Mitigating Snoop-Forge-Replay Attack by Integrating Text-Based and Language-Based Traits with the Keystroke Verification System

S.Sridhar

Abstract— A new attack called the snoop-forge-replay attack is presented on keystroke-based continuous verification systems. The snoop-forge-replay is a sample-level forgery attack and is not specific to any particular keystroke-based continuous verification method or sysem. It can be launched with easily available keyloggers and APIs for keystroke synthesis. Our results ffrom 2460 experiments show that: 1)the snoop-forge-replay attacks achieve alarmingly high error rates compared to zero-effort imposter attacks, which have been the de facto standard for evaluating keystroke-based continuous verification systems; 2)four state-of-the –art verification methods, three types of keystroke latencies, and 11 matching-pair settings(a key parameter in continuous verification with keystrokes) that is examined here were suspectible to the attack; 3)the attack is effective even when as low as 20 to 100 keystrokes were snooped to create forgeries.

Index Terms— Biometrics, Continuous verification, Keystroke dynamics, Snooping, Spoof attacks.

1 INTRODUCTION

KEYSTROKE DYNAMICS, or typing dynamics, is the detailed timing information that elaborately describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard. One of the behavioral biometric of Keystroke Dynamics uses the manner and rhythm in which an individual types characters on a keyboard or keypad. The keystroke rhythms.of a user are measured to develop a unique biometric template of the users typing pattern for future authentication.Raw measurements available from almost every keyboard can be stored even recoreded to determine Dwell time (timing of key pressed) and Flight time (the time between "key up" and the right next "key down"). The saved keystroke timing data is then processed through a unique and specific neural algorithm, which determines a primary pattern for future comparison.

Similarly, vibration information may be used to create a pattern for future use in both identification and authentication tasks.Data needed to analyses keystroke dynamics is obtained by keystroke logging. Normally, all that is reversed when logging a typing duration is the sequence of characters corresponding to the order in which keys were pressed and timing information is discarded. When reading email, the receiver cannot tell from reading the phrase "I saw 3 zebras!" whether: 1) that was typed rapidly or slowly, 2) the sender used the left shift key, the right shift key, or the caps-lock key to make the "i" turn into a capitalized letter "I",3) the letters were all typed at the same pace, or if there was a long pause before the letter "z" or the numeral "3" while you were looking for that letter, 4) the sender typed any letters wrong initially and then went back and corrected them, or if they got them right the first time.

S.Sridhar, is currently pursuing masters degree program in computer science engineering in Anna University, India, Mobile-904739973. E-mail: ssridhar.ooty@gmail.com

2 BIOMETRIC FUNCTIONING.

2.1 Basic Stage

Keystroke Dynamics technology extracts and measures the distinctive characteristics found by typed sequences of characters, and creates a statistically unique signature from the typing patterns of a person. These distinctive features include the duration for which keys are held during the session and the elapsed time between successive keystrokes.

2.2 Final Stage

Scientific research has proven that these Keystroke Dynamics is always completely reliable and accurate. A National Bureau of Standards (NBS) study concluded that keystroke biometric authentication achieved at least 98% accuracy. From then, the technology of keystroke biometric has further improved to the future level that is comparative and competitive to other biometric solutions such as fingerprint and voiceprint and other biometric methods.

2.3 Nothing Extra or Type at Logon

With all other authentication methods, you are always asked to provide some more unique codes. With Type Sense, you will be asked to type what you always enter at logon: your username and password. Type Sense is completely transparent to the users.

2.4 Flexible Enrolment

Type Sense can be enrolled at that cureent moment immediately in the registration process by intensive training, or gradually over a period of time by adaptive learning.

Each of the user will be exactly loaded with their latency and their username when every new users are entering for their

first logon.

3BACKGROUND

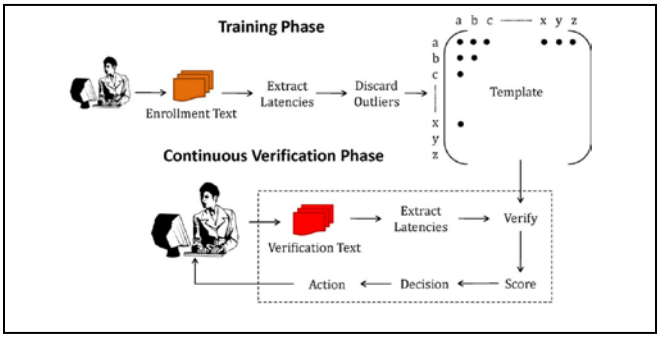
Here I illustrate continuous user verification with key-strokes.Details follow.

