Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Secured System for Keylogging - Resilient Visual Validation

[1]J. Babi Chaitanya, [2]Maddali M. V. M. Kumar
[1]PG Student, [2]Assistant Professor
[1, 2] MCA Department, St. Ann's College of Engineering & Technology,
Andhra Pradesh, India

***Abstract - Keylogging or keyboard capturing is the exercise of recording (or logging) the keys struck on a keyboard, ordinarily in a secretive way so that the individual exploiting the keyboard is unconscious that their activities are being observed. It likewise has exceptionally authentic uses in investigations of human-computer interaction. There are various keylogging techniques, extending from hardware and software based methodologies to acoustic examination. Including human in validation protocols, while guaranteeing, is not simple in light of their restricted capacity of calculation and remembrance. We exhibit how careful visualization outline can improve the security as well as the convenience of validation. We propose two visual validation protocols: one is a one-time-password protocol, and another is a password-based validation protocol. Our approach for genuine arrangement: we had the capacity attain to abnormal state of ease of use while fulfilling stringent security necessities.***

***Keywords— Virtual Validation, Smartphone, Malicious code, Keylogging, Protocols, Barcode, QR Code***

## I. INTRODUCTION

Keylogging exhibits an extraordinary test to security supervisors. Dissimilar to customary worms and viruses, certain sorts of key loggers are everything except difficult to discover. Key loggers are a kind of malware that malignantly track customer information from the comfort attempting to recuperate individual and private information. Growing machine use for essential business and individual activities using the Internet has made feasible treatment of keylogging basic. Cybercriminals have fictional various schedules to get sensitive information from your endpoint devices. On the other hand, few of them are as effective as keystroke logging. Keystroke logging, generally called keylogging, is the hold of imprint characters. The data caught can incorporate report content, passwords, user ID's, and other potentially touchy bits of information. Using this approach, an assailant can get essential data without breaking into a cemented database or file server.

A keylogger is modifying, proposed to capture the larger part of a customer's support strokes, and a while later make use of them to copy a customer in money related trades. Case in point, at whatever focuses a customer sorts in her watchword in a bank's sign in box, the keylogger gets the mystery word. The risk of such keyloggers is prevalent and can be display both in PCs and open corners; there are constantly circumstances where it is imperative to perform monetary trades using an open machine regardless of the way that the best concern is that a customer's watchword is prone to be stolen in these machines. Far and away more terrible, keyloggers, frequently root kitted, are tricky to identify since they won't appear in the assignment director methodology list. To relieve the keylogger attack, virtual or onscreen consoles with random console game plans are generally utilized as a part of practice. Both procedures, by revising letter sets randomly on the catches, can baffle basic keyloggers. Lamentably, the keylogger, which has control over the entire PC, can without much of a stretch capture each occasion and read the video buffer to make a mapping betwixt the clicks and the new letter set. An alternate moderation procedure is to utilize the console snaring counteractive action strategy by bothering the console interferes with vector table [1].

On the other hand, this strategy is not general and can interfere with the working framework and local drivers. It is insufficient on depends just on cryptographic strategies to counteract attacks which mean to swindle clients' visual experience while living in a PC. Regardless of the fact that all vital data is safely conveyed to a client's machine, the attacker living on that client's machine can without much of a stretch watch and change the data and show legitimate looking yet misleading data. Human client's contribution in the security convention is off and on again important to keep this sort of attacks however people are bad at muddled estimations and don't have a sufficient memory to recall cryptographically solid keys and marks. Accordingly, ease of use is a critical variable in outlining a human-including convention [2].

Our methodology to taking care of the issue is to present a transitional gadget that extensions a human client and a terminal. At that point, rather than the client straightforwardly conjuring the general confirmation convention, she conjures a more complex yet easy to use convention by means of the moderate helping gadget. Each association between the client and a middle of the road helping gadget is envisioned utilizing a Quick Response (QR) code. The objective is to keep client encounter the same as in legacy confirmation routines however much as could be expected, while forestalling keylogging attacks. Consequently, in our conventions, a client does not have to retain additional data aside from a customary security token, for example, secret key or PIN, and dissimilar to the former writing that safeguards

against ought to surfing attacks by obliging complex reckonings and far reaching inputs. All the more particularly, our methodology envisions the security procedure of validation utilizing an advanced mobile phone helped increased reality.

