A Novel Keystroke Dataset for Preventing Advanced Persistent Threats

Xiaofei Wang*[a], Rashik Shadman*[b], Daqing Hou[c], Faraz Hussain[d] and Stephanie Schuckers[e]

Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY, U.S.A.

Keywords: Keystroke Dataset, Keylogger, Authentication, APT.

Abstract: Computer system security is indispensable in today's world due to the large amount of sensitive data stored in such systems. Moreover, user authentication is integral to ensuring computer system security. In this paper, we investigate the potential of a novel keystroke dynamics-based authentication approach for preventing Advanced Persistent Threats (APT) and detecting APT actors. APT is an extended and planned cyber-attack in which the intruder logs into a system many times over a long period of time to gain administrative access and to steal sensitive data or disrupt the system. Since keystroke dynamics can be made to work whenever an APT actor is typing on the keyboard, we hypothesize that it naturally be a good match for APT detection. Furthermore, keystroke dynamics promises to be non-intrusive and cost-effective as no additional hardware is required other than the keyboard. In this work, we created a novel dataset consisting of keystroke timings of Unix/Linux IT system administration commands. We evaluated the authentication performance of our novel dataset on three algorithms, i.e., the Scaled Manhattan distance, and the so-called new distance metric (Zhong et al., 2012) with/without fusion. We compared our result with that of the state-of-the-art CMU dataset. The best 95% confidence interval of EER for our Linux Command dataset was (0.038, 0.044) which was very close to that of the CMU dataset (0.027, 0.031) despite the small size of our dataset.

1 INTRODUCTION

According to NIST (2012), an Advanced Persistent Threat (APT) is "an adversary that possesses sophisticated levels of expertise and significant resources which allow it to create opportunities to achieve its objectives by using multiple attack vectors (e.g., cyber, physical, and deception). These objectives typically include establishing and extending footholds within the information technology infrastructure of the targeted organizations for purposes of exfiltrating information, undermining or impeding critical aspects of a mission, program, or organization; or positioning itself to carry out these objectives in the future."

An APT is a well-resourced adversary engaged in sophisticated malicious cyber activity that is targeted and aimed at prolonged network/system intrusion. APT objectives could include espionage, data theft, and network/system disruption or destruction. Since a great deal of effort and resources are required to carry out APT attacks, CISA: Cyber & Infrastructure Security Agency (2023a) believes that APTs are often carried out by nation-state actors that select high-value targets, such as large corporations or critical government agencies. According to NIST (2012), *the advanced persistent threat: (i) pursues its objectives repeatedly over an extended period of time; (ii) adapts to defenders' efforts to resist it; and (iii) is determined to maintain the level of interaction needed to execute its objectives."*

In this work, we propose a novel application of behavioral biometrics (Ray-Dowling et al., 2023), particularly keystroke dynamics (Banerjee and Woodard, 2012), to continuously detect APT actors as an additional layer of security beyond the initial logins. This is motivated by the observation that once inside the targeted system, the APT actors will attempt to achieve administrative rights or other objectives by typing commands on command lines. Indeed, this observation appears to be consistent with the steps that APT actors took as documented in several recent high-profile APT cases, e.g., CISA: Cyber & Infras-

[a] https://orcid.org/0009-0008-4412-4917
[b] https://orcid.org/0009-0007-6470-9042
[c] https://orcid.org/0000-0001-8401-7157
[d] https://orcid.org/0000-0001-8971-1850
[e] https://orcid.org/0000-0002-9365-9642
[*] Equal contribution.

Wang, X., Shadman, R., Hou, D., Hussain, F. and Schuckers, S.
A Novel Keystroke Dataset for Preventing Advanced Persistent Threats.
DOI: 10.5220/0012558000003654

In *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024)*, pages 894-901
ISBN: 978-989-758-684-2; ISSN: 2184-4313

tructure Security Agency (2023b); DOJ (2023a,b). Behavioral biometric authentication identifies a person based on the unique patterns exhibited when they interact with a computing device, such as a tablet, a smartphone or a computer (including mouse and keyboard). Research has shown that like fingerprints or iris, the typing patterns of individuals also tend to be unique. Therefore, it is possible to differentiate imposters from genuine users using their typing patterns and keystroke dynamics. Based on these, we propose to study the detection of APT actors based on their keystroke dynamics when typing commands on command lines.

Since keystroke dynamics can be made to work whenever an APT actor is typing on the keyboard, it would naturally be a good match for APT detection. As elaborated in Section 2, recognizing that intrusion detection based on the content of command lines alone is a hard problem, Schonlau et al. (2001) suggested using command lines in conjunction with other approaches including behavioral biometrics such as keystroke dynamics. This paper takes the first step in this direction by collecting a Linux command keystroke dynamics dataset and conducting algorithmic evaluation.

A keystroke biometric authentication system offers several benefits. Firstly, it is cost-effective since no extra hardware is needed; the regular keyboard alone suffices. Additionally, it is non-intrusive as it does not impose any additional burden on the user. Keystroke dynamics functions by initially generating a user-specific enrollment template based on their typing patterns. During the verification process, a user's test sample is compared with their own enrollment template, and a matching score is calculated and compared with a set threshold for decision-making.

