DETECTION OF SESSION HIJACKING

A Thesis submitted at the University of Bedfordshire

In partial fulfilment for the degree of

Masters of Science

In

Computer Security and Forensics

Jerry Louis

1031051

Supervisor: Dr. Xiaohua Feng

Department of Computer Science and Technology

University of Bedfordshire

January 2011
Acknowledgments

I would like to express my gratitude and special thanks to my supervisor “Dr. Xiaohua Feng” for her continued support, invaluable assistance and supervision towards the completion of my thesis.

I would also like to thank the MSc in Computer Security and Forensics & CATS Faculty for their continued support throughout my course.

Finally I would like to thank my family for their valuable support towards my studies.

You’re sincerely,
Jerry Louis
Abstract

In today’s world the computer networks have become vulnerable to numerous types of attacks. Either it is wireless network or wired network, one of the most common or effective attacks up-to-date are man-in-the-middle attack, within which session hijacking has been the most attempted attack. The success rate of a session hijacking attack is significantly higher when compared to other attacks. This paper analysis the underlying problem in the detecting strategy and provides a customized solution to detect session hijacking efficiently. This thesis proposes a dual strategy towards developing a defensive mechanism against the session hijacking attempts, the two strategies are IN-Network strategy and OUT-Network Strategy. The implementation to test the above mentioned strategies are carefully designed in order to get the optimum results. The implementation is tested on the user-end, attacker-end and finally on the server end, also this testing of the proposal is performed on two platforms, Windows Operating system – Windows 7 and Linux Operating System – BackTrack. The performance in each of the operating system and the network strategies are carefully iterated to discuss the performance of each of the parameters in its host operating system. The results finally show that the defensive methodology is successfully implemented on both the network strategies.
# Table of Contents

1. Introduction ................................................................................................................. 1
   1.1 Background ............................................................................................................. 2
   1.2 Aims & Objectives ............................................................................................... 3
   1.3 Thesis Layout ........................................................................................................ 4

2. Literature Review .......................................................................................................... 5
   2.1 Computer Security in general ............................................................................... 6
   2.2 Session Hijacking ................................................................................................. 11
   2.3 Detection tools and techniques for Session Hijacking ........................................ 15
   2.4 Recent Developments on Session Hijacking ....................................................... 16

3. Session Hijacking detection tools & Implementations ............................................... 18
   3.1 User System .......................................................................................................... 22
   3.2 Server System ....................................................................................................... 22
   3.3 Attacker System .................................................................................................... 22

4. Survey analysis ............................................................................................................ 23

5. System Design ............................................................................................................. 24

6. Flow Chart & Pseudo code ........................................................................................ 25
   6.1 IN-Network Strategy flowchart ............................................................................ 28
      6.1.1 IN-Network Strategy Pseudo Code ................................................................. 26
   6.2 OUT-Network Strategy flowchart ......................................................................... 27
      6.2.1 OUT-Network Strategy Pseudo Code ............................................................... 29
1. INTRODUCTION

In this age of communication everything happens over the internet, from businesses to shopping, from banking to education, internet has become an integral part of every human being. As the internet grows larger and larger the security threat has grown even more strongly. It is one thing to setup a strong e-business but totally a different issue in order to protect the information that’s on the internet; security plays a major role in every aspect of communication or transaction over any network. Every user out there needs to be assured that their information, money, transaction and communication are safe and trustable with the related network they are engaged in. The more accessibility and comfort is brought into the internet such is the extent of vulnerability on the resource management. So it has become a tedious task for every system administrator and computer security professional to be competent enough of to develop various defensive mechanisms to handle uncertain attacks and protect the user information. One of the most important tasks will be to keep the system up-to-date so that the attacks can be predicted or identified and tackled with appropriate counter measures. There are various security threats that hover on the internet starting from man-in-the-middle attack, Denial-of-service attack, Distributed-Denial-of-service attack, ARP spoofing, This thesis aims to address one of the most effective security attack known as Session Hijacking. Session hijacking is a man in the middle attack approach in which the attacker listens to the traffic and sniffs all the packets on the network by using a sniffer like wireshark, ethereal, etc. Using the captured packets the attacker then tries to extract user information like usernames, passwords, etc. to further the attack. All the information the attacker captures may not be useful, but if the attacker retrieves any sensitive information about the user then this information could be used to get access into the user account and steal valuable information or damage the user account. This thesis tries to tackle the session hijacking attack and nullify any effect that the attacker can have over the captured information. Yes, it has become very important to safeguard any information that is out there on the internet, the trend of attacks on the computer is increasing every day and it’s important that we as administrators handle these issue at the right time and the right way.
A research on the existing and proposed approaches of this type of attacks has been reviewed and will be used as a basis for the approach in this implementation.

The approach in this thesis to handle session hijacking attacks is of two types:

- Client side detection
- Server side detection

The results will be formulated to compare the detect-ability rate both on Client side and Server side detections. This comparison will be based on the existing tools in the network security field and the proposed tool in this thesis.

1.1 Background

The transmission control protocol was first introduced as a network protocol in 1974. Since then it has become a core network transmission protocol and a basis for various applications to be built on this protocol. The security of these applications (HTTP, SMTP, and SNMP, SSH etc.) has become a serious threat to the developers and the administrators. There have been various researches in the field of network security to provide or introduce an efficient and a viable defensive mechanism to handle such threats and prevent them. Amongst all the types of network security attacks Man-In-The-Middle-attack (MITMA) has been the most efficient and successful attack. Since the MITM attack uses the flaws in the TCP protocol to capture or rather listen to packets without bringing any attentions from the user basically working in stealth mode. There are various types of MITM attacks but the scope of this thesis is to develop a framework for the detection of session hijacking. Because of the unsuccessful and inefficient tools to detect and prevent session hijacking attacks, this motivated me to choose this topic as my research area (Zheng, Poon, & Beznosov, 2009).

The fundamental question in the detection of session hijacking attack is:

“How effective is the detection rate of such intrusions?”

This thesis strives to answer this fundamental question in the most efficient way and the results will serve as the basis for answering the above question. The results will also compare between various test cases and also tested against different operating system like Windows and Linux flavors.
1.2 Aims & Objectives

The aim of this project is to develop an approach that will detect and prevent session hijacking attack.

**Brief:** This project aims to introduce an approach on the server side that will make the session information much more secure. By increasing the security of the sessions we can continuously monitor to see any illegal session and prevent them from gaining access to the session contents.

**The following are the objectives of this thesis:**

1. Demonstrate a real-time session hijacking.
   a. To demonstrate a session hijacking we will be using the following tools and operating system(subject to change) on the attacker end:
      i. Back Track 4 (Open source Linux platform operating system)
      ii. Ferret and Hamster (Open source hijacking{Side jacking} tools)
      iii. Wireshark or Ethereal (Packet Capturing tool)

2. To implement the approach on the server end to detect such an attack in progress and prevent the attack.
   i. To demonstrate the above we will be using a simple website with a login page which exchanges session cookies to keep a session alive and store information about the session until the user terminates the session(by logging out of the website).
   ii. A website with login page
   iii. A web server that responds with a session
   iv. A web browser that communicates between the client and the server to display the contents of the requested page.
3. In the user end implement and test detection application to detect devices in promiscuous mode.

1.3 Thesis Layout

The outline of the plan to be carried throughout this thesis is described in the following section:

- Research on the problem area – Session Hijacking detection mechanism
- Understanding the underlying issue in detecting session hijacking
- Define the approach to be constructed to evaluate the underlying problem
- Research on the necessary tools and techniques required as the basis for the implementation
- Create a customized test bed
- Implement the solution in the test bed
- Iterate the test scenarios and capture the results
- Representing the results in the graphical form
- Comparing the results with existing approach

The above plan is just an outline of this thesis; there several other steps and researches that was conducted and carried out in order to set-up the complete test bed and implement the solutions. The inputs including the survey conducted to get broader perspective on the research area and also to analyze the practical implication of the detection mechanisms.
2. LITERATURE REVIEW

As a foundation for this thesis many research papers were reviewed and the key characteristics of those research implementations have been used to formulate the framework of this project. The CookieMonster was a process driven experiment to analyze the strength of the algorithm used to generate this cookies. BookScar tool was the tool developed which automated the entire process of archiving the cookies generated and checking for the strength of algorithm (Pauli, Engebretson, Ham, & Zautke, 2011). The basic four steps involved in the cookie monster experiment are:

- The BookScar was installed in the attacker’s machine and used form authentication as the authentication for login process from a web application (Pauli, Engebretson, Ham, & Zautke, 2011).
- By logging into web application on the attacker’s machine the cookies were collected from the BookScar tool (Pauli, Engebretson, Ham, & Zautke, 2011).
- Runtime variable was used to store these cookies (Pauli, Engebretson, Ham, & Zautke, 2011).
- These cookies were archived in a database and the time stamp was also saved (Pauli, Engebretson, Ham, & Zautke, 2011).
- Finally the session between attacker and the web application was destroyed by generating a new unique key and serving it to the attacker (Pauli, Engebretson, Ham, & Zautke, 2011).

