Increasing Auditability in Web Application Security

Andrew M. Kahonge, William Okello-Odongo, Evans K. Miriti

School of Computing and Informatics, University of Nairobi, Kenya

***Abstract*— As more services become web based and open to a larger audience, security is become a key concern. We discuss the idea of auditability of a transaction in the web application environment and how current logs may not capture minimum information required to have a complete audit record. We then propose a solution to this that involves a design as well as a tool that can be integrated into an existing web application to generate supplementary logs of database activity and user profile information with a focus on auditability of transactions. Finally we talk about results of tests that we conducted of this tool on an actual web application.**

***Keywords*— Auditability, Web Security, Internet, Audit Trails, Log Management, Supplementary Logs**

I. INTRODUCTION

In the field of communication protocols, a definition is given by [1] 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. Logging and auditing is usually performed, not as a replacement of, but in addition to an (a priori) access control system [2]. In addition, sometimes the access control system itself is audited for flaws or errors. An audit of the access control mechanism can be sufficient when it is certain that the mechanism cannot be turned off between the audits. The use of audit logs has been recommended by security standards as a means of evaluating an IT system and therefore providing assurance [3].

Given that an auditable event is a single event (within a business process) that could lead to the compromise of the integrity and/or security of an information system and therefore directly or indirectly compromise a business process, recommendations by [4] state that an organization may select the relevant auditable events in the business processes that would be logged. As much as this would assist in keeping the logs to a minimum, it would miss out on attacks that by-pass the security controls in that business process and pose a risk to the data. Therefore we find that auditable events should encompass actions that are performed on the regardless of the business process where the actions are performed.

II. RISK, THREAT VULNERABILITY AND IMPACT

Risk can be defined as the potential impact (positive or negative) to an asset [5]. It's a combination of the likelihood that a particular vulnerability will be either intentionally or unintentionally exploited by a threat-source and the magnitude of the potential harm that could result. A threat is an unwanted event that may result in harm to an asset. Vulnerability is a weakness or flaw in a system that can be accidentally. Impact is the magnitude of the potential loss or seriousness of the event triggered or intentionally exploited. The determination of risk for a particular threat/vulnerability can be derived by multiplying the threat likelihood (e.g., probability) and the threat impact:

```
Risk = Threat likelihood * Threat Impact
```

Further, [6] identifies seven pillars of information security; Information security policy, Security awareness, Identity and access management, Network and data security, Monitoring, Risk assessment, Contingency. Threats can be from the outside or from the inside. The latter, includes threats introduced from within an organization by a trusted entity where the participant plays either a know role in the activity or unknowingly introduces a risk to the organizational security boundaries [7].

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. There are two main types of attacks facing web applications; server attacks and client attacks. Recently research indicates that among all attacks, SQL Injection and Cross-Site Scripting attack are the most common and most serious attacks [8]. Inherent Vulnerability in Web Server 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. Using an attack graph [9] have described several scenarios where an attacker gains access to the web server by exploiting remote-to-admin vulnerability, then gains access to the database server through remote-to-user vulnerability.

Inherent Weakness of Web Script a web script is also another source of vulnerability. The script may be developed with a weakness or fault that may go undetected. For example, a page that is only meant for administrators may fail to apply this control thereby allowing normal users to access privileged operations and alter data in the web application.

A. Detecting Web-Based Attacks

The two most common approaches to detecting web-based attacks are signature-based detection and anomaly based detection [10]. Signature-based detection relies on detecting patterns of known attacks to identify malicious behavior. While they are accurate, they have to be kept up-to-date with current attacks to be effective. Any attacks that are not in the signature or pattern database will therefore not be detected. This weakness can be exploited by creating different versions of a single attack.

Also, a number of security standards have been developed to help build more secure systems. Alongside this, scientific models have also been proposed to cope with security management. In a report by [11] several scientific models are

discussed and it is argued that two are most appropriate on an e-Commerce web application as opposed to a conventional software system. The report mentions that Abuse Cases and Business Process Modeling for Analysis and Design would work best since the employ Use Case Modeling and Business Modeling respectively. Other criteria used are that the two approaches are complete as far as the product life cycle is concerned, are business centered to motivate security by business means.

