Cross-Site Scripting Attacks

Security, Privacy, and Trust in Mobile Communications

About the Series

Similar to computers, the mobile landscape is also facing various security and privacy related threats. Increasing demand of sophisticated handheld mobile devices including smartphones, tablets, and so forth, is making them an attractive target of security threats. Since these devices store confidential data of the end users, and exploitation of vulnerabilities of the underlying technologies can create a havoc on massive scale, it becomes inevitable to need to understand and address the threats associated with them and to analyze the level of trust that can be established for mobile communication scenarios.

This series will present emerging aspects of the mobile communication landscape, and focuses on the security, privacy, and trust issues in mobile communication based applications. It brings state-of-the-art subject matter for dealing with the issues associated with mobile and wireless networks. This series is targeted for researchers, students, academicians, and business professions in the field.

If you're interested in submitting a proposal for a book to be included in the series, please email Gabriella.Williams@tandf.co.uk

Series Editors:
Brij B. Gupta

Computer and Cyber Security
Principles, Algorithm, Applications, and Perspectives
Brij B. Gupta

Smart Card Security
Applications, Attacks, and Countermeasures
B.B. Gupta, Megha Quamara

Cross-Site Scripting Attacks
Classification, Attack and Countermeasures
B.B. Gupta and Pooja Chaudhary

For more information about this series please visit: https://www.crcpress.com/Security-Privacy-and-Trust-in-Mobile-Communications/book-series/SPTMOBILE

Cross-Site Scripting Attacks

Classification, Attack, and Countermeasures

B. B. Gupta and Pooja Chaudhary

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

First edition published 2020
by CRC Press
6000 Broken Sound Parkway NW, Suite 300,
Boca Raton, FL 33487-2742

International Standard Book Number-13: 978-0-367-36770-1 (hbk)

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Dedicated to my parents and family for their constant support during the course of this book

—B. B. Gupta

Dedicated to my parents, siblings, and my mentor for their guidance and motivation throughout the journey of completion of this book.

—Pooja Chaudhary

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

List of Figures

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

List of Tables

Preface

With the advancements of web development technologies and innovations like internet of things (IoT), internet services are accessible even in remote areas smoothly. The proliferation of internet triggers abrupt escalation in the utilization of the social network. These networks have interwoven into the daily routine lives of people in the form of virtual platforms, which facilitate ease of communication. Users connect with new loved ones and re-establish the lost connections irrespective of the geographical location. The data shared by social actors is not only beneficial to the different organizations to analyze and maintain a strong customer relationship but also fascinates the attacker to utilize it for his/her selfish motive. The highly concentrated topology of the social networks, use of advanced features like AJAX and Java Script, and a strong trust relationship among the social actors are the key characteristics of the social sites being focused by the attacker. These sites have become the hotbed of malicious files affecting the privacy of social media users.

Cross-Site Scripting attack comes under the umbrella of code injection-based vulnerability and is ranked at no. 3 among all the web application-based vulnerabilities. This has contaminated almost 80 percent of the popular web applications over the internet today. *Cross-Site Scripting Attacks: Classification, Attack, and Countermeasures* provides a detailed study of the XSS attack. This book primarily focuses on the classification of the

key contribution of the research work accomplished in the area of XSS. Moreover, this book mainly addresses a novel mitigation technique to protect against the XSS attack. It also puts light on the open challenges and future research recommendations for further progression in the XSS domain.

Specifically, the chapters contained in this book are summarized as follows:

Chapter 1: Security Flaws in Web Applications—This chapter primarily focuses on the various types of security issues and web-based vulnerabilities exploited by the data snooper to launch various types of attacks.

Chapter 2: Security Challenges in Social Networking: Taxonomy and Statistics—This chapter provides a classification of the different types of security attacks specific to the social platforms. It also highlights statistics depicting the usage of social media among internet users, harmful effects of using it on the young generation, and so on.

Chapter 3: Fundamentals of Cross-Site Scripting (XSS) Attack—This chapter provides deep insight into Cross-Site Scripting attack, its classification, incidences of the XSS attack, and various consequences of the XSS attack. Furthermore, it describes existing defensive methodologies against the XSS attack with their strengths and weaknesses. It also provides a comparative study of all these techniques.

Chapter 4: Clustering and Context-Based Sanitization Mechanism for Defending against XSS Attack—This chapter discusses what are the various challenges that exist in the existing state-of-the-art techniques. Later on, it also elaborates an efficient and robust mechanism to thwart XSS attack on social network to overcome such challenges to some extent. It also discusses its strengths and limitations.

Chapter 5: Real-World XSS Worms and Handling Tools—This chapter discusses the types of XSS worms that can have a severe impact on the social actors. Moreover, it also describes the different types of tools that aid in detecting and mitigating the XSS attack from web applications.

Chapter 6: XSS Preventive Measures and General Practices—This chapter discusses the general methods and practices which can be applied at the development level of browsers or web applications or both, to safeguard against the XSS attack. It also sheds light on the path for future research through highlighting the existing issues in currently available solutions.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Acknowledgments

First of all, we would like to pay our gratitude to God by bowing our heads for lavishing on us with continuous blessings and enthusiasm for completing this book. Writing a book is not a work of an individual, but it is the outcome of the incessant support of our loved ones. This book is the result of inestimable hard work, continuous efforts, and assistance of loved ones. Therefore, we would like to express our gratefulness to each one of them who are linked with this book directly or indirectly, for their exquisite cooperation and creative ideas for meliorating the quality of this book. Along with this feeling, we would like to appreciate CRC Press, Taylor & Francis Group, staff for their assistance and persistent support. We are grateful, from the bottom of our hearts, to our family members for their absolute love and uncountable prayers. This experience is both internally challenging and rewarding. Therefore, again special thanks to all who helped us in making this happen.

November 2019
B. B. Gupta
Pooja Chaudhary

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Author Bio

B. B. Gupta received PhD degree from Indian Institute of Technology Roorkee, India, in the area of Information and Cyber Security. He published more than 200 research papers in International Journals and Conferences of high repute including IEEE, Elsevier, ACM, Springer, Wiley, Taylor & Francis, Inderscience, etc. He has visited several countries, i.e. Canada, Japan, USA, UK, Malaysia, Australia, Thailand, China, Hong Kong, Italy, Spain, etc. to present his research work. His biography was selected and published in the 30th Edition of *Marquis Who's Who in the World*, 2012. Dr. Gupta also received Young Faculty Research Fellowship award from Ministry of Electronics and Information Technology, Government of India, in 2018. He is also working as a principal investigator of various R&D projects. He is serving as Associate Editor of *IEEE Access*, IEEE TII, and Executive Editor of *IJITCA*, Inderscience. At present, Dr. Gupta is working as Assistant Professor in the Department of Computer Engineering, National Institute of Technology, Kurukshetra, India. His research interest includes Information security, Cyber Security, Mobile security, Cloud Computing, Web security, Intrusion detection, and Phishing.

Pooja Chaudhary is currently pursuing her PhD degree from National Institute of Technology (NIT), Kurukshetra, Haryana, India, in Information and Cyber Security area. She has

completed her Master of Technology (MTech) degree in the area of Cyber Security from National Institute of Technology (NIT), Kurukshetra, Haryana, India. She has received her BTech degree in Computer Science and Engineering from Bharat Institute of Technology, Meerut, India, affiliated to Uttar Pradesh Technical University. Her areas of interest include Online Social Network (OSN) security, big data analysis and security, database security and cyber security, and internet of things security. She has published a number of research papers with various reputed publishers, i.e. IEEE, Springer, Wiley, Inderscience, and so on.

CHAPTER 1

Security Flaws in Web Applications

The advancement in technology along with the digitalization of business drives us onto a new span of computing. Innumerable web applications have been designed embracing new and improved features. However, this progress leaves numerous web application vulnerabilities that are destabilizing the secure infrastructure of an organization. Therefore, this chapter concentrates on providing comprehensive details of the most prominent and dangerous vulnerabilities that are contaminating the digital world and affecting businesses worldwide. More elaborately, the authors have encapsulated the related statistics of critical vulnerabilities from reliable sources, giving insights into the security threats corresponding to different business domains. Finally, a comprehensive assessment of the vulnerabilities has been accomplished with respect to a method of rating identified risk paths.

1.1 WEB APPLICATION VULNERABILITIES

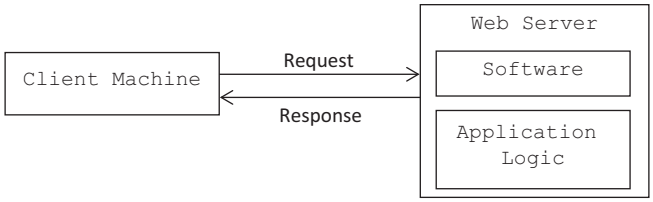
Over the past decade, the internet has not only evolved into a digital platform where people can search for anything, but has

also become the lifeline of many businesses. Digitalization led to rapid business invention. Web application lies at the core of most business including the government sector, manufacturing sector, finance sector, and many more [5, 6, 9]. This transformation of business to the digital space helps an organization bring its services at the edge, i.e. in the hands of the user. Consequently, the user can access these services anywhere, anytime, thereby spanning business boundaries. For most organizations, software applications solely are businesses like e-commerce business. Organizations disburse a huge amount of and extensive efforts to provide a good digital experience to their customers; however, only protected and safe applications can serve their purpose effectively. Yet developing software components without any vulnerability is still a dream. Instead of developing these software applications as a single isolated component, today, organizations use third-party components to develop applications through the integration of discrete components. Thereby, new hidden vulnerabilities exist and are being exploited at a faster rate, more than the rate of identification and developing patches to fix them by the organization [2, 20].

