Journal of Engineering Sciences
Vol 11, Issue 6,June/2020
ISSN NO:0377-9254

AN EFFICIENT DESIGN AND DEVELOPMENT OF SECURE WEB APPLICATION

BANGI GURU PAVAN, Dr.M.Padmavathamma

MCA STUDENT, DEPT. OF COMPUTER SCIENCE, SRI VENKATESWARA UNIVERSITY, TIRUPATI

PROFESSOR, DEPT. OF COMPUTER SCIENCE, SRI VENKATESWARA UNIVERSITY, TIRUPATI

Abstract

Developing a secure Web application is a troublesome undertaking. In this manner designers need a rule to assist them with developing a secure Web application. The rule can be utilized as an agenda for the designer to accomplish a base standard of the secure Web application. This examination assesses how great is OWASP rule in helping designers to construct a secure Web application. The created framework is then tried utilizing code reviewing and entrance testing to distinguish the accomplishment of the framework security for the application. In the wake of applying the testing procedures from Open Source Security Testing Methodology (OSSTMM) on the Top Ten Critical vulnerabilities as characterized by OWASP, a standard measure score is determined. The score is utilized to choose the degree of security of the created web application. A high rate score would demonstrate that the rule helps in building a secured web application. Henceforth, the outcome demonstrated that the OWASP rule is successful in guaranteeing the dependability of the framework and can be utilized as a referral by other web engineers particularly in developing applications for a college.

Keywords: Web Application, Security.

I. INTRODUCTION

World Wide Web has evolved from a system that delivers static pages to a platform that supports distributed applications, referred to as web applications, and become one among the foremost prevalent technologies for information and repair delivery over the web . The increasing popularity of web applications are often attributed to many factors, including remote accessibility, cross-platform compatibility, fast development, etc. The AJAX (Asynchronous JavaScript and XML) technology also enhances the user experiences of web applications with better instructiveness and responsiveness.

As web applications are increasingly wont to deliver security-critical services, they become a valuable target for security attacks. Many web applications interact with back-end database systems, which can store sensitive information (e.g., financial, health), the compromise of web applications would end in breaching a huge amount of data , resulting in severe economical losses, ethical and legal consequences. A breach report from Verizon [1] shows that web applications now reign supreme in both the amount of breaches and therefore the amount of knowledge compromised.

The Web platform may be a complex ecosystem composed of an outsized number of components and technologies, including HTTP protocol, web server and server-side application development technologies (e.g., CGI, PHP, ASP), browser and client-side technologies (e.g., JavaScript, Flash). Web application built and hosted upon such a posh infrastructure faces inherent challenges posed by the features of these components and technologies and therefore the inconsistencies among them.

Current widely-used web application development and testing frameworks, on the opposite hand, offer limited security support. Thus secure web application development is an error-prone process and requires substantial efforts, which might be unrealistic under time-to-market pressure and for people with insufficient security skills or awareness. As a result, a high percentage of web applications deployed on the web are exposed to security vulnerabilities. consistent with a report by the online Application Security Consortium, about 49% of the online applications being reviewed contain vulnerabilities of high-risk levels and quite

13% of the websites are often compromised completely automatically [2]. A recent report [3] reveals that over 80% of the websites on the web have had a minimum of one serious vulnerability.

Motivated by the urgent need for securing web applications, a considerable amount of research efforts are dedicated to this problem with several techniques developed for hardening web applications and mitigating the attacks. Many of those techniques make assumptions on the online technologies utilized in the appliance development and only address one particular sort of security flaws; their prototypes are often implemented and evaluated on limited platforms. A practitioner may ponder whether these techniques are suitable for his or her scenarios. And if they can't be directly applied, whether these techniques are often extended and/or combined. Thus, it's desirable and urgent to supply a scientific framework for exploring the basis causes of web application vulnerabilities, uncovering the connection between the prevailing techniques, and sketching an enormous picture of the present research frontier during this area. Such a framework would help both new and experienced researchers to raised understand web application security challenges and assess existing defenses, and encourage them with new ideas and trends.

