Instance Based Learning Model for Timing Analysis of Keystrokes to Perform Timing Attacks on the Secure Shell Protocol

Ajayi E. Akinyemi[1], Durojaye D. Samson[2], F. M. Dahunsi[3] and B. K. Alese[4]

[1] Department of Computer Science
Federal Polytechnic,Idah
Kogi State, Nigeria

[2] Department of Mathematical Sciences
Kogi State University
Anyigba, Nigeria

[3] Department of Computer Science
Federal University of Technology,
Akure,Ondo State, Nigeria

[4] Department of Computer Science
Federal University of Technology,
Akure,Ondo State, Nigeria

ABSTRACT— ***The research present Instance Based Learning Model for timing analysis of keystrokes to perform timing attacks on the Secure Shell protocol. SSH is designed to provide a secure channel between two hosts. Despite the encryption and authentication mechanisms it uses, SSH has two weakness: First, the transmitted packets are padded only to an eight-byte boundary (if a block cipher is in use), which reveals the approximate size of the original data. Second, in interactive mode, every individual keystroke that a user types is sent to the remote machine in a separate IP packet immediately after the key is pressed, which leaks the inter-keystroke timing information of users' typing. The research shows how these seemingly minor weaknesses result in serious security risks. The research picks up the ideas of Song et al.(2001) and show that there are problems with their practicability today. The research implements a countermeasure against timing attacks which it analyses and then shows a possibility to handle it. The research also presents a method to collect keystroke timing characteristics from users silently. Evaluation of Instance based learning and Hidden Markov Model was done to show how effective an Instance based learning model can handle timing analysis of keystrokes and timing attacks on secure shell.***

Keywords— Secure Shell; Authentication; Encrypting Key., Key strokes

1. 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. In the past, most enterprises maintained their own modem bank dial-up access to their resources so that critical data elements were transmitted over the public network but unfortunately modem banks are expensive to maintain and don't scale well.

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. 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.

Secure Shell is a protocol that provides authentication, encryption and data integrity to secure network communications. Implementations of Secure Shell offer the following capabilities: secure command-shell, secure file transfer, and remote

access to a variety of TCP/IP applications via a secure tunnel. Secure Shell client and server applications are widely available for most popular operating systems.

Secure Shell (SSH) is a protocol for securing remote access across an insecure network. Information transferred using SSH is encrypted using strong encryption algorithms, providing strong protection from eavesdroppers on the network. Secure Shell offers a good solution for the problem of securing data sent over a public network. For example, using Secure Shell and the Internet for securely transferring documents and work products electronically, rather than using a traditional overnight courier can provide substantial cost savings.
Secure Shell provides three main capabilities, which provides for many creative secure solutions and these are:

i. Secure Command-shell
ii. Secure File transfer and
iii. Port Forwarding.

Secure Shell protocol also provides four basic security benefits namely:

i. User Authentication
ii. Host Authentication
iii. Data Encryption and
iv. Data Integrity

Aim

The objectives of this research work are:

(a) design of an Instance Based Learning model for the timing analysis of keystroke and timing attack on secure shell to reveal information per keystroke pair
(b) develop a software to implement the model designed in (a) above.

Objectives

The research work is expected to produce a system that:
(a) minimizes the time needed to reveal a password, a user typed in a SSH- session.
(b) analyses user(s) pattern on a network.

2. REVIEW OF RELATED LITERATURE

2.1 Overview of Secure Shell

Secure SHell (SSH) is a protocol for secure remote login and other secure network services over an insecure network.SSH (Secure SHell) is a network protocol which provides a replacement for insecure remote login and command execution facilities, such as telnet, rlogin and rsh. SSH encrypts traffic in both directions, reventing traffic sniffing and password theft. SSH also offers several additional useful features such as:

- ✓ Compression: traffic may be optionally compressed at the stream level.
- ✓ Public key authentication: optionally replacing password authentication.
- ✓ Authentication of the server: making "man-in-the-middle" attack more difficult.
- ✓ Port forwarding: arbitrary TCP sessions can be forwarded over an SSH connection.
- ✓ X11 forwarding: SSH can forward your X11 sessions too.
- ✓ File transfer: the SSH protocol family includes two file transfer protocols.

Secure Shell (SSH) provides an open protocol for securing network communications that is less complex and expensive than hardware-based VPN solutions. Secure Shell client/server solutions provide command shell, file transfer, and data tunneling services for TCP/IP applications. SSH connections provide highly secure authentication, encryption, and data integrity to combat password theft and other security threats. VanDyke Software clients and servers are mature native Windows implementations that offer a range of SSH capabilities and are interoperable with SSH software on other platforms.

2.2 Historical Background of Secure SHell

Secure SHell (SSH) was created by Tatu Yl¨onen in 1995 and it was first released under an open-source license. Later versions were to bear increasing restrictive licenses, though they generally remained free for non-commercial use. He went on to form SSH Communications security which sells commercial SSH implementations to this day. The earlier versions of his code implement what is now referred to as SSH protocol v.1. The first version of Secure Shell (SSH1) was designed to replace the non-secure UNIX "rcommands" (rlogin, rsh, and rcp).

In 1997 a process began to make the SSH protocols Internet standards under the auspices of the IETF. This lead to the development of version 2 of the SSH protocol. In the rewrite, the protocol was split into a transport layer, and connection and authentication protocols. Several security issues were also addressed as part of this process.
In 1999, the OpenBSD1 team discovered (by way of OSSH2) the early free versions for Tatu Yl¨onen's original code and set about cleaning them up to modern standards. The result was named "OpenSSH" and debuted in the OpenBSD 2.6 release of December 1999. OpenSSH was extended by Markus Friedl to support SSH protocol v.2 in early 2000. OpenSSH remains the most popular, complete and portable free SSH implementation and have been included in many OS releases. Secure Shell version 2 (SSH2) addresses some of the more serious vulnerabilities in SSH1 and also provides an improved file transfer solution.