In our work, we developed a keylogger to record the keystrokes and created a novel keystroke dataset of 33 subjects typing a set of eleven *system administration commands*. We then extracted different features of the keystrokes, including timing features. To investigate the potential of this dataset in authentication, we measured the performance of three widely-used efficient algorithms using our novel dataset and compared it with the state-of-the-art CMU dataset (Killourhy and Maxion, 2009).

This paper is organized as follows. Section 2 describes related work. Section 3 describes the data collection procedure. Sections 4, 5, and 6 present our algorithms, experimental setup, and the evaluation of both the new dataset and the CMU dataset. Lastly, Section 7 concludes the paper.

2 RELATED WORK

Commands Usage-Based APT Detection. Prior work in behavior-based masquerader detection has been focused on statistical analysis of Unix command lines since different users tend to prefer to use different sets of commands. In particular, (Schonlau et al., 2001) presented a useful collection of pioneering masquerader research based on a profile of command-line usage, in which six masquerader detection techniques were applied to a dataset collected using the Unix acct auditing mechanism. In terms of detecting masqueraders, the best detection result reported was for a Bayes One-Step Markov model, which achieved a hit rate of 69.3% with a corresponding false-alarm rate of 6.7%. In terms of minimizing false alarms (targeted at 1%), their best result was obtained by using a "uniqueness" metric that achieved a 39.4% hit rate with a corresponding false-alarm rate of 1.4%. Using Schonlau et al.'s dataset (Schonlau et al., 2001), Maxion and Townsend (Maxion and Townsend, 2002) extended the work from one-class classifiers to the Naïve Bayes binary classifier, achieving a 56% improvement in masquerader detection with a corresponding false-alarm rate of just 1.3%.

Due to a limitation of the Unix acct auditing mechanism, Schonlau et al.'s dataset (Schonlau et al., 2001) includes only the user-id and the commands themselves. To overcome this limitation and study the effects of an enriched command line that also includes options and arguments, Maxion (Maxion, 2003) applied the binary naïve bayes classifier to the Greenberg dataset (Greenberg, 1988). The enriched command lines were found to facilitate correct detection at the 82% level, far exceeding previous results, with a corresponding 30% reduction in the overall cost of errors, and only a small increase in false alarms. Maxion and Townsend subsequently performed an in-depth error analysis that reveals more insights about the factors at work in the detection process (Maxion and Townsend, 2004). It is also noteworthy that unlike Schonlau et al.'s work, Maxion uses only 10 rather than 100 commands in masquerader detection.

Recognizing that intrusion detection based on command lines is a hard problem, Schonlau et al. was the first to suggest using command lines in conjunction with other approaches including behavioral biometrics such as keystroke dynamics. This paper takes the first step in this direction by collecting a Linux command keystroke dataset and conducting algorithm evaluation.

Keystroke Datasets. In the literature, there are several keystroke datasets. Wahab et al. (2021) collected keystroke data from students and university staff. The subjects filled out an account recovery web form of multiple fields viz Full name, Address, City, Zip, Phone, Email, Declaration, and Password. 500,000 keystrokes were collected from 44 students and university staff. Tschinkel et al. (2017) collected keystrokes obtained from spreadsheet and web browsing input. Authentication was performed using both text and numeric keypad entries. Vural et al. (2014) collected keystroke data for short passphrases, free-text questions and transcription tasks. 39 subjects participated in the data collection. In the case of CMU dataset (Killourhy and Maxion, 2009), there were 51 subjects. Each subject typed the password ".tie5roanl" 400 times in 8 sessions (50 times per session). A more detailed discussion of several keystroke datasets can be be found in a recent survey on keystroke dynamics (Shadman et al., 2023).

Runtime Feature-Based APT Detection. In prior work, a classification model was introduced to detect Advanced Persistent Threats (APT) (Chandran et al., 2015). To build the model, they gathered APT malware samples from the internet and extracted various features. These features were used to train the model, which was subsequently tested on a target system. Whenever an APT attack was detected on the system, the model triggered an alert signal. They employed various models for this task, and the Random Forest model exhibited the highest accuracy of 99.8%.

In (Mirza et al., 2014), APT countermeasures focused on detecting malware through a technique called windows function hooking. This approach was employed to identify zero-day attacks, which are previously unknown and unaddressed vulnerabilities. Malware often calls specific windows functions that are essential for completing the attack. These particular functions are typically APIs accessed by specialized Dynamic Link Libraries (DLLs). By utilizing function hooking, they intercepted low-level DLLs, enabling observation of potential malware executions. While these methods can be effective, they typically rely on prior knowledge of attacks to train the model.

