Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/092L23

Proxy Server Authentication for Blocking HTTP-Cache-Poisoning Attacks

Wookey Lee[1], *Simon S. H. Park*[1], *Chasung Lim*[2], *Jinho Kim*[3,*] *and Sangwon Kang*[3]

[1] Informatics Engineering Lab., Dept. IE, Inha University, Incheon, Korea
[2] AhnLab Security Emergency Response Center, AhanLab, Seongnam, Korea
[3] Department of Computer Science, Kangwon National University, Chuncheon, Korea

Received: 27 May 2014, Revised: 26 Jul. 2014, Accepted: 27 Jul. 2014
Published online: 1 Apr. 2015

Abstract: E-commerce systems have usually been processed by credit cards and public certificate via web sites where the client passes through web proxy server or the route of proxy server. In these systems, private information such as credit card numbers and passwords need to be protected by SSL (Secure Sockets Layer) or TLS (Transport Layer Security) encryption. But private information is still vulnerable to sniffing attacks through changing certificates of proxy servers, which is called the attacking of SSL-in-the-middle proxy. This paper analyzes credit card security systems which are defenseless against the hacking of false proxy server. It also proposes an effective method for protecting against the attacks of authentication proxy server Man-In-The Middle.

Keywords: SSL, TLS, MITM, Proxy Server, HTTPS

1 Introduction

The current advances in information and communication technologies have been a driving force leading to comprehensive social change; in particular, tasks that are performed in the office can be completed anywhere and quickly via the Internet. However, despite its convenience, the Internet has detrimental effects.; Internet-based damages attacks such as cyber terror, worms and viruses , DDos (Distributed Denial Of service) attacks, and cracking occur continuously, and personal private information leaks by crackers hackers (e.g., the Auction incident in which the personal information of the website?s users was hacked) have emerged as socially serious issues.

Recently, in most electronic commerce websites, data are transmitted by encrypting shared channels between servers and clients through SSL (Secure Sockets Layer) and TLS (Transport Layer Security) [1][2][3][4] protocols, but this system cannot provide a perfect security solution. On the other hand, a proxy server that efficiently distributes loads through caching is widely used; however, it can be misused as MITM (Man-in-the-Middle) [5][6][7] tools by malicious programs or scripts. MITM attacks, which are programmed to sniff between a client and its server, can be variously applied in different ways, including by sniffing through falsified certificates, where encrypted confidential information is sniffed while being transmitted to the server.

In this paper, we analyze the risks that can occur when confidential information, which is transmitted to the web using a web browser, passes through a web proxy. Furthermore, we propose a new defense method against such risks.

The structure of this paper is as follows: Chapter 2 discusses the principle of a proxy server and the functions of proxy caching; Chapter 3 identifies the risks when a malicious program or cracker changes the proxy settings; Chapter 4 describes the proposed defense method against these risks; and Chapter 5 provides our conclusion.

2 Related studies

2.1 Web Proxy Server

A proxy server is used to provide functions to reduce network and web server loads and to increase service speeds for users [8][9][10] by temporarily storing the

* Corresponding author e-mail: jhkim@kangwon.ac.kr

access information of users in a proxy cache and allowing multiple users to exchange such information. In addition, the proxy server also functions as an authentication server in sections with limited bandwidth [11]. A proxy server is used mainly for web caching, a function that transmits the data stored in the web cache to the user without accessing the corresponding server when a web object, including HTML pages, is stored temporarily. This function can reduce bandwidth consumption, server load, and lag; however, it can also be abused for HTTP-cache-poisoning attacks [12]. In terms of service delivery to clients through a proxy server, the network environment in a client-proxy, server-web server communication form has greater LAN (Local Area Network) bandwidth than the WAN (Wide Area Network) bandwidth, and is widely used in schools and businesses, as shown in Fig. 1. In such an environment, the proxy server also functions as a firewall [13].

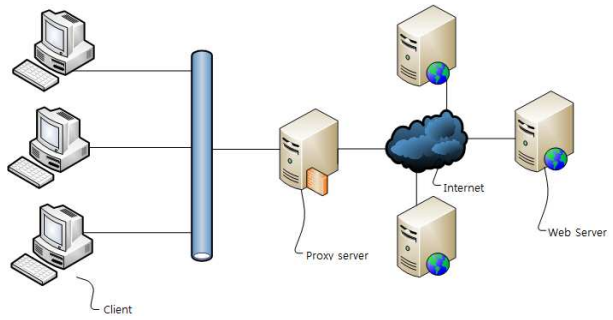


Fig. 1: Proxy Server

Web caching for HTTP-cache-poisoning can be classified into three cases, as follows: First, caching can be executed in a client application. For example, a virtual cache in the web browser improves the speed of web browsing, but it is inefficient because it is intended for a single client.

Second, as shown in Fig. 1, a proxy server is placed between the client and the server. Most proxy caches can be set up similarly to help increase the speed of web browsing by providing service to multiple clients. When a web document is requested from the proxy cache, such request is fulfilled without accessing the server if the document that is stored in the memory, disc, or other storage device can be sent directly to the client. If the response cannot be performed from the cache, the proxy server has direct access to the web server, obtains the requested web document, and stores it in the cache. If the proxy server does not have sufficient space to store the requested document, it must retrieve the document from the cache. Generally, in a cache replacement policy, an algorithm, which maximizes the rate of the documents that are successfully provided as part of services by the cache among the entire requested documents, is used. Because of their effectiveness, businesses rely on proxy servers; however, if the server is misused by crackers with malicious intentions, serious security problems can occur.