2.3 Types of Secure Shell

Basically, there are two type of secure shell namely:

(a). Secure Shell version 1 this version replace the non-secure UNIX "rcommands" (rlogin, rsh, and rcp)

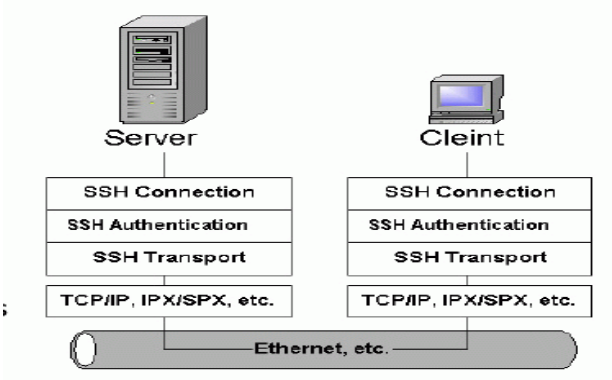

Figure 1: Secure Shell version 1

(b). Secure Shell version 2 this version addresses some of the more serious vulnerabilities in SSH1 and also provides an improved file transfer solution
Both the SSH-1 and the SSH-2 protocols use public key technology to establish a session key.

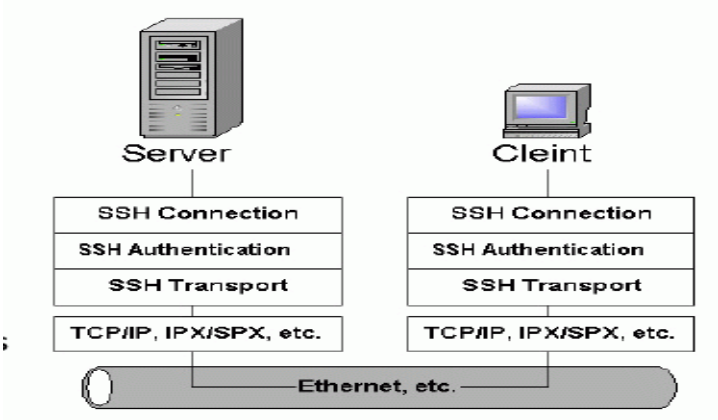

Figure 2: Secure Shell version 2

The session key is used to encrypt and decrypt packets using a symmetric cipher. The table below shows the Format of both SSH-1 and SSH-2 packets in the correct order.

2.4 Reasons for Using Secure Shell

i. Designed to be a secure replacement for rsh, rlogin, rcp, rdist, and telnet.
ii. Strong authentication. Closes several security holes (e.g., IP, routing, and DNS spoofing).
iii. Improved privacy. All communications are automatically and transparently encrypted.

iv. Secure X11 sessions. The program automatically sets DISPLAY on the server machine, and forwards any X11 connections over the secure channel.
v. No retraining needed for normal users.

SSH1				SSH2		
Field Size	*(bytes)*	*Encrypted?*		*Field Size*	*(bytes)*	*Encrypted*
Length of Data Payload(N)	4	No		Length of Data Payload(M)	4	No
Random Padding		Yes		Random Padding(P)	1	Yes
Packet Type	1	Yes		Packet Type	M – P – 1	Yes
Data Payload	N	Yes		Data Payload	N	Yes
MAC	Mac length	No		MAC	Mac length	No

vi. Never trusts the network. Minimal trust on the remote side of the connection. Minimal trust on domain name servers Pure RSA authentication never trusts anything but the private key.
vii. 0Client RSA-authenticates the server machine in the beginning of every connection to prevent Trojan horses (by routing or DNS spoofing) and man-in-the-middle attacks, and the server RSA-authenticates the client machine before accepting *.rhosts* or */etc/hosts.equiv* authentication (to prevent DNS, routing, or IP-spoofing).
viii. Host authentication key distribution can be centrally by the administration, automatically when the first connection is made to a machine.
ix. Any user can create any number of user authentication RSA keys for his/her own use.
x. The server program has its own server RSA key which is automatically regenerated every hour.
xi. An authentication agent, running in the user's laptop or local workstation, can be used to hold the user's RSA authentication keys.
xii. Arbitrary TCP/IP ports can be redirected through the encrypted channel in both directions
xiii. The software can be installed and used (with restricted functionality) even without root privileges.
xiv. Optional compression of all data with gzip (including forwarded X11 and TCP/IP port data), which may result in significant speedups on slow connections.

2.5 SECURE SHELL CAPABILITIES

Secure Shell provides three main capabilities, which provides for many creative secure solutions and these are:

i. Secure Command-shell
ii. Secure File transfer and
iii. Port Forwarding

i. Secure Command Shell

Command shells such as those available in Linux, Unix, Windows, or the familiar DOS prompt provide the ability to execute programs and other commands, usually with character output. A secure command-shell or remote logon allows you to edit files, view the contents of directories and access custom database applications. Systems and network administrators can remotely start batch jobs, start, view or stop services and processes, create user accounts, change permissions to files and directories and more. Anything that can be accomplished at a machine's command prompt can now be done securely from the road or home.

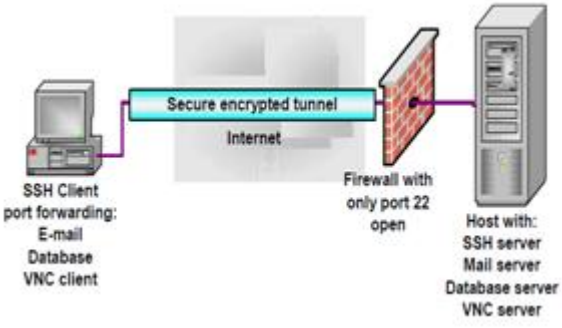

Figure 3:Port forwarding allows multiple TCP/IP applications to share a single secure connection

