Web Applications Security Risk Quantification Based on Review of Past Six OWASP Top-10

Amjad Zareen[1], Dr Mansoor Ahmed Khan[2]

[1]MS Info Sec, Institute of Avionics and Aeronautics
Air University Islamabad, Pakistan
amjad.zareen@yahoo.com
[2]Institute of Avionics and Aeronautics, Air University
Islamabad, Pakistan
mansoorkhan75@gmail.com

Abstract. Since year 2003, OWASP has remained prominent in cyber security paradigm by continually publishing Top-10 Web Application Security risks. In this paper, Web Application Security Risk quantification has been produced and visualized. Review of past six OWASP Top-10 has been presented in such a manner which reveals evolution and helps to induce relationshep among risks. Transformation of risks demonstrated that web applications attacks such as Injection, Cross Site Scripting (XSS) and Broken Authentication which were present about two decades ago still prevail in present day scenarios. Data visualization using Python has been demonstrated while developing a keywords finding script from any research text as part of this research.

Keywords: OWASP Top-10, Web Applications, Access, Security, Control, Injection, Risks, Vulnerablities.

1 OWASP Top-10

The Open Web Application Security Project known as OWASP is an open community dedicated to enable organizations to develop, purchase and maintain web applications that can be trusted [1]. OWASP acts as great resource for finding web application security development libraries, penetration testing tools, standards and books. OWASP is kind of organization that claims to be free from commercial pressures and provides impartial, useful and cost-effective information about web applications security. Vulnerable web applications and security issues continue to undermine various sectors in electronic industries. The goal of OWASP Top-10 project is to raise awareness about web applications security by identifying most critical risks being faced by the organizations [2]. The OWASP is referred by many security standards, books, tools and organizations. About 99% of web applications that were tested in 2012 have reveled one or more serious security

10.11228/dpsc.02.01.001

vulnerabilities [3]. On average, the number of security issues affecting each web application increased from approximately 12.5 in 2010 to 13.5 in 2011 before declining to 12.1 in 2012 [4]. Study of 2018 cyber threats reveled that; three most common attacks on websites: SQL Injection, Path Traversal and Cross-Site Scripting (XSS) have remained the same for many years [5]. Web applications vulnerability statistics of year 2018 further divulge that, on average, each web application contained 33 vulnerabilities, out of which 6 were of high severity. In 19 percent of tested web applications, vulnerabilities allow an attacker to take control of the application and server OS [5]. The number of critical vulnerabilities per web application grew by 3 times compared to 2017. With such a high number of vulnerabilities per web application, it is no wonder that, web application attacks are focal point for hackers now a day.

1.1 History and Methodology

The OWASP Top-10 was first released in 2003. OWASP 2004 Top-10 list represented the combined wisdom of OWASP experts. Then it was released again with minor updates in 2007. Till 2007, the web applications vulnerabilities were listed based on their prevalence. OWASP 2007 Top-10 methodology extracted Top-10 web application security issues based on vulnerability trends. Afterwards, OWASP risk ranking methodology devised a formula: Risk = Likelihood * Impact [6]. The OWASP 2010 version reorganized and prioritized risks not just by occurrences. Risk ranking methodology used in year 2010 included three Likelihood factors: prevalence, detectability, ease of exploit and one Impact factor: technical impact. Prevalence statistics from number of different organizations were obtained and their average data produced Top-10 likelihood list by prevalence. This was then combined with likelihood factors: detectability and ease of exploit in order to calculate likelihood rating for each weakness. The result was then multiplied by estimated average technical impact to come up with an overall risk ranking. OWASP Top-10 for 2013 marked the project's tenth anniversary and followed similar approach as of year 2010. OWASP Top-10 project felt accelerated changes over the recent past years while categorizing Top-10 for year 2017. OWASP Top-10 for year 2017 was re-factored and methodology was fully revamped. They utilized a new data call process, worked with the community, reordered risks, rewritten each risk from the ground up and added references for web applications development frameworks and commonly used programming languages and the available time to take rectification actions. The simplified description of latest web applications risks described in OWASP Top-10 and their transformation has been explained in later part of this paper.

