International Journal of Technology and Engineering System (IJTES)
Vol 7. No.2 2015 Pp. 124-129

available at : www.ijcns.com
ISSN: 0976-1345

AUTHENTICATION PROTOCOLS THAT FOR NET BANKING SECURITY

[1]PRIYADHARSINI S, BELINA N [2]
([1]M.E. II Year student), ([2]Associate professor) Dept of Computer Science Engineering
Sri Ram Engineering College Perumalpattu, Chennai.
[1]priyadharsini72@yahoo.in, [2]belinafeli@gmail.com

ABSTRACT:
Key logging or keyboard capturing, is the action of recording (or logging) the keys struck on a keyboard, naturally in a hidden mode so that the person using the keyboard is unaware that their actions are being monitored. It also has very legally used in studies of interaction between human and computer. There are numerous key logging methods, ranging from software and hardware-based approaches to acoustic examination. It involving human in authentication protocols, while hopefully, is not easy because of their limited ability of calculation and memorization. Demonstrating visualization design to be careful and can augment not only the security but also the usability of substantiation. Proposed two visual authentication protocols: that is in the name of one-time-password (OTP) protocol, and second protocol is password-based authentication protocol. The loom for real-world consumption : we were able to achieve a high level of usability while fulfilling rigorous security requirements.
Keywords: **Key logger, Authentication, Malicious attacks, Android smart phones, Keyboard.**

INTRODUCTION

Threats against electronic and financial services can be classified into two major classes: credential stealing and channel breaking attacks [2]. Credentials such as users' identifiers, passwords, and keys can be stolen by an attacker when they are poorly managed. For example, a poorly managed personal computer (PC) infected with a malicious software (malware) is an easy target for credential attackers. On the other hand, channel breaking attacks—which allow for eavesdropping on communication between users and a financial institution—are another form of exploitation.While can be present both in personal computers and public cubicle; there are always cases where it is necessary to perform financial transactions using a public computer although the biggest concern is that a user's password is likely to be stolen in these computers. Even bad, key loggers, frequently core kitted, are hard to spot since they will not show up in the task manager process list.

classical channel breaking attacks can be prevented by the proper usage of a security channel such as IPSec [13] and SSL (secure sockets layer) , recent channel breaking attacks are more challenging. Indeed, "key logging" attacks— or those that utilize session hijacking, phishing and pharming, and visual fraudulence— cannot be addressed by simply enabling encryption. Chief among this class of attacks are key loggers. A key logger is a software designed to capture all of a user's keyboard strokes, and then make use of them to impersonate a user in financial transactions. For example, whenever a user will type their password in a sign in box of bank's account, the key logger capture the password. The peril of

To mitigate the key logger attack, virtual or onscreen keyboards with random keyboard arrangements are widely used in practice. Both techniques, by rearranging alphabets randomly on the buttons, can frustrate simple key loggers. Unfortunately, the key logger, which has control over the entire PC, can easily capture every event

video buffer to create a mapping between the clicks and the new alphabet. Another mitigation technique is to use the keyboard hooking prevention technique by perturbing the keyboard interrupt vector table [5]. However, this technique is not universal and can interfere with the operating system and native drivers. Our approach to solving the problem is to introduce an intermediate device that bridges a human user and a terminal. Every interaction between the user and an intermediate helping device is visualized using a Quick Response (QR) code. The goal is to keep user-experience the same as in legacy authentication methods as much as possible, while preventing key logging attacks. Thus, in our protocols, a user does not need to memorize extra information except a traditional security token such as password or PIN, and unlike the prior literature that defends against should-surfing attacks by requiring complex computations and extensive inputs. More specifically, our approach visualizes the security process of authentication using a smartphone aided augmented reality. To securely implement visual security protocols, a smartphone with a camera is used. Instead of executing the entire security protocol on the personal computer, part of security protocol is moved to the smartphone. This visualization of some part of security protocols enhances security greatly and offers protection against hard-to-defend against attacks such as malware and key logging attack, while not degrading the usability. However, we note that our goal is not securing the authentication process against the shoulder surfing attacker who can see or compromise simultaneously both devices over the shoulder, but rather to make it hard for the adversary to launch the attack.