2.2 Web Proxy Server

SSL (Security Sockets Layer) was introduced in 1994 for safe communication on the Netscape web browser, and in 1996, the IETF (Internet Engineering Task Force) proposed SSL v3.0. Since then, SSL has been complemented and revised because of its security vulnerability, and in 1999, SSL was renamed TLS (Transport Layer Security), which was later standardized to RFC 2240 (TLS v1.0).

Located between application layers as shown in Fig. 2, SSL/TLS is easily encrypted with a variety of applications.; SSL consists of the Record Layer, Handshake Protocol, Change Cipher Spec Protocol, and Alert Protocol.

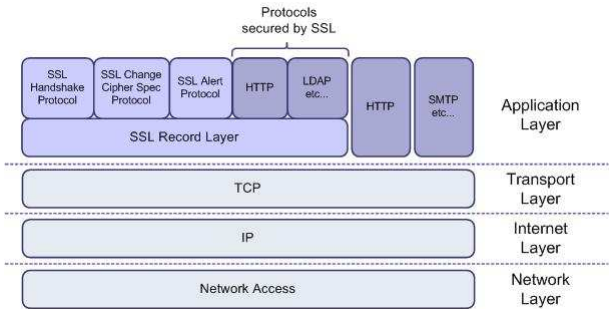


Fig. 2: Architecture of SSL/TLS

2.2.1 SSL/TLS Handshake Protocol

In the Record Layer, the client requests the server for a security parameter using the handshake protocol. Subsequently, the server establishes a parameter in response to the client's request. The parameters established by handshaking are activated to be used as the change cipher spec protocol, and data is protected and transmitted by SSL/TLS through the application data protocol. Any errors that occur in the communication process are managed by the alert protocol. In the process of performing the handshake protocol, the client and the server are mutually authenticated and the elements, including the encryption algorithm, encryption key, and MAC algorithm, which can maintain session conditions, are established. The handshake protocol is divided mainly into the Full Handshake protocol and the Abbreviated Handshake protocol; The operating processes of the

former, which are shown in Fig. 3 and, can be summarized as follows:

Step 1. The client sends a ”ClientHello” message to the server and waits for the server’s reply. If the server fails to send the ”ServerHello” message immediately, an error occurs and the connection fails.

Step 2. The client and the server present the protocol version, session ID, cipher suite, and compression method for Hello; each generates a random number and then exchanges the number with one another. In addition, the client and server share a pre-master secret and master secret that are required to generate a key through the server’s Certificate and *SecureKeyExchange*, and the client’s Certificate and *ClientKeyExchange*. The server can request a certificate from the client; in this case, the server waits for the client’s request after sending *ServerHelloDone*.

Step 3. If the server requests a certificate from the client for a certificate, the client must send it to the server. Information on the pre-master secret of the key exchange algorithm, which is selected in the Hello of the *ClientKeyExchange* message, should also be sent to the server. Moreover, the client runs the determined parameters through the change cipher spec protocol, and then sends the ”Finished” message to the server to complete the protocol.

Step 4. When the server receives all responses from the client, the server executes the change cipher spec protocol similar to the client, sends a ”Finished” message to the client, and completes the handshaking process.

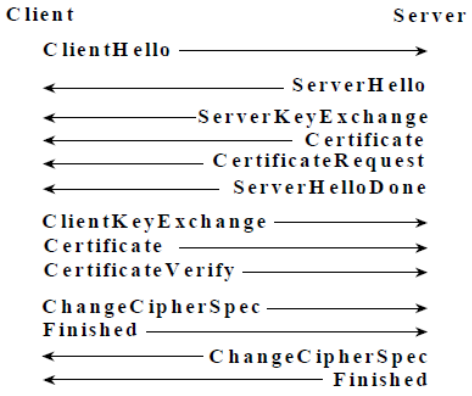


Fig. 3: Full Handshake Operating Procedures

When the client requires a new session, an empty session ID is sent. In the case of creating a new connection using the previously generated session, the client needs to send the ClientHello message to the server, including the session ID that it wants to reuse. However, if the server fails to find the session ID sent by the client, it sends the client an empty session ID with an error message. If there is a corresponding session ID, the client and the server exchange the change cipher spec using the Abbreviated Handshake protocol. Fig. 4 shows the operation procedures of an Abbreviated Handshake. If an existing session is reused through the Abbreviated Handshake protocol, the status of the existing session remains the same, and the change cipher spec exchanged between the server and the client is used in order to remain steady.

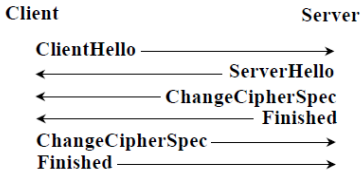


Fig. 4: Abbreviated Handshake Operating Procedures

3 Weaknesses of HTTP-Cache-Poisoning

In general, the client receives direct services through the web server if a temporary proxy has not been assigned by the administrator. However, if a malicious script or cracker modifies the settings, the proxy server can be abused and falsified for HTTP request sniffing; even an SSL-based security system is not free from confidential information sniffing with a falsified certificate [14][15].