3 DATA COLLECTION: THE LINUX COMMAND KEYSTROKE DATASET

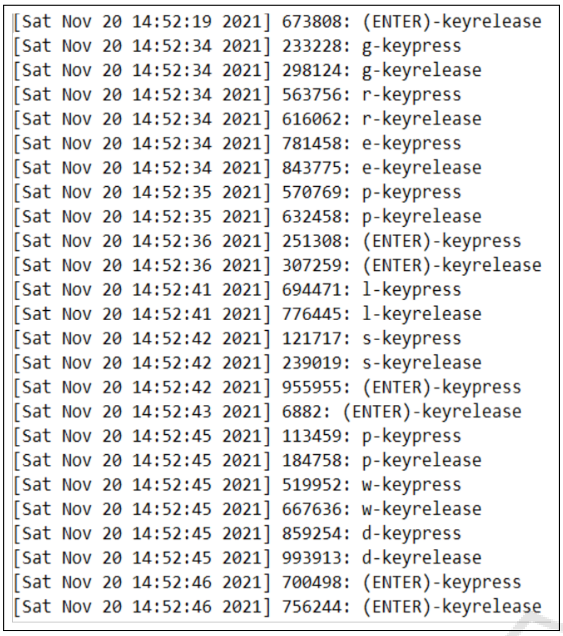
In order to generate a novel Linux Command keystroke dataset, we performed the data collection on Linux. The Linux operating systems record keystroke events as a user types on the keyboard. We can access these events through the Linux IO library: <linux/input.h>. We developed a Linux keylogger which once initialized, continues working in the background, until discontinued. This keylogger records the event of every key press and key release as shown in Figure 1. It also records the time of every key press and key release. As a result, latency (time between consecutive key presses/ key releases) and hold time (time between press and release of a key) can be extracted. Table 1 depicts the list of common Linux system administration commands that we used in our data collection.

Table 1: Eleven common Linux administration commands participants typed during data collection.

Linux Commands	**Functionality**
grep	Print lines that match patterns
ls	List directory contents
pwd	Print name of current/working directory
uptime	Tell how long the system has been running
ps	Report a snapshot of the current process
fdisk	Manipulate disk partition table
kill	Send a signal to a process
cd	Change directory
ifconfig	Configure a network interface
du	Estimate file space usage
df	Report file system disk space usage

We recruited 33 subjects to participate in the study. The task was for the subject to type the eleven Linux commands in Table 1 in a fixed order. These commands have a total of 40 characters. We conducted one session for most of the users, which took about 45 minutes. There were 50 repetitions of the Linux commands by each user. We collected those keystrokes and extracted the timing features. In light of the CMU password dataset (Killourhy and Maxion, 2009), this list of eleven Linux commands, when combined, can be considered as working like a long password when typed in a fixed order.

For the data collection, we used our Linux keylogger that records the timestamp of every key press and key release in a log file. Figure 1 depicts a sample log file. In the log file, time is recorded in HH:MM:SS format followed by the number of microseconds. This log file is used as input to a python code. The python



```
[Sat Nov 20 14:52:19 2021] 673808: (ENTER)-keyrelease
[Sat Nov 20 14:52:34 2021] 233228: g-keypress
[Sat Nov 20 14:52:34 2021] 298124: g-keyrelease
[Sat Nov 20 14:52:34 2021] 563756: r-keypress
[Sat Nov 20 14:52:34 2021] 616062: r-keyrelease
[Sat Nov 20 14:52:34 2021] 781458: e-keypress
[Sat Nov 20 14:52:34 2021] 843775: e-keyrelease
[Sat Nov 20 14:52:35 2021] 570769: p-keypress
[Sat Nov 20 14:52:35 2021] 632458: p-keyrelease
[Sat Nov 20 14:52:36 2021] 251308: (ENTER)-keypress
[Sat Nov 20 14:52:36 2021] 307259: (ENTER)-keyrelease
[Sat Nov 20 14:52:41 2021] 694471: l-keypress
[Sat Nov 20 14:52:41 2021] 776445: l-keyrelease
[Sat Nov 20 14:52:42 2021] 121717: s-keypress
[Sat Nov 20 14:52:42 2021] 239019: s-keyrelease
[Sat Nov 20 14:52:42 2021] 955955: (ENTER)-keypress
[Sat Nov 20 14:52:43 2021] 6882: (ENTER)-keyrelease
[Sat Nov 20 14:52:45 2021] 113459: p-keypress
[Sat Nov 20 14:52:45 2021] 184758: p-keyrelease
[Sat Nov 20 14:52:45 2021] 519952: w-keypress
[Sat Nov 20 14:52:45 2021] 667636: w-keyrelease
[Sat Nov 20 14:52:45 2021] 859254: d-keypress
[Sat Nov 20 14:52:45 2021] 993913: d-keyrelease
[Sat Nov 20 14:52:46 2021] 700498: (ENTER)-keypress
[Sat Nov 20 14:52:46 2021] 756244: (ENTER)-keyrelease
```

Figure 1: Sample log file generated by Linux keylogger. This log file contains the key press time and key release time of every key in HH:MM:SS format and also in microsecond after second.

code extracts all the timing features from the log file and outputs the timing values which are used as data input to the authentication algorithm. Two timing features between every pair of successive keys of a Linux command as well as the hold time of every key were considered:

- PP = key press time of key2 – key press time of key1
- RP = key press time of key2 – key release time of key1
- H = key release time – key press time

These are the features that were used to compare the typing pattern of two users.

Table 2 depicts timing features of first two Linux commands, ‘grep’ and ‘ls’. As it is shown in Table 2, no timing feature was considered between the keystrokes of two consecutive Linux commands, e.g. between ‘p’ of ‘grep’ and ‘l’ of ‘ls’. Figure 2 depicts an example CSV file of the processed keystroke data. In the .csv file, the first column shows the index of the user. The second column shows the session number of the user. The third column shows the number of repetition of the Linux commands in the corresponding session. In the next columns, the keystroke timing data is inputted. This .csv file was used as input to the authentication algorithms.

Upon request our dataset can be made freely available to others for research purposes.

4 ALGORITHMS

In this section, we describe the algorithms that we use to evaludate our dataset. The Scaled Manhattan distance and the new distance metric were introduced in earlier work. We apply fusion to the new distance metric for better performance.