Keystroke Latencies:

Widely used latencies in the literature are: 1) *key hold* latency—is the time between press and release of the same key, 2) *key press* latency—is the time between press of a key and press of the next key, and 3) key interval latency—is
the time between the release of a key and press of the next key. Experimenting with key hold, key interval, and key press latencies.

Template:

A template stores the keystroke signatures of a user. I used 26-by-26 matrix as the template. Each cell corresponds to an English alphabet pair.
For example, with key press latencies, if cell " " has, it means that the user(during enrollment) typed thrice with 110 ms, 90 ms, and 100 ms delay between the press of and the press of and the meanis calculated for all the delay is 100 ms with 10 ms standard deviation. Unlikekey press and interval latencies, a key hold latency by definition is associated with a letter (and not letter pair). Because this template holds only letter pairs, when used key hold latencies, each cell stored the key hold latencies of the first letter of it letter pair (e.g., cell " " stored key hold latencies of only when the next letter typed is). The template is homogeneous, meaning it stores only one type of latencies (i.e., either key hold, interval, or press).



Outlier Detection:

Latency values that markedly deviates from majority of the latency values of a user which can be distorted the typing profile of a user, especially if the profile contains statistics sensitive to outliers (e.g., mean). Several studies (e.g., [2], [10], and [11]) performed outlier detection and reported performance gains. Here with this experiments use of a distance based outlier detection method that worked well in an earlier work [11].

Overview of Continuous Verification of Keystroke:

Matching Pairs: Because there are no constraints on what a user types during continuous verification, some keystrokes typed during the verification phase may not have reference signatures in the template. This can happen because the enrollment text used for building the template may not have all the letter pairs present in the 26-by-26 matrix. This problem can be resolved by performing verification using letter pairs that are common to the template and the verification text. Following to thesecommon letter pairs as *matching pairs*, used to denote the number of matching pairs.

4SNOOP-FORGE REPLAY ATTACK

The attack presented in this paper falls under the generative attacks on behavioral biometric systems.
Here, the attacker secretly steals a victim's keystroke timing information. For example, if the victim typed the text"this is snooped text", the attacker records a seriesof timestamps—(time when was pressed), (time whenwas released), , andso on.An attacker can snoop a victim's keystroke timing informationusing a hardware keylogger and even using some of the software keylogger. Software keyloggers have become the most popular forms of keyloggers because they can be easily developed, are easily available,2 and can be deployed from remote locations onto a victim's machine (e.g., using trojans and spyware). We used keystroke data collected from 150 participants during the period 13–21 October 2009 as snooped keystrokes (see Table I and Section VI-A for details). This data was collected using a software keylogger developed in C#. The snooped keystrokes were used to attack templates that were built from keystrokes collected approximately six months afterthe snooped keystrokes.

5CREATING A KEYSTROKE FORGERY

Creation of a keystroke forgery of a victim user. A forgery has two parts: 1) "dummy" text and 2) a series of latencies between the press and release of letters in the dummytext from the selected papers in Wikipedia . For some clarification, a forgery of any large document can have the dummy text"this is dummy text". The key hold and interval values for this text come from the snooped keystroke latencies of U_i.The goal of the emulator is to use the snooped latencies to inject key press and release events for the dummy text in a way that the verifier thinks that it is the victim U_i.who is typing the dummy text.

The emulator algorithm, gives the steps to forge and replay a victim user 's typing pattern asfollows:

Algorithm 1: Replay the forgery of user U_i
Input: Dummy text file containing 497,184 words from coca and text 20 Wikipedia pages. Key hold(e.g.,$kh_{n:v}$ $kh_{v:SPACE}$,etc.) and key interval(e.g.,. $ki_{n:v}$ $ki_{v:SPACE}$,etc.) latencies computed from U_i's snooped keystrokes. Here, "$kh_{x:v}$" denotes the *snooped* key hold latencyof x when the next character typed is y and "$kh_{x:v}$" denotes the *snooped*key interval latency between characters *x* and *y*.
Output: A replay of user U_i's keystroke forgery.
1 **Initialization:**
2 *n* ← Number of characters in the dummy text file.
3 *dummyTextArr*[0:n -1] ← Copy each character in the dummy text file into the array;
/* *Eachcell in the dummyTextArr holds a character in the dummy*

text file/*
4 *dummyIndex←0;/*Index to the first character in the dummyTextArr*/*
5 *trap_counter ← 0;/*Counter to ensure that character pairs in the dummy text that not do*
have corresponding snooped latencies do not available even after traversing 500
charactersin the dummy text, then character pairs is reset to a random character in
dummyTextArr (Line 25)/*
6 *first ← Ø;/* A variable to store* first character.**/*
7 *second ← Ø;/* A variable to store second character.*/*
8 *startTime ← System time at the start of the program;*
9 *currentTime ← Current system time:*
10 **while**(currentTime – startTime ≤ P hours)/*We set P to 24.*/**do**
11 first← dummyTextArr[dummyIndex];
12 second← first;
13 **while**dummyIndex<n and trap_counter≤500
do
14**if** ($kh_{first:second}$and $ki_{first:second}$) is snooped /*checks if letter pair from the dummy
text has corresponding snooped latencies.*/ **then**

