Keystrokes Timing Analysis and Timing Attacks System on Secure Shell: Instance Based Learning (IBL) Model Approach Revisited.

Ajayi, E.A., Alese, B.K. & Dahunsi, F.M.
Computer Science Department
Federal University of Technology
Akure Ondo State, Nigeria
Email: ebeconsult@myself.com, bkalese@futa.edu.ng,fmdahunsi@futa.edu.ng

Ajayi, B.F.
Computer Science Department
Kebbi State Polytechnic
Dakingari, Kebbi State

ABSTRACT

The paper presents a sophisticated approach to Timing Analysis of Keystrokes and Timing Attacks on Secure Shell called Instance Based Learning (IBL) model. Secure Shell (SSH) is a protocol for securing remote access across an insecure network. SSH has well established encryption and authentication mechanisms but it has two major weaknesses: firstly, the approximate size of the data is revealed because transmitted packets are padded only to an eight-byte boundary (if a block cipher is in use). Secondly, when it is in an interactive mode, every individual keystroke a user types is sent to a remote machine in a separate IP packet immediately after the key is pressed**.** This leaks the inter-keystroke timing information of users when typing. This paper shows how these seemingly minor weaknesses result in serious security risks and therefore presents the implementation of an Instance Based Learning Model Timing Attacks System against Secure Shell protocol. The result shows that there is a need for software engineers to do more work in the design of secure protocol to provide safety to users in cashless economy.

Keyword: SSH, Instance Based Learning, Timing Attacks, protocol, Web Application Server

Aims Research Journal Reference Format:
Ajayi, E.A., Alese, B.K., Dahunsi, F.M. & Ajayi, B.F. (2016): Keystrokes Timing Analysis and Timing Attacks System on Secure Shell: Instance Based Learning (IBL) Model Approach Revisited. Advances in Multidisciplinary Research Journal. Vol 2, No.2 Pp 135-146.

I. INTRODUCTION

As Internet access becomes increasingly inexpensive and available, it has become a viable replacement for traditional couriers, telephone, and fax, as well as remote dial-up access to an enterprise's internal computer resources. One of the greatest challenges in using the Internet technology to replace more traditional communication methods is security [8, 22, 23]. Insecurity of Internet Technology has led my developers to tries several methods and approaches to see that their clients' information is secure and safe. Just a few years ago, people commonly used astonishingly insecure networking applications such as telnet, rlogin, or ftp, which simply passed all confidential information (including users' passwords) in the clear form over the network [8].

This situation was aggravated through broadcast-based networks that were commonly used (e.g., Ethernet) which allowed a malicious user to eavesdrop on the network and collect all communicated information. Fortunately, many users and system administrators have become conscious of this issue and have taken countermeasures. To curb eavesdroppers, security researchers have designed the Secure Shell (SSH), which offers an encrypted channel between the two hosts and strong authentication of both the remote host and the user, yet it is not secure as it supposed to be and our work emphases this through our new approach.

Information transferred using SSH is encrypted using strong encryption algorithms, providing strong protection from eavesdroppers on the network. Secure Shell provides three main capabilities which provides for many creative secure solutions and these are Secure Command-shell, Secure File transfer, Port Forwarding. There are two type of SSH namely SSH-1 and SSH2. There are three classes of Attack on SSH namely:

a. Cryptographic/ Protocol Attack: This is a method for circumventing the security of a cryptographic system by finding a weakness in a code, cipher, cryptographic protocol or key management scheme[3,4,5,22,23]. A well-known example of this attack is the insertion attack against SSH-1.
b. Implementation Attack: Implementation attack refers to a type of cryptanalysis attack that does not target cryptographic algorithms and protocols directly. This attack aims at implementation of cryptographic systems (e.g. smart cards, USB tokens) to gain knowledge about secret information. These attacks refer only to certain implementations of the SSH- Server or the SSH-Client. Since the number of SSH implementations is usually high, the attacks appear very frequently [22, 23].
c. Side Channel Attacks: Side channel attacks are based on information that is gained from the physical implementation of a cryptosystem [6, 22, 23] rather than brute force or theoretical weakness in the algorithms. Timing analysis, acoustic analysis and power consumption analysis are some instances that belong to this class.

