Analysis of Cookie Based Session Management in HTTP and Security Concerns

Supriya Gupta

Department of Computer Sciences. Govt. P.G. College, Rajouri, J&K, India

Abstract: Web-based applications often use cookies to maintain state in HTTP protocol. However the uses of cookie have grown far beyond their original intention. Cookies are used to store login information, to track pages visited on a site, to store user preferences, to collect personal user information etc. Because cookies are implemented as clear text, they may be compromised easily; any sensitive information that is conveyed in them is exposed to intruders. This paper presents the security vulnerabilities in cookie based session management and expose various threats that cookies pose to information security. It also demonstrates how browser add-ons, proxies can be used by a malicious intermediary to alter the HTTP headers as they travel in either direction to impersonate legitimate user sessions.

Key words: Cookies; State; Security; Intruders; Vulnerabilities; Proxies

Introduction

HTTP is a stateless protocol [1]. This means that each request for a page is treated as a new request by the server. As a result, information from one request is generally not available to the next request. The stateless nature of HTTP was a serious problem in developing shopping cart, webmail and other interactive applications. The solution was the addition of a new technology called state management using cookies that allows the state of a client session with a server to be maintained across a series of HTTP transactions. A cookie is a small text file that is saved on a user's hard drive by a Web server to store information about a particular user or session. To initiate a session the server returns an extra response header to the client, Set-Cookie [2], in its response message, and the user agent returns the unchanged cookie information in a Cookie header in subsequent requests to the origin server if it chooses to continue the session. The server may choose to include a new cookie with its responses, which would supersede the old one and it relies on the client to save the server's state and to return it on the next visit. By receiving back the cookie, the server is able to identify the user and retrieves the user's session from the session database; thus, maintaining the user's session. A cookie-based session ends when the user logs off or closes the browser. The value of the cookie is typically chosen pseudorandomly. Any data associated with the session, such as the session id of current application user, a database key, the session state itself, etc are stored on the server using the cookie's value as an index. The uses of cookie have grown far beyond their original intention. Cookies are used to store login information so that users don't have to keep entering name and password each time they visit, to track which pages visited on a site, to store user preferences, to collect personal user information etc. The information in the Set-Cookie2 and Cookie headers is unprotected [2]. As a consequence: 1) Any sensitive information that is conveyed in them is exposed to intruders. 2) A malicious intermediary could alter the headers as they travel in either direction, with unpredictable results. This paper analyzes the cookie based session management and explores various security issues associated with it.

*Corresponding author(s):
mangotwin22@gmail.com (Supriya Gupta)

Experiment

A particular kind of session hijacking attack called sidejacking [3] is examined in this paper that involves sniffing cookie information and using it to impersonate a user and gain unauthorized access to a web-based service. By forging the session, an attacker can impersonate a valid client, and thus gain information and perform actions on behalf of the victim. In almost any cookie-based web application and webmail programs like Google's Gmail, Microsoft's Hotmail and Yahoo Mail, users first authenticate using an HTML form, if the user entered correct credentials, then a browser cookie is set to track the session. By stealing the already authenticated session cookie and replaying the same back to the Gmail server, an attacker can easily log into the Gmail account without the need of any user name and password. The victim continues to use his/her session unaware that somebody else is also in his/her account. The attack continues till the user log off.

In our experiment we performed a session hijacking attack by intercepting the communication of a user logging into his Facebook account. Using this intercepted communication we impersonated that user and access his account from our attacking machine. The attacks were tested on the wireless network of University of Jammu to ensure that they work as expected. One of the nodes on network with IP address: 172.18.223.213 and MAC address: 00-21-00-59-1E-0F was chosen as attacker. The IP address of the default gateway was 192.170.1.1 and its MAC address was 00-0D-ED-6C-F9-FF.

An open source packet analyzer, Wireshark, was used to sniff all the traffic of the victim as he browses Facebook. In order to capture the right packets ARP cache poisoning [4] technique was employed. Once the traffic of the victim browsing to Facebook was captured, the information in the cookie header was copied to a file. Figure1 shows the cookie captured.