In broad terms, security mechanisms have been classified by [12] into two main categories; network-centric or host-centric, depending on their deployment model and which type of activity they observe and inspect. Examples of the former include network sensor devices and the latter are host or server based protection mechanisms.

III. Authentication Levels of a Web Application

When a user contacts a website as shown in step *a* of Fig. 1 , an authentication request may be prompted for username and password. Two common methods for this are script controlled authentication and web server controlled authentication. The former would include form-based techniques that render a web page on 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.

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.

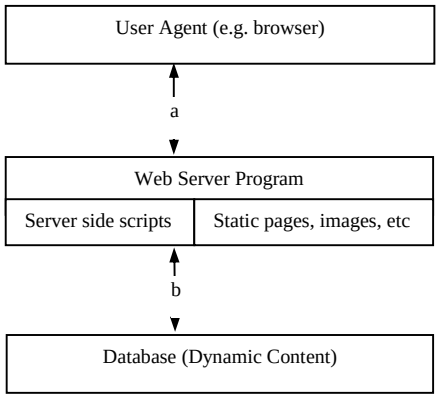

Fig. 1 Authentication levels in a web environment

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 Fig. 1. by either N:1 or 1:1 mapping of actual or external user and database user respectively. 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 [13] 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, including Drupal, use this approach [14].

IV. Log Management

A log has been defined by [15] as the record of events occurring within an organization's systems and networks. They are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network. Originally, logs were used primarily for troubleshooting problems, but logs now serve many functions within most organizations, such as optimizing system and network performance, recording the actions of users, and providing data useful for investigating malicious activity. Logs have evolved to contain information related to many different types of events occurring within networks and systems. Because of the increased use of application software such as web and database systems, the number, volume, and variety of computer logs has increased great. Thereby creating the need for computer security log management, which is the process for generating, transmitting, storing, analyzing, and disposing of computer security log data. In a web environment, a there can be a number of points where logs are generated including authentication servers, web servers, database servers, routers and firewalls.

A. Log Generation and Storage

Due to several hosts and applications generating their own logs, a number of complications arise when one needs to perform analysis. Some of these problems arise as a result of many log sources that have inconsistent log content, inconsistent timestamps, and inconsistent log formats.

A number of audit methodologies or methods, procedures and techniques, have been used to tackle this daunting task of log analysis. Meyer describes a four-stage methodology [16] as illustrated below. In this methodology is the planning phase that involves identifying log sources are and preparing a central log server to be the repository for log data. The log infrastructure from each source is also evaluated as well as the method of submitting this data to the central log server. This methodology compares to a Web server log analysis that had been proposed by [17] that includes determining the types of information server administrators and decision makers need; developing a program that can parse through, manipulate, and present value-added information from the log files; and analyzing the information generated from the program.

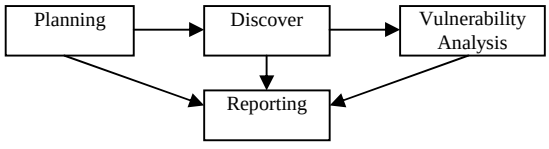

Fig. 2 Four stage testing methodology [16]

One key difference is that the latter puts more focus at the planning or initial stage and allows for possible inclusion of additional information into the logs and thereby improving auditability.

B. Minimum Information

To determine the minimum possible or sufficient information that a log should gather, we refer to the

recommendation made by [4]. This states that an audit record should have the following properties to enable a business process event to be auditable;

1. What type of event occurred, e.g. the server, system process, IP address, MAC address etc.,
2. When (date and time) the event occurred, e.g. time stamps where the event occurred
3. Where from the origination of the event occurred, e.g. source and destination addresses, user/process identifiers,
4. Where to the event occurred, e.g. the identity or name of the affected data, system or component,

V. PROPOSED LOG GENERATOR DESIGN

Our solution to gathering additional information to increase auditability involves setting up a tool that can log all activities and profile information of a user that is accessing a web application. We propose a design that allows integration into an existing web application system. The test bed we select for the host web application is a system developed in the Drupal framework. This is an Open Source content management system and uses PhP with MySQL database.