4.1 Algorithm 1: Scaled Manhattan Distance

In (Killourhy and Maxion, 2009), Killourhy and Maxion used 14 anomaly detectors to perform classification of genuine users and imposter users. The Scaled Manhattan distance showed the best performance. As described by Araujo et al. (2004), this detector has an improvement over the Manhattan algorithm. The method of the Scaled Manhattan distance is easily understood and shown in Equation 1. Initially, we need to calculate the mean vector and the mean absolute deviation of each feature. In the testing phase, the same as in the Manhattan distance method, the genuine anomaly score is calculated as difference in i-th features of the test and mean vectors which is then divided by the average absolute deviation from the training phase.

$$d = \sum_{i=1}^{n} \frac{|X_i - Y_i|}{A_i} \tag{1}$$

4.2 Algorithm 2: New Distance Metric

In (Zhong et al., 2012), Zhong, Deng, and Jain introduced a new distance metric by combining ideas from Mahalanobis distance and Manhattan distance. With the CMU dataset, this new distance metric achieved an EER of 0.087, which is better than the 0.096 EER for Scaled Manhattan distance on the same dataset. In the testing phase, we calculate the genuine scores and imposter scores with Equation 2, where x and y are points in genuine training data and testing sample data, respectively, and $S^{-\frac{1}{2}}$ is the inverse of the square root of covariance matrix S. Equation 3 shows the calculation of $S^{\frac{1}{2}}$, the square root of covariance matrix S, where P is the matrix of all eigenvectors of S.

$$||x-y||' = ||x'-y'||_1 = ||S^{-\frac{1}{2}}(x-y)||_1 \tag{2}$$

$$S^{\frac{1}{2}} = P*(P^{-1}*S*P)^{\frac{1}{2}}*P^{-1} \tag{3}$$

Table 2: Example timing features from two Linux commands, *grep* and *ls*.

Linux Command Keys	Timing Features
g	H.g
	PP.g.r
	RP.g.r
r	H.r
	PP.r.e
	RP.r.e
e	H.e
	PP.e.p
	RP.e.p
p	H.p
l	H.l
	PP.l.s
	RP.l.s
s	H.s

4.3 Algorithm 3: New Distance Metric (with Fusion)

Since using more than one sample to generate genuine scores and imposter scores is believed to be more accurate for making classification decisions, we apply fusion to the new distance metric method. In this paper, we use the minimum, average, and maximum of multiple scores to generate the final decision score. We perform fusion with 2, 3, 4, 5 samples per fusion, to evaluate the impact of the number of samples on performance of different fusion methods.

5 EXPERIMENTAL SETUP

Our goal was to comparatively evaluate the performance of the three algorithms on two datasets - our Liunx Command dataset and the Carnegie Mellon University (CMU) dataset. For this, we divided the CMU data into training and testing sets with the same size as in (Killourhy and Maxion, 2009). We randomly selected 40 out of the 51 subjects as training/testing data, for 50 times. Each subject was used as the genuine user once with all the others as imposters. Training made use of each subject's first 200 feature vectors, and the remaining 200 vectors were used as genuine test samples and first 5 samples of every other user as imposter test samples, for a total of 250 imposter test samples.

Table 3: EER results of two algorithms using two different datasets- CMU and Linux commands.

Algorithm	CMU Dataset	Linux Command
Scaled Manhattan Distance	(0.093, 0.096)	(0.140, 0.144)
New Distance Metric	(0.087, 0.089)	(0.281, 0.285)

For our own Linux Command dataset, we randomly selected 25 out of the 33 subjects as training and testing data, for 50 times. For each subject we selected first 20 feature vectors as genuine training samples. The remaining 30 feature vectors were used as genuine testing samples, and the first one sample from each of the remaining 32 subjects was used to form 32 imposter testing samples.

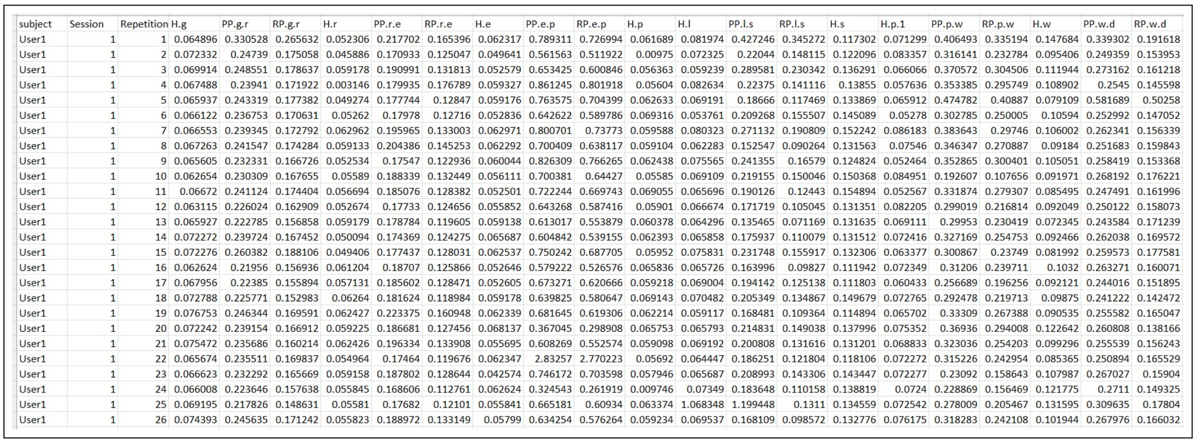
After that, we used the three algorithms for these two datasets to calculate both genuine scores and imposter scores, which were used to plot ROC curves (Receiver Operating Characteristic) and obtain the final Equal Error Rate (EER). EER is a commonly used error rate to evaluate the classification performance in keystroke dynamics. In order to obtain a reliable estimation of performance, we tested each algorithm 50 times and calculated the 95% confidence interval of the EER values.

