Keystroke Dynamics

Name: Danny O’Leary

Student Number: 20067817

Module: Biological Authentication Systems

BSc (Hons) in Computer Forensics and Security

Contents

Introduction

Keystoke dynamics or typing dynamics is the method of identifying or attempting to confirm the identity of a person. The basic process is to take patterns of how a person types in a password for example. It takes in many different characteristics to train a Machine Learning model that will be used to attempt to confirm if a user is who they say they are. The characteristics are often things such as how long a key is pressed down, how long it takes them to enter a password, or the time in between the key presses (Biometrics, 2017).

The history of keystroke dynamics comes from the early days of telegraph. During World War 2, a method called '*Fist of the sender*' which was used to identify the source of where Morse Code was coming from (SearchSecurity, 2017). This was done using the same process as keystroke dynamics, where they were able to detect the pattern of how the message was sent. This is done using the time in between sent dots/dashes.

Since the growing use in computers, and the fact of how accessible they are now it makes it so that keystroke dynamics is one of the biometrics that is easy to implement for most of the worlds population towards some of the more '*Expensive*', or things like fingerprint scanners where you would need hardware to carry out the check.

One of the most common usages of this type of biometric is for what is known as password hardening. Password hardening is the process of is doing something extra to make the password more secure. This process is often used as a form of 2-Factor Authentication where you must do more than one thing to login. As an example, signing in with a password may need you to use your phone where they send you a code via a text message that you need to enter. In the case of keystoke dynamics, this can be used just as your typing behaviour rather than a text message. This means that it seemingly integrates, and the user doesn't have to do anything additional (If the system works in principal) (eWEEK, 2017).

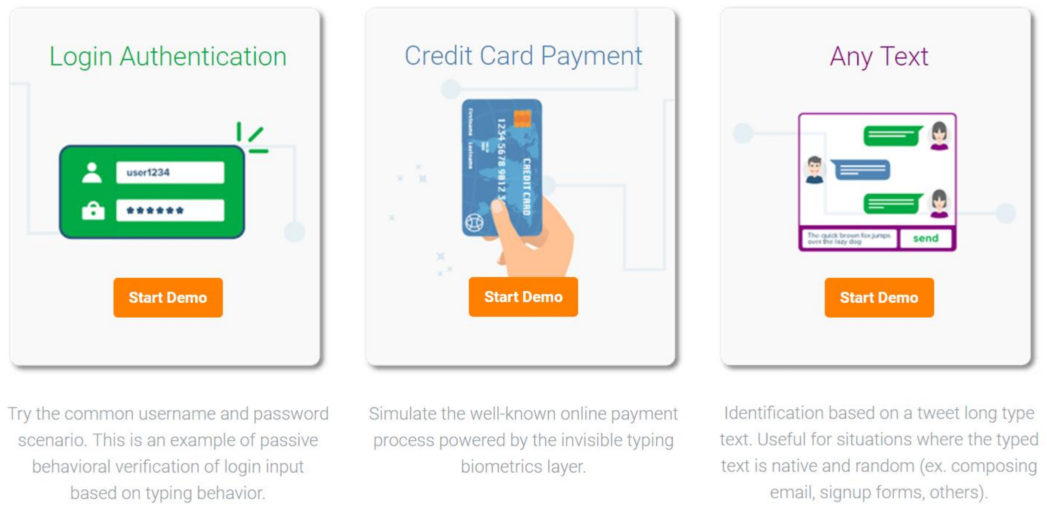
Keystroke dynamics falls under the behavioural type of biometric, and in this report, it will be looked at in detail, and compared to other biometric systems in regard to performance. This report will also have a practical element where I will introduce a simple implementation of a keystroke dynamic system.

Application

Keystroke dynamics is often used for authentication, more so for the use of two-factor authentication in the use of typing passwords into a system. It can also be used in some surveillance systems where they are trying to identify people via their typing style (Biometric, 2017). Biometric companies have adopted some use of keystroke dynamics in their systems and some of these companies are as follows:

TypingDNA

TypingDNA is a company that uses keystroke dynamics to identify a person. It uses an AI engine to match two different typing patterns to see how close they are together. It provides an API for companies to implement solutions that do this for them. As an example, a website might want to implement keystroke dynamics as a form of 2-factor authentication on their website, so they go to TypingDNA, and TypingDNA then provides an api for them to implement a custom solution for this. The website provides a demo where you can try out the software:

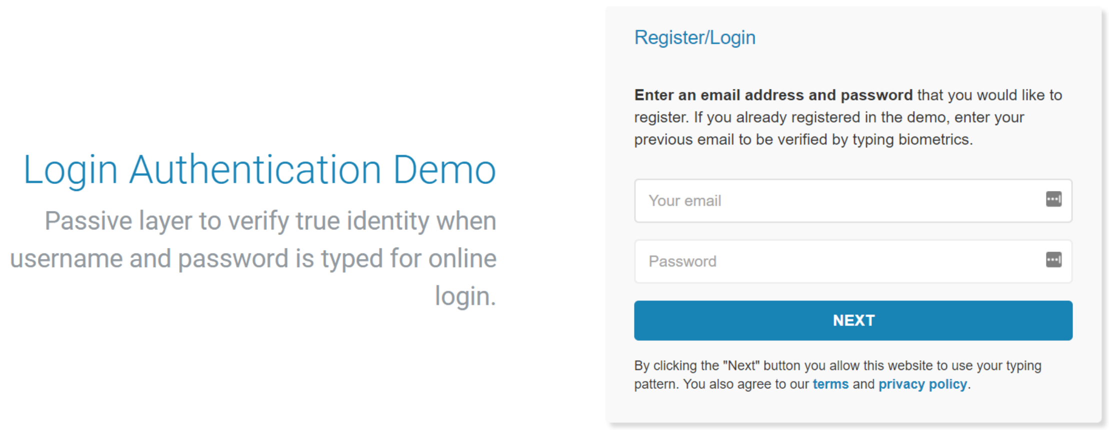


Try the common username and password scenario. This is an example of passive behavioral verification of login input based on typing behavior.

Simulate the well-known online payment process powered by the invisible typing biometrics layer.

Identification based on a tweet long type text. Useful for situations where the typed text is native and random (ex. composing email, signup forms, others).

Figure 1. It provides a picture of the different types of things that the service can be used for. In this example, I am going to try the login authentication system.