An environment running on Open Source technology has been selected purposively because OSS allow easy modification of a running application and also support some key features that are fundamental in our design; overriding system functions. As shown in Fig. 2. the standard interaction between PhP scripts and a MySQL database is that the PhP script will call a set of MySQL database functions such as *mysql_pconnect* and *mysql_query* to connect and run SQL database commands. The technique we are proposing involves setting up an intermediary stage between the web script and the inbuilt mysql functions, specifically mysql_query. See Fig. 3. In doing so we gain access and control of commands passed by the web script to MySQL engine. Meanwhile, we manage to log all activities generated at any time. To achieve this, we use two PhP functions; *rename_function* and *override_function*. This is illustrated in Fig. 3.

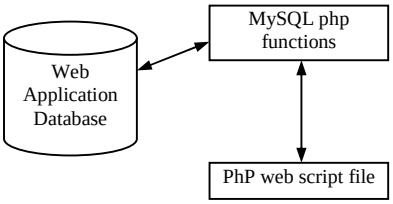

Fig. 3 Standard interaction of web script and database

After renaming and overriding the inbuilt mysql_query function such that all subsequent calls to it land on our own custom function, we have control of SQL statements issued by the PhP web script. The next step is to log these SQL statements and corresponding parameters into a log database separate from the web application's database. Alongside logging the database activity, all other relevant profile information such as current logged-on user can also be logged in accordance to the methodology being used or the security requirements being used.

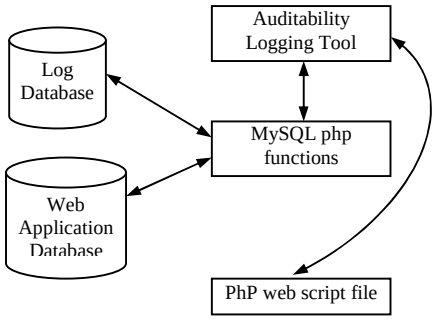

Fig. 4 Modified interaction

For example, is using the recommendations by [4], we can proceed to log the username and perhaps the roles or groups that the user belongs in order to fulfill requirement 4. On the other hand, if one followed the analysis methodology of [17], this would mean obtaining the set of fields or information to be logged from the administrator and other stakeholders concerned with security of the system.

VI. RESULTS

To show that the concept can really work, we developed a logging tool and installed it on a test-bed as shown above. The test-bed was running Drupal version 5 and we achieved the overriding of mysql_query function by inserting and include file command on the database.inc file that points to our tool. This guaranteed that all database queries would pass through our tool. The log database

The information we logged in each call to database commands was the username, user's roles, remote IP address, session id, script name and query string, and of course the database activity information. While logging this information, database activity information was parsed to extract constituent objects, operations, fields and data elements. These objects corresponded to a table, stored procedure or view being referenced. Operations would be database statements like Select, Insert, Update, Execute and so on. Fields would be the database fields and the data elements would be the values being passed on. The log database was designed to store this information accordingly.

Analysis of the web application's performance revealed that there was significant degradation in response time when our tool was running. Several tests showed that the SQL parser in our tool took considerable amount of time to extract the query semantics and was affecting performance the entire web application.

After removing the query parsing step in our tool, performance of the web application was restored. There was no significant reduction in either response time or processing time of web pages.

The final design on the tool involved two parts; log generated and log parser. The log generated would run in the web application and create log information including the raw database SQL statements issued at all times. The log parser would be executed offline and manually to break down the raw SQL statements logged by its counterpart into a format that can then be fed into some log analysis software for auditing.

VII. CONCLUSION AND FUTURE WORK

We have discussed auditability and the need to increase it within the web application environment to improve security. Current web platforms support logging at both web server and database server level. However auditability are often not met especially in cases where the authentication allows sharing of user account for the entire web application of a section of it. Further, these logs exist in silos and it is hard to consolidate them to meet minimum information requirements of auditability.

