International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 11, November 2015
Available at http://internationaljournalofresearch.org

Contemporary Visual Authentication Protocols to Tackle Key Logging Issues

Jahoor Sulthana Patan[1]& Md.Amamatulla[2]

[1]M.Tech (CSE), Nimra Institute of Science and Technology, A.P., India.
[2]Head of the Department, Dept. of Computer Science & Engineering, Vizag Institute of Technology, A.P., India.

Abstract —
An authentication protocol is a type of cryptographic protocol with the purpose of authenticating entities wishing to communicate securely.The design of secure authentication protocols is quite challenging, considering that various kinds of root kits reside in PCs (Personal Computers) to observe user's behavior and to make PCs untrusted devices. Involving human in authentication protocols, while promising, is not easy because of their limited capability of computation and memorization. Therefore, relying on users to enhance security necessarily degrades the usability. On the other hand, relaxing assumptions and rigorous security design to improve the user experience can lead to security breaches that can harm the users' trust. In this paper, we demonstrate how careful visualization design can enhance not only the security but also the usability of authentication. To that end, we propose two visual authentication protocols: one is a one-time-password protocol, and the other is a password-based authentication protocol. Through rigorous analysis, we verify that our protocols are immune to many of the challenging authentication attacks applicable in the literature. Furthermore, using an extensive case study on a prototype of our protocols, we highlight the potential of our approach for real-world deployment: we were able to achieve a high level of usability while satisfying stringent security requirements.

Keywords — Authentication; Smartphone; Malicious code; Key logger

I. INTRODUCTION

Authentication is a fundamental aspect of system security. It confirms the identity of any user trying to log on to a domain or access network resources. Windows Server 2003 family authentication enables single sign-on to all network resources. With single sign-on, a user can log on to the domain once, using a single password or smart card, and authenticate to any computer in the domain.Threats against electronic and financial services can be classified into two major classes: credential stealing and channel breaking attacks [10]. 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 [11], [6]. On the other hand, channel breaking attacks—which allow for eavesdropping on communication between users and a financial institution—are another form of exploitation [12]. While classical channel breaking attacks can be prevented by the proper usage of a security channel such as IPSec [13] and SSL (secure sockets layer) [3], recent channel breaking attacks are more challenging. Indeed, "keylogging" 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 keyloggers [14], [6], [4]. A keylogger is 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 types in her password in a bank's signin box, the keylogger intercepts the password. The threat of such keyloggers is pervasive and can be present both in personal computers and public kiosks; 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 worse, keyloggers, often rootkitted, are hard to detect since they will not show up in the task manager process list. To mitigate the keylogger 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 keyloggers. Unfortunately, the keylogger, which has control over the entire PC, can easily capture every event and read the 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 [7]. However, this technique is not universal and can interfere with the operating system and native drivers. Considering that a keylogger sees users' keystrokes, this attack is quite similar to the shoulder-surfing attack. To prevent the shoulder-surfing attack, many graphical password schemes have been introduced in the literature [5], [8]. However, the common theme among many of these schemes is their unusability: they are quite complicated for a person to utilize them. For some users, the usability is as important as the security, so they refuse to change their online transaction experience for higher security. The shoulder-surfing attack, however, is different from keylogging in the sense that it allows an attacker to see not only direct input to the computer but also every behavior a user makes such as touching some parts of screen. To adopt shoulder-surfing resistant schemes for prevention of keylogger is rather excess considering the usability. Notice that while defending against the shouldersurfing attack is out of the scope of this work, and could be partly done using other techniques from the literature intended for this purpose, the promising future of smart glasses (like Google glasses) makes the attack irrelevant to our protocols if it is to be implemented using them instead of mobile phones. It is not enough to depend only on cryptographic techniques to prevent attacks which aim to deceive users' visual experience while residing in a PC. Even if all necessary information is securely delivered to a user's computer, the attacker residing on that user's computer can easily observe and alter the information and show valid-looking yet deceiving information. Human user's involvement in the security protocol is sometimes necessary to prevent this type of attacks but humans are not good at complicated calculations and do not have a sufficient memory to remember cryptographically strong keys and signatures. Thus, usability is an important factor in designing a human-involving protocol [9].