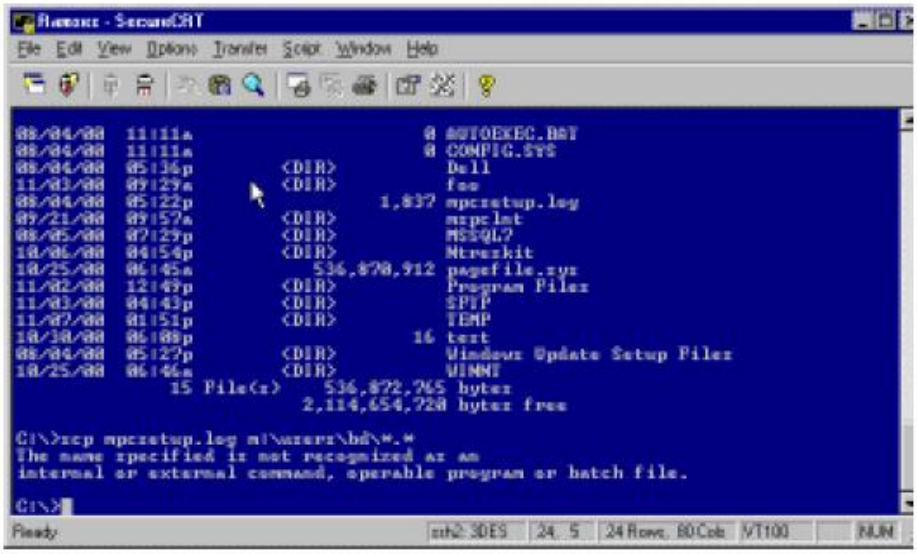

Figure 4: Execute remote commands with the Secure Shell.

ii. Port forwarding

Port forwarding is a powerful tool that can provide security to TCP/IP applications including e-mail, sales and customer contact databases, and in-house applications. Port forwarding, sometimes referred to as tunneling, allows data from normally unsecured TCP/IP applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel, then delivers it to a program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed channel, eliminating the need to open additional vulnerable ports on a firewall or router. For some applications, a secure remote command shell isn't sufficient and graphical remote control is necessary. Secure Shell's port forwarding capabilities can be used to create an encrypted tunnel over which an application can be run. Virtual Network Client, a cross platform GUI remote control application is a good example.

iii. Secure File Transfer

Secure File Transfer Protocol (SFTP) is a subsystem of the Secure Shell protocol. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP has several advantages over non-secure FTP. First, SFTP encrypts both the username/password and the data being transferred. Second, it uses the same port as the Secure Shell server, eliminating the need to open another port on the firewall or router. Using SFTP also avoids the network address translation (NAT) issues that can often be a problem with regular FTP. One valuable use of SFTP is to create a secure extranet or fortify a server or servers outside the firewall accessible by remote personnel and/or partners (sometimes referred to as a DMZ or secure extranet). Using SFTP to create a secure extranet for sharing files and documents with customers and partners balances the need for access with security requirements. Typical uses of a secure extranet include uploading of files and reports, making an archive of data files available for download and providing a secure mechanism for remote administration file oriented tasks. Extranets with business partners have proven to be much more effective for companies than more traditional methods of communication like phone or fax. In fact, SFTP can automate many of these transactions so they take place without human intervention.

A secure extranet is one of the safest ways to make specific data available to customers, partners and remote employees without exposing other critical company information to the public network. Using SFTP on your extranet machines effectively restricts access to authorized users and encrypts usernames, passwords and files sent to or from the DMZ.

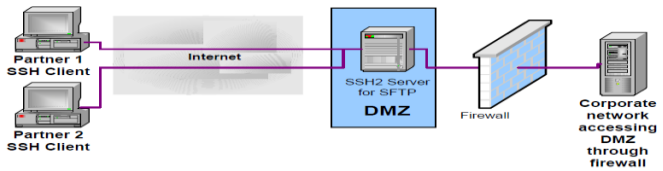

Figure 5: A secure extranet (DMZ) allows secure SFTP access to information assets by partners and internal users

2.6 Security Benefits of Secure Shell

Secure Shell protocol also provides four basic security benefits namely:

i. User Authentication
ii. Host Authentication
iii. Data Encryption and
iv. Data Integrity

i. User Authentication

Secure Shell authentication, encryption and integrity ensure identities and keep data secure

Figure 6: User Authentication

ii. Host Authentication

Authentication, also referred to as user identity, is the means by which a system verifies that access is only given to intended users and denied to anyone else. Many authentication methods are currently used, ranging from familiar typed passwords to more robust security mechanisms. Most Secure Shell implementations include password and public key authentication methods but others (e.g. Kerberos, TLM, and keyboard interactive) are also available. The Secure Shell protocol's flexibility allows new authentication methods to be incorporated into the system as they become available.

a. Password Authentication

Passwords, in combination with a username, are a popular way to tell another computer that you are who you claim to be. If the username and password given at authentication match the username and password stored on a remote system, you are authenticated and allowed access. Some protocols like FTP and Telnet send usernames and passwords as easily visible ASCII text "in the clear", allowing anyone with a snuffer program to easily capture them and then gain access to the system. Secure Shell safeguards against this attack by encrypting all data, including usernames and passwords, before transmission. Although passwords are convenient, requiring no additional configuration or setup for your users, they are inherently vulnerable in that they can be guessed, and anyone who can guess your password can get into your system. Due to these vulnerabilities, it is recommended that you combine or replace password authentication with another method like public key.

b. Public Key Authentication

Public key authentication is one of the most secure methods to authenticate using Secure Shell. Public key authentication uses a pair of computer generated keys – one public and one private. Each key is usually between 1024 and 2048 bits in length, and appears like the sample below. Even though you can see it, it is useless unless you have the corresponding private key:

---- BEGIN SSH2 PUBLIC KEY ----
Subject:
Comment: my public key
AAAAB3NzaC1kc3MAAACBAKoxPsYlv8Nu+fncH2ouLiqkuUNGIJo8iZaHdpDABAvCvLZn
jFPUN+SGPtzP9XtW++2q8khlapMUVJS0OyFWgl0ROZwZDApr2olQK+vNsUC6ZwuUDRPV
fYaqFCHrjzNBHqgmZV9qBtngYD19fGcpaq1xvHgKJFtPeQOPaG3Gt64FAAAAFQCJfkGZ
e3alvQDU8L1AVebTUFi8OwAAAIBk9ZqNG1XQizw4ValQXREczlIN946Te/1pKUZpau3W
iiDAxTFlK8FdE2714pSV3NVkWC4xlQ3x7wa6AUXIhPdLKtiUhTxtctm1epPQS+RZKrRI
XjwKL71EO7UY+b8EOAC2jBNIRtYRy0Kxsp/NQ0YYzJPfn7bqhZvWC7uiC+D+ZwAAAIEA
mx0ZYo5jENA0IinXGpc6pYH18ywZ8CCI2QtPeSGP4OxxOusNdPskqBTe5wHjsZSiQr1g
b7TCmH8Tr50Zx+EJ/XGBU4XoWBJDifP/6Bwryejo3wwjh9d4gchaoZNvIXuHTCYLNPFo