In the above experiment there were too many unexpected errors while testing the BookScar tool. The archival and analysis of this experiment required a massive amount of time and processing power was involved. It was also uncertain that the BookScar tool can be installed on the attacker’s machine. Also the hardware constraint in this experiment was a major concern alongside the different servers involved to achieve their goal.

Session Sidejacking and session hijacking are interchangeably used in network security field. Both these attacks basically steal and compromise a user’s account on the server. Session verification with three different levels was proposed in a research paper by Vinay (Kumar, 2011). This three level verification system was on the
assumption that the server used a Hyper Text transmission protocol secure (HTTPS) connection. The port number used by HTTPS service is 445. The proposed technique works in the following steps:

- When the user logs in to a web application a cookie is stored as a value field on the users browser and named as ‘key’ (Kumar, 2011).
- This cookie value is an encrypted value received from the server.
- The main technique used in this research was using a feature in the hyper text markup language (HTML) version 5 known as local storage (Kumar, 2011).
- The session is locked to the clients IP address; this is the level 1 verification (Kumar, 2011).
- The user agent string from the client’s web browser is also locked to the session; this is the level 2 verification (Kumar, 2011).
- The final level of verification is to run a java script verifying the secret key value when the server redirects user to the homepage (Kumar, 2011).

The main flaw of this experiment is assuming that the server will always have a HTTPS connection. Since not all servers are encrypted with SSL certificate this process of three levels verification will not be efficient in practical situations. However running the java script code to extract secret key value and verify the value against the server is an excellent point to consider. The other major limitation in this technique is that the java runtime environment is required as a pre requisite on the users machine (Kumar, 2011).

2.1 Computer Security in general

Any communication that happens over the internet needs a complete isolation from one another. This forms the basis for a secure communication. A computer security can be defined as any system that is vulnerable to attacks and security needs to be hardened to protect the valuable assets within that system. Hardening a system requires extensive amount of evaluation and testing of the network, infrastructures, etc, to be applied on that particular system and then the results may be used as a consideration/input for developing an effective security methodology for the infrastructure. The security of a communication over a network depends on its effect.
and importance of the information sent over the network. In general there are various
attacks to which a computer, a system or an infrastructure can be affected. The attacks
can be broadly classified into the following categories (Li, Zhang, & Gu, 2004):

- Attacks on Network
- Attacks on Systems
- Social Engineering attacks

Attacks on Network:

The attacks on the network are further divided into the following categories
(Li, Zhang, & Gu, 2004):

- Attacks on protocols
- Attacks on the applications
- Man-in-the-Middle Attacks

Attacks on protocols:

The attacks on protocols are a very common, and it is exponentially increasing
day by day. There are various layers in which the attacks might be conducted; the
easiest would be application layer. Since most of the protocols are designed to work
from the top-down architecture of the Open-Source-Interconnections or commonly
known as OSI model.

The common attacks are:

- **Scanning the hosts:** Ping the host machine using ICMP broadcast
  packets, the reply received in the ICMP packets are then used to
determine the number of users in the network and the host machine IP
  addresses. This gives the attacker a broad picture of the network hosts
  and then allows the attacker to narrow down as to what his intentions
  are towards that particular network (Li, Zhang, & Gu, 2004).

- **Scanning the ports:** By using a simple port scanner the attacker can
  scan a network for any open ports, and investigate further into those
  open ports to gain access into the network thereby gaining access into
  the host machines. In this attack, most networks will have some of the
  commonly used services and their port numbers like port 80 for HTTP,
port 22 for SSH; port 21 for FTP, port 23 for Telnet, etc will become vulnerable for these attacks (Li, Zhang, & Gu, 2004).

- **Reverse DNS and ARP Spoofing:** In reverse DNS spoofing the attacker can modify the Domain name services to change the route of the packets to attacking machine from its original destination. In ARP spoofing the MAC address plays a crucial role in order to listen to the packets in some cases even redirection of the packets (MITM attacks) are possible (Chomsiri, 2008).

- **Denial and Distributed Denial of Services:** In Denial of service, a server is flooded with multiple requests, such that the server will become unresponsive after a certain period of time, thus denying service to any legitimate users who would have been connected to that server, this type of attack is conducted from a single attacker machine. In the distributed denial of service the attacker compromises (in most cases) or combines together with multiple host to flood a server with packets and hence the traffic increases, thus denial of service is achieved. The difference between DOS and DDOS is that in DDOS the attack is conducted by multiple machines at multiple locations.

**Attacks on applications:**

The attack on network application are also increasing, the common reason behind this being that most of the network application are also built on the TCP protocol. The intrusion happens at the application layer for e.g., Session compromises, cross site scripting, etc. (Li, Zhang, & Gu, 2004).

**Man-in-the-middle Attacks:**

The attacker using a sniffer sniffs or listens to packets on a network and uses that information to further the attack. This portion is covered in more detail in the following section.

**Attacks on System:**

Any host machine that is compromised by an attacker falls into this category. This attack may necessarily not arise from a network attack. The attack can be a malicious code hidden in an email, an attacker taking control over the host system by
installing a backdoor disturbed in a tool, Dictionary attacks, inducing Trojans horses, inducing worms into a system, create a virus program and luring a user to download a tool while the tool contains this malicious viruses that will multiply within the system.

**Social Engineering Attacks:**

This is a very difficult attack to be successful at, since now it’s in the user’s hand to control the information he/she gives out to the attacker. Social engineering is where the attacker attracts users to a particular website and tries getting as much information as possible in order to determine what type of network, the type of systems, etc.

**Mitigating these types of attacks:**

There are various types of methods it depends on the types of attack that we are trying to mitigate. Some of the common methods are:

- Intrusion detection Mechanisms
- Firewall
- User Awareness
- Security Testing Methodology
- Virtual Private Networks
- Physical Security

**Intrusion Detection Systems/Mechanisms:**

An intrusion detection system protects an infrastructure at two different level like network level and host machine level. Developers should take careful consideration is developing customized IDS’s for their organizations. For network level, IDS will have to be implemented between the routers and servers, also depending on the level of traffic received. For a typical host system, the IDS can be implemented between the user machines and local network (Bottino, 2006).

**Firewall:**

Firewalls are ideal for analyzing the incoming and outgoing traffics of a network. Firewall can be installed both within the network and outside the network. In a firewall a rule based definition can be given to what type of incoming traffic are
allowed or denied, similarly what type of outgoing traffics are allowed/denied in that particular network, this a filter based approach and can every effective if correctly implemented (Bottino, 2006).

**User Awareness:**

A system can have a very sophisticated system security system, but if the users are not aware of their responsibility regarding the do’s and don’ts on the network, it will become a very serious security threat. The user’s should be made aware about the restricted the devices also the restricted information to be given out.

**Security Testing Methodology:**

To achieve maximum security a good testing methodology should be followed in order to evaluate the infrastructure security level and any possible loopholes. For big organization it’s important to gets a well security testing methods regularly to keep their system up-to-date like vulnerability assessment and Penetration testing.

A vulnerability testing is conducted to assess the systems for any possible security loopholes, while penetration testing is conducted to exploit the system to check for any possible vulnerability.

**Virtual Private Network:**

A virtual private network is a private network that is implemented over the public channel to connect to a remote host (Wood, Stoss, Chan-Lizardo, Papacostas, & Stinson, 2002). By enabling this service we can protect any resources that will access from outside the infrastructure via internet in an encrypted manner using various cryptography technologies.

