Digital Innovations & Contemporary Research in SCIENCE & ENGINEERING
Vol. 5 No. 2, June 2017

Article Citation Format
Ajayi E. A, Ajayi B.F & Alese, B.K. (2017). Comparative Analysis between Instance Based Learning Model and Hidden Markov Model in handling Keystrokes Analysis Problem on Secure Shell. Journal of Digital Innovations & Contemp Res. In Sc., Eng & Tech. Vol. 5, No. 2. Pp 123-136.

Article Progress Time Stamps

Article Type: Research Article
Manuscript Received: 17rd March, 2017
Review Type: Blind
Review/Acceptance Information Sent : 12th June, 2016
Final Acceptance:: 13th June, 2017
DOI Prefix: 10.22624

Comparative Analysis between Instance Based Learning Model and Hidden Markov Model in handling Keystrokes Analysis Problem on Secure Shell

Ajayi E. A, & Ajayi B.F
Computer Science Department
Kebbi State Polytechnic
Dakingari Kebbi State, Nigeria
ebeconsult@myself.com

Alese, B.K.
Computer Science Department
Federal University of Technology
Akure Ondo State, Nigeria
bkalese@futa.edu.ng

ABSTRACT

This paper makes a comparative analysis between Instance Based Learning (IBL) Model and Hidden Markov Model (HMM) approaches to Keystrokes Analysis on Secure Shell. A timing attack is a security exploit that allows an attacker to discover vulnerabilities in the security of a computer or network system by studying how long it takes the system to respond to different inputs. Secure Shell is a protocol for securing remote access across an insecure network. Due to weaknesses found in secure shell protocol in a network environment, gave attackers the opportunity to attack the network system. The research works carry-out a comparative analysis between Instance Based Learning Model (IBL) and Hidden Markov Model (HMM). The result shows that Instance Based Learning Model is more accurate and efficient in handling Timing Attack System problem in a secure shell protocol of a network environment.

Key words: Instance Based Learning Model, Hidden Markov Model, Timing Attack System, Protocol, Cryptographic

1. BACKGROUND TO THE STUDY

Internet access has become increasingly inexpensive and available which 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 [2, 3, 4, 5, 8]. Insecurity of Internet Technology has led my developers to tries several methods and approaches to see that their clients' information is secure and safe.

Some years back, internet users used astonishingly insecure networking applications such as telnet, rlogin, or ftp, which passed all confidential information (including users' passwords) in the clear form over the network [2,3,4,8]. This situation was intensified 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 show this through our new approach[2,3,4].

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:

1. **Cryptographic/ Protocol Attack**: This is a method for circumventing the security of a cryptographic
2. system by finding a weakness in a code, cipher, cryptographic protocol or key management scheme[2,3,4,5,13,18,19]. A well-known example of this attack is the insertion attack against SSH-1.
3. **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 [2, 3, 4, 5].
4. **Side Channel Attacks**: Side channel attacks are based on information that is gained from the physical implementation of a cryptosystem [2, 3, 4, 5, 25] 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. STATEMENT OF PROBLEM

Secure Shell (SSH) has well established encryption and authentication mechanisms but it has two major weaknesses namely:

i. 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).
ii. when SSH 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.

In view of the above problems, our work looks at two models (Instance Based Learning Model and Hidden Markov Model) to handle these minor problems that led to serious security treat to SSH protocol in a network environment.

3. OBJECTIVE

The objective of this research work is to carry-out comparative analysis between Hidden Markov Model and Instance Based Learning Model in handling keystroke analysis on secure shell to reveal information per keystroke pair problem.

4. RELATED WORK

Due to the importance of Keystrokes Analysis 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 [8] 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

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

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 [22], 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 Modelling, Hidden Markov Model and an n- Viterbi algorithm were used to describe how key sequences can be inferred.

In [9], 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 [1], 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.

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

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 analysed 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.

