International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 5, No.1 (Jan-2016)
http://dx.doi.org/10.12785/ijcds/050105

On the Identification of Required Security Controls Suitable for Converged Web and Mobile Applications

Devotha Nyambo[1], Zaipuna Yonah[2] and Charles Tarimo[3]

[1, 2] Nelson Mandela African Institution of Science and Technology, Arusha, Tanzania
[3] University of Dar es Salaam, Dar es Salaam, Tanzania

Received 26 Jun. 2015, Revised 11 Sep. 2015, Accepted 7 Nov. 2015, Published 1 Jan. 2016

Abstract: Contemporary development of information systems for service delivery is at the present a matter of bringing together use of web and mobile applications. However, this advancement in the field of computing is happening at the expense of increased security risks to the system users and owners. This is due to the fact that the advancement in systems security controls is not taking place at the same pace. In the converged web and mobile applications, developers lack formal development standards for security design and verification. As a result, applications are built with ad hoc implementations of security controls depending on context of usage.

In view of the above, this paper attempts to put forward a possible set of security controls considered to be suitable for addressing the security demands in converged web and mobile applications environments. To achieve this objective, use is made of a Livestock Data Center (LDC) system as a case study for analysis and reasoning. By design, the system can be accessed through web and mobile applications. The overall process involved here had the following phases: the first phase involved reviewing existing security controls and assessment of their usage in the converged web and mobile applications. The output from this stage was a review of security controls assessment report. The second phase involved devising and proposing a possible, security assessment model for the converged web and mobile applications. The last phase of this process, involved employing the proposed security controls assessment model and the case study to identify the possible security controls suitable for the converged web and mobile applications.

The approach used for security controls assessment involved a combination of white box and black box techniques. Whereas the platforms used for Web and mobile applications development were PHP and Java, respectively. This last item has been done to practically assess the security controls at an application level, and consequently to come up with suitable controls for the same.

Keywords: converged web and mobile applications, security controls, application level security.

1. Introduction

Development of web and mobile applications is fast changing from traditional standalone applications to support user mobility [1]. This enhancement has resulted into the convergence of web and mobile applications, whereby, an information system is designed to deliver services to its users through both web and mobile application platforms. Part of this trend is the increasing power of mobile devices which can no longer be underestimated due to the capability of facilitating user mobility in regards to information service delivery. Apart from that, mobile phones have enabled people with no computers to be connected to computing systems and access same services as accessed through desktop web applications. For example, smallholder farmers can now be connected to e-agriculture service delivery information systems through their mobile phones. However, the convergence of web and mobile applications has brought along with it new security concerns and challenges. The primary aim of the work presented in this paper is to at least shade some light on the required security controls suitable for the converged and mobile applications through security assessment and reporting.

A livestock Data Center (LDC) system has been selected as a custom case to facilitate our study. The LDC system is an integrated web and mobile applications system for mobilizing livestock data and provides decision support facility to various livestock stakeholders including smallholder farmers, livestock researchers, livestock extension officers and veterinary doctors in Tanzania. The LDC system prototype was developed for the purpose of this assessment. Web and mobile applications for the LDC system were developed under Rapid applications Development model (RAD) [2] by

Email-address: nyambod@nm-aist.ac.tz, zaipuna.yonah@nm-aist.ac.tz, charles@udsm.ac.tz

using livestock keeper's functionalities as selected features for the prototypes. By using a local database server, the controls were assessed. By using the developed LDC prototype system, assessment of the effectiveness of existing security controls in converged web and mobile applications was performed. The identified gaps were noted for further treatment. The identified gaps where found to cause the system to be susceptible to the following threats: sensitive data exposure, weak server side controls, client side injection, and weak authentication and authorization [3].

To this end, the paper is organized into further sections as follows; Section 2: Overview of existing security controls, Section 3: Motivation, Section 4: Review of related work, Section 5: Methodology, Section 6: Results and discussion, Section 7: Proposed security controls, Section 8: Conclusion and future research.

2. OVERVIEW OF EXISTING SECURITY CONTROLS

2.1 Security Controls for Web Applications

There has been a continual development of security controls for web applications due to an increasing number of new and automated tools for attacks. This section is aimed at exploring the existing controls for web applications security that are focused on preventing/mitigating four types of security threats; sensitive data exposure, client side injection, weak server side controls, and weak authentication and authorization.

Sensitive data exposure in web applications may occur through various means, like: through injection attacks, data leak and improper session management. The work by [4] presented a mechanism of preventing sensitive data exposure resulting from SQL injection attacks. They presented WASP (Web Application Sql-injection Preventer), an automated approach for protecting against SQL injection attacks. Other controls for protecting sensitive data include, data encryption at rest, on transit or when exchanged through browser. However, controls for protecting sensitive data depends on the type of data to be protected and the impact of threats modeling for that specific application.

Another contribution is made by [5] through development of Mylar platform for the purpose of storing encrypted data in a database server and decrypt only in users' browser. The use of HTTPS (Hyper Text Transfer Protocol Secure) in web applications is a prefered approach for robust authentication to protect sensitive data, but, many sites do not use this protocal due to demand of more CPU power and slower connections [6].

Authentication and authorization in web applications has been handled through various approaches like: the two factor authentication as either basic or digest authentication [7]. This mechanism has been a means of detecting unauthorized users trying to access web applications or accessing data on remote servers. The two-factor verification in web applications involves the use of password or user identification number and any other token/key/secret question/email confirmation to authenticate a user. However, the strength of the mechanism will depend on whether it is just a basic or digest authentication. Basic authentication does not encrypt user credentials during login while digest authentication will encrypt the user credentials during login and any time they are used or exchanged with the remote server through browser.

2.2 Security Controls for Mobile Applications

The issue of sensitive data exposure is more obvious in mobile applications, because many applications are designed to store some data on a user's device. For example, data stored in an Android device can easily be retrieved through simple commands in accessing the directory.../data/data/[package name] for Android devices [8]. Existing controls for this threat is either by encryption of all data when at rest in a device [9], or avoids the use of mobile device local storage. The latter is less practical due to the fact that, not all times a mobile application user is having reliable internet connection. For this reason, data can be temporally stored on a local database (SQLite database) waiting until connection is restored to be transferred to the intended remote database server. In addition, sensitive functions, e.g. those involved with user and device authentication are not directly embedded in an application source codes. This is because, mobile applications source codes can easily be decoded from .apk files to executable .jar files [10].

Reference [11] proposed an authorization framework for mobile applications that involves two level authentication mechanism, user authentication and device authentication. Device authentication is termed as device fingerprinting, whereby device's MAC address, Operating System details, Wi-Fi profile, location and SIM number are verified. These controls also depend on the sensitivity of the data that a mobile application is trying to access.

3. MOTIVATION

3.1 Problem statement

Development of secure applications is becoming an increasing challenge to mobile application developers due to the fact that, many develop applications without having a formal development process, frameworks, standard and language [12], [13], [14]. This pitfall is leading to insecure integration of mobile and web computing platforms. Apart from that fact, other observed pitfalls include among others: haste to market, lack of security acquaintance and newness of development languages, outsourced development, low budget set for application security, an assumption that a mobile device's Operating System is fully responsible for security, and cross platform applications development and compilation. To the best of our knowledge, there is no set of identified security controls in the literature to assist developers in building secure converged web and mobile applications. In addition, applications developers have no formal approach to assess effectiveness of used security controls in converged web and mobile applications.

3.2 Rationale of the Study

Fervor to this implementation was raised by the results of a study on security threats identification in converged web and mobile applications [3]. The authors presented an approach to threats modeling and specification of security requirements in the LDC system, which uncovered four prominent security threats in converged web and mobile applications. The prominent threats uncovered include: sensitive data exposure, weak server side controls, client side injection, and weak authentication and authorization. These are further explained in the following paragraphs.

a) *Sensitive data exposure:* exposure of sensitive data can generally occur in two ways, data at rest and/or data in transit. In all cases, web and mobile applications are vulnerable to this category of threat. However, their mitigation strategies may differ. Handling sensitive data in mobile devices is more challenging due to the fact that too much risk is associated with storing sensitive data in mobile devices [15]. These devices are exposed to many third party applications and services that can access stored data in a mobile device.

b) *Weak server side controls:* server side controls presents a significant demand in any database system. All issues with input validation and database access privileges are handled in this category. Any loophole in the server side processing may lead into loss or manipulation of data among other effects.

c) *Client side injection:* this category presents two types of threats to web and mobile applications, which are equally of high impact: injection attacks (specifically SQL injection), and Cross Site Scripting (XSS). These attacks can lead to a number of catastrophic impacts depending on the nature of application and sensitivity of data handled in the applications [16].

d) *Weak authentication and authorization:* client authentication and authorization is another category equally vulnerable as others already mentioned. Many applications tend to use traditional authentication procedure that relies only on user credentials. These credentials are no longer safe since they can easily be obtained by a malicious user. From these grounds, it is necessary to consider a strong authentication procedure for both web and mobile applications.