Login Authentication Demo

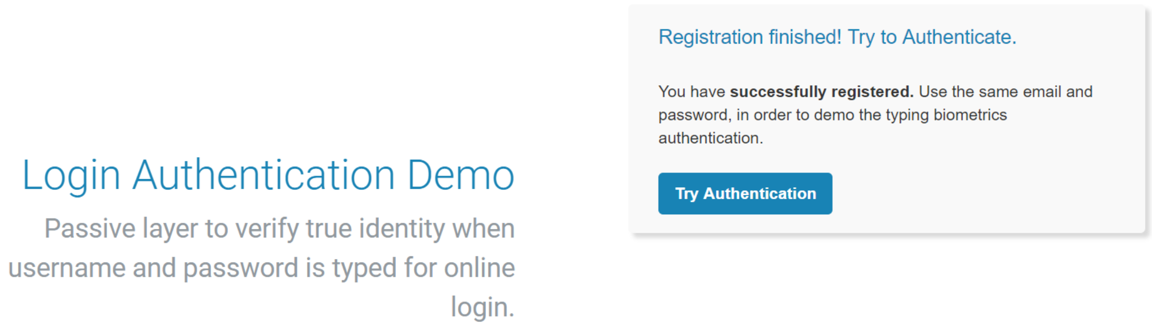
Passive layer to verify true identity when username and password is typed for online login.

Register/Login

Enter an email address and password that you would like to register. If you already registered in the demo, enter your previous email to be verified by typing biometrics.

By clicking the "Next" button you allow this website to use your typing pattern. You also agree to our terms and privacy policy.

Figure 2. It now asks me for an email address and password which will be used as the login authentication example. I will login using my email typed as I normally would for this example.

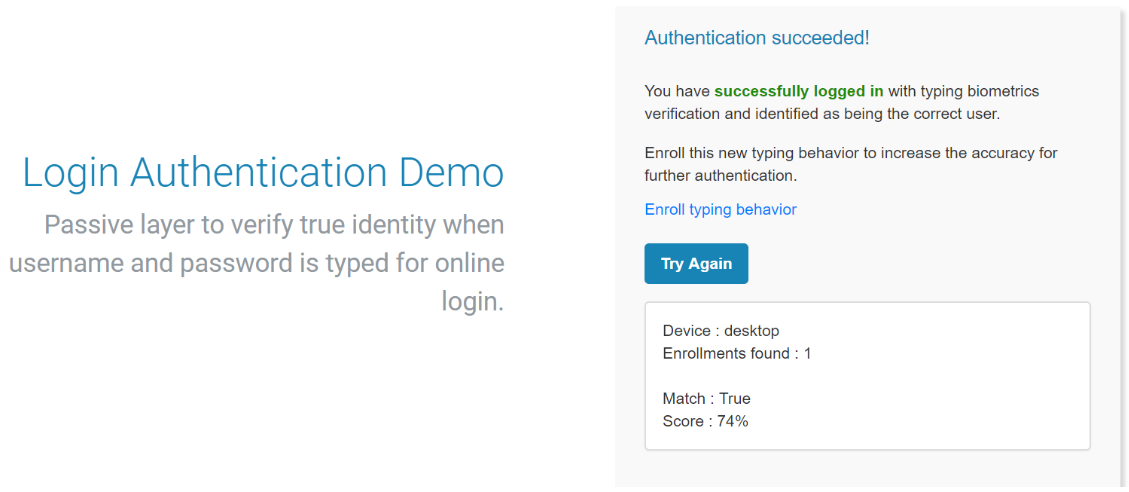

Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

Registration finished! Try to Authenticate.

You have **successfully registered.** Use the same email and password, in order to demo the typing biometrics authentication.

Figure 3. I can now try out the authentication service that they provide. I now try authentication and get the same screen as in figure 1 which I type my password into correctly again


Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

Authentication succeeded!

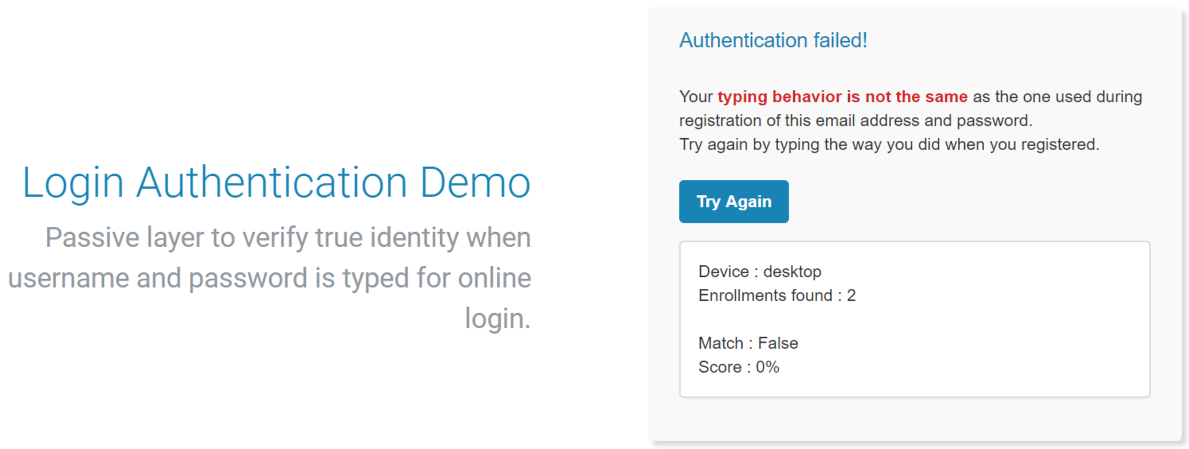
You have **successfully logged in** with typing biometrics verification and identified as being the correct user.

Enroll this new typing behavior to increase the accuracy for further authentication.

```
Device : desktop
Enrollments found : 1

Match : True
Score : 74%
```

Figure 4. As shown in this figure, it shows a successful login attempt using the system. It came back with a 74% score, and

allows me to add this as a potential result. I will add the result to improve the accuracy, and now will attempt to enter the details much slower, and not how I would normally type.



Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

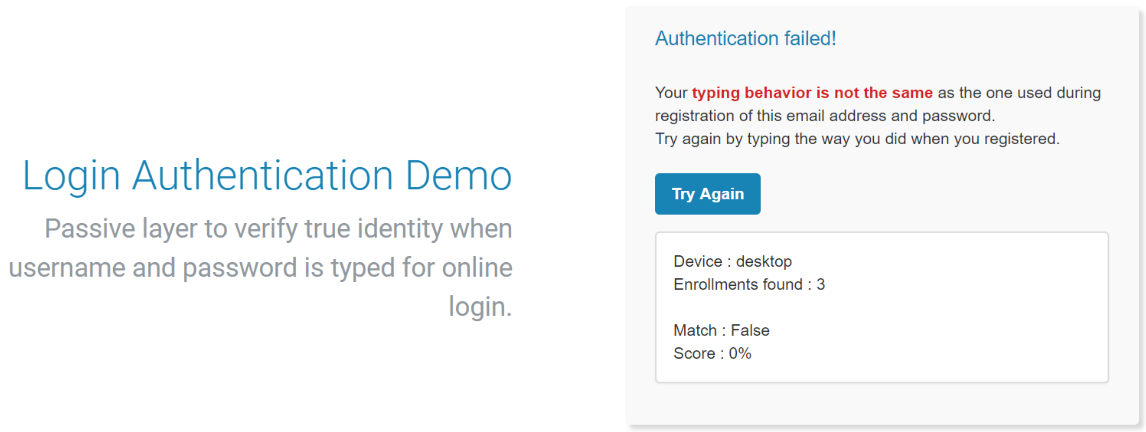
Authentication failed!

