ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Security Requirements Development Framework (^{S}RDF)

Rajendra Kumar* and Mustafa K.
Department of Computer Science
Jamia Millia Islamia (Central University),
New Delhi-110 025, India
verma_rajk@yahoo.com, kmfarooki@yahoo.com

Abstract: Any information system becomes successful for the business organizations when its diverse software modules work in a desired manner. In today's vulnerable world, information systems entail more attention towards security. But, unfortunately, building secure software still remains an issue. Several advances have recently been made on definition of processes for secure software development, but it still remains in infancy. Earlier, security used to be an afterthought, but now-a-days it is widely accepted to be an integral part of each of the phases of SDLC. A prescriptive framework, ^{S}RDF has been proposed on the basis of research findings and industry best practices. It is to accomplish the present need of enriching SRS with requirements pertaining to security needs of particular software as a product. This may provide a roadmap to incorporate the security through SRs in the inception itself.

Keywords: Security Requirements, Web Application Security, Security in Requirements Phase, Requirement Framework

I. INTRODUCTION

The exponential growth of the Web has made an immense impact on many areas of routine life. Web-based applications have forced many radical paradigm shifts in the various sections of modern society like Entertainment, Government, Commerce and Education. Web applications are emerging fast as '*the common interface*' to most of the technological applications. Regular works like banking, on-line reservation etc are the common things that are done by people on periodical basis. The sprawling impact of the Web and the Internet, combined with the rapid ad-hoc approach with which most web engineering projects are realized, have concerns about the stability and success of Web-based application development due to security. Hackers on the Internet have evolved from fame-hungry sabotage to fraud the profitable organized data and identity theft, which in turn compromises security. One of the prime reason for compromising security is the bad software and this happens due to vulnerabilities occurring in the software.

Vulnerabilities are continuously increasing; hence software is under attack at every time [1]. As this evolution continues, it is important for business leaders to consider the security of their web applications as a vital performance indicator of the success of their business. Hence, it becomes necessary to build secure web applications and ensure the development of the same.

Any information system becomes successful for the business organizations when its various software modules work in a desired way. In today's vulnerable world, information systems require more attention towards security. But, unfortunately, building secure software still remains an issue. Several advances have recently been made on definition of processes for secure software development, but it still remains in infancy. Earlier, security used to be an afterthought, but now-a-days it is widely accepted to be an integral part of each of the phases of SDLC [2].

To be really effective, security must be integrated into the SDLC right from system inception. Early integration of security in the SDLC enables agencies to maximize return on investment in their security programs, through early identification and mitigation of security vulnerabilities resulting with much lower cost of security control implementation and vulnerability mitigation. Integration enables security to be planned, acquired, built in, and deployed as an integral part of a project or system. It plays a significant role in measuring and enforcing security requirements throughout the phases of the life cycle. Life cycle management helps document security-relevant decisions and provides assurance to management that security requirements were well attended in all phases. Implementing information security early in the project allows the requirements to mature as needed and in an integrated and cost-effective manner.

Requirements engineering is the branch of software engineering concerned with the real-world goals for the functions and constraints on software systems. Requirement engineering process includes obtaining, modeling, analyzing and extending the requirements [3]. There are some inherent problems in the process. Requirements of different types of system users including customers, developers and system owners vary from one to another. They have different or even contradictory goals. Those goals, though unavoidable, might not be easily expressible. Perhaps, satisfaction of some requirements would result into other uncontrollable limitations [3]. These problems in engineering security requirements must draw more attention due to lack of awareness, experience, expertise, techniques and tools.

More accurate and consistent security sensitive requirements could be a solution towards more secure software. Therefore, a methodology for developing security sensitive requirements for web applications may be a solution of such type of problems. In order to curtain the high impacts of vulnerabilities, it becomes essential to develop security requirements early in the development on a detailed level. Current approaches are well in place but the area still warrants further investigations. A prescriptive framework, ^{S}RDF has been proposed to accomplish the present need. Due to non-availability of any standard framework, completely devoted to web application, ^{S}RDF may prove handy for developing the security requirements.

This proposed framework may be used by the requirement engineers as well as the research community for further development of security requirements in a prescriptive manner.

The rest of the paper is organized as follows: Section II presents a brief discussion on the ^{S}RDF Process, whereas in Section III, Implementation Mechanism is discussed. In Section IV, Tryout Results on the security requirements specification (SRS) of a live project are discussed. Section V presents Conclusion and Future Research Directions in the area.

II. ^{S}RDF PROCESS

It is an emergent need to develop a comprehensive framework to develop the security requirements, based on the vulnerabilities of web applications. Hence, a prescriptive ^{S}RDF is hereby proposed to develop the security requirements for a web application. By adapting ^{S}RDF, requirement engineers may assess the risk aspects of vulnerabilities through ^{S}RDF in a right perspective, which leads to the development of security requirements for web applications. Moreover, ^{S}RDF is an iterative process comprising of a number of stages to reach the ultimate objective.

