International Journal of Mathematical Archive-2(11), 2011, Page: 2462-2469
Available online through www.ijma.info ISSN 2229 – 5046

SECURING SMARTPHONE APPS IN ONLINE ENVIRONMENT

[1]S. Santhipriya, [2]B. V. S. S. R. S. Sastry* and [3]K. Akshitha

[1]*Engineering College, Bhongir, Andhra Pradesh 508116 INDIA*
(Phone: 91-8985622122; E-mail: santhipriya.sunkara@gmail.com)

[2]*Engineering College, Bhongir, Andhra Pradesh 508116 INDIA*
(Phone: 91-9885775832; E-mail: sastry_38@yahoo.com)

[3]*Engineering College, Bhongir, Andhra Pradesh 508116 INDIA*
(Phone: 91-9959521914; E-mail: koluguri.akshitha@yahoo.com)

(Received on: 27-10-11; Accepted on: 10-11-11)

ABSTRACT

In a world where business is moving towards e- Commerce and happening over the Internet, B2B, B2C, and C2C applications have always been an area of major security concern due to the pitfalls of HTTP security and the number of integration points. All the threats that attack enterprise computer centers and personal computer systems are quickly encompassing mobile devices. In this paper, we provide various security issues and mapping of security with SDLC.

Index Terms: *Security, SDLC, Security threats, e- commerce.*

1. INTRODUCTION:

SECURITY is a problem not just for business over the web [1] it primarily covers applications that are hosted on Internet, Extranet, Intranet and Desktop applications.

There are a few standard techniques, like using the HTTPS protocol, RSA security, etc., that we can use. However, security is not just limited to encrypting data, providing a secure login process, or putting web applications behind a firewall. In addition to these, there are other security threats which need to be taken care of.

Before we start discussing further, we should understand what is SMARTPHONE?

By definition *A smartphone is a mobile phone offering advanced capabilities, often with PC-like functionality (PC-mobile handset convergence).*

Therefore the question is: is it really possible to develop a system which is foolproof and will remain so in the future? The answer is *IMPOSSIBLE*. Actually, a 100% secure system is unusable, and the requirement here is contradictory in nature. Therefore what should we do? What approach should we follow?

• Should we let the system be as it is and vulnerable.
• OR make our system at least secured against

known risks and threats and on regular basis keep monitoring the system and upgrading it to make it secured against newly indentified threats or risks. Undoubtedly, the second approach is better and more feasible. Security is a domain and there are many areas within the security domain itself and it is practically impossible to cover everything in just a few pages.[2][3] This paper identifies the security objectives, process, useful tips, and a set of strategies that could be used to make web applications more secured. The primary audiences for this paper are developers and architects; hence the objective of this paper is to provide a head start for the security process and techniques. Subsequently, professionals can later explore in more detail other approaches.

2. SECURITY OBJECTIVES:

Business applications contain lot of confidential data and processes, which include personal and confidential data provided by customers and business houses. Business applications perform numerous transactions 24*7 and any kind of security breach or breakdown of system consequent to security issues may directly or indirectly lead to:

***Corresponding author:** [2]*B. V. S. S. R. S. Sastry**, ***E-mail:** sastry_38@yahoo.com*

- Loss of business data.
- Loss of customer's personal data.
- Loss of financial data
- Loss of employee's data
- Loss of intellectual properties etc.
- Monetary losses.
- Loss in existing Business.
- Loss of new business opportunity
- Loss of credibility.
- Losing competitive edge over the competitor.

The success of e-commerce/m-commerce solely depends on support for secure financial transactions. Therefore the objectives of security should have EFFECTIVE security implementation for systems so that business can be

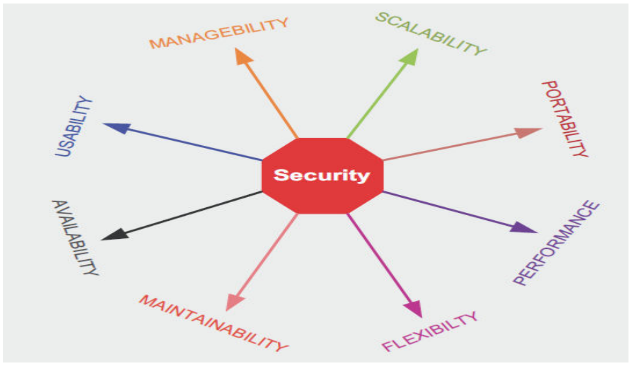

Fig.1: Security's relationship with other Architectural Parameters.

done more freely which directly or indirectly results in more business and improved credibility[1][3]. To achieve this in today's scenarios, the security should be a non negotiable NFR and mandatory to implement. Security Implementation results in an impact in the following terms:

- Project Timelines and Costs – due to extended scope, new process & tools, increase in team size and stakeholders.