Your **typing behavior is not the same** as the one used during registration of this email address and password.
Try again by typing the way you did when you registered.

Device : desktop
Enrollments found : 2

Match : False
Score : 0%

Figure 5. As can be seen, this time to authentication failed. I made sure to make this as believable as possible with things like holding the shift key for longer and that kind of thing. I now decided it would be a good idea to get another person to try without knowing what it was for and see what the results were.



Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

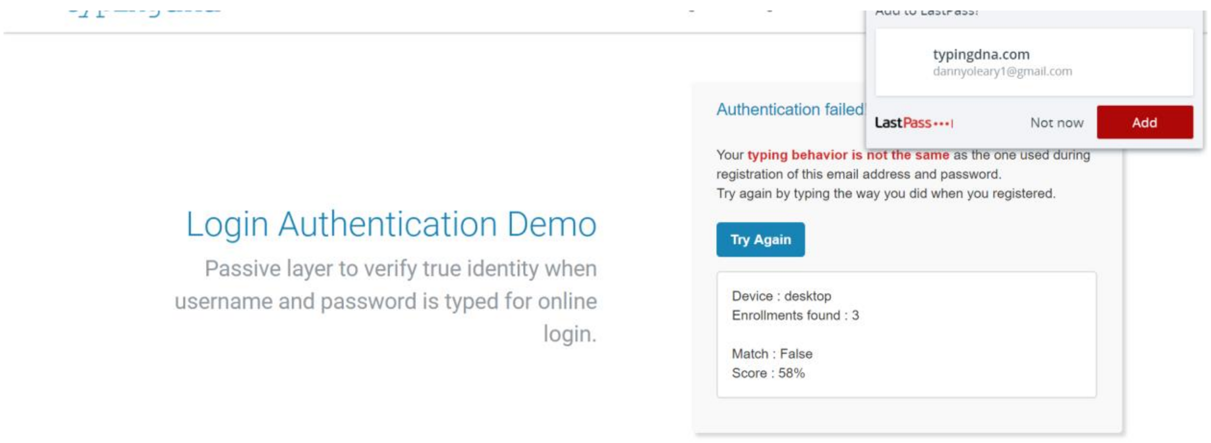
Authentication failed!

Your **typing behavior is not the same** as the one used during registration of this email address and password.
Try again by typing the way you did when you registered.

Device : desktop
Enrollments found : 3

Match : False
Score : 0%

Figure 6. This now shows that the system appears to work somewhat well. Lastly, out of curiosity I wanted to see what would happen if I typed in a password using an automated system that fills it in. E.g. I store the password for this site in a Chrome Extension called LastPass and attempt that to login. I am expecting this to fail too which may be a flaw in this biometric since it is becoming a more common thing to do.


Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

Authentication failed

Your **typing behavior is not the same** as the one used during registration of this email address and password.
Try again by typing the way you did when you registered.

Try Again

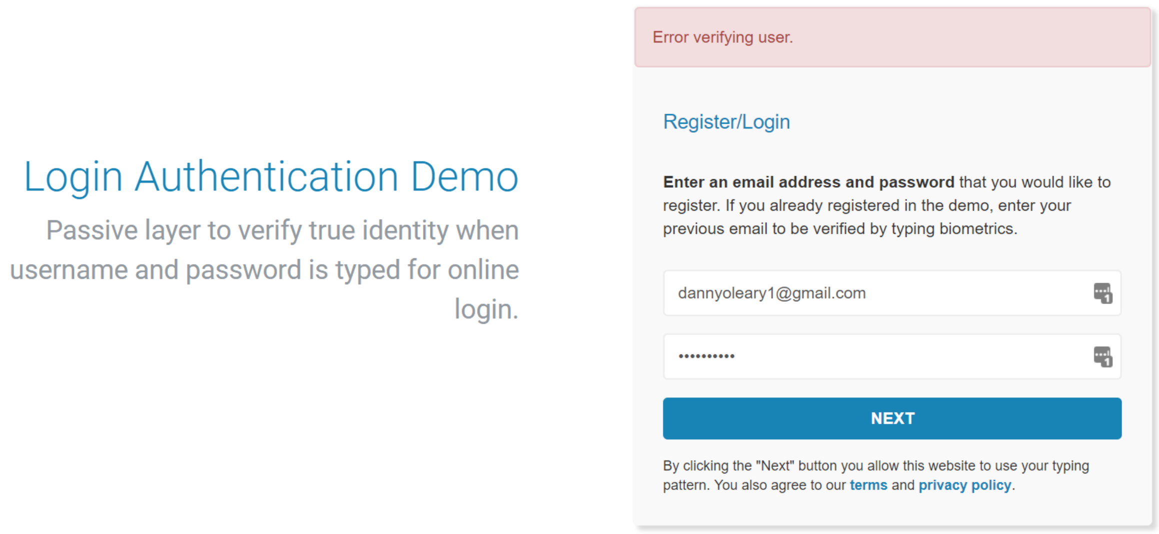
Device : desktop
Enrollments found : 3

Match : False
Score : 58%

typingdna.com
dannyoleary1@gmail.com

LastPass Not now Add

Figure 7. When I was attempting to store the password using LastPass, I got a false-negative on the system. I did enter a character wrong (This worked on some of them however). This is to be expected sometimes but it shows a good example of this system not being perfect.


Login Authentication Demo

Passive layer to verify true identity when username and password is typed for online login.

Error verifying user.

Register/Login

Enter an email address and password that you would like to register. If you already registered in the demo, enter your previous email to be verified by typing biometrics.

dannyoleary1@gmail.com

••••••••••

NEXT

By clicking the "Next" button you allow this website to use your typing pattern. You also agree to our **terms** and **privacy policy**.

Figure 8. When using the automated system, it is unable to verify the user since no keys have been typed into the system. This is probably a good way to handle it (SRL, 2017).

ID Control

ID Control is a Dutch company that have many different biometric technologies in the field. For the case of this report, their keystrokeID product is the only one that is useful. It is also a authentication tool that is set up to analyse keyboard behaviour by the user to compare to what they have recorded for that particular user. According to ID Control its: "*Quick and very secure identification*" which is interesting since they are claiming it to be very secure (Idcontrol.com, 2017).

