The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.78 e-ISSN: 2251-7545

MITM Attack Detection on Computing Networks

Xiaohua Feng and Jerry Louise
Department of Computer Science and Technology, University of Bedfordshire, Luton LU1 3JU, UK
Corresponding author Xiaohua Feng Department of Computer Science and Technology
Xiaohua.feng@beds.ac.uk

Abstract— **Session hacking is one of the important issues in computer security. Here, a new framework is proposed to solve this kind of MITM attack-detection in computer network.**

Keywords: WIRELESS COMPUTING; MITM ATTACK, SESSION HACKING; CLOUD BREACHES; INTRUTION DETECTION and COMPUTER SECURITY TOOLS.

I. Introduction

The fast growth of computing applications especially with cloud computing developments has increased the demand for the reliability and safety equivalent to the provided services, such as networked workstations.

According to security report from Norton [2011] and 7Safe [2010], effective prevention of session hijacking in Computing Communications is still not optimal yet. The existing IDS tools are not quite effective on this type of breaches, especially in wireless computing networks or cloud environments. If an attacker retrieves any sensitive data about a client, this information could be used to get access into the user account and carry out any malicious attacks.

A new framework has been proposed [Louise, et al. 2012] in order to sort out this kind of breaches on networked-computing based infrastructures, to achieve an optimum detection. A serial of implementation experiments have approved an optimal output currently. It also represents an analysis based on our practical testing results of the implementations.

II. Related Work

A. Background

In Computer networked systems, MITM (Man in the middle) breaches cannot be detected as easily as DDoS (Distributed deny of services) attacks. Current IDPS (Intrusion detection and prevention systems) are not pacifically to this kind of attacks; especially when an attack happened in a cloud environment [Feng, 2006]. As one type of MITM attacks, session hijacking detection and prevention is still outstanding. There are some incidences had already happened. Statistics shown that a few attackers had successfully breached existing networks [Norton, 2011]. According to Zheng, Poon and Beznosov [2009], there is still no any sufficiently efficient tool in terms of successfully detect and prevent MITM attacks optimal.

In this paper, a new framework has been proposed with an incoming detection method and an outgoing detection method [Louise, 2011] in order to detect session hijacking effectively therefore to prevent further damages. In particular, for incidence in cloud environment such as banking systems, commercial achieve systems and insurance or other sensitive data transition, detection session hijacking could avoid risks even financial disasters.

Meanwhile "HijackThis" has also been released recently; it is suitable for Windows based systems. Nevertheless, with a hybrid wireless and wired computer cloud systems [Feng, 2012], our methodology [Louise, 2012] is more appropriate.

B. The Framework

Through research, we think in terms of session hijacking, the most appropriate methodology is to having both network incoming detection methods and outgoing detection methods. This is the only effective way currently. The late implementation has proved our hypothesis has met the prediction. The details are demonstrated as follows.

If take network data stream spoofing/sniffing into account, utilize the feature/function of NIC (Network interface card) into a network. This became a effective way to solve session hijacking in our research.

Figure 1.1 Accepts all the packets Mode

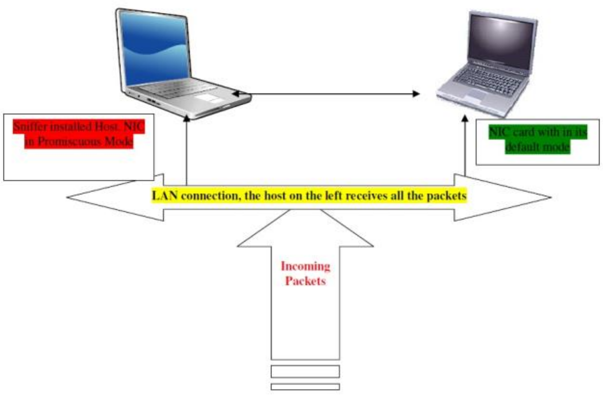

Sniffer installed Host. NIC in Promiscuous Mode
NIC card with in its default mode
LAN connection, the host on the left receives all the packets
Incoming Packets

Figure 1.1 presented a proposed environment that these software working on. Next I will show our methodologies.

III. The Methodology

Two methods have been proposed in this development as illustrated as follows.

A. Incoming Method

An artifact has been developed in the network environment. It is named as an incoming method,

as it works in client side of the application. Exploit .Net applying visual basic can produce a raw socket appliance. The raw socket appliance can provide a tactic. This approach will produce a customised packet, which is an advantage needed at the client side. There is a legitimate IP address in the customised packet produced. However, there is also a false broadcasting address with it in this packet. For instance, FF:FF:FF:00:FF:F0. If a device in a network accepts the false broadcasting address; consequently, an alert will be raised. In this data transmission, there is a machine which NIC (Network interface card) is in an unauthorised mode. Consequently, incidence response strategy starts working.

B. Outgoing Method

An artifact has also been developed in the

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: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.

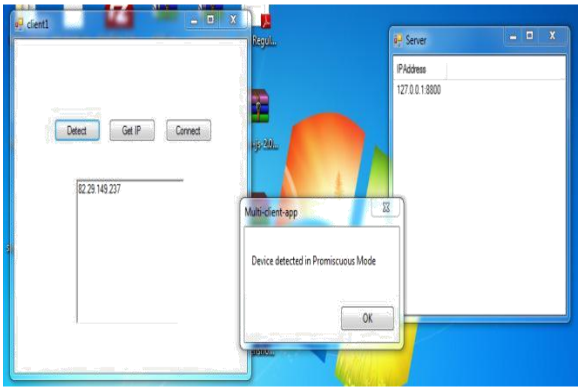
In the OUT-Network strategy, a valve will be introduced as a filter into the Apache Tomcat server. This filter will be used to run a loop in the server which will check each host IP address, even if the session has been compromised by an attacker but the information used by the attacker to get the contents 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.