RKPx3cBXHJZ27khllsjzta53BxLppfk6TtQ=
---- END SSH2 PUBLIC KEY ----

Public-private keys are typically generated using a key generation utility. Both keys in the pair are generated at the same time and, while the two are related, a private key cannot be computed from a corresponding public key. In addition to authentication, keys can also be used to sign data. To access an account on a Secure Shell server, a copy of the client's public key must be uploaded to the server. When the client connects to the server it proves that it has the secret or private counterpart to the public key on that server, and access is granted. The private key never leaves the client machine, and therefore cannot be stolen or guessed like a password can. Usually the private key has a "passphrase" associated with it, so even if the private key is stolen, the attacker must still guess the passphrase in order to gain access. Public key authentication does not trust any information from a client or allow any access until the client can prove it has the "secret" private key.

c. Agent and Agent Forwarding
Secure Shell Agent is a way to authenticate to multiple Secure Shell servers that recognize your public key without having to re-type your pass phrase each time. Additionally, by turning on agent forwarding, you can connect to a network of Secure Shell servers, eliminating the need to compromise the integrity of your private key.

Figure7: Agent Forwarding passes authentication from the first SSH connection to the next, re-authenticating each time.

Notice that the private key only has to exist on the original SSHclient machine and the passphrase only needs to be typed when SSHClient connects to SSHServerA. Without agent forwarding enabled, each Secure Shell machine in the chain (except the last) would have to store a copy of the private key. SSHServerA, when authenticating SSHClient to SSHServerB becomes, in essence, a client and would require a private key to complete the authentication process. Agent support eliminates the need for the passphrase to be typed for each connection in the sequence.

ii. Host Authentication
A host key is used by a server to prove its identity to a client and by a client to verify a "known" host. Host keys are described as persistent (they are changed infrequently) and are asymmetric—much like the public/private key pairs discussed above in the Public key section. If a machine is running only one SSH server, a single host key serves to identify both the machine and the server. If a machine is running multiple SSH servers, it may either have multiple host keys or use a single key for multiple servers. Host authentication guards against the *Man-in-the-Middle attack*

iii. Data Encryption
Encryption, sometimes referred to as privacy, means that your data is protected from disclosure to a would-be attacker "sniffing" or *eavesdropping* on the wire. Ciphers are the mechanism by which Secure Shell encrypts and decrypts data being sent over the wire. A block cipher is the most common form of symmetric key algorithms (e.g. DES, 3DES, Blowfish, AES, and Twofish). These operate on a fixed size block of data, use a single, secret, shared key, and generally involve multiple rounds of simple, non-linear functions. The data at this point is "encrypted" and cannot be reversed without the shared key. When a client establishes a connection with a Secure Shell server, they must agree which cipher they will use to encrypt and decrypt data. The server generally presents a list of the ciphers it supports, and the client then selects the first cipher in its list that matches one in the server's list. Session keys are the "shared keys" described above and are randomly generated by both the client and the server during establishment of a connection. Both the client and host use the same session key to encrypt and decrypt data although a different key is used for the send and receive channels. Session keys are generated after host authentication is successfully performed but before user authentication so that usernames and passwords can be sent encrypted. These keys may be replaced at regular intervals (e.g., every one to two hours) during the session and are destroyed at its conclusion.

iv. Data Integrity
Data integrity guarantees that data sent from one end of a transaction arrives unaltered at the other end. Even with Secure Shell encryption, the data being sent over the network could still be vulnerable to someone inserting unwanted data into

the data stream. Secure Shell version 2 (SSH2) uses Message Authentication Code (MAC) algorithms to greatly improve upon the original Secure Shell's (SSH1) simple 32-bit CRC data integrity checking method.

3. METHODOLOGY

A detailed review of exiting literature in this area would be carried out. The review will highlight the fundamental concept of timing analysis of keystrokes and timing attacks on Secure Shell(s). Various assumptions will be deployed to model attack system that can handle timing analysis of keystrokes and timing attacks problem. Timing Attacks is a side channel attack in which the attacker attempts to compromise a cryptosystem by analyzing the time taken to execute cryptographic algorithms. The design of the generic framework would adopt the work of Noack (2007) as the basis for defining timing analysis of keystrokes and timing attacks model.

According to Noack (2007), to make keystroke timing attacks possible, it is reasonable to assume the scenario shown in figure 1 because suppose a password is transmitted in interactive mode. The nested SSH attack uses three hosts, one client and two SSH-servers. At first, the client connects to server#1. Successfully logged in on server #1, the client connects from there to server #2. Because the password typed by the client is sent as one block from server #1 to server #2, it is not reasonable to eavesdrop the packets sent between both servers. No keystroke timing information will be revealed. Therefore, it is better to eavesdrop the connection between client and server #1 because this link leaks all keystroke timing information for **the typed password.**

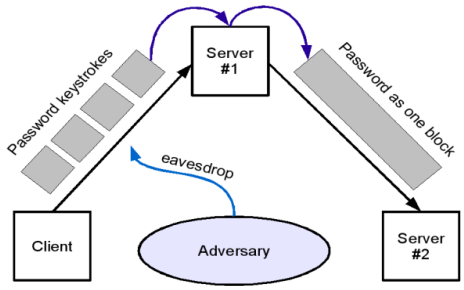

Figure8: The nested SSH attack

The **Instance Based Learning Model** will be adopted in this research work. 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.

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 x_o, x_1……$x_n$is a process that goes through n states, q_1, q_2….. q_n. We denote $y_n$the observed latency of state q_n
To model the process as IBL, we have to make two assumptions:

i. The characters are uniformly distributed so that the probability of transition from the current state to another state depends only of the current state. This assumption usually holds for "good" passwords.

ii. The probability distribution of the latency is dependent only on the current state. This does not always hold because by typing a character the next character typing latency can be affected.