In this paper, we survey the state of the art in web application security, to systematize the prevailing techniques into an enormous picture that promotes future research. supported the conceptual security framework by Bau and Mitchell [4], we organize our survey using three components for assessing the safety of an internet application (or equipped with a defense mechanism): system model, threat model, and security property. System model describes how an internet application works and its unique characteristics; the threat model describes the facility and resources attackers possess; security property defines the aspect of the online application behavior intended by the developers. Given a threat model, if one web application fails to preserve certain security property under all scenarios, this application is insecure or susceptible to corresponding attacks.

This survey covers the techniques which consider the subsequent threat model: 1) the online application itself is benign (i.e., not hosted or owned for malicious purposes) and hosted on a trusted and hardened infrastructure (i.e., the trusted computing base, including OS, web server, interpreter, etc.); 2) the attacker can manipulate either the contents or the sequence of web requests sent to the online application, but cannot directly compromise the infrastructure or the appliance code. We note here that although browser security ([5], [6]) is additionally an important component in end-to-end web application security, research works on this subject usually have a special threat model, where web applications are considered as potentially malicious. This survey doesn't include the research works on browser security in order that it can specialise in the matter of building secure web applications and protecting vulnerable ones. The contributions of this paper are:
We present three aspects of web application development, which poses inherent challenges for building secure web applications and identify three levels of security properties that a secure web application should hold: input validity, state integrity, and logic correctness. Failure of web applications

To fulfill the above security properties is that the root explanation for corresponding vulnerabilities, which permit for successful exploits.

We classify existing research works into three categories: security by construction, security by verification and security by protection, supported their design principle (i.e., constructing vulnerability-free web applications, identifying and fixing vulnerabilities, or protecting vulnerable web applications against exploits at runtime, respectively) and the way security properties are assured at different phases within the life cycle of the online application. We aren't trying to enumerate all the prevailing works but have covered most of the represented works.

We identify several open issues that are insufficiently addressed within the existing literature. We also discuss future research opportunities within the area of web application security and therefore the new challenges that are expected ahead.

II. RELATED WORK

A web application is a dispersed application that is executed over the Web stage. It is a vital piece of

the present Web biological system that empowers dynamic data and administration conveyance. As appeared in Fig. 1, a web application may comprise of code on both the server-side and the customer side. The server-side code will produce dynamic HTML pages either through execution (e.g., Java servlet, CGI) or translation (e.g., PHP, JSP). During the execution of the server-side code, the web application may communicate with the neighborhood document framework or back-end database for putting away and recovering information. The customer side code (e.g., in JavaScript) are inserted in the HTML pages, which is executed inside the program. It can speak with the server-side code (i.e., AJAX) and progressively refreshes the HTML pages. In what follows, we portray three one of a kind parts of web application advancement, which separate web applications from conventional applications.