6 RESULTS & DISCUSSION

After testing the three different algorithms mentioned earlier for both CMU and Linux Command datasets, the results for Scaled Manhattan distance and new distance metric are shown in Table 3, and the fusion method results of the two datasets are shown in Table 4 and Table 5.

Table 3 shows the average performance across all subjects in each dataset. In addition, we also generated sample ROC curves for Scaled Manhattan distance and new distance metric method, respectively, showing two subjects who have the best EER performance and two subjects who have the worst EER performance. Figures 3 and 4 show the ROCs for the Linux Command dataset and CMU dataset, respectively, complementing the result in Table 3.

The fusion method by using the minimal (min)



subject	Session	Repetition	H.g	PP.g.r	RP.g.r	H.r	PP.r.e	RP.r.e	H.e	PP.e.p	RP.e.p	H.p	H.l	PP.l.s	RP.l.s	H.s	H.p.1	PP.p.w	RP.p.w	H.w	PP.w.d	RP.w.d
User1	1	1	0.064896	0.330528	0.265632	0.052306	0.217702	0.165396	0.062317	0.789311	0.726994	0.061689	0.081974	0.427246	0.345272	0.117302	0.071299	0.406493	0.335194	0.147684	0.339302	0.191618
User1	1	2	0.072332	0.24739	0.175058	0.045886	0.170933	0.125047	0.049641	0.561563	0.511922	0.00975	0.072325	0.22044	0.148115	0.122096	0.083357	0.316141	0.232784	0.095406	0.249359	0.153953
User1	1	3	0.069914	0.248551	0.178637	0.059178	0.190991	0.131813	0.052579	0.653425	0.600846	0.056363	0.059239	0.289581	0.230342	0.136291	0.066066	0.370572	0.304506	0.111944	0.273162	0.161218
User1	1	4	0.067488	0.23941	0.171922	0.003146	0.179935	0.176789	0.059327	0.861245	0.801918	0.05604	0.082634	0.22375	0.141116	0.13855	0.057636	0.353385	0.295749	0.108902	0.2545	0.145598
User1	1	5	0.065937	0.243319	0.177382	0.049274	0.177744	0.12847	0.059176	0.763575	0.704399	0.062633	0.069191	0.18666	0.117469	0.133869	0.065912	0.474782	0.40887	0.079109	0.581689	0.50258
User1	1	6	0.066122	0.236753	0.170631	0.05262	0.17978	0.12716	0.052836	0.642622	0.589786	0.069316	0.053761	0.209268	0.155507	0.145089	0.05278	0.302785	0.250005	0.10594	0.252992	0.147052
User1	1	7	0.066553	0.239345	0.172792	0.062962	0.195965	0.133003	0.062971	0.800701	0.73773	0.059588	0.080323	0.271132	0.190809	0.152242	0.086183	0.383643	0.29746	0.106002	0.262341	0.156339
User1	1	8	0.067263	0.241547	0.174284	0.059133	0.204386	0.145253	0.062292	0.700409	0.638117	0.059104	0.062283	0.152547	0.090264	0.131563	0.07546	0.346347	0.270887	0.09184	0.251683	0.159843
User1	1	9	0.065605	0.232331	0.166726	0.052534	0.17547	0.122936	0.060044	0.826309	0.766265	0.062438	0.075565	0.241355	0.16579	0.124824	0.052464	0.352865	0.300401	0.105051	0.258419	0.153368
User1	1	10	0.062654	0.230309	0.167655	0.05589	0.188339	0.132449	0.056111	0.700381	0.64427	0.05585	0.069109	0.219155	0.150046	0.150368	0.084951	0.192607	0.107656	0.091971	0.268192	0.176221
User1	1	11	0.06672	0.241124	0.174404	0.056694	0.185076	0.128382	0.052501	0.722244	0.669743	0.069055	0.065696	0.190126	0.12443	0.154894	0.052567	0.331874	0.279307	0.085495	0.247491	0.161996
User1	1	12	0.063115	0.226024	0.162909	0.052674	0.17733	0.124656	0.055852	0.643268	0.587416	0.05901	0.066674	0.171719	0.105045	0.131351	0.082205	0.299019	0.216814	0.092049	0.250122	0.158073
User1	1	13	0.065927	0.222785	0.156858	0.059179	0.178784	0.119605	0.059138	0.613017	0.553879	0.060378	0.064296	0.135465	0.071169	0.131635	0.069111	0.29953	0.230419	0.072345	0.243584	0.171239
User1	1	14	0.072272	0.239724	0.167452	0.050094	0.174369	0.124275	0.065687	0.604842	0.539155	0.062393	0.065858	0.175937	0.110079	0.131512	0.072416	0.327169	0.254753	0.092466	0.262038	0.169572
User1	1	15	0.072276	0.260382	0.188106	0.049406	0.177437	0.128031	0.062537	0.750242	0.687705	0.05952	0.075831	0.231748	0.155917	0.132306	0.063377	0.300867	0.23749	0.081992	0.259573	0.177581
User1	1	16	0.062624	0.21956	0.156936	0.061204	0.18707	0.125866	0.052646	0.579222	0.526576	0.065836	0.065726	0.163996	0.09827	0.111942	0.072349	0.31206	0.239711	0.1032	0.263271	0.160071
User1	1	17	0.067956	0.22385	0.155894	0.057131	0.185602	0.128471	0.052605	0.673271	0.620666	0.059218	0.069004	0.194142	0.125138	0.111803	0.060433	0.256689	0.196256	0.092121	0.244016	0.151895
User1	1	18	0.072788	0.225771	0.152983	0.06264	0.181624	0.118984	0.059178	0.639825	0.580647	0.069143	0.070482	0.205349	0.134867	0.149679	0.072765	0.292478	0.219713	0.09875	0.241222	0.142472
User1	1	19	0.076753	0.246344	0.169591	0.062427	0.223375	0.160948	0.062339	0.681645	0.619306	0.062214	0.059117	0.168481	0.109364	0.114894	0.065702	0.33309	0.267388	0.090535	0.255582	0.165047
User1	1	20	0.072242	0.239154	0.166912	0.059225	0.186681	0.127456	0.068137	0.367045	0.298908	0.065753	0.065793	0.214831	0.149038	0.137996	0.075352	0.36936	0.294008	0.122642	0.260808	0.138166
User1	1	21	0.075472	0.235686	0.160214	0.062426	0.196334	0.133908	0.055695	0.608269	0.552574	0.059098	0.069192	0.200808	0.131616	0.131201	0.068833	0.323036	0.254203	0.099296	0.255539	0.156243
User1	1	22	0.065674	0.235511	0.169837	0.054964	0.17464	0.119676	0.062347	2.83257	2.770223	0.05692	0.064447	0.186251	0.121804	0.118106	0.072272	0.315226	0.242954	0.085365	0.250894	0.165529
User1	1	23	0.066623	0.232292	0.165669	0.059158	0.187802	0.128644	0.042574	0.746172	0.703598	0.057946	0.065687	0.208993	0.143306	0.143447	0.072277	0.23092	0.158643	0.107987	0.267027	0.15904
User1	1	24	0.066008	0.223646	0.157638	0.055845	0.168606	0.112761	0.062624	0.324543	0.261919	0.009746	0.07349	0.183648	0.110158	0.138819	0.0724	0.228869	0.156469	0.121775	0.2711	0.149325
User1	1	25	0.069195	0.217826	0.148631	0.05581	0.17682	0.12101	0.055841	0.665181	0.60934	0.063374	1.068348	1.199448	0.1311	0.134559	0.072542	0.278009	0.205467	0.131595	0.309635	0.17804
User1	1	26	0.074393	0.245635	0.171242	0.055823	0.188972	0.133149	0.05799	0.634254	0.576264	0.059234	0.069537	0.168109	0.098572	0.132776	0.076175	0.318283	0.242108	0.101944	0.267976	0.166032