2. RELATED WORK

Due to the importance of Timing Analysis of Keystrokes and Timing attacks in security system, so many researches have been carried out on it in different direction. Firstly Dictionary Attacks Using Keyboard Acoustic Emanation [10] shows the study of signals emanating from electronic or mechanical devices has been a thing of many cryptosystem researchers. The paper looked at how information and other related properties from the sound signals from electronic and mechanical devices can be extracted. The researcher(s) develop software that crack acoustic-based password. The research work combines signal processing and efficient data structures and algorithms to successfully reconstruct single words of 7-13 Characters from a recording of the clicks make when typing them on a keyboard. Simple cross-correlation primitive model was used to develop the attack model. The cross-correlation function operates on signals in their time-domain representation, and is computed as follows: given two digitized signals x [.] and y [.], let

$$\text{CC[x, y, t]} = \sum_k x[k].y[t+k] \qquad (1)$$

This is, in effect, a sliding dot-product of the two signals, where signal *y* is shifted by *t* samples over *x*. The cross correlation is maximized at the offset *t* where the two signals are most similar. The research work define *simxcorr*(*x, y*) = *maxt* (*CC*(*x, y, t*)) to be the similarity measure between two given signals.

In [1], the researchers discovered that despite the state-of-the art authentication and encryption algorithms used in SSH, it still leaks information in two major ways. Firstly, the approximate size of the data is revealed because transmitted packets are padded only to an eight-byte boundary (if a block cipher is in use). Secondly, when in interactive mode, every individual keystroke a user types is sent to a remote machine in a separate IP packet immediately after the key is pressed. This leaks the inter-keystroke timing information of users when typing. The work developed an attack system called Herbivore which learns users' password by monitoring SSH Session. Gaussian Modeling, Hidden Markov Model and an *n*- Viterbi algorithm were used to describe how key sequences can be inferred.

In [17], it is generally believed that timing attacks cannot be used to attack general purpose server and Implementation of RSA is not vulnerable to timing attacks. The weaknesses discovered from remote network and other security problems arising from using remote network is a key motivator to the researchers on this research work. The research work x-rays how to extract private keys from an OpenSSL-based web server running on a machine in the local network and also develop a software system to extract private keys from an OpenSSL-based web server. A client attack system was developed to measure the times an OpenSSL server take to respond to decryption queries. The research work had various limitations such as: the network can only be used in their campus only which prevents the system from being used on wide area network, Attacks were mounted between two processes running on the same machine (Interprocesses), and the system can be monitored by Virtual Machine. In [18], the paper looks at ever-present threat of severe network disruption and present how to infer the infection evolution from the history of worm scans seen at a network telescope, analyze and design effective strategies for discovering the sequence with which a worm infected its victims.

$$n_i = n_i + (v - n_{i-1})[1 - (1 - \frac{1}{2^{32}})^R] \qquad (2)$$

where *R* is the total number of scans sent by the *ni*–1 infected hosts.

Mechanism for identifying the hit-list using the inter-arrivals of new Witty infected sources was developed. With timing attacks against trusted path in [19], a system that gains information about other users' using CPU time was developed. The researcher made use of multi-user workstation using Bayesian model to determine gain information about other users' passwords using CPU timings. The system can only be used on Linux platform. This paper analyzed some of the methods used by the various researchers and come up with a sophisticated model called Instance Based Learning Model that improve the timing analysis of keystrokes and timing attacks on secure shell compare to the results obtained from using Hidden Markov Model by other researcher. The Instance Based Learning Model enables us detect when a password is typed, how long the password is, reveals the password, identify such user(s) with their typing pattern.

3. INSTANCE BASED LEARNING (IBL) MODEL TIMING ATTACK SYSTEM ARCHITECTURE

This section look at the architectural design of the timing analysis of keystrokes within the SSH protocol using Instance Based Learning (IBL) Model as presented in [1, 22, 23].

A. Timing Attacks System Model

Some assumptions used throughout this work are defined as follow:

- The considered users may/may not be familiar with keyboard typing.
- The statistics are based on random text with an average length of 2000 characters, which is chosen by the test person itself with no restriction to what he/she typed. It does not matter whether the typed text by the test persons is meaningful or not and it can be repeated as many as possible which is against the assumption presented in [1, 22, 23].
- The network latency may be bigger than the inter keystroke timings due to problem with transmission medium itself or propagation time on SSH.
- The network latency is not constant. It can either be low or high.
- The adversary can eavesdrop the users' encrypted communication

B. Realization of the Attack

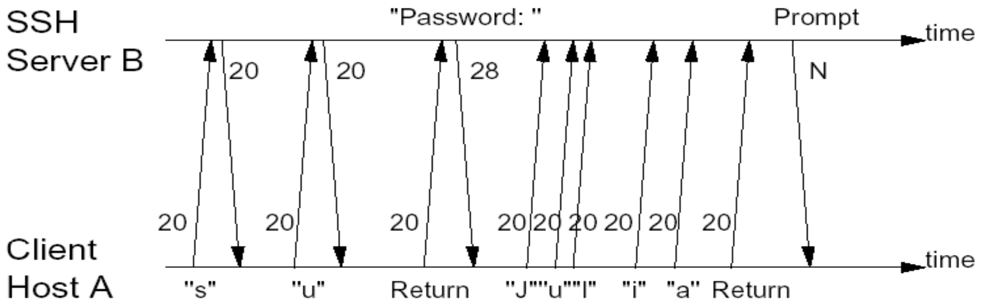


Figure 1: Schematic Illustration of a sniffing scenario [1, 2, 22, 23]

