Authentication Techniques by Typing Characteristics
TJHSST Senior Research Project Research Paper
Computer Systems Lab 2009-2010

Luke Knepper
luke@lukeknepper.com

April 6, 2010

Abstract

This project will examine various applications of authenticating users by use of their typing characteristics. The purpose of this project is to determine the effectiveness of analyzing the user's typing patterns in order to ensure user security. Log-in-time authentication and continuous authentication based on the user's typing characteristics will both be explored, analyzed, and tested. One application will analyze the user's typing characteristics when the user attempts to gain access to, or log-in to, a system. Another application will measure the user's keystroke data while the user uses the program, and then feed the typing data through a neural network to determine authentication status. A program simulating this method in use was created, and tests were run to analyze the accuracy of using a neural network algorithm to authenticate the user. This process will be beneficial to user security because it will ensure that an intruder does not gain wrongful access to a user's account while that user is logged in to the system.

Keywords: authentication, security, typing patterns, neural networks

1 Background

1.1 Introduction

Current authentication techniques span all three tiers of security:

- Tier 1 - Identification (usernames)
- Tier 2 - Knowledge and Possession (passwords, ID cards, security questions, etc.)
- Tier 3 - Skills and Capabilities (captchas, voice recognition)

In the online world, usernames can be stolen, passwords and security questions can be guessed, and captchas can be cracked. However, authentication using a user's typing characteristics, a tier 3 security method, does not have these weaknesses and further has numerous advantages over the other approaches. Although authentication through analysis of typing characteristics has been previously proposed, it is noted that previous approaches have been restricted to the analysis of the user's typing patterns when the user is typing a simple, static word, such as U.S. Pat. No. 6,151,593 to Cho, et al (2000). These approaches have a significant weakness, because someone with malicious intent could create a program to record the characteristics of the user's keystrokes when typing this word and then simulate that process to gain access to the user's account. The approach presented herein addresses that weakness by dynamically generating textual content for the user to type and then analyzing the typing characteristics when the user enters that dynamically generated content into the system, thus creating a biometric for that user which is compared to a previously captured biometric for that user. The comparison is performed using a neural network which is trained using the previously-measured typing characteristics. The results of the comparison are compared to a predetermined threshold to determine if the user will be granted access to the system. This approach makes the authentication algorithm considerably more secure since the text to be typed is dynamically generated by the system and more advanced typing characteristics are used.

The authentication will be primarily done using neural network methods. Neural networks are based off of the way that the human brain works: they are made up of a bunch of nodes (like the brain's neurons) which have inputs, weights, and outputs (see FIG. 7 below). Each node collects its inputs, performs a simple calculation on them (such as a sum), multiplies the total

by its weight, and then outputs the final total to the next node(s). The network starts by an input vector (in this case the typing data) which gets sent to the first level (or hidden layer) of nodes and passed down throughout the rest of the layers. The final layer is a single node which outputs a value between 0 and 1, with 1 being a success (or allowed access) and 0 being a failure (or a rejection). A predetermined threshold determines how the final value should be treated (i.e. if the value is above the threshold, it's treated as a 1, and vice versa).

There is currently a patent (US 6151593) for an authentication scheme by typing pattern analysis. This method reads in the time between keystrokes for a user when typing their password and then trains a three-layered neural network to this combination. It does not allow for dynamically-generated content to be used, and does not test the different lengths of passwords. This is the most basic application of typing techniques for authentication, and this patent application extends beyond these simple methods. Further, multiple typing pattern log-in software packages exist, such as Psylock, but they all have the same weakness as the patent above: they rely on a static password instead of dynamic generated content.