Figure 2: CSV file containing all the timing features of a user. The first, second, and third columns show the index of the subject, the session number, and the number of attempt in the corresponding session, respectively. The later columns show all the timing features. Here, H.g means hold time of ‘g’ and PP.g.r means time difference between key press time of ‘r’ and key press time of ‘g’. Similarly RP.g.r indicates time difference between key press time of ‘r’ and key release time of ‘g’. All the timing values are in second.

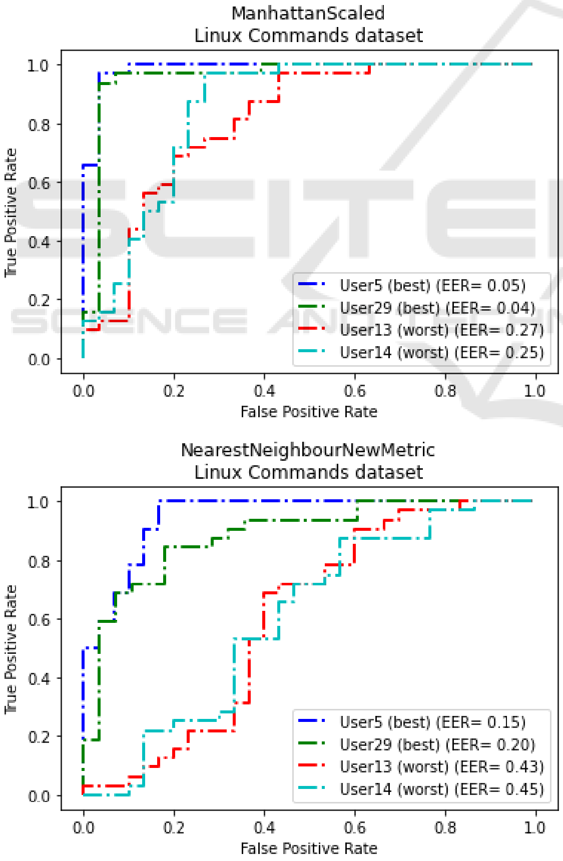


Figure 3: ROC curves for Scaled Manhattan Distance (top) and New Distance Metric (bottom), on Linux Command Dataset, with two subjects who have the best EER and two subjects who have the worst EER.