Figure1 explains a sniffing scenario in secure shell environment given a SSH-user and an adversary. The adversary is successful when she reveals the users password because at the client side, the link is not error free and recreating packet is not in place thus keystrokes timing information will be revealed. When a password is typed, e.g. after a "su"-command, the echo of the characters is disabled. An Attacker can take advantage of this by observing the data stream (i.e. attackers can measure the arrival time of the password keystrokes packets, and get the inter keystroke timing of the super user's password). If data flows only in one direction (client to server), it is highly probable that the user is typing a password. Then an attacker gains the knowledge that a password is transmitted and also its exact length. Figure 1 show the dataflow, when the user types a password [1, 2, 22, 23].

C. Instance Based Learning (IBL) Model Timing Attack System

The timing attack system is a system that uses instance based learning model to measure inter arrival time of a keystroke, analyses such keystrokes and infers the keystroke press in order to determine the password transmitted in a SSH session. Figure 2 shows a block diagram of a timing attack system [2, 22, 23].

1. Components of Attack System Schematics

a.) Network Sniffer Module (NSM): This sub-unit monitors network data. Sniffers usually act as network probes or "snoops" They examine network traffic, making a copy of the data without redirecting or altering it. (e.g. wire shark). The sniffer eavesdrops the connection, only the size of the transmitted payload and the time between the packets of the interest [1, 2, 22, 23].

b.) Information Analysis Module(IAM): This module is also known as “the Parser”: The module processes the sniffer’s output file, searches for a SU command, and extracts the password packets arrival-time intervals(i.e. it detects if a password or normal input is typed). Only if the user types a password, the corresponding keystroke timings are forwarded to the instance based learning algorithms. The Information analysis module or parser filters out password keystrokes [1, 2, 22, 23].

c.) Instance Based Learning Algorithms (IBLA): This module processes the timings and produces a list of the n most likely passwords according to the statistics. The modules are independent, and not automated. This means that time will be activated manually in the correct order for the full attack. The module calculates possible password candidates [1] using the highest number of latency that matches the query latency.

$$\overrightarrow{X_{nn}} = \arg\max\left(d\left(\overrightarrow{X}, \overrightarrow{X_q}\right)\right) \qquad (3)$$

$$Y = f\left(\overrightarrow{X_q}\right) = f\left(\overrightarrow{X_{nn}}\right) \qquad (4)$$

where query latency= $\overrightarrow{X_{nn}}$ and possible password candidate = $f\left(\overrightarrow{X_q}\right)$

d.) Keystroke Timing Characteristics Module (KTCM): specifies the characteristics of a special user general timing characteristics which helps to determine user’s pattern on the network [22, 23].

e.) Round-Trip-Times (RTT) Module: This module measures the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received to both host.

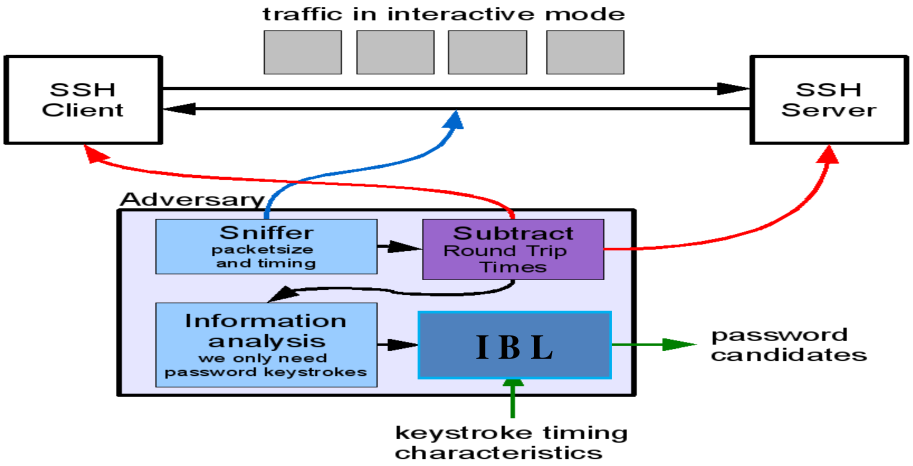


Figure 2: Timing Attack Systems [2, 22, 23]

2.) Timing problems

OpenSSH has implemented a countermeasure against timing attacks. If a password is typed and thus the echo of the characters is disabled, OpenSSH sends blank packets back to the user, to complicate timing attacks. So the information analysis component cannot distinguish if the client typed a password or not [1, 2, 22, 23]. But there is a possible solution for this problem. An attacker can measure the Round-Trip-Times (RTT) to both hosts, he eavesdrops. If these times are exact enough, he can reveal the even presented timing information by subtracting the RTT's from the eavesdropped timings. So the attacker can distinguish, whether the echo-mode is activated or not. Moreover this optimization brings another advantage, as the attacker can use the more precise timings to obtain better results from the instance based learning algorithm. Figure 2 shows attack system schematics, which includes a RTT-measurement component. The data stream within the adversary has changed also, as long as all timing information goes through the RTT-subtractor [2, 22, 23].