15 $KH_{first:second}$← $kh_{first:second}$;
$KI_{first:second}$← $ki_{first:second}$;/*Forge latencies. "KH" and "KI" denote latencies in
a forge.*/
16 replay
(first,$KH_{first:second}$,$KI_{first:second}$);
/*Replay dummy text by generating key press and release events of first when second
is the next character*/
17 first ←second; trap_counter←0;
18 **end**
19 **else**
20 trap_counter←trap_counter+1;
21 **end**
22 dummyIndex← dummyIndex+1;
23 second ← dummyTextArr[dummyIndex];
24 **end**
25 dummyIndex←Reset to a random cell of dummyTextArr;
26 currentTime← Current system time;
trap_counter←0;
end

6 Replaying A Forgery of Victim

Keystroke Emulator:

The developed a keystroke emulatorthat *injects* synthetic key press and release events. We programmedthe emulator in Visual C++ and used SendInputAPI. The goal of the emulator is to use the snooped latencies to inject key press and release events for the dummy text in a way that the verifier thinks that it is the victimU_i who is typing the dummy text. The emulator algorithm, referred as "Algorithm 1", gives the steps to forge and replay a victim user 'sU_i typing pattern. At this point, emphasize that Algorithm 1 is *one of the many* possible ways to generate snoop-forge-replay attacks.While maintaining the general idea of snooping and replaying keystrokes, the attacker can evolve Algorithm 1 in severalways. For example, the attacker can make them stop working heuristics to imputemissing latency values or snoop only selected latencies from a victim, to generate desired text or system commands.

6.1Keystroke Data Collection

The participant to type *two* types of free text:1) *copy text*—each participant typed several paragraphs of English text from a document provided by us; and 2) *selftext*—participant had to compose and type text. The participants were allowed to make spelling mistakes, typographical errors and if they chose, could correct them using Backspace or Delete keys. The keystroke data collection software provided GUI (e.g.,ext boxes, buttons, and character counters) for typing copy and self texts. Each participant was required to type at least 1800 characters of copy text. For typing copy text, we provided paper copies of five well known sample texts to the participants. A participant received one of the five sample texts randomly.

Copy versus self text—

Typing self text is a exact representation of a user's typing process. However, conducting pilot trials in laboratory before undertaking full-scale data collection and observed that typing 1200–1800 characters of self text took considerably more time than typing copy text of the same length and in most cases fatigued participants. To achieve a trade-off between participation time and obtaining realistic typing samples, choose to collect a mixture of copy and self texts.

6.1Zero Effect Imposter Attack

Extracting Verification Attempts:

From a user's typing sample, extracted verification attempts as follows: 1) read the text in the order it was typed and extract latencies until matching pairs are obtained; 2) present the matching pairs to the verifier to obtain a verification score (this constitutes one verification attempt); 3) read the text from the point where it was stopped in Step 2 until matching pairs are obtained; and 4) repeat steps 2 and 3 until the text ends. This procedure partitions the text into contiguous, nonoverlapping, variable-length windows, each containing exactly matching pairs. Each window corresponds to one verification attempt.
While experimenting with values: 20, 40, 60, 80, 100, 120, 150,300, 350, 500, and 750.*Relative (R) and Absolute (A) Verifiers [1]:* Given a verification attempt, "R" verifier outputs a score as follows. Two arrays and are constructed. contains the matching pairs ranked in ascending order of their corresponding mean latencies (in the template). contains the matching pairs ranked in ascending order of their latencies in the verification attempt. The "R" measure between and is computed as the normalized array disorder between. The "R" measure lies between 0 and 1, 0 (or 1) indicates a perfect match (or mismatch) between the verification attempt and the template. The "R" measure is given as where the maximum disorder of an

array of elements is given by: , if is even; else if is odd. The "A" measure verifier outputs a score as follows: for each matching pair, two latency values are considered: 1) the average latency value stored in the template and 2) the average latency value in verification attempt. The larger of the two is divided by the smaller. A matching pair becomes *valid* if the ratio falls between 1 and a threshold (after some trial and error experiments, we choose 1.45 as threshold).

The "A" measure of 0 (or 1) provides a perfect match (or mismatch) between the verification system and the template.
Each of the keystrokes are calculated as per the system designed.