3.1 Proxy Setting Changes Caused by Registry Falsification

Fig. 5 shows the basic process of our experiment. If a cracker modifies the registry value of a given client such that a proxy server is used, the client would then use the Internet through said proxy server and store the data transmitted through the web browser without noticing that its Internet use involves the proxy server. The cracker stores the transmitted data and retransmits it to falsify certificates or parameters [6]. The cracker can then log into the client’s account using the sniffed information. The parameters are variables that are sent in addition to data through the web browser. There are three types of proxy server cracking and changing settings.

3.1.1 Proxy Server Setting Adjustment by a Script

This section explains how a script adjusts the registry value. A cracker can change the victim’s registry value by leaving a message on a bulletin board or by sending an

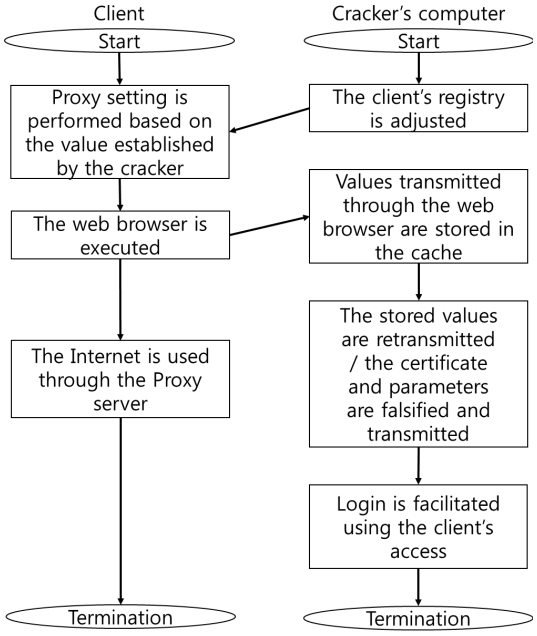


Fig. 5: Basic process of proxy cracking

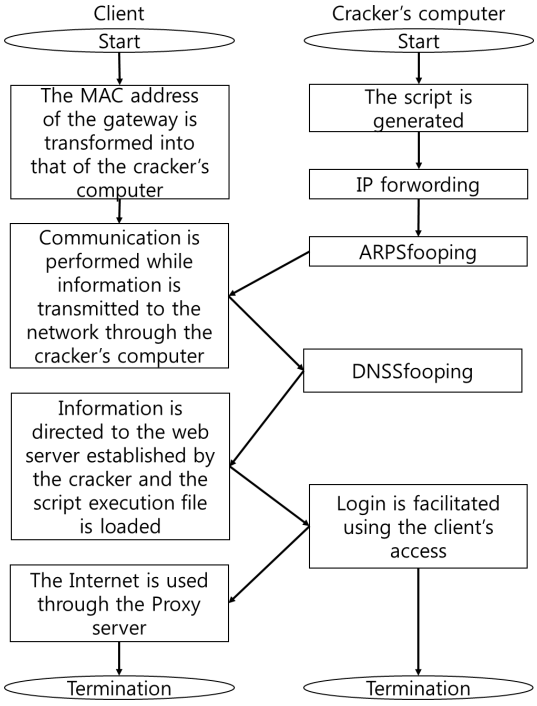


Fig. 6: Falsification of the proxy by ARP Spoofing

email, both of which contain an HTML code (e.g., <embed src="http://the cracker's IP address/name of the script execution file">) that executes the script execution file on the cracker's computer. If the bulletin board or email service bans writing a script for security purposes, an HTML code can allow the cracker to execute the script execution file on the cracker's computer. A script execution file is a file that forces execution of a script on a client's computer. (Examples are Flash files, VBScript, Javascript, etc.) The cracker uploads an HTML code on the bulletin board or sends the victim an email that contains an HTML code. The client opens the cracker's message on the bulletin board or email. Then, the HTML code is executed and the Flash file on the cracker's computer is loaded onto the victim's computer. The script execution file from the cracker's computer changes the client's registry value. The client receives services on the web through the proxy server that the cracker assigned.

3.1.2 Proxy Server Change by ARP Spoofing

This cracking method can be used only when the cracker and the victim are in the same network, i.e., the same switch. As shown in Fig. 6, a switch seeks the table that can store the MAC (Media Access Control) addresses of each port and create a broadcast segment by port. A MAC address is the physical address of a LAN card, and "broadcast" refers to the signal transmission to the entire network. The cracker creates a website in which a script is executed, and executes IP forwarding in order to relay the packet during ARP (Address Resolution Protocol) Spoofing. Then, the cracker performs ARP Spoofing to the client. ARP Spoofing notifies the client that the MAC address of the gateway is transformed into that of the cracker, such that the client falsely sends the packet to the cracker's MAC address. A gateway is a system that connects two different communication networks with two different structures. As shown in Fig. 7, once the client's computer is cracked through ARP Spoofing, the client's packet is sent to the cracker's computer, whereas the malicious forwarding remains unnoticed because IP forwarding was performed in the cracker's computer. The cracker's computer performs DNS Spoofing to the client. While being DNS Spoofed, the client's computer recognizes a DNS request when it sends the packet to Port 53, and sends a web request to the website on which the cracker's script is executed. Although the client attempts to access the desired web server, it unwittingly accesses the cracker's designated web server, and the HTML code that the cracker wrote calls the script execution file. The script execution file is executed in the cracker's computer, and the client's registry value is changed. Thereafter, the client receives all the requested information on the web through the cracker's proxy server.

