Journal of Information Security, **2013, 4, 138-143**
http://dx.doi.org/10.4236/jis.2013.43016 Published Online July 2013 (http://www.scirp.org/journal/jis)

Web Security and Log Management: An Application Centric Perspective

Andrew Mwaura Kahonge, William Okello-Odongo, Evans K. Miriti, Elisha Abade
School of Computing and Informatics, University of Nairobi, Nairobi, Kenya
Email: andrew.mwaura@uonbi.ac.ke, wokelo@uonbi.ac.ke, eamiriti@uonbi.ac.ke, eabade@uonbi.ac.ke

Received March 6, 2013; revised April 7, 2013; accepted April 15, 2013

ABSTRACT

The World Wide Web has been an environment with many security threats and lots of reported cases of security breaches. Various tools and techniques have been applied in trying to curb this problem, however new attacks continue to plague the Internet. We discuss risks that affect web applications and explain how network-centric and host-centric techniques, as much as they are crucial in an enterprise, lack necessary depth to comprehensively analyze overall application security. The nature of web applications to span a number of servers introduces a new dimension of security requirement that calls for a holistic approach to protect the information asset regardless of its physical or logical separation of modules and tiers. We therefore classify security mechanisms as either infrastructure-centric or application-centric based on what asset is being secured. We then describe requirements for such application-centric security mechanisms.

Keywords: Web Security; Internet; Application Centric; Infrastructure Centric; Network Centric; Host Centric; Log Management and Monitoring

1. Introduction

With the growth of the Internet and Web technology, a lot of services are now offered to users globally. A wide range of applications have been launched ranging from informational websites, social networks, e-commerce and Software as Service (SAAS). As mentioned in [1] this growth may be attributed to the web application software architecture that meets the needs of an Internet-scale distributed hypermedia system. It is possible to integrate and build more applications and components in the web much more easily as compared to traditional desktop software approaches.

Alongside this success wave of Internet applications, a lot of vulnerabilities have been found and indeed exploited necessitating service providers to continue invest a great deal of effort and resources in monitoring and containing breaches. Examples of attacks include hackers exploiting improper coding standards of a web application running on the web server or inherent vulnerabilities of the web server itself [2]. It is important to be a step ahead in securing such systems. Several ways have been developed including security audit methodologies and security architectures to improve detection and prevention of risks. Methods such as installation of smart firewalls, secure end to end communications through Virtual Private Networks and Secure Socket layers have been used to improve security and have indeed been adopted for several types of web applications. Also, in line with the fact that electronic records such as computer and network logs are considered as the most important data in digital forensic [3-5], monitoring tools and techniques have been widely utilized including logging of transactions such as system logs, database logs and web server logs as well as the use of intelligent log monitoring and analysis software. However, all these methods and efforts have not guaranteed keeping systems secure or noticing attempts on them.

In line with the three principles of Information Security; Confidentiality, Integrity and Availability, eleven dimensions of protecting information assets are drawn by [6]. The idea of these dimensions is that a holistic approach is necessary where all the dimensions are used to improve security. Specifically, two of these dimensions are monitoring & evaluation where observations of the system and appropriate actions are performed. The first dimension of monitoring has also been noted by [7] to be a pillar of information security especially critical for web applications where monitoring of web servers, database servers and authentication servers through analysis of log files.

We explain the risks affecting web applications and the significance of logging and log analysis as far as monitoring is concerned. Then, we give a summary of classifications of security mechanisms on to which previous efforts of implementing security have previously been focused. Finally, we present a new classification that aims at providing a different focus of implementing security especially for web based applications. To support the proposed idea, a vulnerability scenario is discussed.

2. Web Application Authentication Levels

When a user contacts a website as shown in step *a* of **Figure 1**, an authentication request may prompt for a username and password. Two common methods for this are script controlled authentication and web server controlled authentication. The former includes form-based techniques that render a web page onto the user agent with data entry components to input username, password and other authentication data. The latter is where the web server manages validation of the user and will only serve the requested web resource if the user is valid. Basic authentication and Integrated Authentication are common examples of this.