In moving towards a solution, we have discussed the design of an open-source based tool that can be used to improve auditability. The tool logs database activity as well as additional information about the user or other profile data that may be of interest as far as security is concerned. The design of this tool allows it to be installed on an existing web application or host web site. Consequently, we installed it on an actual web application developed on the Drupal framework and talked about tests and modifications that we performed on it.

The results show that the tool has no significant impact on the host application. Also, it's design is open to capture necessary profile information that the administrator or security stakeholder may deem important in increasing auditability of a transaction

Further work can be done toward integrating this design into the software development process of general applications and perhaps also those that not Open Source. Further, the SQL parser that we used in our tool had significant impact on the web application and we excluded it from the core log generator. Therefore, if this can be improved, then it can be integrated into the generator to achieve a compact design and tool.

REFERENCES

[1] Nataliya Gust, Cedric Fournet, and Francesco, Zappa Nardelli, "Reliable Evidence: Auditability by Typing," in *14th European Symposium on Research in Computer Security: ESORICS*, Berlin Heidelberg, 2009, pp. 168-183.

[2] J. G Cederquist et al., "Audit-based compliance control," *International Journal of Information Security*, vol. 6, pp. 33–151, 2007.

[3] ISO/IEC, *Common Criteria for Information Technology Security Evaluation.*, 2009.

[4] European Payments Council, *The use of audit trails in security systems: Guidelines for European banks*. Brussels: EPC AISBL Secretariat, 2010.

[5] Roger Meyer, *Auditing a Corporate Log Server*.: SANS Institute InfoSec Reading Room, 2006.

[6] Krishna Raj Kumar, "A Model for Information Security Management in Government," *ISACA Journal, volume 4*, 2011.

[7] T.C. Dodge, A.J. Ferguson, and D.M. Cappelli, "Introduction to Insider Threat Modeling, Detection, and Mitigation Track," in *45th Hawaii International Conference on System Sciences*, 2012, pp. 2381 - 2381.

[8] Dwen-Ren Tsai, A.Y. Chang, Peichi Liu, and Hsuan-Chang Chen, "Optimum Tuning of Defense Settings for Common," *Security Technology, 2009. 43rd Annual 2009 International Carnahan Conference on*, pp. 89-94, 2009.

[9] R.P. Lippmann et al., *Evaluating and strengthening enterprise network security using attack graphs*. MASSACHUSETTS: MIT, 2005.

[10] Stemmer Joel, University of Twente, 2012.

[11] A. Zuccato, "Towards a systemic holistic security management," Karlstad, Sweden, 2002.

[12] Clarke, Oberheide Jonathan, "Leveraging the Cloud for Software Security Services," in *Unpublished PhD Thesis*. Michigan: University of Michigan, 2012.

[13] A. Roichman and E. Gudes, "Fine-grained access control to web databases," in *In Proceedings of the 12th ACM symposium on Access control models and technologies, SACMAT*, New York, 2007, pp. 31–40.

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

[15] Karen Kent and Murugiah Souppaya, "Recommendations of the National Institute of Standards and Technology," *Guide to Computer Security Log Management*, 2006.

[16] Roger Meyer, *Auditing a Corporate Log Server*.: SANS Institute InfoSec Reading Room, 2006.

[17] Moen and McClure, "Web Server Transaction Log Analysis Methodology," *An Evaluation of U.S. GILS Implementation*, 1997.

Andrew M. Kahonge is a lecturer at the School of Computing and Informatics, University of Nairobi. He received his BSc in Computer Science at the same school in 2001 and proceeded to University of Birmingham and for MSc in Advanced Computer Science in 2004. His research interests include web security and distributed systems. He is also a Certified Information Systems Auditor (CISA).

William Okello-Odongo is a Professor in the School of Computing and Informatics in the University of Nairobi. He lectures and does research in a variety of areas in Computer Science including Computer Networks and Network Security.

Evans K. Miriti is a lecturer at the School of Computing and Informatics, University of Nairobi. He received his BSc in Computer Science at the same school in the year 1999 and continued on for an MSc in Applied Computer Science. Evans is actively involved in research in machine learning and robotics. He is also a Certified Information Systems Auditor (CISA).