**Physical Security:**

A physical security is the security given to the location where the data center, sever room or any information storage area, office infrastructure is situated. It is important secure the physical location, since any damage done on a physical location will never be recoverable and it’s lost forever. Security alarms, ID cards for employees, biometric entry, etc. are some of the technologies that can be used.
2.2 SESSION HIJACKING

A typical session hijacking is a well-known man-in-the-middle attack in the world of network security and its one of the favorite attack for the attackers because of the nature of the attack. A user who is already logged in(authenticated) to a webserver and has a valid session existing between the user and the server, the attacker takes control over such a session, basically hijacks the session from the user and continues the connection to the server pretending to be the user (Andrew & Daniel, 2006). This has become increasingly common because the attackers are in a great advantage of not having to waste hours and hours to crack a password, or to try and conduct a dictionary attack against the server, since the user has already been authenticated and in a active session it makes it so much easier to just listen to the traffic on the network without the knowledge of the user (Andrew & Daniel, 2006).

The primary reason it becomes easier for a attacker to sniff the network and impersonate the user to the server is because when the user initially authenticates himself/herself to the server, in some cases only does the server uses a secure encrypted connection like HTTPS to authenticate that particular session of the user rest of the connection is sent in plain text (Andrew & Daniel, 2006).

There are three types of Session Hijacking attacks:

- **Active Session Hijacking**
- **Passive Session Hijacking**
- **Hybrid Session Hijacking**

**Active Session Hijacking:**

An active session hijacking is the one in which the attacker takes control over an active session of the victim and starts to masquerade as a genuine user by communicating to the server. There are several methods to drop a user’s connection to the server, one of the most common is to flood the targets machine with huge amount of traffic, and this type of attack is known as Denial of Service. By doing this the attacker puts the user into offline mode, now the attacker has full control over the session. Throughout this process the attacker is in stealth mode listening and monitoring the packets traversing over the network using a packet sniffing tools e.g. wireshark, ethereal, etc.
The below screenshot (1) shows a capture of packets using a sniffer called Wireshark on the network, this screenshot was taken while trying to conduct a session hijacking attack for this project.

**Screenshot (1): Capture of packets in Wireshark**

As shown in the screenshot there are various parameters that are seen which can be used against the target machine by the attacker. The following figure (1) shows a generic session hijacking in active mode:

**Figure (1) Active Mode Session Hijacking**
As illustrated in the above figure (1), it clearly shows how a typical session hijacking attack is conducted between a client and a server by an attacker. The traffic is constantly monitored using the packet capturing tool and then the packets are analyzed to understand which packet contains the session information required to authenticate to server.

**Passive Session Hijacking:**

In a passive session the attacker listens to all the data and captures them for future attacks, in most cases to perform any type of a hijacking attack it is important that the attacker starts off with passive mode. The disadvantage in the passive mode attack is that the attacker might not be that efficient in succeeding on the user impersonating to the server, unless the user session is still alive in most cases it will not be, if the user logs off from the server.

Another typical man-in-the-middle attack is session-replay, unlike session hijacking, in a session-replay the attacker captures all the packet and alters the packet information before sending it to the server for the authentication, this a typical man-in-the-middle attack because the attacker is in-fact between the user and server modifying the packet and sending. The following figure (2) shows a typical session replay attack.

![Session Replay Attack Diagram](image)

**Figure (2) Session Replay attack**

As shown in the above figure (2), the attacker interrupts the traffic between the server and client, and modifies before sending it back. This type of attack brings up a lot of suspicion for a network administrator or the user itself. This is lot more time consuming when compared to other types of man-in-the-middle attacks.
Hybrid Session Hijacking:

In hybrid session hijacking the attacker implements both the modes of attack that is passive and active mode to successfully complete the attacks. In this type of attack, the attacker monitors the pattern of traffic that has been sent over the network and the attacker chooses a session to impersonate. A typical example will be a public unprotected wireless network, where the attacker has access to multiple sessions in progress. All the attacker has to do in this situation is to wait for the right session and hijack the session from the user.

Further the session hijacking attacks can be categorized into 2 different sub types which dependent on the Spoofing techniques used for the attack (Andrew & Daniel, 2006):

- **Blind Spoofing-attack**
- **Non-blind Spoofing-attack**

Blind Spoofing-attack:

Spoofing is a technique of compromising the target machine without attracting any attentions to the attack. When traffic cannot be seen and if the attacker just dumps the traffic between the client and the server, and then by guessing the tcp sequence number tries to authenticate into the server. This makes it a most difficult type of attack to perform, and in most cases the attacker ends up spending a lot of time without any success (Andrew & Daniel, 2006).

Non-Blind Spoofing-attack:

This is the most common type of attack, since in non-blind spoofing the attacker can see the traffic between the client and the server machine. It makes it easy for the attackers to analyze the packets in an active mode and further the attack by impersonating as the user to the server. This becomes difficult in a switched network, as the switches do not broadcast all the packets to all the hosts, rather to a particular host. But with some advanced configuration, if the attacker can compromise the VLAN (Virtual LAN) port then session hijacking is possible in the network (Andrew & Daniel, 2006).
Some of the tools used in session hijacking are:

- Hunt
- T-Sight
- Juggernaut
- TTY Watcher
- Hamster and Ferret
- Wireshark
- Ethereal

2.3 Detection tools and techniques for Session Hijacking

To protect against session hijacking there are various intrusion detection tools and some advanced techniques. The below list are only a few commonly used tools:

- Arp-ON
- ARP-PING
- ANTI-SNIFF
- Cookie Monster
- Wavelet based detection
- Cisco Intrusion Detection System (IDS)
- Sans Intrusion Prevention System (IPS)

**Arp-ON:** This tool is aimed to secure the Address resolution protocol, and avoid any MITM attacks (Darknet, 2000).

**ARP-PING:** This is a Linux tool, and allows the user to ping a Media Access Control (MAC) address directly. This can implemented to detect the attacker using a sniffer on the network (Beyond-Security, 1998).

**ANTI-SNIFF:** In this tool, the user can detect any sniffer on the network used for packet capturing (Storm, 2011).

**Cookie-Monster:** This tool was developed for analyzing the strength of the cookie by archiving and analyzing (Pauli, Engebretnson, Ham, & Zautke, 2011).
**Wavelet-based-detection:** In this techniques, the author analysis the signal strength using wavelet transform to detect a session hijacking (Long & Sikdar, Wavelet Based Detection of Session Hijacking Attacks in Wireless Networks, 2008).

### 2.4 Recent Developments on Session Hijacking

The researches have been working towards creating a sensible system that can not only detect but also prevent any session hijacking attempts. There are several research based algorithms and techniques that have been proposed and have also been successfully implemented.

Some of recent developments are:

- Ensuring a secure cookie generation
- Implementing the CIA technology, C-Confidentiality, I-Integrity, A-Authentication
- Using an encrypted connection for the transmission like HTTPS
- Locking a particular session to its corresponding users.

**Ensuring a secure cookie generation**

A cookie is the one in which all the user information is sent to authenticate a particular user to the server. Some of the information embedded in a cookie is:

- Username
- Password
- Session ID
- Session creation date - timestamp
- Expiration time of the Session

Whenever as session is hijacked, in a passive or active mode the attacker analysis this cookie information sent over the network to authenticate and compromise the users account. In the cookie monster paper, the author clearly indicated how a cookie can be archived and analyzed for its strength using various algorithm techniques (Pauli, Engebretson, Ham, & Zautke, 2011).

**Implementing a CIA technology:**
A low-cost robust solution was suggested, that is to change the cookie generated for each transmission between the client and server, which implements the CIA technology. This is achieved by creating a java-script on the server side and making the cookie available only to the client’s browser, this will also prevent cross-site-scripting (Jeffrey & Mostafa, 2011). The protocol used to secure the wireless network was later developed as standard usable plug-in for the browsers like Firefox, Google chrome, etc (Jeffrey & Mostafa, 2011).

**Using an encrypted connection:**

Hypertext transmission protocol security (HTTPS) was introduced in order to secure any network based transmissions. This is type of protocol uses a secure socket layer (SSL) also known as Transport Layer Security (TLS v1) to implement HTTPS. The only drawback in this technology is that not every website can implement because of the cost involved in implementing and maintaining such a service (Rupinder, Jason, & Andrew, 2006).