5. METHODOLOGY

This section look at the architectural design of the keystrokes analysis within the SSH protocol

5.1 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 [2, 3, 4,5,16, 22].
- 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

Digital Innovations
& Contemporary Research in
SCIENCE & ENGINEERING
Vol. 5 No. 2, June 2017

5.2 Realization of the Attack

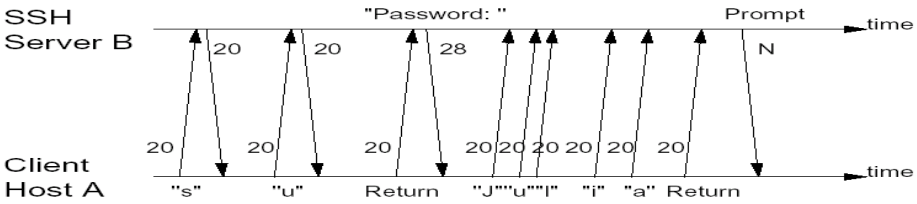

Figure 1: Schematic Illustration of a sniffing scenario [2, 3, 4, 5, 16, 22]

Figure 1 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 [2, 3, 4, 5, 16, 22].

5.3 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, 3, 4, 5, 16, 22].

5.3.1 Basic 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 [2, 3, 4, 5, 16, 22].

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 [2, 3, 4, 5, 16, 22].

c) **Keystroke Analysis Algorithms (KAA):** 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.

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

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

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 [5].

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.

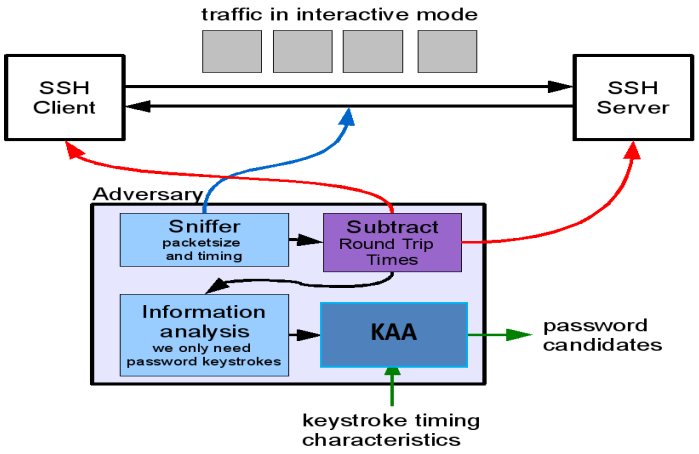

Figure 2: Timing Attack Systems [2, 3, 4, 5, 16, 22]

5.3.2 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 [2, 3, 4, 5, 16, 22].

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, 3, 4, 5, 16].

6. 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 3a: Command for creating self-signed certificate on windows

Unix:

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

Figure 3b: Command for creating self-signed certificate on unix

   and 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 key tool 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 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 4: 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 5a: Creating a keystore from scratch on windows

Unix:

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

Figure 5b: Creating a key-store from scratch on unix

C. Web Application Home Page

Figure 6 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.

Figure 6: Web application home page

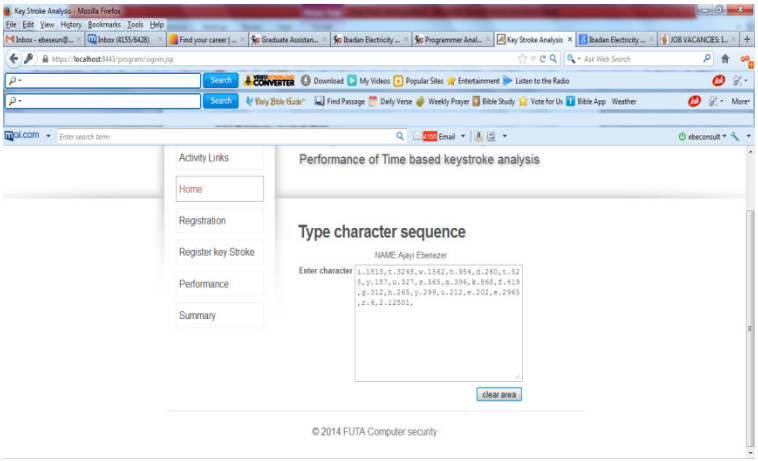

Figure 7: Keystorokes Capturing Page

Figure 7 shows Keystroke capturing page that accepts characters typed from the keyboard into the text area and time interval between every character typed, this time value is sent to the server to be saved for that particular person or user session. Every user using this page is giving a unique number for identification and every activity (character and time value are recoreded).The time value and character obtained from this page are analysed to uniquely identify the user subsequently. The "Clear area space" button is used to clear or remove the content of the text area to provide a free space for typing.

