Wireless Network Security by SSH Tunneling

Aniket Burande, Ankita Pise, Sameer Desai, Yohan Martin, Sejal D'mello

Information Technology Department, Atharva College of Engineering,
Malad (West), Mumbai 400095, India

Abstract- The wireless communication, like Wi-Fi, is insecure and is vulnerable to attacks like packet sniffing. The login credentials in HTTP packets can be extracted from the IEEE 802.11 packets using tools like Wire-shark and can be misused by the attackers. This puts the whole data and the system into risks. The main goal of this paper is to show that packet sniffing can be avoided by 'SSH tunneling'. The concept of SSH tunneling can actually be a good defensive mechanism against the packet sniffing attacks and can also make the communication over wireless networks secured. Also, why SSH tunneling is beneficial has been justified.

Index Terms- IEEE 802.11, Packet sniffing, SSH, Wireshark

I. Introduction

In many corporations, remote access to business applications has become mission critical. Almost all the businesses use internet for their daily tasks. But internet-based remote access does add significant risks. Sensitive data can be captured, modified or reused anywhere between remote workers and the corporate firewalls. Packet sniffing and session hijacking are the two dangerous attacks and are on a constant rise . Due to the easy availability of sophisticated tools, it has become easier to carry out these attacks. If not controlled, these two attacks can bring down entire system.

The objective of this paper is to present SSH tunneling as a way to secure the connection between the client and the server and avoid packet sniffing as well as session hijacking attacks. A Secure Shell (SSH) capability called port forwarding allows non secure TCP/IP data to be tunneled across public and private networks through a connection that is secured and encrypted.

This paper is organized as follows: In section II, we describe in great detail about the packet sniffing attack and how it can be done using different tools like Wireshark. In section III, we explain the concept of SSH tunneling and explore how SSH is configured and how it works. Later, in section IV, concluding remarks are presented, followed by a list of references**.**

II. Packet Sniffing

Packet sniffing, defined as packet or protocol analysis, describes the process of capturing and intercepting live data as it flows across a network in order to get a summary of what is happening on the network. Packet sniffer, a tool used to capture the raw data going across the network, is basically used to perform packet analysis or packet sniffing. Packet analysis can help us to define network characteristics, learn who has been using the network, determine the bandwidth and its usage, recognize peak network usage times, detect various possible attacks or harmful activities, and find unsecured and abnormal applications. There are different types of packet sniffing programs. Each program is designed or built with different goals and objectives in mind. Some names of the popular packet analysis programs are tcpdump (a command-line program), OmniPeek, and Wireshark (both GUI-based sniffers).

A packet sniffer intercepts and log packets in a network which can be decoded later according to some specification. Wireless packet sniffers are widely used in network management and research communities due to their capability to monitor network traffic at the MAC layer as well as layers above. Particularly, packet sniffers are widely used as tools to diagnose or discover network problems and track a particular activity, but they can also be misused to intercept important or private information such as username and passwords.

There are four main mode of operation in IEEE 802.11 Network Interface Cards (NICs). First one is named as master mode (Access Point), second one is client mode (Station), third one is ad-hoc mode and last one is monitor mode. In addition, the NIC can be set to operate in promiscuous mode. Here, the filtering of all the packets received based on the MAC address is disabled and this mode allows the NIC to process all the packets that it senses in the network. A NIC operated in promiscuous mode is to be associated to an Access Point (AP) or joined in an ad hoc network in order to sniff the traffic of the wireless network. On the other hand, the NIC can listen on a particular channel in monitor mode that allows sniffing of packets using a packet sniffer such as Wireshark in an entirely passive manner without having to join the wireless network and being unable to transmit packets.