1.1.1 Fundamentals of Web Application Architecture

Web application builds upon multiple modules [19]. It consists of a web server, web browser, application information residing in the server, and the data store working at the backend that is accessed by the application. Complex web application may include many more modules; however, the basics remain the same.

- ***Web Server:*** It is a computer machine that executes web server software to respond to the user's request. It listens to port 80 (http) or port 443 (https). It basically hosts various web sites' information including HTML files, style sheets, and JavaScript documents. Example: Microsoft IIS web server [17], Apache Web server [1], etc.



- ***Web Browser:*** It is the computer application used to request web content. It is used to retrieve web pages on the World Wide Web (WWW) and displays it to the user. Example: Mozilla Firefox, Google Chrome, Safari, etc.
- ***Application Logic/Information:*** It is the program logic that helps in processing the user's request. Basically, it interacts with the request and interprets the parameters sent by the browser to achieve its objective. For instance, a PHP interpreter residing at the server side helps it to process PHP scripts at the server side itself.
- ***Back-End Data Store:*** It is the database which stores the information accessed by the application logic. It may be anything like file database, SQL commands database, etc. It is located on a different machine than the web server, connected through a network.

1.1.2 Background and Motivation

It was discovered by the Web Application Vulnerabilities Statistics report, in 2017, that of the total vulnerabilities reported, 17% were highly severe vulnerabilities, 69% were moderately severe, and 14% came under the category of low-severity vulnerabilities [21]. These vulnerabilities can cause major financial and technical impacts to the organization depending upon the range of severity level they lie in. Software applications may comprise of vulnerabilities of different severity levels. Figure 1.1 depicts the statistics of an application containing vulnerabilities corresponding to their severity levels. There was an increase in highly

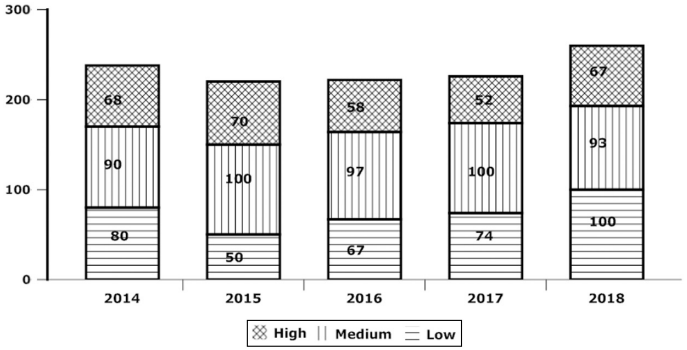


FIGURE 1.1 Percentage of web application as per vulnerability severity level.

severe vulnerabilities by 5% in 2018 as compared to 2017. Use of untrusted third-party components or use of outdated components may be the major cause for the exploitation of embedded vulnerabilities, for example, use of default configuration or use of older versions of software. Therefore, it is quite clear that more effort has to be put into either developing secure and effective software applications by incorporating secure coding practices in the development phase, or designing and deploying defensive mechanisms to detect these flaws.

The advancement of web design technologies is a great force in developing dynamic and more user-friendly applications. Moreover, the emergence of industry 4.0 and progression of the World Wide Web incorporated a wide range of technologies including client-side technology, server-side technology, and advanced protocols.

Use of technologies like HTML5, AJAX, and JavaScript makes applications more versatile in nature. Irrespective of the context, every organization depends on software applications for business expansion. These web applications are developed by using different programming platforms like PHP, Java, ASP.NET, and others. PHP is the most widely used technology for designing applications

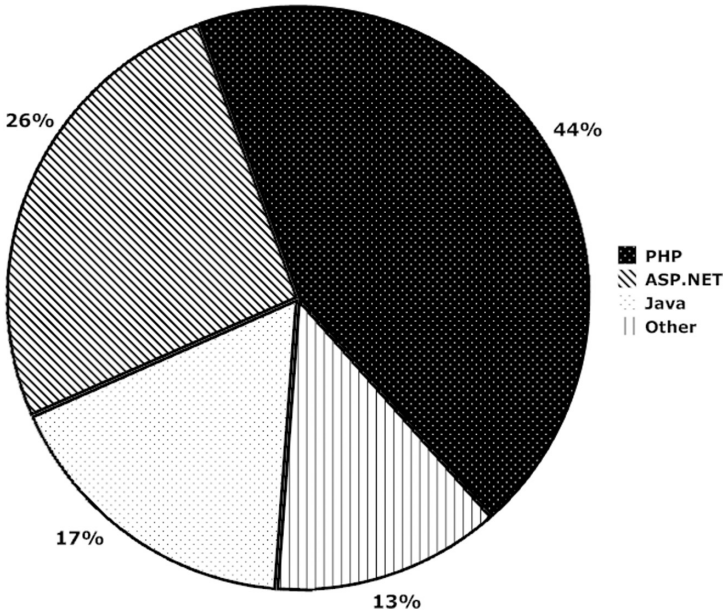


FIGURE 1.2 Percentage of web application developed using programming languages.

[16]. Figure 1.2 shows that almost 44% of web applications are designed using PHP as the base language, 26% are based on ASP.NET, and so on. Other category includes languages like Python, Ruby, etc. Also, it has been noted here that PHP and ASP.NET are the widely used technologies for web application development nowadays. Even though web application plays a crucial role in the extension of the business, these contain some hidden flaws that the attacker might exploit. These flaws may be categorized as high, medium, and low severity level depending upon their impact on the web application if the attacker exploits them. Figure 1.3 shows the average number of vulnerabilities corresponding to each severity level identified in each web application developed using one of the programming languages like PHP, ASP.NET, Java, and others [15].

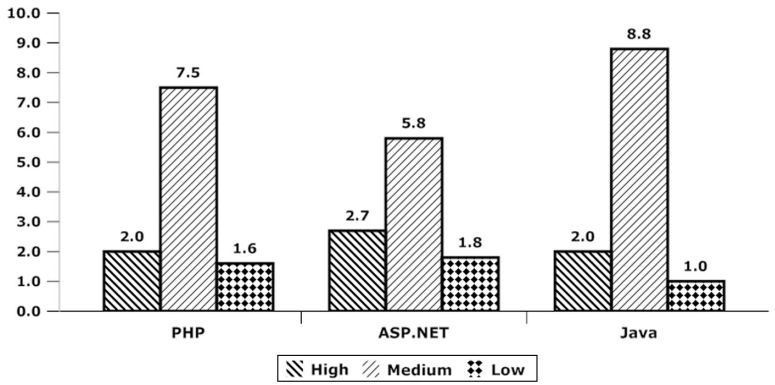


FIGURE 1.3 Average number of vulnerabilities in each web application as per severity level.

Development tools like PHP, ASP.NET, and Java are in trend for designing web applications for any organization including the government sector, finance, manufacturing, IT, mass media, and so on. Figure 1.4 reveals some statistics on the number of vulnerabilities detected in web applications developed using these tools and technologies over the years [3]. It is also noted here that there has been a continuous fall in the number of vulnerabilities found in web applications developed using PHP since 2016, meaning patches have been developed for mitigating vulnerabilities; however, complete eradication of vulnerabilities from applications is still a dream due to heterogeneity.

1.1.3 Related Statistics

There exist various vulnerabilities which are continuously tainting web applications belonging to every domain; however, a report by White Hat Security in 2017 [23] labels some of the frequently found vulnerabilities. Identification of these vulnerabilities depends on the type of assessment employed. To perform effective security assessment, organizations employ both static and dynamic testing in tandem.

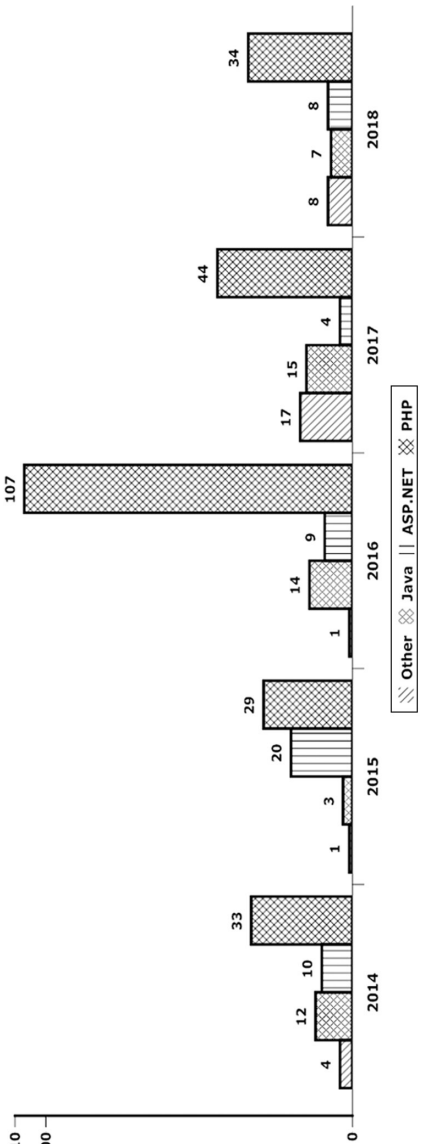


FIGURE 1.4 Vulnerabilities found in the latest developing technologies.

Static testing refers to analyzing the software application to identify any kind of security flaws during the development phase itself. It may be of high, medium, or low severity. Figure 1.5 reflects the major class of vulnerabilities found during static testing of the web applications. Unpatched library and application misconfiguration are the two most prevalent web application vulnerabilities because developers nowadays utilize the concept of modular programming where each module is reusable and easily accessible, but is less secure and uses default configuration as provided by the developer.

