Secure Paradigm For Web Application Development

B. Subedi[1], Abeer Alsadoon[1], P.W.C. Prasad[1], A. Elchouemi[2]

[1]School of Computing and Mathematics, Charles Sturt University, Sydney, Australia

[2]Walden University

Abstract— **Security protection is usually thought to be a separate process in web application development phases but the external security protection mechanisms are not effective to control threats and vulnerabilities in web applications. As a consequence, researchers have realized security development should be an integral part of System Development Lifecycle of web applications. This article presents a universal secure paradigm which the web developers can apply in the development process to enhance the security features of web applications. The proposed paradigm is an extension to security development practices with agile methodology. It consists of three phases, i.e., inception, construction and transition. Inception can be mapped to analysis stage of traditional development life cycle process and transition refers to security assurance stage before deployment whereas construction phase is iterative process of security development.**

Keywords*— *development lifecycle; web application; security; vulnerabilities; threats; risks

I. Introduction

Although it is not essential for users to browse a website to access a web application service but majority of web applications are deployed in cloud environment to make accessible via Internet. Therefore, hackers always try to access, alter or damage these contents maliciously. The attackers seek for weak parts in the application called vulnerabilities to take control over all, or some part of the applications.

A report highlighted that 86% of 30,000 websites tested had at least one serious vulnerability, and most had more than one [1]. Out of them more than 75% of websites had vulnerabilities due to the weaknesses in web application development process. Therefore, secure development principles are required to apply in each stage of web application development lifecycle [1].

The goal of this work is to propose a secure paradigm that can be effectively practiced in development lifecycle to build secure web based application. The proposed solution is the security enhancement practices that can be easily adopted by all types of web developers. It is the extension to agile methodology to handle security issues in early stage and mitigate them in development process.

The article is organized as follows. Next section II presents common issues with web application security followed by literature review on secure development models in web application development in section III. The explanation on the methodology used, current secure paradigm, proposed new paradigm in sections IV, V and VI respectively. Finally, the survey analysis discussion in VII followed by the conclusion and recommendations in VIII.

II. Background

The security issues in Web Application can be classified into three categories. The first issue is called vulnerability referred to some flaw or weakness in application which hackers or attackers can exploit to damage confidentiality, integrity, or availability of data and resources processed by the application [2]. The second issue called threat can be exploited if one or more vulnerability exists in the system [3]. Means external agents may want to or can result harm to the system in any form [3]. The third is called risk referred to possibility of something bad happening. The risk is a state where threat and vulnerability overlap [3]. It means there is a risk when the application have a vulnerability that a given threat can attack. Each issue can be of various types. Table 1

TABLE 1Common Vulnerabilities and Variation

Types	**Variation**
Weak Authentication	Broken Authentication, Week or blank passwords
Missing Validation	Not validated Redirects and Forwards, Hidden field manipulation
Broken Session Management	NA
Design flaws	Model defect
Code flaws	Insecure direct object references, bugs and defects, object cross reference, improper error handling, output encoding / escaping, insecure statement and syntax
Misconfiguration	Security misconfiguration, insecure cryptographic storage
System calls	Dangerous system calls

TABLE 2 Common Risks and Variation

Type	**Variation**
Privacy Risk	NA
Exploitation Risk	Risk of losing data integrity, risk of failure, damage potential, structural severity
Business Risk	Risk of losing business
Technical Risk	NA

TABLE 1 Common Threats and Variation

Type	Variation
Injection	SQL injection, Command line injection, General script injection
Forgery	Tampering, Spoofing, Cross site request forgery, Man in the middle, protocol sniffing
Attack	Denial of service attacks (DDos), System attack, Service attack
Malware	Malicious code search, Malicious file execution
Cross Scripting	Cross Site Scripting,
Session Hijacking	NA

highlights the most common vulnerabilities, Table 2 the most common risks but not limited to those in web application , and Table 3 the most common threats.

Security measures are tools, techniques and procedures which are applied in web application to mitigate vulnerability, control threats and reduce risks. Researchers have found several security measures to address the web application security issues. These include Web Application Firewalls, API based security control mechanism, Static and dynamic code analysis etc. None have shown effective in addressing all of the security issues. Current research is focused on finding security measures which have been applied in web application development process and seek a universal paradigm that could effectively address all of the security issues.