Our drive was equally built on a survey which involved about 54 respondents from enterprises, hubs and independent developers, which revealed that 63% of application developers are not aware of security risks assessment and 74% do not use any security frameworks/models in developing their applications, as shown in **Fig. 1**. However, during the survey developers revealed that, they are aware of mobile devices insecurities and have noted some new security challenges in their developments. Mentioned challenges included: client side injection, phishing and use of weak passwords for authentication and authorization. Password based authentication is a challenge because, developer has no way of assuring the safety of the password than the user him/herself. From these grounds, we realized there is a need of identifying required security controls suitable for converged web and mobile applications by using the LDC system web and mobile applications prototypes.

3.3 Significance of the Study

Focus of this paper is to firstly, assess the effectiveness of available security controls as applied to converged web and mobile applications, in which a security controls assessment model for converged web and mobile applications is presented. Secondly, to identify and propose required security controls suitable for converged web and mobile applications in respect to the four categories of prominent threats identified in [3].

4. REVIEW OF RELATED WORK IN SECURITY CONTROLS ASSESSMENT

Security mechanism assessment for web and mobile applications has been in practice with various approaches and tools. This section explores some of the existing approaches and tools for assessing security controls in web and mobile applications. Our goal here is to relate the approach and tools we have used to other approaches and tools reported in the literature. This review is limited to Android platform as far as mobile applications are concerned. An approach of static analysis and code review is presented, use of various local proxy tools is explained, as well as assessment by fault injection and behavior monitoring.

Among existing tools for Vulnerability Assessment and Penetration Testing (VAPT) for web and mobile applications include, Zed Attack Proxy – ZAP [17], AppScanner [18], and Fiddler [19]. The choice of a tool to use for application testing and assessment of security loopholes is mainly dependent on protocols in use and the tester's confidence in using the tool. ZAP has been practically used to penetrate web applications and uncover a number of vulnerabilities [20], [21]. Likely, AppScanner is an automated cloud based tool for analyzing mobile applications [22]. Fiddler is a web debugging proxy that works for http(s) protocol. Apart from web applications, fiddler has also been used for mobile applications with a combination of phone emulator [23]. Fiddler is a multiplatform tool for both web and mobile applications. One challenge with the use of these automated tools is that they all rely on predetermined set of rules and procedures; and the fact that they present a black box testing mechanism. As such, the capability to uncover new security threats, say those which were not predetermined, is hindered. Therefore, using these tools need to be accompanied by other approaches for best results.

Static and dynamic analysis presents an alternative approach to security controls assessment in web and mobile applications. Although, most literature show the application of this approach in web applications, the approach can as well be adopted for mobile applications as in [24]. Reference [25] presented the use of the static and dynamic web applications security assessment for two types of prominent web applications security: SQL injection and cross site scripting (XSS). By using nine web applications a static context sensitive information flow tracking analysis was done. Static analysis in their experiment, with given applications generated results that were used to optimize the dynamic analysis. Static and dynamic analysis is also presented by [26]. However, [26] focused also on assessing the correctness of an application sanitization process. The work contributed a novel idea of assessing a sanitization process by modeling the way an application process input values. Through this approach, previous unknown vulnerabilities (after sanitization routines) were uncovered. For effectiveness of sanitization procedures, a combination of various techniques has been recommended as in [27]. The approach used for web application security assessment included dynamic analysis with penetration testing.

Source code review presents another approach to assessing web and mobile applications security. This approach falls under white box testing; since a security analyst is aware of the source code and overall information flow and interactions of an application. Source code review for mobile applications has been practically used and elaborated by [28], [29], [30]. Source code review as a white box technique is important mostly to mobile applications whose codes can easly be decoded and studied by attackers. Through this, critical functions might be exposed. Reviewing source code in this context is also to make sure that none of the critical functions are exposed in the application code. Static code analysis is also motivated as a compliment to other assessment approaches in web applications for best results [27].

Another approach for conducting security assessment in web and mobile applications is by fault injection and behavior monitoring. This approach refers to the use of pre-defined patterns that are injected into an application and then observe how the application responds. Generally, it is a black box testing mechanism with no interest in the application source codes. As elaborated by [31] with the use of reverse engineering technique SQL injection patters were used to test for SQL injection. Depending on how the application responds to the injection, the Injection Knowledge Manager (IKM) selects the best injection pattern for more deep injection testing. Their approach combined all techniques in software engineering including: dynamic analysis, black box testing, fault injection, and behavior monitoring.
Reviewed work adequately describes security controls assessment in web and mobile applications in methods that we contend are not suitable for converged web and mobile applications. Therefore, based on this review two major findings have been made: 1) the approaches described involve one or two similar techniques of assessment, white box or black box assessment; 2) scenarios used in assessment includes either a web application or a mobile application, not both.

5. METHODOLOGY

5.1 Security Controls assessment model

Existing literature clearly elaborates the assessment of web and mobile applications security and effectiveness of used security controls in native web and mobile applications. Our goal here is to assess the effectiveness of existing security controls in converged web and mobile applications. We therefore, present our assessment model that adopts white box and black box testing techniques. This approach is of greater benefit to the mobile applications, which are more vulnerable than the web applications since, application code can easily be decoded and studied by a malicious part. The security controls assessment model for converged web and mobile applications is designed with five major building blocks as shown in **Fig. 2**: code review, improve security controls, automated black box penetration, security alerts assessment, and document implemented security controls. Description of these building blocks is as follows:

a) *Code review:* security code review as defined by our assessment model is a typical adoption of white box testing technique in software engineering. Review of business process and functional requirements, review of entry and exit points, and review of database transactions logic and syntax used presents three major processes in this block. The significance of code review in the context of converged Web and mobile applications lies on the fact that, knowledge of the mobile application source code is very important since the code is intended to reside within the client device. In addition, the use of automated penetration tools is not sufficient since the tools have no knowledge of the application's context and therefore it is possible to mislead or bypass certain vulnerabilities [32]. The OWASP code review guide was used for this purpose to provide the mechanics for reviewing code for specific types of vulnerabilities [32].

b) Improve security controls: after code review, we improve security controls implemented in the coding such as authentication and authorization functions, user input, and data handling techniques. This approach of modifying security controls enables adjustments to fit with the converged web and mobile applications. This is because, existing controls are focused on traditional web and mobile applications.

c) Automated black box penetration: by using automated tools, applications can be assessed to uncover vulnerabilities or ensure that implemented controls are of desired standard. As a typical adoption of black box testing technique, in this process, we have no concern with the source code.

d) Security alerts assessment: a penetration test will normally generate important alerts or notifications as test results to be considered for assessment. It is important that penetration sessions be logged in a separate file for review. Depending on alerts produced, the assessment process may go back to code review process and loop the assessment process until a desired level of security vulnerability is achieved.

e) Document implemented security controls: documentation of security controls used is as important as keeping records for best practices. This block will present key results in this assessment process as improved security controls for converged web and mobile applications are identified and documented.

5.2 Test Bed Environment Set up

Our test bed environment is designed for the black box assessment segment which involves an automated penetration to the web and mobile applications to assess the effectiveness of implemented security controls. Everything in the test environment runs on a Windows 8. Two types of penetration testing tools, ZAP 2.2.2 [17] and Fiddler2 [19], were selected for web and mobile applications, respectively. The server was WAMP (Apache 2.2.22) running locally for both, web and mobile applications. Specific method used for testing the mobile applications is by using a phone emulator and a proxy. This is an easy and cheapest method for Android applications [23]. Our design is that, the web and mobile applications access the same database server for all functionalities in the LDC system. The test bed environment model is as shown in **Fig. 3**.

5.3 LDC System Web and Mobile Applications Prototypes

The LDC system web and mobile applications prototypes were developed by following the Rapid Application Development (RAD) methodology with selected features for the purpose of this assessment. The web application was ran through Google Chrome browser while, the mobile application was ran through an Android phone emulator. Implemented functionalities include those of a livestock keeper as one category of users in the LDC system. The web and mobile applications prototype presents some of selected features from a livestock keeper's functionalities in the LDC system. Screenshots showing some parts of these applications are shown in **Figs. 4 to 7.**

Figures 6 and 7 show the LDC system mobile app prototype screen shots showing user registration and flock registration. The web application was built under Macromedia Dreamweaver, which supports frameworks including ASP, ColdFusion, Scriptlet, and PHP [33]; while, a native mobile application was built under the Android framework, which adopts the concept of Model View Controller [34]. The use of these frameworks was aimed at building an understanding of the status of converged web and mobile applications developed without a formal security framework. As our assessment model, **Fig. 2** presents it, applications' security controls can be improved following a code review process and penetration approach to test the effectiveness of the security controls for greater improvement of the security threats mitigation scheme.

6. RESULTS AND DISCUSSION

Security controls assessment used in this paper is as depicted in **Fig. 2**. White box assessment technique was employed as well as black box assessment technique. Code review process was done in order to understand coding pitfalls in development frameworks used in creating the test applications prototype. Knowledge from the OWASP code review guide [32] was borrowed in checklist preparation for mobile application codes and web application codes. Understanding the pitfalls is not enough for secure development, so we drive in black box penetration technique which helps in identifying security loopholes in applications.