The framework is useful along with functional requirement phase. By following strictly the steps prescribed at various stages of ^{S}RDF, requirement engineers would be able to assess the risk aspects of the vulnerabilities, which in turn help to develop realistic and meaningful security requirements. It may be used for every web application by requirement engineers' inputs of domain. The thrust largely rests upon security vulnerabilities, their severity and risk associated with web applications. For each of the security vulnerabilities, various attributes/properties are identified to determine the severity and corresponding risk. A mathematical formulation is proposed for the calculation of the severity and risk. Then the tolerance level of the risk is also assessed, and accordingly the practical and significant security requirements can be developed in a technical manner. The high level architecture of ^{S}RDF is given in the Fig. 1:

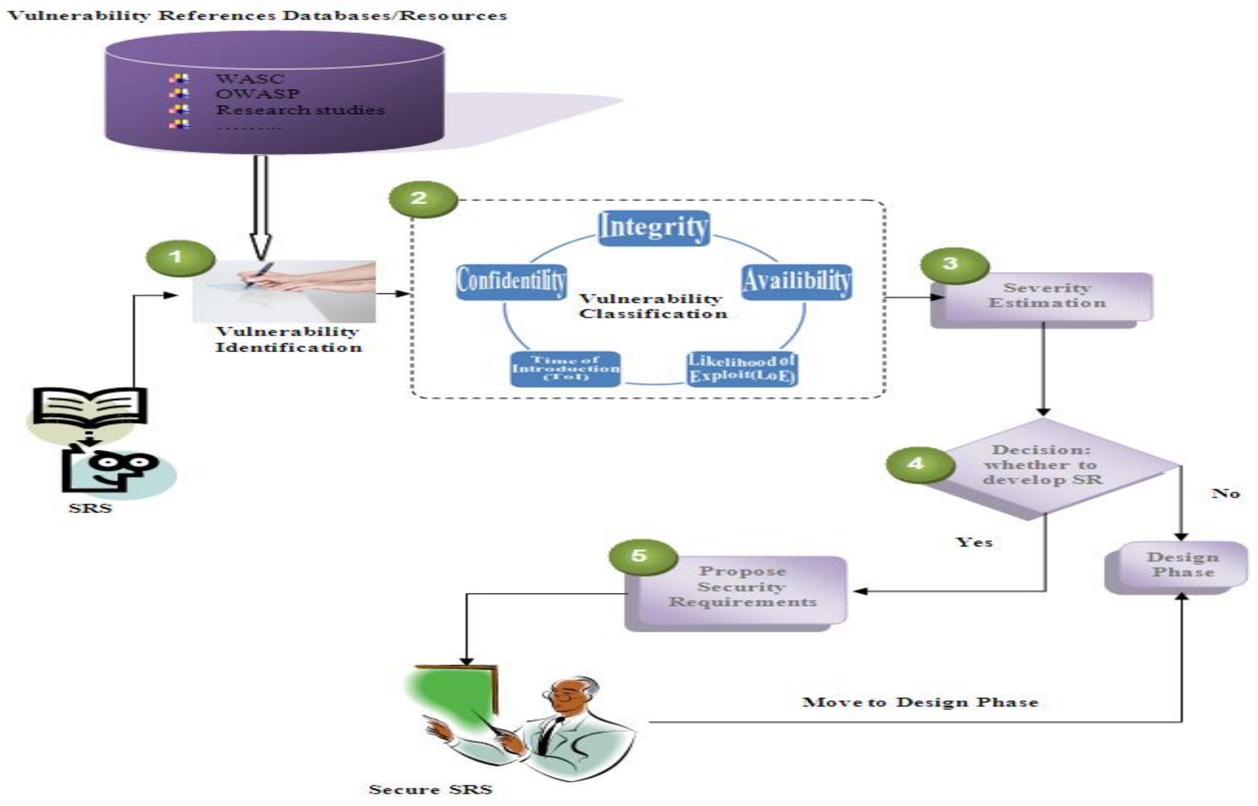
Vulnerability References Databases/Resources
WASC
OWASP
Research studies
....
1
Vulnerability Identification
SRS
2
Integrity
Confidentility
Availibility
Vulnerability Classification
Time of Introduction (ToI)
Likelihood of Exploit(LoE)
3
Severity Estimation
4
Decision: whether to develop SR
No
Yes
Design Phase
5
Propose Security Requirements
Move to Design Phase
Secure SRS

Figure 1: Architecture of the ^{S}RDF

A. *Vulnerability Identification:*

Vulnerability is referred to as a set of conditions that leads or may lead to an implicit or explicit failure of primarily confidentiality, integrity, or availability of an information system [4]. According to CERT/CC, more than 90% of vulnerabilities leak out during system development and they are the result of ignoring known vulnerabilities found in other systems [5]. The same report states that ten vulnerabilities known worldwide are responsible for 75% of security breaches in today's software applications. In the other words, if the developers get to know these ten vulnerabilities, about majority of them can be avoided. Keeping this fact in view, most important and prevalent security vulnerabilities for web applications were identified and given in Table 1.

