Research Journal of Applied Sciences, Engineering and Technology 3(9): 948-952, 2011
ISSN: 2040-7467

Deploying a Web Client Authentication System using Smart Card for E-Systems

[1]Shadi Aljawarneh, [1]Maher debabneh and [2]Shadi Masadeh and [3]Abdullah Alhaj
[1]Isra University,
[2]Applied Science, University Amman, Jordan
[3]The University of Jordan, Amman, Jordan

Abstract: The aim of this study is to make sure that the requested confidential digital object is authenticated and hence no tampering can be performed upon it particularly at the client-side. Thus a novel web client authentication system using smart card has been developed, called Dynamic HMAC Validation System (DHVS). The DHVS is based on HMAC technology where the ciphering key is dynamically generated every time the client sends a request. This solution is implemented as a prototype. Such prototype consists of two parts: the DHVS at the client-side, and DHVS at the server-side. The evaluation results show that the DHVS provides a higher level of client authentication for different kinds of e-systems and indeed this is an ultimate security requirement.

Key words: Authentication, HMAC, e-systems, tampering, vulnerabilities, web application, system security

INTRODUCTION

Although a number of techniques exist for authentication, web sites and web applications continue to use weak authentication schemes that are vulnerable for eavesdroppers, particularly in e-commerce environments. Such successful attacks are often occurred because of careless use of authenticators stored on the client-side (Fu *et al*., 2001; Lam *et al*., 2008).

In 2009, the WhiteHat (Anonymous, 2009) security has reported a number of web attacks and such percentages are as follows:

- Cross-Site Request Forgery (11% of web sites): It allows an adversary to force an unsuspecting user's browser to make HTTP requests that are not intended.
- For instance, an adversary can force a user to compromise one own banking, e-commerce or other web site accounts invisibly without user's knowledge. Since the forged request is coming from the authorized user, the web site will recognize it as being the intent of that user.
- Insufficient Authentication (10% of web sites): Insufficient Authentication vulnerabilities are normally found within the business logic of a web application. A successful exploitation leads to an adversary gaining unauthorized access to protect resources of a web site. For instance, while it is logged-in as a normal user, an adversary might impersonate another user on a web system such as web financial, healthcare and general content management systems where there is a huge focus on a complex business logic functionality.
- HTTP Response Splitting (9% of web sites): HTTP Response Splitting is a web attack technique where a single HTTP request is sent to a web site in such a way that the HTTP response appears to look like two requests. In relation with the network architecture of the web site/web application or the behavior of a web browser, the "second" HTTP response that is under the control of an adversary can be used to poison cache servers, and deface web pages.

The traditional mechanism of client authentication is similar to server authentication except that a web server re- quests a digital certificate from a client to verify that a client is who it claims to be. The digital certificate must be an X.509 (Housley *et al*., 1999) certificate and signed by a Certificate Authority (CA) trusted by a server. It can only use client authenti- cation when a server requests a certificate from a client. It should be noted at this stage that not all servers do support client authen-tication, including the Host On-Demand Redirector. The later versions of the IBM Communications Servers (CS/NT, CS/AIX, etc.) support client authentication (Web Sphere, 2009).

When a server requests a digital certificate, a client has the choice to send a certificate or attempt to connect without one. A server allows the connection if the certificate of a client can be trusted. When a client attempts to connect without a certificate, a server might give the client access but at a lower security level (Web Sphere, 2009).

The existing web client authentication systems should defend against any adversary. However, web sites may con-tinue to be vulnerable to attacks such as eavesdropping, server impersonation, and stream

Corresponding Author: Shadi Aljawarneh, [1]Isra University, Jordan

tampering. Currently, the best defense against such attacks is to use the Secure Sock- ets Layer (SSL) with some form of client authentication (Eric Rescorla, 2000; Fu *et al*., 2001).

Clients need to make sure that only the authorized people can access and alter sensitive personal information that they share with web sites. At the same stage, web sites need to make sure that the only authorized users have access to the services and the content provided. Therefore, in an attempt to solve these issues, client authentication addresses the needs of both parties. This paper discusses the design interaction of the proposed solution. Implementation details are also reported in this paper.

A client authentication involves proving the identity of a client (or user) to a server on Internet (Fu *et al*., 2001). The task of authenticating the server to the client is also important however is not the focus on this study because most existing academic and technical research focuses on the protection of server-side resources (Cannon and Wohlstadter, 2009).

The adversaries are interested in targeting the referenced objects and page code on the fly. For example, it is possible to replace an original image by another image containing malicious code. A victim requests the altered image and then it can disrupt the contents of a web server or client machine.