Another team working under L. Maisuria compared the accuracy of neural networks compared to cluster algorithms. A multi-layered perceptron-based neural network which learned on the Hebbian learning theory was used, as were ten different metrics to compare the clusters for the clustering. They tested the different algorithms by recruiting twenty volunteers to participate in three different sittings. In the first sitting, they all trained their neural networks by typing in their password sixty times. In the next sittings, they attempted to log in to their accounts and break into the accounts of others. The sittings were spaced out by one week. The study found that the clustering methods were slightly more accurate than the neural networks in rejecting impostors. They only found an average of 80% to 90% accuracy in rejection rate, not enough to comprise a stand-alone security system but certainly good enough to be used in conjunction with traditional methods. They found that all keystrokes should be measured, including the beginning and end strokes to the enter key, for the highest accuracy. They found that allowing impostors to observe the users typing before attempting to break in to their accounts had little effect on the accuracy.

An independent team of researchers, headed by Peacock et al., tested the effect of many variations, including neural network set-up, password length, acceptance stringency, data used, and function used. They found the most

effective neural network structure from their tests was to use a set-up where many independent neural networks are trained on different cores (i.e. parallel processing) using randomly generated starting weight vectors. During the training, the best weight vectors are picked and created using genetic algorithms. They found the smaller (more stringent) acceptance ranges came up with a good amount of false alarms (when it didn't let the correct user in, happened 22% of the time) but also minimized break-ins (when the incorrect user was let in, happened 3% of the time). They also found the most effective password length was 7 characters, a mid-sized password (the longer passwords had no break-ins but many false alarms, and the shorter passwords had many break-ins). They concluded that a linear evaluation function was more effective than a quadratic function and that averaging was more effective than counting each training run. They suggest their results can be improved (75% success, 22% false alarm and 3% break-in for their best algorithm).

The main advantage of the approach described herein over previous approaches is the capability to authenticate a user based on a dynamically generated chunk of text that the user is required to type in.

2 Drawings

FIG. 1 is a mockup of an account set-up screen for a computational system.

FIG. 2 is a flowchart for training the authentication system.

FIG. 3 is a flowchart of the continuous authentication algorithm.

FIG. 4 is the GUI of the proof-of-concept program

FIG. 5 is the GUI of the data collection applet

FIG. 6 is the GUI of the continuous authentication simulation program

FIG. 1: STAGE 1

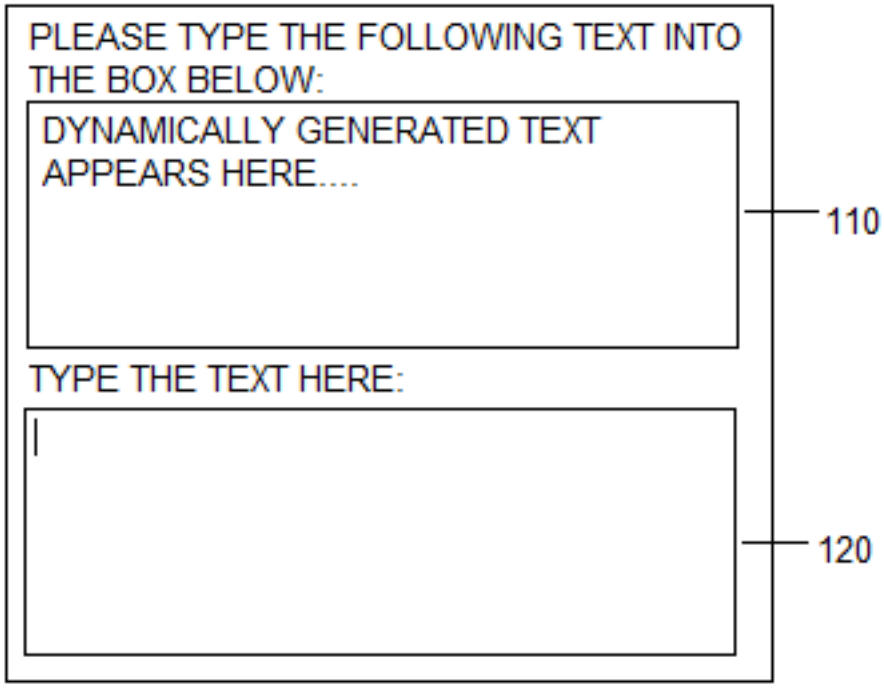

PLEASE TYPE THE FOLLOWING TEXT INTO THE BOX BELOW:
DYNAMICALLY GENERATED TEXT APPEARS HERE....
110
TYPE THE TEXT HERE:
120