6.1 Code Review and Black Box Penetration

Checklist prepared for code review in web and mobile applications was based on four categories of prominent threats in converged web and mobile applications as identified in [3]. The following paragraphs summarizes results obtained in each category of threat based in code review and automated penetration. Some screen shots have been cropped and included to display assessment results from code review and using Zed Attack Proxy for web application and Fiddler for mobile application.

a) Sensitive data exposure: development of a mobile application prototype by using Android development framework does not incorporate the implementation of security control to mitigate sensitive data exposure in SQLite databases and on transmission over the network. A sample source code that stores user credentials in a SQLite database showed that data were stored in plain text. However, this was not the case in the developed web application because, source code was embedded with encryption features to protect data from user input to database storage.

b) Client side injection: an application becomes vulnerable to this category of attack when user input is not strongly typed or pre-processed before embedding it with the SQL query. A sample code in **Figs. 8 and 9** shows data captured from user being sent to the database directly. This coding practice creates a loophole for malicious users to inject harmful queries that can result into manipulation of the database. Penetration results were not able to detect any injection vulnerabilities as shown in **Fig. 10**. This is contrary to the vulnerable code in **Fig. 8**. In a way, this further confirms the importance of combining the two assessment techniques.

c) Weak server side controls: at an application level, processing user input before inserting them into a database has not been performed to make sure that data stored is as expected. Moreover, used development approach allowed registration data from malicious users. This was observed from the database table holding registered users as per screen shot in **Fig. 11**.

d) Weak authentication and authorization: user authentication has been observed to rely only on stored user credentials. Apart from that, due to weak server side controls malicious users could register into the system by using automated tools such as Zed Attack Proxy, as shown in **Fig. 11**.This phenomenon will allow such user to login back into the system remotely and manipulate stored data. Moreover, Zed Attack Proxy results could show the credentials for and authentication functionality (**Fig. 12**), creating more loopholes in authentication and authorization. Due to the type of studied applications, no other vulnerability of high impact regarding authentication and authorization was found.

6.2 Summary of Results

Sensitive data exposure was highly observed by the use of a mobile application due to the fact that some data stored into the SQLite database could be fetched and read through a SQLite browser. From a Web application point of view, data can be protected at an acceptable level of vulnerabilities, but the use of a mobile application risks the security of these data and, hence, mobile devices cannot be trusted to keep sensitive data. The observed mitigation scheme is not to store any sensitive data in an SQLite database. Client side injection vulnerabilities observed in this assessment could be noted through code review but not in penetration testing (**Fig. 8** and **Fig. 10**). These findings support our key idea of combining white box and black box testing techniques. Although this assessment was able to reveal some coding pitfalls, which can lead into injection attacks like SQL injection, a thorough assessment can be done to completely mitigate client side injection attacks. By using the LDC system, we have created a stage in the development process for developers to review their source code and consequently uncover flaws that could not be found through black box penetration.

The use of one level/factor authentication has been found to have pitfalls as shown in **Fig. 11**. If and only if the system could register a user together with device used, this security breach could not be possible because it is obvious that the circled credentials came from the same device. We strongly support and recommend the two factor authentication mechanism as suggested by [11]. A successful user authentication should be followed by device authentication for both web and mobile applications. Server side controls in mobile application have not been assessed and we leave this for future work.

7. PROPOSED SECURITY CONTROLS

Based on the results from security controls assessment in section Six, we are proposing suitable security controls that can be used to mitigate the four categories of threats as identified in [3]. Knowledge towards identification of these security controls was partly borrowed from [11] and OWASP [35]. **Fig. 13** shows a summary of proposed application level security controls. A description of these controls for application level security is aimed at providing a clarification on how they can be implemented to make sure that data accessed or used through web and mobile applications is secured against mentioned categories of security threats. The list of proposed security controls is not exhaustive and will depend on the type, context, audience and importance of an application or system. With that point in mind, development teams can identify more and stronger security controls as needed. Description is accompanied by sample codes implemented to test the effectiveness of the proposed controls in the LDC Web and mobile applications prototypes.

7.1 Sensitive Data Exposure

a) *Encrypt all sensitive data:* Data protection at application level will very much depend on type and sensitivity of data. Such data might be authentication credentials, or data temporally stored in local databases; e.g. SQLite database in Android mobile devices. However, some devices provide a feature where users can enable automatic data encryption such as in iOS 5 by using 256-bit Advanced Encryption Standard (AES) [36]. Likewise, Android devices offer on-device data encryption, only when users enable it. In protecting sensitive data, developers should not rely on user options to encrypt data but enforce encryption controls at application level to lower the impact of exposure. Encryption/hashing algorithms differ in digest strength due to differing number of bits used, again depending on data sensitivity selection can range from an algorithm with few bits to many bits hash values. For example, in the case reported in this paper, we implemented the use of SHA-1 (Secure Hash algorithm 1) which produces a 160 bit hash value. SHA-1 is resistant to brute force attacks and best of use in digest authentication rather than the use of MD5 that produces a 128 bit hash value and easily decoded. Depending on type of data stored and its sensitivity, a stronger encryption algorithm can be chosen for the same. **Fig. 14** shows a screen short of sample codes (PHP codes) implemented to encrypt data.

b) *Do not enable auto-complete in input forms*: Enabling auto-complete in input forms allows a browser to store a record of input data. This is risky when capturing sensitive data, because a malicious user might retrieve such data through browser histories. In both web and mobile applications, this option should be disabled for all sensitive data input. For example, in HTML input forms, this option is disabled at the input options (autocomplete=”off”).

7.2 Client Side Injection

a) *Validate all user inputs*: User input validation is a first step of filtering input data to avoid acceptance of irrelevant data or malicious data. Input constraints enforced on input forms should explicitly be enforced on the database as well, e.g. input type and number of allowed values. Functionalities to count and validate input length and input type should explicitly be defined in both web and mobile applications.

b) *Enforce the use of bind variables*: Client side injection can be categorized in the form of Cross Site Scripting attacks or Injection types of attacks (e.g. SQL injection). We present a tested approach of mitigating SQL injection attacks in converged web and mobile applications by using bind variables in prepared SQL statements. The use of bind variables has a number of benefits in overall performance assessment including: increased throughput and prevention of SQL injection attacks. Using bind variables or parameterized SQL queries forces automatic escape of string by JDBC (Java database Connectivity) driver, the resulting string is treated as a typical user data that cannot be interpreted by a SQL database server. For example, insertion of an injection query “1 ‘or 1 = 1” will not be successful because the input will be escaped and cannot be executed directly by the database server. **Fig. 15** shows a screenshot of a sample PHP code for escaping user data through bind variables. Similar approach can be implemented in Java code for all user inputs, which is sent to a database as shown in **Fig. 16.** The java script shown describes the bind variables in prepared statement with hard coded input, but even if the input is fetched from input forms it will be passed as variables in the pstmt.setString() or pstmt.setInt() functions.

7.3 Weak Server Side Controls

Server side controls are prevalent in mobile applications and little has been done to mitigate the impact caused by this threat. Among reasons for this prevalence include: newness to mobile application programming languages, cross platform development and compilation techniques. We recommend the use of Model View Controller (MVC) [34] approach in applications development, which reduces the exposure of server side controls source codes to application users. Moreover, input forms can be embedded with server side controls by defining them in built-in HTML controls using runat=”server” attribute. Also, a white list of users and devices can be used to strongly enforce controls at server side, and use of mapping values can also be used to hide page redirects options.

8. CONCLUSION AND FUTURE RESEARCH

In this paper, we have presented an assessment of security controls and consequently identified some new controls suitable for converged web and mobile applications in systems such as the LDC system. By using white box and black box assessment techniques, we have presented an assessment model to help applications developers conduct security controls assessment to their web and mobile applications. Security controls presented in this work for converged web and mobile applications are suitable for systems similar to the LDC system. As such, they can be strengthened to suit a more robust system with web and mobile applications.

For the purpose of this work, application prototypes used were of selected features only (livestock keeper's functionalities) and so could not capture all functional requirements of the LDC system. As a result of this, data analysis for decision support functionality was not included in the prototypes. We leave this for future research. In addition, the authentication approach presented in this paper has not been implemented to attest its effectiveness in web and mobile applications. As a theoretical recommendation, we believe the approved success of Zero Trust Pattern from [11] signifies the success of our proposed approach. This is because, our approach build on top of [11] by adding user credentials on device authentication to have one complete authentication scheme for users and devices.

Results of the study show the susceptibility of the system to the mentioned security threats. This can be used as a proof to validate that existing security controls for traditional web and mobile applications security are inadequate for the converged web and mobile applications. Therefore, there was a need to identify suitable security controls for converged web and mobile applications to mitigate the four categories of security threats. In terms of contributions to the body of knowledge, generally, this paper has three major contributions: 1) we have presented a design for security controls assessment model in converged web and mobile applications; 2) we have presented an assessment of existing security controls to observe their effectiveness in converged web and mobile applications, and; 3) we have proposed a set of required security controls suitable for the converged web and mobile applications. In addition, this paper does not present results from an exhaustive assessment, so development teams can follow and loop the procedure for assessment until an acceptable level of vulnerabilities is obtained.