B. *Vulnerability Classification:*

The second step in ^{S}RDF is vulnerability classification. It refers to the classification based on several relevant attributes. Some frameworks describe vulnerabilities by classifying the techniques used to exploit them, others characterize vulnerabilities in terms of the software and hardware components and interfaces that make up the vulnerability, and also others classify vulnerabilities by their nature. For our research work, we classified vulnerabilities on the basis of CIA as Security vulnerabilities and exposures may be exploited to compromise the confidentiality, integrity and availability (CIA) of computing systems [6].

To develop a complete set of security requirement, it is necessary to know that in which phase vulnerability can occur. Therefore, the time of introduction in SDLC phase of each vulnerability is also recognized. It is advised that more prevalent vulnerabilities need to be mitigated first, likelihood of exploit attribute is considered as an add-on to this notion. For the classification of vulnerabilities, ^{S}RDF proposes the following attributes:

a. ***Confidentiality*****:** It refers to the prevention of unauthorized disclosure of information.
b. ***Integrity*****:** It refers to the prevention of unauthorized modification of information.
c. ***Availability*****:** It refers to the prevention withholding of information.
d. ***Likelihood of Exploit*****:** Likelihood of vulnerability/threat occurring is the estimation of the probability that the threat will succeed in achieving an undesirable event. The presence, tenacity and strengths of vulnerability/threats, as well as the effectiveness of safeguards must be considered while assessing the likelihood of the vulnerability/threat occurring.
e. ***Time of Introduction*****:** To categorize vulnerabilities according to when they were introduced in the software lifecycle. For example, a vulnerability may be due to a bad algorithm being chosen during design phases. Another may be due to a bad implementation of a correctly chosen algorithm, and therefore the vulnerability was introduced at some point during the implementation phases of the program. So, 5 classes have been proposed: requirement and analysis, design, implementation, deployment, and maintenance [7].

Based on these aforementioned attributes the classification of the most important and prevalent twenty-seven security vulnerabilities has been accomplished and shown in Table 1:

Table 1: Vulnerabilities Classification

S. No.	Name of the Vulnerability	Confidentiality	Integrity	Availability	ToI	LoE
1.	Insufficient Authentication	H	M	L	R, D, I	H
2.	Insufficient Authorization	H	H	H	R, D, I, O	H
3.	Integer Overflows	L	H	H	I	M
4.	Insufficient Transport Layer Protection	H	H	L	R, D, O	H
5.	Remote File Inclusion	H	L	L	R, D, I	L
6.	Format String	H	H	H	I	H
7.	Buffer Overflow	H	H	H	R, D, I, O	H
8.	Cross-site Scripting	H	H	H	R,D,I	H
9.	Cross-site Request Forgery	H	H	L	R, D	H
10.	Denial of Service		H	H	R,D, I, O	M
11.	Brute Force	H	H	H	R,D,I	H
12.	Information Leakage	H	M	L	R,D,I	H
13.	Server Misconfiguration	H	L	H	R, D	L
14.	Application Misconfiguration	H	L	M	R, D	M
15.	Directory Indexing	H	L	L	R, D	L
16.	Improper Filesystem Permissions	H	M	L	R,D,I	H
17.	Credential/Session Prediction	H	H	L	R,D,I	H
18.	SQL Injection	H	H	L	R,D,I, O	H
19.	Improper Input Handling	H	H	H	R,D,I	H
20.	Insufficient Anti-Automation	H	L	L	R,D	H
21.	Improper Output Handling	H	H	L	R,D,I, O	H
22.	OS Commanding	H	H	H	R,D,I	H
23.	Path Traversal	H	H	H	R,D,I	H
24.	Predictable Resource Location	H	L	H	R,D,I, O	H
25.	Session Fixation	H	H	L	R, D, I	M
26.	Insufficient Session Expiration	H	H	L	R, D, I	M
27.	Insufficient Password Recovery	H	H	L	R,D	H

C. Severity Estimation:

The best way to avoid or prevent security incidents is to establish an ongoing vulnerabilities' severity assessment process that must continuously identify the critical ones and mitigate the same from the beginning and throughout their lifecycle i. e. from development through production. In our approach, the researcher recommends to estimate severity of a vulnerability. For the estimation of this severity, the following variables have been identified:

a. CIA: Based on the data compiled from various vulnerability databases, it is observed that successful exploitation of a vulnerability may have high, medium, low impact on CIA. But there exists some vulnerabilities, whose impact in terms of CIA is not known. In these cases, impact may be taken 'high' until the correct values are discovered by the researchers. Based on these observations, a CIA metric is recommended as follows:

Vulnerability Attribute	Rating	Rating value
Confidentiality	Low(L)	3
	Medium (M)	5
	High (H)	7
Integrity	Low(L)	3
	Medium (M)	5
	High (H)	7
Availability	Low(L)	3
	Medium (M)	5
	High (H)	7

b. ***LoE:*** Likelihood of Exploit (LoE) is another important variable, identified to calculate the severity of a vulnerability.