1.2 Comparison and Relationship

The threat landscape of web applications has continually evolved during the past sixteen years with the advent of new attack techniques and technologies. Inevitably, the web applications developers need to remain cognizant of specific risks and mitigation techniques to ensure business continuity with acceptable level

of cyber protection. It is not surprising that OWASP Top-10 list has evolved radically over time in terms of rankings. Based on the OWASP Top-10 review and comparison of six releases 2003, 2004, 2007, 2010, 2013 and 2017 a relationship in tabular form has been produced in a manner which can help finding evolution process among different web applications vulnerabilities with a careful glance [Table-1]. While looking at the relationship among risks and vulnerabilities presented, it can be easily recognized that the vulnerabilities and associated risks such as Injection, Cross Site Scripting (XSS) and Broken Authentication present about two decades earlier still remain prevailing in web applications.

2 Risks That Have Not Transformed

2.1 Injection

Injection attacks is one of the most damaging and high priority vulnerability across the web applications [7]. Injections attacks can result: data destruction, planting of malicious data or code and sensitive information leakage. Injection vulnerabilities such as SQL, OS or LDAP occurs when un-trusted data is sent to the interpreter as part of a legtimate command or query. The attacker's crafted data can trick the interpreter into executing unintended commands. Same can result access to unauthorized data or even destroy valuable records. Injection attacks have dominated the top of web application vulnerability lists for the past decade. Injection vulnerabilities are applicable to frequently accessing relational databases by web applications using SQL commands. Web applications usage may involve operations like searching, registration, online payments and logins etc. which can be used as source for injecting malicious input by attackers. Injection attacks can be conceded simply by using a browser, as most of the time port 80 or 443 are not blocked by firewalls for serving legitimate HTTP and HTTPS requests. Careful crafting and injection of inputs with the help of error analysis can lead the attacker to even take complete control of backend Server / Database / OS. Injection can be of different types. Command Injection type refer an attack where user becomes capable of injecting code into a command line. Unchecked File Uploads become dangerous when user is allowed to upload all kind of files including executable. Code Injection becomes possible where user can directly inject executable code of choice.

2.2 Broken Authentication and Session Management

Broken Authentication and Session management vulnerability usually occur due to non-standardized implementation of web application functions related to authentication and session management [8]. Due to this vulnerability, an application inappropriately allows attackers to exclaim themselves as valid user or an unauthenticated user can act as authorized user.

Table-1: Relationship among OWASP Top-10 past Six Releases

Rating & Risk Score	Year 2003	Year 2004	Year 2007	Year 2010	Year 2013	Year 2017
A1-100	Un-validated parameters	Un-validated Input	Cross Site Scripting (XSS)	Injection	Injection	Injection
A2-90	Broken Access Control	Broken Access Control	Injection Flaws	Cross-Site Scripting (XSS)	Broken Authentication and Session Manage-ment	Broken Authentication
A3-80	Broken Account and Ses-sion Management	Broken Authentication and Session Manage-ment	Malicious File Execution (NEW)	Broken Authentication and Session Management	Cross-Site Scripting (XSS)	Sensitive Data Exposure
A4-70	Cross Site Scripting (XSS)	Cross Site Scripting (XSS)	Insecure Direct Object Reference	Insecure Direct Object References	Insecure Direct Object References	XML External Entities (XXE)
A5-60	Buffer Overflows	Buffer Overflows	Cross Site Request Forgery -CSRF (NEW)	Cross-Site Request Forgery (CSRF)	Security Misconfiguration	Broken Access Control
A6-50	Command Injection Flaws	Injection Flaws	Information Leakage and Improper Error Handling	Security Misconfiguration	Sensitive Data Exposure	Security Misconfiguration
A7-40	Error Handling Problems	Improper Error Han-dling	Broken Authentication and Session Management	Insecure Cryptographic Storage	Missing Function Level Access Control	Cross-Site Scripting (XSS)
A8-30	Insecure Use of Cryptog-raphy	Insecure Storage	Insecure Cryptographic Storage	Failure to Restrict URL Access	Cross-Site Request For-gery (CSRF)	Insecure Deserialization
A9-20	Remote Administration Flaws	Denial of Service	Insecure Communications (NEW)	Insufficient Transport Layer Protection	Using Components with Known Vulnerabilities	Using Components with Known Vulnerabilities
A10-10	Web and Application Server Misconfiguration	Insecure Configura-tion Management	Failure to Restrict URL Access	Un-validated Redirects and Forwards (NEW)	Un-validated Redirects and Forwards	Insufficient Logging & Monitoring (NEW)

A session hijacking can also occur where a hacker takes control of a user session after successfully obtaining or generating an authentication session ID. This can be achieved by using captured, brute-force or reverse-engineering of session IDs to seize control of a legitimate user's Web application session while that session is in progress.