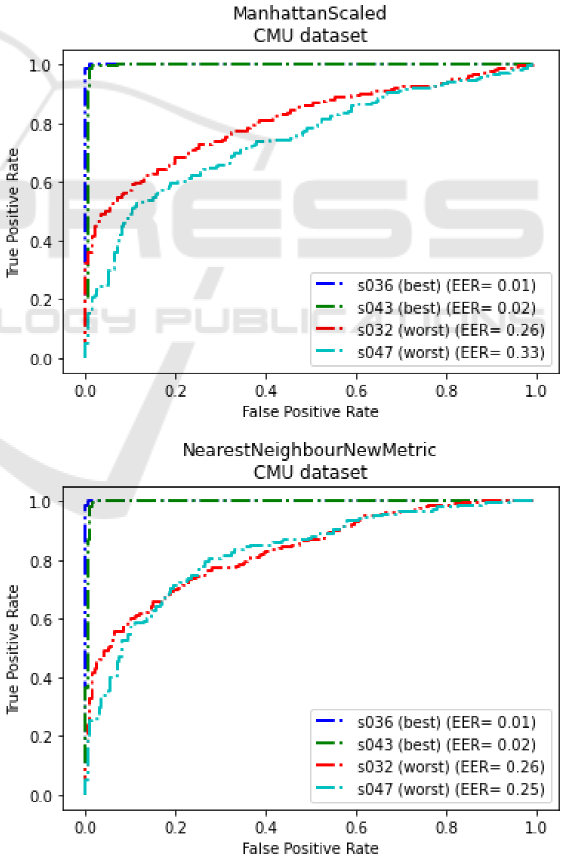


Figure 4: ROC curves for Scaled Manhattan Distance (top) and New Distance Metric (bottom), on the CMU Dataset, with two subjects who have the best EER and two subjects who have the worst EER.

Table 4: EER results of three fusion methods (minimum, maximum and mean) for new distance metric with 2-5 samples per fusion for the Linux command dataset.

Samples	Min	Max	Mean
1	(0.281, 0.285)	(0.281, 0.285)	(0.281, 0.285)
2	(0.181, 0.186)	(0.283, 0.289)	(0.214, 0.220)
3	(0.097, 0.104)	(0.322, 0.335)	(0.198, 0.209)
4	(0.062, 0.069)	(0.332, 0.347)	(0.182, 0.192)
5	(0.038, 0.044)	(0.362, 0.379)	(0.202, 0.219)

Table 5: EER results of three fusion methods (minimum, maximum and mean) for new distance metric with 2-5 samples per fusion for the CMU dataset.

Samples	Min	Max	Mean
1	(0.087, 0.089)	(0.087, 0.089)	(0.087, 0.089)
2	(0.053, 0.055)	(0.082, 0.085)	(0.058, 0.061)
3	(0.045, 0.047)	(0.080, 0.082)	(0.043, 0.045)
4	(0.034, 0.037)	(0.077, 0.079)	(0.034, 0.036)
5	(0.027, 0.031)	(0.090, 0.092)	(0.038, 0.040)

score of several samples at one time as the final score has a productive influence on the performance of both CMU dataset and Linux Command dataset. Moreover, the fusion method improved the EER more for the Linux Command dataset than the CMU dataset. The confidence interval of EER changed from (0.281, 0.285) to (0.038, 0.044) in the case of Linux Command dataset and from (0.087, 0.089) to (0.027, 0.031) in the case of CMU dataset. The maximal (max) score fusion method used the highest score of several samples, and its EER was the worst in all three methods for both datasets. The mean score fusion method used the average score of the multiple samples and was in the middle position in terms of EER.

Due to the small size of our Linux Command dataset, the EER results were not as good as those of the CMU dataset. However, our dataset is the first dataset containing Linux command keystrokes and it has the 95 percent probability that the EER value following in (0.038, 0.044). This shows the quality of our dataset to authenticate the system administrators and prevent APT.

In Table 3, we can see that new distance metric performs better than the Scaled Manhattan distance with CMU dataset. However, in the case of our own Linux Command dataset, new distance metric does not perform better than the Scaled Manhattan distance. We hypothesize that this is due to relatively small training set for the Linux Command dataset (20 samples). To investigate, we rerun the CMU experiment with the same amount of training data as in the Linux dataset (20 samples). As a result, the confidence interval of EER with Scaled Manhattan distance method changed from (0.093, 0.096) to (0.087, 0.090) and the EER of new distance metric method increased from (0.087, 0.089) to (0.219, 0.223). This result confirmed our earlier hypothesis that the small enrollment profile size causes the poor performance of the new distance metric. This would also indicate that the new distance metric is more ‘data hungry’ than the Scaled Manhattan distance.

Our Linux command dataset has only 50 samples per user. To further investigate the impact of data size, we asked two of the original volunteers to increase their data samples to 400 for the list of Linux commands, the same amount as in the CMU dataset. Using the new data, we calculated the EER for those two subjects. The EER values for Scaled Manhattan distance are 0.067 and 0.090, and for new distance metric are 0.107 and 0.177. We observe that using a large enrollment profile improves the EER performance more for the new distance metric than the Scaled Manhattan distance method.