III. Literature Review and related work

As web development is inherent to software development, developers have been practising various forms of system development lifecycles (SDLC) to build web. The common practices are traditional waterfall, spiral or current agile approaches. To build a secure web the researchers have proposed security development lifecycles adding security development principles in common development practices. Though fulfilling functional goals are important to meet requirements of the application but security goals are unavoidable for availability, reliability, integrity and indeed the security of the application.

A. *Current Secure Paradigms in Web Application Development*

Masood and Java [4] have proposed Threat Modelling and Static Code Analysis techniques in coding phase of the development. The threat modelling can identify threats and help finding mitigating options. Similarly, static code analysis helps detecting vulnerabilities in the source code. But major process the model lacks is testing. Without effective testing the security assurance cannot be validated.

Integrated security activities at each stage of SDLC provided by Teodoro et al. [5] have extensive processes from the very beginning training and awareness to secure coding and post-deployment security assessment activities. Applications prioritization, risks classification, security requirements definition, threat modelling and architecture design review phases try to solve vulnerabilities in analysis and design activities. Although it has tried to address most of the security issues, it inherits the drawbacks of traditional waterfall method of being inflexible to change or flaws in earlier stages.

Huang et al. [6] have tried to introduce risk analysis in their hybrid (agile + waterfall) paradigm. Still security enhancement processes are missing in the other stages. Similarly, hybrid V-model by Abdulrazeg et al. [7] sounds quite impressive in System Requirements Specification (SRS) stage but fail to provide security practices in other stages like design, implementation and testing. The model recommends scrum agile approaches in generating System Requirements Specification. The elicited security requirements are mapped with misuse case diagram, reviewed and improved in next scrum cycle. This methodology is good at improving security goals but lack the implementation details.

Model oriented security requirements engineering by Salini and Kanmani [8] proposed use of Unified Modeling Language (UML) diagrams in analysis phase and design phase. The process starts by generating use case diagram followed by identifying security goals, objectives, threats and vulnerabilities. The threats are categorized under security goals by doing risk assessment. After that misuse case diagrams are generated to identify security requirements. Finally, the use case diagrams are refined based on security requirements. With the security added use case model class diagrams, activity diagrams, and sequence diagrams are modelled. The paradigm is extensive in the analysis and design phase whereas it lacks implementation, testing and validation details.

Risk of Software Vulnerability Exploitability assessment process by Younis et al. [9] have given methodology of security level assessment in implementation and verification phase of security life cycle. As the vulnerabilities in application are the sign of risk and threat, securing the program source is the key to application security. This allowed assessment of system security based on systematic evaluation and not subjective.

B. *Current Security Tools Implemented in Various Stages of Web Application*

Beside the paradigms the literature review has also analyzed current tools applicable in each stage of web application development. Masking technique by Goldsteen et al. [10] is applicable in implementation stage. This technique intercepts network messages between a web application server and a web browser and alter them to mask sensitive information. It helps to mitigate packet sniffing and man in the middle attack. Distributed denial-of-service (DDos) attack detection method by Liao et al. [11] can be applied after post deployment. As the core feature in the method is use of machine learning by analyzing request behavior of users it can categories general users and malicious users. The malicious users can be blocked thereafter. An Interface Design Secure Measurement Model by [12] have tried to mitigate the security vulnerabilities and defects present in interface design by metric formulation of factors. Pathak et al. [13] and Busch et al. [14] both have depicted use of UML for designing security features in web application. Beside functional requirements the security requirements can be integrated in the UML diagrams like use case diagram, misuse case diagram, class diagram, flow chart, state diagram, sequence diagram, activity diagram etc. which are impressive to demonstrate security goals in design phase. By inspecting and reviewing the modelled design, the flaws can be detected effectively. Another tool called Web Application Security by Static Analysis and Runtime Inspection (Webs SARI) by Huang et al. [15] helps finding security vulnerabilities in source code.

Through analysis of impact on the security factors and coverage of security practices in all stages of Web Application development lifecycle The Extended Agile Development Process to develop acceptably secure software by Othmane et al. [16] is applied effectively in a web portal which have given significant result. The detailed analysis of the model is presented in section IV

IV. Current Secure Paradigm (The Extended Agile Development Process to Develop Acceptly Secure Web Application)

As being flexible to change due to incremental iterative development approach the Agile approach is one of the popular web development paradigm. In this approach the application requirements are listed in general description called user stories. At each iteration few of the stories are selected, analyzed and developed. At the end of each iteration the application is tested and feedbacks are collected. The feedbacks are amended in next iteration along with the development of new user stories. Othmane et al. [16] have added security engineering activities like threat modelling, risk estimation, secure design and coding, security tests, and security assurance via expert review to effectively handle security factors in the web application development. The strengths and limitations of the model are explained in below sub sections. The analytic diagram of the current solution is shown in figure 1.