In this paper, we show how visualization can improve security as well as convenience by proposing two visual verification conventions: one for password-based validation, and the other for one-time-password. Through thorough investigation, we demonstrate that our conventions are safe to a number of the testing attacks relevant to different conventions in the writing. Besides, utilizing a broad research endeavour on a model of our conventions, we highlight the capability of our conventions in genuine arrangement tending to client's inadequacies and constraints. The main contributions of this paper are as follows:

1. Two protocols for validation that uses visualization by method for increased reality to give both high security and high convenience. We demonstrate that these conventions are secure under a few certifiable attacks including keyloggers. Both conventions offer favourable circumstances because of visualization both as far as security and ease of use.
2. Model usage as Android applications which show the ease of use of our conventions in true organization settings. The rest of this paper is organized as follows. In section II, related work; section III contains proposed methodology; Enhancement in section IV; experimental results in section V; finally conclusion in section VI.

## II. RELATED WORK

A barcode is an optical machine-readable representation of data, and it is extensively worn in our daily life since it is adhered to all types of products for detection. In a nutshell, barcodes are primarily two types: linear barcodes and matrix (or two dimensional, also familiar as 2D) barcodes. While linear barcodes – exposed in Figure 1(a) - have a limited capacity, which depends on the coding method worn that can range from 10 to 22 characters, 2D barcodes - exposed in Figure 1(b) and Figure 1(c) - have higher capacity, which can be more than 7000 characters. For example, the QR code - a broadly used 2D barcode - can hold 7,089 numeric, 4,296 alphanumeric, or 2,953 binary characters [2], building it a very good high-capacity candidate for storing plain and encrypted contents alike.


(a) Barcode (code 128)

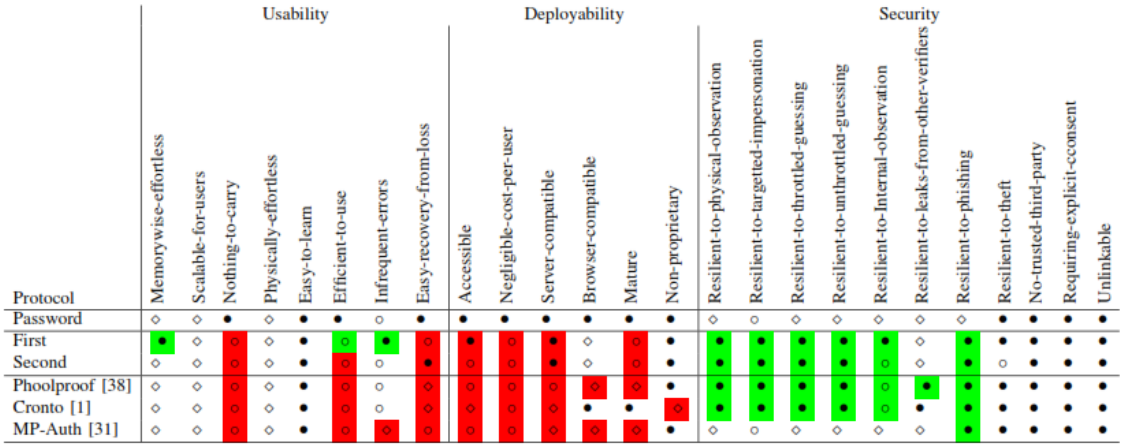
(b) QR code (c) QR code

Figure 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).