BehavioSec

Much like the other two companies, BehavioSec is also providing an authentication system with access to an API where comparisons will be provided for you. The interesting thing that BehavioSec does over the other two companies is not only use Keystroke dynamics. They also monitor mouse movements for example. They claim that this will also work for mobile devices including phones, and tablets. It is also claimed to be useful for bot detection on a site if you are trying to prevent bots from breaking certain things on a website (Behaviosec, 2017).

Evaluation

In this section, I will discuss how Keystroke dynamics can be evaluated as a biometric system. I think that one of the strongest things that Keystroke dynamics provides as a biometric system is that it is widely accessible by most people. Everything that uses a keyboard could be used for this biometric system, but really where it shines is on the use of internet activity such as Coursera using it to make sure it's a certain student sitting an exam. The way that Coursera provides this is by continuously taking typing samples, and pictures to make sure it is the person that started the course who is still sitting it (They no longer take pictures, and just use keystroke dynamics for it now) (Medium, 2017).

Regarding other biometric systems, it is not very unique. Many people will have similar typing techniques, but with certain other biometrics such as Iris it can be very unique to a certain individual. For this reason, any system that needs to be highly secure with confidential information would probably be best to use another biometric system, or use this in combination with another system. For example, I don't think using the company's product discussed above; TypingDNA would be a good idea for credit card transactions which it does provide (They may have a system in place that accepts a lot more false-negatives than false-positives however, so it still works well). I think it would be better to use that, and then some other biometric system/ form of authentication as well as the password.

One of the problems that is faced within keystroke dynamics is that the way users enter their passwords can vary depending on several different variables. This can include entering them more slowly at night time than in the day time, or they may be eating with one hand while typing. This can have a drastic effect on how the biometric system performs. One of the biggest issues is the difference in devices. As an example, typing a password on one laptop that are used to would be much different than typing it on a laptop that you are not used to (Biometrics, 2017).

Collecting the data for the biometric is good towards a lot of other biometrics since it can work in the background. This may be a legal issue if the system isn't making you aware of what it is doing but currently it is still a good system for collecting a lot of data since it can be collected very easy when a user is logging in to a system if it's being used as a biometric system for example. It also isn't frowned upon by certain people like some other biometric systems such as finger prints that they associate with crime. It is also only storing characteristics and no information that can link it back to a person.

It would be a hard system to try and attempt a brute force log in to since it is monitoring the rhythm of how keys are being pressed. You would have to be able to mimic how a user types the password with significant accuracy to be granted access to the users account in an authentication system. As well as this, it requires you to enter the password correctly too, so you would need to have access to that.

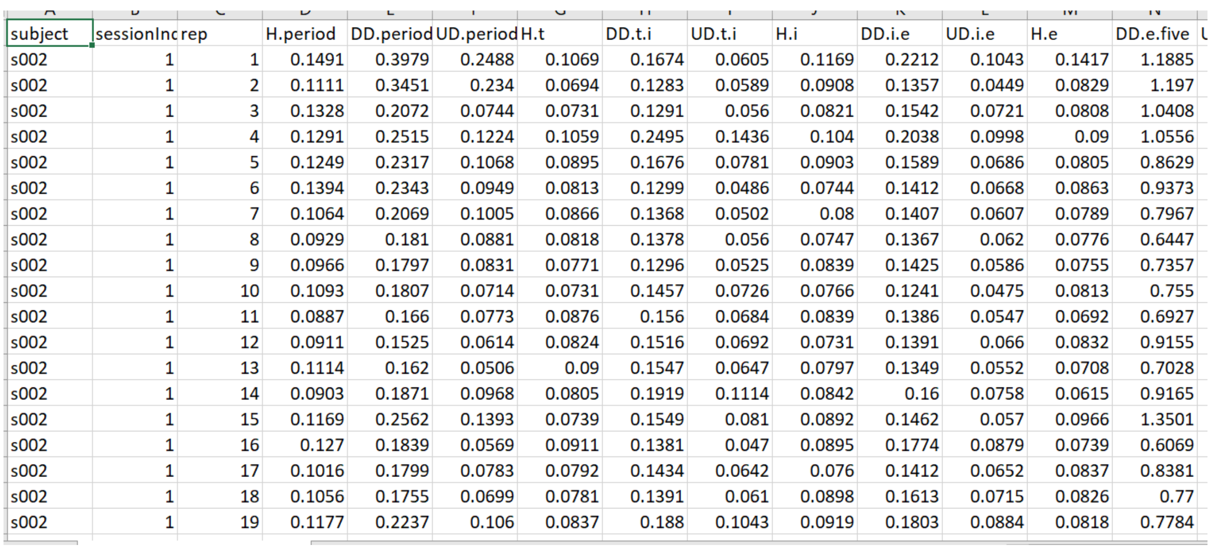
The performance is probably the biggest issue that Keystroke dynamics has. Since there's a lot of variance in how someone types something, the system has higher False Accept Rate, and False Reject Rate than a lot of biometric systems. Since this is the case, this is not a great system for systems where you need extremely secure systems.

Practical 1. Machine learning model with keystroke dynamics for equal error rate

Overview

The aim of this practical will be to show how keystroke dynamic systems work in practice. A machine learning model will be made from a dataset. I aim to test it with imposters where the aim will be that the model will be able to calculate the equal error rate for this specific system, and thus proving how accurate it could be in practice. This will use a dataset from CMU Keystroke Dynamics Benchmark Dataset (Cs.cmu.edu, 2017). The system works as follows:

- A user types a password 400 times.
- It monitors different features:
  - Hold Time – This is the time that the key is held pressed down. It will count the time from the instant each key in the password is held for.
  - Keydown-Keydown Time – This is the time that the key is pressed down until the next key is pressed.
  - Keyup-Keydown Time – This is the time that one key is released, and the next one is being pressed.