A. Current Solution Strengths

The model has three phases. Inception is the first phase where threat modelling, risk estimation and security goals identification is done to identify inputs, possible attacks, authorization, risks and their exploitation level. The second phase is construction which undergoes incremental iterative development until all the functional and security goals are achieved. As the beauty of agile is incremental iterative development approaches it looks for security metrics as soon as each iteration is complete, collects feedbacks and amends it in next iteration. By this, stakeholders are always aware about picture of undergoing web application and as soon as the security flaws are identified they can be mitigated early as possible. In addition, the security practices in the model help mitigating the vulnerabilities in implementation and testing. The final phase is Transition which is explicitly allotted for extensive security assurance activities.

B. Current Solution Limitations and Possible Solutions

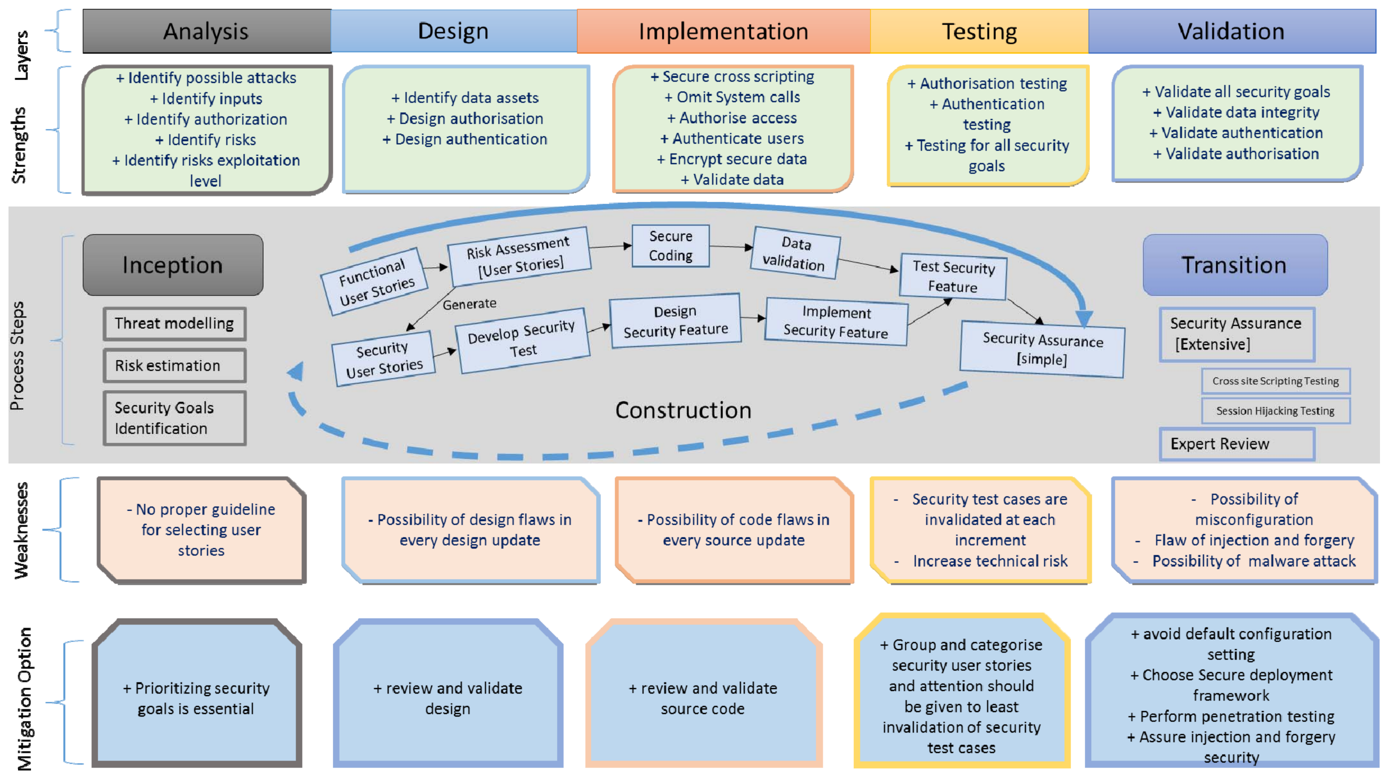
The limitations determined through analysis are listed below:

1) No proper guideline for selecting user stories

This limitation can be solved by finding a method to prioritize security goals.