Wireshark has sophisticated wireless protocol analysis support to help administrators troubleshoot wireless networks. With the appropriate driver support, the traffic can be captured by Wireshark, "from the air" and decode it into format that helps administrators to track down factors that are causing poor performance, irregular or weak connectivity, and few other common problems. In order to capture packets, Wireshark uses application programming Interface (APIs) for capturing network traffic known as pcap. Live packet data can be captured from a variety of interfaces such as IEEE 802.11, Ethernet and loopback interface to name a few. Through Wireshark GUI, captured data can be displayed along with its detail protocol information, captured data can be filtered, and statistical graphs for input/output can be generated. Further, packets can be tracked easily using the GUI. Packet filtering employs primitive expressions joined with and/or primitives allowing filtering to be done on many fields such as IP address, host name or Ethernet host address, TCP and UDP port numbers.

During packet sniffing, the information inside a network packet on a TCP/IP network undergoes capturing, decoding, inspection and interpreting. The main purpose is to sniff and steal information, particularly credit card numbers, user IDs, port numbers, passwords, network details, etc. Sniffing is commonly known as a "passive" type of attack, wherein the attackers can be active but silent/invisible on the network. This makes it difficult to identify and diagnose, and hence it is a very dangerous type of attack. The TCP/IP packet contains vital information required for two network interfaces to communicate with each other.

It contains fields such as source and destination IP addresses ports, sequence numbers and the protocol type. By its very nature, the TCP/IP protocol only ensures that a packet is constructed, placed on an Ethernet packet frame, and reliably sent from the sender to the receiver across networks. However, by default, the mechanisms to ensure data security do not exist. Thus, it becomes the responsibility of the network layers on the upper side to ensure that information in the packet is not to be tampered with.

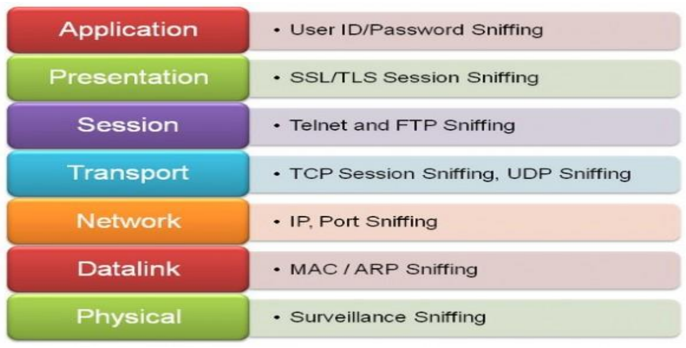

Figure 1: Sniffing at different ISO/OSI levels

III. SSH TUNNELING

A. Basic

Secure Shell (SSH) tunneling is a method that creates a virtual tunnel over the wireless network, with client and server being the endpoints of the tunnel. The virtual tunnel can be imagined of as an encrypted path that carries the traffic between the client and the server. Any traffic that flows through this tunnel gets encrypted by default.

Whenever any client wants to communicate with the server, he first initializes the virtual tunnel. The establishment of the tunnel is done with the help of the concept called as port forwarding which is explained in the following section of web browser.

There are four entities in the working of SSH tunneling as shown in figure 2. They are as follows:

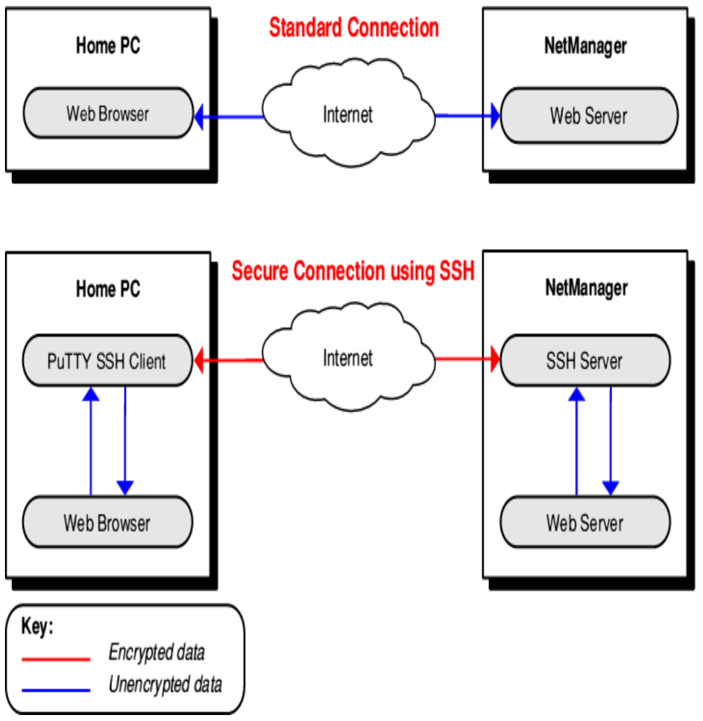

Figure 2: Network flow diagram with & without SSH tunneling

1)Web browser (Client side)
2)SSH client (Client side)
3)SSH server (Server side)
4)Web server (Server side)

1) Web Browser: This is the actual client that needs to communicate to the server. The client uses the web browser to access the server. SSH tunneling can be done using any standard browser, i.e. it is browser independent.

In the web browser the user needs to change the settings to direct the traffic to proxy server. This concept is called as port forwarding. The address of the proxy server is that of SSH client (which is explained in the next section). This configuration is important as it now forces the traffic to flow from web browser to SSH client rather than from web browser to web server.

2) SSH Client: The SSH client takes the traffic from the web browser. It uses encryption algorithms to encrypt the traffic. Any good encryption algorithm can be used for this purpose.

After the data is ready to be sent, the SSH tunnel is established. The SSH client uniquely identifies a particular SSH server. Hence, the traffic sent from the SSH client can be only taken in by the corresponding SSH server that SSH client has identified.

After the tunnel is established, the traffic is sent over the internet but through the tunnel, to the SSH server.

3) SSH Server: The SSH server, as the name suggests, resides on the server side. It is situated at the other end of the virtual tunnel. It takes the traffic that came in through this tunnel. The SSH server uses decryption algorithms to change the data back to its original form. The decryption algorithms used are in accordance

with the algorithms used by the SSH client. Once the traffic is decrypted to its normal form it is sent to the Web server.

4) Web Server: The web server receives the request through the SSH server and does it's processing. The Web server is completely agnostic with respect to the traffic flow to and from it. After the response to the request is ready, it is sent back to SSH server, which then sends it over to SSH client and so on.

The communication between Web browser to SSH client and SSH server to Web server is out of the tunnel. Hence, essentially the tunnel exists only between the SSH client and the SSH server, as that is the only path on the internet.

The beauty of the SSH tunneling is that, once a tunnel is created between the client and server, it stays active throughout the session. Hence, the tunnel once established for a session, can be used to transport any amount of data to and fro without being concerned about the possibility of attacks. This also implies that the overhead of establishing the tunnel has to be done only once.

B. SSH Architecture

IETF RFCs 4251 through 4256 defines SSH as the "Secure Shell Protocol for remote login and other secure network services over an insecure network." The shell is made up of three important elements:

- Transport Layer Protocol: This protocol accommodates server authentication, reliability, privacy, and integrity along with perfect forward privacy. This layer can also provide optional compression and can be run over a TCP/IP connection but can also be used on top of any other dependable data stream.
- User Authentication Protocol: This protocol helps in authentication of the client to the server and runs over the transport layer.
- Connection Protocol: This protocol helps in multiplexing of the encrypted tunnel to numerous logical channels, running over the User Authentication Protocol.

C. Configuration

The configuration setup of SSH tunneling can be understood very well with the help of an example.

Let us say we want to browse yahoo.com in a secure way using SSH tunneling.

To create the SSH tunnel execute the following code from the SSH client:

```
ssh -L 9001:yahoo.com:80 (SSH server port)
```

The 'L' switch indicates that a local port forward is needed to be created. The switch syntax is as given below:

```
ssh -L <local-port-to-listen><remote-host>:<remote-port>
```

Now the SSH client will connect to SSH server (usually running at port 22) binding port 9001 of SSH client to listen for local requests thus creating a SSH tunnel between SSH client and SSH server. At the SSH server end it will create a connection to 'yahoo.com' at port 80. So SSH client doesn't need to know how to connect to yahoo.com. Only SSH server needs to worry about that. The channel between SSH client and SSH server will be encrypted while the connection between SSH server and 'yahoo.com' will be unencrypted.