II. METHODOLOGY

1. System Model

Our system model consists of four different entities (or applicants), which are a consumer, a smartphone, a consumer's terminal, and a server. The consumer is an ordinary human, partially by human's inadequacy, including limited potential of performing multifaceted computations or remembering complicated cryptographic officials, such as cryptographically strong keys. With a consumer's terminal like as a desktop computer or a laptop, the consumer can log in a server of a financial organisation (bank) for financial transactions. Also, the consumer has a smartphone, the third system unit, which is prepared with a camera and lay up a public key certificate of the server for digital signature proof.

Finally, the server is the last system unit, which fit in to the financial organization and performs back-end operations by interrelating with the consumer (terminal or smartphone) on behalf of the bank. Assuming a smartphone unit in our system is not a fantastic statement, since most cell phones nowadays qualify (in terms of routing and imaging facilities) to be the device used in our work. In our system, we assume that there is no direct channel between the server and the smartphone. Also, we note that in most of the protocols proposed here, a smartphone does not use the message communication channel—unless otherwise is openly stated—so a smartphone can be replaced by any device with a camera and some proper processing power such as a digital camera, a portable music player with camera (iPod touch, or mobile gadget with the aforementioned capabilities) or a smart watch/glasses.

2. Trust and Attacker Models

For the trusted entities in our system, we assume the following: First, we assume that the channel between the server and the user's terminal is secured with an SSL connection, which is in fact a very sensible supposition in most electronic banking systems. Second, we presume that the server is protected by every means and is immune to every harass by the attacker; hence the attacker's concern is not breaking into the server but attacking the user. Finally, with respect to the key logger attack, we assume that the key logger always resides on the terminal. As for the attacker model, we assume a malicious attacker with high incentives of breaking the security of the system. The assailant is able of doing any of the following:

The assailant has a full control above the terminal. Thus,

- While exist in a consumer's terminal, the mugger can imprison consumer's credentials such as a password, a private key, and OTP (one time password) token string.
- The assailant can deceive a consumer by showing a indisputable-looking page that in fact transfers money to the attacker's account with the captured diplomas that he obtained from the compromised terminal.
- Or, just after a consumer successfully gets genuine with a valid credential, the hacker can seize the authenticated session.

The attacker is able of creating a fake server to start on phishing or pharming attacks. For the smartphone in Protocol 1, we suppose that it is always trusted and resistant to compromise, which means no malware can be installed on it. Notice that this supposition is in line with other suppositions made on the smartphone's trustworthiness when used in similar protocols to those

presented in this paper. We, however, note that relaxing this assumption still could provide a certain level of security with Protocol 2. Protocol 2 uses two factors (password and the smartphone), and thus, the supositions can be tranquil so that not only the terminal but also smartphone could be compromised (one of them at a time but "not both together"). The non-simultaneous concession supposition obviously excludes the shoulder-surfing attacker. In our protocols, we also assume several cryptographic primitives. For example, in all protocols, we assume that a user has a pair of public/private keys used for message signing and verification. In Protocol 1, we assume that the server has the capability of generating one time pads, used for authentication. In Protocol 2, we assume users have passwords used for their authentication. Notice that these assumptions are not far-fetched as well, since most banking services use such cryptographic credentials. For example, with most banking repairs, the use of digital certificates given by the bank is very common. In addition, the use of such cryptographic credentials and upholding them on a smartphone does not require any technical background at the user side, and is suited for wide variety of users. Further details on these credentials and their use are explained along with the specific protocol where they are used in this paper.