2) Possibility of design and code flow in every design and source update

Proper validation technique should be added to mitigate



Layers
Analysis
Design
Implementation
Testing
Validation
Strengths
+ Identify possible attacks
+ Identify inputs
+ Identify authorization
+ Identify risks
+ Identify risks exploitation level
+ Identify data assets
+ Design authorisation
+ Design authentication
+ Secure cross scripting
+ Omit System calls
+ Authorise access
+ Authenticate users
+ Encrypt secure data
+ Validate data
+ Authorisation testing
+ Authentication testing
+ Testing for all security goals
+ Validate all security goals
+ Validate data integrity
+ Validate authentication
+ Validate authorisation
Process Steps
Inception
Threat modelling
Risk estimation
Security Goals Identification
Functional User Stories
Risk Assessment [User Stories]
Secure Coding
Data validation
Test Security Feature
Generate
Security User Stories
Develop Security Test
Design Security Feature
Implement Security Feature
Security Assurance [simple]
Construction
Transition
Security Assurance [Extensive]
Cross site Scripting Testing
Session Hijacking Testing
Expert Review
Weaknesses
- No proper guideline for selecting user stories
- Possibility of design flaws in every design update
- Possibility of code flaws in every source update
- Security test cases are invalidated at each increment
- Increase technical risk
- Possibility of misconfiguration
- Flaw of injection and forgery
- Possibility of malware attack
Mitigation Option
+ Prioritizing security goals is essential
+ review and validate design
+ review and validate source code
+ Group and categorise security user stories and attention should be given to least invalidation of security test cases
+ avoid default configuration setting
+ Choose Secure deployment framework
+ Perform penetration testing
+ Assure injection and forgery security

Fig 1 Detailed analysis of current paradigm (The Extended Agile Development process to develop acceptably secure web application)

this limitation.

3) Security test cases are invalidated at each increment

To reduce complexity of this problem the security user stories can be grouped, categorize and attention should be given to least invalidation of security test cases.

V. Proposed Secure Paradigm for Web Development

Figure 2 presents the proposed Secure Paradigm for web development which is the enhanced model to security practices in agile. The paradigm has attempted to solve the issues raised in agile approaches to security development in web application.

A. Inception as a First Phase in Paradigm

Inception is the early phase of development where scope of the project and model of initial architecture is defined. Along with the initial architecture security goals should be identified, group and prioritized. The necessary steps for identifying, grouping and prioritizing the security goals are described below.

1) Security Goals identification in Inception

A team of product owner, developer and security expert conduct a workshop to identify vulnerabilities, threats and risks of proposed web application. Any desired threat modelling and risk estimation techniques or a novel approach can be used to identify the security goals. As an agile approach the stakeholders can toss the possible security issues. During the process presence of security expert is essential to assess the impact of each security issues.

2) Prioritizing Security Goals in Inception

The development team conduct security impact analysis and prioritize the goals. Similar to the identification process the perceptions of likelihood of occurrence of vulnerabilities, threats and risks are collected from developers and security experts. In the same way product owner also analyses severity of the identified issues. After that priority of security goals are created based on criticality. The mitigating techniques for vulnerabilities and threats having very high impact should be implemented earlier.