3.1.3 Proxy Setting Change by Malicious Program

This section explains how a malicious program changes a given client's registry value. Fig. 8 shows the process of

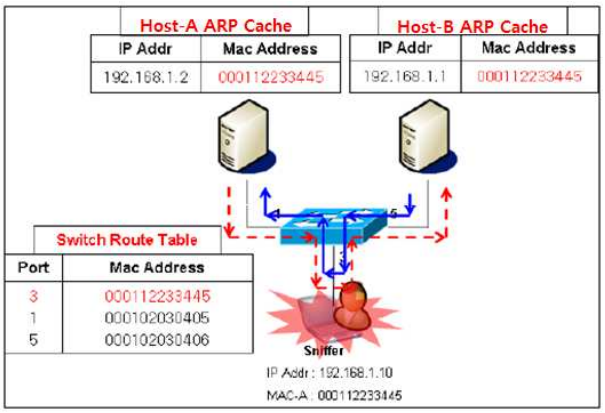


Fig. 7: ARPSpoofing ARP Reply

installing a malicious program on a given client using ARP Spoofing. Although the client requests the web server for clean web pages, the web server, infected by ARP Spoofing, sends the client falsified web pages. Then, by executing falsified codes such as <iframe>, the client accesses the server that spreads malicious programs, and unwittingly downloads and installs malicious programs.

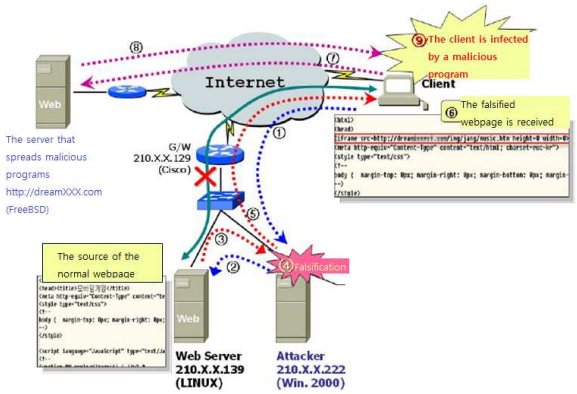


Fig. 8: Infection of a malicious program using ARP Spoofing

Fig. 9 explains how a malicious program infects the client's control part. For Windows operating systems, the registry value is adjusted to change the proxy setting to be in the "Enable" condition. The proxy server's IP address for the registry value is changed to the proxy server's address and port number of the cracker. There are two cases of registry changes in the example shown in Fig. 10. First, a binary file is a numeral system file that uses binary numbers represented uniquely by zeroes and ones in order to express data in the computer. Second, the ASCII (American Standard Code for Information Interchange) is the most common scheme for text files in the computer or the Internet. An ASCII file is defined with the English alphabet, numbers, and 7-bit binary integers (seven strings comprised of a combination of zeroes and ones) for special characters; a total of 128 characters are defined. For example, 0x00000001, the value of ProxyEnable, executes the proxy server, whereas 0x00000000 indicates "no execution."

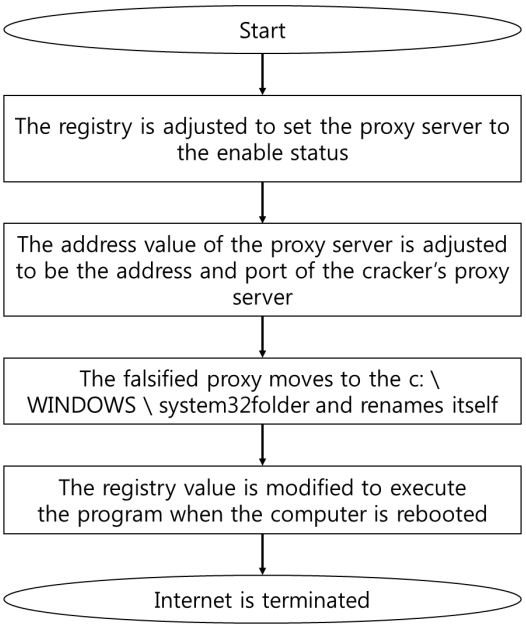


Fig. 9: Process of malicious falsification of proxy

Language C#

My Computer\HKEY_CURRENT_UXER\Software\Microsoft\Windows\CurrentVersion\Internet Settings
-> ProxyEnable REG_DWORD 0x00000001//(522)
-> ProxyServer REG_SZ 220.149.XXX.XXX:8080 //(523)

Fig. 10: Registry falsification code

The client, infected with a proxy falsification code, goes through the following process. When the client's registry value is changed by a malicious code or script, the communication is performed through the false proxy server. Because Windows basically allows a change of proxy settings, security solutions or virus detectors are unable to verify whether a change is a malicious attempt or the user's intention.