D. Instance Based Learning (IBL) Model

In machine learning, instance-based learning or memory-based learning is a family of learning algorithms that, instead of performing explicit generalization, compare new problem instances with instances seen in training, which have been stored in memory. It is called instance-based because it constructs hypotheses directly from the training instances themselves. This means that the hypothesis complexity can grow with the data: in the worst case, a hypothesis is a list of *n* training items and the computational complexity of classification a single new instance is *O*(*n*)[12,22,23].

A major advantage that instance-based learning model has over other methods of machine learning is its ability to adapt its model to previously unseen data. Where other methods generally require the entire set of training data to be re-examined when one instance is changed, instance-based learners may simply store a new instance or throw an old instance away. This family of classifiers was used in medical field as second-opinion diagnostic tools and as tools for the knowledge extraction phase in the process of knowledge discovery in databases [11, 22, 23].

1) **Keystroke Timing as IBL**

For this system, each character pair is a non-observable state. The latency between two keystrokes is the observed output. Each state corresponds to a pair of characters, so that typing the sequence k_o, k_1…………k_T is a process that goes through T states, q_1, q_2….. q_T . We denote *yt* the observed latency of state *qt*.

To model the process as IBL, we have to make two assumptions:

- The characters are uniformly distributed so that the probability of transition from the current state to another state depends only on the current state. This assumption usually holds for “good” passwords.
- The probability distribution of the latency is dependent only on the current state but in some cases it does not hold due to typing latency of the character which changes based on individual pattern.

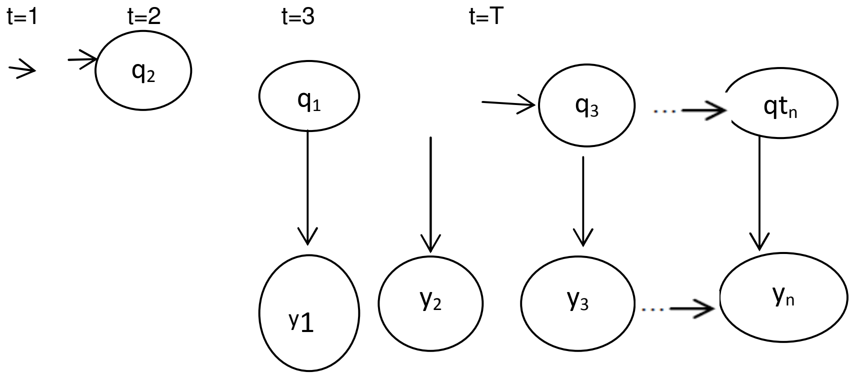


Figure 3: State diagram for Instance Based Learning Model [22, 23]

The model was developed as follows:

$$x = \{x_{1,} \dots , x_{n,}\} \qquad (5)$$

Where X are characters typed by the user.

$$q = \{q_1 , q_2 \dots q_n\} \qquad (6)$$

Where q is a sequence of character pairs from x $\{x_{1,} x_{2,\dots} \cdot , x_{n.}$

$$y = \{y_1 , y_2 \dots\dots y_n\} \qquad (7)$$

Where y are sequence of latencies of q and

$$q_1 = x_1 \rightarrow x_2 \qquad (8) \qquad\qquad q_2 = x_3 \rightarrow x_2 \qquad (9)$$

$$q_n = x_n \rightarrow x_{n-1} \qquad (10)$$

$$y_1 \rightarrow q_1 \qquad (11) \qquad\qquad y_2 \rightarrow q_2 \qquad (12)$$

$$y_3 \rightarrow q_3 \qquad (13)$$

$$y_n \rightarrow q_n \qquad (14)$$

(y is the latency observable for q and q is the character pairs interval).

Therefore

f_r (q|y) forms a unique set that essentially gives a ranking for each possible character sequence q which forms the data set.

$$x = \{x_i = f_r(q|y)_i\} \qquad (15)$$

each (x) is an n-tuple of attribute value

$$\overrightarrow{x_i} = \; < a_{i, \dots,} a_k > \qquad (16)$$

where $\overline{X}_1$ represents a set of data containing the latency for every character pairs.