3) Grouping Security Goals in Inception

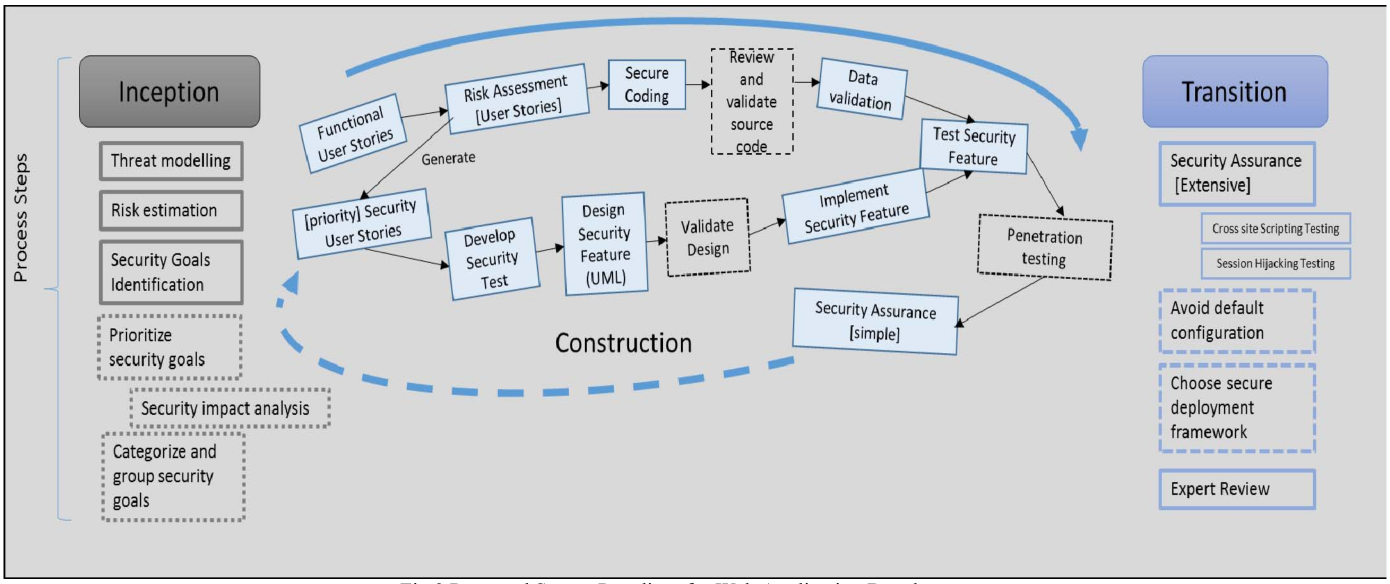
In most cases identification and impact of security issues are different but the implementation process is same. For example, broken authentication, weak or blank passwords, authorization etc. come under the same group authentication during implementation. Grouping and categorization of security goals helps testing and assurance process to be comprehensive and reduces the chance of test case invalidation in each iterative cycle. Security experts and experienced developers can work out to group the similar security goals. This process simplifies implementation and reduces technical risks.

B. Construction as a Second Phase in Paradigm

The construction phase is for developing the web application in incremental iterative process. In each iteration the product owner selects functional user stories and security user stories. The security user stories are selected based on priority of the security issues. Risk assessment is conducted on functional user stories which can generate more security issue. For example, let's consider a user story "As a student, I want to view and update my personal information". In this case it can be assessed that Authentication, Authorization, Input validation etc. security goals can be identified. After collecting security goals for the iteration, security test case is developed. Security design is made using UML which helps in cross validation due to consistency in representation. On the other side secure coding practices are used to implement functionalities. Concurrently security features are also implemented. After coding, code review is conducted to identify the vulnerabilities followed by data validation and testing. Automatic and light weight penetration testing is conducted at the end of each cycle to ensure the minimal security goals are achieved.

C. Transition as a Third Phase in Paradigm

Transition phase is before deployment for production in



Process Steps
Inception
Threat modelling
Risk estimation
Security Goals Identification
Prioritize security goals
Security impact analysis
Categorize and group security goals
Functional User Stories
Risk Assessment [User Stories]
Secure Coding
Review and validate source code
Data validation
Test Security Feature
Generate
[priority] Security User Stories
Develop Security Test
Design Security Feature (UML)
Validate Design
Implement Security Feature
Penetration testing
Security Assurance [simple]
Construction
Transition
Security Assurance [Extensive]
Cross site Scripting Testing
Session Hijacking Testing
Avoid default configuration
Choose secure deployment framework
Expert Review

Fig 2 Proposed Secure Paradigm for Web Application Development

real world. Before going live in production all identified security goals must be tested again to assure it. The tests can be cross site scripting testing, session hijacking testing, penetration testing etc. and more others. Use of default ports, default settings, default passwords etc. can exploit to high security risk. Therefore, custom configuration should be used in production. Deployment frameworks like Apache, Internet Information Services (ISS) also have security holes. These frameworks should be tested to be secure in production. Finally, experts review is essential to overall assurance.

VI. METHODOLOGY

The research methodology adopts the process of Literature Review, Analysis, Propose a new solution and Expert Survey Analysis.