Once the client establishes a proxy, the client's web browser opens a given port, and the client receives services from the web server through the proxy server. At this time, the proxy server established by the cracker with the malicious intention receives and stores the values

requested by the client in the cache and relays them to the web server, instead of functioning as a generally used proxy server. Once the client stores the transmitted values in the false proxy server, cracking is facilitated. Furthermore, the cracker can falsify and send the client's requested data in the proxy server, such that redirection can be performed if requested by the website. For example, redirection occurs when the client requests to connect to "bank.com;" however, the proxy server does not direct the client to its desired website, but to "www.hacker.com," the cracker's desired website in which the end user ??the victim - can mistakenly leave personal information. The client is unable to notice the cracking because it is not aware of the existence of the proxy server. The malicious proxy server can sniff personal information by falsifying parameters, cookies, and certificates and by storing the client's footprint in the websites in the cache.

3.2 SSL Communication-based Credit Card Information Sniffing

The client whose registry values set up for a proxy server has been infected by a malicious code unavoidably sends personal information, including credit card information, CVC, and Verified by Visa password, to the cracker's proxy server when the client makes online purchases because such purchases are completed through the proxy server. The cracker's proxy server stores the information in the cache, as shown in Fig. 11 and Fig. 12, and then sends the client's personal information to the credit card company.

Although commercial security solutions run when users employs their credit card for purchases, this cracking attack manipulates the proxy server settings, which the commercial security solutions recognize as a basic Windows function instead of a cracking attempt; therefore; such an attack is unnoticed.

Fig. 11: Credit card number sniffing

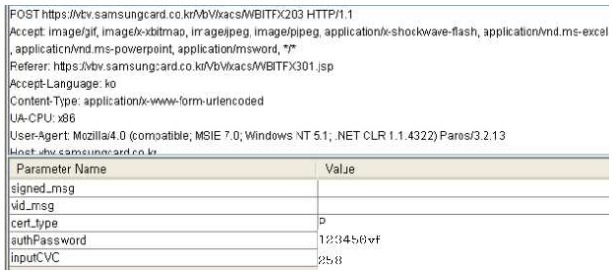


Fig. 12: Credit card information (CVC and PASSWORD) sniffing

3.3 Falsification of SSL-based Certificate using the Proxy Server

When the client that is infected by a cracking program, i.e., by receiving proxy services, makes a request to the web server based on SSL communication, the web server sends its public key to the cracker's proxy server, and in turn, the cracker's proxy server sends the client a falsified public key. Although the end user that received the falsified public key is warned that the public key has not been authenticated by a trusted party, because generally users are unaware of public keys, the user would press the confirmation button, encrypt values such as the ID and password using the falsified public key, and send such values to the web server. Because the values are encrypted with the cracker's falsified public key, the cracker's proxy server can decode the values. Prior to sending the values to the web server, the cracker's proxy server is able to encrypt these values with the web server's public key. Then, the end user logs in to the web server and performs the required actions, e.g., check emails, prior to logging out. Fig. 15 shows the process of falsifying the certificate in the proxy server.

Credit card information or passwords are encrypted using SSL prior to being transmitted. In other words, the information that a cracker attempts to crack is, generally, encrypted by SSL, such that the cracker can reduce overloading the proxy server if it sniffs the information using SSL in the proxy server, for which a malicious proxy falsification program executes the code as shown in Fig. 14; therefore, the cracker sniffs only the HTTPS information transmitted by the client.

Fig. 15 shows the Proxy Architecture that sniffs only the SSL information.

4 Prevention of HTTP-Cache-Poisoning Crackings

Methods to prevent HTTP-cache-poisoning cracking can determine whether the proxy server established by the user or the network administrator has been adjusted, and if so, such methods can terminate the session when the

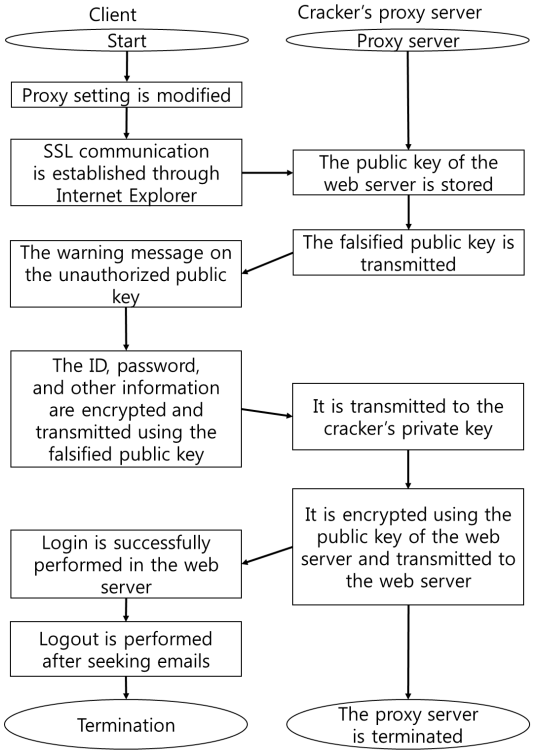


Fig. 13: Process of certificate falsification

Language C#

My Computer\HKEY_CURRENT_UXER\Software\Microsoft\Windows
\CurrentVersion\Internet Settings
-> ProxyEnable REG_DWORD 0x00000001//(522)
-> ProxyServer REG_SZ https=220.149.XXX.XXX:8080 //(523)

Fig. 14: Registry falsification code

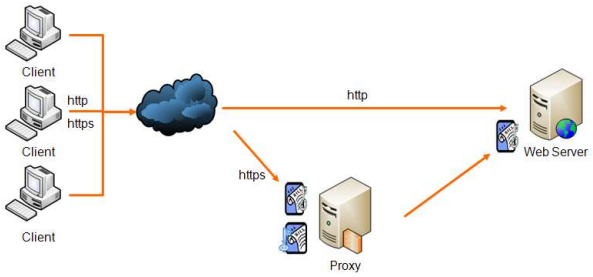


Fig. 15: Process of certificate falsification

data are transmitted to the proxy server that has been modified against the user's intention.