7PREVALENCE ANALYIS

7.1 Misspelling analysis:

This calculates the rate of Misspelt word when the user types the text wrongly. This performs the step when the number of times each of the words whose misspellings are being identified was found in the entire text body is counted and recorded. The attacking can be found then this recorded rate match with the original latency of the text. The attack can be found and mitigated when the calculated score does not match with the original record of data .Secondly The Type of word for which user has latency outliers can be verified .This can be done by estimating the keystroke press and release for Normal and outlier user. Thirdly The forgers digraph to be detected and verified. This can be done by Extracting Digraph Latencies from the snooped keystroke timing. The mean and standard deviation of the digraph is evaluated. If These mean and Standard deviation of this digraph is not found in the template Then the required user is said to be forged one.

Prevalence analysis is given by Number of words is indirectly proportional to the exact number of words in the typed column.

8EQUAL ERROR RATE OR CROSSOVER ERROR RATE

The rates at which both accept and reject errors are equal. The value of the EER can be easily obtained from the ROC curve. The EER is a quick way to compare the accuracy of devices with different ROC curves. In general, the device with the lowest EER is most accurate.

Integratingtext-based and language-based traits into the verificationprocess, such as—1) the rate at which a user misspells words orrepeats letters, 2) type of words for which user has latency outliers,3) how the user revises text i.e., revision pattern, and so on,the impact of the attack can be mitigated. In future work, will pursue the problem of designing keystroke based verificationsystems that are resilient to snoop-forge-replay attacks.

9CONCLUSION

A new sample-level attack called "snoop-forge-replay" attack that synthesizes keystroke forgeries using timing information stolen from victim users has been presented. The results from 2640 experiments (involving 150 users,four state-of-the-art continuous verifiers, three types ofkeystroke latencies, and 24 attack configurations) reveal that snoop-forge-replay attacks are very effective in increasing EERs. With 20 to 1200 snooped keystrokes, the average snoop-forge-replay attack EERs were between 0.487 and0.912. In comparison, the baseline EERs with zero-effort impostor attacks were between 0.03 and 0.285 (i.e., the attack increased EERs from between 69.33 to 2730.55 %). Theresults additionally show that effective keystroke forgeries canbe created with a) as low as 20 to 100 characters of snooped text and b) old legacy keystroke timing information. The main reason for the success of snoop-forge-replay attackis that keystroke based continuous verification methods solely rely on user's latency information, which can be easily forged has been demonstrated.

ACKNOWLEDGMENT

I, author of this work, wish to thank K.Manikandan, who guided and provided immense support throughout. This work was supported in part by a grant from Mrs.Akhila, who taught me biometrics deep and clean.

REFERENCES

[1] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[2] T. Shimshon, R. Moskovitch, L. Rokach, and Y. Elovici, "Continuous verification using keystroke dynamics," in *Proc. Int. Conf. ComputationalIntel. and Security*, Los Alamitos, CA, USA, 2010, pp. 411–415.

[3] U. Uludag and A. K. Jain, "Attacks on biometric systems: A case studyin fingerprints," in *Proc. SPIE Security, Steganography and WatermarkingofMultimedia Contents VI*, Jan. 2004, vol. 5306, pp. 622–633.

[4] R. Maxion and K. Killourhy, "Keystroke biometrics with number-pad input," in *Proc. 2010 IEEE/IFIP Int. Conf. Dependable Systems andNetworks (DSN)*, pp. 201–210, 28 2010-July 1 2010.

[5] L. Ballard, D. Lopresti, and F. Monrose, "Forgery quality and its implications for behavioral biometric security," *IEEE Trans. Syst., Man,Cybern. B*, vol. 37, no. 5, pp. 1107–1118, Oct. 2007.

[6] L. Ballard, D. Lopresti, and F. Monrose, "Evaluating the security of handwriting biometrics," in *Proc. 10th Int. Workshop on the Foundationsof Handwriting Recognition*, 2006, vol. 15, pp. 461–466.

[7] L. Ballard, F. Monrose, and D. Lopresti, "Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing," in *Proc. 15th Conf. USENIX Security Symp.*, California, USA, 2006, vol.15.

[8] D. Stefan, X. Shu, and D. Yao, "Robustness of keystroke-dynamics based biometrics against synthetic forgeries,"

Comput. Security, vol. 31, no. 1, pp. 109–121, Feb. 2012.

[9] L. C. F. Araujo, L. H. R. Sucupira, M. Lizarraga, L. Ling, and J. B. T. Yabu-uti, "User authentication through typing biometrics features,"*IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 851–855, Feb. 2005.

[10] S. Joshi, "Naive Bayes and Similarity Based Methods for Identifying Computer Users Using Keystroke Patterns," Ph.D. dissertation, Louisiana Tech University, Ruston, LA, USA, 2009.