Recently, to discover more flaws, dynamic testing of the application has become popular. Dynamic testing of the web application is performed while the application is running in a real environment to detect those vulnerabilities which are unidentified during static testing. It is essential to perform this testing so that more and more vulnerabilities can be identified, thus yielding a more secure and robust application. Figure 1.6 shows major classes of the vulnerabilities which get identified in dynamic testing [18].

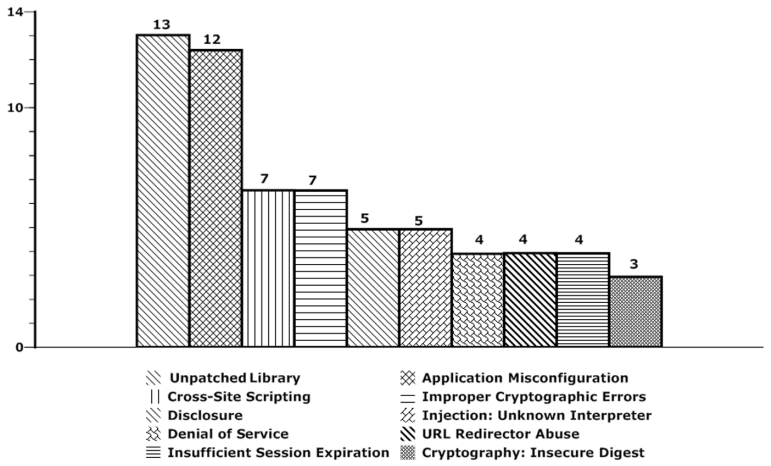


FIGURE 1.5 Vulnerabilities found during static testing (in %).

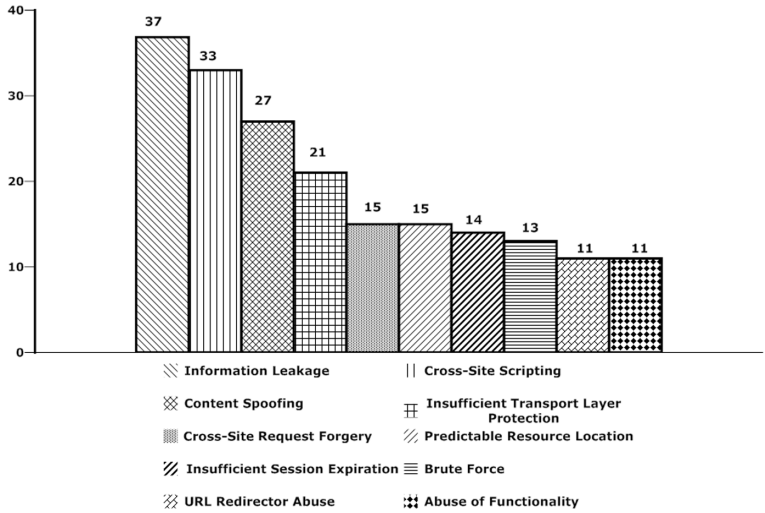


FIGURE 1.6 Vulnerabilities found during dynamic testing (in %).

Through comparing static testing and dynamic testing results, it is found that the prominent vulnerabilities of static testing are not a part of vulnerabilities found during dynamic testing. However, Cross-Site Scripting (XSS) [4] is the most dangerous vulnerability as it is part of both testing. This means that developers left some loopholes, making XSS pave its way in web applications. Consequently, mitigating XSS is of major concern and it is becoming the most dangerous flaw in web applications. Therefore, identification and mitigation of XSS vulnerability is an open research challenge [7, 8, 10–14].

For a long period of time, security personnel paid attention only to the development phase with the perception that they could recognize all the vulnerabilities that might be present in the application; however, it has been observed that few vulnerabilities are identified and fixed during the development phase. It raises major security concerns and yields abundant threats to the application when it is in the real environment, giving an open opportunity

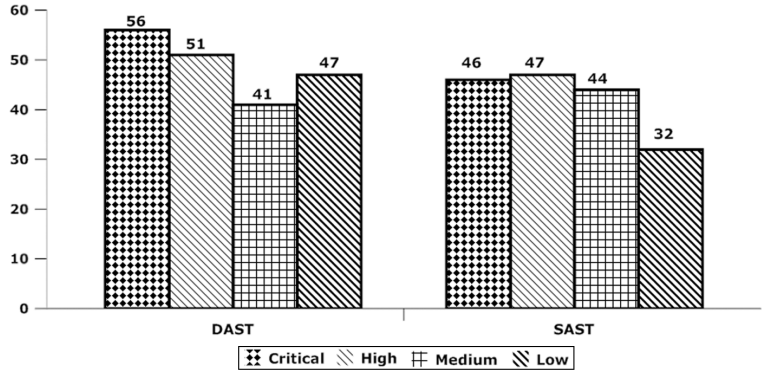


FIGURE 1.7 Vulnerabilities detection rate SAST vs. DAST (in %).

to the attacker to exploit the latent flaws. Figure 1.7 revealed that a major portion of the security error is found in dynamic testing as compared to static testing [23]. It is shown here that the percentage of the vulnerabilities identified and fixed during dynamic testing is large in comparison with static testing, whether they are of high-, critical-, or medium-severity level.

Rapid growth of more innovative and complex application development techniques induces complex applications and raises difficulty exponentially in identifying and resolving vulnerabilities. Insecure web applications are affecting every domain like e-commerce, manufacturing, IT, public sector, etc. As the risk imposed through the exploitation of latent vulnerabilities in web applications can vary from low to high, it is vital to resolve them earlier with accuracy. Another report divulged by the Open Web Application Security Project OWASP [18] highlights the most common top 10 vulnerabilities embedded in web applications belonging to almost every sector. Figure 1.8 lists out these top 10 vulnerabilities.

These vulnerabilities exist because of many reasons including insecure coding, use of modular programming without security testing of components, use of default configurations,

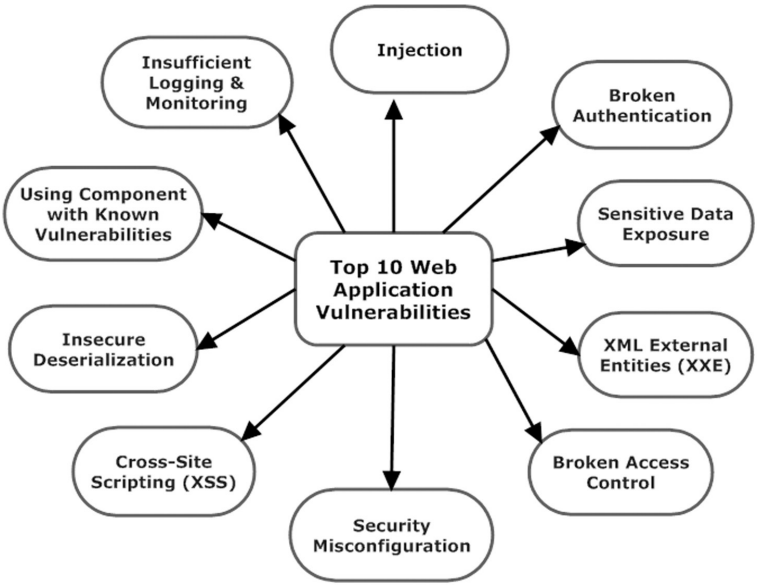


FIGURE 1.8 Top 10 web application vulnerabilities.

security negligence during the development phase, and many more. Therefore, OWASP provides information regarding the existing most dangerous vulnerabilities which aids developers, application designers, and organizations to remain updated about these vulnerabilities so that these can be found earlier, thereby reducing associated risk.

1.2 DIFFERENT DOMAIN-CENTRIC WEB APPLICATION VULNERABILITIES

With the development of web 2.0, there has been a surge of dynamic web applications in the digital world of the internet which allows users to interconnect with them by providing user-specific data. In today's modern era, web applications corresponding to each business have become their lifelines. Each enterprise offers its services to its customers via its web applications including the

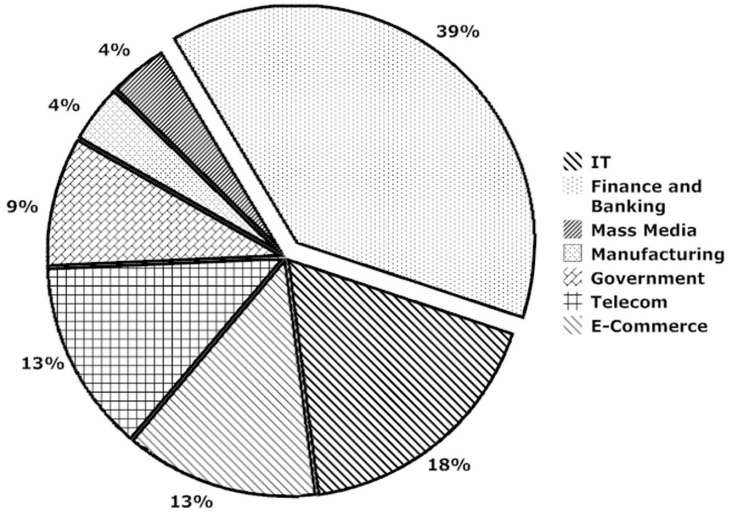


FIGURE 1.9 Percentage of web applications corresponding to different industries.

public sector, banking, e-commerce, IT sector, social media, and any other business. Figure 1.9 highlights a portion of the digital world occupied by different industries through their respective web applications [21].