To this end, we strongly recommend an inclusion of a security controls assessment model into a security framework for converged web and mobile applications. Through this, applications will be going to the market with a significant level of trust from both users and developers. Our future direction towards secure converged web and mobile applications is the design and development of a holistic-security-framework to assist applications developers build secure applications.

ACKNOWLEDGMENT

We sincerely appreciate the support on this work from the Nelson Mandela African Institution of Science and Technology (NM-AIST) through the school of Computation and Communication Science and Engineering (CoCSE).

REFERENCES

[1] M. Chen, D. Zhang & L. Zhou, "Providing web services to mobile users: the architecture design of an m-service portal", International Journal of Mobile Communications, Vol. 3, no. 1, 2005, pp:1-18.

[2] Zeinoun, "The Rapid Application Development Process", Cambridge Technology Enterprises, Inc, 2005.

[3] D. Nyambo, Z. Yonah, and C. Tarimo, "An Approach for Systematically Analyzing and Specifying Security Requirements for the Converged Web-Mobile Applications", International Journal of Computing and Digital Systems, Vol. 3, no. 3, 2014, pp: 207-216.

[4] W. G. Halfond, A. Orso and P. Manolios, "WASP: Protecting Web applications using positive tainting and syntax-aware evaluation", Software Engineering, IEEE Transactions on. Vol. 34, no. 1, 2008, pp:65-81.

[5] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan, "Building web applications on top of encrypted data using Mylar", MIT CSAIL and † Meteor Development Group, 2013, pp:1-16.

[6] Visual Studio. Setting up HTTPS with Secure Sockets Layer (SSL) for Team Foundation Server. Available: http://msdn.microsoft.com/en-us/library/aa833872.aspx#DisAd. 2013, Last accessed 6th May 2014.

[7] G. Liu, "Proxy based adaptive two factor authentication having automated enrollment, U.S. Patent Application 10/463,369, 2003.

[8] J. Lessard, and G. Kessler, "Android Forensics: Simplifying Cell Phone Examinations", Digital Device Forensics Journal. Vol. 4, no. 1, 2010, pp:1-12.

[9] R. Rayarikar, S. Upadhyay and P. Pimpale, "SMS Encryption using AES Algorithm on Android. International Journal of Computer Applications. Vol. 50, no.19, 2012, pp:12-17.

[10] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A Study of Android Application Security", In USENIX Security Symposium, 2011, pp: 123-130.

[11] K. P. Yadav and R. Mishra, “Mobile Application Security Framework”, IT Best Practices Alert, Network World, 2013, pp:1-5.

[12] T. Wasserman, “Software engineering issues for mobile application development”, FoSER 2010.

[13] J. Lounsbury, “Application Security: From Web to Mobile”, Different Vectors and New Attacks. Security in Knowledge, 2013, pp:2-30.

[14] K. Johnson and J. Jardine, “2013 SANS Mobile Application Security Survey: A SANS White Paper”, SANS Analyst Program, 2013.

[15] J. Payne, “Secure mobile application development”, IT Professional, Vol. 15, no.3, 2013, pp:0006-9.

[16] R. Johari and P. Sharma, “A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection. In Communication Systems and Network Technologies (CSNT), 2012 International Conference on, pp: 453 – 458.

[17] S. Bennetts, “OWASP Zed Attack Proxy”, In AppSec USA 2013 Owasp, 2013.

[18] S. Amini, J. Lin, J. Hong, J. Lindqvist and J. Zhang, “Towards Scalable Evaluation of Mobile Applications through Crowdsourcing and Automation”, 2012, CMU-CyLab-12-006.

[19] E. Lawrence, “Fiddler: The free web debugging proxy for any browser, system or platform”, Available: http://www.telerik.com/fiddler, 2007, Last accessed 7th Feb 2014.

[20] S. Whittaker, “Hands on Web App security testing”, Available: http://vsltd.co/sectraining, 2012, Last accessed 7th May 2014.

[21] A. Crenshaw, “Darknets and hidden servers: Identifying the true IP/network identity of I2P service hosts, Black Hat DC, Vol. 201, no. 1, 2011, pp: 120 – 129.

[22] S. Amini, J. Lin, J.I Hong, J. Lindqvist, and J. Zhang, “Mobile Application Evaluation Using Automation and Crowdsourcing”, Available: http://petools.soic.indiana.edu/files/2013/06/petools2013_submission_10-2.pdf, 2013, Last accessed 7th May 2014

[23] K. Shah, “Penetration Testing Android Applications. White paper”, Available: <http://www.foundstone.com, 2013, Last accessed 12th Feb 2014.

[24] S. R. Basavala, N. Kumar and A. Aggarwal, “Mobile Applications-Vulnerability Assessment Through the Static and Dynamic Analysis”, In Proceedings of the Conference on Advances in Communication and Control Systems-2013, Atlantis Press, 2013, pp: 87 – 95.

[25] M.S. Lam, M. Martin, B. Livshits and J. Whaley, “Securing web applications with static and dynamic information flow tracking”, In Proceedings of the 2008 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation, 2008, pp: 3-12.

[26] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, “Saner: Composing static and dynamic analysis to validate sanitization in web applications”, In Security and Privacy, SP 2008 IEEE Symposium on, 2008, pp: 387-401.

[27] Petukhov and D. Kozlov, “Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. Computing Systems Lab, Department of Computer Science, Moscow State University, 2008, pp13.

[28] Shabtai, Y. Fledel, U. Kanonov, Y. Elovici and S. Dolev, “Google Android: A state-of-the-art review of security mechanisms”, arXiv preprint arXiv:0912.5101, 2009.

[29] G. Delac, M. Silic and J. Krolo, “Emerging security threats for mobile platforms”, In MIPRO, 2011 Proceedings of the 34th International Convention, IEEE, 2011, pp: 1468-1473.

[30] A. K. Jain and D. Shanbhag, “Addressing security and privacy risks in mobile applications” IT Professional, Vol. 14, no. 5, 2012, pp: 28-33.

[31] Y. W. Huang, S. K. Huang,T. P. Lin and C. H. Tsai, “Web application security assessment by fault injection and behavior monitoring. In Proceedings of the 12th international conference on World Wide Web 2003, pp. 148-159.

[32] OWASP, “OWASP Code Review Guide 2008 V1.1”, OWASP Foundation, 2008, pp: 10-15.

[33] S. Bansal, “Open Source Alternative to Dreamweaver, Available: http://www.opensourcealternative.org/alternatives/web-development/open-source-alternative-to-dreamweaver/, 2012, Last accessed 15th May 2014.

[34] Z. Mednieks, L. Dornin, G. B. Meike and M. Nakamura, “About the Android Framework”, In: A. Oram and Roumeliotis, R Programming Android, 2nd ed, United States of America: O'reilly Media, 2012, pp:171-199.

[35] OWASP, “OWASP Top 10 - 2013. The Ten Most Critical Web Application Security Risks” The Open Web Application Security Project, 2013, pp:1-22.

[36] R. Shelson, “Mobile data encryption techniques: On-device and on-the-go”, Available: http://searchconsumerization.techtarget.com/tip/Mobile-data-encryption-techniques-On-device-and-on-the-go, 2012, Last accessed 22th May 2014.

Ms. Devotha Nyambo is a holder of MSc. Degree in Information and Communication Science and Engineering from the Nelson Mandela African Institution of Science and Technology (NM-AIST), Tanzania. She is currently focused on design and development of secure web-mobile applications for livestock management and genetic improvement systems. Her ambition is to model and develop a smallholder dairy farm computer object that will enhance adoption of various genetic improvement systems in African's smallholder livestock keeping.

Eng. Dr. Zaipuna O. Yonah MIET, MIEEE - holds a B.Sc. degree (with Hons - 1985) in Electrical Engineering from University of Dar es Salaam - Tanzania; and M.Sc. (1986) and PhD (1994) Degrees in Computer-Based Instrumentation and Control Engineering from the University of Saskatchewan, Saskatoon - Canada. In Tanzania, he is a Registered Consulting Engineer in ICTs. Dr. Yonah has over 30 years of practice. His work spans the academia, industry and policy making fields. He is currently associated with The Nelson Mandela Institution of Science and Technology – (school of Computation and Communication Science and Engineering), and the IEEE Inc.
He is one of the pioneers driving the national broadband agenda in Tanzania. He believes that ICTs, as tools for development, promise so much: *interactivity, permanent availability, global reach, reduced per unit transaction costs, creates increased productivity and value, jobs and wealth, multiple source of information and knowledge*. Armed with such a belief, his current work aims at creating and delivering value through ICT-enabled services in the shortest times possible. His research interests include: ICT4D, Cyber Security, ICT Policy and Regulation, Mobile and Web applications, high-capacity broadband networks, Intelligent Instrumentation and Control Engineering; and ICT enabled 21st Century Education delivery (ICT4E).

Dr. Charles N. Tarimo is an active researcher on ICT security issues, with research interests focused on operational and practical issues with regard to aspects of security requirements development, designing, implementation, and management of different technical and non-technical ICT security controls within organizations/enterprises as well as research on similar issues at the national level. He has been collaboratively working with other researchers to carry out different research studies in the field of Information and Communication Security and published the research findings at various International Conferences. Dr. Tarimo is currently an employee of the University of Dar es Salaam, working at the College of Engineering and Technology, serving as the University's ICT Manager. But also he is involved in the teaching of related subjects in computer engineering, such as computer hardware and software engineering, computer and networks security, computer networking as well as artificial intelligence.