FIG. 2: TRAINING ALGORITHM FLOWCHART

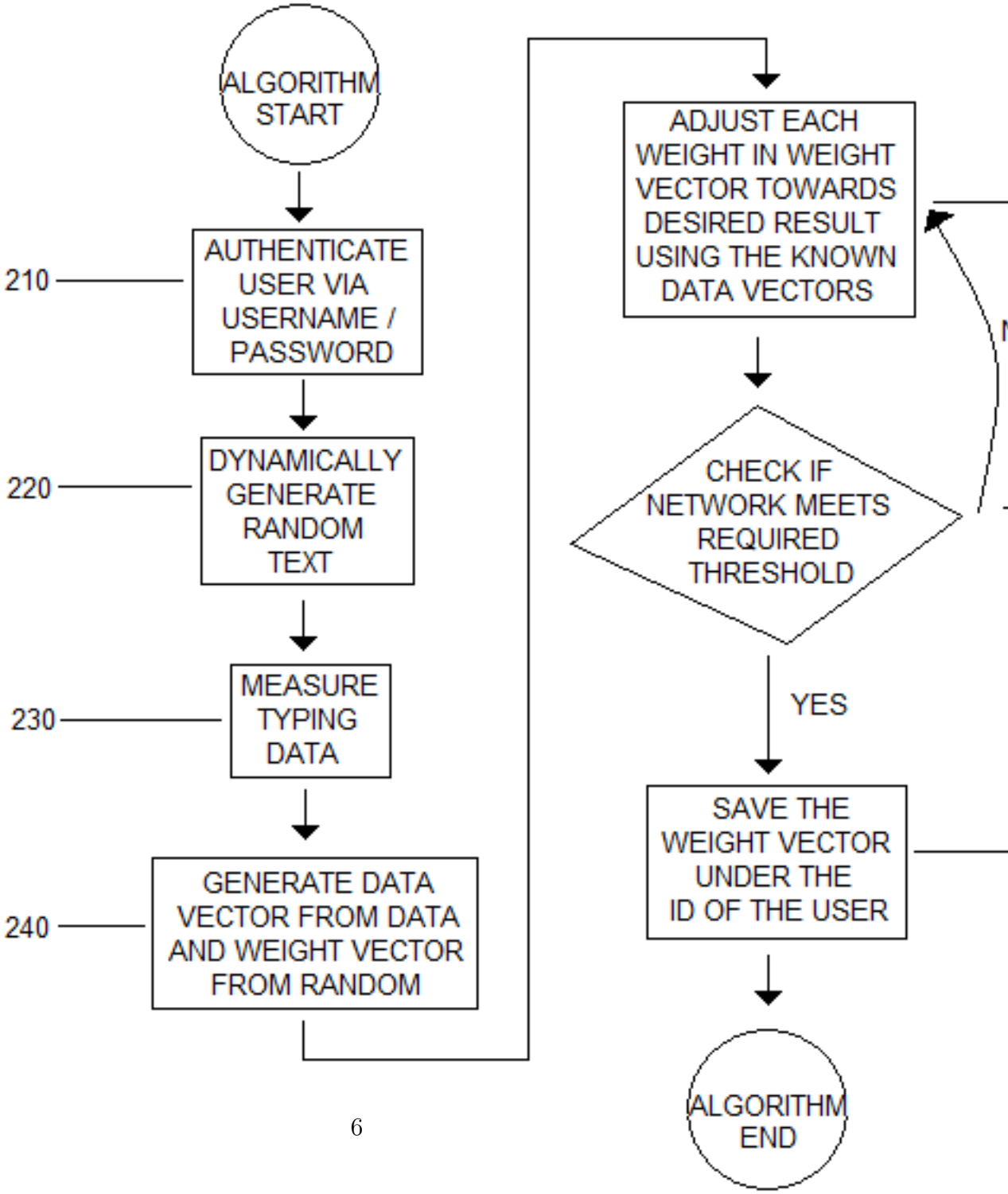