subject	sessionInc	rep	H.period	DD.period	UD.period	H.t	DD.t.i	UD.t.i	H.i	DD.i.e	UD.i.e	H.e	DD.e.five
s002	1	1	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605	0.1169	0.2212	0.1043	0.1417	1.1885
s002	1	2	0.1111	0.3451	0.234	0.0694	0.1283	0.0589	0.0908	0.1357	0.0449	0.0829	1.197
s002	1	3	0.1328	0.2072	0.0744	0.0731	0.1291	0.056	0.0821	0.1542	0.0721	0.0808	1.0408
s002	1	4	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436	0.104	0.2038	0.0998	0.09	1.0556
s002	1	5	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781	0.0903	0.1589	0.0686	0.0805	0.8629
s002	1	6	0.1394	0.2343	0.0949	0.0813	0.1299	0.0486	0.0744	0.1412	0.0668	0.0863	0.9373
s002	1	7	0.1064	0.2069	0.1005	0.0866	0.1368	0.0502	0.08	0.1407	0.0607	0.0789	0.7967
s002	1	8	0.0929	0.181	0.0881	0.0818	0.1378	0.056	0.0747	0.1367	0.062	0.0776	0.6447
s002	1	9	0.0966	0.1797	0.0831	0.0771	0.1296	0.0525	0.0839	0.1425	0.0586	0.0755	0.7357
s002	1	10	0.1093	0.1807	0.0714	0.0731	0.1457	0.0726	0.0766	0.1241	0.0475	0.0813	0.755
s002	1	11	0.0887	0.166	0.0773	0.0876	0.156	0.0684	0.0839	0.1386	0.0547	0.0692	0.6927
s002	1	12	0.0911	0.1525	0.0614	0.0824	0.1516	0.0692	0.0731	0.1391	0.066	0.0832	0.9155
s002	1	13	0.1114	0.162	0.0506	0.09	0.1547	0.0647	0.0797	0.1349	0.0552	0.0708	0.7028
s002	1	14	0.0903	0.1871	0.0968	0.0805	0.1919	0.1114	0.0842	0.16	0.0758	0.0615	0.9165
s002	1	15	0.1169	0.2562	0.1393	0.0739	0.1549	0.081	0.0892	0.1462	0.057	0.0966	1.3501
s002	1	16	0.127	0.1839	0.0569	0.0911	0.1381	0.047	0.0895	0.1774	0.0879	0.0739	0.6069
s002	1	17	0.1016	0.1799	0.0783	0.0792	0.1434	0.0642	0.076	0.1412	0.0652	0.0837	0.8381
s002	1	18	0.1056	0.1755	0.0699	0.0781	0.1391	0.061	0.0898	0.1613	0.0715	0.0826	0.77
s002	1	19	0.1177	0.2237	0.106	0.0837	0.188	0.1043	0.0919	0.1803	0.0884	0.0818	0.7784

Figure 1. This is what the dataset looks like.

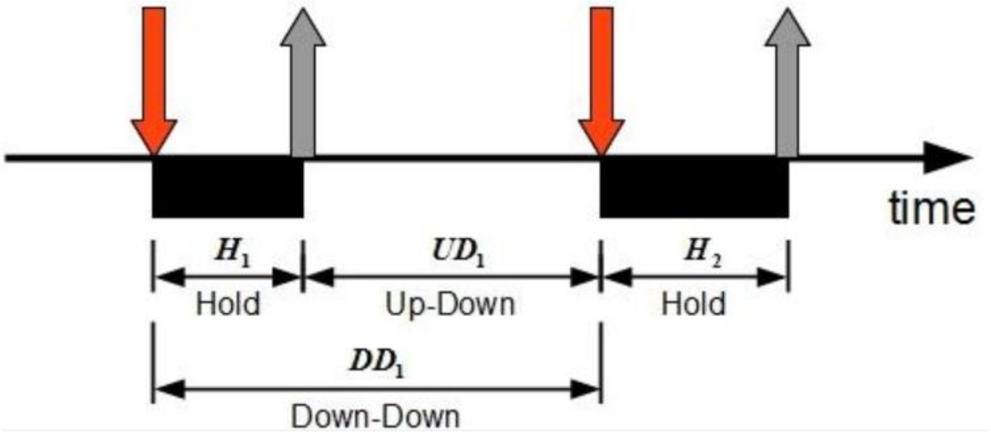


Figure 2. This picture shows the breakdown of key presses. From the dataset, this are the columns for each entry. For example "H.period" corresponds to the hold time of the '.' (period) key (bot1, 2017).

There's an entry for each of these in three features for every key in the password which is "*.tie5Roanl*".

To carry this practical out, I used a python script, and class to train a machine learning model on the data which will be talked about below. The first step in this process was to make sure that the data was in a format that the machine learning model will be able to understand. To do this I used the python library pandas (Pandas.pydata.org, 2017).

Code Snippet:

```
import pandas

import numpy as np

from sklearn import svm

from sklearn import metrics

from collections import Counter

from EER import evaluateEER


data     = pandas.read_csv("DSL-StrongPasswordData.csv")

subjects = data["subject"].unique()
```

This code starts off with just importing the code that is needed for the overall script. It reads in the data using pandas and stores it in a variable called data. Subjects is a variable that stores the different subject tags that are found in the file. E.g. s01, s02, … etc.

The next step is to create a machine learning model to train, and test against. For this, I chose to use the OneClassSVM model from scikit learn. This will use what is known as a decision function to carry out analysis on. The basic concept behind a decision function is that it takes in a dataset as input, and from this it outputs a decision from this as the output. In this case the dataset can be split into two and it will say if the data is like the training data or not. It is important to understand why OneClass can be beneficial in Machine Learning, and in relevance to this report, why it would be useful for our dataset. The benefit of using a OneClass implementation of a Machine Learning algorithm rather than Two classes is because sometimes it is not easy to collect a dataset of information that corresponds to the '*negative*' example. An example of this would be in an intrusion detection system that is designed to read network traffic on the fly, and attempt to spot anomalies in the dataset. For this, a OneClass implementation might be an extremely useful approach because it doesn't need any anomalies to be inputted into the algorithm where as with a Two Class approach this would be needed. For the dataset in question, either could be used but I chose to use the OneClass implementation as it was something I'm familiar with and because it is a good approach if you don't already have a big dataset (Rvlasveld.github.io, 2017). The basic premise of a SVM (Support Vector Machine) is that it takes it a labelled dataset, and it outputs an optimal hyperplane which is then used for categorizing new examples (Medium, 2017).

Code Snippet:

```
class OneClassSVM:
    def __init__(self, subjects):
        self.user_scores = []
        self.imposter_scores = []
        self.mean_vector = []
        self.subjects = subjects

    def training(self):
        self.clf = svm.OneClassSVM(kernel='rbf',gamma=26)
        self.clf.fit(self.train)

    def testing(self):
        self.u_scores = -self.clf.decision_function(self.test_genuine)
        self.i_scores = -self.clf.decision_function(self.test_imposter)
        self.u_scores = list(self.u_scores)
        self.i_scores = list(self.i_scores)

    def evaluate(self):
        eers = []
        for subject in subjects:
            genuine_user_data = data.loc[data.subject == subject, \
                                     "H.period":"H.Return"]
            imposter_data = data.loc[data.subject != subject, :]
            self.train = genuine_user_data[:200]
            self.test_genuine = genuine_user_data[200:]
            self.test_imposter = imposter_data.groupby("subject"). \
                                 head(5).loc[:, "H.period":"H.Return"]
            self.training()
            self.testing()
            eers.append(evaluateEER(self.u_scores, \
                                    self.i_scores))
            return np.mean(eers)
```

This is what the model will be used and the code will be discussed more in detail below for certain sections of the code. The data is broken down into different categories as can be seen in the below code:

Code Snippet

```
  eers = []

        for subject in subjects:

            genuine_user_data = data.loc[data.subject == subject, \

                                             "H.period":"H.Return"]

            imposter_data = data.loc[data.subject != subject, :]

            self.train = genuine_user_data[:200]

            self.test_genuine = genuine_user_data[200:]

            self.test_imposter = imposter_data.groupby("subject"). \

                                   head(5).loc[:, "H.period":"H.Return"]
```

What this does is, loops through every subject that is in the dataset, and then assigns several variables that are going to be used for each of them. The first is the genuine_user_data variable, this holds all the data that belongs to the subject in question. The next is the imposter_data which holds the data for everyone except for the current subject. The train variable takes the first 200 entries from the genuine_user_data and will be used to train the model. The test_genuine variable then takes an additional 200 afterwards which will be used to test the model after it has been trained. Lastly, the test_imposter holds data for each of the subjects that compiles a dataset that is just of different subject's data (Not the full data, just 5 of each).

The next thing that will happen is that the training function will be called which will be used to train the model.

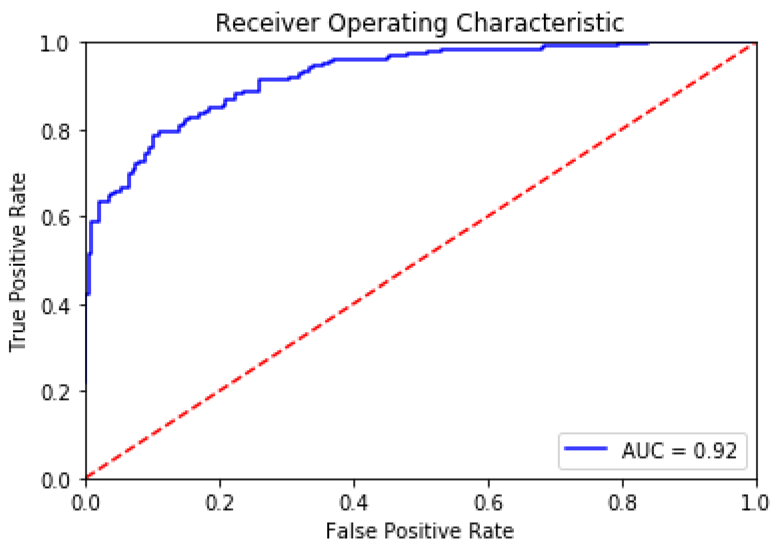
Code Snippet:

```
def training(self):

        self.clf = svm.OneClassSVM(kernel='rbf',gamma=26)

        self.clf.fit(self.train)
```

This is relatively straightforward. A object is made called '*clf*' which is the OneClassSVM with variables set as the kernel being rbf and the gamma 26 (These were the values that most success was had with), and then the variable train is used to carry out the training on the model.

Next is calling the testing function which will be responsible for making the scores for both the user and the imposter set. This is done using a decision function which was discussed earlier on and will from there be able to give an equal error rate. I won't go through the code since it's just calling a scikit learn function to create it using both the variables test_genuine, and test_imposter. After this it calls to the calculation for Equal Error Rate code which is not my own (A lot of the code for this can be found there too but is slightly edited) and can be found at the tutorial linked in the references.

The equal error rate determines the point where the False Accept Rate is equal to the False Reject Rate in a biometric system. In calculations, the lower the equal error rates value is, the higher the accuracy of the biometric system is. In this case the equal error rate for the model that was trained was 0.170437017995.



Sample roc curve from the dataset. I would like to see how it performs against other biometric systems' but I cannot find any equal error rate examples online, but I do imagine that it isn't the best performing metric going around. Next, I done some basic analysis on the scores that I got back so I added in the following code:

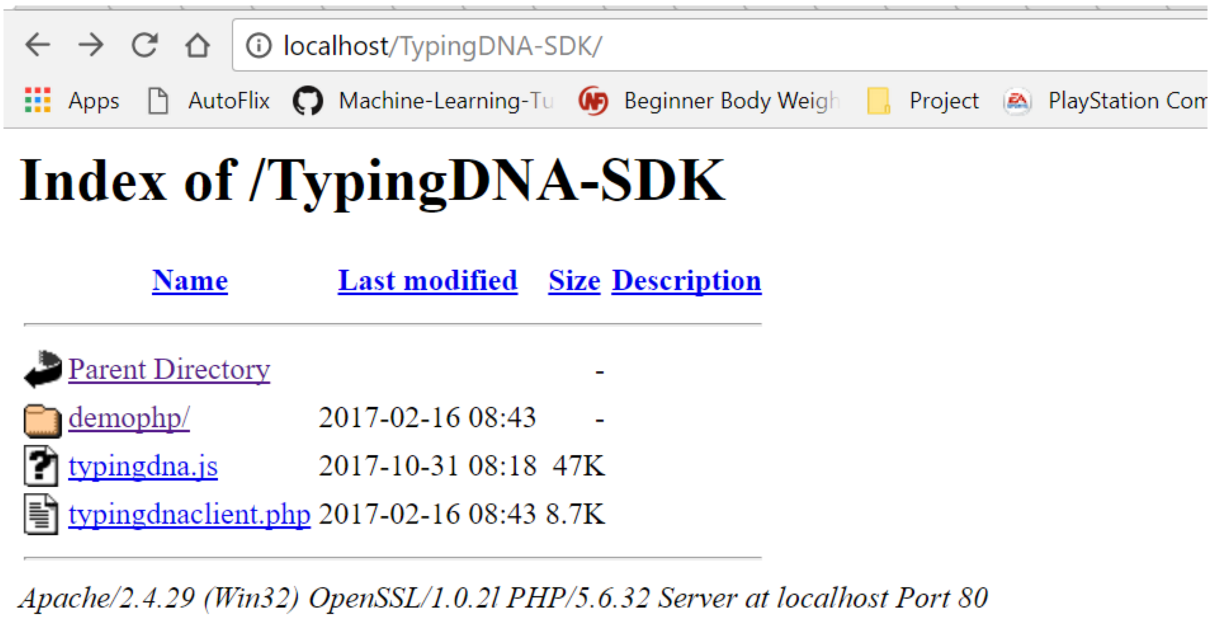
Code Snippet:

```
self.u_scores2 = [i for i in self.u_scores if i <= 2]
self.i_scores2 = [i for i in self.i_scores if i>=2]
print (len(self.u_scores))
print (len(self.u_scores2))
print (len(self.i_scores))
print (len(self.i_scores2))
```

This is another example of attempting to seeing how accurately it performs. The lower the value, the closer it fits towards a genuine score, so I have categorised it into being less than 2 to see how much of the test it got right. After that, I check the values greater than 2 in the imposter score to see how many it picks up of them. The values of the genuine were as follows: 200 in total, and the values under 2 were 158. This is about 80% accurate in its predictions which is ok but not amazing. In the imposter scored, there was 250 in total and it predicted 214 correctly. This is about 86% accurate, which is bad as it is letting people into the system that do not belong in there. I then attempted to lock down the system further, so it would only pick up values less than 1.5/greater than 1.5. The results for this for the genuine scores were 200 in total, and correctly predicted 133 which is about 67% accurate. Then, I tried with the imposter score at the same value and got back 250 in total, 234 predicted correctly which is 94% accurate. As you can see, there's a huge trade off in how this system performs and probably needs some additional functionality to improve its accuracy. However, it does showcase how this system works.