In order to verify proxy setting changes, a packet pattern analysis can help verify whether the data were transmitted to a proxy. Fig. 16 shows an analysis of the packet transmitted to the client. Unlike Fig. 17, this analysis confirms that data were transmitted to the Proxy-Connection server. If the user or administrator did not make such proxy settings, the session should be blocked to prevent confidential information from being transmitted to the server, along with a warning. If the proxy server settings made by the user or network administrator are in use, a digital signature using the RSA algorithm can be used to authenticate the proxy server prior to transmitting confidential information, preventing MITM attacks using the proxy server.

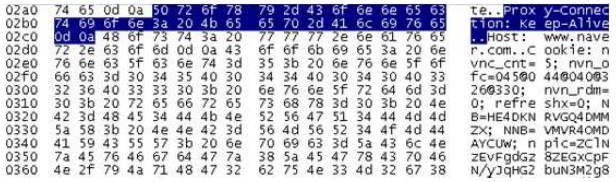


Fig. 16: Proxy-Connection

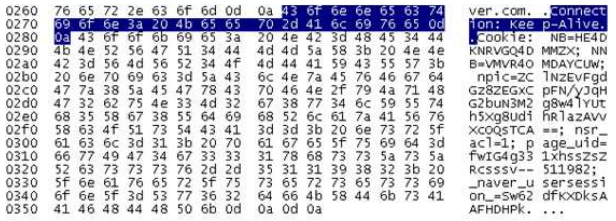


Fig. 17: Connection

4.1 Prevention of Unapproved Proxy Settings

In terms of the method to prevent cracking through proxy falsification, the integrity verification function that uses the hash function is required to confirm whether the client's registry proxy settings have been falsified by a malicious program. Moreover, a security solution should display a warning message prior to the transmission of crucial information via the Internet by determining whether the client's proxy server has been adjusted by a malicious program or script against the user's intention. The processes proposed in this paper are described in Fig. 18. When the client requests the server for a service, the server provides the client with the requested service, and if the proxy settings are in use, a warning is given to the client that the proxy settings are in use; moreover, the end user is notified about cracking risks prior to allowing the user to employ the random proxy server. However, many of the current security systems ignore the risks

associated with data transmission through proxy servers, as indicated in the example shown in Chapter 3. By verifying the integrity of proxy settings, displaying a warning message on proxy use before transmitting crucial information as suggested in this paper, and generally avoiding passing through a proxy server unless the proxy setting is allowed according to the user's intention, damages by MITM attacks can be prevented.

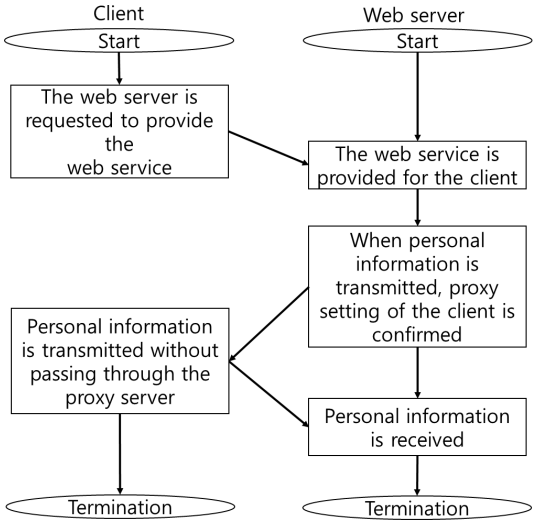


Fig. 18: Process of detecting unauthenticated proxy

4.2 Proxy Server Authentication

RSA (Ron Rivest-Adi Shamir-Leonard Adleman) is a cryptosystem that authenticates the proxy server using a public key algorithm. In the event that an attacker changes the client's proxy server settings, this verification algorithm confirms whether the changed proxy is approved by the administrator; if it is not, the session is blocked. In the network environment in which the administrator uses a proxy, as shown in Fig. 19, the proxy digests data using a hash function and then encrypts the value with a private key. The value that was encrypted with the private key is sent to the client in addition to the certificate; the client then extracts the public key from the certificate in the proxy and verifies the transmitted data against the certificate to confirm that the signature is valid. If the proxy is valid, the client sends its important data to the proxy.

The symbols in the protocol that our paper suggests are listed in table 1.

Fig. 20 shows the proxy authentication protocol.