- Architecture – security is inversely related to almost all the architectural parameters like scalability, availability, flexibility, portability, performance, maintainability, usability and manageability (See Figure 1).

3. SECURITY PROCESS:

Security is largely an afterthought, normally we think about the security in pre-deployment and post development stage [2][4]. This largely ignores the security related issue that may be the part of the developed application. Enterprise Security primarily covers:

Physical, Network, Information, Application. Security implementation requirements can-not be achieved just by making application, as it also requires security policies for:

- Client, server and network security.

- Physical security like installing tracking system, restricted access to server rooms, activity monitoring systems, locking devices, etc.

Therefore, this paper will be focusing on application security and not the network, physical or information security. For effective security implementation:

- We should treat security as the part of the application SDLC [5] [7] for successfully identifying and resolving all the security issues within an application.

- We have to identify Security objectives, develop security guidelines, define security architecture, identify all types of security threats, and perform security reviews and testing regularly.

- All the stakeholders[6] have to provide required support at relevant time. Any stakeholder who is responsible for building software which is capable for handling known threats is also known as Secure software lifecycle professional (SSLP).

Security is a continuous process (Figure 2). Just like other Processes it has a starting point and an end point, however it ends only when the system is moved out of business[8].

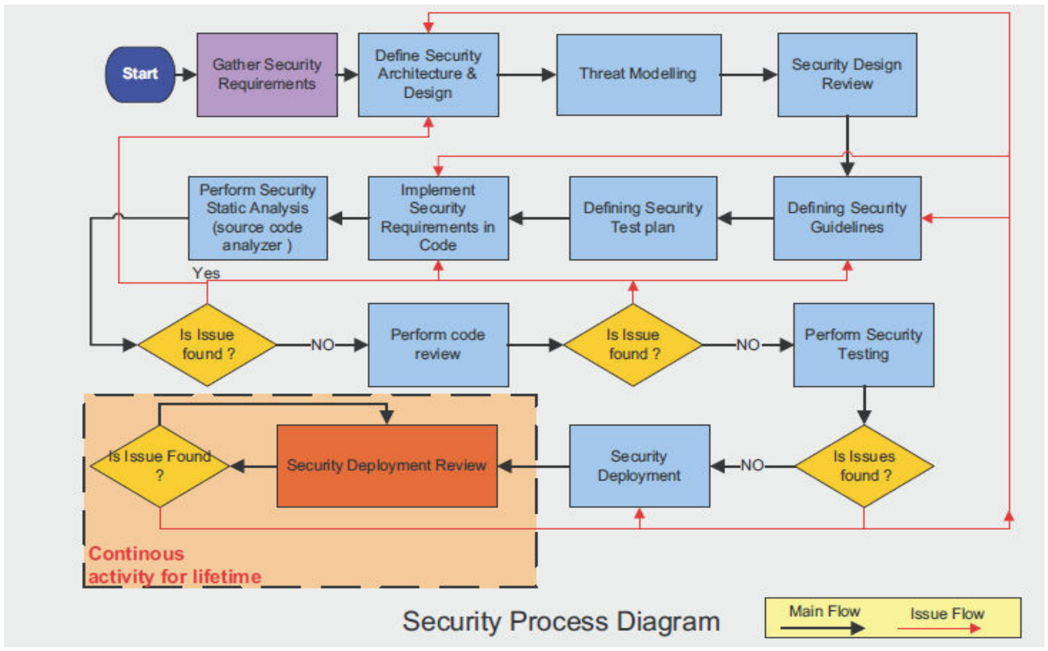
Start
Gather Security Requirements
Define Security Architecture & Design
Threat Modelling
Security Design Review
Perform Security Static Analysis (source code analyzer)
Implement Security Requirements in Code
Defining Security Test plan
Defining Security Guidelines
Yes
Is Issue found ?
NO
Perform code review
Is Issue found ?
NO
Perform Security Testing
Is Issue Found ?
Security Deployment Review
Security Deployment
NO
Is Issues found ?
Continous activity for lifetime
Security Process Diagram
Main Flow
Issue Flow

Fig. 2: Security Process

4. RISK MANAGEMENT VERSUS SECURITY:

Usually people interchangeably use risk management and security words and think that they are same. Risk management deals with uncertain events and the modality to control them for smooth execution of the project [8][9]. Whereas security is an aspect where we make the system (which is result of Project mgmt) secured.