Firstly, through Literature Review information on current solutions are gathered. Secondly the potential paradigms and tools are analysed based on the impact on three main security issues vulnerabilities, threats and risks. Thirdly the strengths of compatible solutions are combined to find a new universal secure paradigm. Finally, survey is conducted to collect experts view on the proposed solution. The survey is conducted to collect the views of software developers specialized in web development as well as cyber security. 125 questions related to web security, vulnerabilities, threats and risks which are possible impacts of the proposed framework were prepared. It aims to identify impacts of the proposed secure paradigm in mitigating security issues of web application.

A. Participants

20 respondents (software engineer or web application engineer) have provided feedback on the survey. 95% of the respondents have experience above 2 years and 55% have experience above 5 years. Among those respondents 55% have experience or qualification related to security. After collecting the data, analysis was done in Statistical Package for the Social Sciences (SPSS) tool. Each objectives of the proposed solution are analysed in Bivariate Correlation Table.

B. Questionnaire design and Measurement Scale

Initially the literature review, and analysis and evaluation of previous related research was conducted to collect the factors and their components which have significant impact on security of web application. Secondly the security factors at each stage of development lifecycle were analysed. Thirdly the possible outcomes of applying proposed paradigm at each stage were determined. Finally, data collected from above findings were used to generate detailed questionnaire for developers and security experts in web application development.

The survey consists of mixture of closed ended and open ended questions, as well as Likert scale questions. These structured questions focused on collecting experts and developer's perception in impact of proposed paradigm to mitigate security issues of web application. Though proposed paradigm adopts incremental iterative agile methodology, each iteration passes through each stage of analysis, design, implementation, testing and validation. Therefore, the impact of proposed paradigm at each stage has been analysed.

C. Data Analysis

The basic principle behind the analysis is to determine the impact of proposed paradigm to mitigate security issues in each stage of development process. Therefore, Bivariate Correlation analysis was conducted in IBM Statistical Package for the Social Sciences (SPSS) tool. This measures the strength of relationship of proposed parameters in the paradigm to mitigate security issues. Indeed, to analyse the experts and developer's perception of the proposed universal secure paradigm.

VII. RESULTS OF ANALYSIS

The survey has collected the views of web developer and security experts how the vulnerabilities, threats and risks are interrelated. The current result is the first step in justification of the proposed solution. In the next stage it has to be implemented in real web development project.

A. Correlations

Referring to , in first row handling most critical vulnerabilities and threats with first priority has strong positive relationship with control of critical threats (r = .661**, p = 0.007), as well as reduce of critical risks (r = .732, p = .004).

TABLE 2 Correlations of impact of prioritizing security goals on mitigating security issues

		Most critical **vulnerabilities** must be handled with first priority	Mitigating the critical vulnerabilities helps to control critical **threats**	Controlling the critical threats reduces the critical **risks**
Most critical **vulnerabilities** must be handled with first priority	Pearson Correlation	1	$.661^{**}$	$.732^{**}$
	Sig. (2-tailed)		.007	.004
	N	15	15	13
Mitigating the critical vulnerabilities helps to control critical **threats**	Pearson Correlation	$.661^{**}$	1	$.857^{**}$
	Sig. (2-tailed)	.007		.000
	N	15	15	13
Controlling the critical threats reduces the critical **risks**	Pearson Correlation	$.732^{**}$	$.857^{**}$	1
	Sig. (2-tailed)	.004	.000	
	N	13	13	13
*. Correlation is significant at the 0.05 level (2-tailed).				
**. Correlation is significant at the 0.01 level (2-tailed).				

Similarly, in second row there is a strong positive relationship between control of critical threats and reduce of critical risks (r = .857, p = .000). From the strong relationship it can be concluded that the idea of handling critical vulnerabilities with priority help mitigating critical threats. Similarly, the web development and security experts also agree to the concept that mitigating the critical threats reduces critical risks.

VIII. Discussion

Analysing Impact of grouping security goals on reducing security issues experts believe that grouping security goals significantly impact to reduce vulnerabilities, threats and technical risks. Analysing Impact of UML Design in cross validation it is found that most web developers and security experts agree that cross validation of every design is essential to omit the design flaws. Similarly, they have common opinion to cross validate the design again if it is changed or updated. The UML design significantly help in securing designs through cross validation because of uniqueness in representation. Beside the secure web application, the deployment environment and configuration parameters also play vital role in securing the application. The voice of developers and experts is towards deploying the web application in a secure framework like OS, web/app server, use secure DBMS, libraries etc.