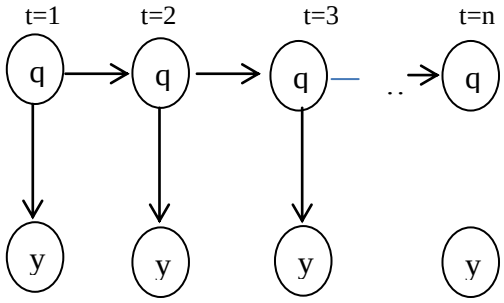

Figure 9 State diagram for keystrokes analysis

The model was developed as follows:

$X=\{x_1, x_2, \ldots, x_n\}$ (1.1)

Where X is a characters typed by the user.

$q = \{q_1, q_2, \ldots, q_n\}$ (1.2)

Where q is a sequence of character pairs from X

$y = \{y_1, y_2, \ldots, y_n\}$ (1.3)

Where y is a sequence of latencies of q and

$q_1 = x_1 \quad x$ (1.4)

$q_2 = x_3 \quad x_2$ (1.5) $q_n = x_n \quad x_{n-1}$ (1.6)

$y_1 \quad q_1$ (1.7)

$y_2 \quad q_2$ (1.8)

$y_3 \quad q_3$ (1.9)

$y_n \quad q_n$ (2.0)

Therefore ,

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

$X = \{x_i = fr(q|y)_i\}$ (2.1)

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

$\overrightarrow{x_i} = \ < a_{i,\ldots,} a_k >$ (2.2)

where $\overrightarrow{x_i}$ represent a set of data containing the latency for every charcter pairs

There is a function that map X onto some set Y

where Y represent the user(s))

$f: X \rightarrow Y$ (2.3)

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

$D = \{< \overrightarrow{X_1}, f(\overrightarrow{X_1}) >, \ldots < \overrightarrow{X_m}, f(\overrightarrow{X_m}) >\}$ (2.4)

This data forms the contents of the database then query point $\overrightarrow{X_q}$, was sent to the application to predict $f(\overrightarrow{X_q})$

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

$\overrightarrow{X_{nn}} = \text{argmin}\left(d\left(\overrightarrow{X}, \overrightarrow{X_q}\right)\right)$ (2.5)

Where:d is the distance function and $\overrightarrow{X_{nn}}$ is the list of possible user(s)

we assign the nearest neighbour's output to the query

$h(\overrightarrow{X_q}) = f(\overrightarrow{X_{nn}})$ (2.6)

we find the highest point (this find the highest numbers of latencies matches the query latency)

$\overrightarrow{X_{nn}} = \text{argmax}\ (d(\overrightarrow{X}, \overrightarrow{X_q}))$ (2.7)

we give query its value as Y=$f(\overline{X_q}) = f(\overline{X_{nn}})$ (2.8)

3.1 Timing Attack Model Setup

The timing attack model consists of:

i. **Network sniffer Module**: 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.

ii.Round-Trip-Times (RTT) Module: This module measure round-trip-time to both hosts**.**

iii. Information analysis Module. This module is also known as "the Parser": The module processes the snuffer'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 forward to the instance based learning algorithms. The Information analysis module or parser filter out password keystrokes.

iv. Instance based learning: 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 they should be activated manually in the correct order for the full attack. The module calculates possible password candidates.

v. Keystroke timing characteristics Module specify the characteristics of a special user general timing characteristics.

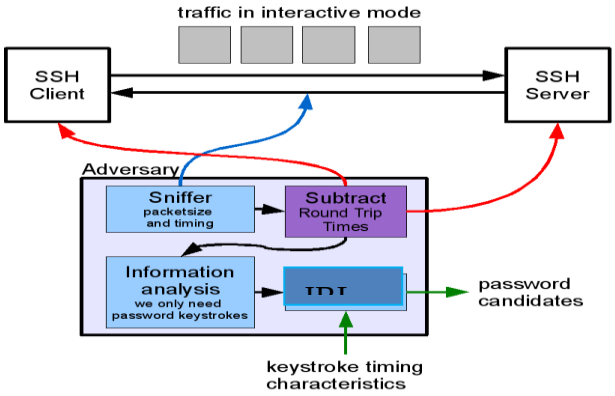

Figure 10: A Keystroke timing characteristics Module

In order to determine precise keystroke timing information over a real network, we developed a **Web Application Software** (a simple client and server programs). When the server is run on a machine, it waits for connections from the client. When the client is run from another machine, it establishes a connection with the server socket. (By specifying a pre-processor directive in the code, we can set the programs to use SSL or simple unencrypted sockets.) Then, whenever a key is pressed on the client, that key is sent to the server, along with the time difference since the last key was pressed. When the server receives one of these packets, it prints the key, the received time difference, and the time difference since the last packet it received from the client. This allows us to observe the timing difference between adjacent keystrokes, both on the client side, where the timing can be measured without network delay, and on the server side, where the time difference is a function of both the actual keystroke timing and the network delay. In order to see how network traffic influences our ability to observe keystroke timing parameters, we will perform this experiment several times, using different machines.

Tools for system implementation will include Java Script, Js/Flash, wire shak and SPSS package.

4. SYSTEM DESIGN AND ANALYSIS

4.1 Keystroke Analysis Attack

Users and system administrators commonly use highly insecure networking applications such as telnet,rlogin, or ftp, which simply pass all confidential information, including users' passwords, in the
clear over the network. 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 to collect all communicated information (Dawn et al,).