When level *a* is completed and the web resource request is within the database server, such as is the case for content management systems, the server side script will initiate a connection to the database server. Then, depending on the audience or type of system, the web application may be designed to authenticate at level *b* with the database, as shown in **Figure 1**, by either N:1 or 1:1 mapping of actual or external user and database user respectively.

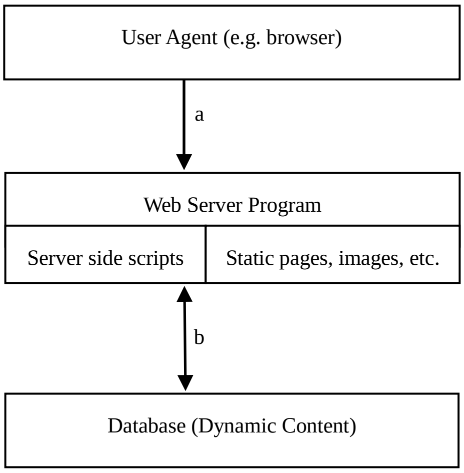
User Agent (e.g. browser)
a
Web Server Program
Server side scripts
Static pages, images, etc.
b
Database (Dynamic Content)

Figure 1. Authentication levels in a web environment.

Many-to-one database connections may be desirable in applications a scenario of Internet facing where a large section of the system performs read-only operations regardless of which users are logged in. It may also be desirable for achieving connection pooling [8] where the database services thousands of requests with only a few database connections. For such a case, a single database username and password is used to connect to the database. Most popular web application frameworks use this approach [9].

As much as this multiplexing reduces administrative overheads in the database and may have performance and scalability advantages, it introduces a blind spot when auditing database activity [10]. Analyzing database transaction logs may never reveal who did what.

3. Risks Affecting Web Applications

Unlike traditional desktop systems, web applications suffer native vulnerabilities due to their architecture and also due to the fact that they are exposed to a wider audience. Recently research indicates that among all attacks, SQL Injection and Cross-Site Scripting attack are the most common and most serious attacks [11].

3.1. Inherent Vulnerability in Web Server

This may occur if the web server has certain vulnerability and this gets exploited by an attacker. Some web servers have features that allow remote administration such as file management. If the controls are not adequate enough, they may be sources of vulnerability. As illustrated through an attack graph [12] an attacker may gains access to the web server by exploiting remote-to-admin vulnerability, then gains access to the database server through remote-to-user vulnerability.

3.2. Inherent Weakness of Web Script

A web script may be developed with a weakness or fault that may go undetected. For example, a page that is only meant only for read operations may fail to apply necessary control thereby allowing an attacker to perform privileged operations.

SQL Injection is one such attack. It is an application layer attack and exploits the improper coding standards [2] in web applications that by design can allow SQL commands to be injected through web forms.

Command Injection is similar to SQL injection and it uses the web program to execute commands such as operating system calls on its behalf. Among the various existing types of software vulnerabilities, command injections are particularly common [13]. A hacker would exploit the command set available to the web program once they know its platform and would then execute commands such as shell commands or other system application or system commands.

Cross-Site Scripting is mainly intended to attack the user of a website and by utilizing the property of the website that allows users to enter data [11]. Here a hacker enters part of HTML syntax or Java/VB Script syntax as a data in a data entry form such as a blog page.

4. Auditability and Log Analysis

In the related field of communication protocols, a definition is given by [14] that states that a protocol is *auditable* with respect to a property if it logs enough evidence to convince an *impartial* third party or judge of that property. This definition also goes further to state that the judging entity evaluates if some evidence enforces a given property in an impartial and transparent manner. Therefore, a program is auditable if, at any audit point, an impartial judge is satisfied with the evidence produced by the program. This in-turn provides Security Assurance [15] or grounds for confidence that the program or product meets its security objectives.