3. Linear and Matrix Barcodes

A barcode is an optical machine-readable illustration of information, and it is broadly used in our daily life as it is attached to all types of products for recognition. In this a nutshell, barcodes are of two types: linear barcodes and matrix (or two dimensional, otherwise known as 2D) barcodes. While linear barcodes- shown in figure I(a)- have a limited capacity, which belongs on the coding technique used that can range from 10 to 22 characters.

2D barcodes shown in figure I(b) and figure I(c)- have higher capacity, which can be more than 700 characters. For example, the QR code a broadly used 2D barcode- can grasp 7,089 numeric, 4,296 alphanumeric, or 2,953 binary characters [4], making it a very good ability in high candidate for storing plain and encrypted contents alike.

Figure I. Three different barcodes encoding the statement

"Virtual reality". (a) is a linear code(barcode 128), and (b) and (c) are matrix barcodes (of the QR code standard). Both linear and matrix codes are used in industries and more popular, but not limited to automotive industries, manufacturing of electronics components and bottling industries, and among many others. This model of advertisement and having of using barcodes in areas that are in touch with users- created the need for barcode's scanners developed specifically for smartphones. The creation of barcode scanners for smartphones such as iPhone and android phones.

4. Keylogging resistant visual authentication protocols

In this section, we describe two protocols for user authentication with visualization. Before getting into the details of these protocols, we evaluate the documents for algorithms that are used in our protocols as building blocks. Our system will make use the following algorithms:

1. Encrk(_): an encryption algorithm which takes a key k and a message M from set M and outputs a ciphertext C in the set C.
2. Decrk(_): a decryption algorithm which takes a ciphertext C in C and a key k, and outputs a plaintext (or message) M in the set M.
3. Sign(_): a signature generation algorithm which takes a private key SK and a message M from the set M, and outputs a signature.
4. Verf(_): a signature verification algorithm which takes a public key PK and a signed message (M; _), and returns valid or invalid.
5. QREnc(_): a QR encoding algorithm which takes a string S in S and outputs a QR code.
6. QRDec(_): a QR decoding algorithm which takes a QR code and returns a string S in S.

Any public key encryption scheme with IND-CCA2 (Indis-tinguishability against Adaptive Chosen Ciphertext Attacker) security would be good for our application. A public key encryption scheme with IND-CCA2 adds

random padding to a plaintext, which makes the ciphertext different whenever encrypted, even though the plaintext is the same [26]. This restriction on the type of the used public key encryption scheme will prevent an attacker from checking whether his guess for the random layout is right or not. Thus, the security of the scheme is not dependent on the number of possible layouts but the used encryption scheme. If no such encryption is used, the adversary will be able to figure out the layouts used because he will be able to verify a brute-force attack by matching all possible plaintexts to the corresponding ciphertext. On the other hand, when such encryption is used, the 1-1 mapping of plaintext to cipher text does not hold anymore and launching the attack will not be possible at the first place. Also, any signature scheme with EUF-CMA (existential-unforgeability against adaptive chosen-message attacker) can be used to serve the purpose of our system. For details on both notions of security, see [11]. In particular, and for efficiency reasons, we recommend the short signature in [6].

A .Authentication With Random Strings

In this section, we introduce an authentication protocol with a one time password (OTP). The following protocol (referred to as Protocol 1 in the the rest of the paper) relies on a strong assumption; it makes use of a random string for authentication. The protocol works as follows:

1) The consumer connects to the server and sends his ID.
2) The server checks the ID to retrieve the user's publickey (PKID) from the database. The server then picks a fresh random string OTP and encrypts it with the public key to obtain EOTP = EncrPKID (OTP).
3) In the terminal, a QR code QREOTP is displayed prompting the user to type in the string.
4) The user decodes the QR code with EOTP = QRDec(QREOTP). Because the random string is encrypted with user's public key (PKID), the user can read the OTP string only through her smartphone byOTP =Decrk(EOTP) and type in the OTP in the terminal with a physical keyboard.
5) The server make sure the result and if it matches what the server has sent earlier, the consumer is realistic. Otherwise, the user is denied. In this protocol, OTP is any combination of alphabets or numbers whose length is 4 or more depending on the security level required.