Practical 2. Using TypingDNA with a web application

For this section of the practical, I have decided to download the TypingDNA API which has a free trial of 200 free requests (Which should be fine for this practical). This is made out of PHP files alongside TypingDNA's javascript files which was provided by TypingDNA and edited to work as part of a web application by me. To run this project, I will be using XAMPP to set it up on my localhost.


localhost/TypingDNA-SDK/

Index of /TypingDNA-SDK

Name	Last modified	Size	Description
Parent Directory		-	
demophp/	2017-02-16 08:43	-	
typingdna.js	2017-10-31 08:18	47K	
typingdnaclient.php	2017-02-16 08:43	8.7K	

Apache/2.4.29 (Win32) OpenSSL/1.0.2l PHP/5.6.32 Server at localhost Port 80

Figure 1. Navigating to the project name on localhost now displays a directory of the files. Including the demo.

The next step is to go into the demo itself:

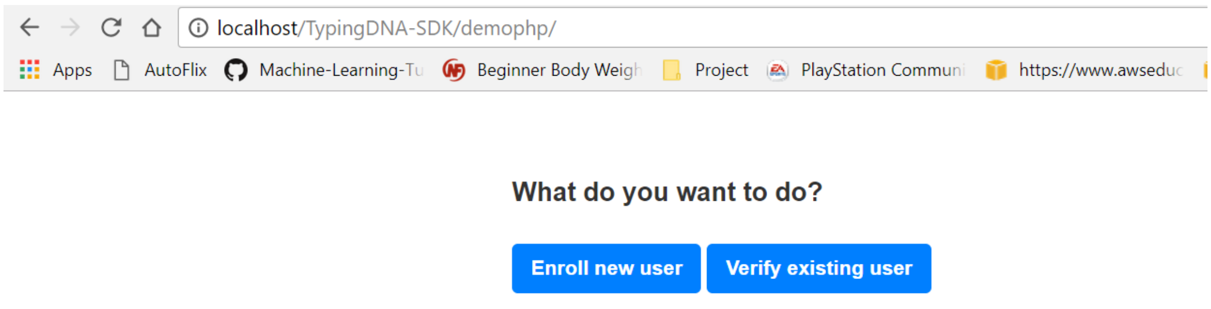
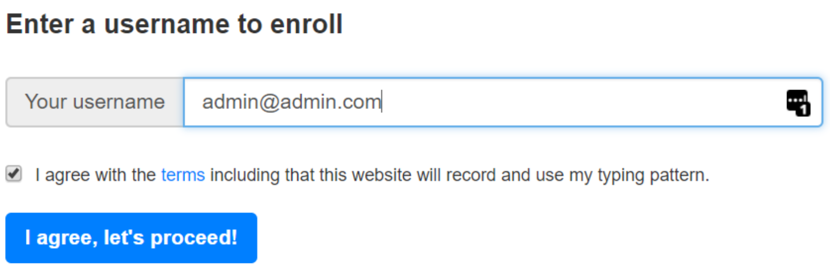


Figure 2. This is the page that is now displayed. You can either enrol a new user/verify an existing user.

First I am going to create a new user with the username admin@admin.com.

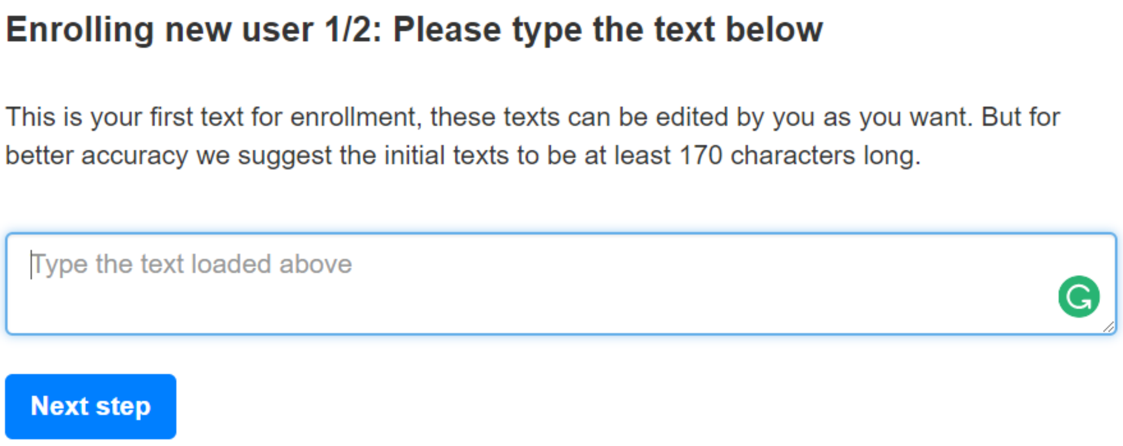

Enter a username to enroll

Your username | admin@admin.com

☑ I agree with the terms including that this website will record and use my typing pattern.

Figure 3. The username in question, now I can begin with the next step in creating the user.

The next step is to record typing systems as samples of the features that will be used to identify if the user is the correct user or not. To do this TypingDNA have supplied a sample text to be entered and recorded by their system:

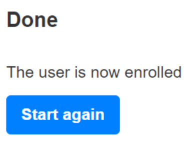

Enrolling new user 1/2: Please type the text below

This is your first text for enrollment, these texts can be edited by you as you want. But for better accuracy we suggest the initial texts to be at least 170 characters long.

Type the text loaded above

Figure 4. This is the text that needs to be entered into the system. For this demo I will be entering them myself as the admin. Note that this will be done in how I would normally type. There is 2 provided texts, the second one will not be shown below.