Logs provide information about the current and past states of systems and are therefore invaluable parts of system security [16] as they are useful for performing auditing and forensic analysis. Log analysis is performed, not as a replacement of, but in addition to the existing access control systems [17]. A web administrator will use data in the logs to understand how the attacker gained access to the system and also find out the type and extent of damage caused [18]. The use of audit logs has been recommended by security standards as a means of evaluating an IT system and therefore providing assurance [19]. In the web environment, logging is done by the several supporting hosts such as the web server, database and authentication server. Due to several hosts and applications generating their own logs, complications will occur such as timestamps, inconsistent content and formats. This causes a major challenge of consolidating them for holistic analysis [20].

Analyzing distinct log files from separate systems mainly assists in measuring limited impact of user activity from the point of view of the host or system being monitored. As much as this activity may provide database or web administrators with critical server-related data, it does not give a holistic perspective [21]. Further, when a breach is discovered in a web application, it is normal practice to interrogate web server logs and identify entries that may imply activity that resulted in the breach. Once the breach point is established, further analysis proceeds to identify and study the vulnerable web application, perhaps even its source code, after which appropriate action can be taken. Even in the presence of automated tools for log analysis, a substantial part this may end up being done manually and can take a long time especially in cases of a very busy web site.

5. Network-Centric vs Host Centric Security

Recently, security mechanisms have been classified by [22] as either network centric or host centric depending on their deployment model and which type of activity they observe and inspect. Network centric approaches would include Network Access Control or Intrusion Detection systems while antivirus software and Host based Intrusion Detection Systems installed at hosts would be classified as host centric. As much as they are crucial in an enterprise, network centric mechanisms lack necessary depth or context as they examine bits or data that is passing across the network especially in cases where the data is passed over SSL. Further, examining components of the network individually does not comprehensively analyze overall network security [12].

Research on methods of obtaining a larger picture of what is happening in the network and a number of analysis tools have been proposed. One such method proposed by [23] analyses security by generating an attack graph and builds a network security state which links the relationship between devices in the network.

On the other hand, even though host centric security mechanisms are able to achieve better depth than network centric approaches, they lack a holistic view [24, 25]. This means that in an environment where a large enterprise system spans more than one host or one network, a challenge exists to secure this information asset. Many web applications do span across hosts as it is normal practice to segregate the database from the business logic physically or virtually. A strictly network centric or host centric approach may not be adequate in such a case.

6. Application Centric Security

We therefore classify security mechanisms in a different dimension by focusing on what asset is being secured. We propose two classifications; infrastructure centric and application centric. Security models that are either host centric or network centric would fall under infrastructure centric because they focus on protecting the infrastructure used to support systems and applications. On the other hand, application centric security mechanisms would include all efforts and tools used to protect an application wherever and however it exists. This would include Collaborative Distributed Intrusion Detection techniques focused on protecting an application holistically. The techniques would require collaboration of several host or network centric components that work as one and monitor, collect and inspect traffic and transactions across multiple entities of the infrastructure. This is because any relatively large enterprise system will have transactions that result in the creation of audit data on multiple systems and it has been recommended by [26] that where a single business or system transaction will result in the creation of audit data on multiple systems, design consideration must be provided on how the complete audit information about that transaction will be collated and made available to an security auditor.

Two such considerations have been proposed by [8] and also by [10] and both suggest that the target web application system be re-written to embed some form of user identity in each database request so that there is a way for the database to log the additional information. Both approaches will at least assist to attain requirements of an independent database audit. However, we find that there are still shortcomings. If no retraceable link is created between the database and web server logs, other important aspects would be lost. For example, it is not be possible, from the database logs alone, to tell which sections or pages of the web application an attacker has used to penetrate the system. Also, the methods do not consider the expected roles based on the business process and logic of the system.

Another effort in toward providing complete audit information has been proposed by [9] and concentrates on reading network packets that are transmitted between the user and the web server and between the web server and database server. It reconstructs HTTP sessions and DB connection information and has a way of matching a user of each web-action to which web-action generated which SQL. The limitation of this method, however, is due to the fact that it reads text from the data packets to determine variable names and values. This prevents it from recognize the semantics of transaction data and further linking or classifying the log data. We realize that an additional consideration would need to be made at a higher level than that of the network for this to be a fully application centric security mechanism.