Vulnerability attribute	**Rating**	**Rating value**
Likelihood of Exploit	Low (L) Medium (M) High (H)	3 5 7

c. ***TOI Metric:*** A vulnerability can be introduced during any phase in the development life cycle. The logic behind taking this variable into the severity metric is quite simple. During the assessment of any SRS, for any vulnerability V_1, if it is taken care properly at very initial stage i. e. during the requirements phase, its impact will be low, whereas for design phase, it will be medium. Moreover, if proper mitigation mechanisms are not in place and the vulnerability comes in the implementation phase, obviously, its impact will be high. Based on these assumptions, a 'time of introduction' variable is established and assigned the rating according to the introduction of vulnerability in a SDLC phase as follows:

Vulnerability attribute	**Rating**	**Rating value**
Time of Introduction	Requirement (R)- Low (L) Design (D)- Medium (M) Implementation (I)- High (H)	3 5 7

After determining the variables, which are playing a major role in severity determination, it becomes important to impose some statistical tool based on the requirements. But before this, first of all some relationship among these variables must be identified. From the available research studies, it is established that:

Severity $\propto$ C'
$\propto$ I'
$\propto$ A'
$\propto$ LoE

Where, C'= compromise of C
I' = compromise of I
A'= compromise of A

For the given conditions, the most suitable statistical tool for projection that may be used is multiple linear regression. This is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of Multiple Linear Regression (MLR) is to model the relationship between the explanatory and response variables.

In this, there are n explanatory variables, and the relationship between the dependent variable and the explanatory variables is represented by the following equation:

$y = b + m_1x_1 + m_2x_2 + m_3x_3 + \ldots + m_nx_n$

where, b is the constant term, and m_1 to m_n are the coefficients relating the *n* explanatory variables ($x_1, x_2, x_3, \ldots x_n$) to the variables of interest. Based on the multiple linear regression model, the following equation has been formulated to analyze the severity of a vulnerability:

Vulnerability Severity = b + (m1 * C) + (m2 * I) + (m3 *A) + (m4 * R_{phase}) + (m5 * D_{phase}) + (m6 * I_{phase}) + (m7 * LoE) …………..(1)

In MLR, we must have the values for dependent as well as independent variables. In the present scenario, the values for all the independent variables are taken from the available vulnerabilities databases, whereas for dependent variable, namely severity, an experts' feedback is taken. In this, a complete exercise has been accomplished through a questionnaire in which every expert is asked to give a severity rating for each of the vulnerability with respect to high/medium/low. For our calculation, we analyzed these qualitative terms as given in the earlier section; high as 7, medium as 5, and low as 3. After receiving the feedback from various experts, the average is computed of the severity level for each of the vulnerability.

D. Vulnerability Prioritization:

The issue of vulnerability prioritization has been actively discussed in the literature and the need for vulnerability prioritization in organizations is widely recognized [8][9][10][11]. This work uses the notion that organizations should prioritize their remediation efforts based on the value of their assets and the severity of the vulnerability [9]. Empirical research has also shown that the actual impact of security incidents varies significantly among different types of organizations, businesses and users [12][13]. Since different organizations perceive the severity of a particular vulnerability differently; they also prioritize its mitigation differently [10]. Based on the severity estimation discussed in subsection C, vulnerabilities can be prioritized as described as follows:

a. ***High:*** This type of vulnerabilities should be addressed as quickly as possible.
b. ***Medium:*** This type of vulnerabilities should be addressed, but only after High level weaknesses have been addressed.
c. ***Low:*** It is not urgent to address the vulnerability, or it is not important at all.

A tabular presentation of the same can be given as follows:

Vulnerability Severity Score	**Priority**
≤ 7	High
≤ 5	Medium
≤ 3	Low

After assessing the severity level, need of the application for the security perspective must be reexamined. For example, if a vulnerability comes under the umbrella of 'high', but the application whose SRS is under examination, does not require security up to an high extent, mitigation of the vulnerability may be avoided.

E. Proposal for Security Requirements:

Based on the tolerance level defined in the earlier subsection, security requirements may be proposed for the mitigation of each vulnerability, depending upon the need as well as nature of the web application/project. These may be inclusive and not mutually exclusive of other recommendations. Here, we are providing a sample of *Session Management*. On the basis of this representation, the security requirements may be given for the rest of the vulnerabilities.

