Verification of Web Content Integrity: A new approach to protecting servers against tampering

Shadi Aljawarneh, Christopher Laing, and Paul Vickers
School of Computing, Engineering & Information Sciences
Northumbria University, Newcastle upon Tyne, Newcastle, UK
{shadi.aljawarneh, christopher.laing, paul.vickers}@unn.ac.uk

Abstract

The provision of web services is a real-time process, conducted in ad-hoc, 'off the cuff' manner. Consequently the verification of the data content and the identification of any authorized data interference or manipulation are not without problems. Some progress has been made in addressing the verification of server content integrity, but current solutions are restricted by the limitations of the SSL protocol, the statelessness of HTTP, and difficulties with automatic code analysis. This paper reviews the problems associated with unauthorized data manipulation of static and dynamic web content, presents a web security real-time framework that can be used to verify the static and dynamic web content of a requested page. It is suggested that such a framework will offer an increased level of user confidence, since the framework will provide a much greater protection against web server subversion.

Keywords

Statelessness of HTTP, web security, Request-Response model, web-based system, add-on security mechanisms, e-Form, tampering, integrity of data

1. Introduction

Users need assurances that their private information is kept confidential and is transferred properly over a web-based system. Consequently, reputable organizations must ensure that their private information and transactions are conducted correctly without compromise [1-4, 12]. According to the Computer Emergency Response Team (CERT) [5] there has been a sharp increase (5990 in 2005 to over 8000 in 2006) in the number of security vulnerabilities[1] which threaten web content.

Static and dynamic web server content can be tampered with by changing the (i) style classes, (ii) referenced objects (images, audio, video, and other objects), (iii) source code of the web page itself, (iv) running listener software containing malicious code on the server to intercept a requested page before the client receives it [1-4, 7, 15, 17]. Consequently, the integrity of web content can be violated on the server even though the communication channel between the server and client is secure.

Up till now, data integrity has received little attention in information security research [2, 3]. Furthermore, there is little published research in methods for testing web content integrity [6].

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) cryptographic security protocols were developed to support the authenticity and integrity of data transit [1-4, 7, 13, 18], and as such, they do not provide an absolute solution. Furthermore, several verification and protection tools which use cryptography, and hashing[2] to protect web content against tampering, are not capable of verifying the integrity of web content before a request or response enters the secure communication channel [2, 3]. This might be because these mechanisms are purely technical solutions, and a technical mechanism alone does not provide a standard policy [4].

This paper is structured as follows. Section 2 identifies the web security issues and describes the data flow over HTTP Request-Response model. Section 3 gives an overview of the existing approaches and schemes. Section 4 outlines a proposed integrity verification system. Finally, section 5 draws conclusions and discusses possible future work.

[1] Weaknesses in a computing system that can result in harm to the system or its operations [18].
[2] Hashing is a technique that aims to ensure the integrity of data by generating unique hash values analogous to fingerprint [8].

2. Web Security

Stein [18] outlines a number of definitions of web security from the user's perspective. For some, web security is the ability to view Internet content in peace and safety. For others, it is the ability to conduct safe business and financial transactions. For web authors, it is the confidence that individuals will not damage their web sites.

Studies and surveys indicate that the objectives of web security (integrity, availability, and confidentiality of data) are being violated [1-4]. In an attempt to remedy this, we focus on the integrity of data; we are not concerned with the confidentiality and availability of data. It should be noted that data integrity refers to the trustworthiness of information resources, thereby ensuring that only an authorized client can alter the data - unauthorized tampering may result in incorrect or malicious behaviour of the web application.

Because of the difficulty of code analysis and blind add-on security mechanisms [12, 16, 17], Jacobs and Malloy [12] suggest that software engineering principles (such as analysis, design, implementation and testing) should be integrated into web security. The purpose is to identify what a user and an organization need for every stage of the software engineering life cycle. Therefore, this can detect the web security vulnerabilities at each stage instead of processing the security vulnerabilities at the implementation stage. However, this integration may require the rebuilding of existing web applications – some web applications might consist of complex structures, consisting of multi-programming languages and imported binary components with little or no documentation. Consequently, it may be difficult to define for legacy web application at every stage of the software engineering life cycle all web policies that maintain security vulnerabilities.