With our HTTP data intercepted and prepared for use, we used Webscarab, a web security application testing tool, to actually execute the attack. WebScarab has been developed as open source by the Open Web Application Security Project[1] (OWASP). It serves as a proxy intercepting web browser web requests and web server responses, allowing the operator to review and modify requests created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. WebScarab is able to intercept both HTTP and HTTPS communication. WebScarab defaults to using port 8008 on localhost for its proxy. Once the proxy settings had been applied, the Facebook is accessed in the browser. Using WebScarabs' Edit request the cookie header was replaced with that of victim's cookie which was saved previously. Figure 2 shows modification of cookie in HTTP header. After accepting

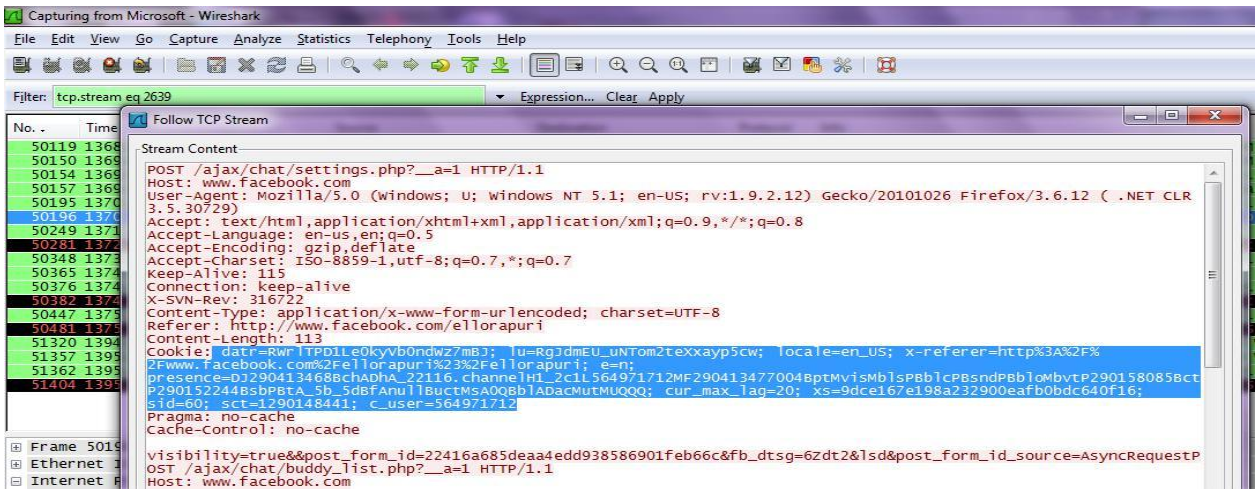

Figure 1: Capturing Victim's Facebook Cookie

changes we logged in to the victims Facebook account without need for user name and password. Figure 3 shows the victim's Facebook account. Once we get into the account we can perform any task like changing status information, send messages etc. as the victim just replaying that cookie again.

Figure 4 shows changing victim's profile settings. browsing you will need to save the two tools straight out of the pet store, named Hamster and Ferret. Both tools can be downloaded from the link "http://hamster.erratasec.com/". These are both command-line tools so the hamster folder can be extracted to an easy to get to location.

Alternatively, you can download and use Backtrack 4. BT4 is a Linux live-CD distribution designed specifically for hacking and penetration testing that comes with a myriad of preinstalled and precompiled tools, with Hamster/Ferret being two of them. You can download BT4 from "http://www.backtrack-linux.org/". You will then find Hamster in the /pentest/sniffers/hamster folder. The screenshot examples used in the rest of this tutorial are taken from BT4.

The first step involved in this form of session hijacking is to capture the traffic of the victim user as he browses Facebook. This traffic can actually be captured using any packet sniffing application such as TCPDump or Wireshark, but in order to capture the right packets you will need to employ a technique such as ARP cache poisoning (discussed in the first article in this series).