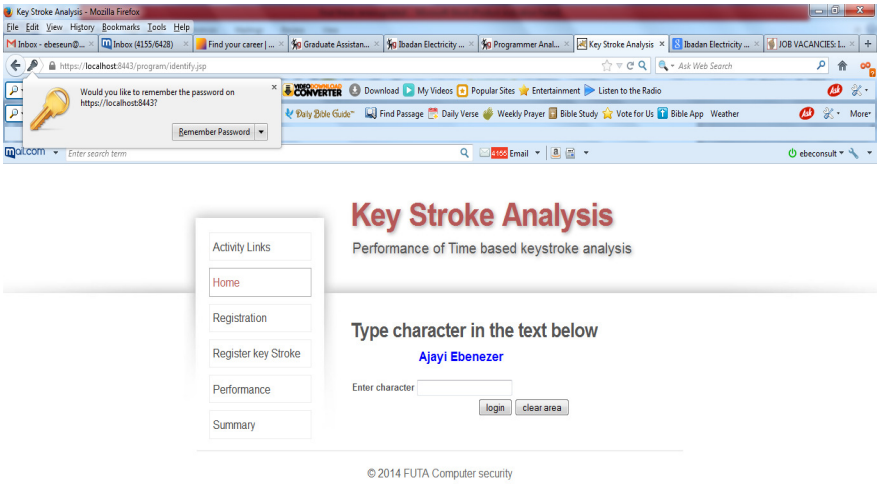

Figure 8: Keystroke Performance Analysis Page

Figure 8 shows the Keystrokes Performance Analysis Page. The data gathered from the keystroke performance analysis page above is used to determine the user on the performance of the keystroke attack page. This page is displayed when the "performance" link on the web application home page is clicked.

The page tries to determine the user as he/she type character on the text area on the page and reveal the user's name and password accordingly. The table1shows the outcome of the activities carried out by the clients (users) on a web browser using the application. Figure 9 shows the graph of IBL and HMM latency comparison typed by each user.

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

S/N	Key sequence	HMM Latency			IBL Latency		
		Start Latency(msec)	End Latency(msec)	Mean Latency (msec)	Start Latency(msec)	End Latency(msec)	Mean Latency (msec)
1	a->e->n	248.257	197.001	222.63	201	124	162.50
2	b->e->n	134.548	325.465	230.01	222	178	200.00
3	e->e->n	245.268	189.544	217.41	199	121	160.00
4	h->e->n	120.117	149.703	134.91	115	139	127.00
5	i->e->n	168.367	206.581	187.47	160	211	185.50
6	k->e->n	152.465	191.877	172.17	150	189	169.50
7	m->e->n	164.501	212.866	188.68	160	215	187.50
8	n->e->n	105.127	123.233	114.18	102	121	111.50
9	r->e->n	168.150	326.265	247.21	221	170	195.50
10	s->e->n	206.624	206.502	206.56	191	190	190.50
11	u->e->n	189.254	263.705	226.48	250	180	215.00
12	y->e->n	269.264	202.869	236.07	200	210	205.00
13	z->e->n	163.557	234.820	199.19	200	189	194.50