These web applications pave the way for organizations to approach their customers by availing multiple online services. However, only secure applications can impart these services safely. In 2018, almost 83% of vulnerabilities were identified in web applications due to insecure coding. Because of technological advancements, web applications are being designed and delivered faster than ever before, affecting their security and attracting attackers to exploit latent vulnerabilities. Figure 1.10 shows that approximately 32% of web applications have been ranked as having a very poor level of security, giving rise to innumerable cyberattacks.

Despite incorporating security features while developing web applications, there are various hidden vulnerabilities that are

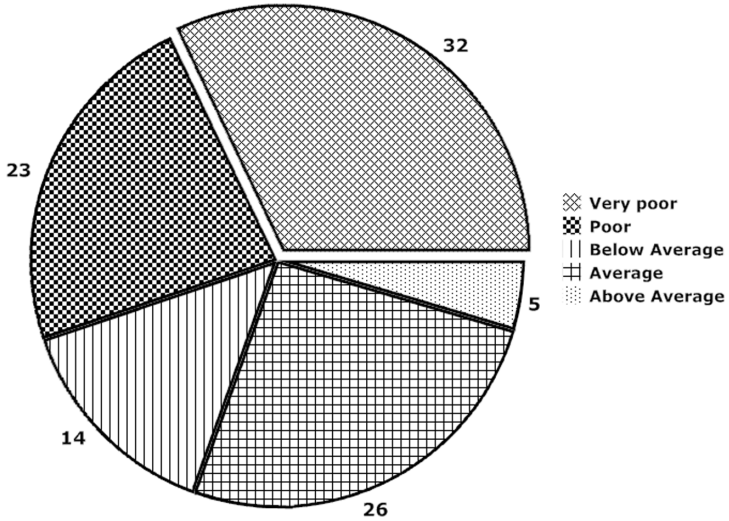


FIGURE 1.10 Percentage of web applications with security level.

embedded in them. There may be many reasons for the weak security level of web applications such as ignorance to secure coding, user unawareness, and default configuration, which help the attackers trigger new attacks. Figure 1.11 highlights glimpses of the average number of attacks that have been performed over different industries [22].

It has been identified in a report [22], in 2018, that the consequences of these attacks affect users of that particular industry. There are many web applications that process users' credentials, store personal information, and consequently lead to leakage of data. Figure 1.12 shows some of the consequences of attacks on web applications.

1.3 COMPREHENSIVE DETAIL OF MOST DANGEROUS VULNERABILITIES

This section offers a brief overview of the top 10 vulnerabilities unveiled by OWASP [18]. It provides information about different

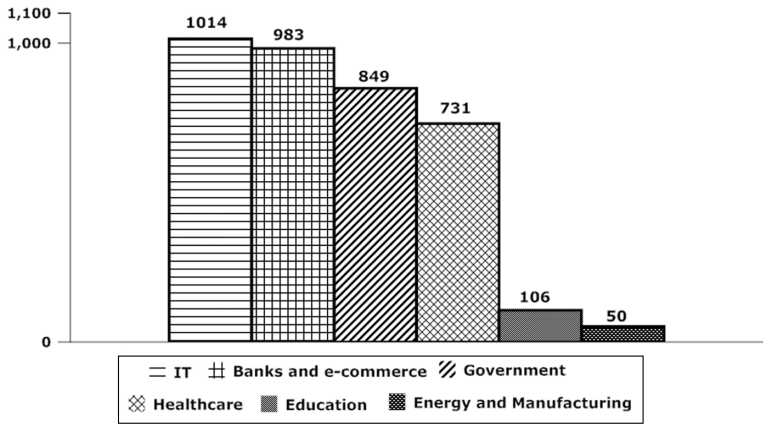


FIGURE 1.11 Average number of attacks on different industries.

paths exploited by an attacker to cause damage to an organization. As every sector including banking, government, e-commerce, financial, healthcare, social media, manufacturing, IT, and telecom make greater use of digital platforms to expand its business, all are prone to various types of vulnerabilities embedded in web

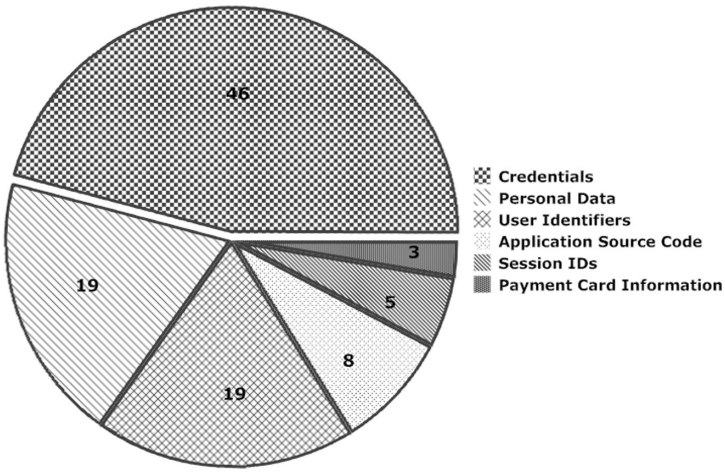


FIGURE 1.12 Consequences of attacks on users.

applications. Consequently, awareness of these flaws is indispensable while developing applications.

1.3.1 Overview of Web Application Vulnerabilities

In this module, a comprehensive description of the top 10 most dangerous vulnerabilities is illustrated. We have briefly explained each of the web application vulnerabilities by illuminating only the important factor behind it. Table 1.1 summarizes these vulnerabilities. Proper understanding of each of the vulnerabilities including its root cause and exploitation method is mandatory to come up with the solution to recognize and defend against these vulnerabilities. It would be better for any organization to recognize all the latent flaws in the web application earlier so that the associated risk level could be compensated for easily or may be completely exempted from it.

It is completely dependent on the awareness of the security personnel to deal with these vulnerabilities. Sometimes, it might not be easy to develop the defending solution even if you are familiar with these flaws. The next section illustrates how these vulnerabilities are exploited by the attacker through risk path identification.

1.3.2 Risk Path Assessment

A person with illegitimate intentions or an attacker delves into a web application to search for every possible path that could be exploited for imposing severe damage to the victim or targeted organization. Each of these identified paths represents a threat or a risk to an organization. Associated risk may severely affect the organization, thereby making it essential to build a robust and reliable web application. Figure 1.13 illuminates the process of path identification or exploitation by an attacker.

To assess the overall risk associated with the exploitation of existing flaws, there is a need to evaluate the probability associated with each factor like threat agents, attack payload, and security controls and integrate it with the overall impact on an

TABLE 1.1 Brief Description of Web Application Vulnerabilities

S. R. No.	Web Application Vulnerability	Description
V1.	Injection	Injection attack occurs as a result of the relay of malicious data by the attacker as a command or query which gets interpreted at the victim's browser, resulting in the alteration of information flow or theft of sensitive data without user consent.
V2.	Broken Authentication	This vulnerability provides privileges to the attacker to either bypass or breach the authentication mechanisms employed by the web application. Consequently, the attacker might get access to the user credentials, session tokens, and IDs to impersonate as the legitimate user.
V3.	Sensitive Data Exposure	This attack comes out to be the result of the loss of confidentiality between user and web application. This results in the theft of sensitive data like password, healthcare, and financial information by the attacker to trigger crimes like credit card fraud, cyber social clones, etc.
V4.	XML External Entities (XXE)	XML allows a user to refer to external resources in XML document, which gets substituted into the document by the XML parser during its execution. This vulnerability is utilized by an attacker to trick the XML parser to retrieve the resources of his interest. External entities may be capable of scanning internal ports, revealing sensitive data, performing server-side request forgery, and denial of service attack.
V5.	Broken Access Control	Users are allowed to perform their functionalities according to the assigned privileges. This is enforced through access control policies. When these policies are not properly imposed then the attacker compromises the entire web application's security by gaining admin privileges, modifying the access rights of other users, and misusing confidential information for its selfish motive.

(*Continued*)

TABLE 1.1 (CONTINUED) Brief Description of Web Application Vulnerabilities

S. R. No.	Web Application Vulnerability	Description
V6.	Security Misconfiguration	This type of vulnerability arises due to insecure configurations that are typically kept default in an application. An attacker can easily identify them through unpatched flaws, unprotected files, directories, etc., to pave the way for other serious attacks.
V7.	Cross-Site Scripting (XSS)	It is a type of code injection vulnerability which exists due to the improper validation of the data injected by any user. This flaw is exploited by the attacker to inject a malicious scripting code into the web application which when processed by the parser results in account hijacking, session token stealing, and redirection to the attacker's site.
V8.	Insecure Deserialization	This vulnerability occurs due to the improper deserialization. Deserialization is the process of converting some formatted data into objects. This vulnerability is utilized by the attacker to trick deserializer to process untrusted data resulting in remote code execution, denial of service attack, privilege escalation attack, etc.
V9.	Using Component with Known Vulnerabilities	A web application may include various components like libraries and frameworks to serve requests for the user. These components always run with all the privileges as the application. If the vulnerable component is employed then the attacker may exploit the weakness to gain control of the entire system or may lead to data loss.
V10.	Insufficient Logging and Monitoring	This is basically an opportunity to the attacker to infect the system with the same strategy used earlier as these systems do not maintain proper logs and monitor network activities. It results in tampering or data loss and sometimes control over the entire system.