Our approach to solving the problem is to introduce an intermediate device that bridges a human user and a terminal. Then, instead of the user directly invoking the regular authentication protocol, she invokes a more sophisticated but user-friendly protocol via the intermediate helping device. 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 keylogging 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 smartphoneaided augmented reality. The visual involvement of users in a security protocol boosts both the security of the protocol and is re-assuring to the user because she feels that she plays a role in the process. 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 keylogging attack, while not degrading the usability. However, we note that our goal is not securing the authentication process against the shouldersurfing 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. PROBLEM STATEMENT

The design of secure authentication protocols is quite challenging, considering that various kinds of root kits reside in PCs (Personal Computers) to observe user's behavior and to make PCs untrusted devices. Involving human in authentication protocols, while promising, is not easy because of their limited capability of computation and memorization. Therefore, relying on users to enhance security necessarily degrades the usability. On the other hand, relaxing assumptions and rigorous security design to improve the user experience can lead to security breaches that can harm the users' trust. In this Project, we demonstrate how careful visualization design can enhance not only the security but also the usability of authentication. To that end, we propose two visual authentication protocols: one is a one-time-password protocol, and the other is a password-based authentication protocol. Through rigorous analysis, we verify that our protocols are immune to many of the challenging authentication attacks applicable in the literature. Furthermore, using an extensive case study on a prototype of our protocols, we highlight the potential of our approach for real-world deployment: we were able to achieve a high level of usability while satisfying stringent security requirements.

III. Related Work

A. System Model

Our system model consists of four different entities (or participants), which are a user, a smartphone, a user's terminal, and a server. The user is an ordinary human, limited by human's shortcomings, including limited capabilities of performing complex computations or remembering sophisticated cryptographic credentials, such as cryptographically strong keys. With a user's terminal such as a desktop computer or a laptop, the user can log in a server of a financial institution (bank) for financial transactions. Also, the user has a smartphone, the third system entity, which is equipped with a camera and stores a public key certificate of the server for digital signature verification. Finally, the server is the last system entity, which belongs to the financial institution and performs back-end operations by interacting with the user (terminal or smartphone) on behalf of the bank.

Assuming a smartphone entity in our system is not a farfetched assumption, since most cell phones nowadays qualify (in terms of processing and imaging capabilities) 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 in this paper, a smartphone does not use the communication channel— unless otherwise is explicitly 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.

B. 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 realistic assumption in most electronic banking systems. Second, we assume that the server is secured by every means and is immune to every attack by the attacker; hence the attacker's concern is not breaking into the server but attacking the user. Finally, with respect to the keylogger attack, we assume that the keylogger 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 attacker is capable of doing any of the following: _ The attacker has a full control over the terminal. Thus, – While residing in a user's terminal, the attacker can capture user's credentials such as a password, a private key, and OTP (one time password) token string. – The attacker can deceive a user by showing a genuine-looking page that actually transfers money to the attacker's account with the captured credentials that she obtained from the compromised terminal. – Or, just after a user successfully gets authenticated with a valid credential, the attacker can hijack the authenticated session. _ The attacker is capable of creating a fake server to launch phishing or pharming attacks. For the smartphone in Protocol 1, we assume that it is always trusted and immune to compromise, which means no malware can be installed on it. Notice that this assumption is in line with other assumptions made on the smartphone's trustworthiness when used in similar protocols to those presented in this paper [3], [2], [4]. 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 assumption can be relaxed 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 compromise assumption 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 services, the use of digital certificates issued by the bank is very common. Furthermore, the use of such cryptographic credentials and maintaining 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.