IV. Applications

A. Incoming Method Network Application

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. 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 Figure 4.1 screenshot.

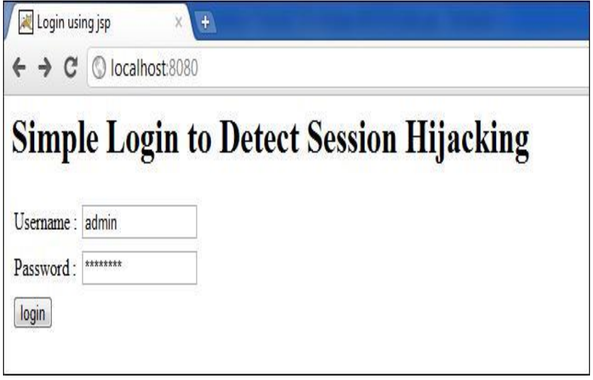


Device detected in Promiscuous Mode

Figure 4.1 Incoming Method example

B. Outgoing Method Network Application

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 Figure 4.2 below screenshot 2 a webpage which is connects to the Apache Tomcat server.



Simple Login to Detect Session Hijacking

Username : admin

Password : ********

Figure4.2 Outgoing Method 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.

The Figure 4.2 has described the scenario and demonstrated the proposal. Next an analysis will be carried out about this.

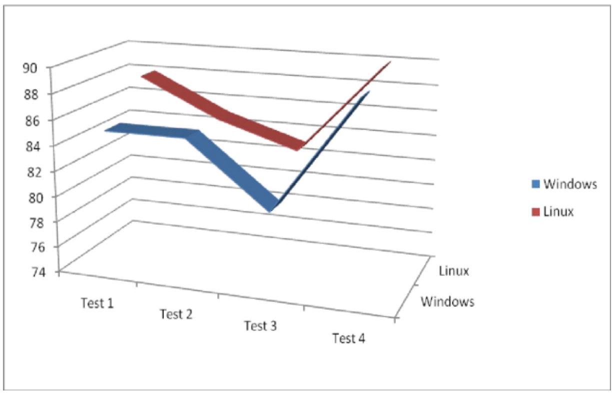


Figure 5.1 A Comparison of Delectability between Multiple Operating Systems (%)

V. Evaluation

According to our testing result, as demonstrated in Figure 5.1, the existing IDS tools are not as effective as ours on this type of attacks. If an attacker discovers any sensitive data, this information could be used to get access into the user's account and carry out any malicious attacks. In the contrast, our new software tool can effectively detect MITM attack-- session hijacking earlier in Computing Communications networks in order to prevent further damage in the network breach. Up to date, our software is the best in comparison with others to this kind of security attacks. Figure 5.1 also shown 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 the 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.

VI. Conclusions

In this paper, we presented the new model and analyzed its application in for cloud forensics. There are some significant features already emerged during shown in the investigation process. However, D due to their complexity in the subject area, further a full analytical comparison with to existing analytical work is still in progressing and will be reported next soon.

Finally, our new proposed MITM attack detection tool has been represented. Nevertheless, there are still many technical challenges such as the prevention of MITM in security field need to be sorted; our further work is in progress.

References

[1] 7Safe and UoB (2010): *UK Security Breach Investigations Report*, An Analysis of Data Compromise Cases, 7Safe, U.K. 2010

[2] Adida Ben (2008): *SessionLock: Securing Web Sessions against Eavesdropping*. IEEE World Wide Web Conference Series, 2008.

[3] Feng, X, Yue, Y and Han L. (2012): *Digital Forensics and Ethical Hacking*, the 8th HEA Forensics teaching workshop, Sunderland, UK November 2012.

[4] Long, X. and Sikdar, B. (2008): *Wavelet Based Detection of Session Hijacking Attacks in Wireless Networks*. Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008.

[5] Louise Jerry (2011): Detection of Session Hijacking, University of Bedfordshire, Luton LU1 3JU, UK

[6] Louise Jerry, Feng X. and Epiphaniou G. (2012): *THE NEW FRAMEWORK ON COMPUTER HIJACKINGS DETECTION*, the University of Manchester, ARSR2012, the 1st Conference on Applied Radio Systems Research, Manchester, the 29th -30th May 2012.

[7] Norton (2011): *Norton Cybercrime Report* 2011, U.K.

[8] Noiumkar P. and Chomsiri T. (2008): *Top 10 Free Web-Mail Security Test Using Session Hijacking. Convergence and Hybrid Information Technology*, 2008. ICCIT'08. Third International. Conference, pp. 486 - 490

[9] Pauli J. Engebretson P. Ham M. and Zautke M. (2011): *CookieMonster: Automated Session Hijacking Archival and Analysis*. Information Technology: New Generations (ITNG), 2011 Eighth International Conference, pp.403 - 407.

[10] Robert D. G. (2009): *Errata Security*. Retrieved January 28, 2012, from Errata Security: http://erratasec.blogspot.com/2009/03/hamster-20-and-ferret-20.html.

[11] Securiteam (1998): *Beyond-Security*, Securiteam Retrieved, January 26, 2012, from Securiteam. http://www.securiteam.com/tools/2LUQLQ0RPO.html

[12] Whitaker Andrew and Newman Daniel P. (2006): *Penetration Testing and Network Defense*. Cisco Press.

[13] Zheng O. Poon J. and Beznosov K. (2009): *Application-Based TCP Hijacking*. IEEE, 2009.