As shown in Figure 1, the integrity of web content relies on the integrity of the HTTP Request-Response model. Therefore, if this model fails, the data integrity might be violated [1, 7]. The model can fail because web servers and web browsers do not properly manage the statelessness of HTTP, in which, each client request results in a new connection between a web browser and a web server. The Common Gateway Interface (CGI) supports maintaining state through the use of hidden variables or cookies that keep the track of current information for each request. However, it is possible to save the HTML form to a disk, modify the hidden values of form fields, and then reload this altered form into a web browser for rendering [10, 18]. Zhou [9] has identified some problems in web server models, these include: web server models cannot ensure the security of continuity of HTTP conversations on a server – they are more concerned with the implementation of the cryptographic rules rather than implementation of a security analysis of system functions.

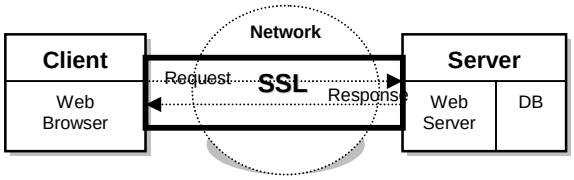

Figure 1. HTTP Request-Response Architecture

3. Integrity Verification Approaches

Below we survey a number of approaches and schemes to give a more detailed overview of integrity verification. There are three established techniques in common usage for ensuring the integrity of web content: SSL to secure the communication channel, form-field validation to protect against harmful data at the client and server-sides, and firewalls to protect against malicious code and other attack strategies.

3.1 Form Field Validation Scheme

A form field validation scheme is the first defence against tampering at the application level. Developers have adopted a number of validation approaches to prevent loss of web content integrity: server-side validation, client-side validation, double-checking validation, and digital signature and data validation on the client and/or server-sides. The double-checking approach adopts an alternative validation scheme at the server–side, even though the validation scheme is bypassed at the client-side [1-4, 7].

A validation scheme is necessary and required for both client and server-sides, but is not sufficient to protect web content integrity against tampering, because fundamentally it is designed to verifying basic properties of the input data: length, range, format, default value, and type. Therefore, a validation scheme is useless if any malicious script or listener is already installed on the server [10, 17].

As a result of the transparency of code at the web browser level, the following approaches can cause loss of content integrity at the HTML form level:

- Hidden fields manipulation: an individual saves the HTML form to a disk, modifies a hidden field value (such as the price of a product), and then reloads this tampered form into a web browser for rendering [10].

- Script manipulation: an individual removes the client validation modules on a web browser to submit illegal data to a web server. A web server accepts the tampered form and then the data is saved in backend database. Many web application security vulnerabilities come from input validation problems including Cross-Site Scripting (XSS) and SQL injection [2, 3, 11, 17]. This approach is possible through removing all script modules between the <script> and </script> tags, removing the event-handler that invokes the validation modules, or turning off script and Java applet options via web browser settings.
- Another approach is to analyze a validation module to submit illegal information – an individual could apply reverse engineering techniques on validation modules [2, 11, 17].

However, HTTP provides the REFERER header to help detecting a tampered form. A REFERER contains the URL of an original form. But this header is not sufficient to alert a web server and web browser because it is possible to tamper with the URL information of the header [2].

3.2 Network and Application Firewalls

Network firewalls provide protection at the host and network level [4, 10, 18]. These security defences cannot be used to detect tampering problems for following reasons [4]:

- They cannot stop malicious attacks that ask to perform illegal transactions. Firewalls are designed to prevent vulnerabilities of signatures and specific ports.
- They cannot manipulate the form operations such as asking the user to submit some information, or validate the false data. Therefore, the firewalls cannot distinguish between the original request and response conversation, and the tampered conversation.
- They do not track a conversation and do not secure the session information. For example, they cannot track when session information in cookies is exchanged over HTTP Request-Response model.

Given the above, some security problems still remain. We now discuss three recent approaches that attempt to address one or more of these problems.