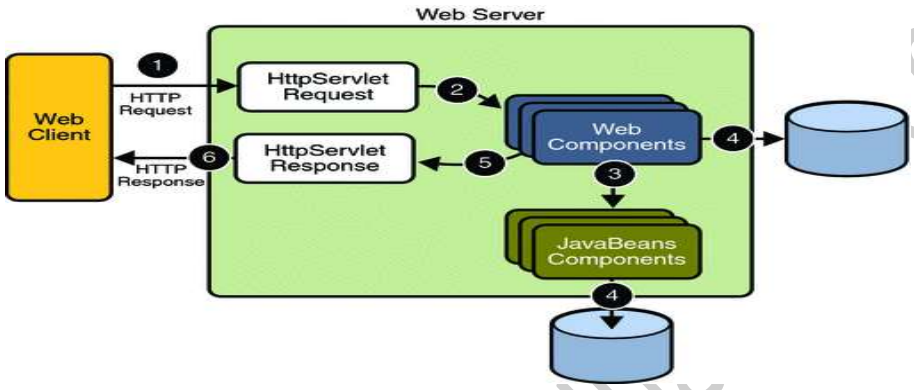

Fig. 1. Overview of Web Application

A. Programming Language

Web application improvement depends on web programming dialects. These dialects incorporate scripting dialects that are designed explicitly for web (e.g., PHP, JavaScript) and broadened customary universally useful programming dialects (e.g., JSP). A distinctive element of many web programming dialects is their kind of frameworks. For instance, some scripting dialects (e.g., PHP) are powerfully composed, which implies that the kind of a variable is resolved at runtime, rather than arrange time. A few dialects (e.g., JavaScript) are pitifully composed, which implies that an announcement or a capacity can be performed on an assortment of information types through certain pigeonholing. Such sort frameworks permit designers to mix a few kinds of builds in a single record for runtime translation. For example, a PHP record may contain both static HTML labels and PHP capacities and a web page may install executable JavaScript code. The portrayal of application information and code by an unstructured succession of bytes is a one of a kind element of the web application that helps upgrade the advancement effectiveness.

B. State Maintenance

The HTTP convention is stateless, where each web demand is free of one another. Be that as it may, to actualize non-inconsequential functionalities, "stateful" web applications should be based on this stateless foundation. Hence, the deliberation of web meeting is embraced to help the web application to distinguish and relate a progression of web demands from a similar client during a specific period. The condition of a web meeting records the conditions from the verifiable web demands that will influence the future execution of the web application. The meeting state can be kept up either at the customer side (by means of treat, concealed structure or URL reworking) or at the server-side.

In the last case, a special identifier (meeting ID) is characterized to file the express meeting factors put away at the server-side and gave to the customer. For instance, most web programming dialects (e.g., PHP, JSP) offer designers an assortment of capacities for dealing with the web meeting. For instance, in PHP, meeting start() can be called to instate a web meeting and a pre-characterized worldwide exhibit $ SESSION is utilized to contain the meeting state. In either case, the customer

assumes an indispensable job in keeping up the conditions of a web application.

C. Rationale Implementation

The business rationale characterizes the usefulness of a web application, which is explicit to every application. Such usefulness is showed as a proposed application control stream and is generally coordinated with the route connections of a web application. For instance, validation and approval are a typical piece of the control stream in many web applications, through which a web application confines its delicate data and special activities from unapproved clients. As another model, online business websites for the most part deal with the grouping of activities that the clients need to perform during shopping and checkout.

A web application is generally executed as a few autonomous modules, every one of which can be straightforwardly gotten to in any request by a client. This one of a kind component of web applications fundamentally confounds the authorization of the application's control streams across various modules. This assignment should be performed through a tight joint effort of two methodologies. The main methodology, which is polished by most web applications, is interface stowing away, where just open assets and activities of the web application are introduced as web connections and presented to clients. The subsequent methodology requires unequivocal checks of the application state, which is kept up by meeting factors (or industrious items in the database) before touchy data and activities can be gotten to.

III. WEB APPLICATION SECURITY PROPERTIES, VULNERABILITIES, AND ATTACK VECTORS

A secure web application needs to fulfill wanted security properties under the given danger model. In the region of web application security, the accompanying danger model has generally thought to be: 1) the web application itself is favorable (i.e., not facilitated or possessed for vindictive purposes) and facilitated on a trusted and solidified foundation (i.e., the confided in figuring base, including OS, web server, mediator, and so on.); 2) the assailant can control either the substance or the succession of web demands sent to the web application, yet can't legitimately bargain the framework or the application code. The vulnerabilities inside web application executions may abuse the expected security properties and take into account comparing fruitful adventures.