2.3 Cross Site Scripting (XSS)

Cross Site Scripting (XSS) vulnerability occurs whenever an application takes data that originates from a user or program and sends it to the browser without validating or properly encoding [9]. XSS allows hackers to execute scripts in the victim's browser, which can hijack user sessions, deface web site, redirect the user to malicious site or conduct phishing attacks. Cross Site Scripting (XSS) was the most frequently found vulnerability in apps tested in 2012 [3].

2.4 Cross Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of attacker. CSRF can be as powerful as the web application that is being attacked. Cross-Site Request Forgery (CSRF) was found in only 5% of applications during year 2017, as many web development frameworks now a days include CSRF defenses [10].

2.5 Un-Validated Redirects and Forwards

Web applications frequently redirect and forward users to other web pages and websites and use un-trusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages [11].

2.6 Security Misconfiguration

Security misconfiguration is commonly found issue in OWASP Top-10. This is result of: insecure or use of default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers and verbose error messages containing sensitive information [10]. All operating systems, development frameworks, libraries and web applications need secure configurations with proper error handling messages and updated security patch in order to safeguard against this risk.

3 Risks That Have Evolved

3.1 Broken Access Control

This results when restrictions on what authenticated users are allowed to do are not properly enforced [10]. Attackers can exploit these flaws to access other users' accounts, view sensitive files or use unauthorized functions. Example of such attack scenario could be; when an authenticated user of online banking while performing own transaction may get access to other user accounts. Access control also known as authorization mechanism, is how a web application grants access to content and functions to legitimate users and deny others. These checks are performed after authentication and govern what 'authorized' users are allowed to do. Access control sounds like a simple problem but is difficult to implement correctly and require due diligence and care. A web application's access control model is closely tied to the content and functions that the web application provides. In addition, the users may fall into a number of groups or roles with different abilities or privileges [12]. Developers frequently underestimate the difficulty of implementing a reliable access control. Many of access control schemes were not designed deliberately, but have evolved along with the web applications. In such cases, access control rules are inserted in various locations all over the source code. As the application reaches the deployment stage, the ad-hoc collection of rules becomes so unwieldy that it transforms almost impossible to understand. Many of such flawed access control schemes are not difficult to discover by hackers. In addition to viewing unauthorized content or performing illegitimate transactions, an attacker might be able to change or delete content, perform unauthorized functions or even take over complete web administration. In 2004 OWASP Top-10 Remote Administration Flaws merged Broken Access Control category as a special case of that category.

Remote Administration Flaws. One specific type of access control problem is administrative interfaces that allow web application administrators to manage their applications over the Internet or Intranet [13]. Such features are frequently required to manage users, data and content for the web applications. In many occasions, sites support a variety of administrative role to allow finer granularity of site administration. Due to their power, these interfaces are frequently prime targets for attack by both outsiders and insiders.

Insecure Direct Object Reference. A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as file, directory or database [14]. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data. It appeared first time in OWASP year 2007 Top-10 when Broken Access Control present in 2003 / 2004 divided into Insecure Direct Object Reference and Failure to Restrict URL Access. Insecure Direct Object Reference remained present in OWASP Top-10 for year 2010 and 2013.

Failure to Restrict URL Access. Frequently, the only protection for a URL is that links to that page are not shown to unauthorized users [15]. However, a motivated, skilled or just plain lucky attacker may be able to find and access hidden pages, invoke functions, Web Services or APIs and able to manipulate data access by crafting URLs. Security by obscurity is not sufficient to protect sensitive functions and data in an application. Access control checks must be performed before a request to a sensitive function, Web Service or APIs is granted, which ensures that the user is authorized to access that functionality. In 2013 OWASP Top-10 Failure to Restrict URL Access expanded into Missing Function Level Access Control.

Missing Function Level Access Control. Most web applications verify function level access rights before making that functionality visible in the User Interface. However, applications need to perform similar access control check on the server side when each function is accessed [14]. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization. Insecure Direct Object References and Missing Function Level Access Control from OWASP Top 10 year 2013 were merged into Broken Access Control in year 2017.

3.2 Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application / web / database server, underlying operating system and infrastructure cloud platforms [11]. Secure configuration settings needs to be defined, implemented and maintained. Additionally, firmware and software need to be kept up to date with latest and tested security patches. Security Misconfiguration appeared on 2010 OWASP Top-10 and was further divided in 2013 OWASP Top-10 into Security Misconfiguration and Using Components with Known Vulnerabilities. When Security Misconfiguration appeared in 2010 OWASP Top-10; it covered Malicious File Execution and Information Leakage / Error Handling listing of 2007 OWASP Top-10.