The closest attempt for holistic audit trail information is presented in an approach by [27] where an auditability tool is attached to the candidate web application that is to be monitored. This auditability tool sits between the web scripts and the database as shown in **Figure 2**.

Alongside logging database activity such as SQL queries run, the tool also logs all other relevant profile information such as current logged-on user, script file name, and so on in accordance to the security requirements. Results presented by [27] assert that the additional layer of logging causes little degradation in performance of the candidate web application and therefore a possible means of application centric monitoring and potentially also other aspects of security.

Application centric security mechanisms will therefore guide performing design considerations that are holistic in nature. They will describe the auditability requirements of transactions of a business process. For those applications that reside across multiple servers, considerations will include accurately collating audit data or logs and including all necessary data elements in the audit data. In a web application environment, for example, application centric security mechanism will require a design consideration to link log data produced by the web server, database server and any other participating server for each web request and response.

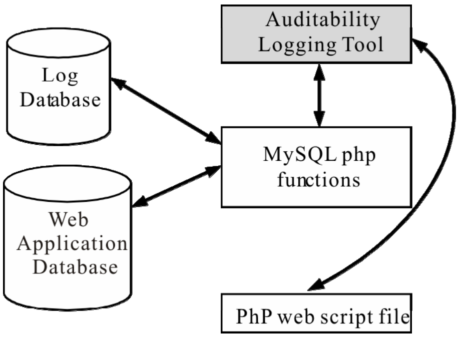

Figure 2. Holistic logging via intermediate tool [27].

A key outcome of application-centric security is that any log based audit and analysis method should give better results. For example, the four stage methodology that involves planning, discovery, vulnerability analysis and reporting [20] becomes more effective especially in the discovery phase because the centralized log consolidation phase is now straight forward and accurate. Additionally, signature-based detection and anomaly-based detection [5,28] can be used more effectively since more rich information is gathered. Likewise the data preparation phase of the Cross-Industry Standard Process for Data Mining (CRISP-DM) reference model [29] can now be performed and give much better results.

7. Vulnerability Scenario

When performing risk management, several controls are usually defined with the intention of mitigating risks caused by possible vulnerabilities in the system. Let us consider an example of the following risk; "*unauthorized access to the user administration panel*". Such a risk may be exploited due to vulnerabilities such as a weak password or when data transmission between the administrators and the server is in clear text. To mitigate it, a control would be that *all remote administration access is allowed only via encryption with SSL*. Another would be that *all administrator passwords must be strong and be changed regularly*. Subsequently, the security audit to assess whether controls are implemented and are working in the system will involve forming a control activity for each carry out several tests to advise on the same. In our example, the control activities would be two: 1) *Inspect all web server logs of admin pages* and 2) *Inspect password strength and expiration policy*. For each of the control activities, one test would be to *open all web server logs and observe the client IP and SSL encryption level for admin sections and perform a password strength and expiry test respectively*.

Let us consider a web environment with the above controls in place and discuss a possible loophole. A hacker accesses the public section of the website and, by exploiting a vulnerable page in the website, uses SQL injection to study the user tables in the database and makes some modifications into the database. The hacker creates a new user X with a really strong password and assigns admin level privileges and then attempts to login to the user administration section with these new details using encrypted SSL. The login attempt works! More importantly, when the auditor does their tests, they reveal that, indeed, no access was done in user administration sections in non-SSL connections and also that the passwords are strong enough to prevent password hacking. Even in the event that some of the actions by the hacker raise flags in the audit process, there is no way to holistically tell the exact extent of the damage done or the impact.

An immediate mitigating control to the problem would be to design extensive controls in the entire web solution including proper database user access planning, web development standards and so on. Specifically in our example above, an additional control to restrict administration access to specific IP address would lessen the risk. However, this still does not stop other risks associated with SQL injection.

