Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.199

IJCSMC, Vol. 9, Issue. 3, March 2020, pg.01 – 09

ENHANCED PRIVACY AND SECURITY IN OTP GENERATION TECHNIQUE AGAINST OTP STEALING ATTACK

Mrs. B.SATHYABAMA[1]; Mr. M.VINOTHKUMAR[2]
[1]Assistant Professor, [2]Final MCA Student
PG and Research Department of Computer Application
Hindusthan College of Arts and Science
Coimbatore, Tamil Nadu, India
[1]msathyaimpossible@gmail.com

Abstract- Mobile apps have brought tremendous impact to businesses, social, and lifestyle in recent years. Various app 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 unauthorized apps. More researches are carried out to secure the OTPSMS from the unauthorized app, and then also can't protect the OTP from anonymous user apps. To solve this problem and protect the SMS, we proposed the Code Inject method, which inject the code into the OTP. The OTP sending through the SMS is not the original OTP, it is modified OTP by Code Inject method. The authorized apps only have the technique to get the original OTP from the modified OTP. With the modified OTP, the app can't transact the amount from bank. Code Inject method consists of Operation function choose and the critical number generate, this will generate the modified OTP for the original OTP.

Keywords– Hacking, OTPSMS, Secured Online Transaction.

I. INTRODUCTION

SMS authentication systems completely rely on the security provided by the cellular network. In this subsection, we discuss the security of cellular networks and vulnerabilities that allow SMS messages to be intercepted over-the-air. Cellular operators use the GSM, 3G, and CDMA technologies to provide mobile services such as SMS messages. However, GSM is insecure due to several vulnerabilities such as a lack of mutual authentication and weak encryption algorithms.

In particular, there is no mutual authentication between mobile phones and base stations in GSM networks, hence fake base station attacks are possible. These are generally used to intercept mobile traffic

(including SMS) of the end users. GSM uses different algorithms to encrypt wireless communication between mobile phones and base stations. The algorithm is weak and can be broken in a few seconds. Recent advances in GSM research show that there is no end-to-end security. It is possible to capture GSM traffic using low cost devices and decrypt the traffic due to weak algorithms. The communication between mobile phones and base stations can be eavesdropped and decrypted using protocol weaknesses.

A SIM card, which holds the telephone number you defined for receiving authorization SMS codes from the bank (hereinafter only SIM card"), is the most important component for receiving authorization SMS codes. Avoid lending your mobile phone and SIM card to third persons, even if you can observe how they use the phone and the SIM card is replaced. Use your PIN code to protect your telephone from unauthorized use by a third person if your phone could ever be outside of your supervision or control. Keep this code secret and do not provide it to any third person or lose it. An authorization code delivered to you by the bank must be kept secret and do not provide the SMS authorization code to anyone else.

II. OFFERED and ANTICIPATED SYSTEM

OFFERED SYSTEM

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.

Another system used to protect SMS messages by changing the Android framework. In particular, when an SMS message arrives, Secures SMS searches the message text.

Disadvantages:

- ✓ Malicious apps could intercept SMS messages to retrieve authorization codes and then block the SMS broadcasting stealthily without informing users.
- ✓ Malicious apps are unable to block SMS broadcasting, and the system SMS app will get the SMS messages.

ANTICIPATED SYSTEM

- ✓ 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. Underlying framework; it cannot be transparently supported as a user-level solution.

Advantages:

Dynamic lightweight approach for tracking and protecting authorization codes in Android Mobile.

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.

III. IMPLEMENTATION

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 numbers 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 = 1.7605561-11

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 difficult 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 SmsManager 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 SmsManager, 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.