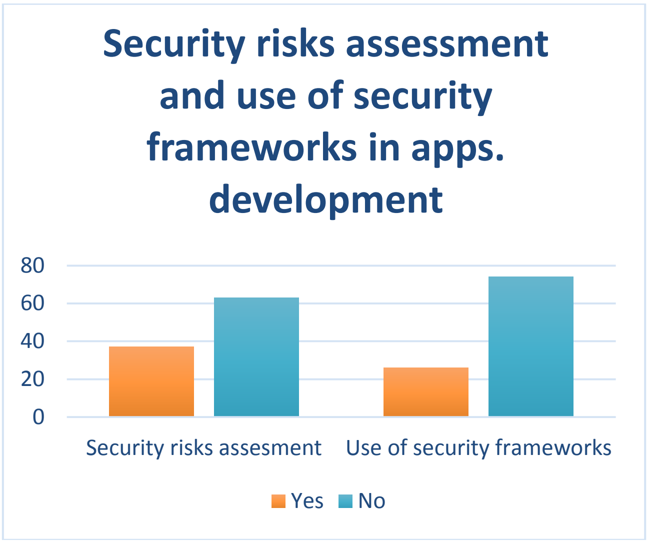


Figure 1: Respondents' results on security risks assessment and use of security frameworks.

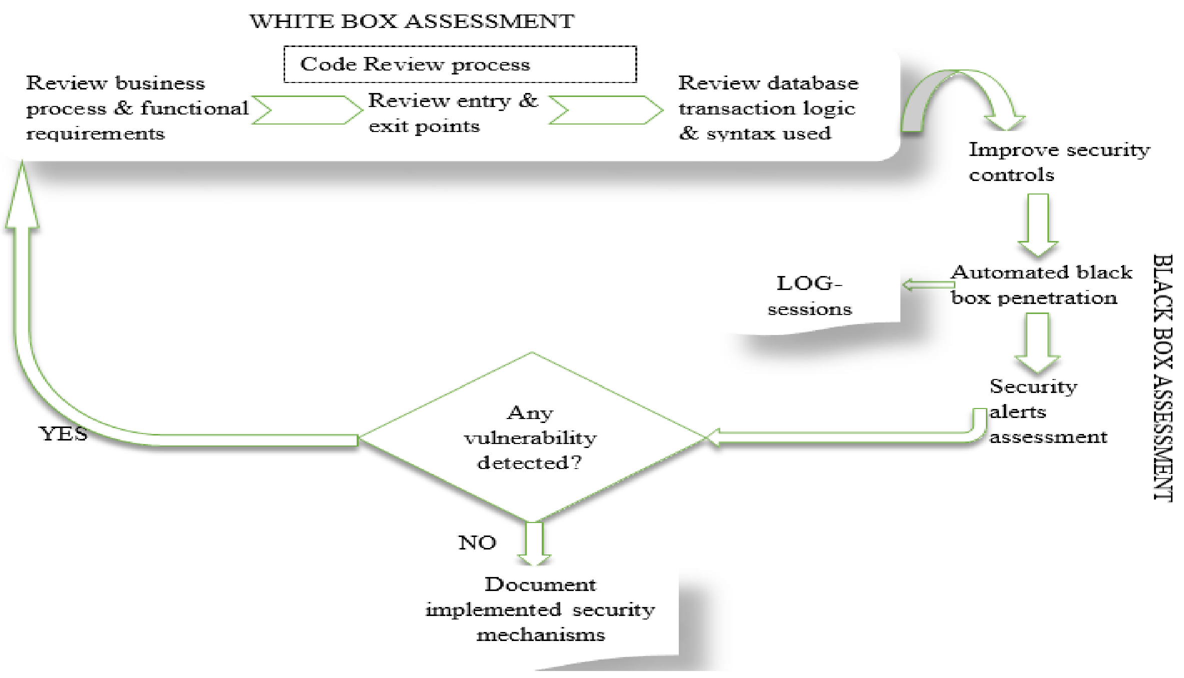

WHITE BOX ASSESSMENT
Code Review process
Review business process & functional requirements
Review entry & exit points
Review database transaction logic & syntax used
Improve security controls
Automated black box penetration
LOG-sessions
Security alerts assessment
BLACK BOX ASSESSMENT
Any vulnerability detected?
YES
NO
Document implemented security mechanisms

Figure 2: Security Controls Assessment Model for Converged Web and Mobile Applications.

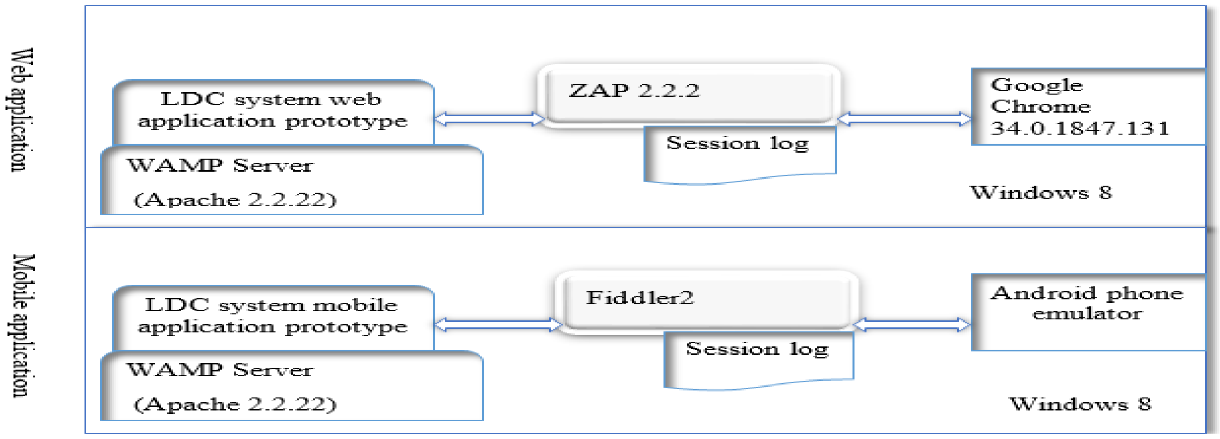
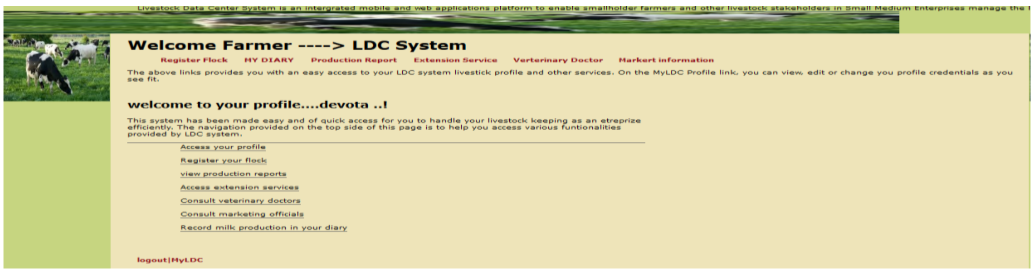

Web application
LDC system web application prototype
WAMP Server (Apache 2.2.22)
ZAP 2.2.2
Session log
Google Chrome 34.0.1847.131
Windows 8
Mobile application
LDC system mobile application prototype
WAMP Server (Apache 2.2.22)
Fiddler2
Session log
Android phone emulator
Windows 8

Figure 3: Black box penetration test bed environment model.


Livestock Data Center System is an intergrated mobile and web applications platform to enable smallholder farmers and other livestock stakeholders in Small Medium Enterprises manage the

Welcome Farmer ----> LDC System

Register Flock MY DIARY Production Report Extension Service Verterinary Doctor Markert information

The above links provides you with an easy access to your LDC system livestick profile and other services. On the MyLDC Profile link, you can view, edit or change you profile credentials as you see fit.

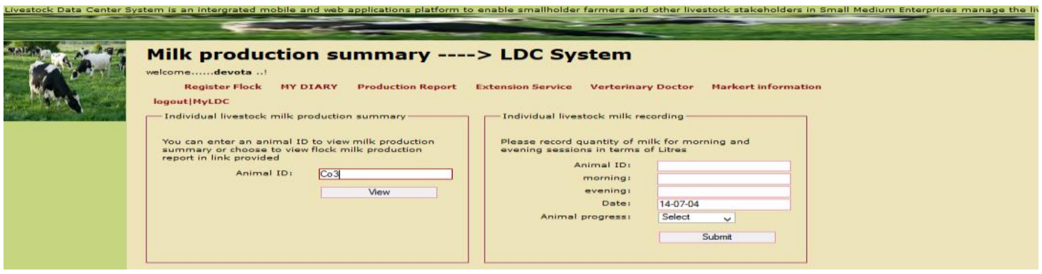
welcome to your profile....devota ..!