Analysing the survey results the experts realized that UML designs are unable to identify invalid inputs. In fact, invalid inputs should be identified in just after the identification of security goals in risk assessment of the construction phase. Those identified invalid inputs should be reflected in the security design.

Similarly, experts agree to the concept of code review but want it to be done just before the test. Means to establish the relationship between the security parameters of the source code, the code review and testing can be conduct simultaneously.

IX. Conclusion and Recommendations

This paper concludes that security practices can be implemented in iterative development approach. Prioritizing and grouping security goals reduces technical risks in implementation and testing process, and the code review and security testing process in construction stage enables ensuring the delivery of secure web application at end of each iteration.

The main benefits of this approach are: Firstly, it can be easily adopted by all groups of web developers. Secondly it helps mitigating the security issues in great extent. On the other hand, developers can reuse the identified security goals as a basepoint to enhance the security of next similar web application project.

Although this model is tested by collecting reviews from web developers and security experts there was limitation of time to apply in ongoing project. This paradigm can be better tested by applying in any web application development project. The future recommendation includes applying the paradigm in real project plus automating the development and security assurance process.

References

[1] W. H. Security, "Website Security Statistics Report," White Hat Security, 2013. https://info.whitehatsec.com/rs/whitehatsecurity/images/2015-Stats-Report.pdf [Accessed : 24-Feb-2016]

[2] N. DuPaul. , "Application Security Vulnerability: Code Flaws, Insecure Code," 2016. http://www.veracode.com/security/application-vulnerability [Accessed: 20-Feb-2016]

[3] pen-tests.com., "Difference Between Threat, Vulnerability and Risk,"2016. http://www.pen-tests.com/difference-between-threat-vulnerability-and-risk.html [Accessed: 15-Feb-2016]

[4] A. Masood and J. Java, "Static analysis for web service security - Tools &amp; techniques for a secure development life cycle," in IEEE International Symposium on Technologies for Homeland Security (HST), pp. 1-6, 2015.

[5] N. Teodoro and C. Serr , "Web application security: Improving critical web-based applications quality through in-depth security analysis," in International Conference on Information Society (i-Society), pp. 457-462, 2011.

[6] W. Huang, R. Li, C. Maple, H.-J. Yang, D. Foskett, and V. Cleaver, "A novel lifecycle model for Web-based application development in small and medium enterprises," International Journal of Automation and Computing, vol. 7, pp. 389-398, 2010.

[7] A. Abdulrazeg, N. Norwawi, and N. Basir, "Extending V-model practices to support SRE to build secure web application," in International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 213-218, 2014.

[8] P. Salini and S. Kanmani, "Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems," International Journal of Information Security, pp. 1-16, 2015.

[9] A. A. Younis, Y. K. Malaiya, and I. Ray, "Using Attack Surface Entry Points and Reachability Analysis to Assess the Risk of Software Vulnerability Exploitability," in IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE), pp. 1-8, 2014.

[10] A. Goldsteen, K. Kveler, T. Domany, I. Gokhman, B. Rozenberg, and A. Farkash, "Application-Screen Masking: A Hybrid Approach," IEEE Software, vol. 32, pp. 40-45, 2015.

[11] Q. Liao, H. Li, S. Kang, and C. Liu, "Feature extraction and construction of application layer DDoS attack based on user behavior," in 33rd Chinese Control Conference (CCC), pp. 5492-5497, 2014.

[12] S. T. Lai, "An Interface Design Secure Measurement Model for Improving Web App Security," in International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), pp. 422-427, 2011.

[13] N. Pathak, G. Sharma, and B. M. Singh, "Designing of SPF based secure web application using forward engineering," in 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 464-469, 2015.

[14] M. Busch, N. Koch, and S. Suppan, "Modeling Security Features of Web Applications," in Engineering Secure Future Internet Services and Systems Springer International Publishing, pp. 119-139, 2014.

[15] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing web application code by static analysis and runtime protection," presented at the Proceedings of the 13th international conference on World Wide Web, , 2004.

[16] L. B. Othmane, P. Angin, H. Weffers, and B. Bhargava, "Extending the Agile Development Process to Develop Acceptably Secure Software," IEEE Transactions on Dependable and Secure Computing, vol. 11, pp. 497-509, 2014.