Specifically, a secure web application should safeguard the accompanying pile of security properties, as appeared in Fig. 2. Info legitimacy implies the client information ought to be approved before it very well may be used by the web application; state uprightness implies the application state ought to be kept untampered; rationale rightness implies the application rationale ought to be executed effectively as proposed by the engineers. The over three security properties are connected such that disappointment in saving a security property at the lower level will influence the affirmation of the security property at a more significant level. For example, if the web application neglects to hold the info legitimacy property, a cross-website scripting assault can be propelled by the assailant to take the casualty's meeting treat. At that point, the aggressor can capture and alter the casualty's web meeting, bringing about the infringement of state honesty property. In the accompanying areas, we portray the three security properties and show how the exceptional highlights of web application advancement muddle the security design for web applications.

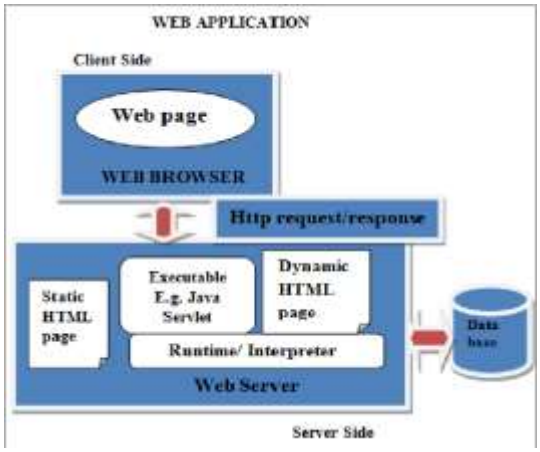

Fig. 2. Web Application Security Properties

Info Validity

Given the danger model, client input information can't be trusted. In any case, for the untrusted client information to be utilized in the application (e.g., creating web reaction or SQL inquiries), they must be first approved. In this way, we allude to this security property as information legitimacy property:

All the client information ought to be approved accurately to guarantee it is used by the web application in an expected manner.

The client input approval is regularly performed by means of disinfection schedules, which change untrusted client contribution to believed **information by separating dubious characters or builds inside client input. While straightforward** on a fundamental level, it is non-trifling to accomplish the fulfillment and rightness of client input disinfection, particularly when the web application is customized utilizing scripting dialects. In the first place, since client input information is engendered all through the application, it must be followed right to recognize all the sterilization focuses. Be that as it may, the dynamic highlights of scripting dialects must be taken care of properly to guarantee the right following of client input information. Second, right disinfection needs to consider the unique situation, which determines how the client input is used by the application and deciphered later either by the web program or the SQL mediator. Hence various settings require particular sterilization capacities. In any case, the feeble composing highlight of programming dialects makes setting touchy sterilization testing and mistake inclined.

In current web improvement rehearses, disinfection schedules are typically positioned by engineers physically in a specially appointed manner, which can be either deficient or incorrect, and in this way bring vulnerabilities into the web application. Missing cleansing permits vindictive client contribution to stream into believed web substance without approval; defective sterilization permits malevolent client contribution to sidestep the approval strategy. A web application with the above vulnerabilities neglects to accomplish the information legitimacy property, accordingly it is defenseless against a class of assaults, which are alluded to as content infusions, information stream assaults, or info approval assaults. This sort of assault implants malevolent substance inside web demands, which are used by the web application and executed later. Instances of info approval assaults incorporate cross-site scripting (XSS), SQL infusion, registry traversal, filename consideration, reaction parting, and so forth. They are recognized by the areas where pernicious substance gets executed. In the accompanying, we delineate the most two famous info approval assaults.

SQL Injection: A SQL infusion assault is effectively propelled when malignant substance inside client input stream into SQL questions without right approval. The database confides in the web application and executes all the questions gave by the application. Utilizing this assault, the aggressor can implant SQL catchphrases or administrators inside client contribution to control the SQL inquiry structure and result in unintended execution. Outcomes of SQL infusions incorporate confirmation sidestep, data exposure, and even the decimation of the whole database. The intrigued peruser can allude to [7] for additional insights regarding SQL infusion.