There is a function that map X onto some set Y, where Y represent the user(s).

$$f : X \rightarrow Y \qquad (17)$$

The Data is a set of tuples <x_i, target function values>

$$D = \{< \overrightarrow{X_i}, f(\overrightarrow{X_i}) >, \dots < \overrightarrow{X_m}, f(\overrightarrow{X_m}) >\} \qquad (18)$$

This data forms the contents of the database

then query point $\overrightarrow{X_q}$, was sent to the application to predict $f\left(\overrightarrow{X_q}\right)$

Next, find the nearest member of data set to the query based on the threshold value

$$\overrightarrow{X_{nn}} = \arg\min\left(d\left(\overrightarrow{X}, \overrightarrow{X_q}\right)\right) \qquad (19)$$

Where: d is the distance function and $\overrightarrow{X_{nn}}$ is the list of possible user(s) assign the nearest neighbour's output to the query

$$h\left(\overrightarrow{X_q}\right) = f\left(\overrightarrow{X_{nn}}\right) \qquad (20)$$

find the highest point (thus find the highest number of latencies that matches the query latency)

$$\overrightarrow{X_{nn}} = \arg\max(d(\overrightarrow{X}, \overrightarrow{X_q})) \qquad (21)$$

then, give query its value as

$$Y = f\left(\overrightarrow{X_q}\right) = f\left(\overrightarrow{X_{nn}}\right) \qquad (22)$$

E. Collecting timing information with JavaScript and JSP

To gather a lot of timing information require for IBL attacks system, an opponent will extend a website with sniffing algorithms on window's operating system or Unix's operating system since most of the SSH-users have an internet connection. Since HTTP is a stateless not-interactive mode protocol, most of the users will not anticipate that timing information can be collected while filling out a survey. But it is possible to do so. Even without using a Java- or Flash-applet which could appear conspicuous. JavaScript is fully adequate for this task [2].

At first the opponent has to create a HTML-form which includes some questions, the user will probably answer with a lot of characters. In addition to the text-boxes, the system has one field with id "content" and type "secret", to carry out the information to the web server. Because it is easy with JSP to establish a database connection and to interpret data that comes from the browser as POST-Request, we use this language to build the timing statistics collector [2].

The JavaScript program in listing 1 used to collect keystroke timings:

```
var last = 0;
function init()
{
document.onkeypress=keyboard_inter_func-tion;
}
function keyboard_inter_function()
{
var time = (new Date()).getTime();
var x = document.getElementById("content").value();
x.innerHTML += e.charCode + "." + (time-last) + ",";
last = time;
}
```

Listing 1: JavaScript to collect keystroke timings

4. IMPLEMENTATION

A. Web Application Server

A web server is a program that, using the client/server model and the World Wide Web's hypertext transfer protocol (HTTP), serves the files that form web pages to web users (whose computers contain HTTP clients that forward their request). Web servers often come as part of a larger package of internet- and internet-related programs for File Transfer Protocol (FTP) files, and building and publishing web pages. Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems (now Oracle), and provides a "pure Java" HTTP web server environment for Java code to run. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

B. Web Server Setup

The description below uses the variable name $CATALINA_BASE to refer the base directory against which most relative paths are resolved. If Tomcat 6 has not been configured for multiple instances by setting a CATALINA_BASE directory, then $CATALINA_BASE will be set to the value of CATALINA_HOME, the directory into which Tomcat 6 has been installed.

To install and configure SSL support on Tomcat 6, the follow simple steps were carried out.

1. Create a key-store file to store the server's private key and self-signed certificate by executing the following command:

Windows:

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

Figure 4a: Command for creating self-signed certificate on windows

Unix:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```

Figure 4b: Command for creating self-signed certificate on unix

The next phase is to specify a password value of "programit" or any password of choice.

2. Uncomment the “SSL HTTP/1.1 Connector” entry in $CATALINA_BASE/Conf/server.xml and modify as describe below Tomcat currently operates only on JKS, PKCS11 or PKCS12 format keystores. The JKSformat is Java's standard "Java KeyStore" format, and is the format created by the keytool command-line utility. This tool is included in the JDK. The PKCS12 format is an internet standard, and can be manipulated via (among other things) OpenSSL and Microsoft's Key-Manager.

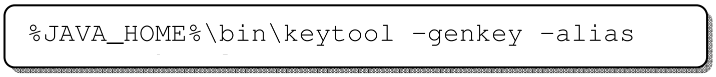
Each entry in a key-store is identified by an alias string. Whilst many key-store implementations treat aliases in a case insensitive manner, case sensitive implementations are available. The PKCS11 specification, for example, requires that aliases are case sensitive. To avoid issues related to the case sensitivity of aliases, it is not recommended to use aliases that differ only in case.