Host broadcasts ARP packets to spoof ARP tables of all computers on the LAN in order to associate attacker's MAC address with the IP address of default gateway. Any traffic meant for that IP address would be mistakenly sent to the attacker instead. The attacker could then choose to forward the traffic to the actual default gateway or modify the data before forwarding it (man-in-the-middle attack). The attacker could also launch a denial-of-service attack against a victim by associating a nonexistent MAC address to the IP address of the victim's default gateway. While spoofing ARP tables, attacker's system can act as gateway (or ip-forwarder) without other users' recognition on the LAN. This attack makes possible many sorts of "man in the middle" attacks. The tool used in demonstrating and testing was WinArpAttacker[2].

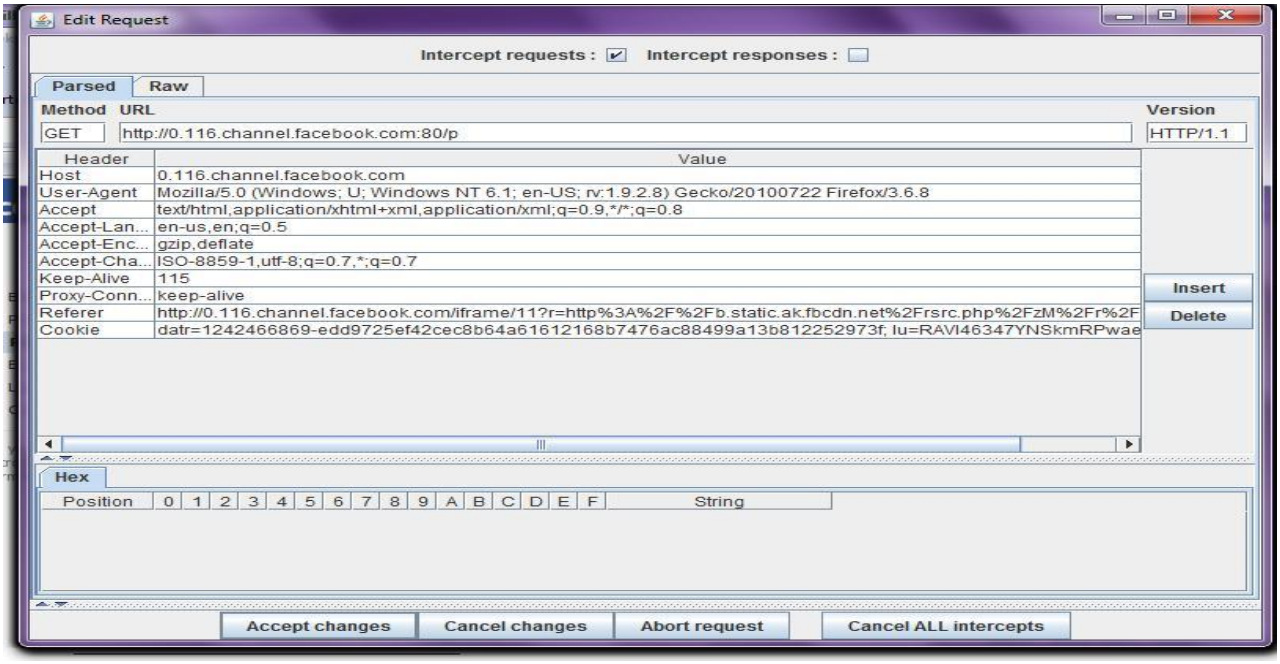

Figure 2: Altering HTTP Cookie Header

WinArpAttacker is based on wpcap3, so it requires wpcap driver before running it. WinArpAttacker provides a menu based approach to perform ARP cache poisoning and man in the middle attacks against switched networks. An open source packet analyzer, Wireshark, was used to sniff all the traffic on the LAN. These tools were used without any modifications in performing the attacks. The attacks were tested on the wireless network of University of Jammu to ensure that they work as expected. One of the nodes on network with IP address: 172.18.221.213 and MAC address: 00-21-00-59-1E-0F was chosen as attacker. The IP address of the default gateway was 192.170.1.1 and its MAC address was 00-0D-ED-6C-F9-FF. Both the tools were initiated on the attacker machine..WinArpAttacker all the active hosts on the network, were scanned and then the ARP SniffLan attack was initiated by sending gratuitous ARP reply packets , associating attacker's MAC address (00-21-00-59-1E-0F) with the IP address of gateway (192.170.1.1), to all hosts on the network and the underlying network traffic was analyzed using Wireshark. Figure 1 shows the packets captured by Wireshark as soon as the attack was initiated. Figure 2 presents the packet payload in detail.