Cross-Site Scripting: A cross-webpage scripting (XSS) assault is effectively propelled when malevolent substance inside client input streams into web reactions without right approval. The web program deciphers all the web reactions returned by the confided in web application (as indicated by the equivalent inception approach). Utilizing this assault, the assailant can infuse malignant contents into web reactions, which get executed inside the

casualty's web program. The most widely recognized outcome of XSS is the divulgence of delicate data, e.g., meeting treat robbery. XSS normally fills in as the initial step that empowers additionally refined assaults (e.g., the infamous MySpace Samy worm [8]). There are a few variations of XSS, as per how the noxious contents are infused, including put away/constant XSS (vindictive contents are infused into persevering stockpiling), reflected XSS, DOM-based XSS, content-sniffing XSS [9], and so forth.

State Integrity

State upkeep is the reason for building stateful web applications, which requires a secure web application to safeguard the respectability of application states. Be that as it may, The association of an untrusted party (customer) in the application state upkeep makes the confirmation of state uprightness a difficult issue for web applications.

A few assault vectors focus on the vulnerabilities inside meeting the board and state upkeep systems of web applications, including treat harming (altering the treat data), meeting obsession (when the meeting identifier is unsurprising), meeting seizing (when the meeting identifier is taken), and so forth. Cross-site demand imitation (i.e., meeting riding) is a famous assault that falls in this classification. In this assault, the assailant fools the casualty into sending made web demands with the casualty's substantial meeting identifier, be that as it may, for the aggressor's benefit. This could bring about the casualty's meeting being altered, delicate data unveiled (e.g., [10]), monetary misfortunes (e.g., an aggressor may manufacture a web demand that trains a defenseless financial website to move the casualty's cash to his record), and so on.

To protect state uprightness, a few viable methods have been proposed [11]. The customer side state data can be ensured by honesty confirmation through MAC (Message Authentication Code). Meeting identifiers should be created with high arbitrariness (to shield against meeting obsession) and transmitted over secure SSL convention (against meeting commandeering). To moderate CSRF assaults, web solicitations can be approved by checking headers (Referrer header, or Origin header [12]) or related one of a kind mystery tokens (e.g., NoForge [13], RequestRodeo [14], BEAP [15]). Since the strategies for saving state trustworthiness are generally full grown, along these lines falling past the extent of this overview.

Rationale Correctness

Guaranteeing rationale rightness is vital to the working of web applications. Since the application rationale is explicit to each web application, it is difficult to cover all the viewpoints by one depiction. Rather, a general depiction that covers the most widely recognized application functionalities is given as follows, which we allude to as rationale accuracy property:

Clients can just access approved data and activities and are upheld to follow the planned work process gave by the web application.

To execute and implement application rationale accurately can be trying because of its state upkeep instrument and "decentralized" structure of web applications. To start with, the interface concealing procedure, which follows the rule of "security by indefinite quality", is insufficient, which permits the assailant to reveal shrouded joins and legitimately get to unapproved data or tasks or abuse the expected work process. Second, express checking of the application state is performed by designers physically and in a specially appointed manner. In this manner, all things considered, certain state checks are absent on startling control stream ways, because of those numerous section purposes of the web application. Additionally, composing right state checks can be blunder inclined, since static security approaches as well as powerful state data ought to be thought of. Both absent and defective state registers present rationale vulnerabilities with web applications.

A web application with rationale imperfections is helpless against a class of assaults, which are generally alluded to as rationale assaults or state infringement assaults. Since the application rationale is explicit to each web application, rationale assaults are additionally quirky to their particular targets. A few assault vectors that fall (or somewhat) inside this class incorporate confirmation sidestep, parameter altering, compelling perusing, and so forth. There are additionally application-explicit rationale assault

vectors. For instance, a defenseless online business website may permit a similar coupon to be applied on different occasions, which can be misused by the assailant to red.

OWASP guideline is applied throughout the software development life cycle (SDLC) phases in application development which are system planning, system analysis, system design, implementation, and testing as shown in Figure 1. Security requirements defined within the OWASP like authentication and authorization, input validation, and session handling is applied to make sure the system being developed is secure from security risks.