3.3 Client-side Encryption Approach

Hassinen and Mussalo [1] propose a client-side encryption system to protect confidentiality, data integrity and user trust. They encrypt the data inputs using a client encryption key before submitting the content of HTML Form. The key is either located on a client smart card connected to the client machine or it is read over an HTTP or HTTPS connection. A downloaded web page includes a signed applet which handles the encryption and decryption values. This applet also reads the encryption key from a local file. In addition, a downloaded web page includes JavaScript methods that invoke the applet methods for encryption and decryption. The approach uses a one-way hash function. It computes the hash value which is inserted into the main data input before encryption. After a new request, the JavaScript function invokes the applet decryption method to decrypt the parameter value and places the returned value in the corresponding input field. The message validation includes finding a new hash value from the decrypted message and comparing it to the hash value which is received with the message. If they are the same, the data is accepted, otherwise, the data is deemed to have been illegally altered and the validation will fail. This system has two main requirements [1]: (i) it must be able to run on any major web browser, and (ii) without the need to install additional hardware or software on the client.

However, the following weakness will arise if this approach is adopted:

- Java applets can access the client's local file system. If the applet is signed the user will be asked whether he wants to trust the applet code. Thus, an individual can replace the original signed applet with a faked applet to access the client's web content.
- Another potential weakness is the loss of the client smart card with its Personal Identification Number (PIN). As the smart card includes the encryption key, if it is lost, the whole web-based system can be compromised.
- This approach fails to address requirement (ii). if a Java applet is installed on the smart card to protect the Java applet against tampering. This is because the client machine needs a card reader and necessary drivers.
- Transparency of code is a significant weakness of this approach, as encryption and decryption applet methods and JavaScript methods can be removed or cancelled. If this happens, the encryption will not take place and the submitted values will be in plain text.

In addition to the above weaknesses, the client-side smart card approach has the following limitations:

- It requires new web applications and fails to address existing web applications. This is because

it requires a client smart card and encryption and decryption methods.
- It does not verify the server's web content because it assumes that the risk is coming only from the external clients at the HTML form level.
- It does not possess a risk analyzer to protect user information on the web server if the web server has been tampered with. Furthermore, it does not provide alerts to clients if the system has been tampered with.

3.4 Dynamic Security Surveillance Agent (DSSA) Approach

Sedaghat, Pieprzyk, and Vossough [2, 3] propose a DSSA tool on the server-side that automatically intercepts an HTTP request to verify the integrity of the requested page before the web server responds to the client. This tool is positioned between the web server and client machines. DSSA uses a timestamp signature and hash function to verify the integrity of requested web pages and all their referenced objects. If the hash value of the requested web page and its referenced objects equals the hash value which is registered in a secure database, the web server accepts the HTTP request and then it responds to the client.

However, the following weakness will arise if this approach has been adopted:

- Tampering is still a potential problem. This is because DSSA does not verify dynamic web content, which is generated on the fly and on demand. Therefore, some referenced objects can be tampered with and DSSA cannot alert the web administrator or clients. Consequently, data integrity can be violated.

In addition, DSSA approach has the following limitations:

- It does not verify dynamic server content. Therefore, dynamic web pages are still a significant challenge – there are difficulties in analyzing dynamic web pages.
- It assumes that every web page is independent from every other web page even though some referenced objects are shared by more than one web page.
- It only supports client-side code and scripting.

3.5 Adaptive Intrusion-Tolerant Server Approach

This approach [14] offers a general architecture for intrusion-tolerant enterprise systems. This consists of redundant servers running on diverse operating systems, intrusion-tolerance proxies that are positioned between web servers and client machines to verify the behaviour of servers, and their monitoring and alerts components.

When a client request arrives a proxy "leader" intercepts the request, checks it, analyzes it, and filters out incorrect requests. The leader also forwards a request to a number of application servers, depending on the current enforced policy. Furthermore, a leader intercepts the responses. Finally a leader finds a hash value for all responses. If they match, the leader sends a response to the user, otherwise, a report is sent to a monitoring component to take the correct actions and alerts.

However, the following weakness will arise if this approach has been adopted:

- This approach does not verify the integrity of referenced objects that are generated dynamically on the fly.
- The performance is too slow if the number of application servers is greater than three to check the integrity of server web content.

4. Web Content Verification (WCV) System

Although verification and protection mechanisms have been adopted to protect web content over the HTTP Request-Response model, a number of the web security issues are still unresolved – the integrity of web content can be compromised on the client and server sides, and important web security policies have yet to be answered in a systematic way. We are developing a server web content verification (WCV) system to identify the tampering problems. The WCV system comprises of a number of security components, and a web security framework that enforces collection of web policies.