On receiving an ARP response, all devices on the network updated their ARP caches replacing the MAC address of gateway with that of attacker (as seen in the response packet) though they had not sent an ARP request. The traffic sent to the gateway thus reaches the attacker machine. Figure 3 shows the packets received by the attacker as a result of ARP spoofing attack.

ARP Spoofing prevention and detection techniques

ARP cache poisoning problem is known to be difficult to solve without compromising efficiency. The only possible defense is the use of static (non-changing) ARP entries [6]. To prevent spoofing, the ARP tables would have to have a static entry for each machine on the network. The overhead in deploying these tables, as well as keeping them up to date, is not practical. Also some operating systems are known to overwrite static ARP entries if they receive Gratuitous ARP packets. Furthermore, this also prevents the use of DHCP configurations which frequently change MAC/IP associations. The second recommended action is port security also known as Port Binding or MAC Binding. Port Security prevents changes to the MAC tables of a switch, unless manually performed by a network administrator. It is not suitable for large networks, or networks using DHCP. The various other ARP spoofing prevention and detection techniques along with the issues in deploying them are discussed next.

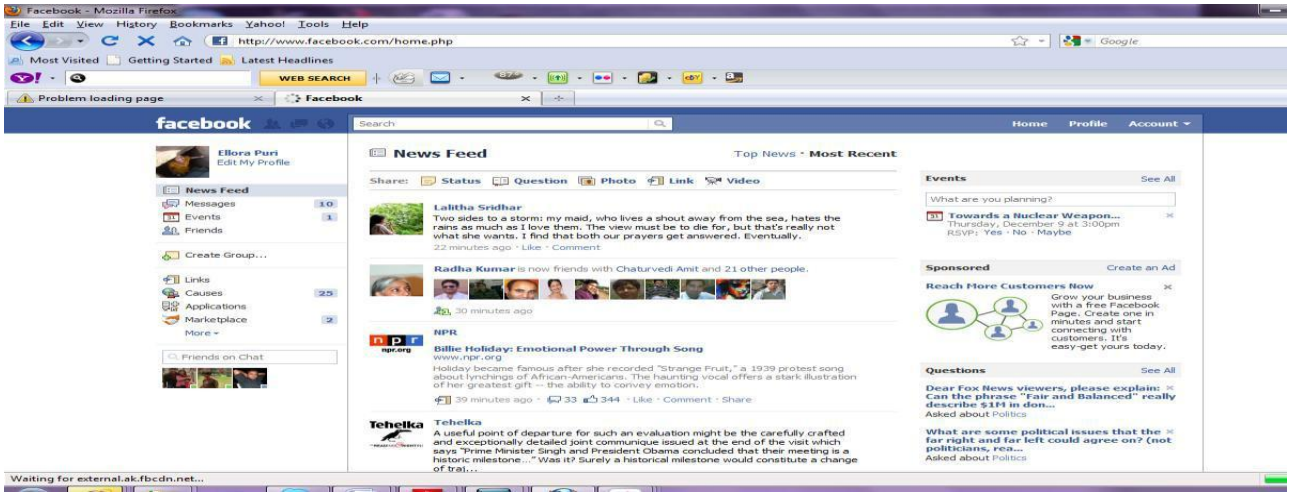

Figure 3: Successfully Hijacked Facebook account

A. Prevention Techniques: ARP spoofing prevention techniques are designed to avoid ARP spoofing attacks by covering up the potential vulnerabilities. The major vulnerabilities in ARP protocol are: 1) the ARP is stateless; 2) lack of authentication and 3) the broadcasting of ARP requests. Various prevention techniques have been suggested that address one kind of above mentioned vulnerabilities or the other. These techniques may employ changes in the design about how ARP requests are made and how they are permitted. The techniques are categorized based on the vulnerability they address.