FIG. 3 : AUTHENTICATION ALGORITHM FLOWCHART

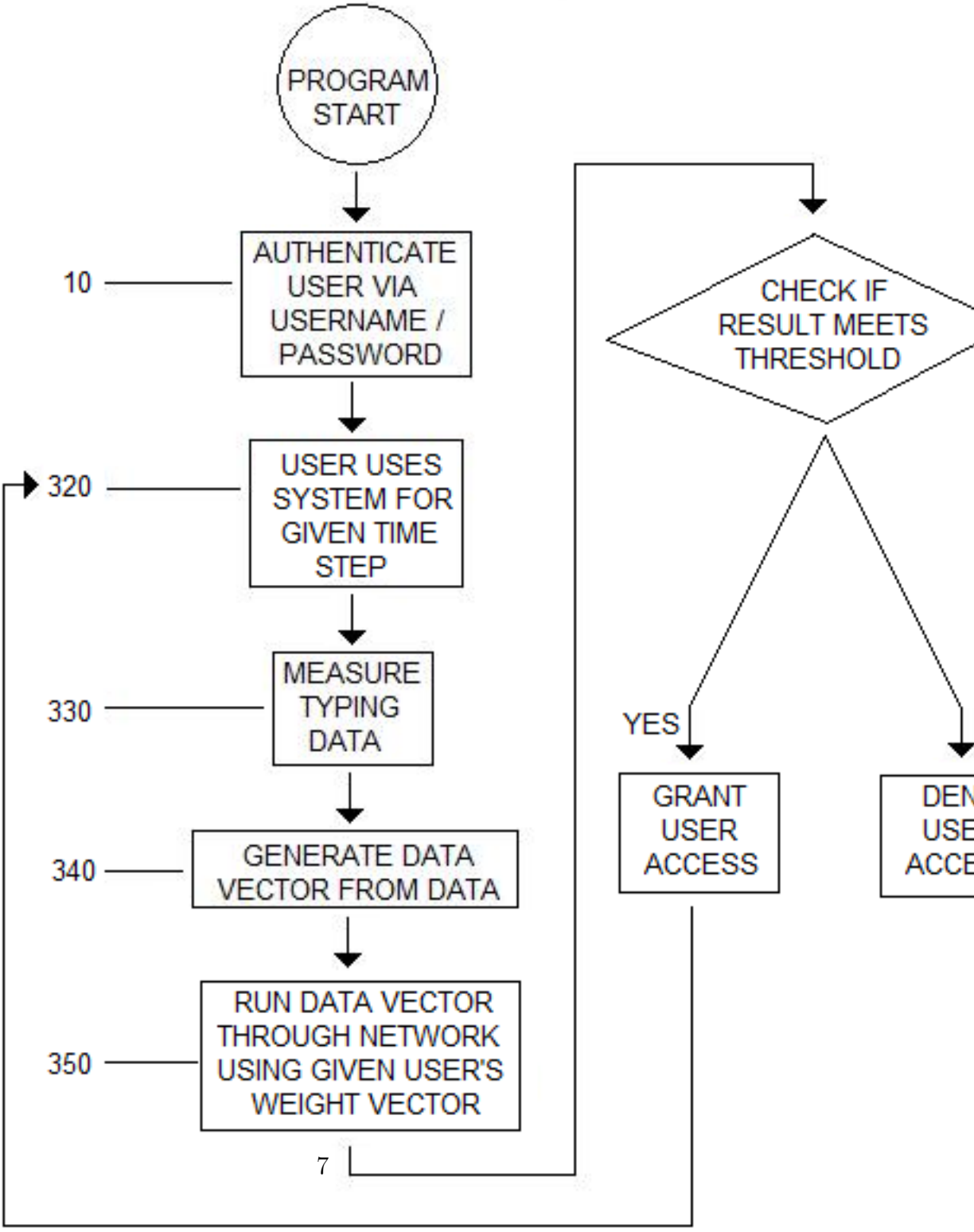


FIG. 4:

FIG. 5:

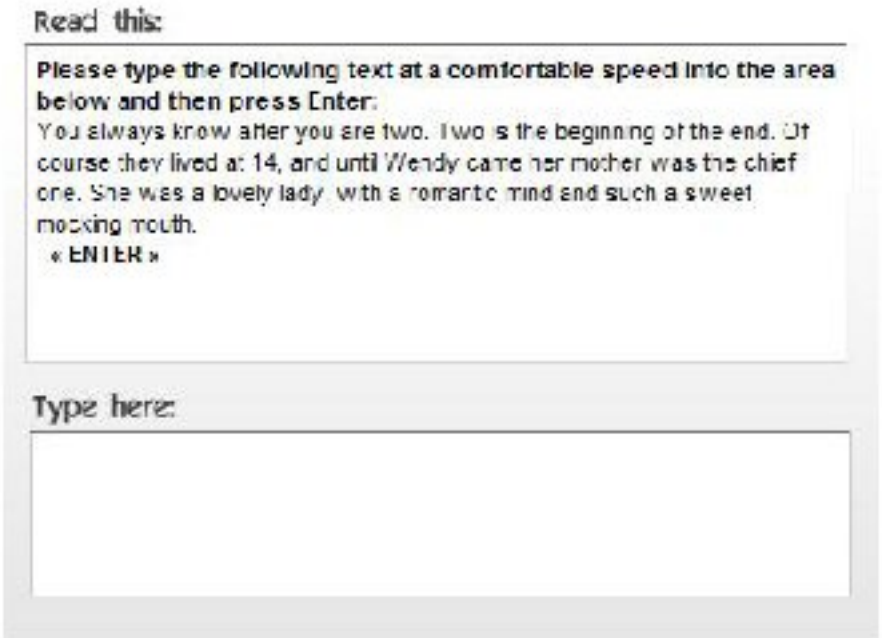
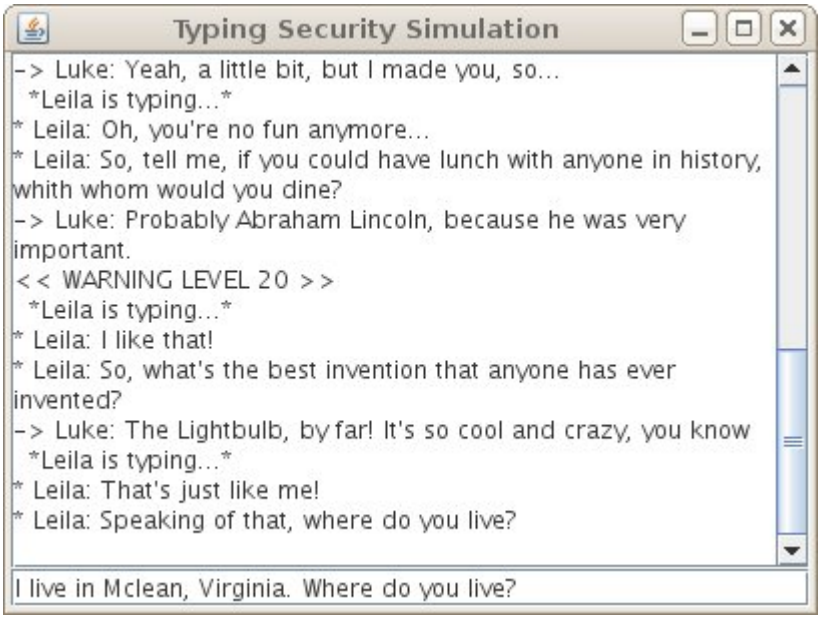