7 CONCLUSION

Our experiments evaluated and compared the performance of our Linux Command dataset with that of the CMU dataset. The new distance metric performs better than Scaled Manhattan distance detector for CMU dataset. In the case of our Linux Command dataset, without fusion, the new distance metric does not improve the Equal Error Rate over the Scaled Manhattan distance, probably due to the limited amount of keystroke data. However, introducing fusion methods to the new distance metric solves this problem. The fusion method has also improved the EER for CMU dataset. In all three fusion methods, using a minimum score to generate one final matching score from several samples yields the best performance.

Increasing the enrollment profile size for two specific users in Linux Command dataset seem to have improved the performance of the Scaled Manhattan distance less than the new distance metric. The reason appears to be that the new distance metric is more sensitive to data size for performance.

As future work, we plan to enlarge our Linux Command dataset with more subjects and with more samples per subject. We shall also investigate more efficient algorithms to improve the performance. We hypothesize that by fusion of command lines and keystroke dynamics, we will significantly improve the performance of intrusion detection. Lastly, we plan to field-test the effectiveness of this method in preventing and detecting advanced persistent threats (APT) (NIST, 2012).

ACKNOWLEDGMENTS

This work were partially supported by NSF Award CNS-1650503. Wang, Hou, and Schuckers were also supported by NSF Award TI-2122746.

REFERENCES

Araujo, L. C., Sucupira, L. H., Lizarraga, M. G., Ling, L. L., and Yabu-uti, J. B. (2004). User authentication through typing biometrics features. In *Biometric Authentication: First International Conference, ICBA 2004, Hong Kong, China, July 15-17, 2004. Proceedings*, pages 694–700. Springer.

Banerjee, S. and Woodard, D. (2012). Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern recognition research*.

Chandran, S., Hrudya, P., and Poornachandran, P. (2015). An efficient classification model for detecting advanced persistent threat. In *2015 international conference on advances in computing, communications and informatics (ICACCI)*, pages 2001–2009. IEEE.

CISA: Cyber & Infrastructure Security Agency (2023a). Advanced persistent threats and nation-state actors–helping cybersecurity defenders protect against and respond to apts. [Online]. Available: CISA APT (2023), https://www.cisa.gov/topics/cyber-threats-and-advisories/advanced-persistent-threats-and-nation-state-actors.

CISA: Cyber & Infrastructure Security Agency (2023b). People's republic of china state-sponsored cyber actor living off the land to evade detection. [Online]. Available: Volt Typhoon (2023), https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-144a.

DOJ (2023a). Chinese Military Personnel Charged with Computer Fraud, Economic Espionage and Wire Fraud for Hacking into Credit Reporting Agency Equifax. [Online]. Available: DOJ Equifax Indictment (2020), https://www.justice.gov/opa/pr/chinese-military-personnel-charged-computer-fraud-economic-espionage-and-wire-fraud-hacking.

DOJ (2023b). U.S. Charges Russian FSB Officers and Their Criminal Conspirators for Hacking Yahoo and Millions of Email Accounts. [Online]. Available: DOJ Yahoo Indictment (2017), https://www.justice.gov/opa/pr/us-charges-russian-fsb-officers-and-their-criminal-conspirators-hacking-yahoo-and-millions.

Greenberg, S. (1988). Using unix: Collected traces of 168 users. *University of Calgary*.

Killourhy, K. S. and Maxion, R. A. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE.

Maxion, R. A. (2003). Masquerade detection using enriched command lines. In *2003 International Conference on Dependable Systems and Networks*, pages 5–14. IEEE.

Maxion, R. A. and Townsend, T. N. (2002). Masquerade detection using truncated command lines. In *Proceedings international conference on dependable systems and networks*, pages 219–228. IEEE.

Maxion, R. A. and Townsend, T. N. (2004). Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability*, 53(1):124–147.

Mirza, N. A. S., Abbas, H., Khan, F. A., and Al Muhtadi, J. (2014). Anticipating advanced persistent threat (apt) countermeasures using collaborative security mechanisms. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pages 129–132. IEEE.

NIST (2012). *NIST Special Publication 800-39 Managing Information Security Risk*. CreateSpace, Scotts Valley, CA.

Ray-Dowling, A., Hou, D., and Schuckers, S. (2023). Stationary mobile behavioral biometrics: A survey. *Computers & Security*, 128:103184.

Schonlau, M., DuMouchel, W., Ju, W.-H., Karr, A. F., Theus, M., and Vardi, Y. (2001). Computer intrusion: Detecting masquerades. *Statistical science*, pages 58–74.

Shadman, R., Wahab, A. A., Manno, M., Lukaszewski, M., Hou, D., and Hussain, F. (2023). Keystroke dynamics: Concepts, techniques, and applications. [Online]. Available: arXiv:2303.04605, 2023.

Tschinkel, B., Esantsi, B., Iacovelli, D., Nagesar, P., Walz, R., Monaco, V., and Bakelman, N. (2017). Keylogger keystroke biometric system. [Online]. Available: Research Gate (2017).

Vural, E., Huang, J., Hou, D., and Schuckers, S. (2014). Shared research dataset to support development of keystroke authentication. In *IEEE International joint conference on biometrics*, pages 1–8. IEEE.

Wahab, A. A., Hou, D., Schuckers, S., and Barbir, A. (2021). Utilizing keystroke dynamics as additional security measure to protect account recovery mechanism. In *ICISSP*, pages 33–42.

Zhong, Y., Deng, Y., and Jain, A. K. (2012). Keystroke dynamics for user authentication. In *2012 IEEE computer society conference on computer vision and pattern recognition workshops*, pages 117–123. IEEE.