There has been an extensive assemblage of deal with the issue of client validation and in the connection of e- banking money. Of uncommon investment are validation protocols those utilization graphical passwords. To the best of our insight, our protocols are the first of their sort to utilize visualization for enhancing security and ease of use of validation protocols according to the path reported in this paper. A closely related vein of research is trust establishment for group communication using cognitive capabilities. Examples of such works include SPATE [3], GAnGS [4], and Safe Slinger [5]. None of these works use visualization as reported in this work, although they provide primitives for validation users and establishing trust. Another closely related work is - Seeing-is-Believing‖ (SiB) [6] (extended in [7]), which uses visual channels of 2D barcodes to resist the man-in-the-middle attack in device pairing. Though we utilize similar tools by using the 2D barcodes for information representation, and the visual channel for communicating this information, our protocols are further more generic than those proposed in [6]. Our protocols are tailored to the problem settings in hand, e-banking, with a different trust and attack model than that used in [6] - which results into different guarantees as explained earlier in this paper. To prevent against phishing, Parno et al. suggested the use of trusted devices to perform mutual validation and eliminate reliance on perfect user behaviour [8]. Slightly touched upon in this paper are keyloggers as potential attacks for credentials stealing, which are reported in [9], and other malwares which are reported in [10]. We limit our attention to the baseline, the password-based validation, and a few phone based validation protocols as shown in Table I. We notice that our first protocol meets the first metric by not requiring a password, meets the fifth, sixth, seventh, and eighth as shown in our user studies. This particularly explains much of the matching in deployability and security colourings for our protocol. The mismatch in "resilience to leaks from other verifiers" does not apply to our work, since it assumes a single verifier. The first protocol is more memory wise efficient, efficient to use, and infrequent errors than password based on the fact that it relies on OTP (not memorized by the user). In this paper we have shown that our protocols are secure even when one of the participants in the validation process (the terminal or smartphone) is compromised.

Table i a comparison with other works based on their usability, deployability, and security, for other systems and how they compare to our work. Comparisons are in relation with password-based validation, where stands for the case where the metric doesn't apply, stands for meeting the metric, means that the metric can be made to work in the design, green and red are indicators for better and worse than the case of the password-based validation, respectively.




## III. KEYLOGGING-RESISTANT VISUAL VALIDATION ALGORITHMS

In this section, we describe two protocols for user validation with visualization. Before getting into the details of these protocols, we appraise the notations for algorithms worn in our protocols as making 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 cipher-text C in the set C.
- Decrk (·): a decryption algorithm which takes a Cipher-text C in C and a key k, and outputs a plain-text (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 σ.
- Verf (·): a signature verification algorithm which takes a public key PK and a signed message (M, σ), 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 Cipher-text Attacker) security would be good for our application. A public key encryption scheme with IND-CCA2 adds random padding to a plain-text, which makes the cipher-text different whenever encrypted. 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 - enforceability against adaptive chosen-message attacker) can be used to serve the purpose of our system.

### *A. Validation with Random Strings:*

In this section, we bring in validation protocol with a one-time-password (OTP). The following protocol relies on a physically powerful assumption; it makes use of arbitrary string for validation. 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 OT P and encrypts it with the public key to obtain EOT P = EncrPKID (OT P).
3. In the terminal, a QR code QREOT P is displayed prompting the user to type in the string.
4. The user decodes the QR code with EOT P = QRDec (QREOT P). Because the random string is encrypted with user's public key (PKID), the user can read the OTP string only all the way through her Smartphone by OT P = Decrk (EOT P) and type in the OT P in the terminal with a physical keyboard.
5. The server checks up the outcome and if it matches what the server has sent earlier, the user is validated. 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. Validation Protocol with Password and Randomized Onscreen Keyboard:*

Our second protocol, which is cited to as Protocol 2 in the rest of this paper, uses a password shared betwixt the server and the user, and a randomized keyboard. The protocol works as follows:

1. The user connects to the server and sends her ID.
2. The server checks the acknowledged ID to repossess the user's public key (PKID) from the database. The server prepares π, a random permutation of a keyboard arrangement, and encrypts it with the public key to acquire EKBD = EncrPKID (π). Then, it encodes the cipher-text with QR encoder to obtain QREKBD = QREnc (EkID (π)). 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 (π = DecrSKID (EKBD)) on the smartphone's screen.
4. When the user sees the empty keyboard with the QR code throughout an application on the smartphone that has a private key, alphanumeric 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.

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 several issues related to our protocols, such as session hijacking, transaction verification, and securing transactions.

## IV. ENHANCEMENT

1. In this paper we developed enhancement is offline transaction. Mostly transactions are done through online only. But for time consuming and quick transaction we proposed offline transaction. In offline transaction user generate one file, inside that file user account no, transaction amount and etc., are available. Those details are prepared by user when they are in offline. When user entered into online, they just load this file into the applications for fund transaction. Using this offline transaction, user timings are more consumed.
2. Another enhancement is IMI security. Main purpose of this is, to avoid malicious transaction. When other user knows my username and password means, they can use my details for fund transfer without my knowledge. To avoid this we are providing IMI security. Every user registration server stores their IMI number into their database. Another malicious user, use my username and password in their mobiles means IMI no vary so proper transaction will not occur.