Session management is required to analyze, how the user's session is maintained and managed during their interaction with the application. Authentication is only a small part of managing a user's access to data. Critical to the process and often overlooked is the area of session management. Insecure session management can negate the security implemented by both authentication and access control (authorization) mechanisms. Due to the limitations of HTTP, all interactive applications need to connect the user's stateless HTTP session with their stateful application session. Typically, this is achieved through a Session ID sent to and associated with the user during the login process, often as a cookie. Having authenticated, this Session ID is the only token that the application checks when determining what data the user may access, and what privilege they have. As such, it is vital that the Session ID is treated with at least the same high level of security as the user's credentials. Compromise of the Session ID will allow an attacker to assume the identity of the user. To strengthen session management, Web application designers and developers should take into account following recommendations [14][15][16][17]. Table 2, representing a set of security requirements proposed, which may be incorporated for the mitigation of session management related vulnerabilities:

Table 2: Session Management Security Requirements

SR 1:	Properly manage HTTPS sections, especially if other portions of a Web application can be visited with HTTP.
SR 2:	Use adequate directives for caching and cookie transmission.
SR 3:	Protect Web applications against XSS attacks.
SR 4:	Design thoroughly session termination mechanisms.
SR 5:	Reduce the possibility of having multiple tokens for the same user at the same time or of having static user tokens [49].
SR 6:	Create securely a unique session identifier after the individual is successfully authenticated.
SR 7:	Associate the session identifier viz. session token, key/string in a strong manner with the session such that the system resists attacks against the session management function.
SR 8:	Ascertain that session identifiers are not containing sensitive information and also ensure that it is not predictable, readily reverse engineered, or susceptible to a brute-force attack.
SR 9:	Verify that the session identifiers are assigned from a sufficiently large key space.
SR 10:	Pass the session identifier to/from the user/client computer in a manner that will not result in inadvertent disclosure of the value/s in use.
SR 11:	Verify that the integrity and authenticity of a session identifier received back through internet is secure. /*an approved hash function must be used*/
SR 12:	Keep the session information saved on the user/clients/computer to the minimum extent.
SR 13:	Ensure that the application is not leaving behind any sensitive and /or personnel data on the user/clients computer namely in cache or cookie upon termination of a session.
SR 14:	Force the user /clients browser to discard web pages from the cache.
SR 15:	Retain the session information by the application server and delete upon the termination of the session.
SR 16:	Provide a logout facility to deactivate the session.
SR 17:	Implement the standard mechanisms to detect brute force attacks.
SR 18:	Display a reminder on the importance of prominently login out if, access is no longer required.
SR 19:	Ascertain that the sessions must be subjected to an inactivity time out to reduce the chance of an abundant active session being exploited.
SR 20:	Use shorter inactivity time outs depending on the risk involved and the nature of the transactions.

F. Structured Representation Model of Security Requirements:

Security requirements (SRs) presented in the aforementioned tables has been placed in the order of implementation for the mitigation of vulnerabilities belonging to each individual vulnerability family. A structured representation model for the session management is given in following Figure 2:

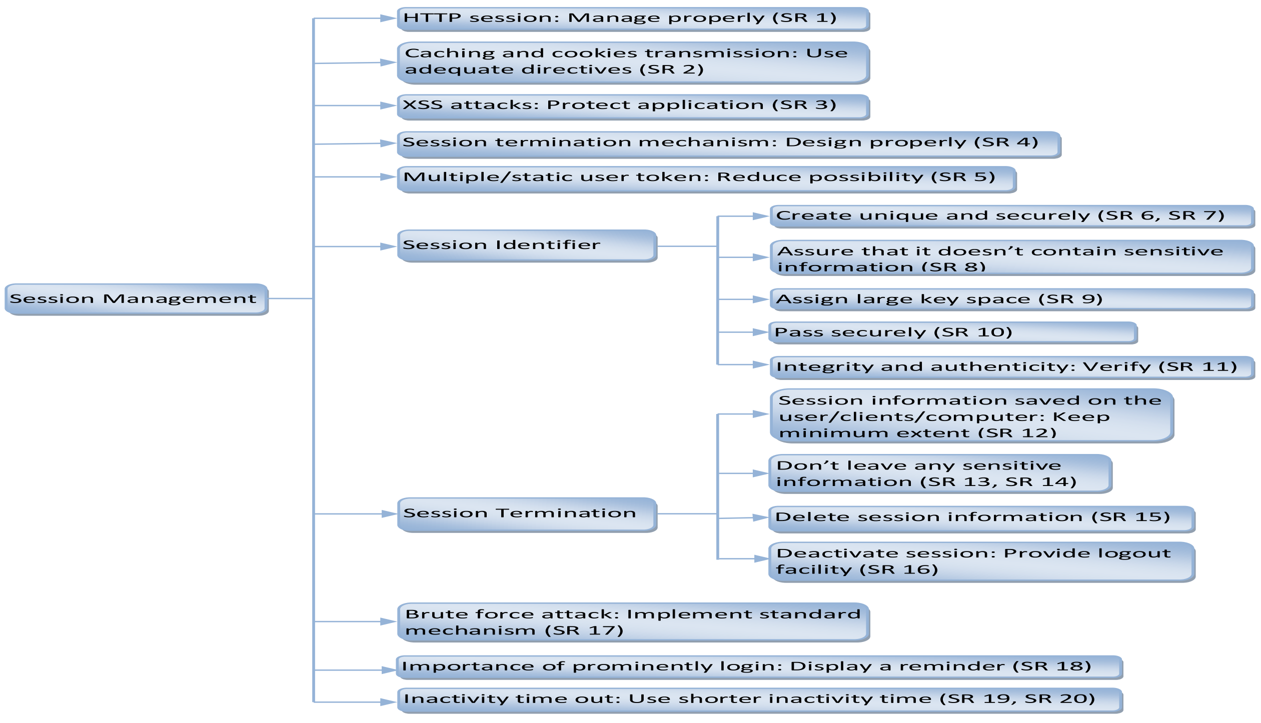
Session Management
HTTP session: Manage properly (SR 1)
Caching and cookies transmission: Use adequate directives (SR 2)
XSS attacks: Protect application (SR 3)
Session termination mechanism: Design properly (SR 4)
Multiple/static user token: Reduce possibility (SR 5)
Session Identifier
Create unique and securely (SR 6, SR 7)
Assure that it doesn't contain sensitive information (SR 8)
Assign large key space (SR 9)
Pass securely (SR 10)
Integrity and authenticity: Verify (SR 11)
Session Termination
Session information saved on the user/clients/computer: Keep minimum extent (SR 12)
Don't leave any sensitive information (SR 13, SR 14)
Delete session information (SR 15)
Deactivate session: Provide logout facility (SR 16)
Brute force attack: Implement standard mechanism (SR 17)
Importance of prominently login: Display a reminder (SR 18)
Inactivity time out: Use shorter inactivity time (SR 19, SR 20)

Figure 2: Structured Representation Model of Session Management SRs

III. IMPLEMENTATION MECHANISM

Following are the guidelines/ steps for implementation of the ^{S}RDF:

A. SRS will be taken as input in the ^{S}RDF.

B. The first step as per the ^{S}RDF will be vulnerability identification based on the well accepted databases available in the literature namely OWASP.

C. Second step is the classification of identified vulnerabilities in the above step based on the several attributes given in ^{S}RDF.

D. Next step is severity estimation based on the statistical formulation, proposed in ^{S}RDF, which in turn provides a quantified value for each vulnerability.

E. In the fourth step, value of the severity may be compared with the threshold values as given in ^{S}RDF, which are classified in three terms namely high, medium, and low.

F. If the severity level is tolerable, requirement engineers should hand over the SRS to designers.

G. Further, if it is not tolerable, proposed security requirements must be incorporated into the SRS for the mitigation of vulnerabilities. Although, for building the secure web application, mitigation of all the vulnerabilities must be done. But, the depending upon the time, nature and the most important aspect i. e., need of the application, security requirements as proposed in ^{S}RDF may be incorporated.

H. After incorporating these security requirements, SRS will be called secure SRS, which will serve as output of the framework. This secure SRS should again submitted for the designing purpose.

IV. TRYOUT RESULTS

The proposed framework, for development of security requirements has been validated by using SRS of one live project, Web Store System. This Web Store System is designed to allow new online store owners a quick and easy means to setup and perform sales and other core business over the internet.

As per implementation mechanism of ^{S}RDF, first step is vulnerability identification. In the subsections A, all the possible vulnerability that may occur for a web application, have already been identified. But, in the scenario of this project, 21 vulnerabilities may occur. Therefore, the detailed classification and severity estimation of these already identified vulnerabilities, as prescribed in ^{S}RDF is hereby accomplished. The Table 3 presents the values of severity computed through statistical formulation for the vulnerabilities as well as project need (security is highly needed as discussed above) and final recommendation in terms of security requirements:

Table 3: Vulnerabilities and their Severity

Vulnerability ID	Name of the Vulnerability	Severity
V1	Insufficient Authentication	H
V2	Insufficient Authorization	H
V3	Integer Overflows	M
V4	Insufficient Transport Layer Protection	H
V5	Format String	H
V6	Buffer Overflow	H
V7	Cross-site Scripting	H
V8	Cross-site Request Forgery	H
V9	Brute Force	H
V10	Information Leakage	H
V11	Server Misconfiguration	M
V12	Application Misconfiguration	M
V13	Directory Indexing	M
V14	Improper Filesystem Permissions	H
V15	Credential/Session Prediction	H
V16	SQL Injection	H
V17	Improper Input Handling	H
V18	Improper Output Handling	H
V19	Session Fixation	H
V20	Insufficient Session Expiration	H
V21	Insufficient Password Recovery	M