1. Authenticating the Sender:

***a) Secure Address Resolution Protocol*:** Bruschi, Ornaghi & Rosti [7] suggested a secure version of ARP in which each host has a public/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. Messages are digitally signed by the sender, thus preventing the injection of spoofed information. It proposed a permanent solution to ARP spoofing but the biggest drawback is that it required changes to be made in the network stack of all the hosts. Moreover S-ARP uses Digital Signature Algorithm (DSA) that leads to additional overhead of cryptographic calculations. Goyal & Tripathy [6] proposed a modification to S-ARP based on the combination of digital signatures and one time passwords based on hash chains to authenticate ARP <IP, MAC> mappings. Their scheme is based on the same architecture as S-ARP, but its clever use of cryptography allows it to be significantly faster.

b) TARP: Lootah, Enck, & McDaniel [8] introduced the Ticket-based Address Resolution Protocol (TARP) protocol that implements security by distributing centrally generated MAC/IP address mapping attestations, which they called tickets, to clients as they join the network. The host with the requested IP address sends a reply, attaching previously obtained ticket and the signature on the ticket proves that the local ticketing agent (LTA) has issued it. The requesting host receives the ticket, validating it with the LTA's public key. If the signature is valid, the address association is accepted; otherwise, it is ignored. With the introduction of TARP tickets, an adversary cannot successfully forge a TARP reply and, therefore, cannot exploit ARP poisoning attacks. But the drawback is that networks implementing TARP are vulnerable to two types of attacks – active host impersonation, and DoS through ticket flooding. Furthermore an attacker can impersonate a victim by spoofing its MAC address and replaying a captured ticket but as long as the ticket is valid.

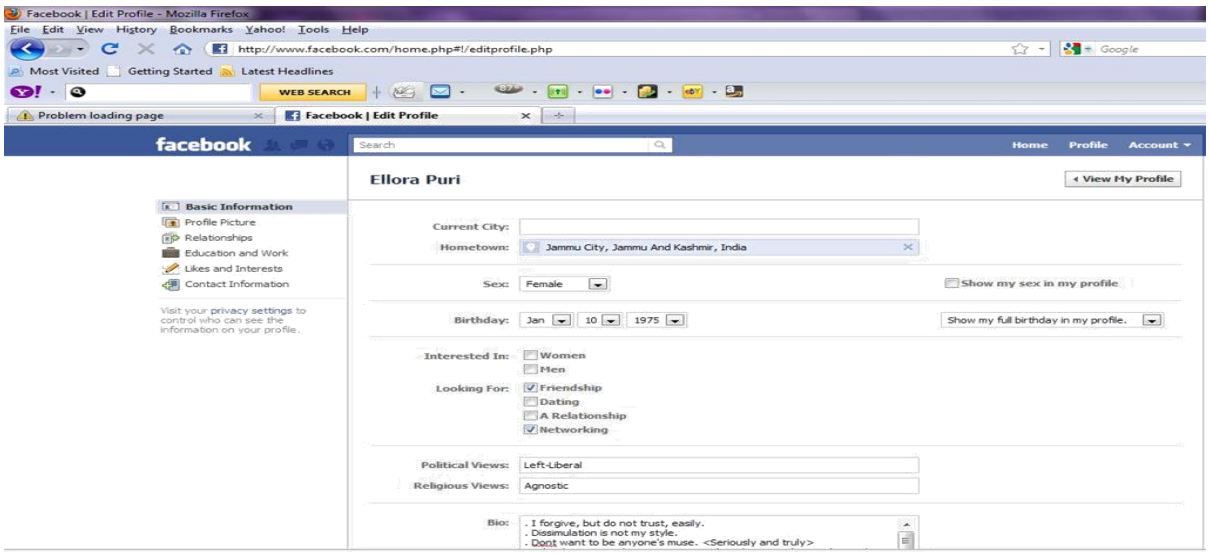

Figure 4: Changing Victim's Profile

c) Deploying a Virtual Private Network (VPN) to provide authentication and client-to-gateway security of transmitted data also provides a partial solution. On a VPN protected network an attacker can still redirect and passively monitor the traffic via the ARP based attacks, but he can only gain access to an encrypted data stream. Attackers still have the ability to cause a denial of service by feeding bogus data into the ARP caches of clients, but the compromise of data will no longer be an issue