This system has been made easy and of quick access for you to handle your livestock keeping as an etreprize efficiently. The navigation provided on the top side of this page is to help you access various funtionalities provided by LDC system.

Access your profile
Register your flock
view production reports
Access extension services
Consult veterinary doctors
Consult marketing officials
Record milk production in your diary

logout|MyLDC

Figure 4: LDC system web application prototype showing farmer's home screen.


Livestock Data Center System is an intergrated mobile and web applications platform to enable smallholder farmers and other livestock stakeholders in Small Medium Enterprises manage the li

Milk production summary ----> LDC System

welcome......**devota** ..!

Register Flock MY DIARY Production Report Extension Service Verterinary Doctor Markert information

logout|MyLDC

Individual livestock milk production summary

You can enter an animal ID to view milk production summary or choose to view flock milk production report in link provided

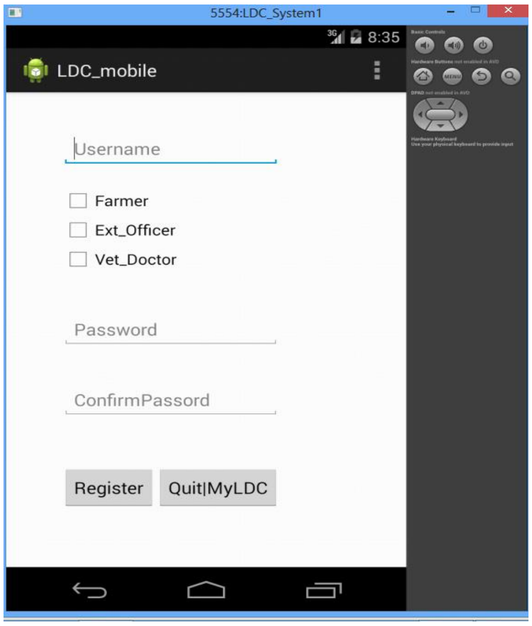
Animal ID: Co3

Individual livestock milk recording

Please record quantity of milk for morning and evening sessions in terms of Litres

Animal ID:
morning:
evening:
Date: 14-07-04
Animal progress: Select

Figure 5: LDC system web application prototype showing input forms for milk production.



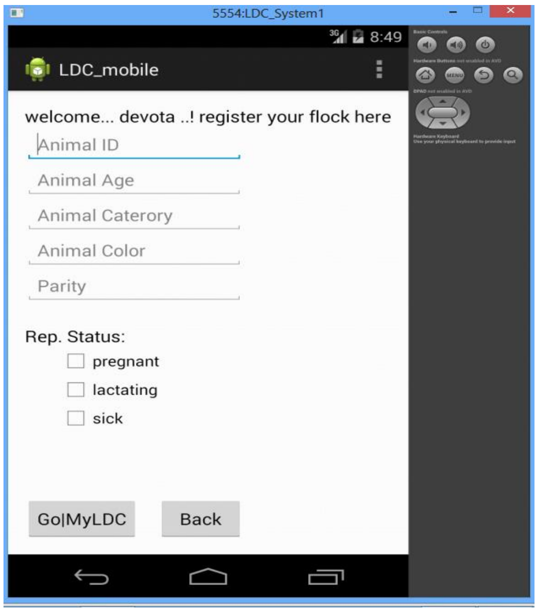
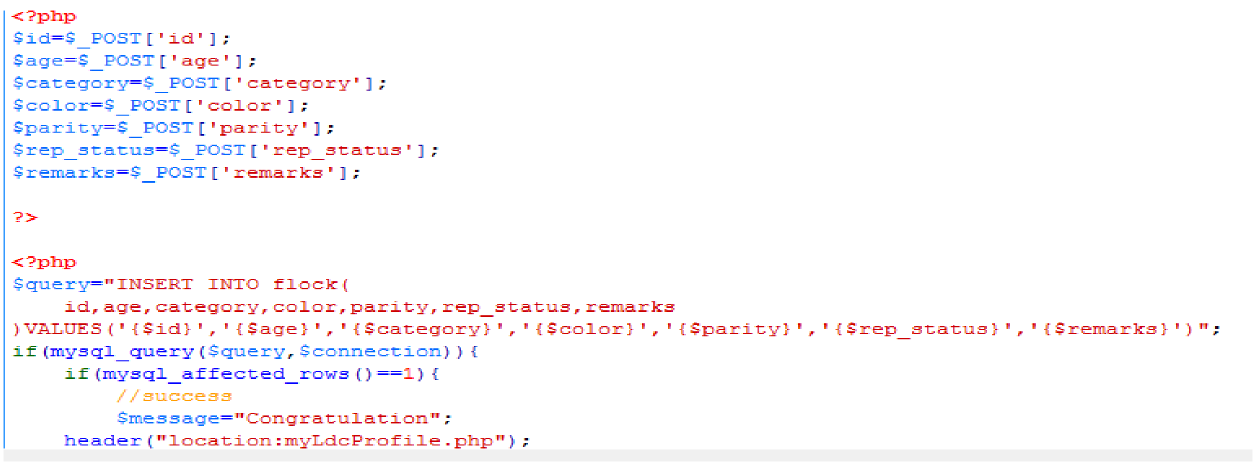


Figure 6: User registration. Figure 7: Flock registration.

a)

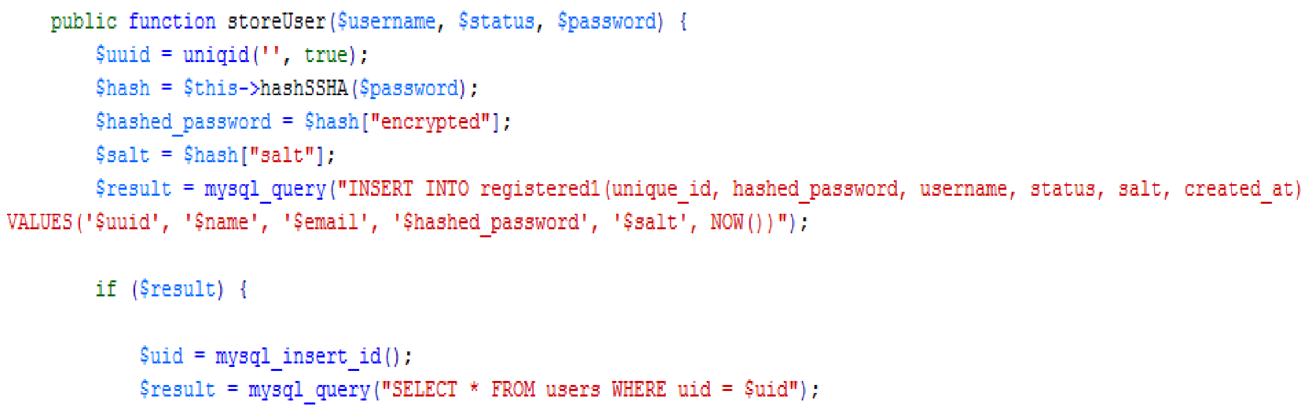


```
<?php
$id=$_POST['id'];
$age=$_POST['age'];
$category=$_POST['category'];
$color=$_POST['color'];
$parity=$_POST['parity'];
$rep_status=$_POST['rep_status'];
$remarks=$_POST['remarks'];

?>

<?php
$query="INSERT INTO flock(
    id,age,category,color,parity,rep_status,remarks
)VALUES('{$id}','{$age}','{$category}','{$color}','{$parity}','{$rep_status}','{$remarks}')";
if(mysql_query($query,$connection)){
    if(mysql_affected_rows()==1){
        //success
        $message="Congratulation";
    header("location:myLdcProfile.php");
```

Figure 8: Screen shot of a vulnerable flock registration source code.

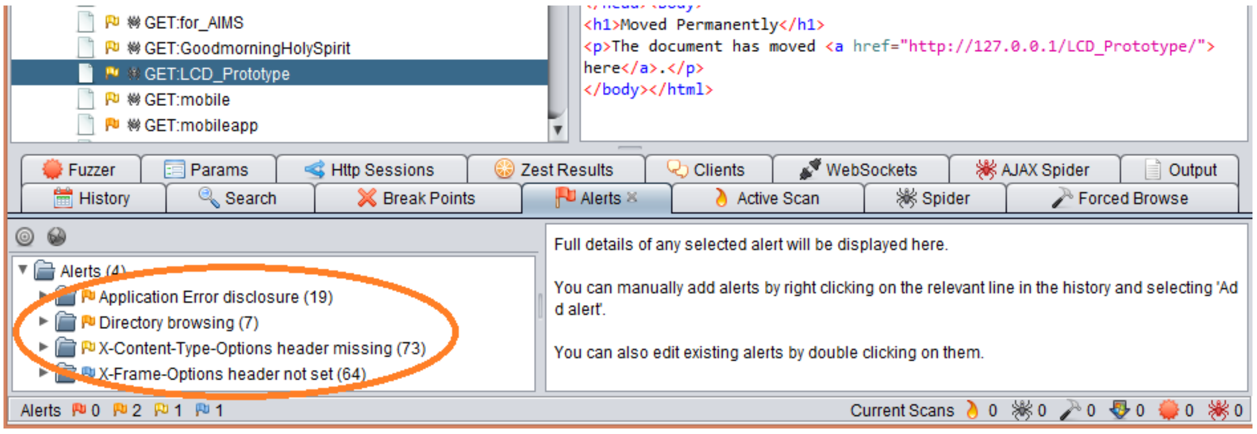


```
    public function storeUser($username, $status, $password) {
        $uuid = uniqid('', true);
        $hash = $this->hashSSHA($password);
        $hashed_password = $hash["encrypted"];
        $salt = $hash["salt"];
        $result = mysql_query("INSERT INTO registered1(unique_id, hashed_password, username, status, salt, created_at)
VALUES('$uuid', '$name', '$email', '$hashed_password', '$salt', NOW())");

        if ($result) {

            $uid = mysql_insert_id();
            $result = mysql_query("SELECT * FROM users WHERE uid = $uid");
```