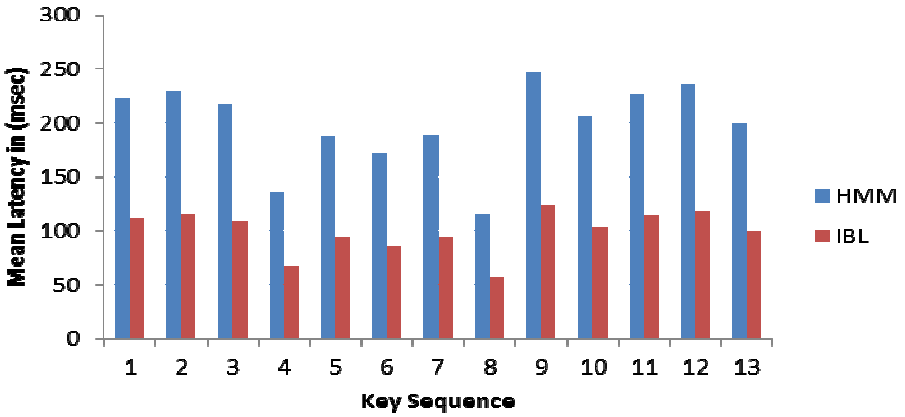

Figure 9: Graph of IBL and HMM latency comparison

In table 1 and figure 9, the Instances Based Learning Model was compared with Timing Analysis of Keystrokes and Timing Attacks on SSH a research carried out by Noack, 2007. He observed the typing bearings of different users which can be averaged out by applying 2nd degree HMM's (Hidden Markov Model) and other statistical methods (that is a filter that eliminates the inner cycle of the user) can be investigated. But the information gain from the n-Viterbi algorithm was very unequal. Applying a 2nd degree HMM with more dependencies but one state, is however useful.

The researcher wanted to underline this by an experiment which is based on a 2nd degree HMM. A 2nd degree HMM will help gain more information from each keystroke, since the timing information is more detailed (also for touch typing writers). Nevertheless when comparing with Instance Based Learning Model, it can be safely concluded that Instance Based Learning Model is more efficient and effective in handling keystrokes analysis than Hidden Markov Model because the time taken (i.e. mean latency) in identify a user on the system using Instance Based Learning Model is smaller and more precise when compare to Hidden Markov Model.

7. DISCUSSION OF FINDINGS

Several theories, approaches, techniques, models and methodologies were studied before Instance Based Learning Model and Hidden Markov Model were used for this research to carry out the comparative analysis. The research work discussed the keystroke timing analysis attacks on SSH using Instance Based Learning Model and Hidden Markov Model. The researchers also analysed if it were possible to perform the timing analysis attack as it was described in [2],[3],[4],[5],[16] and [22] work. The research work discovered that, there are countermeasures against timing analysis implemented in the latest version of OpenSSH describe in [2],[3],[4],[5] and [16] because it is not possible to prevent side channel attacks completely. However, there are methods to complicate those attacks as describe in [2],[3],[4],[5] and [16], 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],[3] and [16].

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. The disadvantage of this method is that fast writers could be annoyed by the debased performance. The Implementation of an Instance Based Learning Model and Hidden Markov Model for Keystrokes Analysis on Secure Shell with a distinct differences in their output provide 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],[3],[4],[5],and [16] 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 Hidden Markov Model used in implementing the work by[2],[3],[4],[5],[16] and [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 compare with Hidden Markov 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 keystrokes analysis on Secure Shell (SSH).

The Table shows the mean latency 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 latency 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 9 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 Instance Based Learning Model is more efficient and effective in handling keystrokes analysis on Secure Shell than Hidden Markov Model

A. Critical Evaluation of Results in Instance Based Learning Model and Hidden Markov Model

i. **Number of Character**: The users have the opportunity to type up to 2000 characters on both IBL and HMM system. The IBL 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 but HMM has a minimum number of character to be typed by the user.

ii. **Type Bearing:** The user's type bearing is another factor that was considered when developing the IBL 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 IBL system will classify you correctly but in HMM system, categories of user's typing skill must be specified before classification can be done.

iii. **Capacity to Crack:** Instance Based Learning Model approach to keystrokes analysis on a secure shell has capacity to identify any user and reveal their information with a precise time irrespective of their characters combination. This shows how powerful the developed system is in term of efficiency and effectiveness when compare Hidden Markov Model

B. System Performance for Instance Based Learning Model and Hidden Markov Model

i. **Sequence Time Threshold (STT)** refers to as the number of time of a latency sequence between characters pair that can be used to identify a user on the system.

ii. **Records Correctly Classified (R_{cc})** refers to as the total number of users that were correctly identified for a particular number of sequence time set as threshold.

iii. **Record Wrongly Classified (R_{wc})** refers to as the total number of users that were wrongly correctly identified for a particular number of sequence time set as threshold.