4.1 Work assumptions

In reference to the literature review, we adopt two main assumptions. First, SSL only secures the data in transit [1-4, 7, 13, 18]. As was mentioned in the previous sections, SSL provides digital certificates to encrypt the data in transit against hacking listeners and eavesdropping attacks. A third party or a certificate authority controls these digital certificates.

Second, the data resulting from user interaction via an e-Form component on the client is untrustworthy. User input may contain malicious code that harms a web server or a backend database [10]. Furthermore, the client validation scheme can be violated. As result,

all user data that is sent as an HTTP request to a server is untrustworthy. Therefore, we also assume a validation scheme operates on both sides (client and server) to check the integrity of a request before it is processed on a web server. This means if the client validation scheme is subverted, there is still another validation scheme running on the server before processing a request.

A significant challenge to identify tampering problems is the difficulty of analyzing dynamic web pages which can be generated or modified through server and client scripts embedded in the web page [10, 15, 16]. We classify the web content into four classes: identical interaction elements, almost identical interaction elements, less identical interaction elements, and non-identical interaction elements. This classification relies on modelling interaction elements and modelling elements of a web site through improving the relationship of input unit of Bypass Test Strategy[3] [17]. The aim of this classification is to understand how to dynamically produce units of web content. Moreover, another important aim is to simplify a method of automatic and even manual analysis of a web site that contains a large number of elements and each web page can include a large number of interaction elements (i.e. forms, frames, links, and non-HTML objects).

The WCV system will interact with every element in a web site. For example, it is possible to represent the logo image of a university in one or more web pages of the university web site, so we do not need to compute the hash value for this logo object for every time it is used.

4.2 Architecture of Web security Framework

The proposed WCV system will include a web security framework. As shown in Figure 2, this framework consists of a number of web-based components: web register, response hashing calculator, integrity verifier, and recovery. The web register component has two stages: monitoring and registration. The integrity verifier (manager) component that mediates between the web server and client's machines manages the HTTP requests and responses through executing a state protocol that enforces a number of web policies to elements of a web-based system. The web policy will outline how each component will be instituted, when it will be enforced, and who will be allowed to use and response. The response hashing calculator component calculates the hash value of each HTTP response on the web server before sending the response to the integrity verifier for further processing. Finally, the recovery component recovers the marked data, which is stored if the action of enforced web policy from integrity verifier is invalid. In addition, Figure 2 illustrates how the proposed framework is separate from the web server. Note that the components of framework do not need to run on a dedicated machine, they can be run as separate processes on the server-side.

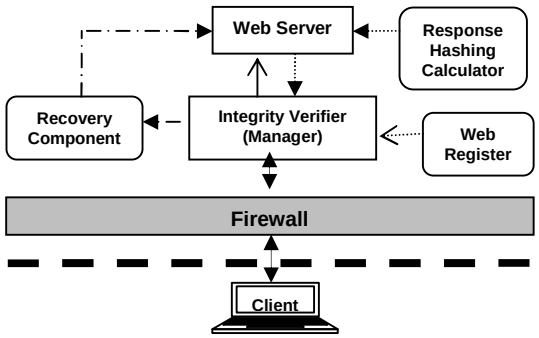

Figure 2. Schematic view of the web security framework architecture

Our proposed system offers integrity of data, and higher level of trustworthiness to organization and user. We believe that the proposed framework will be capable to verify web pages and referenced objects on the designated directories of web server, and web content on the fly against tampering. To ensure the survivability[4] of web services, we are enabling WCV system to detect and protect web content against tampering and recover the original copy of the compromised object. We are exploring risk analyzer techniques to protect user information on the web server if the web server has been tampered with. Furthermore, at the monitoring stage, if the system has been tampered with, it provides alerts to the web administrator.

The WCV system possesses a number of properties: it does not require modifications to existing web application architectures, it does not require any additional changes on the client-side and adopts minimal changes on a server-side, and it can be run on all major web browsers. Our method is targeted at sensitive web sites and web applications that provide

[3] A strategy is to create tests on the client for web applications that intentionally violate explicit and implicit checks on user inputs. It includes three levels of testing: value level, parameter level, and control flow level. Further details are described in [17].

[4] Providing continued reliable and correcting service to clients and organizations.

sensitive transactions between organizations and users. For example, e-Banking, e-Finance, and e-Ticket reservations – services that support sensitive transactions between clients and organizations.