Lately, everybody is now aware of this issue and has taken countermeasures. To curb eavesdroppers, security researchers 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. SSH is a protocol for secure network transmission. It is intended for replacing the unsecured protocols like Telnet,rlogin,rsh,ftp etc. The protocol uses cryptographic authentication of user and server, automatic session encryption and integrity protection for the transmitted data. There are two different protocols: SSH1 andSSH2, which differ in both their architecture and the cryptographic algorithms they use. Although using state of the art encryption algorithms, SSH has two weaknesses: First, packets are padded to eight-byte boundary, which helps inferring on short passwords. Second, in interactive mode, a packet is sent on every keystroke, which leaks the inter keystroke timing (lustig et al, 2001).This proposed system model aim to construct a secure transmission medium between a web server and a client and monitors and monitors the keystroke of the client and sends the information to the server for analysis. The objective of the system is:

(a) Design of an instance based learning technique for the timing analysis of keystroke and timing attack on secure shell to reveal information per keystroke pair
(b) Develop a software to stimulate the model designed in (a) above

Figure 11 below show eavesdrop attack which represents the system design. Assume the user has already established a SSH session between the local host *A* and remote host *B*. Then the user wants to open another SSH session from *B* to another remote host *C* as shown in Figure 3.1. In this case, the user's password for *C* is transmitted, one keystroke at a time, across the SSH-encrypted link *A--B* from the user to *B*, even though the SSH client on machine *B* patiently waits for all characters of the password before it sends them all in one packet to host C for authentication (as designed in the SSH protocol . It is easy to identify such a nested SSH connection using techniques. Hence in this case the eavesdropper can easily identify the packets corresponding to the user's password on link *A--B*, and from this learn the length and the inter-keystroke timings of the users' password on host *C*.

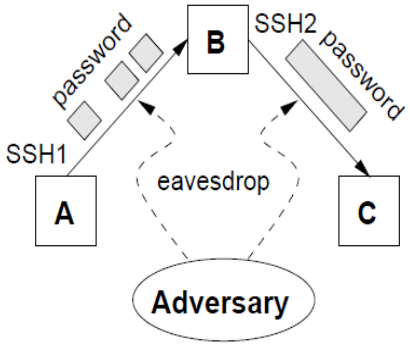

Figure 11: SSH eavesdrop attack

4.2 Statistical Analysis of Inter-keystroke Timings

As a first study towards inferring key sequences from timing information, a technique for statistical analysis of the inter-keystroke timings is developed. First, a description of how training data are collected and display some simple timing characteristics of character pairs. A model of the inter-keystroke timing of a given character pair as a Gaussian distribution, then a description of how to estimate quantitatively the amount of information about the character pair that one can learn using the inter-keystroke timing information.

4.2.1 Data Collection

The two keystrokes of a pair of characters (k_a, k_b)generate four events: the press of k_a, the release of k_a, the press of k_b, and the release of k_b. However, because only key presses (not key releases) trigger packet transmission, an eavesdropper can only learn timing information about the key-press events. Since the main focus of this system is in the scenario where an adversary learns timing information on keystrokes by simply monitoring the network, the focus will only be on key-press events. The time difference between two key presses is called the *latency* between the two keystrokes. The term *inter-keystroke timing is used* to refer to the latency between two keystrokes. In order to characterize how much information is leaked by inter-keystroke timings, because passwords are probably the most sensitive data that a user will ever type, the focus will only be on information revealed about passwords (rather than other forms of interactive traffic). The focus on passwords creates many challenges. Passwords are entered very differently from other text: passwords are typed frequently enough that, for many users, the keystroke pattern is memorized and often typed almost without conscious thought. Furthermore, well-chosen passwords should be random and have little or no structure (for instance, they should not be based on dictionary words). As a consequence, naive measurements of keystroke timings will not be representative of how users type passwords unless great care is taken in the design of the experimental methodology. This system methodology is carefully designed to address these issues. Due to security and privacy considerations, the system will not gather data on real passwords; therefore, a data collection procedure intended by the system is to mimic

how users type real passwords. A conservative method is to pick a random password for the user (where each character of the password is chosen uniformly at random from a set of 10 letter keys and 5 number keys, independently of all other characters in the password), have the user practice typing this password many times without collecting any measurements, and then measure inter-keystroke timing information on this password once the user has had a chance to practice it at length. However, when the goal is to try to identify potentially relevant timing properties (rather than verify conjectured properties), this conservative approach is inefficient. In particular, users typically type passwords in groups of 3–4 characters, with fairly long pauses between each group. This distorts the digraph statistics for the pair of characters that spans the group boundary and artificially inflates the variance of the measurements. As a result there would be need to collect a great deal of data for many random passwords before this effect would average out. In addition, it takes quite a while for users to become familiar with long random passwords. This makes the conservative approach a rather blunt tool for understanding inter-keystroke statistics.

4.3 Keystrokes and Keystrokes Analysis

A keystroke is typing one character on a keyboard (not stroking your keyboard like a cat). Every time you hit a key, you perform a keystroke. Therefore, 5400 keystrokes in one hour means hitting 5400 keys in one hour, or 90 keys a minute (5400 keystrokes/60 minutes).

Sometimes keystrokes per hour (KSPH) or keystrokes per minute (KSPM) are used to measure typing speed instead of words per minute (WPM). Keystrokes include Layout Kitchen, which enables users to design their own virtual keyboards. Such virtual keyboards can be used not only for typing, but also to launch applications, speak, run AppleScripts and much more.

Keystrokes is a full-function, advanced on-screen keyboard providing people with physical impairments as well as graphic tablet and touch screen users full access to the computer. Keystrokes enables you to use a mouse, trackball, head pointer or other mouse emulator to type characters into any standard Operating System application.

4.3.1 Keystroke Dynamics or Typing Dynamics

The concept underlying keystroke analysis (also known as keystroke dynamics) is the ability of a system to recognize patterns, such as characteristic rhythms, during keyboard interactions.

Keystroke dynamics, or **typing dynamics**, is the detailed timing information that describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard The behavioral biometric of Keystroke Dynamics uses the manner and rhythm in which an individual types characters on a keyboard or keypad. The keystroke rhythms of a user are measured to develop a unique biometric template of the users typing pattern for future authentication. Raw measurements available from most every keyboard can be recorded to determine Dwell time (the time a key pressed) and Flight time (the time between “key up” and the next “key down”). The recorded keystroke timing data is then processed through a unique neural algorithm, which determines a primary pattern for future comparison. Similarly, vibration information may be used to create a pattern for future use in both identification and authentication tasks.