**Locking a Session to its corresponding user:**

A very effective technique to prevent session hijacking is to lock a session its particular user and this will ensure that even if the session is hijacked, the information the attacker has is of no use. In a research paper on locking session, the author indicates that using a unique fragment identifier, unique HMAC algorithm with a secret key shared between the client and the server. The token generated from the initial login over the SSL is retained and reused into the browser’s fragment identifier, thus making an efficient and low-cost solution to secure the session and eavesdropping (Ben, 2008).

The developments pointed out above are research based; there are some tools that are developed by various infrastructure giants like Cisco, juniper, etc.

The following are few of the tools are:

- CISCO Intrusion detection tool (Andrew & Daniel, 2006).
- Juniper network VPN devices
- State of the art – Switches by Cisco.
- Firewalls
3. SESSION HIJACKING DETECTION TOOLS IMPLEMENTAION

In this thesis we will be implementing the detection mechanism on both the client end and the Server end. This will provide maximum protection against any type of session hijacking attack. The proposal in this thesis classifies the detection strategies as the following:

- **In-Network Strategy**
- **Out-Network Strategy**

As we have seen every approach in the research field for session hijacking detection mechanism, the authors have either proposed a detection mechanism for a specific network that is within a LAN, or for an outside network. But this paper proposes an approach that will protect targets against session hijacking attacks from both inside the network and also outside the network.

**In-Network Strategy:**

A network in which the defense is defined for an attack that can occur within a network is known as In-Network Strategy. In the In-Network Strategy the detection of a session hijacking is seen from the client’s perspective. The proposal defines a mechanism to be adapted in-order to detect any session hijacking attempts from the user-end.

One of the basic criteria for the session hijacking attack, the attacker needs to listen to the traffic be it in active mode or passive mode, and choose a session to hijack. However, in order to listen to the traffic, a sniffer needs to be installed on the attacker machine and this sniffer will listen to all the traffic that traverses through the network. A sniffer tools like wireshark or ethereal needs to set the mode of the Network Interface Card (NIC) to receive all traffic. The basic functionality of Network Interface card is to accept only packets that are designated to it or any broadcast packet to check if the broadcast is for that particular host (Daiji, 2001). To set a NIC card into a mode where it can receive all the traffic even if it is not intended to that host or IP address, such a mode is known as **Promiscuous Mode** (Daiji, 2001). As shown in Figure (3).
This is independent of what operating system is installed on the host machine, although there are some restrictions on the Windows operating systems.

The strategy proposed in this thesis is to use this characteristic of a sniffer against it to detect the network Interface card in promiscuous mode.

There are several methods available to detect a sniffer in promiscuous mode; this paper will be using Internet Control Message Protocol (ICMP) to detect a sniffer in the network. ICMP packet is a simple ping packet which echo’s the response from the host, the following Screenshot (2). The echo replies are from the host to which the ping was intended to. If the host is unavailable it will reply a destination not reachable error as shown in the screen capture. Using the ICMP protocol we can send a fake PING packet to the network, a host whose Network Interface Card (NIC) is set to promiscuous mode is designated to receive all the packets from that network. This is