FIG. 6:



3 Procedure

FIG. 1 shows an example GUI for the initial typing characteristics measurement stage. A block of text is dynamically generated and displayed to the user (110). The user is prompted to type the displayed text into a text field (120). While the user is typing, the system records the time the user presses and releases each key. Once the user has completed typing the dynamically generated text, the user?s typing information is then passed on to the neural network algorithm described in FIG. 2.

FIG. 2 shows the process through which the neural network is trained for each new user. This algorithm executes when the user is creating a new account. The algorithm generates a data vector, which is a vector representing the users typing characteristics, from the typing data, and a weight vector, which is used by the neural network, from random values (240). The data vector contains information that is vital about the user?s typing characteristics, including but not limited to time of depression of each key and time elapsed between each keystroke. The weight vector contains values which represent the weights of each node in the neural network. The neural network is made up of multiple layers of nodes which each have given inputs, weights, and outputs. The first layer of the network contains a node for every

element in the data vector. The last layer of the element contains only one node, which outputs the final result. Hidden nodes in the middle layers provide an intermediary between the first and last layers. Each node takes its given inputs from the previous layer (or from the data vector, as is the case with the first layer), performs a mathematical function on this data using the nodes values and weights, and then outputs the final result to the next nodes (or as the final output of the program, as is the case with the last layer). If the output is above the threshold for acceptable values, it will be treated as an acceptable output for authentication, and if it is not then it will not be sufficient for authentication. The neural network must be trained when the user first creates an account (5 and 6). The program runs data vectors to which it already knows the final result (e.g. the data vector generated from the user?s typing, which has a desired output value of 1 (or success), and data vectors stored in the database generated from other users? typing, which have desired value 0 (or failure)) through the network, and adjusts the weights to achieve the desired result until the network returns the optimal result (i.e. changing the weight vector does not improve the results to a measurable extent). If the value returned with the new user?s data vector does not meet the threshold for acceptable values, the training process is repeated with a new set of randomized weight vectors. Once an optimal weight vector is created, it is stored in the database under that user.

FIG. 3 describes the authentication algorithm which takes to authenticate the user. The algorithm is the same whether the system is continuously monitoring typing patterns during program use or using a one-time measurement of typing patterns at log-in time, with the only difference being the log-in time algorithm generates text for the user to type at step 320. The user is first authenticated via their username and password with which they created their account on this system (310). Then the program measures the user's typing patterns (320), either after a set amount of time for continuous authentication or after prompting the user to type dynamically generated text for log-in authentication. The program measures the user's typing characteristics by recording the times when keys on the keyboard are depressed or released. This data is then used to compute statistics which describe the user's typing characteristics, e.g. the user depresses the 'A' key for a measured average of 80 milliseconds, or the user takes a measured average of 200 milliseconds between releasing the 'A' key and pressing the 'B' key. The algorithm generates a data vector, which is a vector composed of the aforementioned statistics and representing the users typing characteris-

tics, from the typing data (340). The data vector is then run through the neural network (350) to return a final output between 0 and 1. If the output meets the threshold for acceptable values for authentication (360), then the user will be granted access to the system; otherwise, the user will be denied access.

4 Results

4.1 Neural Network Coding

Java classes which encompass the functionality of different types of neural networks are being coded. At present, a simple single-layer neural network has been completed and runs without error, and is currently being tested for accuracy. A flexible multi-layer neural network is near completion, and is currently in the de-bugging phase. The advantage of programming the neural networks in this fashion is that they can easily be swapped between programs, or during a program, for testing.