Information Leakage and Error Handling. Web applications can unintentionally leak information about their configuration, internal workings, structure and methods or may violate privacy through a variety of misconfigured error handling [16]. Attackers can use this weakness in order to gather information for stealing sensitive data or conducting serious attacks.

Malicious File Execution. Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, which can result in devastating attacks, such as total server compromise [17]. Malicious file execution attacks affect PHP, XML and any framework where filenames or files from users are accepted.

3.3 Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs and authentication credentials such as passwords [11]. Attackers may steal or modify weakly protected data to conduct online fraud, identity theft or other cybercrimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with browsers. When Sensitive Data Exposure appeared in 2013 OWASP Top-10; it covered Insecure Cryptographic Storage and Insufficient Transport Layer Protection listings of 2010 OWASP Top-10. Insufficient Transport Layer Protection was previously named as Insecure Communications in 2007 OWASP Top-10. Insecure Cryptographic Storage of 2007 OWASP Top-10 evolved from Insecure Storage of 2004 OWASP Top-10 which itself evolved from 2003 OWASP Top-10 Insecure Use of Cryptography.

Insecure Cryptographic Storage. Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing [17]. Attackers may steal or modify weakly protected data to conduct identity theft, credit card fraud or other crimes. Encryption algorithms with correct implementation and assured key management are desired.

Insufficient Transport Layer Protection. Applications frequently fail to authenticate, encrypt and protect the confidentiality and integrity of sensitive network traffic while in-transit [18]. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not implement them correctly.

Insecure Communications. Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications [18]. Man in the Middle attack protection mechanism are desirable for such scenarios.

Insecure Storage. Most web applications need to store sensitive information, either in a database or on in a file system. The information might be passwords, credit card numbers, account records or any other proprietary data [18]. Frequently, symmetric encryption techniques are used to protect sensitive information. While encryption has become relatively easy to implement and use, developers still frequently make mistakes while integrating it into a web applications. Developers may overestimate the protection gained by using encryption and not be as careful in securing other aspects.

3.4 Using Components with Known Vulnerabilities

Components, such as software libraries, frameworks, and other modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or system takeover. Applications using components with known vulnerabilities may undermine application defenses and enable

a range of possible attacks and impacts. Security evaluation of such components is prerequisite prior to using them in a sensitive development environment.

3.5 XML External Entities (XXE)

This is a new category in OWASP Top 10 for year 2017 primarily supported by (source code analysis security testing tools (SAST) data sets. OWASP Top-10 for year 2017 asked the community to provide insight into two forward looking weakness categories. After over 500 peer submissions and removing issues that were already supported by data such as Sensitive Data Exposure and XXE, the two new issues were listed as XML External Entities (XXE) and Insecure Deserialization. Un-validated Redirects and Forwards, while found in approximately 8% of web applications, was overall edged out by XXE in OWASP Top 10 of year 2017 [10].

3.6 Insecure Deserialization

This permits remote code execution or sensitive object manipulation on affected platforms. It occurs when untrusted data is used to abuse the logic of an application, inflict a Denial of Service (DoS) attack or even execute arbitrary code upon it being deserialized. Serialization refers to a process of converting an object into a format which can be persevered to disk. Examples include: save to a file or a datastore, sent through streams or sent over a network. The format in which an object is serialized, can either be binary or structured text like XML, JSON etc. Deserialization is the opposite of serialization and involves transforming serialized data back into an object. Safe deserialization of objects must be considered as normal practice in software development [19]. The trouble can start when deserializing is done on untrusted inputs.

3.7 Insufficient Logging & Monitoring

Lack of this can significantly delay malicious activity and breach detection, incident response and digital forensics investigations. Insufficient Logging & Monitoring risk differs from other risks. While it cannot lead to a direct intrusion, existence of this results in failure to detect the intrusion in a timely manner. A failure that can cost significant in monetary as well as technical terms.

4 Risk Analysis and Quantification

Many different approaches exist for carrying out information security risk analysis for web applications. The OWASP approach is customized for web applications security and is based on simplest formula as follows:-

Risk = Likelihood * Impact [6]

The above formula has been applied for the purpose of quantification and graphical depiction on Top-3 risks which consistently remained present in Web Applications. Likelihood for a risk has been labelled numerically on scale of 10 to 100. Risk having A1 rating in OWASP Top-10 has been assigned a score of 100 while A10 rating is quantified as 10. Impact factor has been assumed constant for simplification. Most of the times, impacts such as reputation loss, financial loss, data loss etc. are truly hard to quantify in real sense. Based on the risk analysis quantification; Injection attack has been found most prevalent and ranked as most dangerous as depicted in Figure-1.