**Figure (3) NIC in Promiscuous Mode accepts all the packets**

LAN connection, the host on the left receives all the packets

Incoming Packets

Sniffer installed Host
NIC in Promiscuous Mode

NIC Card with in its default mode
where we will try and manipulate the Sniffer making it believe that it is a legitimate packet and forcing it to send a response to the ping packet.

```
C:\Windows\system32>ping 192.168.0.1
PING 192.168.0.1 with 32 bytes of data:
Reply from 192.168.0.1: bytes=32 time=1ms TTL=64
Reply from 192.168.0.1: bytes=32 time=1ms TTL=64
Reply from 192.168.0.1: bytes=32 time=1ms TTL=64
Reply from 192.168.0.1: bytes=32 time=1ms TTL=64
Ping statistics for 192.168.0.1:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
   Approximate round trip times in milliseconds:
       Minimum = 1ms, Maximum = 1ms, Average = 1ms
C:\Windows\system32>ping 192.168.0.100
PING 192.168.0.100 with 32 bytes of data:
Reply from 192.168.0.100: Destination host unreachable.
Reply from 192.168.0.100: Destination host unreachable.
Reply from 192.168.0.100: Destination host unreachable.
Ping statistics for 192.168.0.100:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Screenshot (2): Ping results for successful reply and also unsuccessful response.

The details of this implementation are explained in the test-bed section of the thesis.

**Out-Network Strategy:**

A network in which the defense is defined for an attack that occurs from outside the network is known as Out-Network Strategy. In the OUT-Network Strategy the detection of a session hijacking is seen from the Server’s perspective. The proposal defines a mechanism to be adapted in-order to detect any session hijacking attempts from the server-end.

The server can play a crucial role in defining and detecting an attack, since the server only reads the session information in a cookie as it comes to it and processes it without having to notice from where it is come. If there are username and password included in the cookie, the server processes them to a database for instance and cross verifies if the credentials are correct and then serves the request content with a session ID, for that particular session. Now it’s the attackers turn, who is watching all this process happen, now the attacker swoops in and steal the session information after the
initial authentication process has been completed and uses this session information to successfully receive the contents from the server. Typical cookie information is as shown in the following screenshot (3) this screenshot was taken from a Google Chrome browser after authenticating to the Facebook server.

As we can see there are more than one cookie created for each session, but the session ID will remain the same throughout that particular session. Now for each host outside the network, we will have a unique IP address and port no’s used in-order to interact with the server, and that is how a server identifies as to where the request is coming from and where the request needs to be send.

Using the uniqueness of the IP-address on the internet we can differentiate the session requests from various hosts, now the fundamental question will arise as to how to save and use this IP address against each request. For every request the user makes to the server, we will assume that the initial request made to the server at a specific point of time, from a specific host and identify this as unique. Now we use the filter technique to separate the genuine requests from false requests or hijacked requests. The details of this implementation are explained in the test-bed section.
3.1 User System

The following are specification of the user system used for the testing:

- Operating System: Windows 7
- IBM ThinkPad T60 laptop
- Processor: Intel Dual Core – 1.6GHz
- Memory: 2GB RAM
- Hard disk: 160 GB HDD
- A local browser

3.2 Server System

For the experiment of this implementation we used the following specifications:

- Server Installed: Apache Tomcat Version 7.0.5
- Server has been Installed on a windows 7 operating system
- The server here has a localhost identity
- And it is in the same machine as the user.

3.3 Attacker System

Attacker system is the specification:

- Back-Track Linux operating system
- Ferret and Hamster tool to perform the attack
- Intel Core i3 processor – 2.0Ghz
- Memory: 3GB
- Hard Disk: 500 GB

The above specifications are some of the main design requirements used in order to conduct the testing of this approach. There were constant updating and alternate patches were required to be used at some stage during the testing.
4. SURVEY ANALYSIS

A survey was conducted to analyze the research problem being discussed in this thesis. The main focuses of the survey are as follows:

- Session Hijacking Awareness
- Effective Solutions to handle Session Hijacking
- Known Detection Mechanisms

The response was that the awareness of the session hijacking between users where very less, when compared to a normal computer user to that of an administrators. Some of the advanced answered that they knew about a session hijacking attack, but were still not aware any effective solutions that can be deployed, other than an expensive Intrusion Detection System. The effective solutions included, user awareness, restricting users from installing third party software on a LAN (No Administrative Privileges). The following graphs 1 show the awareness between different user categories and effective solutions to implement session hijacking.

Graph (1) shows awareness of session hijacking attack between various user categories
5. SYSTEM DESIGN

The system design consideration for the project was a more specific to the In-Network and Out-Network strategies. To test the proposed framework in this thesis a client-Server application was developed using the following tools:

- For IN-Network Strategy:
  - Development Language: .Net Framework
  - Raw Socket libraries
  - Windows Command Prompt
  - Network Interface Card (NIC)

- For OUT-Network Strategy:
  - Apache Tomcat Server
  - For filter construction – valve’s in apache tomcat
  - Webpage development
    - HTML Code
    - JavaScript Code

In the IN-Network strategy, the application developed in .Net using Visual basics to create a raw socket application. The raw socket application will give this approach the required advantage of creating a customized packet. In this customized packet will have a valid IP address but a fake broadcast address like “FF:FF:FF:FF:00”, if a host accepts this fake broadcast packet then the user will be alerted that there is a host whose Network interface card in promiscuous mode.

In the OUT-Network strategy, a valve will be introduced as a filter into the Apache Tomcat server. This filter will be used to run a loop in the server which will check each host IP address, even if the session has been compromised by an attacker but the information used by the attacker to get the contents from the server will prevented. This valve will store the session information of the client as defined programmatically; the JavaScript used to create the webpage is a server-side scripting language. The JavaScript will be executed in the server side and will not be visible to the user which provides an optimal security to the website.
6. FLOW CHART & PSUEDO CODE

6.1 IN-Network Strategy flowchart

Start

User launches the application

Clicks on the preprogrammed button called Detect

The code executed sends a fake packet into the network

If the packet was not accepted by a host

No devices detected in Promiscuous Mode

If the packet was accepted by a host

Warning: Device detected in Promiscuous Mode

Continue Browsing

Close Connection
6.1.1 IN-Network Strategy Pseudo Code

For the detection of session hijacking on the user-end, when the user clicks on the detect button, a raw socket is opened and a fake ICMP packet is created and sent into the network. The following is the Pseudo Code:

Open RAW Socket

{
    Send Byte - ICMP Packet (IP Address, Altered MAC Address)
}

User Warning Message!

If Ping. Respond = true Then

{
    “Detected NIC in Promiscuous Mode”
}

Else

{
    “No Device in Promiscuous Mode”
}

End

Close Raw Socket

Close Connection

The above logic is applied and used in windows 7 machine. But the vb.net application does not support non windows platform. Hence the approach used in Linux operating system is using a third party tool to change the MAC address and evaluate the corresponding results.
6.2 OUT-Network Strategy flowchart

Start

User requests for a webpage
E.g. www.mywebsite.com

The server responds with a page that requires the user to enter username and password

User enters credentials

Server checks if the credentials are valid

YES

The server starts a session with the user

NO

Request the user to enter valid credentials

A

B
Attacker who is on the same network sniffs the packets between the user and the server.

If the user doesn't exist in the database then the server closes the connection.

The attacker, who has the session information, now tries to hijack the user session.

Close Connection

New request for the existing session (Session Hijacking)

When a new request arrives at the server for an existing session, the server compares IP address of the new request with already existing user information like the IP address, session created time, browser fingerprint details to validate if the request is legitimate or an attack.

If the user information doesn’t match then the server will block (or close the connection) the IP address with a new request and inform the user that there was an attempt for duplicate session.

A new session is established between the user and the server.

Once user logs out the server closes the session and the connection.

(Detection of SH)

Stop
6.2.1 OUT-Network Strategy Pseudo Code

For the detection of Session Hijacking on the server end a valve is created on the Apache Tomcat Server, this valve will act as a filter between the client and the server. Every time a new request is made by the client, the valve adapted in the server stores the value of IP address and check in a loop every time another request for the same session is obtained. The following Pseudo Code:

Step1: Storing the value of the IP Address in the temp variable

‘Get CurrentUser IP Address = temp_IP_session_Name

Step2: Checking this against each request for that particular session

Get Current Client_IP

If Client_IP = temp_IP_Session (Variable IP Stored)

{  
  ‘Process the request

}

Else

{  
  ‘Deny request

}

End If

Finally the client IP which is stored in the Session will be valid for only that session, once the session is closed the temp variable is destroyed. The advantage in creating a valve in apache tomcat as a filter is that it completely customizable and more parameters of the session can be included into the valve component (Tomcat).
7. IMPLEMENTATION – TEST BED

In this section we implement the proposed mechanism to test the practicality of the application and detection of the session hijacking. The implementation is on the basis of the following 2 categories as described earlier:

- IN-Network Strategy
- OUT-Network Strategy

The following subsections explain the implementation in brief with screenshots. The implementation is based on three ends the user or client end, the server end and finally the attacker end.

7.1 User End

For the implementation at the user end, as discussed above the categories are further divided into inside and outside the network. For IN-Network strategy we will be testing the detectability of the promiscuous mode and for the OUT-Network Strategy we will be login into a webpage and authenticating the user to the apache tomcat server.

7.1.1 IN-Network Strategy

Here a client-server application developed using visual basics, which uses a raw socket to connect to the server. This section checks the session hijacking attempts within a LAN or WLAN.

7.1.1.1 Multiple-Client-APP Application

The steps for testing multiple-client-app to detect if a device is in promiscuous mode in the network on a Windows operating system are as follows:

- Launch the application
- Click on the Detect button
- A raw socket is opened and a raw packet is sent with a fake broadcast address.
- A message pop is alerted
- If a device is found in promiscuous mode within a network, the message shown is as seen in the screenshot (4).
If no device is found in promiscuous mode, then the message is displayed as shown in the screenshot (5).

Screenshot (4) Device detected in Promiscuous mode

Screenshot (5) when a device is not detected in a Promiscuous mode

The steps for detecting if a device is in promiscuous mode in the network on a Linux-BackTrack operating system are as follows:

- Launch PACKETH preinstalled application
- Enter the destination MAC Address (A different Broadcast address)
- Enter the source MAC address
- Enter the Source and Destination IP address

Here the source and destination IP address remains valid and targeted towards the network, but instead of sending the right broadcast address which is default, using this PackETH we fake the broadcast MAC address to “FF:FF:FF:FF:FF:00”. By default the broadcast address is sent out to identify which host has this IP address, an Address resolution protocol (ARP) process is performed to convert the IP address into MAC address and MAC address into IP address. This can be identified when a device in the network is using a sniffer or in promiscuous mode is when the byte of date (PING – ICMP packet) is accepted by the particular IP address. This process is shown in the following screenshot (6).