4.2 Proof of Concept

A simple proof-of-concept was completed in October '09. The program prompts two users to both type a sentence and uses their data to train a simple single-layer neural network (shown in FIG 5). It then prompts the users with a third sentence and instructs one of them, whose identity is unknown to the computer, to type the third sentence. It runs the final data through the trained neural network and determines which user typed the third sentence.

The proof-of-concept program has been tested twenty times, with the results shown below:

- Trial 1 – Correct
- Trial 2 – Correct
- Trial 3 – Correct
- Trial 4 – Correct
- Trial 5 – Correct

- Trial 6 – Correct
- Trial 7 – Correct
- Trial 8 – Incorrect
- Trial 9 – Correct
- Trial 10 – Correct
- Trial 11 – Correct
- Trial 12 – Correct
- Trial 13 – Correct
- Trial 14 – Correct
- Trial 15 – Correct
- Trial 16 – Incorrect
- Trial 17 – Correct
- Trial 18 – Correct
- Trial 19 – Correct
- Trial 29 – Correct

There were 18 correct runs, 2 incorrect runs out of 20 total runs, for a 90% accuracy overall. This shows that the concept can be used and refined to create an accurate authentication system, however it supports the idea that it cannot be a standalone system but instead will have to be used hand in hand with traditional authentication methods, such as passwords and usernames. The simple structure of the neural network leaves much to be desired.

4.3 Data Collection

A data collection applet (FIG 5) has been completed and posted on the internet at http://www.lukeknepper.com/research. It has been well publicized via the use of social networks and viral marketing, and has collected in excess of 1,500 data samples, well more than are needed to run tests of neural network accuracy. This data collection applet gave the user a segment of random text (which was randomly selected from a large document of precomposed text, rather than being dynamically generated) and prompted the user to type in this text to a textbox below. It recorded the users keystrokes each time a key event was fired by saving the key's number, key direction, and keystroke time (in milliseconds). This data was sent to a server and saved via a simple PHP script.

4.4 Continuous Authentication Simulation

A program (FIG 6) has been nearly completed to simulate the continuous authentication process in action. This program is a simulation of an instant messenging program, where the user interacts with a pre-programmed bot which randomly asks the user questions to which the user is supposed to type answers. The program measures the user's typing characteristics upon the user's first response, when the user is prompted to type a simple sentence containing every letter of the alphabet, and trains a neural network to this typing data. The program then measures the user's typing characteristics for every subsequent response and runs these data through the created neural network. If the neural network does not output a sufficient value (i.e. one that is greater than the standardized output) then the warning level is raised. Once the warning level reaches the 100% threshold, the system shuts down and locks the user out of the system. The warning level is lowered with every response which matches the original typing characteristics.

This program currently runs without error and has been unofficially tested by casual tests with other students in the Computer Systems Lab. It still has a few areas which need to be refined before more intensive testing can be performed on the simulation's performance.

References

[1] Cho, S. and Han, D. "Apparatus for authenticating an individual based on a typing pattern by using a neural network system." http://www.freepatentsonline.com/6151593.html

[2] Maisuria, L. "A COMPARISON OF ARTIFICIAL NEURAL NETWORKS AND CLUSTER ANALYSIS FOR TYPING BIOMETRICS AUTHENTICATION."

[3] Nallusamy, R. and Dr. Duraiswamy, K. "Neural networks for dynamic shortest path routing problems-A survey." 2008.

[4] De Oliveira Paula, Marcus V.S. et. al. "User Authentication based on Human Typing Pattern with Artificial Neural Networks and Support Vector Machine "

[5] Ponnath, Abhilash. "Instantaneously Trained Neural Networks."

[6] Peacock, A. et al. "Typing Patterns: A Key to User Identification." http://www2.computer.org/portal/web/csdl/doi/10.1109/MSP.2004.89

[7] Stomski, Paul J. Jr. and Adel S. Elmaghraby. "SELECTION OF A NEURAL NETWORK FOR VISUAL INSPECTION."