2. Statefullness:

a) Some developers attempted to add protection to the IP stack on the terminal devices. The Antidote patch [9] requires a machine to send a request to the previous MAC address before changing an ARP entry. The machine will only change the entry if the request to the previous address is not answered. Again, this approach does not give any real protection against damage as the attacker simply needs to ensure that the attack occur when the machine with the previ-ous MAC address is down or unreachable. Also in the case of heavy load, the patch can actually cause communication to these systems to fail. Another similar approach, Anticap [10], does not allow updating of the host ARP cache by an ARP reply that carries a different MAC address then the one already in the cache. This unfortunately makes it drop legal gratuitous ARP replies as well, which is a violation to the ARP protocol specification [1].

b) Fuzzy logic approach: Trabelsi & Hajj [13] proposed a solution in which the prevention mechanism is based on the use of a stateful ARP cache that uses fuzzy logic approach to differentiate between normal and malicious ARP replies. Each host in the network collects two numerical values describing the Trust Level (TL) and Importance (Im) of each host, where the Im factor is calculated as the percentage of communication with a host and the TL is assigned some initial value that decreases exponentially if the host is an attacker and increases linearly otherwise. The collected information is stored in a database. Later on, it will be used to classify certain hosts as attackers or honests. This technique is not so effective practically as it is not adequate to consider that a host is not an attacker just because of high percentage of communication with that host.

3. Unicasting ARP requests

a) Using Central ARP server: Tai et al. [11] proposed an improved ARP in which the ARP request packets are not broadcasted but instead unicasted to an ARP server which will have all the <ip, MAC> mappings of all the hosts connected to the network. This significantly reduces ARP signaling and processing overhead. In order to grab the mapping of <ip,MAC> of any host, all packets transferred between each host in the network are listened and try to build up the ARP table based on the DHCP messages passed between each host and the DHCP server. But this approach requires continuous scanning of DHCP messages in order to update the ARP cache in case there is the IP address of a machine changes. And the major drawback is that it will not be able to grab <ip, MAC> mapping of any host if DHCP is not enabled for the network.

B. Detection Techniques:

1. The Request-Reply Mismatch Algorithm [12]: In this algorithm a sniffer listens for ARP packets, keeping a table of pending requests keyed by MAC address. Entries are removed from the table when the matching reply arrives after a timeout period. If a reply is seen without a matching request being present in the table, the administrator is notified. This algorithm performs well for small networks but for large networks the algorithm may incorrectly consider an attack. This is a form of passive detection techniques in which the ARP requests/responses on the network are sniffed to construct a MAC address to IP address mapping database. If there is a change in any of these mappings in future ARP traffic then an alarm is raised to inform that an ARP spoofing attack is underway. The most popular tool in

this category is ARPWATCH [14]. The main drawback of the passive method is a time lag between learning the address mappings and subsequent attack detection. In a situation where the ARP spoofing began before the detection tool was started for the first time, the tool will learn the forged replies in its IP to MAC address mapping database.

2. Active detection: Ramachandran and Nandi [15] presented an active technique to detect ARP spoofing. Based on the rules derived from the correct behavior that a host's network stack should exhibit when it receives a packet, the inconsistent ARP packets are filtered. Then a TCP SYN packet is sent to the host to be authenticated. Based on the fact that the Spoof Detection Engine does/does not receive any TCP packets in return to the SYN packet it sent, it can judge the authenticity of the received ARP response packet. This technique is considered to be faster, intelligent, scalable and more reliable in detecting attacks than the passive methods.

3. Detection on switches via SNMP: Carnut & Gondim [12] used counters provided by SNMP management framework for packets in/out and bytes in/out flowing through each switch port to detect the ARP imbalance i.e. the difference between the ARP packets entering and leaving the port respectively. As the attacker resends nearly the same amount of packets through the very port it received, so they nearly cancel out. Only the packets the attacker issues during the poisoning component of the attack make this number positive. Host that is the most imbalance emitter determines a candidate attacker and that receives unreplied packets determine the candidate victim. The algorithm is easy to implement but the false positives rate is very high when implemented in actual network.