Step 1. Proxy server S generates $P_{DH} = (G, p, g)$, a Diffie-Hellman [16] parameter. G is a cyclic group that possesses prime number p as a digit, and ϑ is the constructor of G. Proxy server S randomly selects b in $[1, p]$; therefore, Proxy server S calculates $Y_b = g^b \mathrm{mod} P$. Using the private key of RSA d_s, Proxy server S calculates a signature value $\sigma_s = Sign_{d_s}(P_D H, Y_b)$ and then transmits $(Cert_s, (P_D H, Y_b), \sigma_s)$ to Client C.

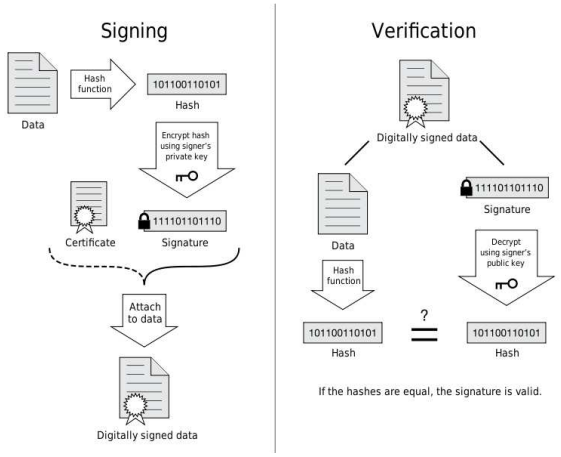


Fig. 19: Digital signature and verification

Table 1: Symbols in the protocol

Symbol	Description
Θ	Security parameter.
$PKE = (PK, PE, PD)$	Public key encryption algorithm. PK, a key generation function, generates a pair composed of a private key and a public key (e, d) by adding Θ. $PE_e(m)$, an encryption algorithm, uses e to output an encrypted statement c for a non-encrypted statement m. $PD_d(c)$, a decryption algorithm, uses d to output a non-encrypted statement m for a encrypted statement c.
$\Sigma = (Gen, Sign, Verify)$	Signature algorithm. Gen, a key generation function, inputs Θ to generate a pair composed of a signature key and a verification key (d, e). $Sign_d(m)$, a signature generation algorithm, uses d to generate a signature σ for a non-encrypted statement m. $Verify_e(m, \sigma)$, a signature verification algorithm, uses e to output 1 if the signature σ for the non-encrypted statement m is correct, and if not, to output 0.
c, s	IDs of Client C and Proxy P.
$b \xleftarrow{R} [1, p]$	b that is randomly selected among numbers ranging from 1 to p.
$Cert_s$	Public key certificate of S.
H	Hash function.

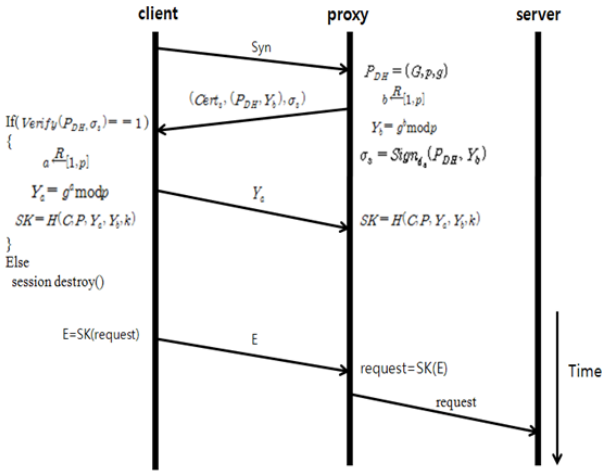


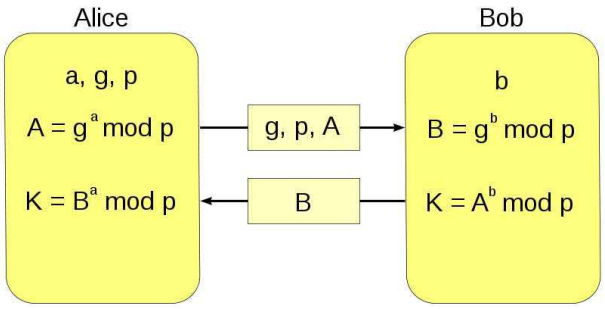
Fig. 20: Proxy authentication protocol

Step 2. Client C verifies $Cert_s$. If it is verified, Client C extracts a public key from $Cert_s$ and verifies Signature θ_s. If $Verify(P_DH, \theta_s) = 1$ is output, Client C confirms Proxy server S has not been falsified, and if the signature is invalid, the session is terminated.

Step 3. Client C selects a random number a in $[1, p]$ and calculates $Y_a = g^a \bmod p$, which is sent to Proxy server S.

Step 4. Client C and Proxy server S calculate a session key $SK = H(C, P, Y_a, Y_b, k)$ that can be used in a symmetric key encryption system.

Step 5. The values that are transmitted from Client C to Proxy server S are encrypted with Session key SK, prior to the transmission.



$$k = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

Fig. 21: Diffie-Hellman key exchange

Completeness. If all the protocols are executed without an error, Client C and the unfalsified Proxy server S generate the same key , as shown in Fig. 21, and exchange the same session key SK= . The client and the proxy transmit an encrypted packet to one another with the shared session key, which prevents MITM attacks between the client and the server.