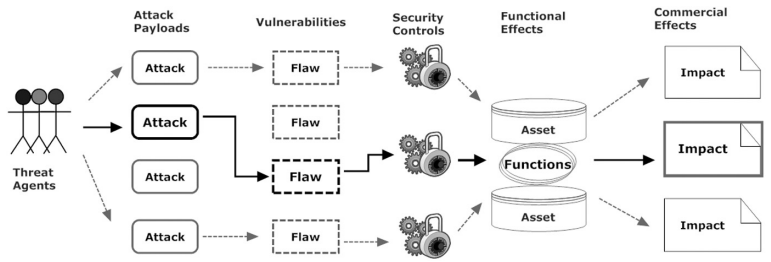


FIGURE 1.13 A scenario depicting risk path exploitation.

organization. Sometimes these paths can be easily identified during the development phase but not always. Likewise the associated damage may vary from no loss to complete loss of business. Therefore, identifying the most dangerous vulnerabilities and proposing mitigation mechanisms is the most pressing current demand.

1.3.3 Mapping Vulnerabilities with Risk Rating Methods

As we have illustrated in Figure 1.9, various risk paths may exist in web applications which may be exploited by the various threat agents (or attackers). Each of these paths comprises of various steps such as exploitation of vulnerability using attacking payload (exploitation), identification of vulnerability with its dominance (identification and dominance), and its impacts on business. Therefore, to assess the top 10 most dangerous vulnerabilities against these steps, Table 1.2 highlights the evaluations scheme of each of the steps identified [18].

TABLE 1.2 Evaluation Scheme of Risk Path Identified

		Vulnerability		
Threat Agents	**Exploitation**	**Dominance**	**Identification**	**Impact**
Specific to the Application Context	Easy	Rare	Simple	Low
	Average	Normal	Moderate	Medium
	Difficult	Broad	Hard	High

It is crucial to understand any web application vulnerabilities before a solution could be fabricated. Hence, Table 1.3 illustrates the mapping between web application vulnerability and risk path as per the steps shown in Table 1.2. Threat agents may be specific to the application context; therefore, each of the vulnerabilities can be exploited differently and may impose severe impacts of low to high severity level.

1.4 TOWARD BUILDING SECURE WEB APPLICATIONS

Heretofore, we reviewed the most severe and dominant web application vulnerabilities including the risk factors that these vulnerabilities can impose on any organization. In this section, we evaluate, from a generalized perspective, each web application and determine its measurement for identified risk factors.

These vulnerabilities exist in almost every web application, and their impact depends on the proficiency of the attacker to trigger an attack and the type of organization. It is unveiled that a small vulnerability may be catastrophic for an organization but may not pose a serious impact on another. Table 1.4 presents this evaluation for identified vulnerabilities. As the aftereffects of any attack may vary, there is a need to develop secure web applications from the development stage itself. There are some stages/activities that can be incorporated in the development cycle. Let's discuss these stages.

- ***Identification and Management of Risk:*** This stage deals with the detection of the risk that may be exploited in applications when they are released. The organization utilizes Dynamic Application Security Testing (DAST) to figure out the findings that help in creating and monitoring the risk metrics. These metrics assist in risk analysis, so that remediation solutions can be prioritized.
- ***Secure Patch Release Assurance:*** Amendment is an ongoing activity; every application must be updated with time.

TABLE 1.3 Mapping of Web Application Vulnerabilities with Risk Path

Top 10	Exploitation	Vulnerability: Dominance and Identification	Impacts
V1.	Injection flaw, for instance, SQL injection, LDAP injection, and OS injection, is the result of the insertion of malicious data into the web application via any input field like post, comment, form fields, etc. Any data can behave as malicious attack vector may be environment variables, URL parameters, etc.	Injection vulnerability dominates in all kinds of web application. An injection may occur in the form of SQL, LDAP, NoSQL, XPath, XML parsers, object relational mapping queries, etc. It can be easily identified through the use of scanners and code examination.	It results in the loss of confidentiality and integrity. It may shut down the entire system, leading to denial of access and control hijacking. Business-related impacts depend upon the context of the application and data used.
V2.	Attackers bypass the authentication method by utilizing various techniques like dictionary-based attack, brute force for mapping ID and password, and so on.	This attack is ubiquitous due to the implementation of identity validation and access control. The attacker can easily detect this vulnerability manually and utilize automatic mechanisms to exploit it.	The attacker gains control of the user account or get the entire system control, if admin is compromised. On the basis of the context of the application, it may be social identity clone, breach of user privacy, or financial fraud.
V3.	Deciphering is a complex task to achieve. Therefore, an attacker performs attacks to steal secret keys or performs passive attacks like eavesdropping and man-in-middle attack to steal sensitive information.	This vulnerability exists either because of no usage of crypto system or weak mechanism used for secret key generation and encryption. It is easy to detect server-side vulnerability when data is in transit; however, it is difficult to do when at rest.	This attack completely compromises the individual's privacy, which includes sensitive data like credit card number, health-related information, and any information which must be kept in secret from a person's perspective.

(*Continued*)

TABLE 1.3 (CONTINUED) Mapping of Web Application Vulnerabilities with Risk Path

Top 10	Exploitation	Vulnerability: Dominance and Identification	Impacts
V4.	The attacker can exploit this vulnerability by either using abused XML parser or inserting some malicious data into the XML document exploiting vulnerable code or any dependency on external references.	During XML processing, many older XML parsers require to specify the origin of the external references. Source code analysis is done to identify this vulnerability by checking for any dependencies or integration. Many automated tools are also used to find out the vulnerability existence in the web application.	The attack results in the remote access of the system, data disclosure, port scanning, and DoS attack. Its severity may vary as per the application context depending upon the privacy requirement.
V5.	The attacker can utilize static or dynamic application testing to search to figure out whether access control policies are enforced properly or not. It is the hardcore task of the attacker to gain unauthorized access.	This vulnerability is commonly found due to the flaw in the functional testing and ineffective access control policy regulation. Along with static and dynamic testing, manual testing is an effective approach to detect ineffective access control.	The attacker impersonates as a legitimate user gaining access to its data and may cause modification or destroy data, i.e. masquerade attack. Its severity may vary as per the application context depending upon the privacy requirement.

(*Continued*)

TABLE 1.3 (CONTINUED) Mapping of Web Application Vulnerabilities with Risk Path

Top 10	Exploitation	Vulnerability: Dominance and Identification	Impacts
V6.	The attacker identifies the default insecure configuration like unpatched errors, accounts with default configuration, and insecure files to gain control of the system.	This vulnerability is commonly found at any level in the application like database, networking services, web server, storage, and application server. It can be recognized easily with the help of automated scanners for scanning insecure configurations, use of accounts with default configurations, etc.	This vulnerability allows the attacker to gain access to the data in an unauthorized way or sometimes gaining control of the entire system. The severity level depends on the level of security requirement in the application context.
V7.	The attacker may utilize freely available framework or tools to detect XSS vulnerability in the web application.	Almost one-third of web applications are vulnerable to this attack. They can be detected with the help of automated scanners.	XSS attack results in account hijacking, phishing, disclosure of data, misuse of personal information, etc.
V8.	This attack is difficult to trigger; the attacker may alter some of the parameters that result in the redirection to the object for which the attacker is not authorized to use.	As it is not prevalent so far, its detection requires human intervention; however, some tools are there to detect insecure deserialization.	This attack may result in the remote code execution which leads to system control or system crash.

(*Continued*)

TABLE 1.3 (CONTINUED) Mapping of Web Application Vulnerabilities with Risk Path

Top 10	Exploitation	Vulnerability: Dominance and Identification	Impacts
V9.	The attacker can easily find an exploit for the known vulnerability. There is a need to perform some tasks for the checking of new vulnerability.	Applications with more third-party components' usage without proper validation are more infected with this attack. Automate scanners aid in identifying, but new exploitation may require efforts.	Depending upon the context of application this attack may cause severe harm including the loss of data and personal information.
V10.	This vulnerability sets the foundation for a large number of attacks. The attacker takes advantage of insufficient logging and lack in networking-related activities to achieve their motive.	One way to detect this flaw is by the careful monitoring of the events along with penetration testing. All the results must be logged properly to realize the damages. It takes a longer time to detect.	The attacker is capable enough to launch some large attacks and extract or destroy data, as the lack of monitoring is a plus point for the attacker.

Therefore, this stage ensures that any newly released component/patch of application is secure; i.e. it will not add new vulnerability and risk path to the current secure version of the application. The organization employs Static and Dynamic Application Security Testing (SAST and DAST) to achieve the main motto of this stage. It has also been assured that the remediation solutions implemented by the organization are successful in restricting the risk.

- ***Empowering Application Developers:*** This activity supports the organization through a reduction in the number of

TABLE 1.4 Evaluation of Web Application Vulnerabilities against Risk Factors

Risk →		Vulnerability Abuse		
Vulnerability ↓	Exploitation	Dominance	Identification	Impacts
V1	Easy	Normal	Simple	High
V2	Easy	Normal	Moderate	High
V3	Average	Broad	Moderate	High
V4	Average	Normal	Simple	High
V5	Average	Normal	Moderate	High
V6	Easy	Brad	Simple	Medium
V7	Easy	Broad	Simple	Medium
V8	Difficult	Normal	Moderate	High
V9	Average	Broad	Moderate	Medium
V10	Average	Broad	Hard	Medium

vulnerabilities that may arise due to the negligence of secure coding by the developers. Under this, the organization provides training on application security tools to the developers so that security issues can be detected and removed before they go unnoticed in any version release. Training sessions may be conducted depending upon the risk identified in applications and released patches. For this a questionnaire survey may be conducted by the security experts within the organization.