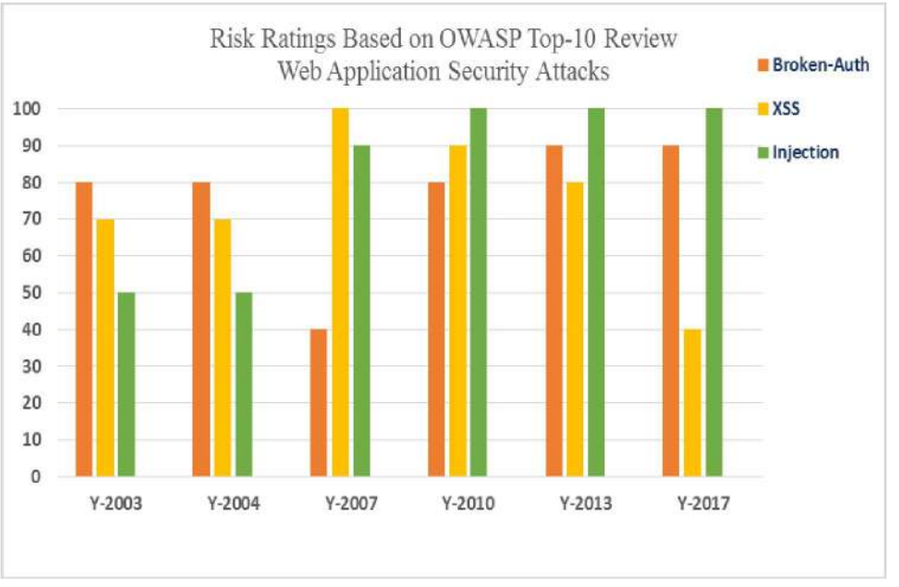

Figure 1. Risk Ratings Based on past six OWASP Top-10 Review

5 Keywords Discovery from Research

Keywords are words that capture the spirit of any research paper. Keywords make research paper searchable and ensure to attract citations. It is significant to contain the most relevant keywords that help other authors. Conference and Journal ask for varied number of keywords. Research text data parsing and its visualization using Python script has been demonstrated for finding variable number of keywords from any research text. The results obtained from this paper are appended in Figure-2.