5 Conclusion

If the client's network settings are modified by a malicious program or script, the client can be exposed to MITM attacks as the data is transmitted to an unintentional proxy server. In this regard, the connection between the client and the proxy server is likely established after ensuring a high level of security. However, as data are transmitted to the proxy server without an authentication process, end users that employ an Internet banking system cannot determine whether their confidential information is transmitted to a previously established proxy server.

In this paper, we analyzed the weaknesses of this proxy setting method and suggested a security function associated with client proxy settings, which are required as commercial banking security solutions to prevent proxy falsified cracking, as well as a proxy authentication protocol using a public key algorithm in order to fundamentally block proxy cracking.

Furthermore, future studies should be conducted on the authentication process of the web proxy server proposed in this study and authenticated proxies that can be applied to proxies in various fields should be conducted.

Acknowledgements

This work was partly supported by Inha University and by the Research Grant from Kangwon National University.

References

[1] Dierks, T., and Rescorla, E., "The TLS Protocol Version 1.1," RFC 4346, 2006.
[2] Oppliger, R., Hauser, R., and Basin, D., "SSL/TLS Session-Sware User Authentication Revisited," In Proc. COMPSEC, pp.64-70, 2008.
[3] Oppliger, R., Hauser, R., and Basin, D., "SSL/TLS Session-Aware User Authentication," Computer and Communication, **41**, pp.59-65, 2008.
[4] Oppliger, R., Hauser, R., and Basin, D., Rodenhaeuser, A., Kaiser B., "A Proof of Concept Implementation of SSL/TLS Session-Aware User Authentication," In Proc. KiVS, pp.225-236, 2007
[5] Burkholder, P., "SSL Man-in-the-Middle Attacks," SANS Institute, pp.15, 2002.
[6] Bringer, J., and Chabanne H., "Trusted-HB: A Low-Cost Version of HB Secure Against Man-in-the-Middle Attacks," IEEE Transactions on Information Theory, **54**, pp.4339-4342, 2008
[7] Petraschek, M., Hoeher, T., Jung, O., Hlavacs, H., and Gansterer, W., "Security and Usability Aspects of Man-In-The-Middle Attacks on ZRTP," JUCS, **14**, pp.673-692, 2008.

[8] William, S., Abrams, M., Standrige, C., Abdulla, G., and Fox, E., ”Removal Policies In Network Caches for World-Wide Web Document,” In Proc. SIGCOMM, pp.293-305, 1998.
[9] Korkea, M., ”Scalability in Distribured Multimedia System,” Master Thesis, Helsinki University of Technology, 1995.
[10] Cao, F., and Cao, Z., ”A Secure Identity-Based Proxy Multi-Signature Scheme” In Proc. ISCI, **179**, pp.292-302, 2009.
[11] Novotny. J., and Tuecke, S., ”An Online Credential Repository for the Grid: MyProxy,” In Proc. IEEE ISHPDC, pp.104-111, 2001.
[12] Klein, A., ”HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics,” Sanctum Inc, pp.31, 2004.
[13] Liu, A., Yuan, Y., Wijesekera, D., and Stavrou, A., ”SQLProb: a Proxy-Based Architecture Towards Preventing SQL Injection Attacks,” In Proc. SAC pp.2054-2061, 2009.
[14] Oppliger, R., and Gajek, S., ”Effective Protection Against Phishing and Web Spoofing,” In Proc. CMS, **3677**, pp.32-41, 2005.
[15] Adelsbach, A., and Gajek, S., Schwenk, J., ”Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures,” In Proc. ISPEC, pp.204-216, 2005.
[16] Diffie, W., and Hellman, M. E., ”New directions in cryptography,” IEEE Transactions on Information Theory, **22**, pp.644-654, 1976.

Wookey Lee Wookey Lee received the B.S., M.S., and Ph.D. from Seoul National University, Korea, and the M.S.E. degree from Carnegie Mellon University, USA. He currently is a Professor in Inha University, Korea. He has served as a PC member for many conferences such as CIKM, IEEE DEST, and as an organizing chair for many conferences such as DASFAA and VLDB. He won the best paper awards in KORMS and KIISE. He is the EIC of Journal of Information Technology and Architecture, and an editor for WWW Journal. His research interests include Graph and Mobile DB, Patent Information, and Data Warehousing.

Simon S. H. Park Simon Soon-Hyoung Park received his B.Sc. and M.Sc. in Industrial Engineering, with high honors, from Inha University, Korea, in 2012. He is currently in doctor degree in Inha University. His research interests include Graph theory, social network analysis, similarity measure, and patent analysis

Chasung Lim Chasung Lim received the B.S. from Sungkyul University and M.S. from Inha University, Korea, He currently is engineer about analyzing application vulnerability, malicious code reverse engineering and APT(Advanced persistent threat) solution engine design and development. His research interests include exploit, application vulnerability and reverse engineering.

Jinho Kim Jinho Kim received B.S. (1982) in Electrical Engineering from Kyungpook National University and M.S. (1985) and Ph.D. (1990) degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) respectively. From 1990, he joined at the Department of Computer Science, Kangwon National University. Now he is a professor at Kangwon National University.

Sangwon Kang Sangwon Kang received B.S. (1996) in Information Communication Engineering from Myungji University and M.S. (2009) in Media Engineering from Korea University. He is currently a Ph.D. candidate at Dept. of Computer Science, Kangwon National University and a Deputy Manager at Division of Public customers, KT, Korea.