The Instance Based Learning Model and Hidden Markov Model were tested to determine their performances using various values of threshold and the following were deduced:

a. when the number of time sequence latency for identification is set to 6, forty-four (44) users were correctly identified using Instance Based Learning Model while (42) users were correctly identified using Hidden Markov Model.

b. when the number of time sequence latency for identification is set to 7 and 8, forty-one (41) users were correctly identified using Instance Based Learning Model while thirty-nine (39) users were correctly identified using Hidden Markov Model.

c. when it is set at 9, thirty-eight (38) users were correctly identified using Instance Based Learning Model while thirty-six (36) users were correctly identified using Hidden Markov Model.

The percentage of correctly classified records and wrongly classified records are obtained from the Equation 5.1 and 5.2.

Percentage of successfully identified record user (P) can be estimated using the equation:

$$P_s = \frac{R_{CC}}{T_r} \times 100 \qquad 5.1$$

Where R_c is the total number of records correctly classified and T_r is the total records

P_e, the percentage of wrongly identified records can be derived from:

$$P_e = 100 - P_s \qquad 5.2$$

Table 2 below shows the record of the performance of the system given a varying sequence time number as threshold for the total number of users (50) registered. For better performance of the system, sequence time threshold is set to 6.

Table 2 Summary of threshold used for performance in Instance Based Learning Model and Hidden Markov Model

S/n	Total Record	Sequence time Threshold	Record Correctly Classified		Correctly Classified (%)		Record wrongly Classified		Wrongly Classified (%)	
			IBL	HMM	IBL	HMM	IBL	HMM	IBL	HMM
1	50	6	44	42	88	84	6	8	12	16
2	50	7	41	39	82	78	9	11	18	22
3	50	8	41	39	82	78	9	11	18	22
4	50	9	38	36	76	72	12	14	24	28

8. CONCLUDING REMARKS

In this research work, it has been revealed that when handling keystrokes analysis on Secure Shell, Instance Based Learning Model can be more efficient and effective compare to Hidden Markov Model because the time taken (i.e. mean latency) in identify a user on the system is smaller and more precise. More so, it has been established that Secure Shell (SSH) is not secure as people popularly believed because there are serious security treats emanated from the protocol when carryout keystrokes analysis because, hackers can easily learns when a password is typed, how long the password is and reveals the password irrespective of characters combination that form the password.

9. CONTRIBUTIONS TO KNOWLEDGE

The research work produced a system that:

(a) reveal a password that a user typed in a SSH- session.
(b) that analyses user(s) pattern when typing using computer keyboard on a network.
(c) carryout comparative analysis between Instance Based Learning and Hidden Markov Model.

REFERENCES

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

[2] 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.

[3] Ajayi, E.A., Alese, B.K., Dahunsi, F.M. and 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.

[4] [4] Ajayi, E.A., Ajayi B.F., Alese, B.K., Dahunsi, F. M., and Oyeniran, A.T. (2017) . Keystrokes Timing Analysis and Timing Attacks System on Secure Shell: Instance Based Learning (IBL) Model Versus Hidden Markov Model, Proceedings of the 8th iSTEAMS Multidisciplinary, Caleb University, Lagos, Nigeria. Pp 73-84.

[5] 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.

[6] 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.

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

[8] 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).

[9] Brumley D. and Boneh,D.(2005). Remote Timing Attacks are Practical. Computer Networks, Volume

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

[11] 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.

[12] 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..

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

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

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

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

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

[18] Sadeghi, A. and Wachsmann, C.(2004). Network. Lance Publishers NY

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

[20] 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.

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

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

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

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

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

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