B.An Authentication Protocol with Password and Randomized Onscreen Keyboard

Our second protocol, which is referred to as Protocol 2 in the rest of this paper, uses a password shared between the server and the consumer, and also a randomized keyboard. A high-level event-driven code describing the protocol. The protocol works as follows:

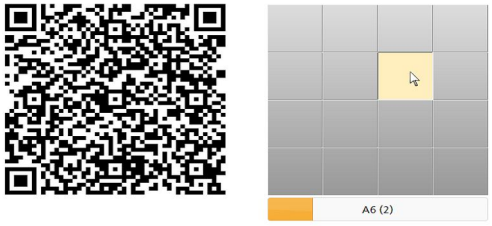
1) The consumer connects to the server and sends her ID.
2) The server checks the received ID to retrieve the user's public key (PKID) from the database. The server prepares _, a random permutation of a keyboard arrangement, and encrypts it with the public key to obtain EKBD = EncrPKID (_). Then, it encodes the ciphertext with QR encoder to obtain QREKBD = QREnc(EkID (_)). The server sends the result with a blank keyboard.
3) In the user's terminal, a QR code (QREKBD) is displayed together with a blank keyboard. Because the onscreen keyboard does not have any alphabet on it, the user cannot input her password. Now, the user executes her smartphone application which first decodes the QR code by applying QRDec(QREKBD) to get the ciphertext (EKBD). The ciphertext is then decrypted by the smartphone application with the private key of the user to display the result (_ = DecrSKID(EKBD)) on the smartphone's screen.
4) When the user sees the blank keyboard with the QR code through an application on the smartphone that has a private key, alphanumerics appear on the blank keyboard and the user can click the proper button for the password. The user types in her password on the terminal's screen while seeing the keyboard layout through the smartphone. The terminal does not know what the password is but only knows which buttons are clicked. Identities of the buttons clicked by the user are sent to the server by the terminal.
5) The server make sure whether the password is correct or not by confirming if the correct buttons have been clicked.

III. RESULTS AND DISSCUSION

Some of the technical issues in the two protocols that we have introduced in the previous sections call for further discussion and clarification. In this section, we elaborate on how to handle quite a few issues related to our protocols, such as session seizing, transaction confirmation, and securing transactions.

(a) QR Code Scanning (before) (b)QR Code Scanning (after)

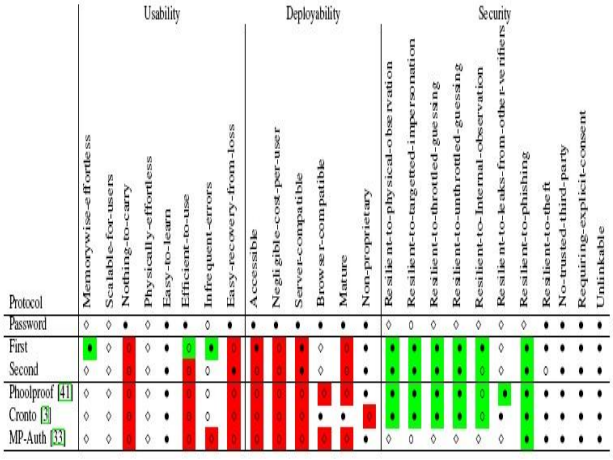
(c) Keyboard on Terminal (d) Keyboard on Smartphone



(e) Clicking Password on Blank Keyboard

Figure II. Photographs of the prototype we have developed to demonstrate our authentication protocols. (a) and (b) show the moments of a QR code scanning of a keyboard layout. (c) shows the blank keyboard shown at the terminal (on LCD screen). (d) shows the decoded randomized layout of the keyboard obtained from the QR code after decryption as viewed on smartphone. Note that the yellow square on which the mouse cursor is hovering in the terminal is shown through the smartphone to assist user's input. (e)shows that a user is clicking the password on the blank keyboard while seeing numbers through the smartphone.