To have a typical measurement, the score value for the vulnerabilities mentioned above is defined in Table 1 below. The table has been modified from the simplified web application framework to guage the At the top of testing, all scores are going to be summed up and therefore the percentage are going to be calculated. This percentage are going to be analyzed to work out whether the online application is secure or not.

IV. PROPOSAL WORK

Web application security is technology-centric, supported by organizational policies and procedures, enforcement , and practices of the people related to the event and operations of Web applications.

Technology Solutions

First of all, technology controls should be addressed across the appliance infrastructure, involving networks, hosts (Web server, application server, and database server), and application. Especially, the subsequent technical controls should be used:

Protecting Internet communications. Protected communications are implemented through the utilization of knowledge encryption methods and deployment of cryptographic technologies 14. Encryption ensures the confidentiality of sensitive and important information while it's transmitted over the web . Public key cryptography with hash digest makes sure the integrity of data when it's in transit. By adding digital signatures, the authentication of the knowledge and nonrepudiation are often ensured.

Securing channels of communication. Secure Sockets Layer (SSL) of TCP/IP is that the commonest sort of securing communication channels. The SSL protocol provides encoding , server authentication, and knowledge integrity for TCP/IP connections 9. Hence, it makes sure the confidentiality, authenticity, and integrity of data during the transaction between the merchants and customers.

Intrusion Detection System (IDS) & anti-virus software. IDS detect security breaches and initiate an efficient response if a security breach is detected 14. IDS is the primary line of defense against security breaches. Anti-virus software is installed on servers and user workstations, providing virus protection by identifying and eradicating the foremost common sorts of software viruses to make sure system and data integrity 14. it's the simplest and least expensive thanks to prevent threats to system and data integrity.

Authentication & authorization. Authentication provides the means to verify the identity of the clients of Web applications and services. These clients could be end-users, or other services (Meier et al., 2003). The authentication mechanism includes passwords, personal identification numbers, or PINs, and emerging technologies, like a token, open-end credit , and digital certificate 14. Authorization specifies the allowed resources (e.g. files, databases, tables, and rows) and operations (i.e. on-line transactions) for the authenticated client 12.

Access control enforcement. Access controls define security policies and enforce the defined security policy on the authorized parties. the target of access control is to preserve and protect data integrity and confidentiality. The common access control mechanisms include MAC sensitivity labels, DAC file permission sets, access control lists, roles, and user profiles 14.

Auditing and logging. The activities, like successful and failed login attempts, retrieval, modification, and deletion of knowledge , network communication, and administrative functions, etc., should be audited and logged across the tiers of the appliance infrastructure 12. The auditing of security-relevant events and

therefore the monitoring and tracking of system abnormalities are key elements within the after-the-fact detection of and recovery form, security breaches 14.

Input validation. Unvalidated input has been continually listed because the top one Web application security flaw by OWASP (the Open Web Application Security Project) in 2003 and 2004. Unvalidated input may introduce buffer flows, cross-site scripting, and SQL injection and make the appliance vulnerable in hence 16. an honest practice for Web application developers is to verify user input by constraining certain type, length, format, and range before it's employed by the appliance and reject the invalid data also .

Session management. Because the HTTP protocol doesn't provide the power of persistent sessions, Web applications must create a session by themselves. However, if the session tokens created by Web developers aren't properly protected, an attacker can hijack a lively session and assume the identity of a user. the online application developer should create a scheme to make a robust session to guard them throughout their lifecycle.

Exception management. If error conditions that occur during normal operation aren't handled properly, attackers can gain detailed system information, deny service, and cause security mechanisms to fail or crash the server 16. Web applications should avoid leaking information to the client in a mistake message. Also, an in depth error message should be logged.

Policies and Procedures

To minimize the risks, organizations should develop a coherent corporate policy that takes under consideration the online assets that require to be protected, the character of the risks, also because the technologies and procedures required to deal with the risks 9.