To import an existing certificate into a JKS keystore, please read the documentation (in your JDK documentation package) about key-tool. Note that OpenSSL often adds readable comments before the key, key tool does not support that, so remove the OpenSSL comments if they exist before importing the key using key tool.
To import an existing certificate signed by your own CA into a PKCS12 keystore using OpenSSL the following command would be executed:

```
Openssl pkcs12 -export -in mycert.crt  -inkey mykey.key \  -out
mycert.p12 -name tomcat -CAfile myCA.crt \       -caname root -chain
```

Figure 5: Command for importing existing certificate

To create a new key-store from scratch, containing a single self-signed Certificate, execute the following from a terminal command line:
Windows:



```
%JAVA_HOME%\bin\keytool -genkey -alias
```

Figure 6a: Creating a keystore from scratch on windows

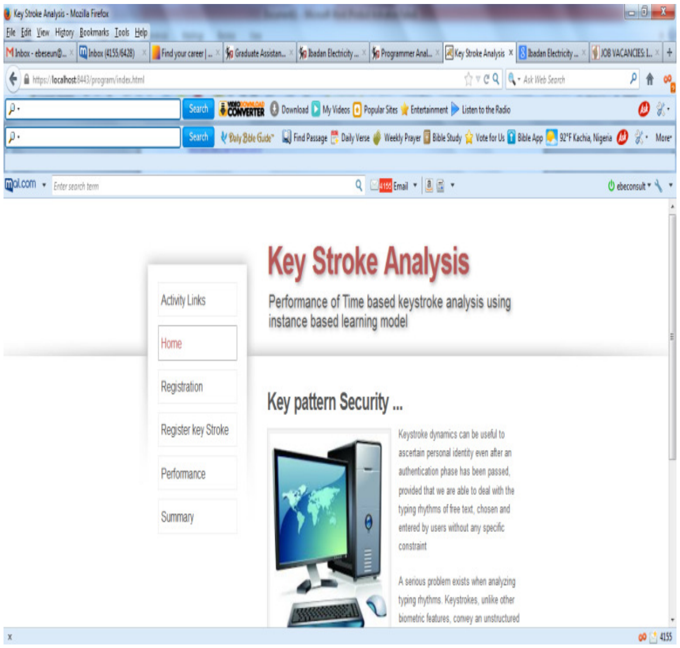
Unix:

```
$JAVA_HOME/bin/keytool -genkey -alias
```

Figure 6b: Creating a key-store from scratch on unix

C. Web Application Home Page

Figure 7 shows the web application home page. This page contains a selection of text that contains some basic information about the application, but most importantly it offers links to the “key stroke analysis” and “Performance” pages. These pages are used to accomplish the key stroke attack analysis and also implement its performance. The web application home page and subsequent pages uses secured http (https) protocol which provides a secured communication between the client and the server.



Key Stroke Analysis

Performance of Time based keystroke analysis using instance based learning model

Key pattern Security ...

Keystroke dynamics can be useful to ascertain personal identity even after an authentication phase has been passed, provided that we are able to deal with the typing rhythms of free text, chosen and entered by users without any specific constraint

A serious problem exists when analyzing typing rhythms. Keystrokes, unlike other biometric features, convey an unstructured

Figure 7: Web application home page

5. RESULTS

The table1shows the outcome of the activities carried out by the clients (users) on a web browser using the application. Also, figure 8 shows the graph of the time sequence for the character pairs typed by each user.

Table 1: Summary of cumulative time for each user keystroke identification in a SSL protocol

S/N	Cumulative Time(msec)	S/N	Cumulative Time(msec)
1	2281	26	4265
2	3637	27	4315
3	3130	28	4032
4	3160	29	3895
5	2795	30	3999
6	1726	31	4557
7	2749	32	4217
8	2040	33	2574
9	2452	34	2468
10	3107	35	2538
11	2611	36	4967
12	3047	37	4815
13	2952	38	4317
14	3207	39	2659
15	3297	40	4613
16	3130	41	5200
17	3096	42	3144
18	3210	43	4479
19	3203	44	4044
20	3290	45	5223
21	3706	46	3697
22	3550	47	4518
23	3696	48	3115
24	3075	49	4597
25	3763	50	4181