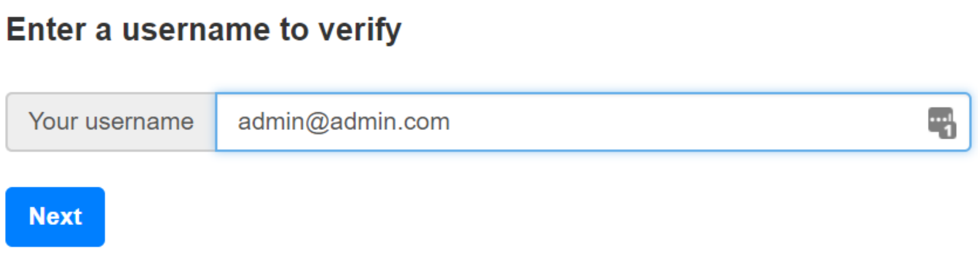
After entering the passwords onto the system, a loading screen is displayed which make take a while to load, and will display the following:


Done

The user is now enrolled

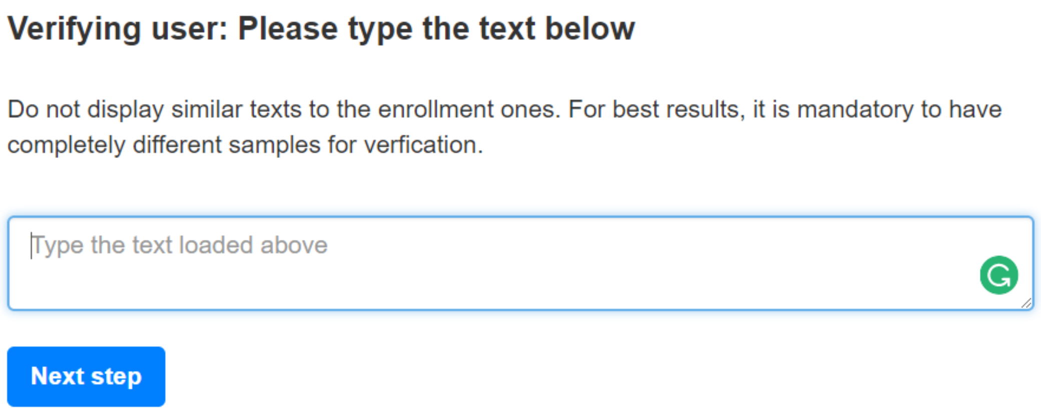
Figure 5. The user admin@admin.com is now made.

The next step is to attempt to verify the user:


Enter a username to verify

Your username: admin@admin.com

Figure 6. The username is provided and next it will ask you to type in a sample text for the username to verify if it is you.


Verifying user: Please type the text below

Do not display similar texts to the enrollment ones. For best results, it is mandatory to have completely different samples for verfication.

Type the text loaded above

Figure 7. This is the what the user must do on this app for verification.

I entered the text in as I would normally type to see what would happen, and it successfully logged in:

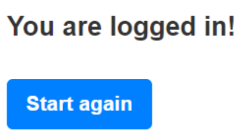

You are logged in!

Figure 8. Successful login

After this I got three different people to log in using the same text to see what the results would be. In each of the cases, it successfully prevented them from logging in:

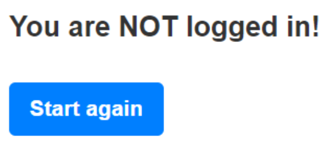

You are NOT logged in!

Figure 9. The result of what happens when someone other than me typed in the text.

Conclusion

From my research into keystroke dynamics, I found out a lot about how the system works and what it is useful for. One of the things that keystroke dynamics has over a lot of other biometrics is that it can work in the background of things like authentication. Most biometrics need to take a password, and then the biometric itself, for example a password and a fingerprint. However, with keystroke dynamic systems, you only need to enter the password and it will be tracked in the background. This makes it a very easy, low cost biometric to implement on a system. However, it isn't the best biometric to use accuracy wise, and can get high false accept rates, and high reject rates within systems. For this reason, I think keystroke dynamic systems must be tested rigorously with lots of data, and I personally think they shouldn't be used for things that you want to maintain top security for. I think the area where it excels is in user login forms for websites where a user would also need to know the password. Therefore, it can be used for things like checking the typing of someone entering their credit card information since the attacker would also need the card information.

I think this biometric has a place in modern day biometric systems, but will never be used for many of the other areas of biometrics such as surveillance or anything where typing isn't possible. I learned a lot from doing the practical work about how to implement these solutions in my own systems, and how quickly you can get '*good*' results for it.

Bibliography

Websites

Cs.cmu.edu. (2017). ***Keystroke Dynamics - Benchmark Data Set*.** [online] Available at: http://www.cs.cmu.edu/~keystroke/ [Accessed 1 Dec. 2017].

Biometrics. (2017). ***Keystroke Dynamics*.** [online] Available at: http://www.biometric-solutions.com/keystroke-dynamics.html [Accessed 3 Dec. 2017].

SearchSecurity. (2017). ***What is keystroke dynamics? - Definition from WhatIs.com*.** [online] Available at: http://searchsecurity.techtarget.com/definition/keystroke-dynamics [Accessed 3 Dec. 2017].

eWEEK. (2017). ***What Is Password Hardening and How Does It Work?*.** [online] Available at: http://www.eweek.com/news/what-is-password-hardening-and-how-does-it-work [Accessed 3 Dec. 2017].

SRL, T. (2017). ***TypingDNA - Behavioral Biometrics Authentication, Typing Biometrics, Keystroke Dynamics*.** [online] Typingdna.com. Available at: https://typingdna.com [Accessed 3 Dec. 2017].

Cs.cmu.edu. (2017). ***Keystroke Dynamics - Benchmark Data Set*.** [online] Available at: http://www.cs.cmu.edu/~keystroke/ [Accessed 1 Dec. 2017].

Behaviosec. (2017). ***Continuous Authentication through passive Behavioral Biometrics - BehavioSec*.** [online] Available at: https://www.behaviosec.com [Accessed 3 Dec. 2017].

Pandas.pydata.org. (2017). ***Python Data Analysis Library — pandas: Python Data Analysis Library*.** [online] Available at: https://pandas.pydata.org/ [Accessed 3 Dec. 2017].

Rvlasveld.github.io. (2017**). *Introduction to one-class Support Vector Machines - Roemer's blog*.** [online] Available at: http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/ [Accessed 3 Dec. 2017].

Medium. (2017). ***Chapter 2 : SVM (Support Vector Machine) — Theory – Machine Learning 101 – Medium*.** [online] Available at: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72 [Accessed 3 Dec. 2017].

Images

bot1, V. (2017). ***User Verification based on Keystroke Dynamics: Python code*. [online] Machine Learning in Action.** Available at: https://appliedmachinelearning.wordpress.com/2017/07/26/user-verification-based-on-keystroke-dynamics-python-code/ [Accessed 1 Dec. 2017].

Medium. (2017). ***Coursera and keystroke biometrics – PersonalData.IO – Medium*.** [online] Available at: https://medium.com/personaldata-io/coursera-and-keystroke-biometrics-550762f2f61b [Accessed 3 Dec. 2017].