Conclusions

The paper described a method of ARP attack in detail. Also, in this paper, an extensive study of proposed solutions to ARP spoofing attacks is conducted. All the proposed detection and prevention techniques that are mentioned above have different scope and limitations. They are either insecure or have unacceptable penalties on system performance. Issues with implementing a solution have also been presented that can be used to assist security instructors in selecting an appropriate solution to be used for building secure LAN network. As a conclusion of the study, a basic method is suggested to categorize ARP spoofing prevention techniques.

Acknowledgements

The author is thankful to Prof. Abdul Karim, Head, Department of Computer Science, Govt. P.G College Rajouri, for his kind support.

References

[1] D. Plummer (1982). RFC826-An Ethernet address resolution protocol or converting network protocol addresses to 48 bit Ethernet address for transmission on Ethernet hardware.

[2] B. Fleck & J. Dimov (2001). Wireless Access Points and ARP Poisoning: Wireless vulnerabilities that expose the wired network. Cigital Inc. Retrieved from http://www.Packetnexus.com/docs/ arppoison.pdf as accessed on 02-04-2010.

[3] T. Ylonen (1996). SSH: Secure login connections over internet. Sixth conference on USENIX Security Symposium, Focusing on Applications of Cryptography, 6, 37-42.

[4] T. Dierks & C. Allen (1999). RFC2246-The TLS protocol

[5] S. Whalen (2001). An introduction to Arp spoofing. Retrieved from http://www.packetstormsecurity.com/papers/protocols/introtoarpspoo fing.pdf as accessed on 02-04-2010.

[6] V. Goyal & R. Tripathy (2005). An efficient solution to the ARP cache poisoning problem. Information security and privacy, Springer Berlin, 40-51. doi: 10.1007/b137750.

[7] D. Bruschi, A. Ornaghi & E. Rosti (2003). S-ARP: a secure address resolution protocol. 19th Annual Computer Security Applications Conference (ACSAC '03), pp. 66.

[8] W. Lootah, W. Enck, & P. McDaniel (2007). TARP: Ticket-based address resolution protocol. The International Journal of Computer and Telecommunications Networking, 51(15), 4322-4327.

[9] T.Ilya (2002). Arp spoofing defense. Retrieved from http://www.securityfocus.com/archive /1/299929 as accessed on 12-04-2010.

[10] T. Demuth & A. Lietner (2005). Arp spoofing and poisoning-traffic tricks. Retrieved from http://www.linux-magazine.com/w3/issue/56/ ARPSpoofing. pdf as accessed on 12-04-2010.

[11] J.L.Tai, N. A. Yahaya & K. D. Wong (2009). Address Resolution Protocol Optimization. Jurnal Kejuruteraan, 21, 11-20.

[12] M. Carnut and J. Gondim (2003). Arp spoofing detection on switched Ethernet networks: A feasibility study. In proceedings of the 5th Simposio Seguranca em Informatica (Symposium Security in Informatics), Brazil.

[13] Z. Trabelsi & W. El-Hajj (2007). Preventing ARP Attacks using a Fuzzy-Based Stateful ARP Cache. In proceedings of IEEE International Conference on Communications (IEEE ICC'07), Glasgow, Scotland.

[14] LBNL Research Group. Arpwatch tool. Retrieved from ftp://ftp.ee.lbl.gov/arpwatch.tar.gz as accessed on 20-04-2010.

[15] V. Ramachandran & S. Nandi (2005). Detecting ARP Spoofing: An Active Technique. Information security and privacy, Springer Berlin, 239-250. doi: 10.1007/11593980_18.

[16] Y.LIU, K.DONG & L. DONG, B. LI (2008). Research of the ARP spoofing principle and a defensive algorithm. WSEAS Transactions on Communications, 7, 516-520.

[17] Ryan D. Riley, Nada Mohaamed Ali, Kholoud Saleh Al-Senaidi & Aisha Lahdan Al-Kuwari (2010). Empowering users against sidejacking attacks.In Proceedings of the ACM SIGCOMM 2010.