Screenshot (6) shows the detection of promiscuous mode in BackTrack.
The blue arrow on the screenshot (6) indicates the destination MAC header changed from that of a default broadcast address, the red arrow indicates that the ping was successful hence confirming that the host machine is in promiscuous mode and finally the yellow arrow indicates the host machine (possible attacker) IP address.

7.1.2 OUT-Network Strategy

To test the session hijacking attempt from outside the network on the user end, the user will initiate a web session using a Java Server Script (JSP) login page.

7.1.2.1 Initiating a Web Session

A sample login webpage is created to authenticate the user with the server, while the user has been authenticated the valve setup as a filter on the server-end will save the current IP address of the host as a temporary variable. The below screenshot (7) a webpage which is connects to the Apache Tomcat server.

![Simple Login to Detect Session Hijacking](image)

**Screenshot (7) Local webpage on the user end**

The localhost mentioned on the serve URL, is because we are connecting to a server that is locally installed and accessed by the browser. The hostname of the server used is “localhost” and the port number used by the server “8080”. This page is java scripted with server side scripting and the code for the webpage is referred in the appendix section.
7.2 Performing Session Hijacking - Attacker End

In this section the attack is demonstrated from the attacker end, two different approaches are taken:

- IN-Network Attack
- OUT-Network Attack

In the IN-network attack the session hijacking attack is attempted from within a network and in the OUT-Network attack the session is hijacked from outside the network.

**IN-Network attack**

The session hijacking attempt was conducted and performed from the Attacker End on a Linux-BackTrack operating system. The following tools were used to perform the attack:

- Ferret and Hamster
- Wireshark
- Proxy settings to access the Hamster on Firefox

**Ferret and Hamster:**

A tool installed in BackTrack to capture a session and replay it to get the contents of the user request from the server (Robert, 2009).

**Wireshark:**

Is a widely used sniffer which analyses the different network attributes like protocols, port no, packets, etc. (WireShark, 2000). Also this allows the attacker to put the Network interface card to promiscuous mode.

**Proxy Settings**

A proxy is set in the Firefox to access the captured packet by wireshark and replay it in the browser before furthering the attack.

7.2.1 Initiating the attack

To initiate the attack the following steps are followed:
Step 1: Wireshark is launched and the appropriate interface is selected to receive the traffic while making sure that the sniffer is set to promiscuous mode. The following screenshot (8) shows the packet capture in Wireshark.

![Wireshark Packet Capture](image)

**Screenshot (8) Wireshark – packet capture in BackTrack**

Step 2: Saving the packet captured. Now starting ferret with \(-r\) option which will extract all the cookies. The following screenshot (9) & (10) describes this step.

![Ferret Starting with -r Option](image)

**Screenshot (9) shows the ferret starting with the \(-r\) option**
Screenshot (10) shows the cookie information extracted from the packet capture

Step3: The final step is to start hamster and set the port no along with IP address to which the hamster is listening to, in this case it was 127.0.0.1:1234 and replay the packet. The following screenshot shows the hamster replay in the Firefox web browser.

Screenshot (10) shows Hamster replaying cookies captured on Firefox
OUT-Network Attack

In this section we will attempt to test the attack to replay the captured session from the users end and authenticate to the Apache Tomcat server. This section covers the session hijacking attempt from outside the network.

7.2.2 Requesting for the session

The attacker who now has the required user’s session information from the capture will attempt session hijacking to the server in this test case it is apache tomcat server. The following screenshot (11) shows the session ID information captured from the victim’s machine.

Screenshot (11) shows the session information captured

The session hijacking attempt is elaborated in the following section, in which the server details are described in detail along with valve configuration.
7.3 Detection of Session Hijacking on the Server End

The detection of session hijacking attempt covers the attempt from the OUT-Network strategy. In this section we discuss the use of filters on the server end.

The server installed for this test bed is Apache Tomcat Version 7; the following screenshot (12) shoes the server which is accessible from the local web browser.

Screenshot (12) showing the server up and running

Valve Configuration on the Apache Server:

The valve component configured as the filter on the Apache server is shown in the following screenshot (13), the highlighted green box points out the portion of the valve configure called as Secure-Session-IP Valve. This valve component is pipelined at the request processing end of the server, with the %a option for the remote IP, hence defining them into a loop makes it much simpler for the valve to filter out and distinguish between each request.

The remote_address_filter of the tomcat serve gives us the utmost help in creating a optimized filter. That is the compares the different IP address against the list be it temporary or permanent. Now this element accepts one of two attributes either allow or deny (Tomcat).
7.3.1 Monitoring for duplicate request

The valve created in the server acts a filter to continuously monitor the incoming requests. Once a request comes in the user’s IP address for that particular session is stored in temporary variable as discussed in the flowchart and the pseudo code

7.3.2 Comparing and blocking requests

First the user authenticates to the server and continues the session. While the user was authenticating to the server the attacker has now captured the session information required to authenticate to the server by impersonating the user. Now the attacker tries to authenticate to the user with the above captured information as shown in the previous section “Attacker End”, the server now will check the temporary variable stored for that particular session and compare the IP address. The IP address does not match the temporary variable which has the actual user IP address, hence the attacker’s request is denied and the server sends out an error message, as shown in the screenshot (14).
Screenshot (14) shows the attacker denied access into the server

Hence the session hijacking attempt was nullified even thought the attacker had the access to session information, but since the IP address did not match the comparing valve filter denied the access to the attacker.
8. RESULTS AND DISCUSSION

In this section the testing results of the IN-Network strategy and OUT-Network Strategy are discussed in brief. The results are classified into the following two categories, those are:

1. Rate of Detectability
2. Time taken to Detect

Rate of Detectability:

The rate of the detection of a Network Interface Card (NIC) in promiscuous mode devices within a network is known as the rate of detectability. The results are tested in two different operating system Windows and Linux.

Time Taken to Detect:

The time taken by the application to detect the presence of a device in promiscuous mode on the client side and the time taken for the server valve to filter out the attacker’s or duplicate session IP address.

IN-Network Strategy – Windows Machine:

The below graph (2) indicates the rate of detectability in a windows machine using the vb.net developed application. The test bed was compared and tested against various third party application against the Multi-client App developed in this proposal.

There are four different test cases that were tested for comparison between three different tools:

- Multi-Client Application
- Anti-Sniff
- Promqry – Windows Proprietary

As seen in the graph the variance in detectability rate is not quite large, in fact the detectability rate is improved in the first test case for the multi-client app when compared to the other two applications. And the test case 4 also shows that the
detectability rate in multi-client application is much higher than the other third party application.

The following graph (3) shows the detectability rate in a Windows machine and Linux machine. Several test cases were confined to test the detectability rate in windows and Linux, the comparison shows that Linux machine has a better detectability rate than windows. The BackTrack flavor of the Linux version which had the PACKETH preinstalled application was used to detect devices in promiscuous mode. The comparison shown is in the percentage unit. This also gives a good understanding of which platform gives us the flexibility in order to perform customized mechanisms. Even though the underlying TCP stack operation is the same for both windows and Linux, the kernel is programmed in a different; there are security drawbacks that do not allow us to develop an efficient detectability tool. The Linux favors programmers and administrators in this issue, since Linux is an open source platform.

Comparison between the Linux-BackTrack and Windows 7 Operating System

The following graph (3) shows the detectability rate in a Windows machine and Linux machine. Several test cases were confined to test the detectability rate in windows and Linux, the comparison shows that Linux machine has a better detectability rate than windows. The BackTrack flavor of the Linux version which had the PACKETH preinstalled application was used to detect devices in promiscuous mode. The comparison shown is in the percentage unit. This also gives a good understanding of which platform gives us the flexibility in order to perform customized mechanisms. Even though the underlying TCP stack operation is the same for both windows and Linux, the kernel is programmed in a different; there are security drawbacks that do not allow us to develop an efficient detectability tool. The Linux favors programmers and administrators in this issue, since Linux is an open source platform.
OUT-Network Strategy:

This section represents server side detection, where a valve is used to differentiate between the IP address of the attacker and the genuine user. The configuration of the valve is shown in the Implementation section and the code used to program the valve is shown in the appendix section. Here we will see the results of the implementation of detection mechanism on the server side, the Apache server has been configured with a Valve component to detect any session hijacking attempt using a straight forward approach of comparing two IP address, the following graph shows the multiple session hijacking attempts and the detectability rate, in terms of time taken to detect the attack.

The time taken for each attack was iterated for 8 different attempts of hijacking, the following graph (4) clearly shows the time taken for each attack in milliseconds. This proves that the detectability rate is much higher with less time taken to detect each attack. The Screenshot (15) clearly indicates the server log file which shows the information of session hijacking attempts logged and their corresponding time taken to log this information.
Graph (4) shows time taken for each attack.

Screenshot (15) shows the time taken to detect these attempts.
9. CONCLUSION & FUTURE WORK

9.1 Conclusion

Testing of the IN-Network strategy and OUT-Network strategy clearly indicates that the detection of a session hijacking attempt from outside the network gives more control, however the strategy used to identify Network Interface Cards (NIC) in Promiscuous mode average at an detection rate of 65 percent. In the results section the graph clearly indicate that the detection rate for different Operating system varies, due to the level of security each operating system kernel exhibits. While the attack which was conducted on the server-end provided an optimum result against hijacking attempts, the server valve clearly distinguishes between the IP address of the attacker from that of the client and without indicating the attacker about the detection. However the attempts to perform detection for the IN-Network strategy proved to be much more efficient on a Linux machine. The time element places an important role, as it is critical for every application to be fast and the time taken to process any request should not take more than a few seconds. This will be the one of the constraints focused in the future work.

9.2 Future Work

The IN-Network strategy can be implemented with different and a more generic approach which can will be applicable to multiple platforms.

Some of the constraints in implementing this mechanism on a production network are:

- The server valve is not an generic component within all the severs
- The detectability rate can be improved on the Windows machine
- The Multi-client application can be developed into a full fledge intrusion detection system, which will require more time and some more critical tools.

On the OUT-Network strategy, as of now the critical parameter used to detect and make a decision is the IP address, but for the future valves to increase the security the valves’ can also check and compare the following parameters:

- User Agent String
- Session ID
- Session Created Time
10. REFERENCES


APPENDIX A: IN-Network Strategy - Multi-Client Application Code

Client side Code:

```csharp
Imports System.Text.RegularExpressions

Public Class client1
    Dim clientSocket As Socket
    Dim byteData(1023) As Byte
    Private Sub client1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        End Sub

    Private Sub btnconnect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnconnect.Click
        clientSocket = New Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
        Dim ipAddress As IPAddress = ipAddress.Parse("127.0.0.1")
        Dim ipAddress End Point = New IPEndPoint(ipAddress, 8800)
        clientSocket.BeginConnect(ipEndPoint, New AsyncCallback(AddressOf OnConnect), Nothing)
        End Sub

    Private Sub OnConnect(ByVal ar As IAsyncResult)
        clientSocket.EndConnect(ar)
        clientSocket.BeginReceive(byteData, 0, byteData.Length, SocketFlags.None, __
            New AsyncCallback(AddressOf OnReceive), clientSocket)
        End Sub

    Private Sub OnRecieve(ByVal ar As IAsyncResult)
        Dim client As Socket = ar.AsyncState
        client.EndRecieve(ar)
        Dim byteRec As Byte() = byteData
        Dim message As String = System.Text.ASCIIEncoding.ASCII.GetString(byteRec)
        clientSocket.BeginReceive(byteData, 0, byteData.Length, SocketFlags.None, __
            New AsyncCallback(AddressOf OnRecieve), clientSocket)
        End Sub

    Delegate _Read(ByVal msg As String)
    Private Sub Read(ByVal msg As String)
        If InvokeRequired Then
            Invoke(New _Read(AddressOf Read), msg)
        Exit Sub
        End If
        RichTextBox1.Text &= msg
    End Sub
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Try
        Dim link As String = "http://www.whatsdatestoday.com/
            Dim myMatch As Match = Regex.Match(res.ReadToEnd, "Welcome \d+, \d+, \d+, \d+", RegexOptions.IgnoreCase)
            If myMatch.Success Then
                Dim IP As String = myMatch.Groups(1).Value
                IP = IP.Replace("Welcome ", "")
                RichTextBox1.Text = IP
            Else
                MessageBox.Show("Doesnot match")
            End If
        End Using
    Catch ex As Exception
        MessageBox(ex.Message, MsgBoxStyle.Information, "Information")
    End Try
End Sub

Private Sub RichTextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RichTextBox1.TextChanged
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        MessageBox.Show(Err.Description)
    Else
        MessageBox.Show(Err.Description)
    End If
End Sub

Private Sub Button2_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        MsgBox("Device detected in Promiscuous Mode")
    Else
        MsgBox("No Device Detected in Promiscuous Mode")
    End If
End Sub
End Class

Public Class Server
    Dim serverSocket As Socket
    Dim clientSocket As Socket

    Private Sub Server_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles MyBase.Load
            Client.Show()
            serverSocket = New Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
            Dim IpEndPoint As IPEndPoint = New IPEndPoint(IPAddress.Any, 8800)
            serverSocket.Bind(IpEndPoint)
            serverSocket.Listen(5)
            serverSocket.BeginAccept(New AsyncCallback(AddressOf OnAccept), Nothing)
    End Sub

    Private Sub OnAccept(ByVal ar As IAsyncResult)
        clientSocket = serverSocket.EndAccept(ar)
        serverSocket.BeginAccept(New AsyncCallback(AddressOf OnAccept), Nothing)
        AddClient(clientSocket)
    End Sub

    Delegate Sub _AddClient(ByVal client As Socket)
    Private Sub AddClient(ByVal client As Socket)
        If InvokeRequired Then
            Invoke(New _AddClient(AddressOf AddClient), client)
        End If
        Exit Sub
    End Sub

    Dimlv1 As New ListViewItem(client.LocalEndPoint.ToString)
    lv1.Tag = client
    lstClients.Items.Add(lv1)

    Private Sub Send(ByVal msg As String, ByVal client As Socket)
        Dim sendBytes As Byte() = System.Text.ASCIIEncoding.ASCII.GetBytes(msg)
        client.BeginSend(sendBytes, 0, sendBytes.Length, SocketFlags.None, New AsyncCallback(AddressOf OnSend), client)
    End Sub

    Private Sub OnSend(ByVal ar As IAsyncResult)
        Dim client As Socket = ar.AsyncState
client.EndSend(ar)
    End Sub

    Private Sub SendMessageToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles SendMessageToolStripMenuItem.Click
            Send("Message from server", lstClients.SelectedItem(0).Tag)
    End Sub

End Class
APPENDIX B: OUT-Network Strategy – Valve Configuration

```java
package com.jerry.tomcat.session.valve;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpSession;
import org.apache.catalina.connector.Request;
import org.apache.catalina.connector.Response;
import org.apache.catalina.valves.ValveBase;
/**
* A valve that intercepts all the requests and checks for session hijacking.
* @author Jerry Louis
*/
public class SessionUnHijackValve extends ValveBase {
    /**
     * This method is invoked before all the requests are handled by the server side
     * logic
     */
    public void invoke(Request request, Response response) throws ServletException,
                  IOException {
        // Starting time for logging
        long start = System.currentTimeMillis();
        // Retrieve the current Session Object
        HttpSession session = request.getSession();
        // Get the original client IP address stored in the session scope
        String storedIp = (String) session.getAttribute("actualIp");
        // Get the current client IP address
        String clientIp = getRemoteAddr(request);
        if(storedIp == null || storedIp.isEmpty()) {
            // Set the client IP address in session
            session.setAttribute("actualIp", clientIp);
            storedIp = clientIp;
        }
    }
```
// If there is a change in the IP address between the requests, it indicates a session
hijacking attempt
    if (storedIp.equals(clientIp)) {
        containerLog.info("Session is authorised, continuing with normal
operation");
    } else {
        containerLog.warn("Session Hijacking attempt detected from IP:
" + clientIp);
    }

    // Invoke next valve or true processing
    getNext().invoke(request, response);
    // Compute processing time
    long time = System.currentTimeMillis() - start;
    containerLog.info("Request took " + time + " ms (" +
    request.getDecodedRequestURI() + ")");
}
/**
 * Returns IP address of the client
 * @param request the incoming <code>Request</code> object
 * @return IP address of the client
 */

public static String getClientIpAddr(Request request) {
    String ip = request.getHeader("X-Forwarded-For");
    if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
        ip = request.getHeader("Proxy-Client-IP");
    }
    if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
        ip = request.getHeader("WL-Proxy-Client-IP");
    }
    if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
ip = request.getHeader("WL-Proxy-Client-IP");
}
if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
    ip = request.getHeader("HTTP_CLIENT_IP");
}
if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
    ip = request.getHeader("HTTP_X_FORWARDED_FOR");
}
if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
    ip = request.getRemoteAddr();
}
return ip;
}
Detection of Session Hijacking

By

Jerry Louis

1031051

Interim Report
# Table of Contents

1. Introduction........................................................................................................02  
2. Aim......................................................................................................................03  
3. Objectives...........................................................................................................03  
4. Literature Review...............................................................................................04  
5. References..........................................................................................................06  
6. Appendices...........................................................................................................07
1. Introduction

In today’s world internet is become a place where everything happens from e-businesses to social affairs, from online education to online shopping, it’s the place everyone goes for different information and it’s a place where one wants to be. Internet is growing exponentially; safeguarding information that is available online has become a very difficult task. One of the well-known attacks in the cyber-security world is Man-in-the-middle-attack; this is basically a type of eavesdropping. There are various attacks and malicious ways intruders steal information online, this type of stealing is also known as hacking. There are various types of attacks a hacker or an attacker would perform like DoS (Denial of Service), Brute force attack, dictionary attack, buffer overflow attacks, Ping of Death Attacks, etc. In this report we will be focusing on Session Hijacking, which is a type of Man-in-the-middle attack. In a session hijacking attack the attacker tries to take over an already existing connection between two devices/hosts. Session Hijacking is a very effective network attack in which the attacker intentionally captures information’s such as a cookie, or a packet that is sent to and fro to the server for authentication. This technique is also called as Snooping.

Session hijacking is of two different types: active hijacking and passive hijacking. In active hijacking the attacker takes over the user session and communicates with the server, basically the attacker forces the user to terminate the connection to the server. In the passive hijacking the attacker monitors the network using some packet capturing tool and then when the attacker finds something of interest, he captures that session information and uses that to gain access into the server, while the victim is also viewing the contents of the same website. There are various advantages for an attacker who can successfully perform a session hijacking attack: one the attacker will gain complete control over the victim’s session without the victim’s knowledge or even worse the server’s knowledge and do any harm to the victims account or steal any valuable information that will be useful to the attacker. This type of attack is very efficient because they are stealth in nature.