IV. INPUT DESIGN AND OUTPUT DESIGN

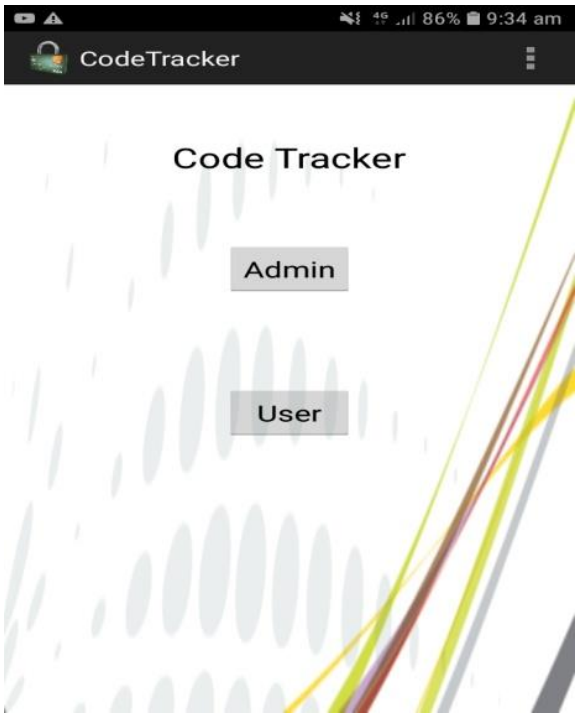

Fig1: Home page

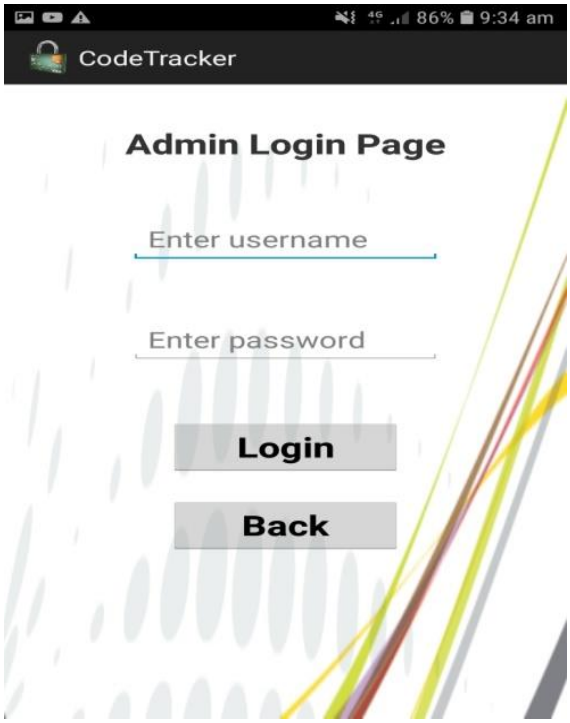

Fig2 : Admin Login Page

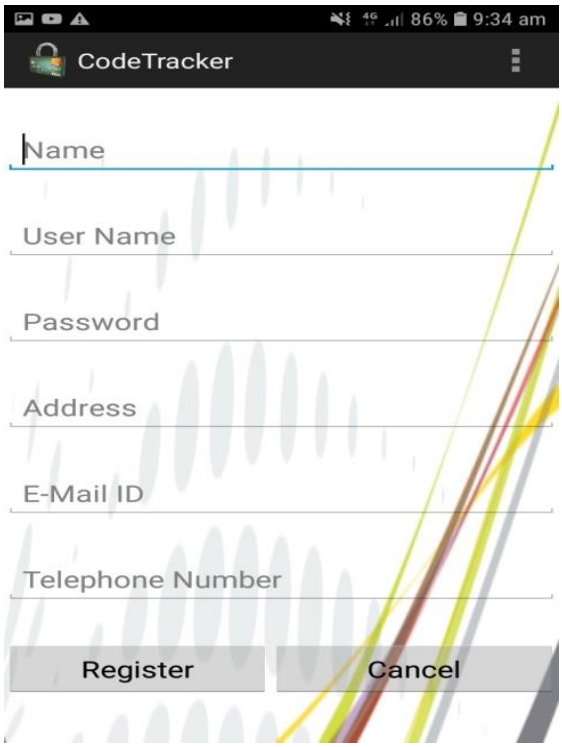

Fig3: User Register

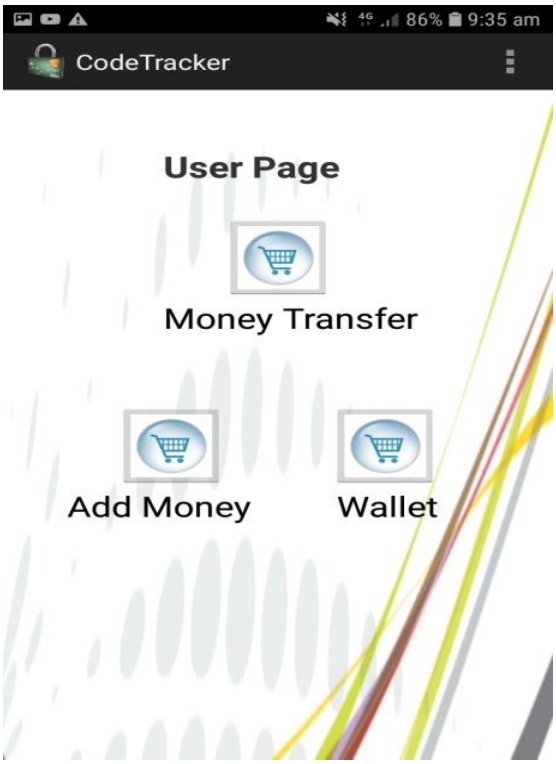

Fig4 : User Page

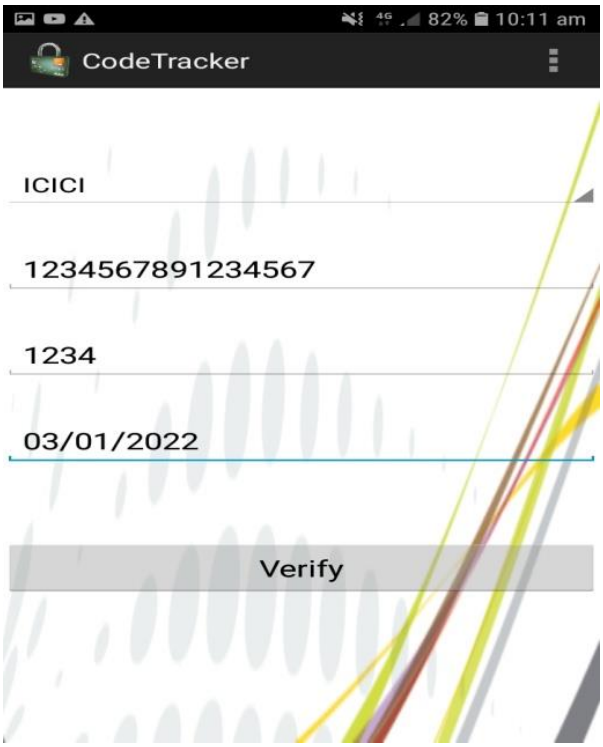

Fig5 : Bank Details

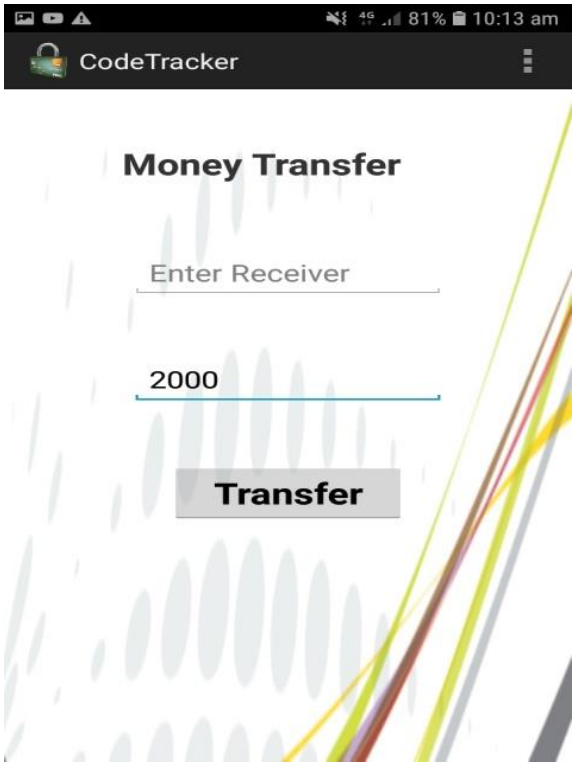

Fig6 : Money Transfer

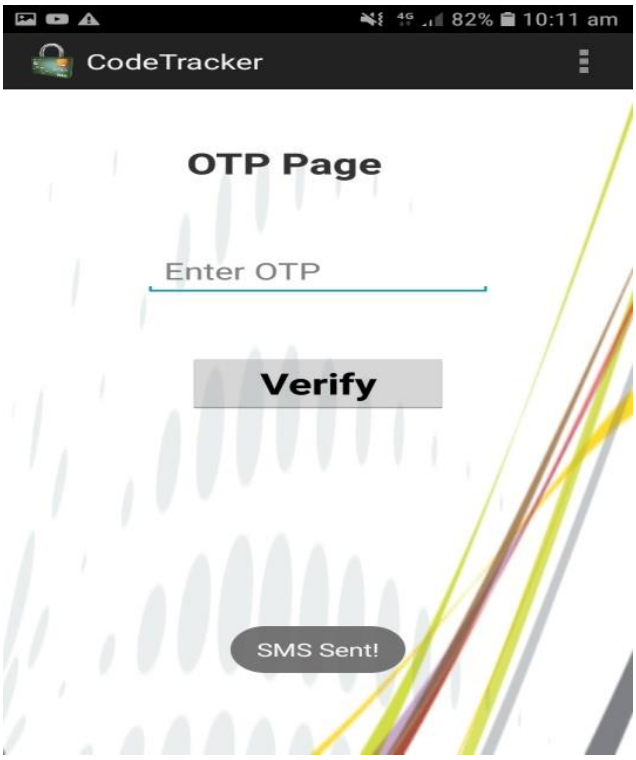

Fig7 : OTP Page

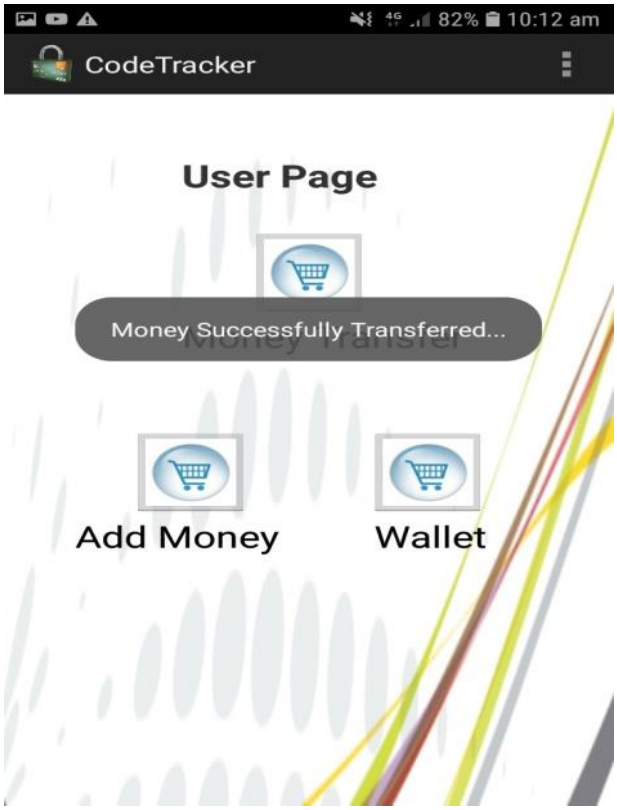

Fig8 : Money Successfully Transferred

V. CONCLUSION

In this paper, we proposed the Code Inject method, which inject the code into the OTP. The OTP sending through the SMS is not the original OTP, it is modified OTP by Code Inject method. The authorized apps only have the technique to get the original OTP from the modified OTP. With the modified OTP, the app can't transact the amount from bank.

Code Inject method consists of Operation function choose and the critical number generate, this will generate the modified OTP for the original OTP. Evaluation results on real-world data sets and larger extensive datasets have demonstrated the cost of preserving privacy in Android can be reduced significantly with our approach over existing ones where all data sets are encrypted our experiments demonstrate its effectiveness and practicality. The performance measurements show that our system has a low performance overhead.

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.