C. Linear and Matrix Barcodes

A barcode is an optical machine-readable representation of data, and it is widely used in our daily life since it is attached to all types of products for identification. In a nutshell, barcodes are mainly two types: linear barcodes and matrix (or two dimensional, also known as 2D) barcodes. While linear barcodes—shown in Figure 1(a)—have a limited capacity, which depends on the coding technique used that can range from 10 to 22 characters, 2D barcodes—shown in Figure 1(b) and Figure 1(c)—have higher capacity, which can be more than 7000 characters. For example, the QR code— a widely used 2D barcode—can hold 7,089 numeric, 4,296 alphanumeric, or 2,953 binary characters [4], making it a very good high-capacity candidate for storing plain and encrypted contents alike. Both linear and matrix barcodes are popular and have been widely used in many industries including, but not limited to, automotive industries, manufacturing of electronic components, and bottling industries, among many others. Thanks to their greater capacity, matrix barcodes are even proactively used for advertisement so that a user who has a smartphone can easily scan them to get some detailed information about advertised products. This model of advertisement—and other venues of using these barcodes in areas that are in touch with users—created the need for barcode's scanners developed specifically for smartphones. Accordingly, this led to the creation of many popular commercial and free barcode scanners that are available for smartphones such as iPhone and Andriod phones alike.

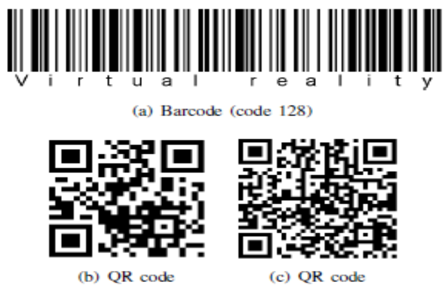

Fig. 1. Three different barcodes encoding the statement "Virtual reality". (a) is a linear barcode (code 128), and (b) and (c) are matrix barcodes (of the QR code standard). While (b) encodes the plain text, (c) encodes an encrypted version using the AES-256 encryption algorithm in the cipher-block chaining (CBC) mode (note this last code requires a password for decryption).

KEYLOGGING-RESISTANTVISUAL AUTHENTICATION PROTOCOLS

In this section, we describe two protocols for user authentication with visualization. Before getting into the details of these protocols, we review the notations for algorithms used in our protocols as building blocks. Our system utilizes the following algorithms:

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.
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.
Sign (): a signature generation algorithm which takes a private key SK and a message M from the set M, and outputs a signature _.
Verf(_): a signature verification algorithm which takes a public key PK and a signed message (M;), and returns valid or invalid.
QREnc(_): a QR encoding algorithm which takes a string S in S and outputs a QR code.
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 (Indistinguishability 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 [1]. 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 [14]. In particular, and for efficiency reasons, we recommend the short signature in.

A. Authentication With Random Strings
In this section, we introduce an authentication protocol with a onetime 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 user connects to the server and sends her ID.
2) The server checks the ID to retrieve the user's public key (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 by OTP = Decrk(EOTP) and type in the OTP in the terminal with a physical keyboard.
5) The server checks the result and if it matches what the server has sent earlier, the user is authenticated. 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 user, and a randomized keyboard. A high-level event-driven code describing the protocol is shown in Figure 1. The protocol works as follows:
1) The user 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 (= ecrSKID(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 checks whether the password is correct or not by confirming if the correct buttons have been clicked.

Conclusion
Moreover, we have shown two realizations of protocols that not only improve the user experience but also resist challenging attacks, such as the key logger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box smartphone devices In this paper, we proposed and analyzed the use of userdriven visualization to improve security and user-friendliness of authentication protocols. 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 protocol on the smart glasses such as the google glass, and conduct the user study.

REFERENCES
[1] —. Google authenticator. http://code.google.com/p/google-authenticator/.

[2] —. Rsa securid. http://www.emc.com/security/rsa-securid.htm.

[3] Cronto. http://www.cronto.com/.

[4] —. BS ISO/IEC 18004:2006. information technology. Automatic identification and data capture techniques. ISO/IEC, 2006.

[5] —. ZXing. http://code.google.com/p/zxing/, 2011.

[6] D. Boneh and X. Boyen. Short signatures without random racles. In Proc. of EUROCRYPT, pages 56–73, 2004.

[7] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajan . 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.

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

[9] 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.

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

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

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

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

[14] 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.