The difficulty for any network security administrator or website developer is to detect such a session hijacking in progress without any expensive intrusion detection system that’s available in the market. We will try and propose an approach which will be help the server detect and prevent session hijacking attacks.
2. Aim

The aim of this project is to develop an approach that will detect and prevent session hijacking attack.

**Brief:** This project aims to introduce an approach on the server side that will make the session information much more secure. By increasing the security of the sessions we can continuously monitor to see any illegal session and prevent them from gaining access to the session contents.

3. Objectives

The following are the objectives of this project:

11. Demonstrate a real-time session hijacking.
   11.1 To demonstrate a session hijacking we will be using the following tools and operating system(subject to change) on the attacker end:
      11.1.1 Back Track 4 (Open source Linux platform operating system)
      11.1.2 Ferret and Hamster (Open source hijacking{Side jacking} tools)
      11.1.3 Wireshark or Ethereal (Packet Capturing tool)

12. To implement the approach on the server end to detect such an attack in progress and prevent the attack.
   12.1.1 To demonstrate the above we will be using a simple website with a login page which exchanges session cookies to keep a session alive and store information about the session until the user terminates the session(by logging out of the website).
   12.1.2 A website with login page
   12.1.3 A web server that responds with a session
   12.1.4 A web browser that communicates between the client and the server to display the contents of the requested page.
4. Literature Review

Session Hijacking has always been a most efficient attack, which makes it even harder for the programmers and developers to create a very strong authentication process. There are several research papers on how to detect session hijacking and counter measures to prevent such attacks. But most of the detection methodologies are to develop an expensive tool, for example: IDS. Most researches till date have always strongly leaned towards the Intrusion detection system side and built different tools or mechanisms that when suspected an intrusion will alert the network administrators of that infrastructure. The approaches involved in these tools are to monitor the network for ARP poisoning or IP spoofing. The most common IDS system is the monitors MAC frame sequence numbers, because the attacker has to correctly guess the sequence number and will be able to gain the session information and hence get access to the user information.

Here are some of the recent developments on detecting a session hijacking:

I. Analyzing the abrupt changes affecting the signal strength on a network due to a session hijacking attack [6].

II. Drawback of Certificate authentication which leads to a vulnerability on the server end and which eventually will become prone to Https session hijacking [7].

III. CookieMonster is a tool that evaluates the session cookie granted for its strength and then archives the cookies (if found tampered) for analysis, if the archived cookies are analyzed then we will be able to figure out if the session was hijacked or not.

The recent development on the “CookieMonster” is an effective tool to detect any uncertainty on the network due to session hijacking attack. In this project we will be taking a different approach in handling the session information between the server and the client. In general when a programmer writes a code for a session, the attributes that usually goes into the code are:

- Session ID or Unique Identifier (UID)
- Data source information
- Username
- Password
- Idle timeout details and etc,

These are some of the typical information that is available in a session cookie. If the attacker has access to this information he will be able to hijack that session and take control over the user session, if it’s an active session hijacking the attacker might even be able to kill the user connection and take complete control over the session. The approach we are trying to achieve is to add additional security to the existing session information that is traversed between a
typical client and a server over the network. The more attributes a session carries the more
the harder it becomes for an attacker to crack or hijack a session.

On an unencrypted network (for e.g. http connection) the attacker can capture the packet over
the network and gain valuable information which can used to hijack a particular session. But
by adding additional information for the session cookies that are created by the server we can
assure that attacker will not be able to easily crack the session information. In this approach
we will try to achieve the following to give a secure session to the user:

✓ Adding the exact time of the session created on the server.
✓ Adding the user agent string usually the browser fingerprint.
✓ Adding the IP address of the user.

By using the exact creation time of the session, it gives an additional layer on the session
cookie which is not easy to manipulate with. The browser fingerprint strategy helps us
analyze the details of the originate user, this will also be helpful if we need to track back to
the user computer if it involves a cyber crime. Finally adding the details of the IP address
gives us an added advantage to compare the IP address of the request originating from the
user computer and the IP address of the attacker for the same session, if the IP address are
different then we can safely assume that the session is hijacked and hence truncate the
incoming request from the attacker.

This additional layer of security in the way session cookies are stored and served will allow
the users to be protected from any harmful session hijacking attacks. This will also allow the
server administrators to look out for such attacks and safeguard the servers from begin
vulnerable to such attacks.

5. References

   ARP-Part3.html the link was last accessed on 29/7/2011 at 12:26PM.

2. http://www.sans.org/reading_room/whitepapers/windows/session-hijacking-windows-
   networks_2124 the link was last accessed on 29/7/2011 at 7:08PM.


4. Andrew Whitaker and Daniel Newman [2006], “Penetration Testing and Network
   Defense”.


- 59 -
6. Xiaobo Long and Biplab Sikdar, Senior Member, IEEE, "A Mechanism for Detecting Session Hijacks in Wireless Networks" IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, VOL. 9, NO. 4, APRIL 2010


9. http://stackoverflow.com/questions/616545/php-sessions-useragent-with-salt the link was last accessed on 31/07/2011 at 5:30PM

10. http://shiflett.org/articles/session-hijacking the link was last accessed on 31/07/2011 at 6:30 PM.
Appendix – D: Project Proposal Form

MSc Project Proposal Form (PPF)

AY10/11, Semester 3 (Jun 2011 – Jan 2012)

<table>
<thead>
<tr>
<th>Student Number</th>
<th>1031051</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Name</td>
<td>Jerry Louis</td>
</tr>
<tr>
<td>Degree Course</td>
<td>MSc Computer Security and Forensics</td>
</tr>
<tr>
<td>Supervisor Name</td>
<td>Dr. Xiaohua Feng</td>
</tr>
<tr>
<td>Title of Project</td>
<td>Detection of Session Hijacking</td>
</tr>
</tbody>
</table>

Description of your artefact

Session hijacking is a type of attack where the victims (users) session details like session ID, cookie information of that session, etc are obtained to take control over that specific user session and use it for malicious purposes.

My artefact is to investigate various methods and techniques for detecting Session Hijacking.

The aim of this project is to develop an approach that will detect a session hijacking and create awareness amongst the users about Session Hijacking and educate the users to exercise different techniques from becoming a victim of such an attack.
<table>
<thead>
<tr>
<th>What methodology (structured process) will you be following to realise your artefact?</th>
<th>The methodology involves demonstrating a Session hijacking on the network virtualization tools like Opnet or Cisco packet tracer or other lab equipments including laptops/desktops, network tools/applications, etc. And finally use the proposed approach to detect a session hijacking in-progress and prevent such an attack.</th>
</tr>
</thead>
<tbody>
<tr>
<td>How does your project relate to your degree course and build upon the units/knowledge you have studied/acquired</td>
<td>This project is part of my Computer Security Module.</td>
</tr>
<tr>
<td>Resources</td>
<td>Journals, research papers, that will provide comparative knowledge of the topics involved. Internet based research to develop the tool used for detection.</td>
</tr>
<tr>
<td>Have you completed &amp; submitted your ethics form?</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Detection of Session Hijacking

Jerry Louis
1031051
MSc: Computer Security and Forensics
University of Bedfordshire
Jtar_hai@yahoo.co.in

Abstract
Session hijacking is a critical threat in the current day and age of electronic commerce. To prevent users from unauthorized access to the network, security applications have been developed to detect session hijacking. This project describes the implementation of an application to detect session hijacking. The implementation of this application was done using Python and Java. The application was tested on a sample network to check its effectiveness. The results of the testing were analyzed and are presented in the following section.

Implementation: The Network Strategy
A network is a collection of devices connected together for the purpose of sharing resources. In this project, the network consists of a server and several clients. The server is connected to the Internet, and the clients are connected to the server through a local area network. The application was tested on this network to check its effectiveness.

Results
The results of the testing were analyzed and are presented in the following section.

Conclusions
The results of the testing were analyzed and are presented in the following section.

References