Application Centric Security would go a long way in solving this problem in three folds. Firstly, actions would be logged showing what pages a certain user has visited and what database objects were accessed and any other dimension information that would facilitate building *Context*. This context information collected over a period of time would then be analyzed to identify *Context Maps* that represent system usage patterns. It would be easy to recognize the normal or allowed context maps and flag those that vary from the norm, thereby enabling detection of potential attacks. Secondly, it would allow fast discovery of the extent of damage once it has happened. This is as a result of its inherent ability to record user behaviour or activity across multiple web modules. Such information used in our earlier example would reveal the acts of that hacker and control measures can be added respectively. Thirdly it can be used as a means to prevent attacks. A tool that is able to read and mine state information from the contextual logs could recognize strange browsing habits and stop them. For example, a public user is normally expected to follow a particular pattern and context. The web pages visited and the database tables read or written would form a context map from which our hacker in the example above would be monitored against.

8. Conclusions and Future Work

We have discussed risks that affect web applications and leveraged on two classifications of security mechanisms to introduce a new, third, dimension of security. The two classifications are Network-Centric and Host-Centric Security and include tools and techniques such as network access control and antivirus software. We have established that as much as these methods are crucial in an enterprise they lack necessary depth to comprehensively analyze overall application security. The main reason is that an enterprise application will usually span across multiple server based on the number of tiers.

We have therefore classified security mechanisms as either Infrastructure Centric or Application Centric based on what asset is being secured. The former will have network-centric and host centric security mechanisms while the former will comprise of methods that try to holistically secure an application regardless of the number of locations or servers it spans physically or virtually. To describe application centric security, we have highlighted a number of efforts and research work toward it and, explained their shortcomings. Essentially, we've described that application centric security of a web application demands for design considerations to be made in ensuring complete auditability of transactions of a business process in the application. By doing so, log analysis and security audit is improved and in the end should influence system security as a whole.

Having established the need for application centric security mechanisms, there needs to be a formal specification on the requirements of the same. Definitely holistic logging and monitoring are a necessary starting point followed by more targeted holistic security measures such as access control.

REFERENCES

[1] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture," *Proceedings of the* 2000 *International Conference on Software Engineering*, Limerick, 4-11 June 2000, pp. 407-416.

[2] Acunetix, "SQL Injection: What Is It?" Web Application Security, 2011.

[3] N. Borhan, R. Mahmod and A. Dehghantanha, "A Framework of TPM, SVM and Boot Control for Securing Forensic Logs," *International Journal of Computer Applications*, Vol. 50, No. 13, 2012, pp. 15-19.

[4] M. Saleh, A. R. Arasteh, A. Sakha and M. Debbabi, "Forensic Analysis of Logs: Modeling and Verification," *Knowledge-Based Systems*, Vol. 20, No. 7, 2007, pp. 671-682. doi:10.1016/j.knosys.2007.05.002

[5] A. Miège and F. Cuppens, "Alert Correlation in a Cooperative Intrusion Detection Framework," *Proceedings of the IEEE Symposium on Security and Privacy*, Berkeley, 12-15 May 2002, pp. 202-215.

[6] B. Solms, "Information Security—A Multidimensional Discipline," *Computer & Security*, Vol. 20, No. 6, 2001, pp. 504-508. doi:10.1016/S0167-4048(01)00608-3

[7] K. R. Kumar, "A Model for Information Security Management in Government," *ISACA Journal*, Vol. 4, 2011.

[8] A. Roichman and E. Gudes, "Fine-Grained Access Control to Web Databases," *Proceedings of the* 12*th ACM Symposium on Access Control Models and Technologies*, *SACMAT*'07, Sophia Antipolis, 20-22 June 2007, pp. 31-40.

[9] Y. Gonen and E. Gudes, "Users Tracking and Roles Mining in Web-Based Applications," *Proceedings of the* 2011 *Joint EDBT/ICDT Ph.D. Workshop*, Uppsala, 25 March 2011, pp. 14-18.