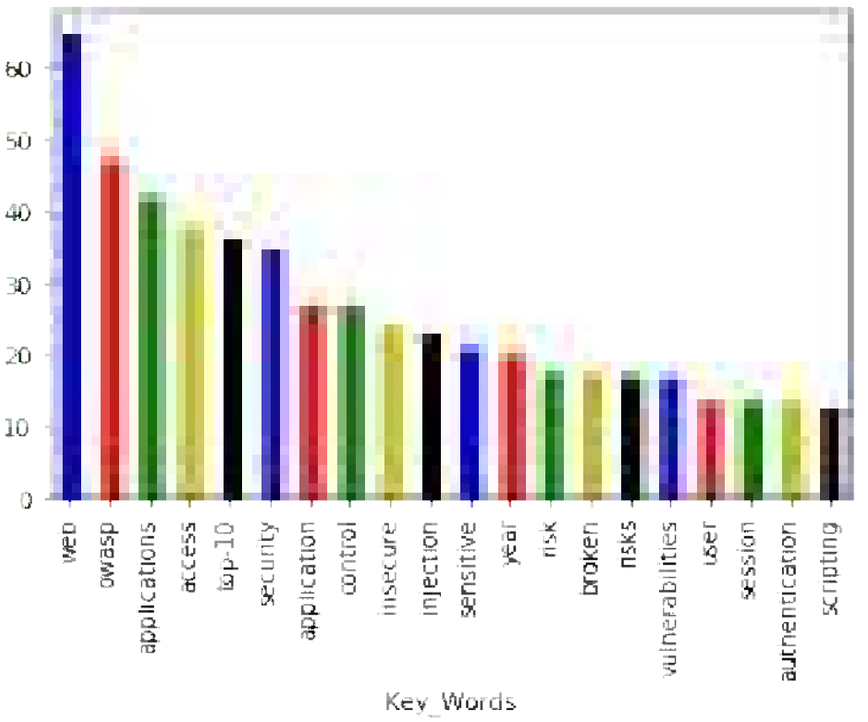

Figure 2. Research Keywords Visualization using Python

6 Conclusion

Web applications risks including Injection, Cross Site Scripting (XSS) and Broken Authentication remained consistently present in OWASP Top-10. Injection has been quantified as the top most risk based on review of past six OWASP Top-10. Modern day attackers are targeting web applications security vulnerabilities for gaining access to sensitive data. Organizations need to show vigour with greater extents to protect web applications. As more organizations are touching towards cloud transformation, web applications security is becoming more crucial. Risks and associated attacks presented in this research predominantly relate to the software development domain and demand serious attention. These problems require secure development, secure implementation and continuous monitoring practices for attaining sustainable cyber security posture. Keeping the same in view a comprehensive model comprising people, technology and processes is deemed essential to address consistently present web applications security risks.

12

References

1. Open Web Application Security Project – OWASP Foundation [Online] Available: https://www.owasp.org [Accessed 10 January, 2020]

2. OWASP Top-10 Project [Online] Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project [Accessed 11 January, 2020]

3. Application Vulnerability Trends Report : 2013 [Online] Available: http://expo-itsecurity.ru/upload/iblock/ffb/cenzic-application-vulnerability-trends-report-2013.pdf [Accessed 11 October, 2018]

4. Web Application Vulnerability Statistics 2013 by Eyal Estrin [Online] Available: http://www.contextis.com/files/Web_Application_Vulnerability_Statistics_-_June_2013.pdf [Accessed 24 December, 2018]

5. Attacks on Web Applications: 2018 in Review [Online] Available: https://www.ptsecurity.com/ww-en/analytics/web-application-vulnerabilities-statistics-2019/ [Accessed 11 January, 2020]

6. OWASP Risk Rating Methodology [Online] Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology [Accessed 10 January, 2020]

7. SQL Injection Attacks on Web Applications by Chandershekhar Sharma ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 3, March 2014 [Online] Available: https://www.researchgate.net/publication/320076267_SQL_Injection_Attacks_on_Web_Applications [Accessed 11 January, 2020]

8. The Importance of Broken Authentication and Session Management to Application Security by Anita D'Amico [Online] Available: https://codedx.com/blog/broken-authentication-and-session-management/ [Accessed 10 January, 2020]

9. Microsoft document for Preventing Cross-Site Scripting (XSS) By Rick Anderson [Online] Available: https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting [Accessed 11 January, 2020]

10. OWASP Top 10-2017 Release Notes [Online] Available: https://www.owasp.org/index.php/Top_10-2017_Release_Notes [Accessed 26 December, 2019]

11. OWASP Top 10-2013 [Online] Available:
https://www.owasp.org/index.php/Top_10_2013-Top_10
[Accessed 24 December, 2019]

12. Broken Access Control [Online] Available:
https://www.owasp.org/index.php/Broken_Access_Control
[Accessed 18 December, 2019]

13. A2 2004 Broken Access Control [Online] Available:
https://www.owasp.org/index.php/A2_2004_Broken_Access_Control
[Accessed 10 December, 2019]

14. Must Known Web Security Risks for Developers By Sanjeev Murthy
[Online] Available:
https://www.pluralsight.com/guides/must-known-web-security-risks-for-developers
[Accessed 18 November, 2019]

15. Top 10 2007-Failure to Restrict URL Access [Online] Available:
https://www.owasp.org/index.php/Top_10_2007-Failure_to_Restrict_URL_Access
[Accessed 10 December, 2019]

16. OWASP Top-10 2007 - Information Leakage and Improper Error Handling
[Online] Available: https://www.owasp.org/index.php/Top_10_2007-
Information_Leakage_and_Improper_Error_Handling [Accessed 15 October, 2019]

17. Common Vulnerabilities in Web Applications [Online] Available:
https://www.infosec.gov.hk/english/business/other_sywa_1.html
[Accessed 10 December, 2019]

18. Next Generation Threat Prevention, WAF, OWASP Top-10|Tech Brief
[Online] Available: https://www.checkpoint.com/downloads/products/cp-ngtp-waf-owasp-top-10-comparison-tech-brief.pdf [Accessed 12 January, 2020]

19. Technology Editorial- What is Insecure Deserialization? [Online] Available:
https://www.acunetix.com/blog/articles/what-is-insecure-deserialization/
[Accessed 26 December, 2019]