LITERATURE REVIEW

The recent approaches are discussed that attempt to ad- dress the lack of web client authentication issue. First, Hassinen and Mussalo (Hassinen and Mussalo, 2005) proposed a client-side encryption system to protect data integrity and user trust. The client encryption key is located on a client smart card or can be stored on the server and transferred over an HTTP connection.

However, integrity of data could be lost if this approach is adopted because Java applets can access the client's local file system. Thus, a criminal can replace the original signed applet with a faked applet to access the client's webcontent. Finally, existing web applications would require modification to implement this technique.

The existing client authentication mechanisms within HTTP (Franks, 1999) and SSL/TLS (Dierks and Allen, 1999) could provide authentication a range of adversaries. However, these mechanisms are not sufficient for use on Internet because (i) the lack of a cen- tral infrastructure such as a public-key infrastructure or a uniform Kerberos (Steiner *et al*., 1988) leads to the proliferation of weak schemes.

Fu *et al*. (2001) have suggested a set of hints for designing a secure client authentication scheme. Using these hints (such as use of cryptography appropriately, use the required amount of security, not relying on the secrecy of a protocol, understanding the properties of cryptographic tools, and limiting exposure of passwords), a simple authentication scheme secure against forgeries by the interrogative adversary is designed. In conjunction with SSL, this scheme is secure against forgeries by the active adversary.

This technique assumed that a user has an existing ac- count on a server which is accessed via a username and password. At the beginning of each session, a server re- ceives the username and password, verifies them, and sets an authentication cookie on the user's machine.

Subsequent requests to a server include this cookie and allow a server to authenticate the request. The design of each cookie ensures that a valid cookie can only be created by a server. Therefore anyone possessing a valid cookie is authorized to access the requested content on a server. However, because cookies are stored on the client-side, adversary can modify the Message Authentication Code (MAC) value which is stored in the cookie.

Alan *et al*. (2009) have developed SCoopFS (Simple Coop- erative File Sharing) to address the problem of sharing files. Although SCoopFS provides server authentication, client authorization, and end-to-end encryption, the user does not, however, ensure that the referenced objects are requested from the correct server.

Dynamic HMAC Validation System (DHVS): As discussed earlier, the problem to be addressed in this research is making sure that the requested confidential digital object comes from the correct server. A client authentication sys- tem is proposed in this paper to identify illegal modifications, and therefore, a Dynamic HMAC Validation System (DHVS) to ensure the authenticity of digital objects before it is developed.
The question then arises, what happens when an altered digital object has been detected? The proposed DHVS methodology can be set up in two steps:

- Detection
- Reporting to web or system administrator via e-mail if any illegal modification occurred

It should be noted that the Hashing Message Authenti-cation Code (HMAC) is a short code to ensure the integrity of data beside the cryptography techniques. It is a one-way hash function that involves a private key. Only the parties that know the private key can compute the MAC value. It is possible to use a hash function to build a MAC using an algorithm such as HMAC (Dittmann *et al*., 2001).

Architecture of DHVS framework: Figure 1 shows problem is solved to steps as represented the flow chart. The DHVS has a number of advantages over other ap-proaches:

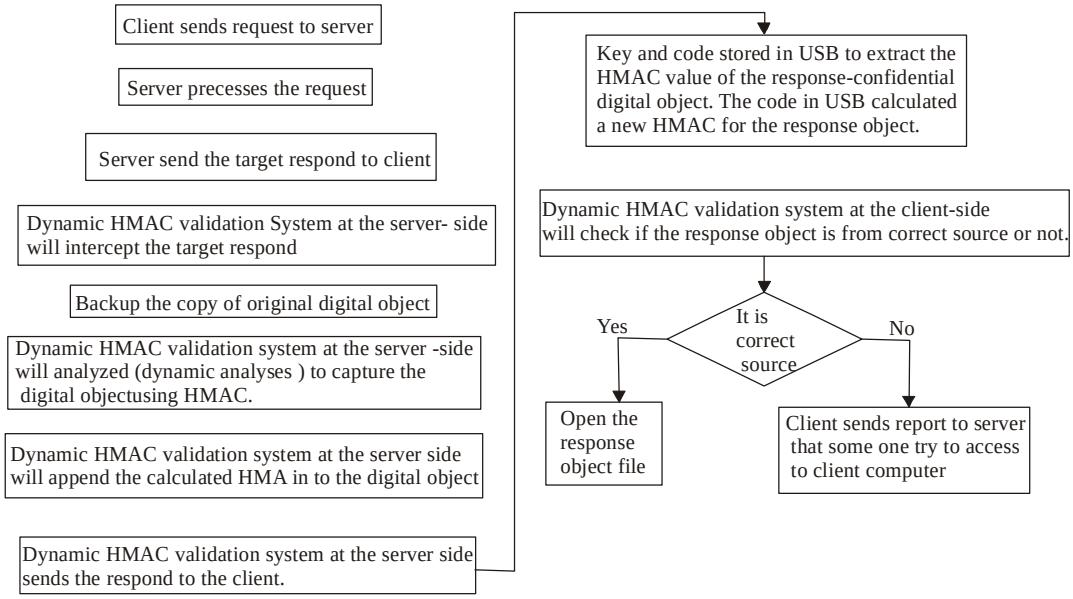

Fig. 1: Flowchart view of DHVS architecture

- It does not require modifications to the existing web application and web sites architectures
- It is compatible with all major web browsers such as IE, Nesscape and Google Chrome
- This system requires the use of a non-malleable MAC

This means that no adversary can generate a valid ciphertext without both the server's secret key and the plaintext. Examples of keyed, non-malleable MACs are HMAC-MD5 and HMAC-SHA1 (Krawczyk *et al*., 1997).

HMAC will be used to cipher digital object file in a web server then the HTTP response is sent with a MAC value. The smart card will be used for storing the validation code at client-side.

In order to check that the HTTP response is from the correct source, the client device is connected such as the USB or smart card at client site, and then the program calculates HMAC value for the digital object file at client side and compares it to the value from a web server.

Functional overview: The functionality steps of the suggested solution are depicted. The following steps illustrate the mechanism of the solution, as shown in Fig. 2:

- Client sends request to server
- Server processes the request
- Server sends the target response to client
- Dynamic HMAC Validation System at the server-side will intercept the target response.
- Backup copy of original digital object
- Dynamic HMAC Validation System at the server-side will analyze (dynamic analysis) to capture the digital object and then it will hash the digital object using HMAC
- Dynamic HMAC Validation System at the server-side will append the calculated HMAC into the digital ob- ject
- Dynamic HMAC Validation System at the server-side sends the response to the client
- Key and code stored in USB to extract the HMAC value of the response-confidential digital object. The code in the USB calculates a new HMAC for the response object
- Dynamic HMAC Validation System at the Client-side will check if the response object is from correct (is au- thenticated) source or not
- If it is correct open the response object file. Otherwise, client sends report the server that someone try to access the client's computer. Implementation of DHVS system

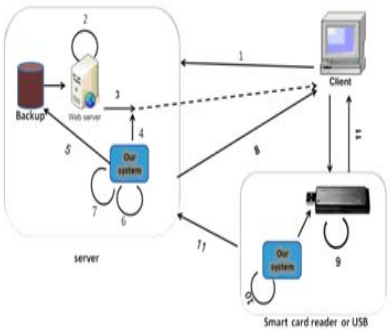

Fig. 2: Architecture design of DHVS prototype

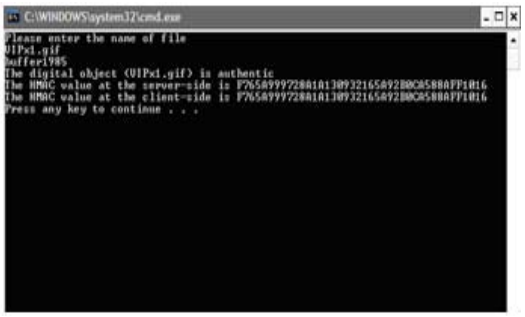

Fig. 3: Console of results

The DHVS system is implemented in Java, Servlets, and Filters. In order to demonstrate that this DHVS system is able to make sure that the digital object became from unique source and correct object file requested digital object file is authenticated and no tampering can performed upon it, par- ticularly at the client side, some experimental testing have been undertaken.

HTTP interceptor mechanism: This mechanism is the manager of the DHVS**.**

Architectural design of the prototype: As shown in Fig. 2, the DHVS prototype consists of two mechanisms: HTTP Interceptor at the server-side, and HTTP Checking at the client-side. system, and is based on the HTTP analyzer filter. The goals of HTTP interface mechanism are:

- Intercepting every request and response
- Analyzing for every response
- Finding out the HMAC value for every requested con- fidential digital object
- Backing up for every requested confidential digital object in case of tampering happens

HTTP checking mechanism: This mechanism is positioned at the client-side. The goals of HTTP Checking mechanism are:

- Analyzing for every response before a web browser renders it
- Calculating the HMAC value for every responded con- fidential digital object
- Comparing between the first HMAC values with the second HMAC value
- Reporting web administrator in case any tampering is occurred

Evaluation: The system in an environment which is composed of Apache with Tomcat container on MS Windows Server 2003 is tested. The tomcat web server contains a copy of target web site as well as a kids web site.