^{S}RDF proposes various security requirements for the mitigation of these 21 above mentioned vulnerabilities. After the study of SRS (complete and ready to submit for designing), researcher found that some of the security requirements have already been taken care of. But there are a number of remaining security requirements, to be incorporated before proceeding to the design phase. Here, as mentioned in earlier section, we are providing security requirements only for session management in the following Table 4:

Table 4: Proposed Security Requirements (SR) and Compliance Status

Vulnerability ID : Name of the vulnerability	**Proposed Security Requirements (SR)**	**Compliance Status (Y/N)**
V1: Insufficient Authentication V21: Insufficient Password Recovery V10: Information Leakage	*Minimal:* Incorporate SR1 to SR8.	SR1 to SR3 were already adapted; further, SR4 to SR8 have been incorporated.
	Clients/organizations: Incorporate SR1 to SR2.	Y
	Administrators: Incorporate SR1 to SR4.	Y
	Users: Incorporate SR1 to SR3.	SR1 was already there; SR2 to SR3 have been incorporated.
	Password Selection Rules: Incorporate SR1 to SR4.	Y
	Password Best Practices: Incorporate SR1 to SR3.	Y
	Password Management: Incorporate SR1 to SR12.	SR1 to SR4 were already adapted; SR5 to SR7 have been incorporated. Further, SR8 to SR10 were already in place and again SR11 and SR12 have been added.
V2: Insufficient Authorization V14: Improper Filesystem Permissions	*Authorization:* Ascertain that SR1 to SR8 have been incorporated.	**Ascertained.**
	User Identification: Ascertain that SR1has been incorporated.	**Y**
	Controls of user ID: Ascertain that SR1 to SR4 have been incorporated.	**Y**
	Information Asset Access Control: Ascertain that SR1 to SR5 have been incorporated.	SR1 to SR3 ascertained; SR4 and SR5 were already in place.
	Application Asset Access Control: Ascertain that SR1 to SR6 have been incorporated.	**Ascertained.**
	Access to Database and Backend Systems: Ascertain that SR1 to SR4 have been incorporated.	**Ascertained**
	Access Rights granted to third party: Ascertain that SR1 to SR5 have been incorporated.	**Ascertained**
V15:Credential/Session Prediction V19:Session Fixation V20:Insufficient Session Expiration	Session Management: Ascertain that SR1 to SR20 have been incorporated.	SR1 to SR5 were already adopted; SR6 to SR12 have been incorporated. Further, SR13 to SR17 were already in place and again SR18 and SR20 have been added.
V3: Integer Overflows V5: Format String V6: Buffer Overflow V7: Cross-site Scripting V8: Cross-site Request Forgery V9: Brute Force V16:SQL Injection V17:Improper Input Handling V18:Improper Output Handling	Data Validation: Ascertain that SR1 to SR 12 have been incorporated.	SR1 to SR5 ascertained; again SR6 to SR12 were already incorporated.
V10:Information Leakage V13:Directory Indexing	Cryptographic Protection of Information: Ascertain that SR1 to SR 4 have been incorporated	Ascertained
	Error Handling: Ascertain that SR1 to SR 9 have been incorporated	SR1 to SR3 ascertained; SR4 and SR5 were already in place.
V4:Insufficient Transport Layer Protection	Data Transport Security: Ascertain that SR1 to SR 8 have been incorporated.	SR1 to SR5 ascertained; again SR6 to SR8 were already incorporated.
	Data Encryption: Ascertain that SR1 to SR 10 have been incorporated.	SR1 to SR7 ascertained; again SR8 to SR10 were already incorporated.
	Encryption key Management: Ascertain that SR1 to SR 17 have been incorporated.	SR1 to SR4 ascertained; SR5 to SR9 have been incorporated. Further, SR10 to SR13 were already in place and again SR14 to SR17 have been added.
V11:Server Misconfiguration V12:Application Misconfiguration	Web Application and Server Configuration: Ascertain that SR1 to SR 13 have been incorporated.	SR1 to SR3 were already adapted; SR4 to SR9 have been incorporated. Further, SR10 to SR12 were already in place.

^{S}RDF was implemented on the SRS of above mentioned real life project. As discussed, 21 possible vulnerabilities that may occur with the developed web application have been identified by our framework. Since, security is highly needed in this application, mitigation of all these vulnerabilities is essential by incorporating security requirements. For each vulnerability, the corresponding SRs have been suggested and the adherence of the same is rechecked in the SRS. On this basis, wherever security requirements were not in place, the same has been adapted. By incorporating all these SRs, SRS of the proposed application will be strengthened with reference to security, which is the current need of the said application.