1.5 CHAPTER SUMMARY

Every business domain depends on the internet for expanding its business boundaries. This has led to the emergence of a large number of web applications available on the internet. Security is no longer optional while developing the application. Insecure development raises various security challenges. Therefore, the focus of this chapter has been to elaborate on the most dominant web application vulnerabilities. It has shown various statistics unveiled by different pioneer security organizations. This chapter

provided a comprehensive overview of the top 10 most harmful vulnerabilities that are more dominant and are being exploited every year despite deploying defensive solutions. It inferred that there are some security loopholes in web applications which present new risk paths. Furthermore, this chapter described each of the vulnerabilities against these risk paths. Precautions are better than cure; therefore, the execution of security aspects during the development phase perhaps helps organizations to understand the current scenario and a course toward improvement.

REFERENCES

1. Apache Software Foundation. (2019) Apache web server. [online] Available at: https://httpd.apache.org/docs/2.4/howto/.
2. Babiker, M., Karaarslan, E., & Hoscan, Y. (2018, March). Web application attack detection and forensics: A survey. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (pp. 1–6). IEEE.
3. Brunil, D., Romero, M., Haddad, H. M., & Molero, A. E. (2009). A methodological tool for asset identification in web applications. In *IEEE Fourth International Conference on Software Engineering Advances* (pp. 413–418).
4. Chaudhary, P., & Gupta, B. B. (2018). Plague of cross-site scripting on web applications: A review, taxonomy and challenges. *International Journal of Web Based Communities*, 14(1), 64–93.
5. Gupta, B., Agrawal, D. P., & Yamaguchi, S. (eds.). (2016). *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security.* IGI Global.
6. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security. Principles, Algorithm, Applications, and Perspectives*. CRC Press.
7. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(1), 1–31.
8. Gupta, B. B., Gupta, S., Gangwar, S., Kumar, M., & Meena, P. K. (2015). Cross-site scripting (XSS) abuse and defense: Exploitation on several testing bed environments and its defense. *Journal of Information Privacy and Security*, 11(2), 118–136.

9. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
10. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
11. Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: A prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.
12. Gupta, S., & Gupta, B. B. (2016). Enhanced XSS defensive framework for web applications deployed in the virtual machines of cloud computing environment. *Procedia Technology*, 24, 1595–1602.
13. Gupta, S., & Gupta, B. B. (2016). JS-SAN: Defense mechanism for HTML5-based web applications against JavaScript code injection vulnerabilities. *Security and Communication Networks*, 9(11), 1477–1495.
14. Gupta, S., & Gupta, B. B. (2016). XSS-SAFE: A server-side approach to detect and mitigate cross-site scripting (XSS) attacks in JavaScript code. *Arabian Journal for Science and Engineering*, 41(3), 897–920.
15. Lawton, G. (2007). Web 2.0 creates security challenges. *IEEE Computer Society*, 40(10), 13–16.
16. McClure, S., Shah, S., & Shah, S. (2017). *Web Hacking: Attacks and Defense*. Addison-Wesley Professional.
17. Microsoft IIS web server. [online] Available at: https://stackify.com/iis-web-server/.
18. OWASP. OWASP Top 10 2017: The ten most critical web application security risks. [online] Available at: https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf.
19. Seth, F., Jeremiah, G., Robert, H., Anton, R., & Petko, D. P. (2011). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Elsevier.
20. Toch, E., Bettini, C., Shmueli, E., Radaelli, L., Lanzi, A., Riboni, D., & Lepri, B. (2018). The privacy implications of cyber security systems: A technological survey. *ACM Computing Surveys*, 51(2), 36.
21. Web application vulnerability statistics report [online]. Available at: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Vulnerabilities-2017-eng.pdf.

22. Web application vulnerability statistics report. (2018). [online] Available at: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Vulnerabilities-2018-eng.pdf.
23. White hat security report. [online] Available at: https://info.whitehatsec.com/rs/675-YBI-674/images/WHS%202017%20Application%20Security%20Report%20FINAL.pdf.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

REFERENCES

1. Apache Software Foundation. (2019) Apache web server. [online] Available at: https://httpd.apache.org/docs/2.4/howto/.
2. Babiker, M., Karaarslan, E., & Hoscan, Y. (2018, March). Web application attack detection and forensics: A survey. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (pp. 1–6). IEEE.
3. Brunil, D., Romero, M., Haddad, H. M., & Molero, A. E. (2009). A methodological tool for asset identification in web applications. In *IEEE Fourth International Conference on Software Engineering Advances* (pp. 413–418).
4. Chaudhary, P., & Gupta, B. B. (2018). Plague of cross-site scripting on web applications: A review, taxonomy and challenges. *International Journal of Web Based Communities*, 14(1), 64–93.
5. Gupta, B., Agrawal, D. P., & Yamaguchi, S. (eds.). (2016). *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security.* IGI Global.
6. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security. Principles, Algorithm, Applications, and Perspectives*. CRC Press.
7. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(1), 1–31.
8. Gupta, B. B., Gupta, S., Gangwar, S., Kumar, M., & Meena, P. K. (2015). Cross-site scripting (XSS) abuse and defense: Exploitation on several testing bed environments and its defense. *Journal of Information Privacy and Security*, 11(2), 118–136.
9. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
10. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
11. Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: A prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.

12. Gupta, S., & Gupta, B. B. (2016). Enhanced XSS defensive framework for web applications deployed in the virtual machines of cloud computing environment. *Procedia Technology*, 24, 1595–1602.
13. Gupta, S., & Gupta, B. B. (2016). JS-SAN: Defense mechanism for HTML5-based web applications against JavaScript code injection vulnerabilities. *Security and Communication Networks*, 9(11), 1477–1495.
14. Gupta, S., & Gupta, B. B. (2016). XSS-SAFE: A server-side approach to detect and mitigate cross-site scripting (XSS) attacks in JavaScript code. *Arabian Journal for Science and Engineering*, 41(3), 897–920.
15. Lawton, G. (2007). Web 2.0 creates security challenges. *IEEE Computer Society*, 40(10), 13–16.
16. McClure, S., Shah, S., & Shah, S. (2017). *Web Hacking: Attacks and Defense*. Addison-Wesley Professional.
17. Microsoft IIS web server. [online] Available at: https://stackify.com/iis-web-server/.
18. OWASP. OWASP Top 10 2017: The ten most critical web application security risks. [online] Available at: https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf.
19. Seth, F., Jeremiah, G., Robert, H., Anton, R., & Petko, D. P. (2011). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Elsevier.
20. Toch, E., Bettini, C., Shmueli, E., Radaelli, L., Lanzi, A., Riboni, D., & Lepri, B. (2018). The privacy implications of cyber security systems: A technological survey. *ACM Computing Surveys*, 51(2), 36.
21. Web application vulnerability statistics report [online]. Available at: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Vulnerabilities-2017-eng.pdf.
22. Web application vulnerability statistics report. (2018). [online] Available at: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Vulnerabilities-2018-eng.pdf.
23. White hat security report. [online] Available at: https://info.whitehatsec.com/rs/675-YBI-674/images/WHS%202017%20Application%20Security%20Report%20FINAL.pdf.
1. Al-Qurishi, M., Al-Rakhami, M., Alamri, A., Alrubaian, M., Rahman, S. M. M., & Hossain, M. S. (2017). Sybil defense techniques in online social networks: A survey. *IEEE Access*, 5, 1200–1219.

2. Almomani, A., Gupta, B. B., Wan, T. C., Altaher, A., & Manickam, S. (2013). Phishing dynamic evolving neural fuzzy framework for online detection zero-day phishing email. *arXiv Preprint ArXiv:1302.0629*.
3. Benenson, Z., Gassmann, F., & Landwirth, R. (2017, April). Unpacking spear phishing susceptibility. In *International Conference on Financial Cryptography and Data Security* (pp. 610–627). Springer, Cham.
4. Boulianne, S. (2019). Revolution in the making? Social media effects across the globe. *Information, Communication and Society*, 22(1), 39–54.
5. Cai, Z., He, Z., Guan, X., & Li, Y. (2016). Collective data-sanitization for preventing sensitive information inference attacks in social networks. *IEEE Transactions on Dependable and Secure Computing*, 15(4), 577–590.
6. Chaudhary, P., Gupta, B. B., & Gupta, S. (2016, March). Cross-site scripting (XSS) worms in Online Social Network (OSN): Taxonomy and defensive mechanisms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2131–2136). IEEE.
7. Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, 59(7), 96–104.
8. Fire, M., Goldschmidt, R., & Elovici, Y. (2014). Online social networks: Threats and solutions. *IEEE Communications Surveys and Tutorials*, 16(4), 2019–2036.
9. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*. CRC Press.
10. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, 7(1), 1–31.
11. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
12. Gupta, S., & Gugulothu, N. (2018). Secure nosql for the social networking and e-commerce based bigdata applications deployed in cloud. *International Journal of Cloud Applications and Computing*, 8(2), 113–129.

13. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
14. Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: A prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.
15. Gupta, S., & Gupta, B. B. (2016). JS-SAN: Defense mechanism for HTML5-based web applications against JavaScript code injection vulnerabilities. *Security and Communication Networks*, 9(11), 1477–1495.
16. Isaac, Mike, & Frenkel, Sheera. Facebook security breach exposes accounts of 50 million users. [online] Available at: https ://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html
17. Jain, A. K., & Gupta, B. B. (2017). Phishing detection: Analysis of visual similarity based approaches. *Security and Communication Networks*, 2017.
18. Jiang, F., Fu, Y., Gupta, B. B., Lou, F., Rho, S., Meng, F., & Tian, Z. (2018). Deep learning based multi-channel intelligent attack detection for data security. *IEEE Transactions on Sustainable Computing*.
19. Kamhoua, G. A., Pissinou, N., Iyengar, S. S., Beltran, J., Kamhoua, C., Hernandez, B. L., Njilla, L., & Makki, A. P. (2017, June). Preventing colluding identity clone attacks in online social networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)* (pp. 187–192). IEEE.
20. Li, C., Zhang, Z., & Zhang, L. (2018). A novel authorization scheme for multimedia social networks under cloud storage method by using MA-CP-ABE. *International Journal of Cloud Applications and Computing*, 8(3), 32–47.
21. Li, H., Chen, Q., Zhu, H., Ma, D., Wen, H., & Shen, X. S. (2017). Privacy leakage via de-anonymization and aggregation in heterogeneous social networks. *IEEE Transactions on Dependable and Secure Computing*.
22. Li, H., Zhu, H., Du, S., Liang, X., & Shen, X. S. (2016). Privacy leakage of location sharing in mobile social networks: Attacks and defense. *IEEE Transactions on Dependable and Secure Computing*, 15(4), 646–660.

23. Liu, J., Tao, Y., & Bai, Q. (2016, August). Towards exposing cyberstalkers in online social networks. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 763–770). Springer, Cham.
24. Mocktoolah, A., & Khedo, K. K. (2015, December). Privacy challenges in proximity based social networking: Techniques & solutions. In *2015 International Conference on Computing, Communication and Security (ICCCS)* (pp. 1–8). IEEE.
25. Olakanmi, O. O., & Dada, A. (2019). An efficient privacy-preserving approach for secure verifiable outsourced computing on untrusted platforms. *International Journal of Cloud Applications and Computing*, 9(2), 79–98.
26. Patel, P., Kannoorpatti, K., Shanmugam, B., Azam, S., & Yeo, K. C. (2017, January). A theoretical review of social media usage by cyber-criminals. In *2017 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1–6). IEEE.
27. Pew Research Report Pew Research Center. (2018). *Social Media Use in 2018*. [online] Available at: https://www.pewresearch.org/internet/2018/03/01/social-media-use-in-2018/.
28. Rathore, S., Sharma, P. K., Loia, V., Jeong, Y. S., & Park, J. H. (2017). Social network security: Issues, challenges, threats, and solutions. *Information Sciences*, 421, 43–69.
29. Sahoo, S. R., & Gupta, B. B. (2019). Classification of various attacks and their defence mechanism in online social networks: A survey. *Enterprise Information Systems*, 13(6), 832–864.
30. Social media active users. [online] Available at: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.
31. Squicciarini, A., Rajtmajer, S., Liu, Y., & Griffin, C. (2015, August). Identification and characterization of cyberbullying dynamics in an online social network. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015* (pp. 280–285). ACM.
32. Tian, Y., Yuan, J., & Yu, S. (2016, October). SBPA: Social behavior based cross social network phishing attacks. In *2016 IEEE Conference on Communications and Network Security (CNS)* (pp. 366–367). IEEE.
33. WhatsApp vulnerability. [online] Available at: https://www.helpnetsecurity.com/2019/05/14/whatsapp-flaw-spyware-cve-2019-3568/.

34. Xu, H., Sun, W., & Javaid, A. (2016, March). Efficient spam detection across online social networks. In *2016 IEEE International Conference on Big Data Analysis (ICBDA)* (pp. 1–6). IEEE.
35. Yan, G., Chen, G., Eidenbenz, S., & Li, N. (2011, March). Malware propagation in online social networks: Nature, dynamics, and defense implications. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (pp. 196–206). ACM.
36. Zhang, Z., Sun, R., Zhao, C., Wang, J., Chang, C. K., & Gupta, B. B. (2017). CyVOD: A novel trinity multimedia social network scheme. *Multimedia Tools and Applications*, 76(18), 18513–18529.
1. Almomani, A., Gupta, B. B., Wan, T. C., Altaher, A., & Manickam, S. (2013) Phishing dynamic evolving neural fuzzy framework for online detection zero-day phishing email. *arXiv preprint arXiv:1302.0629*.
2. Chaudhary, P., Gupta, B. B., & Gupta, S. (2018). Defending the OSN-based web applications from XSS attacks using dynamic javascript code and content isolation. In *Quality, IT and Business Operations* (pp. 107–119). Springer, Singapore.
3. Chaudhary, P., Gupta, S., & Gupta, B. B. (2016). Auditing defense against XSS worms in online social network-based web applications. In *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security* (pp. 216–245). IGI Global.
4. Cross site scripting, OWASP. [online] Available at: https://www.owasp.org/index.php/Cross-site_Scripting (XSS).
5. Dong, G., Zhang, Y., Wang, X., Wang, P., & Liu, L. (2014). Detecting cross site scripting vulnerabilities introduced by HTML5. In *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE.
6. Duchene, F., Rawat, S., Richier, J.-L., & Groz, R. (2014). KameleonFuzz: Evolutionary fuzzing for black-box XSS detection. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy* (pp. 37–48). ACM.
7. Guo, X., Jin, S., & Zhang, Y. (2015). XSS vulnerability detection using optimized attack vector repertory. In *2015 International Conference On Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE.
8. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*. CRC Press.

9. Gupta, B. B., & Badve, O. P. (2017). Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. *Neural Computing and Applications*, 28(12), 3655–3682.
10. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, 7(1), 1–31.
11. Gupta, M. K., Govil, M. C., Singh, G., & Sharma, P. (2015). XSSDM: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE.
12. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
13. Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: A prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.
14. Gupta, S., & Gupta, B. B. (2016). JS-SAN: Defense mechanism for HTML5-based web applications against JavaScript code injection vulnerabilities. *Security and Communication Networks*, 9(11), 1477–1495.
15. Gupta, S., & Gupta, B. B. (2017). Cross-site scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8(1), 512–530.
16. Gupta, S., & Gupta, B. B. (2018). A robust server-side javascript feature injection-based design for JSP web applications against XSS vulnerabilities. In *Cyber Security* (pp. 459–465). Springer, Singapore.
17. Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). A client-server JavaScript code rewriting-based framework to detect the XSS worms from online social network. *Concurrency and Computation: Practice and Experience*, 31(21), e4646.
18. Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). Hunting for DOM-based XSS vulnerabilities in mobile cloud-based online social network. *Future Generation Computer Systems*, 79, 319–336.

19. Jain, A. K., & Gupta, B. B. (2017). Phishing detection: Analysis of visual similarity based approaches. *Security and Communication Networks*, 2017.
20. Jiang, F., Fu, Y., Gupta, B. B., Lou, F., Rho, S., Meng, F., & Tian, Z. (2018). Deep learning based multi-channel intelligent attack detection for data security. *IEEE Transactions on Sustainable Computing*.
21. Khan, N., Abdullah, J., & Khan, A. S. (2015). Towards vulnerability prevention model for web browser using interceptor approach. In *2015 9th International Conference on IT in Asia (CITA)*. IEEE.
22. Lalia, S., & Sarah, A. (2018, March). XSS attack detection approach based on scripts features analysis. In *World Conference on Information Systems and Technologies* (pp. 197–207). Springer, Cham.
23. Lekies, S., Stock, B., & Johns, M. (2013). 25 million flows later: Large-scale detection of DOM-based XSS. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM.
24. Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., & Xiaoxi, W. (2019). MLPXSS: An integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. *IEEE Access*, 7, 100567–100580.
25. Moniruzzaman, M., Bagirov, A., Gondal, I., & Brown, S. (2018, June). A server side solution for detecting WebInject: A machine learning approach. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 162–167). Springer, Cham.
26. Nadji, Y., Saxena, P., & Song, D. (2009, February). Document structure integrity: A robust basis for cross-site scripting defense. In *NDSS*.
27. Panja, B., Gennarelli, T., & Meharia, P. (2015). Handling cross site scripting attacks using cache check to reduce webpage rendering time with elimination of sanitization and filtering in light weight mobile web browser. In *2015 First Conference on Mobile and Secure Services (MOBISECSERV)*. IEEE.
28. Parameshwaran, E. B., Shinde, S., Dang, H., Sadhu, A., & Saxena, P. (2015). DexterJS: Robust testing platform for DOM-based XSS vulnerabilities. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)* (pp. 946–949). ACM.

29. Rocha, T. S., & Souto, E. (2014). ETSSDetector: A tool to automatically detect cross-site scripting vulnerabilities. In *2014 IEEE 13th International Symposium on Network Computing and Applications (NCA)*. IEEE.
30. Ruse, M. E., & Basu, S. (2013). Detecting cross-site scripting vulnerability using concolic testing. In *2013 Tenth International Conference on Information Technology: New Generations (ITNG)*. IEEE.
31. Scholte, T., Robertson, W., Balzarotti, D., & Kirda, E. (2012). Preventing input validation vulnerabilities in web applications through automated type analysis. In *2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE.
32. Steinhauser, A., & Tůma, P. (2019). DjangoChecker: Applying extended taint tracking and server side parsing for detection of context-sensitive XSS flaws. *Software: Practice and Experience*, 49(1), 130–148.
33. Stock, B., Pfistner, S., Kaiser, B., Lekies, S., & Johns, M. (2015). From facepalm to brain bender: Exploring client-side cross-site scripting. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)* (pp. 1419–1430). ACM.
34. Tripathi, S., Gupta, B., Almomani, A., Mishra, A., & Veluru, S. (2013). Hadoop based defense solution to handle distributed denial of service (ddos) attacks. *Journal of Information Security*, 04(3), 150.
35. Van Acker, S., Nikiforakis, N., Desmet, L., Joosen, W., & Piessens, F. (2012). FlashOver: Automated discovery of cross-site scripting vulnerabilities in rich internet applications. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security)* (pp. 12–13). ACM.
36. Vishnu, B. A., & Jevitha, K. P. (2014). Prediction of cross-site scripting attack using machine learning algorithms. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC '14)*. ACM.
37. Wang, R., Jia, X., Li, Q., & Zhang, D. (2015). Improved N-gram approach for cross-site scripting detection in Online Social Network. In *2015 Science and Information Conference (SAI)*. IEEE.
38. Wang, R., Jia, X., Li, Q., & Zhang, S. (2014). Machine learning based cross-site scripting detection in online social network. In 2014 IEEE *International Conference on High Performance*

Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and Syst (HPCC, CSS, ICESS). IEEE.