Now it is possible to browse yahoo.com if we follow the link: http://localhost:9001 in the web browser at SSH client computer. The SSH server computer will act as a gateway which would accept requests from SSH client machine and fetch data and tunnel it back.

.

D. Advantages of SSH

There are many advantages of SSH tunneling and hence are divided broadly into 3 categories as follows:
1) Four security benefits.
2) Advantage when used over "clear text" protocols.
3) General advantages.

1) Four security benefits: SSH tunneling provides the following four basic security benefits:

i) User Authentication
ii) Host Authentication
iii) Data Encryption
iv) Data Integrity

i) 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. Secure Shell implementations include password and public key authentication methods but others (e.g. kerberos, NTLM, 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.

ii) Data Encryption, refers to encrypting the data to avoid it from attacks, using encryption algorithms. The SSH tunneling can use any encryption algorithms or standards.Various types of encryption standards are available, ranging from as low as 512-bit encryption to as high as 32768 bits, including popular ciphers like Blowfish, Advanced Encryption Scheme (AES), Triple DES, CAST-128 and ARCFOUR.

iii) Data integrity means that the data sent from the client is received as it is by the server without any changes in it. The tunneling takes care that the integrity of the data is maintained when the traffic flows through the tunnel and is available is the suitable form to the web server.

The above four advantages are directly available when the tunneling is used. Hence, it is better suited over normal method of only encryption, which doesn't provide the mechanism for authentications and data integrity.

2) Advantage when used over "clear text" protocols: Figure 3 shows how easily a telnet session can be casually viewed by anyone on the network using a network-sniffing application such as Wireshark.

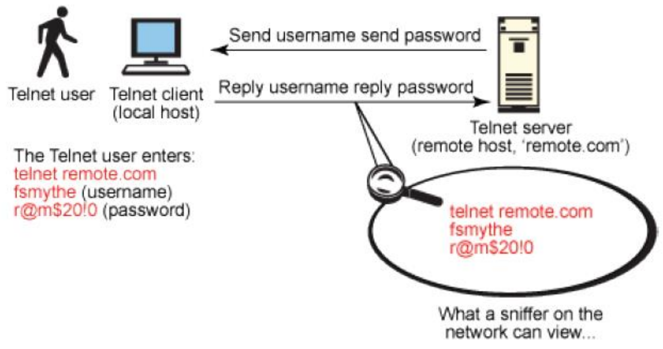
Send username send password
Reply username reply password
Telnet user
Telnet client (local host)
Telnet server (remote host, 'remote.com')
The Telnet user enters:
telnet remote.com
fsmythe (username)
r@m$20!0 (password)
telnet remote.com
fsmythe
r@m$20!0
What a sniffer on the network can view...

Figure 3: Telnet protocol sessions are unencrypted

When using an unsecured, "clear text" protocol such as telnet, anyone on the network can steal your passwords and other sensitive information. Figure 3 shows user *fsmythe* logging in to a remote host through a telnet connection. The user name he enters is *fsmythe* and password is *r@m$20!0*, which both can be viewed by any other user on the same network as our hapless and unsuspecting telnet user.