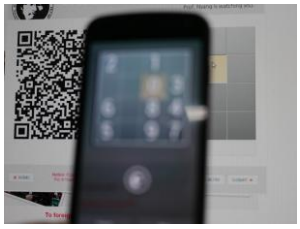
In Protocol 1, OTP tokens that have high entropy and are human-unfriendly making them hard to remember and recall are one-time used. Accordingly, a shoulder surfer would not benefit from launching an attack by trying to observe what the user at the terminal is inputting. The attack is not applicable to this protocol.

In Protocol 2, observing the terminal or the smartphone keyboard layout (on the smartphone screen) alone would not reveal the credentials of the user. Observing both at the same time in a shoulder surfing attack, and mapping stroked keys on the terminal to those on the smartphone screen would reveal the credentials of the user. Being able to successfully launch this attack is a non-trivial task, and requires the attacker to be in very near proximity to the user, which would raise the user's suspicions about the intentions of the attacker. However, because the attacker who successfully conducts all this necessary steps will get a password in Protocol 2, the protocol cannot be said to be secure against the shoulder- surfing attack. We leave it for the future work to make Protocol 2 secure against the shoulder-surfing attack by combining it with shoulder-surfing resistant schemes already explored in the literature.

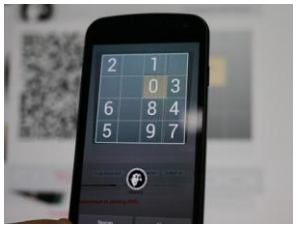
*Experimental Outcomes:*

(a) QR Code Scanning (before)

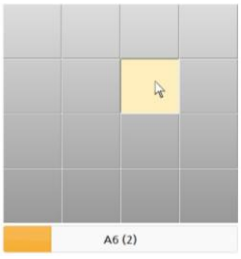
(b) QR Code Scanning (after)



(c) Keyboard on Terminal




(d) Keyboard on Smart Phone




(e) Clicking Password on Blank Keyboard

Figure 2. Pictures of the prototype we have developed to exhibit our validation protocols. (a) & (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.

## V. CONCLUSION

In this paper, we proposed and analyzed the use of user - driven visualization to improve security and user - friendliness of validation protocols. Moreover, we have shown two realizations of protocols that not only improve the user experience but also resist challenging attacks, such as the keylogger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box smartphone 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 validation. 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. Second, we plan to investigate the design of other protocols with more stringent performance requirements using the same tools provided in this work. In addition, we will study methods for improving the security and user experience by means of visualization in other contexts, but not limited to validation such as visual decryption and visual signature verification. Finally, reporting on user studies that will benefit from a wide deployment and acceptance of our protocols would be a parallel future work to consider as well

## REFERENCES

[1] R. Pemmaraju. Methods and apparatus for securing keystrokes from being intercepted between the keyboard and a browser. Patent 182,714.

[2] N. Hopper and M. Blum. Secure human identification protocols. In Proc. of ASIACRYPT, 2001

[3] Y.-H. Lin, A. Studer, Y.-H. Chen, H.-C. Hsiao, E. L. - H. Kuo, J. M. McCune, K.-H. Wang, M. N. Krohn, A. Perrig,

[4] B.-Y. Yang, H.-M. Sun, P.-L. Lin, and J. Lee. Spate: Small-group pki-less authenticated trust establishment. IEEE Trans. Mob. Comput., 9(12):1666–1681, 2010.

[5] 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.

[6] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing - is-believing: Using camera phones for human-verifiable authentication. In Proc. of IEEE Symposium on Security and Privacy, pages 110–124, 2005.

[7] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing - is-believing: using camera phones for human-verifiable authentication. International Journal of Security and Networks, 4(1/2):43–56, 2009.

[8] B. Parno, C. Kuo, and A. Perrig. Phool proof phishing prevention. In Proc. of Financial Cryptography, pages 1–19, 2006.

[9] A. Slowinska and H. Bos. Pointless tainting? : evaluating the practicality of pointer tainting. In Proc. of ACM EuroSys, pages 61–74, 2009.

[10] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: analysis of a botnet takeover. In Proc. of ACM CCS, pages 635–647,2009