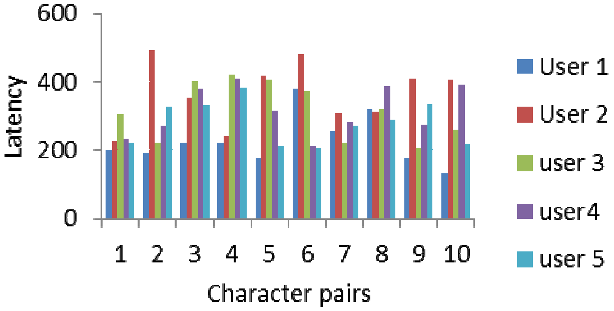


Figure 8: graph for the time sequence for the character pairs typed by each user

6. DISCUSSION

Several theories, approaches, techniques, models and methodologies were studied before Instance Based Learning model was used for this research. The research work discussed the keystroke timing analysis attacks on SSH using Instance Based Learning Model. The researchers also analyzed if it were possible to perform the timing analysis attack as it was described in [1],[2],[22] and [23] work. The research work discovered that, there are countermeasures against timing analysis implemented in the latest version of OpenSSH describe in [2], [22] and [23] because it is not possible to prevent side channel attacks completely. However, there are methods to complicate those attacks as describe in [2], [22] and [23] for example. *1)* When a "blank message" is sent (non echo mode) to the level of a normal echo packet. There are several ways to accomplish this, e.g. the "blank message" could be formed of a terminate or null byte and of that byte which would have been sent in echo mode. This would achieve that the "blank message" is sent subsequent and not immediately [2],[22], and [23]. *2)* In interactive mode (SSH-client), the client cans time-pad the keystrokes that are sent to the server. The time padding could be around 300ms which would damage all keystroke timing statistics.

This can be implemented by filling the keystrokes into a queue and dequeue packets not before the expiration of the time-padding interval. A disadvantage of this method is that fast writers could be annoyed by the debased performance. To not bother SSH-users in private networks, this method could be controlled by the SSH-Server as describe in [2]. The Implementation of an Instance Based. Learning Model for Timing Analysis of Keystrokes and Timing Attacks on Secure Shell provid a fix for the attack that makes this countermeasure non-effective by providing a Round-Trip-Times (RTT) Module subsystem in the attack system as describe in [2] but with **minimal time** taken in breaking the passwords and provide better performance in term of efficiency using various parameters such as **Sequence Time Threshold (STT)** which is the number of time of a latency sequence between characters pair that can be used to identify a user on the system, **Records Correctly Classified (R_{cc})** that is the total number of users that were correctly identified for a particular number of sequence time set as threshold and **Record Wrongly Classified (R_{wc})** which as to do with the total number of users that were wrongly correctly identified for a particular number of sequence time set as threshold compare to the other methods used in implementing the work by[1],[2],[22]. An attacker can measure the Round-Trip-Times (RTT) to both hosts, he eavesdrops. If these times are exact enough, he can reveal the even presented timing information by subtracting the RTT's from the eavesdropped timings. So the attacker can distinguish, whether the echo-mode is activated or not.

Moreover this optimization brings another advantage, as the attacker can use the more precise timings to obtain better results from the Instance Based Learning Model. In Table 1, 50 users were used to determine how efficient and effective the system can be using Instanced Based Learning Model to handle timing analysis of keystrokes and timing attacks on Secure Shell (SSH). The Table shows the cumulative time taken by every user to type 30 characters. The characters pairs typed and the corresponding time taken are transferred through a secure TCP/HTTP protocol from the source (host) to destination (server). The different value for the cumulative time shows that every user has distinctively different time value for typing characters on the keyboard; this can be used to identify users. The graph in figure 8 describe the users' typing pattern on the system which determine how much latency per character pair a user spent on the application. From the graph above, we inferred that user 2 do not know how to type fast on the Keyboard and this affect the rate at which his/her password can be cracked in respective of his/her combination of characters used in forming the password.

A. Critical Evaluation of Results

i. **Number of Character**: The users have the opportunity to type up to 2000 characters on the system. The system was design in such a way that it can identify each user that make used of the system at any point in time and it has the capacity to reveal their information without wasting much time. There is no minimum number of characters to be typed by the user. What really matter is the user's character latency but each typing sequence must be evenly distributed to enable the system to classify correctly.

ii. **Type Bearing:** The user's type bearing is another factor that the research work considered when developing the system. The system was developed in such a way that it can identify a user whether he/she is a slow type or fast type. Therefore, irrespective of your typing skills the developed system will classify you correctly.

iii. **Capacity to Crack:** Instance based learning model approach to timing analysis of keystrokes and timing attack on a secure shell has capacity to identify any user and reveal their information with a precise time irrespective of their characters combination. This show how powerful the developed system is in term of efficiency and effectiveness when compare with other models that has been used before by other researchers.

7. CONCLUSION

In this paper, it has been discovered that Secure Shell (SSH) is not secure as people popularly believed because there are serious security treats emanated from the protocol when carryout timing analysis of keystrokes and timing attacks. Implementation of an Instance Based Learning (IBL) Model for Timing Analysis of Keystrokes and Timing Attacks on Secure Shell learns when a password is typed, how long the password is and reveals the password irrespective of characters combination that form the password.