Few points worth mentioning about risk management and security are given in following paragraph.

• Risk Management is a knowledge area which describes the process for risk identification, planning, monitoring and controlling whereas security is a domain and is a non functional requirement of the project.

• Risk Management is for handling all types of Project Risk, which can be categorized as Organization risks (HR), Project Management, Risk (requirement, schedule, time, cost and quality), external risk (Political, weather) and technical risk (which covers technological change risk, unrealistic requirements). During RISK Mgmt we also plan and control for what happens if security requirements are not met[11]. Security implementation denotes plugging all the open holes in the system either because of technology or implementation or standards.

• Risk Management process spans across Planning, Monitoring and controlling process excluding execution. Whereas, the security execution phase is one of the required phases as there are quite a lot development related activities. In a nutshell, Risk Management is a process of planning and controlling uncertain events or conditions that can impact project and handling for smooth execution, even after the project goes live[12] whereas security is an aspect where we make applications secured. Risk Mgmt plans and controls the security implementation as one of the activity or tasks [13]. Figure 3 provides the clear Association of security across SDLC phases and span of security implementation across application layers.

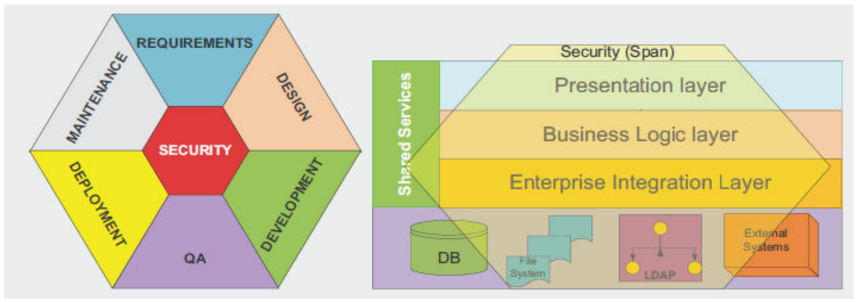

Fig. 3: Association of security across SDLC phases.

5. SECURITY TIPS:

List of security tips described in this section should be followed to make secure web application. Although this list is not comprehensive enough, but definitely a good to start [11][12][13].

A. Web Application Security:

• Avoid or minimize the use of hidden form fields especially for storing sensitive data like encrypted user- id or passwords, roles, payment information etc.

• Do not use user input as it is to construct directory or file names and SQL queries. Also avoid passing this information as it is over the wire.
• Always use fully qualified absolute path and filename instead of relative paths. Given below is one of the example Wrong method

```
../../../<filename>
```
Suggested method
```
/examples/code/<filename>
```
• Make sure the execution of files or programs which have the capability to change the permission or mode for other files should be restricted.

• Do server side validations and do not rely only on client side validation as it is very easy to bypass client side validations. Make sure we remove unwanted characters, invisible characters and HTML tags from user input and if required throw back error(s) to the user.

• It is advisable to separate out the presentation layer from business layer as this will provide greater flexibility and allows better deployment of security.

• Do not write user credentials in code.

• Use the proper commenting mechanism. For example JSP developers put comment in HTML format rather than JSP format.

• Only validated request for URL redirect or forward should be allowed.

• Proper Exception handling needs to be implemented.

• It is advisable not to display application exceptions or error as it is. They should be wrapped within some user friendly messages.

• Make sure that exceptions and errors are properly handled, caught (for e.g. Nullpointer, Array Out Of Bound, Divide By Error, etc.) and thrown or re-thrown as per the requirements. Again given below is a wrong example:

```
try {
stmt1
stmt2
} catch (Exception t) {
}
Stmt4
```
• Secure Communication

• Make sure that the important and sensitive data is always in encrypted form. For example, make use of SSL or IPSEC wherever necessary. Also, it not advisable to transfer sensitive data using GET Protocol.

• Avoid using unsecured protocols, if not possible to avoid, provide secured communication.

• Don’t used HTTP header information for making security decisions.