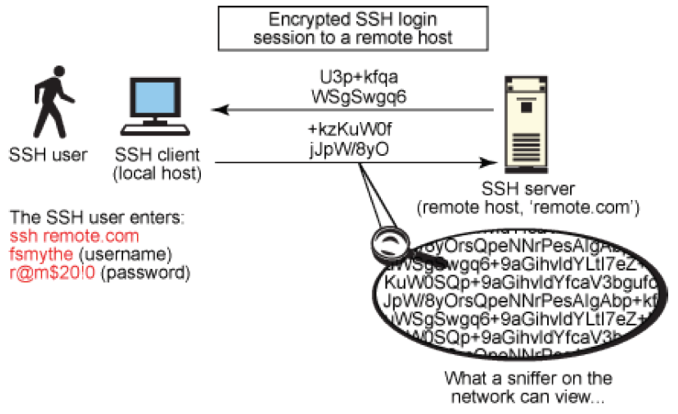

Figure 4: SSH protocol sessions are encrypted

Figure 4 provides an overview of a typical SSH session and shows how the encrypted protocol cannot be viewed by any other user on the same network segment.

3) General Advantages:

a) The SSH tunneling provides complete protection from packet sniffing attack.

b) SSH tunneling also secures session cookies, hence preventing session hijacking attacks.

c) SSH tunneling has the capacity to hold 16 connections at a time and has minimum delays.

IV. CONCLUSION

SSH tunneling is a simple & effective solution for secure content delivery over internet. This solution will effectively secure the internet browsing from packet sniffing. As compared to the other link, network, and application security measures like WEP, IPsec and PGP, along with their installing and configuring, Secure Shell is relatively secure, reliable, quick and easy. By deploying Secure Shell, companies create a comprehensive general-purpose tunneling platform that can be used to implement a wide variety of security policies, ensuring the privacy, authenticity, authorization and integrity of many different applications. This paper illustrates the basics and the implementation details of SSH Tunneling.

REFERENCES

[1] Chris Sanders and Chris Sanders, "Practical Packet Analysis", May 17, 2007 ISBN-13: 978-1-59327-149-7.
[2] "Wireless Sniffing with Wireshark, 2006" Tutorial, Available: http://www.willhackforsushi.com/books/377_eth_2e_06.pdf
[3] ANSI/IEEE, "802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2000.
[4] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance Anomaly of 802.11b", Proc. IEEE INFOCOM, April, 2003, pp. 836-843.
[5] Cisco Access Point: Available at the website: http://www.cisco.com/
[6] Md. Kamrul Hasan, A.H.M. Amimul Ahsan, and M. Mostafizur Rahman, "IEEE 802.11b Packet Analysis to Improve Network Performance" JU Journal of Information Technology (JIT), Vol. 1, June, 2012.
[7] Roger Hill, "Getting started with SSH security and configuration" A hands-on guide, February 1, 2011. Copyright: IBM Corporation, 2011. (www.ibm.com/legal/copytrade.shtml)
[8] Brian Wippich, "Detecting and Preventing Unauthorized Outbound Traffic", GCIH Gold Certification, October 15th, 2007. Copyright: SANS Institute.
[9] Wolf-Bastian P¨ottner and Lars Wolf, "IEEE 802.15.4 packet analysis with Wireshark and off-the-shelf hardware", June, 2010.
[10] C. Lonvick, 'Ed Cisco Systems Inc.', "RFC 4251 - The Secure Shell (SSH) Protocol Architecture", SSH Communications Security Corp, January, 2006, Copyright: The Internet Society.
[11] Sung Jun Ban, Hyeonwoo Cho, Chang Woo Lee, and Sang Woo Kim, "Implementation of IEEE 802.15.4 Packet Analyzer", International Journal of Electrical and Electronics Engineering, May, 2008.
[12] Buddhika Chamith, "SSH Tunnelling Explained", March 21, 2012. Available at the website.

AUTHORS

First Author – Aniket Burande, Student, Atharva College of Engineering. Email id: aniketburande@yahoo.in
Second Author – Ankita Pise, Student, Atharva College of Engineering. Email id: ankita.pise@gmail.com
Third Author – Sameer Desai, Student, Atharva College of Engineering. Email id: gdsameer@yahoo.com
Fourth Author – Yohan Martin, Student, Atharva College of Engineering. Email id: yohanmartin3@gmail.com
Fifth Author – Sejal D'mello, Lecturer, Atharva College of Engineering. Email id: dmello.sejal@gmail.com