5. Conclusions and Future Work

Although several add-on security mechanisms have been adopted to protect web content over the HTTP Request-Response model, a number of security issues are still unresolved. This paper has reviewed existing approaches, considered their strengths, their weakness, and their limitations. In conducting this review, we have focused on one issue, namely the integrity of web content. It has been shown that given the limitations of SSL, a loss of web content integrity is possible because of the statelessness of HTTP.

In an attempt to overcome this problem, we have formulated a systematic web security framework that could provide continued reliable and correct services to external users, even though a web data manipulation problem may have occurred. In the next stage of this research, we will develop a number of mathematical models which will be used to analyze dynamic and static web pages. The proposed framework will execute a state protocol that will enforce web policies and alerts – it is suggested that this system will be able verify the static and dynamic web content of requested web pages.

6. References

[1] Hassinen, M., Mussalo, P. Client controlled security for web applications. In: Werner B, ed. The IEEE Conference on Local Computer Networks. 30th Anniversary, Sydney, Australia, November 15-17, 2005, pp. 810-816. Los Alamitos, California: IEEE Computer Society, 2005.

[2] Sedaghat, S. Web Authenticity, Master Dissertation, UWS, Australia, 2002.

[3] Sedaghat, S., Pieprzyk, J., Vossough, E. On-the-fly web content integrity checker boots user's confidence. Communications of the ACM, pp. 33-37, vol. 45(11), Nov 2002.

[4] Gehling, B., Stankard, D. eCommerce Security. In Proceedings of Information Security Curriculum Development (InfoSecCD) Conference ’05, pp. 32-37, Kennesaw, GA, USA, Sept 23-24, 2005.

[5] CERT. CERT Statistics 1988-2006, Technical Report, 2006, http://www.cert.org/stats, [30/1/2007].

[6] Probert, R.L., Stepien, B., Xiong, P., Formal Testing of Web Content using TTCN-3 in TTCN-3 UserCom'05 Conference proceedings.

[7] Honkala, M., Vuorimaa, P. Secure Web Forms with Client-Side Signatures. In proceeding of 5th International Conference on Web Engineering, P. 340, Vol. 3579, Syd, Australia, Jul 27-29, 2005.

[8] Oppliger, R. Security Technologies for the World Wide Web, 2nd Edition. Norwood MA, 2002.

[9] Zhou, B. An integrated Web security system. In Proceedings of 14th International Workshop on Database and Expert Systems Applications, pp. 204 – 208, 1-5 Sept 2003.

[10] Scott, D., Sharp, R. Specifying and Enforcing Application-Level Web Security Policies. IEEE. Knowl. Data Eng., pp. 771-783, vol. 15(4), 2003.

[11] OWASP. The Ten Most Critical Web Application Security Vulnerabilities. White Paper, v 1.0, 2003.

[12] Jacobs, D.P., Malloy, B.A. An Application-Centred Course on Data-Driven Web Sites. In Proceedings of Frontiers in Education 2001, Reno NV, pp. F2D-10 to F2D-14, Oct 10-13, 2001.

[13] Oppliger, R., Gajek, S. Effective Protection Against Phishing and Web Spoofing, In Proceedings of the 9th IFIP TC6 and TC11 Conference on Communications and Multimedia Security (CMS 2005), Austria, Springer-Verlag, LNCS 3677, pp. 32 – 41, Sept 19 - 21, 2005.

[14] Valdes, A., Almgren, M., Cheung, S., Deswarte, Y., Dutertre, B., Levery, J., Saidi, H., Stavridou, V., Uribe, T.E. An Adaptive Intrusion-Tolerant Server Architecture. System Design Laboratory, SRI international Menlo Park, CA, Feb 2002.

[15] Reis, C., Dunagan, J., Wang, H.J., Dubrovsky, O., Esmair, S. BrowserShield: Vulnerability-Driven Filtering of Dynamic HTML.2006.

[16] Ricca, F. Analysis, Testing and Re-Structuring of Web Applications. In proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04), pp. 474-478, 2004.

[17] Offutt, J., Wu, Y., Du, X., Huang, H. Bypass Testing of Web Applications. IEEE International Symposium on Software Reliability Engineering. pp 187-197, Bretagne France, Nov 2004.

[18] Stein, L. Web Security: A step-by-step reference Guide, Wesley, 1998, ISBN: 0-201-63489-9.