• Restricted Access

• Make sure that only required URL and Services are exposed with proper level of security.

• Ensure to provide only authorized access to the system and application resources.

• Avoid putting everything in the web directory. By this we, we restrict visitors to access our web resources (like data, files, configurations setting, etc) directly from browsers.

• Remove sharing of files and folders from production machines.

• Only required ports are opened. For example, if FTP is not required on the Web Server, it should be closed.

• Make sure that directory listing is off. For example, if somebody visits *http://www.myexample.com/examples/*, in case default page is not available then directory listing should not be displayed.

• Do not perform encryption logic on client side or in executables code (like applets, ActiveX controls, etc) which gets downloaded at client side as it is quite possible that logic will get exposed to the outer world.
• Avoid storing vital sensitive data like passwords, roles etc in cookies or in file which are accessible to use for e.g. web directory.

• Use POST instead of GET method and if possible before processing the request, always check at server side that if request method is not as expected then we should throw error for e.g. we created a form where a POST method request is expected, but we are getting a GET request then we should throw an error.

• Make sure we should not have *MIXED-CONTENT* in HTML page i.e. we should not access any internal resources with HTTP protocol if page is accessed through HTTPS protocol as it will show Mixed Content Popup Window and also allows Active hackers a chance. Given below is an example:

Wrong method
<… src=http://www……>
Suggested method
<…. src=//www…>.
With this what will happen is that the protocol is automatically attached to the URL based on the page protocol. i.e., if page is accessed with HTTP protocol then internal resources will be accesed with HTTP protocol, similarly if page is accessed with HTTPS protocol then internal resources will be accessed with HTTPS protocol.

• Proper Session Management Policies

• System should invalidate user session at logout and when time expires

• We have to ensure that generate new session ID when users transitions from unauthenticated to authenticated.

• Session ID should have enough randomness for a period and it's session life should be limited

• We should encrypt the cookie data and also unauthorized access to session data be avoided.

• Appropriate Logging and Auditing should be implemented and it should be secured. Also, make sure that sensitive data is not logged. Some of the major things that we should log are:

• Startup and shutdown

• Unsuccessful access attempt for resources (for e.g. Denial of access resulting from excessive number of login attempts)

• Unauthorised access attempts.

• User authentication.

• User role assignments.

• Permissions granted to users or groups.

• Actions by trusted users.

• Configuration changes.

• Modifications to data (Sensitive, Public, Classified)

Also, ensure we log the date and time of each event, as well as the success or failure of major events.

• Avoid using database administrator credentials for connecting a Web application to database. Configure specific users and roles in the database with required privileges and use them from web application to connect to database.

• Make sure we encrypt all sensitive data in all types of configuration files like xml, properties, csv files. Also wehave to ensure that the files are not in web directory.

• Make sure, if the application uses third party products API's, it should also be scanned from security perspective.

• Make sure only required and tested code goes in production. For example, we should not push dynamic pages which were developed during development time and subsequently they were no longer in use.

• Make sure our system should be robust enough against unwanted bots and crawlers. For this we should use *robot.txt*, captcha, Meta tags appropriately

• **Avoiding XSS Attacks** – stands for Cross Site Scripting and it occurs when a web application gathers malicious data from a user. Here user does code injection into a vulnerable application in order to gather data from them. It is a real threat for authentication process, stealing user information, changes setting and cookie values, etc. There are three distinct types of XSS vulnerability (type 0, type 1 and type 2).

• **Avoiding CSRF Attacks** – stands for Cross Site Request Forgery. It compels a logged-on user's browser to send a request to a vulnerable web application, which then performs the chosen action assuming that the request is coming from a trusted and authenticated user. It is also known as one click attack or session riding.

• **Avoiding Injection flaws** – It allow attackers to relay malicious code through a web application to another system by sending data to an interpreter as part of a command or query. There are many types of injection flows like SQL, OS commands, LDAP, XPath, XSLT, etc.

• **Support for P3P Headers** – *stands for Platform for Privacy Preferences Project*. It enables Web sites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agent (a.k.a. browser). Policy is delivered to the web page allowing the P3P enabled browser to make decisions by comparing this policy with the user's stored preferences before the page is displayed. The privacy policy can be retrieved as an XML file or can be included, in compact form, in the HTTP header. Compact policies are summarized P3P policies delivered in the HTTP header that provide hints to user agents to enable the user agent to make quick, synchronous decisions about applying policy. We should follow government, industry and domain guidelines for e.g.