Data needed to analyze keystroke dynamics is obtained by keystroke logging. Normally, all that is retained when logging a typing session is the sequence of characters corresponding to the order in which keys were pressed and timing information is discarded. When reading email, the receiver cannot tell from reading the phrase "I saw 3 zebras!" whether that was typed rapidly or slowly:

- the sender used the left shift key, the right shift key, or the caps-lock key to make the "i" turn into a capitalized letter "I"
- the letters were all typed at the same pace, or if there was a long pause before the letter "z" or the numeral "3" while you were looking for that letter
- the sender typed any letters wrong initially and then went back and corrected them, or if he got them right the first time.

4.3.2 Secure Shell Weakness

There are two important weaknesses in Secure Shell (SSH) that help to mount timing attacks:

i. The transmitted characters are padded to the symmetric ciphers block length, which is 8 Byte in general(tracking short passwords)

ii. In interactive mode, every keystroke is sent immediately to the remote host (Keystroke timing leaks information)

This allows the attacker to approximate the transmitted data’s length to an eight byte boundary. Besides that, the inter keystroke timings can be extracted from the SSH data stream. The interactive mode is activated after the initial authentication. That means that the initially transmitted password will not leak timing information.

To make keystroke timing attacks possible, it is reasonable to assume the scenario showing in figure 11because we can suppose that a password is transmitted in interactive mode. The nested SSH attack

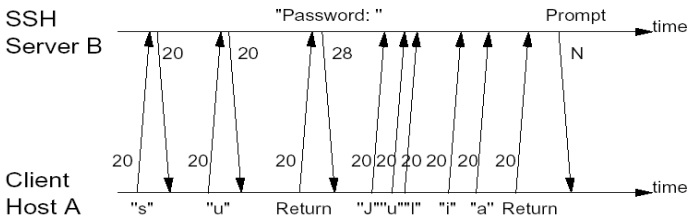

Fig 11: Secure Shell Weakness

uses three hosts, one client and two SSH-servers. At first, the client connects to server #1. Successfully logged in on server #1, the client connects from there to server #2. Because the password typed by the client is sent as one block from server #1 to server #2, it is not reasonable to eavesdrop the packets sent between both servers. No keystroke timing information will be revealed. It is better to eavesdrop the connection between client and server #1, while the user types the password for server #2, because this link leaks all keystroke timing information for the typed password.

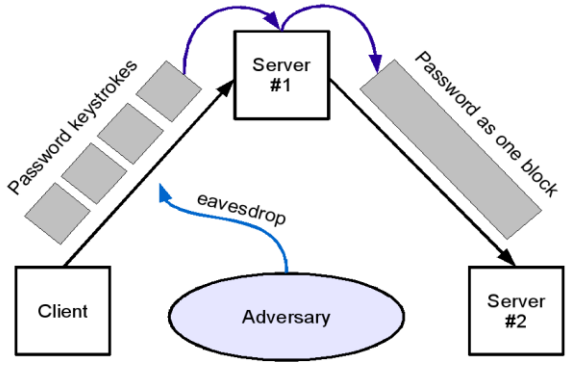

Fig 12: The nested SSH attack

4.3. 3 Practical Timing Attacks on Passwords

A keystroke attack will be effective only when a short sequence of keystroke, which is important, can be inferred from the timing. Passwords qualify to this description. Song et al (2001) suggests two practical attacks to capture keystroke timings of a user's password.

Attack Model

At first we define some assumptions, we use throughout this work.

i. The considered users are familiar with keyboard typing but do not uses touch typing at all.

ii. The statistics are based on random text with an average length of 1000 cha99racters, which is chosen by the test person itself. The only demand on the typed text was that the test persons had to type meaningful words and should not repeat them more than ten times.

iii. The network latency may be bigger than the inter keystroke timings.

iv. The network latency is not constant.

v. The instance based leaning model may not be optimal to gain information.

vi. The adversary can eavesdrop the users' encrypted communication.

Today, there are many users who did not learn touch typing. So we have decided to generalize the assumption from song et al (2001) and take up the non-touch typing writers in the statistic.
In the just mentioned work, the authors limited their statistic to 142 character pairs. We let the users choose their texts itself, to have a more general statistic. This leads on the one hand to more possible character pairs and on the other hand to more realistic statistics for passwords, since passwords are not often selected fully random.

Furthermore it is realistic to assume that the network latency is not constant and may be greater than inter keystroke timings. As assumed in song et al (2001) and NOAK (2007) the hidden markov model is optimal for analyzing keystrokes. But statistics show that in some cases, it makes a time difference, which key is pressed before an analyzed

key pair. In that case, Instance based learning is used to enable us analyses the keystrokes properly, to realized better performances and efficiency.

4.3.4 Realization of the Attacks

Figure 12: Schematic Illustration of a sniffing scenario

Given a SSH-user and an adversary. The adversary is successful when she revealed the users password. 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 12 shows the dataflow, when the user types a password.

4.4 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 infer the keystroke press in order to determine the password transmitted in a SSH session.

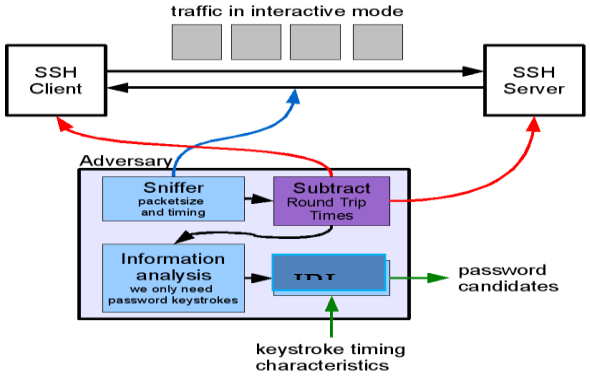

Fig 13. Components of Attack System Schematics

i. **Network sniffer Module**: 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.

ii.Information analysis Module. This module is also known as "the Parser": The module processes the snuffer'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 forward to the instance based learning algorithms. The Information analysis module or parser filter out password keystrokes.

iii. Instance based learning: 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 they should be activated manually in the correct order for the full attack. The module calculates possible password candidates.

iv. Keystroke timing characteristics Module specify the characteristics of a special user general timing characteristics.

v. Round-Trip-Times (RTT) Module: This module measure round-trip-time to both hosts**.**

4.4.2 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.
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 3.4 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.