8. DIRECTION FOR FUTURE WORK

In the future, this work will be extended to fully cover the timing analysis of keystrokes and timing attack on Cloud Computing environment. This is in line with the intent to develop a model that looks at the security issues that can arise with keystrokes analysis in Cloud Computing.

8.1 Challenges encountered while carrying out the Research

Capturing the keystrokes from various user(s) looking at their various characteristics wanted posing a problem in the research work but with the introduction of Instance Based Learning Approach, we were able to overcome the problem. Portability of the System across many operating systems also gives us a lot of problem because most of systems developed earlier were not potable. We were able to overcome the portability problem by writing series of codes to make the system platform independent. In view of the innovation, the system can be installed on both Window Operating System and Linus Operating System.

REFERENCES

[1] Song, D. and Tian, X.(2001).Timing Analysis of Keystrokes and Timing Attacks on SSH, 10th USENIX Security Symposium.

[2] [2] Noack, A.(2007).Timing Analysis of Keystrokes and Timing Attacks on SSH Revisited, Horst G¨ortz Institut f¨ur IT- Sicherheit Ruhr-Universit¨at Bochum.

[3] Lustig, M. and Shabtai, Y.(2001). Keystroke Attack on SSH, Final Project Report at Technion IIT .

[4] Sadeghi, A. and Wachsmann, C.(2004). Network

[5] Security II –Secure Shell," Ruhr- Universit¨at Bochum. http://www.kb.cert.org/vuls/id/13877.

[6] Weak CRC allows RC4 encrypted SSH1 packets to be Modified without notice. http://www.kb.cert.org/vuls/id/.

[7] Side channel attack. http://en.wikipedia.org/wiki/Sidechannel_attack.

[8] Asonov, D., and Agrawal, R. (2004) Keyboard Acoustic Emanations. In Proceedings of the IEEE Symposium on Security and Privacy.

[9] [9] Zhuang, L., Zhou, L., and Tygar, J.(2005). Keyboard Acoustic Emanations Revisited. In Proceedings of the 12th ACM Conference on Computer and Communications Security.

[10] Berger, Y., Wool A and Yeredor A. (2006). Dictionary Attacks Using Keyboard Acoustic Emanations: In Proceedings *of ACM* Conference on Computer and Communication Security (CCS).

[11] Shah, G., Molina A. and Blaze, M.(2006).Keyboards and Covert Channels. 15th USENIX Security Symposium.

[12] Gagliardi,F (2011). Instance-based classifiers applied to Medical databases: Diagnosis and knowledge extraction, Artificial Intelligence in Medicine," 123-139.doi:10.1016/j.artmed.

[13] Hogye, M.,Hughes C., Sarfaty, J., and Wolf, J.(2001).Analysis of the Feasibility of Keystroke Timing Attacks Over SSH Connections. Research Project at University of Virginia..

[14] Menezes, A., Oorschot P. and Vanstone, S.(1997). Handbook of Applied Cryptography. CRC press, (1997).

[15] Russel, S., Norvig, P.:, Artificial Intelligence. A modern approach. Prentice Hall, (1995).

[16] Designer, S. and Song, D.(2001). Passive Analysis of SSH Traffic: ,http://www.securiteam.com/security News/5KP0O0A3PU.html.

[17] Brumley D. and Boneh,D.(2005). Remote Timing Attacks are Practical. Computer Networks, Volume 48, Issue 5, Pages 701–716.

[18] Abu, M., Fabian, R. and Terzis., A(2005) : Worm Evolution Tracking via Timing Analysis," Proceedings of the ACM workshop on Rapid malcode, 52-59.

[19] Trestle, J.(1998). Timing attacks against trusted path. IEE Symposium on Security and Privacy: page 15-134.

[20] Alshahwan, N., and Harman,M.(2011).Automated web Application testing using search based software Engineering. 26th IEEE/ACM International Conference on Automate Software Engineering Lawrence, Kansas. USA.

[21] Silva,L., Alonso, J. and Andrzejak, J.(2007). Using Virtualization to improve software rejuvenation. Network Computing and Applications, Sixth IEEE International Symposium on Pages 33-44.

[22] Alese, B., Dahunsi, F. Ajayi, E., and Durojaye, S. (2013). Instance Based Learning Model for Timing Analysis Keystrokes to Perform Timing Attacks on the Secure Shell Protocol", Asian Journal of Computer and Information Systems (ISSN: 2321 – 5658) Volume 01– Issue 04.

[23] Ajayi, E.A., Alese, B.K., and Dahunsi,F. M.,(2016). Keystrokes Timing Analysis and Timing Attacks System on Secure Shell: Instance Based Learning (IBL) Model Approach Revisited", Proceedings of the iSTEAMS Multidisciplinary Cross-Border Conference University of Professional Studies, Accra Ghana.