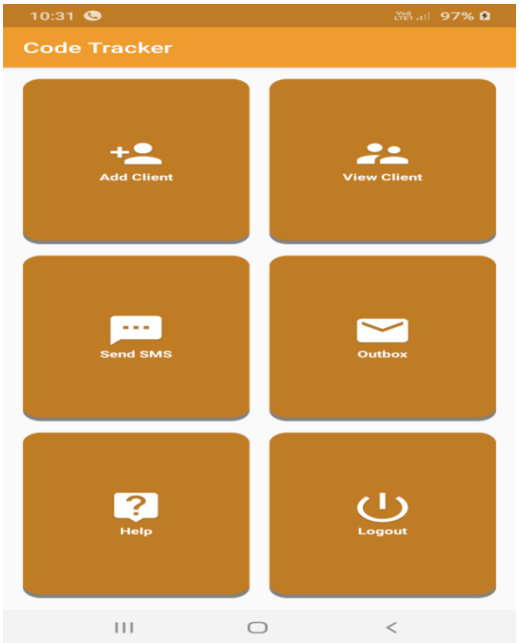

Figure. 4: Admin Menu Page

The admin can add the clients with the details listed in the page as shown in the Fig-5.

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

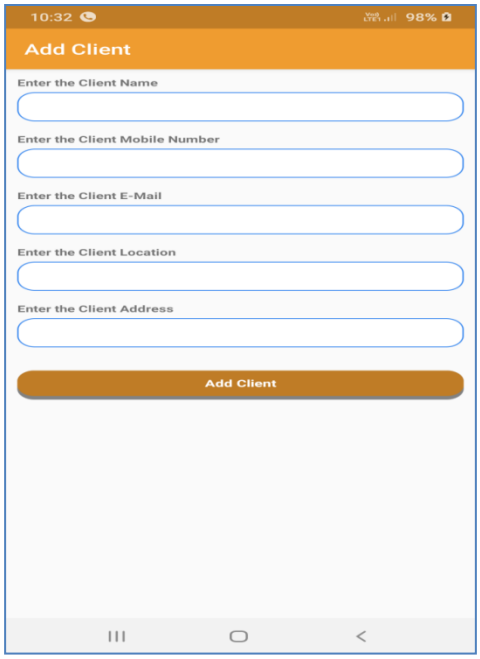

Figure. 5: Add Client

Sending the SMS to a particular client in a normal or in an encrypted form as below screen.

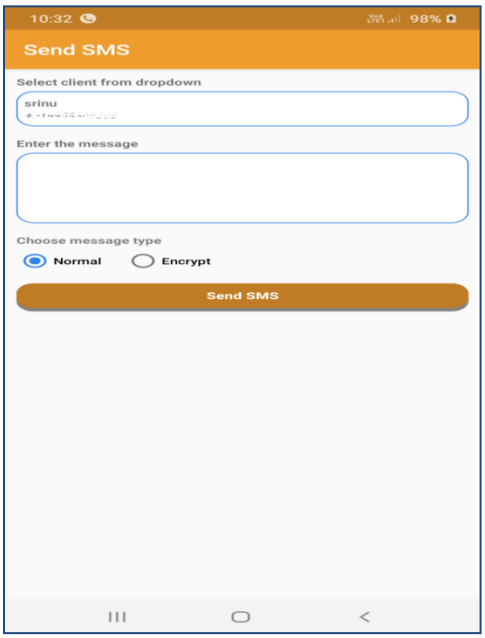

Figure. 6: Send SMS

This screen showing the messages that are sent by the admin to the clients in Fig-7.

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

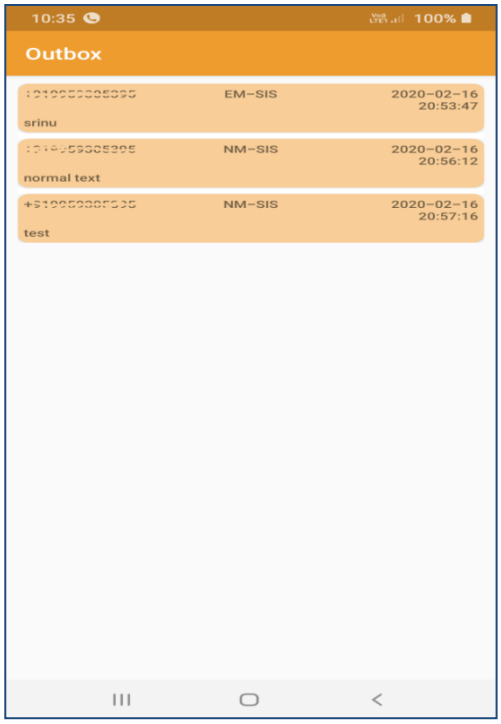

Figure 7: SMS Outbox

This screen showing the messages that are received from the different clients in Fig-8.