A.key loggers



Key loggers are popular and broadly reported in more contexts. In our proposed protocols, input is expected by the consumer, and in each and every protocol one or another type of input is obligatory. Our protocols—while designed with the confines and shortcoming of consumers in mind, and aim at easing the confirmation process by means of hallucination—are aimed explicitly at defending against the key logger harasses. Here, we further detailed on the likely of using key loggers as an harass, and the way they impact each of the two protocols.

*Protocol 1***.** Authentication in this protocol is solely based on a random string generated by the server. The random string is encrypted by the public key of the user, and verified against her private key. The main purpose of using OTP is that it is for one time utilize. Accordingly, if the key logger is installed on the terminal, the hacker clearly will be able to know the OTP but will not be able to reuse it for future authentication. Alternatively, a key logger installed on the smartphone will not be able to log any credentials, since no credentials are input on the smartphone. It is worth noting that the hacker may try to block users from being authenticated and reuse the OTP immediately.

Protocol 2. In the second protocol, a blank keyboard is posted on the terminal whereas a randomized keyboard with the alphanumerics on it is posted on the smartphone. Because the protocol does not require the user to do any keyboard input on the smartphone side, the protocol is immune against the Key logger attack. The user just checks the keyboard layout on the phone and there is no input from a user. Obviously, the terminal might be compromised, but the key logger will be able to only capture what keystrokes are used on the blank keyboard. Thus, the key logger will not be able to know which alphanumeric characters are being clicked.

TABLE I. A comparison with other work based on their usability, deployability, and security. Comparisons are in relation with password based authentication, where stands for the case where metric doesn't apply,
● stands for meeting the metric, ○ stands for the metric can be made to do in the design. Green and red are indicators for better and worse than the case of the password based authentication respectively. friendliness of authentication protocols. Moreover, we have shown two realizations of protocols that not only improve the user experience but also resist challenging harasses, such as the key logger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box smart phone devices. We developed Android

application of a prototype of our protocol and demonstrate its feasibility and potential in real-world deployment and operational settings for user authentication. Our work indeed opens the door for several other directions that we would like to investigate as a future work. First of all, our plan is to implement our

IV. CONCLUSION

In this paper, we proposed and analyzed the use of user driven visualization to improve security and user protocol on the smart glasses such as the google glass, and conduct the user study. The above Table I gives the comparison on work group based on usability, deployability and security. Second we plan to investigate the design of other protocols with more stringent performance requirements using the same tools provided in this work.

V. REFERENCES

[1] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 553–567. IEEE, 2012.

[2] J. Brown. Zbar bar code reader, zbar android sdk 0.2. http://zbar. sourceforge.net/, April 2012.

[3] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu. Gangs: gather, authenticate 'n group securely. In J. J. Garcia-Luna-Aceves, R. Sivakumar, and P. Steenkiste, editors, MOBICOM, pages 92–103. ACM, 2008.

[4] S. Chiasson, P. van Oorschot, and R. Biddle. Graphical password authentication using cued click points. In Proc. of ESORICS, 2008.

[5] D. Crockford. The application/json media type for javascript object notation (json). http://www.ietf.org/rfc/rfc4627.txt?number=4627, July 2006.

[6] D. Davis, F. Monrose, and M. Reiter. On user choice in graphical password schemes. In Proc. of USENIX Security, 2004.

[7] N. Doraswamy and D. Harkins. IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall, 2003.

[8] M. Farb, M. Burman, G. Chandok, J. McCune, and A. Perrig. Safeslinger: An easy-to-use and secure approach for human trust establishment. Technical report, CMU, 2011.

[9] H. Gao, X. Guo, X. Chen, L. Wang, and X. Liu. Yagp: Yet another graphical password strategy. In Proc. of ACM ACSAC, pages 121–129, 2008.

[10] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal, 1988.

[11] Google. Android. http://www.android.com/, 2011.