• PCI – credit card related guidelines.

• HIPPA – health care domain related guidelines.

• GLBA – Financial Institution guidelines OWASP (The Open Web Application Security Project) had releases their version of top security threats in web application.

B. Database Security:

As no database product comes configured securely out of the box it is necessary to follow these steps to prevent attacks from exploiting known vulnerabilities. The checklist given in the subsequent paragraph is not comprehensive; the actual steps for hardening a specific database platform may be more in-depth. Please check the vendor specific security checklist for more details and product specific security tips. These steps should be followed to secure a typical database server, but may not be appropriate in all cases. It is the responsibility of the DBA associated with each platform to ensure they understand the detailed tasks.

• Make sure we do not save sensitive data like Passwords, credit cards numbers in clear text format. All sensitive data should be encrypted.

• If possible, use the latest generation of database server.

• Install the latest vendor-provided patches for the database. Be sure to include patches for database support software that is not directly bundled with the database.

• Every server should be configured to only allow trusted IP addresses and only those ports which are required should be opened.

• Remove sample databases and database users.

• Create alternative administrative users for each DBA, rather than allowing multiple individual users to regularly use the default administrative account.

• Configure specific users and roles in the database only with the privileges required and ensure that access to the database is limited to the minimal access necessary (at the level of Database, tables, rows and columns). For example, reporting applications that just require read-only access should be appropriately limited.

• It is advisable that the database should require authentication before returning any type of data.

• Remove unwanted database stored procedures, triggers.
• Appropriate guidelines (HIPAA, Privacy Act, financial, personnel, etc.) must be used for safeguarding the sensitive data.

• Whereever possible isolates sensitive databases to their own servers. Databases containing *Personally Identifiable Information* (PII), or otherwise sensitive data should be protected from the Internet by a network firewall.

• Administrative/DBA access should be limited to fewer individuals as far as possible.

• The data base for the web application should not be directly accessible from the public network from where external user customer traffic arrives on.

• Use complex names for database users. Use complex passwords for these users.

• Use IPSec or SSL to protect access to databases from other network servers.

• Auditing and logging is implemented, working and also access to log file is secured. Also make sure that sensitive information like passwords are not logged.

6. CONCLUSION:

Security is one of the greatest concerns as today most of the applications that are build, have numerous integration points. Now a day's security capability of delivered application are one of the major criteria to evaluate deliverables as security is mapped directly with the business.

7. ACKNOWLEDGEMENT:

The author wishes to thank his colleagues S. Santhi Priya and K. Akshitha for their support and valuable contribution for this paper.

8. REFERENCES:

[1] http://www.w3.org/security/Faq/www-security-faq.html

[2] http://advosys.ca/papers/printable/web-security.html

[3] http://www.cert.org/advisories/CA-2000-02.html

[4] http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/wcag-guidelines-20.shtml

[5] http://msdn2.microsoft.com/en-us/library/aa302332.aspx

[6] http://msdn.mirosot.com/en-us/library/aa302433.aspx

[7] http://technet.microsoft.com/en-us/library/cc512638.aspx

[8] http://msdn.microst.com/en-us/library/aa480484.aspx

[9] http://msdn.microsoft.com/en-us/library/ms998258.aspx

[10] http://www.fortify.com/servlet/downloads/public/fortify

[11] http://java.sun.com/security/seccodeguide.html

[12] http://advosys.ca/papers/printable/web-security.pdf

[13] https://www.pcisecuritystandards.org

AUTHORS INSTITUTIONAL AFFILIATIONS AND ADDRESSES:

S. Santhi Priya has been graduated with B. Tech in 2002 from Nagarjuna University, Guntur, Andhra Pradesh, India. She has around 9 years of teaching experience. She is currently working with Aurora's Engineering College, Bhongir. Contact her at **santhipriya.sunkara@gmail.com.**

B. V. S. S. R. S. Sastry has been graduated with B. Tech in 2009 from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. He is currently pursuing M. Tech from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. Contact him at **sastry_38@yahoo.com.**

K. Akshitha has been graduated with B. Tech in 2009 from Royal Institute of Technology and Science, Chevella, R. R.Dist, Andhra Pradesh, India. She is currently pursuing M. Tech from Aurora's Engineering College, Bhongir, Andhra Pradesh, India. Contact her at **Koluguri.87@gmail.com.**

****