4.5 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). One advantage that instance-based learning 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.
Gagliardi, 2011 applied this family of classifiers in medical field as second-opinion diagnostic tools and as tools for the knowledge extraction phase in the process of knowledge discovery in databases. One of these classifiers (called Prototype exemplar learning classifier (PEL-C) is able to extract a mixture of abstracted prototypical cases (that are syndromes) and selected atypical clinical cases.

4.5.1 Keystroke Characteristics

By examining the keystrokes characteristics, we can see that different character pairs have different latencies. Gaussian model of the inter keystroke timing is used. Each character pair has a distribution such that

$$P(q|y) = N(\mu q, \delta^2 q) \qquad (3.1)$$

Where q is the character pair, y is the timing measured, and μq and δq are the corresponding normal distribution parameters. This model can be built by gathering statistics, and calculating the mean and variance of the keystroke timing.

4.5.2 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:

i. The characters are uniformly distributed so that the probability of transition from the current state to another state depends only of the current state. This assumption usually holds for ”good” passwords.

The probability distribution of the latency is dependent only on the current state. This does not always hold because by typing a character the next character typing latency can be affected

5. SYSTEM IMPLEMENTATION

Protocol is needed to run some processes in computer network, such as log into one account remotely from another, or transit commands to a remote computer for execution. Various programs exist for these purposes, such as ftp for file transfer, telnet for remote login, and rsh for remote execution of commands. Broadcast-based networks that were commonly used (e.g. Ethernet) allow a malicious user to eavesdrop on the network and to collect all communicated information. Therefore, SSH was designed to replace the insecure application above. This section presents the documentation for the implementation of timing analysis of keystrokes and timing attacks on SSH using instance based learning technique. The software and hardware requirement and settings needed for the system, also the testing of the system functions, as well as the results are show below.

5.1 System Interface

The system designed is a web application hosted on a web server and viewed with a web browser. The web server also hosts the database, serving as a database server that keeps records of every entry and value. The interfaces described below are interface of both the server and client as its being used or configure. The system provides a set of interface and tools for setting up the resources for client request and performance tuning. The client uses a web browser (such as Firefox, internet explorer, safari, or chrome) to access or send request to the server for a particular resources.

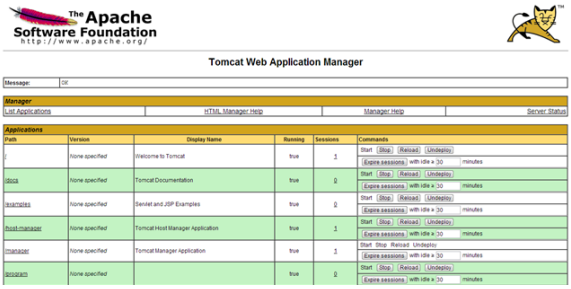

Figure 14 Web Application Manager Page

Figure 15 Web application home page

5.1.1 Keystroke Analysis Page

Clicking on the "key stroke activiy" link on the home page opens this page ("keystroke analysis page").This page 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. Figure 16a and 16b below shows the keystroke analysis page. The "Clear area space" button is used to clear or remove the content of the text area to provide a free space for typing. The "Home" link takes the user back to the web application home page.

Figure16a Keystroke attack analysis page

5.1.2 Performance of keystroke attack Page

Data gathered from the keystroke 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 type character on the text area on the page. Figure 4.11a and 4.11b shows a snap shot of the page. The "Clear area space" button clears the content of the text area and the "Home" links takes their user back to the web application home page

Figure 16b Performance of keystroke attack page

Figure 17 Performance of keystroke attack page

6. CONCLUSION

In this paper, we developed a system to demonstrate an Instance Based Learning Model for timing analysis of keystrokes to perform timing attacks on the Secure Shell protocol.

7. REFERENCES

[1]. Ahmad S. and Christian W. (2004): *Network Security II - Secure Shell*, Ruhr-Universitat Bochum.

[2]. David et al (2007): *Remote Timing Attacks are Practical* Proceedings of the 12th USENIX Security Symposium.

[3]. David et al (2002): *Timing Analysis of Keystrokes and Timing Attacks on SSH**11th USENIX Security Symposium.Edward et al (2005)**:** *Timing Attacks on Web Privacy*

[4]. Michael et al (2001): *Analysis of the Feasibility of Keystroke Timing Attacks over SSH Connections*, Research Project at University of Virginia.

[5]. Michael L. and Yonit S.(2001):*Keystrokes Attack on SSH*, Final Project Report at Technion IIT.Moheeb et al (2005) :*Worm Evolution Tracking via Timing Analysis*, In Proceedings of the 6th ACMConference on Computer and Communications Security Alexandria, Virginia, USA.

[6]. Noack et al(2007):*Timing Analysis of Keystrokes and Timing Attacks on SSHRevisited*, seminar work atthe Chair for Network and Data Security, WS06/07.

[7]. Song et al (2001): *Timing Analysis of Keystrokes and Timing Attacks on SSH*10th USENIX Security Symposium.

[8]. Solar and Song (2001): *Passive Analysis of SSH (Secure Shell)Traffic*,

[9]. www.securiteam.com/securitynews/5KPOOOA3PU.html posted on March 2001. Retrieved on 25, August 2010.

[10]. Trostle (1998):*Timing Attacks against Trusted Path*, IEEE Symposium Proceedings, Oakland, USA.

[11]. Yigael et al (2006):*Dictionary Attacks Using Keyboard Acoustic Emanations.*

[12]. XiaoFeng and Kehuan (2005): *Peeping Tom in the Neighbourhood: "Keystroke Eavesdroppingon Multi-User Systems".*

[13]. Gagliardi, F (2011). "Instance-based classifiers applied to medical databases: Diagnosis and knowledge extraction". *Artificial Intelligence in Medicine* **52** (3): 123-39.doi:10.1016/j.artmed.2011.04.002. http://dx.doi.org/10.1016.artmed.2011.04.002