Figure 9: Screen shot of a vulnerable user registration source code.



```
GET:for_AIMS
GET:GoodmorningHolySpirit
GET:LCD_Prototype
GET:mobile
GET:mobileapp
```

```
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://127.0.0.1/LCD_Prototype/">
here</a>.</p>
</body></html>
```

```
Alerts (4)
  Application Error disclosure (19)
  Directory browsing (7)
  X-Content-Type-Options header missing (73)
  X-Frame-Options header not set (64)
```

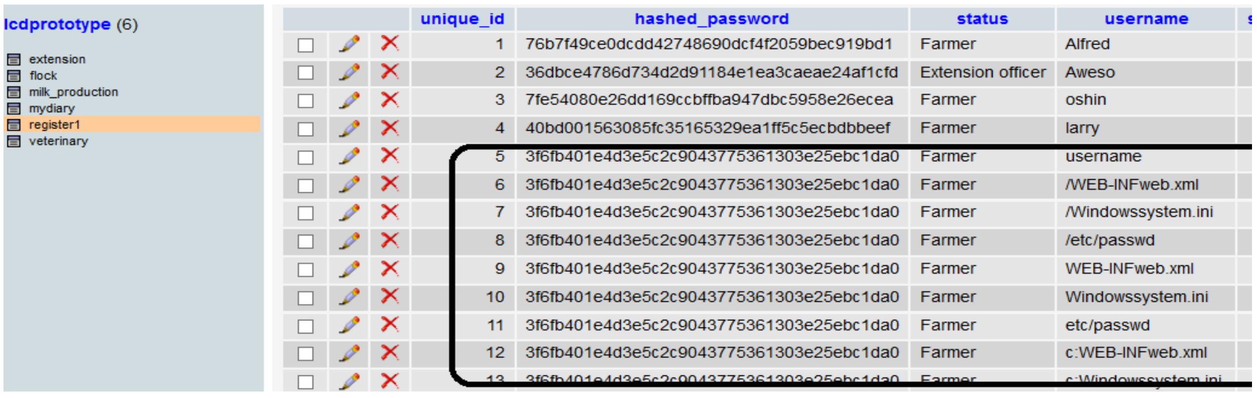
Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Ad d alert'.

You can also edit existing alerts by double clicking on them.

Alerts 0 2 1 1 Current Scans 0 0 0 0 0 0

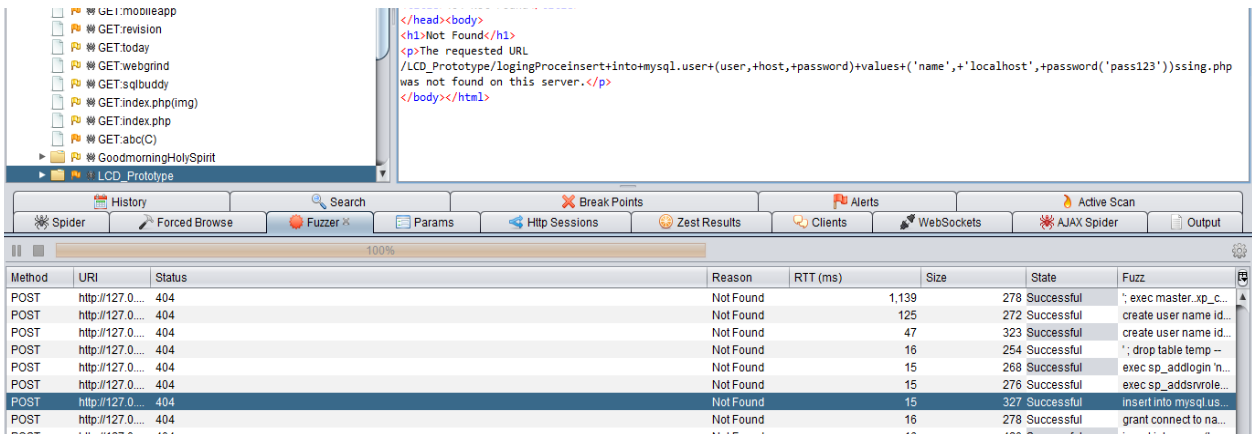
Figure 10: A cropped output of web application scanning to discover vulnerabilities.



lcdprototype (6)
- extension
- flock
- milk_production
- mydiary
- register1
- veterinary

unique_id	hashed_password	status	username
1	76b7f49ce0dcdd42748690dcf4f2059bec919bd1	Farmer	Alfred
2	36dbce4786d734d2d91184e1ea3caeae24af1cfd	Extension officer	Aweso
3	7fe54080e26dd169ccbffba947dbc5958e26ecea	Farmer	oshin
4	40bd001563085fc35165329ea1ff5c5ecbdbbeef	Farmer	larry
5	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	username
6	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	/WEB-INFweb.xml
7	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	/Windowssystem.ini
8	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	/etc/passwd
9	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	WEB-INFweb.xml
10	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	Windowssystem.ini
11	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	etc/passwd
12	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	c:WEB-INFweb.xml
13	3f6fb401e4d3e5c2c9043775361303e25ebc1da0	Farmer	c:Windowssystem.ini

Figure 11: A cropped screen output of registered users showing registration of malicious users.



```
</head><body>
<h1>Not Found</h1>
<p>The requested URL
/LCD_Prototype/logingProceinsert+into+mysql.user+(user,+host,+password)+values+('name',+'localhost',+password('pass123'))ssing.php
was not found on this server.</p>
</body></html>
```

Method	URI	Status	Reason	RTT (ms)	Size	State	Fuzz
POST	http://127.0....	404	Not Found	1,139	278	Successful	'; exec master..xp_c...
POST	http://127.0....	404	Not Found	125	272	Successful	create user name id...
POST	http://127.0....	404	Not Found	47	323	Successful	create user name id...
POST	http://127.0....	404	Not Found	16	254	Successful	'; drop table temp --
POST	http://127.0....	404	Not Found	15	268	Successful	exec sp_addlogin 'n...
POST	http://127.0....	404	Not Found	15	276	Successful	exec sp_addsrvrole...
POST	http://127.0....	404	Not Found	15	327	Successful	insert into mysql.us...
POST	http://127.0....	404	Not Found	16	278	Successful	grant connect to na...

Figure 12: A cropped fuzz result from ZAP showing login credentials embedded in SQL statement.

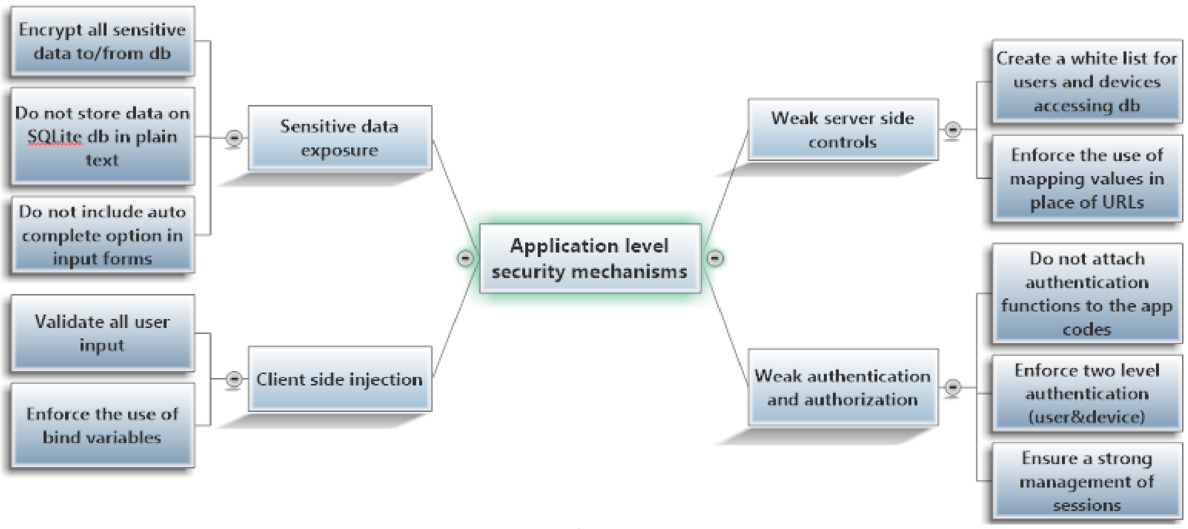
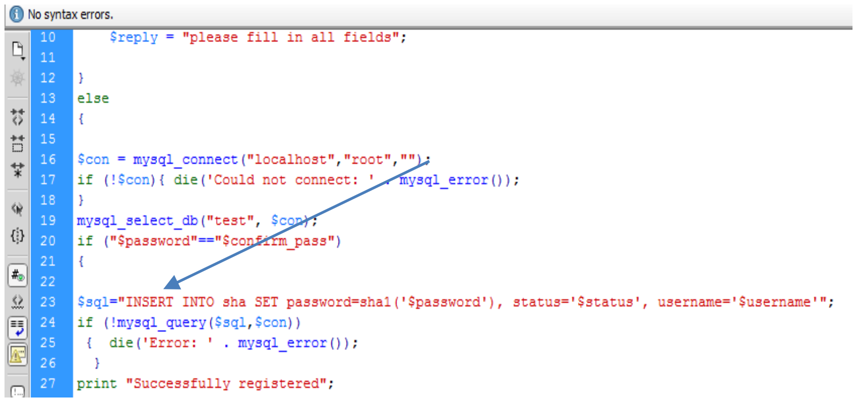