[10] A. Shulman, "Web-Exposed Databases," *Enterprise Tech Journal*, 2007.

[11] D. R. Tsai, A. Y. Chang, P. C. Liu and H.-C. Chen, "Optimum Tuning of Defense Settings for Common on the Web Applications," 43*rd Annual* 2009 *International Carnahan Conference on Security Technology*, Zurich, 5-8 October 2009, pp. 89-94.

[12] R. P. Lippmann, *et al*., "Evaluating and Strengthening Enterprise Network Security Using Attack Graphs," MIT, Cambridge, 2005.

[13] G. V. Jourdan, "Securing Large Applications against Command Injections," 41*st Annual IEEE International Carnahan Conference on Security Technology*, Ottawa, 8-11 October 2007, pp. 69-78.

[14] N. Gust, C. Fournet and F. Z. Nardelli, "Reliable Evidence: Auditability by Typing," 14*th European Symposium on Research in Computer Security*: *ESORICS*, Saint-Malo, 21-23 September 2009, pp. 168-183.

[15] Common Criteria, "Security Assurance Components," *Common Criteria for Information Technology Security Evaluation*, Version 3.1, 2009.

[16] A. A. Yavuz and P. Ning, "BAF: An Efficient Publicly Verifiable Secure Audit Logging Scheme for Distributed Systems," *Annual Computer Security Applications Conference* (*ACSAC*), Honolulu, 7-11 December 2009, pp. 219-228.

[17] J. G. Cederquist, *et al*., "Audit-Based Compliance Control," *International Journal of Information Security*, Vol. 6, No. 2-3, 2007, pp. 33-151. doi:10.1007/s10207-007-0017-y

[18] A. R. Arasteh, M. Debbabi, A. Sakha and M. Saleh, "Analyzing Multiple Logs for Forensic Evidence," *Digital Investigation*: *The International Journal of Digital Forensics & Incident Response*, Vol. 4, No. 1, 2007, pp. 82-91.

[19] ISO/IEC, "Common Criteria for Information Technology Security Evaluation," 2009.

[20] R. Meyer, "Auditing a Corporate Log Server," SANS Institute InfoSec Reading Room, 2006.

[21] Moen and McClure, "Web Server Transaction Log Analysis Methodology," An Evaluation of US GILS Implementation, 1997.

[22] C. O. Jonathan, "Leveraging the Cloud for Software Security Services," Ph.D. Thesis, University of Michigan, Michigan, 2012.

[23] Lufeng, Z., Hong, T., YiMing, C., and JianBo, Z., "Network Security Evaluation through Attack Graph Generation," World Academy of Science, Engineering and Technology, 2009, pp. 412-415.

[24] M. E. Locasto, J. P. Janak and S. Stolfo, "Collaborative Distributed Intrusion Detection," Technical Report, Department of Computer Science, Columbia University, New York, 2004.

[25] A. R. Moheeb, M. Fabian and T. Andreas, "On the Effectiveness of Distributed Worm Monitoring," *Proceedings of the* 14*th Conference on USENIX Security Symposium*, Baltimore, 31 July-5 August 2005, p. 15.

[26] European Payments Council, "The Use of Audit Trails in Security Systems: Guidelines for European Banks," EPC AISBL Secretariat, Brussels, 2010.

[27] A. M. Kahonge, W. Okello-Odongo and E. K. Miriti, "Increasing Auditability in Web Application Security," (*IJEECS*) *International Journal of Electrical*, *Electronics and Computer Systems*, Vol. 11, No. 2, 2012. http://www.ijeecs.org/archive/03.December_2012_IJEECS_p21334.pdf 2012

[28] J. Stemmer, "Detecting Outliers in Web-Based Network Traffic," University of Twente, Enschede, 2012.

[29] CRISP-DM Consortium, "CRISP-DM 1.0. Step-by-Step Data Mining Guide," 1.0 Edition, SPSS, 2000.