V. CONCLUSION AND FUTURE WORK

The proposed framework, ^{S}RDF may be used for the severity estimation of the vulnerabilities in the quantitative manner for the requirements phase of SDLC. Once the final value of the severity is calculated, its tolerance level should be checked. This tolerance level depends upon the nature of the project. Accordingly, three levels of severity e.g. high, medium, low may be fixed. If value of the severity is high, it will not be tolerable and it must be mitigated before moving to the next step. Requirement engineers should repeat the steps from beginning, iteratively. On the other hand, if the value of the severity is medium, it may or may not be tolerable. Depending upon the project type and resources, mitigation techniques should be used. Finally, if value of the severity is low, it may be tolerable and mitigation may or may not be required. Based on the severity determination, the next step should be followed. If severity is tolerable, SRS should be processed for Design Phase, otherwise incorporation of security requirements will be required.

^{S}RDF can be effectively used in classification and severity estimation of vulnerabilities in exhaustive way. ^{S}RDF is also validated through different tryouts on a SRS of live project provided by the industry. However, for the standardization of the results, a large number of sample projects are required. It appears to be an evolving process as new vulnerabilities and their corresponding security requirements shall be identified. Therefore, extension/modification and proposal of new security requirements may also be done. In future, new vulnerabilities and security requirements must be added in ^{S}RDF. A software tool may also be developed for the automation of complete process. This work may provide guidance to the industry as well as academia for developing more secure software.

VI. REFERENCES

[1]. Premchand, “Building India as the Destination for Secure Software Development- Next wave of Opportunities for the ICT Industry”, LNCS Volume 3803/2005, pp 49-65, Springer Berlin/ Heidelberg 2005

[2]. Steve Lipner, The Trustworthy Computing Security Development Lifecycle, proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), p.2-13, December 06-10

[3]. S. A. Hadavi, V. S. Hamishagi, H. M. Sangchi; Security Requirements Engineering; State of the Art and Research Challenges. Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I IMECS 2008, Hong Kong

[4]. IBM Security Solutions, IBM X-Force® 2010 Mid-Year Trend and Risk Report, August 2010

[5]. Nancy R. Mead, Gary Mcgraw, "A Portal for Software Security", Published By The Ieee Computer Society, Ieee Security & Privacy, 2005.

[6]. Hoglund, G. and McGraw, G. (2004) *Exploiting Software: Howto Break Code*. Addison-Wesley.

[7]. Piessens F, A. (2002) Taxonomy of causes of software vulnerabilities in Internet software, Supplementary Proceedings of the *13th International Symposium on Software Reliability Engineering* (Vouk, M., ed.), (pp. 47-52).

[8]. Y. Chen, “Stakeholder Value Driven Threat Modeling for Off The Shelf Based Systems,” International Conference on Software Engineering, IEEE Computer Society Washington, DC, USA, 2007, pp. 91-92

[9]. G. Eschelbeck, “The Laws of Vulnerabilities: Which security vulnerabilities really matter?,” Information Security Technical Report, vol. 10, 2005, pp. 213-219

[10]. Y. Lai and P. Hsia, “Using the vulnerability information of computer systems to improve the network security,”Computer Communications, vol. 30, Jun. 2007, pp. 2032-2047.

[11]. R. Rieke, “Modelling and Analysing Network Security Policies in a Given Vulnerability Setting,” Critical Information Infrastructures Security, First International Workshop, CRITIS 2006, Samos Island, Greece. Volume 4347. Springer. pp. 67-78.

[12]. M. Ishiguro, H. Tanaka, K. Matsuura, and I. Murase, “The Effect of Information Security Incidents on Corporate Values in the Japanese Stock Market,” International Workshop on the Economics of Securing the Information Infrastructure (WESII), 2006

[13]. R. Telang and S. Wattal, “An empirical analysis of the impact of software vulnerability announcements on firm stock price,” IEEE Transactions on Software Engineering, vol. 33, 2007, pp. 544-557

[14]. D. Stuttard and M. Pinto, *The Web Application Hacker’s Handbook: Discovering and Exploiting Security Flaws*, John Wiley & Sons, 2008.

[15]. C. Anley, “Weak Randomness: Part I—Linear Congru-ential Random Number Generators,” Next Generation Security Software, 2007; www.ngssoftware.com/Librar-ies/Documents/02_07_Weak_Randomness.sfl b.ashx.

[16]. M. Kolšek, “Session Fixation Vulnerability in Web-Based Applications,” Acros Security, Dec. 2002; www. acrossecurity.com/papers/session_fi xation.pdf.

[17]. “OWASP Testing Guide v3,” Open Web Application Security Project Foundation, Nov. 2008; www.owasp. org/index.php/OWASP_Testing_Guide_v3_Table _of_Contents.