Application level security mechanisms
Sensitive data exposure
Encrypt all sensitive data to/from db
Do not store data on SQLite db in plain text
Do not include auto complete option in input forms
Client side injection
Validate all user input
Enforce the use of bind variables
Weak server side controls
Create a white list for users and devices accessing db
Enforce the use of mapping values in place of URLs
Weak authentication and authorization
Do not attach authentication functions to the app codes
Enforce two level authentication (user&device)
Ensure a strong management of sessions

Figure 13: Proposed security controls for converged web and mobile applications.


No syntax errors.

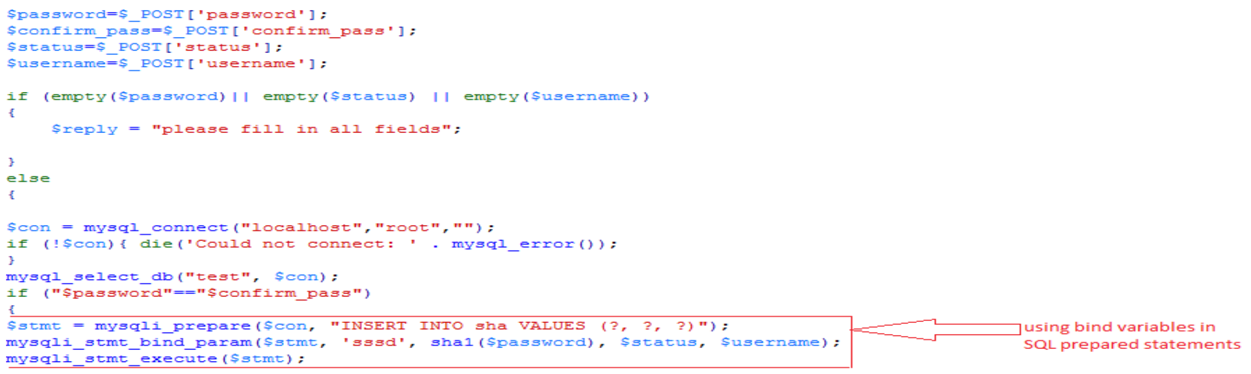
```
    $reply = "please fill in all fields";

}
else
{

$con = mysql_connect("localhost","root","");
if (!$con){ die('Could not connect: ' . mysql_error());
}
mysql_select_db("test", $con);
if ("$password"=="$confirm_pass")
{

$sql="INSERT INTO sha SET password=sha1('$password'), status='$status', username='$username'";
if (!mysql_query($sql,$con))
 {  die('Error: ' . mysql_error());
  }
print "Successfully registered";
```

Figure 14. Use of SHA-1 algorithm for password encryption in web application



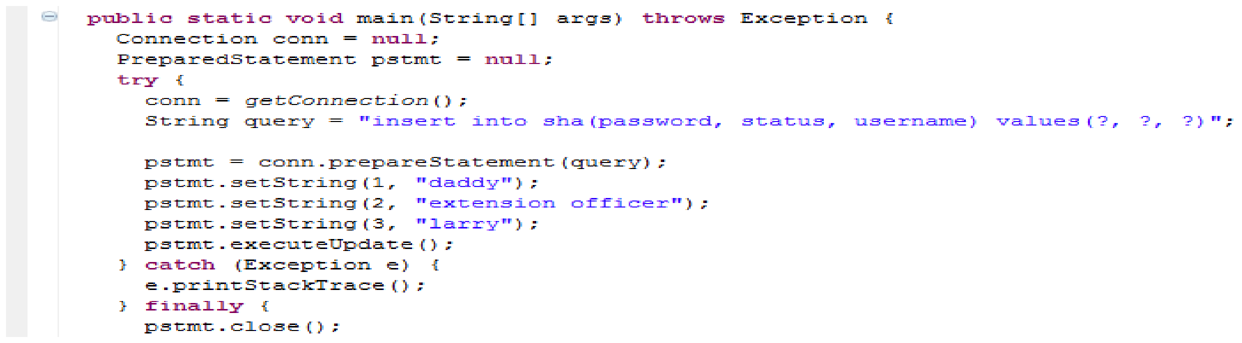
```
$password=$_POST['password'];
$confirm_pass=$_POST['confirm_pass'];
$status=$_POST['status'];
$username=$_POST['username'];

if (empty($password)|| empty($status) || empty($username))
{
    $reply = "please fill in all fields";

}
else
{

$con = mysql_connect("localhost","root","");
if (!$con){ die('Could not connect: ' . mysql_error());
}
mysql_select_db("test", $con);
if ("$password"=="$confirm_pass")
{
$stmt = mysqli_prepare($con, "INSERT INTO sha VALUES (?, ?, ?)");
mysqli_stmt_bind_param($stmt, 'sssd', sha1($password), $status, $username);
mysqli_stmt_execute($stmt);
```

Figure 15: The use of bind variables in PHP scripts to mitigate SQL injection.



```
public static void main(String[] args) throws Exception {
  Connection conn = null;
  PreparedStatement pstmt = null;
  try {
    conn = getConnection();
    String query = "insert into sha(password, status, username) values(?, ?, ?)";

    pstmt = conn.prepareStatement(query);
    pstmt.setString(1, "daddy");
    pstmt.setString(2, "extension officer");
    pstmt.setString(3, "larry");
    pstmt.executeUpdate();
  } catch (Exception e) {
    e.printStackTrace();
  } finally {
    pstmt.close();
```

Figure 16: The use of bind variables in Java scripts to mitigate SQL injection.

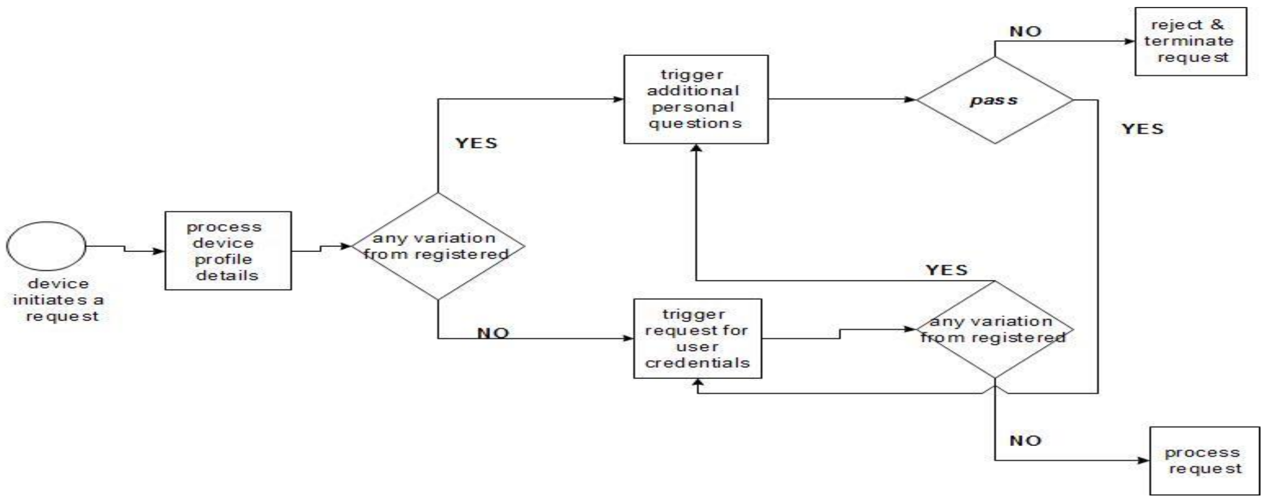

device initiates a request
process device profile details
any variation from registered
YES
NO
trigger additional personal questions
pass
NO
reject & terminate request
YES
trigger request for user credentials
any variation from registered
YES
NO
process request

Figure 17. An authentication approach involving device profile data and user credentials.

Email-address: nyambod@nm-aist.ac.tz, zaipuna.yonah@nm-aist.ac.tz, charles@udsm.ac.tz