Law Enforcement

There are some control organizations or programs to enforce computer crime laws. The Federal Trade Commission (FTC) takes advantage of the web for enforcement , operating a web , real-time complaint form at its website . The database maintained by FTC contained 250, 000 consumer fraud by June 2000. the web Fraud Complaint Center, launched by the Federal Bureau of Investigation (FBI), also collects computer crime complaints from the general public at its internet site . Internet Fraud Council (IFC) is creating a clearinghouse of data regarding Internet fraud. it's also studying and quantifying these incidents and disseminating the knowledge to enforcement agencies 2. These organizations and programs have served a critical role for the us by "facilitating the flow of data between law enforcement agencies and victims" 10.

Conclusion

The guideline was evaluated using OSSTMM proposed by Pete Herzog, with the event of the economic Training Management System (ITMS) Web application as a case study. This study has successfully applied the OWASP guideline to the ITMS Web application. The results of all criteria that were evaluated indicated that OWASP contributed significantly to developing a secured Web application a minimum of in reducing the amount of security vulnerabilities, especially for Web-based university applications. Overall, taking under consideration security doesn't make web design more complicated; it should be one among many natural elements of web design nowadays. it's not hard to think about if it's included within the process of web design right from the start . Incomplete development processes leave the applications in danger , regardless of how structured the company's development process could also be . to realize a greater level of application security, mature development practices that focus specifically on Web application security got to be implemented.

References

1. Andrew Jaquith, Frank Heidt, & Chris Wysopal, (2003). Security Evaluation: Microsoft Windows Server 2003
2. with .NET Framework and IBM WebSphere.
3. Andrew Jaquith. (2002). The Security of Applications: Not All Are Created Equal

4. Boiler. (2003). Hacking Techniques: Issue#2 – Boucing Attacks.
5. David Endler, Brute-Force Exploitation of Web Application Session IDs. Retrieved from http://www.cgisecurity.com /lib/SessionIDs.pdf
6. Guptha1K. G., V Vuna, V. S., GOPAIAH, J., Goud, K. R. M., & Anuradha, K. (2015). Challenging Security Issues Using BIG Data on Mobile Application Development. *IJRDO - Journal of Computer Science Engineering (ISSN: 2456-1843)*, *1*(7), 01-11. Retrieved from https://www.ijrdo.org/index.php/cse/article /view/904
7. Ed Skoudis. (2002). Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses. Upper Sadle River, NJ: Prentice-Hall PTR.
8. Grizalis, Stefanos & Spinellis, Diomidis. (1997). Addressing Threats and Security Issues in World Wide Web Technology. In Proceeding CMS '97 3rd IFIP TC6/TC11,33-46. IFIP, Chapman & Hall.
9. Mark Curphey. (2004). The OWASP Testing Project, (2004, December).
10. Mark Curphey. (2004). The Ten Most Critical Web Application Security Vulnerabilities.
11. Mark Curphey. (2007). SpoC 007 - The OWASP Web Security Certification Framework.
12. OWASP Top Ten Web Application Vulnerabilities Version 1.0. (2003). Retrieved from OWASP Top 10 2004. (2004).
13. Shynlie Simmons, Hacking Techniques: Web Application Security. (2005, November) East Caroline University.
14. Stuart McClure, Joel Scambray, & George Kurtz, (2001). Hacking Exposed: Network Security Secrets and Solutions, Third Edition (3). : Osborn/McGraw Hill.

AUTHOR PROFILE

BANGI GURU PAVAN as Pursuing Master of Computer Applications from Sri Venkateswara University. Thupati in the year of 2017- 2020. Research interest in the field of Computer Science in the area of Internet Security,Deep Learning .

Prof. Dr. M.Padmavathamma has working as Professor In Department of Computer Science, S.V.University, Tirupati, AP. India. She has vast experience of 26 years in teaching. She has guided 10 PhD's, 12 M.Phils and published 53 articles in International/National Journals. She has attended and chaired many International conferences conducted by various International organizations at various places around the world. Currently she is director of projects funded by UGC, DST India. Her Areas of interest are Network Security, Cloud computing and Data Mining.