Different kinds of flaws and vulnerabilities have been generated that permits for the tampering of files in a web server. During the testing, all the attacks launched against the web servers were fully detected by the DHVS.

Above 20 tampering attacks had manually launched. This result has shown that the proposed DHVS provides authentication for digital objects at the client-side. As shown in Fig. 3, the VIPx1.gif is authentic, where, the HMAC at the client-side is F765A999728A1A13 0932165A92B0CA588AFF1016, and the HMAC at the server side is also F765A999728A1A130932165A92B0 CA588AFF1016. Hence, they are identical.

It should be noted that the name of confidential digital object must begin with ”VIP” word, to determine if the dig- ital object is confidential object or not.

As the validation process is performed online in real-time, it should induce a time overhead in the service. The results presented in this section have been obtained on the same set of requests, using the same architecture of web servers.

CONCLUSION

A web client authentication is a common involvement for new and existing web applications and web sites as more and more personalized and access-controlled services move online. However, several web sites use authentication schemes that are extremely weak and vulnerable to attack. These problems are most often due to careless use of authenticators stored on the client. To address this problem, a novel DHVS system has presented. Results from a series of experimental tests appear to suggest that the DHVS satisfies completely the security objective for client authentication purposes.

ACKNOWLEDGMENT

We would like to thank King Abdulla Fund for Development (KAFD) and King Abdulla for Design and Development Bureau (KADDB) for the financial support during the research and development phases.

REFERENCES

Alan, T.C., H. Karp, M. Stiegler, 2009. Not One Click for Security. Technical Report, HP Laboratories-Hewlett-Packard Development Company. Retrieved from: http://whitepapers.techrepublic.com.com/abstract.aspx?docid=972301.

Anonymous, 2009. Retrieved from: http://www.slideshare.net/jeremiahgrossman/whitehat-security-website-security-statistics-report-full-q109.

Cannon, B. and E. Wohlstadter, 2009. Enforcing security for desk-top clients using authority aspects. AOSD '09: Proceedings of the 8th ACM International Conference on Aspect-oriented Software Development, New York, USA, pp: 2550-266, ACM. ISBN: 978-1-60558-442-3.

Dierks, T. and C. Allen, 1999. The TLS Protocol Version 1.0 RFC Editor, United States.

Dittmann, J., P. Wohlmacher and K. Nahrstedt, 2001. Using cryp-tographic and watermarking algorithms. IEEE Multimedia, 8(4): 54-65.

Eric Rescorla SSL and TLS, 2000. Designing and Building Secure Systems, Addison-Wesley.

Franks, J., P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen and L. Stewart, 1999. HTTP Authentication: Basic and Digest Access Authenti-Cation. RFC 2617, Network Working Group.

Fu, K., E. Sit, K. Smith, and N. Feamster, 2001. Dos and don'ts of client authentication on the web. SSYM'01: Proceedings of the 10th Conference on USENIX Security Symposium, pages 9-19, Berkeley, CA, USA, 2001. USENIX Association.

Hassinen, M. and P. Mussalo, 2005. Client Controlled Security for Web Applications. In: Wener, B. (Ed.), The IEEE Conference on Local Computer Networks 30th Anniversary, IEEE Computer Society Press, Australia, pp: 810-816.

Housley, R., W. Ford, W. Polk and D. Solo, 1999.Internet X.509 Public Key Infrastructure, Retrieved from: http://www.ietf.org/rfc/rfc2459.txt.

Krawczyk, H., M. Bellare and R. Canetti, 1997. HMAC: Keyed-hashing for Message Authentication. RFC, 2104. Network Working Group.

Lam, M.S., M. Martin, B. Livshits and J. Whaley, 2008, Secur-ing Web Applications with Static and Dynamic Information Flow Tracking. In PEPM '08: Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program manipulation, New York, USA, ACM. pp: 3-12.

Steiner, J.G., C. Neuman and J.I. Schiller Kerberos, 1988. An Authentication Service for Open Network Systems. Usenix Conference Proceedings, pp: 191-202.

Web Sphere, 2009. Client Authentication, Retrieved from: http://publib.boulder.ibm.com/infocenter/hodhelp/v9r0/index.jsp?topic=/com.ibm.hod9.doc/help/secclientauth.html.