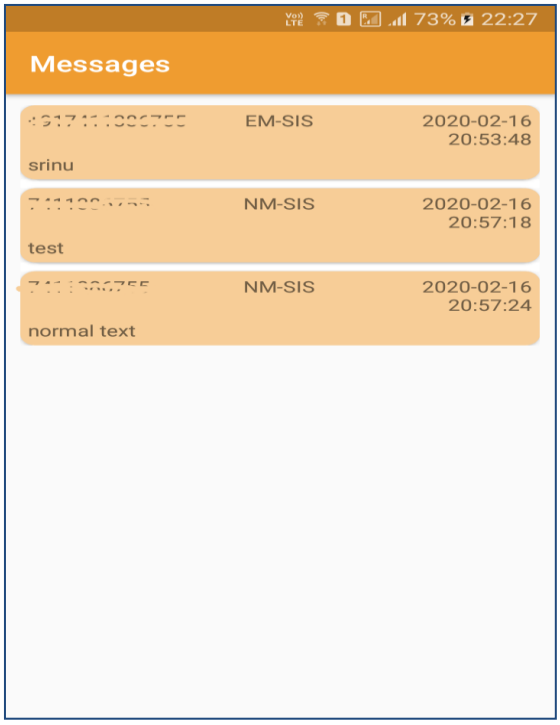

Figure 8 Client Message Box

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

V. CONCLUSION

In this project, we design a 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.In future, we consider the performance and computational overhead at server side and we proposed proxy based outsourced techniques.

REFERENCES

1. We Steal SMS: An Insight Into Android. KorBanker Operations.Accessed: Dec. 26, 2017. [Online]. Available: https://www.fireeye.com/blog/threat-research-/2014/09/we-steal-sms-an-insight-into-andro-idkorbanker-operations.html
2. SophosLabs Report Explores Mobile Security Threat Trends, RevealsExplosive Growth in Android Malware. Accessed: Dec. 26, 2017. [Online].Available: https://news.sophos.com/-en-us/2014/02/24/sophoslabs-reportexplores-mobile-security-threat-trends-reveals-explo-sive-growth-inandroid-malware/
3. Special Report on Android Malware in 2016. Accessed: Dec. 26, 2017.
4. [Online]. Available: http://zt.360.cn/1101061855-.php?dtid=1101061451&did=490301065
5. Y. Zhou and X. Jiang, ‘‘Dissecting Android malware: Characterization andevolution,’’ in Proc. IEEE Symp. Secur. Privacy, May 2012, pp. 95–109.
6. Android 4.4. Accessed: Dec. 26, 2017. [Online]. Available:https://developer.andro-id.google.cn/about/versions/kitkat.html
7. Shanthini, M., P. Rajasekar, and H. Mangalam. "Design of low power S-box in Architecture Level using GF." International journal of engineering research and general science (IJERG), pp-1–9 (2014).
8. Rajasekar, P., and H. Mangalam. "Design of Low Power Optimized MixColumn/Inverse MixColumn Architecture for AES." International Journal of Applied Engineering Research 11 (2016): 922-926
9. Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, ‘‘Taming informationstealing smartphone applications (on Android),’’ in Proc. 4th Int. Conf.Trust Trustworthy Comput., 2011, pp. 93–107.
10. D. Kim and J. Ryou, ‘‘SecureSMS: prevention of SMS interceptionon Android platform,’’ in Proc. 8th Int. Conf. Ubiquitous Inf. Manage.Commun., 2014, p. 32.
11. W. Enck et al., ‘‘TaintDroid: An information-flow tracking system forrealtime privacy monitoring on smartphones,’’ ACM Trans. Comput. Syst.,vol. 32, no. 2, p. 5, Jun. 2014.
12. Rajasekar, P. and Mangalam, D. (2016) Efficient FPGA implementation of AES 128 bit for IEEE 802.16e mobile WiMax standards. Circuits and Systems, 7, 371-380. doi: 10.4236/cs.2016.74032.
13. M. Sun, T. Wei, and J. C. S. Lui, ‘‘TaintART: A practical multi-levelinformation-flow tracking system for Android runtime,’’ in Proc. ACMSIGSAC Conf. Comput. Commun. Secur., 2016, pp. 331–342.
14. M. Backes, S. Bugiel, O. Schranz, P. von Styp-Rekowsky, and S. Weisgerber, ‘‘ARTist: The Android runtime instrumentation and security toolkit,’’in Proc. IEEE Eur. Symp. Secur. Privacy, Apr. 2017, pp. 481–495.

ISSN: 2278-4632
Vol-10 Issue-5 No. 1 May 2020

Author's Profile:

D. Saritha Reddy has received her M.Tech degree in *Computer Science* from *Acharya Nagarjuna University, Guntur* in 2010and pursuing Ph.D with area *Computer Networks.* At present she is working as *AssistantProfessor* in Narayana Engineering College, Gudur, AndhraPradesh, India.

K. Sravani has received her B.Sc degree in *Computer Science* from SV Arts & Science Degree College, Gudur affiliated to *VikramaSimhapuri University*, *Nellore* in 2017 and pursuing PG degree in *Master of Computer Applications* (MCA) from Narayana Engineering College, Guduraffiliated to *JNTU, Ananthapur,* AndhraPradesh, India.