39. Wang, R., Xu, G., Zeng, X., Li, X., & Feng, Z. (2018). TT-XSS: A novel taint tracking based dynamic detection framework for DOM cross-site scripting. *Journal of Parallel and Distributed Computing*, 118, 100–106.
40. Xiao, W., Sun, J., Chen, H., & Xu, X. (2014). Preventing client side XSS with rewrite based dynamic information flow. In *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programing (PAAP)*. IEEE.
41. XSS incidents information. [online] Available at: http://www.xssed.com/.
42. Zhang, Q., Chen, H., & Sun, J. (2010). An execution-flow based method for detecting cross-site scripting attacks. In *2010 2nd International Conference on Software Engineering and Data Mining (SEDM)*. IEEE.

1. 523 XSS vectors available. [online] Available at: http://xss2.technomancie.net/vectors.
2. @XSS vector twitter account. [online] Available at: https://twitter.com/XSSVector.
3. Aggarwal, C. C., & Zhai, C. (2012). A survey of text clustering algorithms. In *Mining Text Data* (pp. 77–128). Springer, Boston, MA.
4. Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G.. (2008). Saner: Composing static and dynamic analysis to validate sanitization in web applications. In *IEEE Symposium on Security and Privacy. SP 2008* (pp. 387–401). IEEE, Oakland, CA.
5. Chaudhary, P., & Gupta, B. B. (2018). Plague of cross-site scripting on web applications: A review, taxonomy and challenges. *International Journal of Web Based Communities*, 14(1), 64–93.
6. Chaudhary, P., Gupta, B. B., & Gupta, S. (2019). A framework for preserving the privacy of online users against XSS worms on online social network. *International Journal of Information Technology and Web Engineering*, 14(1), 85–111.
7. Drupal social networking site. [online] Available at: https://www.drupal.org/download.

8. Elgg social networking engine. [online] Available at: https://elgg.org.
9. ESAPI, OWASP Enterprise Security API. (2009). [online] Available at: http://www.owasp.org/index.php/ESAPI#tab=Project_Details (accessed February 2010).
10. Gupta, B. B., & Agrawal, D. P. (eds.). (2019). *Handbook of Research on Cloud Computing and Big Data Applications in IoT*. IGI Global.
11. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, 7(1), 1–31.
12. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
13. Gupta, S., & Gupta, B. B. (2015, May). PHP-sensor: A prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications. In *Proceedings of the 12th ACM International Conference on Computing Frontiers* (p. 59). ACM.
14. Hansen, R. XSS (cross site scripting) cheat sheet. Filter evasion cheat sheet. [online] Available at: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.
15. Heiderich, M. Html5 security cheatsheet. [online] Available at: http://html5sec.org.
16. Hooimeijer, P., Livshits, B., Molnar, D., Saxena, P., & Veanes, M. (2011). Fast and precise sanitizer analysis with BEK. In *Proceedings of the 20th USENIX Conference on Security* (pp. 1–1). USENIX Association.
17. Humhub social networking site. [online] Available at: https://www.humhub.org/en.
18. Joomla social networking site. [online] Available at: https://www.joomla.org/download.html.
19. Joshi, R. C., & Gupta, B. B. (eds.). (2019). *Security, Privacy, and Forensics Issues in Big Data*. IGI Global.
20. Jsoup HTML parser. [online] Available at: https://jsoup.org/.
21. Livshits, B., & Chong, S. (2013). Towards fully automatic placement of security sanitizers and declassifiers. *ACM SIGPLAN Notices*, 48(1), 385–398.

22. Metzler, D., Dumais, S., & Meek, C. (2007). Similarity measures for short segments of text. In *European Conference on Information Retrieval*. Springer, Berlin, Heidelberg.
23. Samuel, M., Saxena, P., & Song, D. (2011). Context-sensitive auto-sanitization in web templating languages using type qualifiers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (pp. 587–600). *ACM*.
24. Sarmah, U., Bhattacharyya, D. K., & Kalita, J. K. (2018). A survey of detection methods for XSS attacks. *Journal of Network and Computer Applications*, 118, 113–143.
25. Saxena, P., Hanna, S., Poosankam, P., & Song, D. (2010). FLAX: Systematic discovery of client-side validation vulnerabilities in rich web applications. In *NDSS Symposium*.
26. Saxena, P., Molnar, D., & Livshits, B. (2011). SCRIPTGARD: Automatic context-sensitive sanitization for large-scale legacy web applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security* (pp. 601–614). ACM, Chicago, IL.
27. Technical attack sheet for cross site penetration tests. [online] Available at: http://www.vulnerability-lab.com/resources/documents/531.txt.
28. WordPress. [online] Available at: http://wordpress.org/.
29. XSS filter evasion cheat sheet. [online] Available at: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.
30. Zhang, Z., & Gupta, B. B. (2018). Social media security and trustworthiness: Overview and new direction. *Future Generation Computer Systems*, 86, 914–925.
1. Burp scanner. [online] Available at: https://support.portswigger.net/customer/portal/articles/1783127-using-burp-scanner.
2. Cao, Y., Yegneswaran, V., Porras, P. A., & Chen, Y. (2012). PathCutter: Severing the self-propagation path of XSS JavaScript worms in social web networks. In *NDSS*.
3. Chaudhary, P., Gupta, B. B., & Gupta, S. (2019). A framework for preserving the privacy of online users against XSS worms on online social network. *International Journal of Information Technology and Web Engineering*, 14(1), 85–111.
4. Chaudhary, P., Gupta, S., & Gupta, B. B. (2016). Auditing defense against XSS worms in online social network-based web applications. In *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security* (pp. 216–245). IGI Global.

5. Faghani, M. R., & Nguyen, U. T. (2013). A study of XSS worm propagation and detection mechanisms in online social networks. *IEEE Transactions on Information Forensics and Security*, 8(11), 1815–1826.
6. Faghani, M. R., & Saidi, H. (2009). Social networks' XSS worms. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering (CSE'09)*. IEEE.
7. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*. CRC Press.
8. Gupta, B. B., & Gupta, A. (2018). Assessment of honeypots: Issues, challenges and future directions. *International Journal of Cloud Applications and Computing*, 8(1), 21–54.
9. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, 7(1), 1–31.
10. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
11. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
12. Gupta, S., & Gupta, B. B. (2017). Detection, avoidance, and attack pattern mechanisms in modern web application vulnerabilities: Present and future challenges. *International Journal of Cloud Applications and Computing*, 7(3), 1–43.
13. Gupta, S., & Gupta, B. B. (2018). Robust injection point-based framework for modern applications against XSS vulnerabilities in online social networks. *International Journal of Information and Computer Security*, 10(2–3), 170–200.
14. Gupta, S., & Gupta, B. B. (2019). Evaluation and monitoring of XSS defensive solutions: A survey, open research issues and future directions. *Journal of Ambient Intelligence and Humanized Computing*, 10(11), 4377–4405.
15. Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). A client-server JavaScript code rewriting-based framework to detect the XSS worms from online social network. *Concurrency and Computation: Practice and Experience*, 31(21), e4646.

16. htmlLawed. [online] Available at: https://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/.
17. HTML purifier. [online] Available at: http://htmlpurifier.org/.
18. ImmuneWeb on-demand. [online] Available at: https://www.immuniweb.com/products/ondemand/.
19. Netsparker. [online] Available at: https://www.netsparker.com/.
20. OWASP Antisamy. [online] Available at: https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project.
21. OWASP HTML sanitizer. [online] Available at: https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project.
22. OWASP Xenotix XSS exploit framework. [online] Available at: https://www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework.
23. OWASP Zed Attack Proxy (ZAP). [online] Available at: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
24. Kozlowski, M. Power Fuzzer: web application vulnerabilities scanner. [online]. Available: https://www.powerfuzzer.com/.
25. Loureiro, N. Probely: web vulnerability scanner. [online]. Available at: https://blog.probely.com/web-security-testing-101-c08bc9117768.
26. Sahoo, S. R., & Gupta, B. B. (2019). Hybrid approach for detection of malicious profiles in twitter. *Computers and Electrical Engineering*, 76, 65–81.
27. Seth, F., Jeremiah, G., Robert, H., Anton, R., & Petko, D. P. (2011). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Elsevier.
28. Subgraph Vega vulnerability scanner. [online] Available at: https://subgraph.com/vega/.
29. W3af. [online] Available at: http://w3af.org/.
30. WebScaracb. [online] Available at: https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.
31. XSSer. [online] Available at: https://xsser.03c8.net/.
1. Content security policy. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP.
2. Gupta, B. B. (ed.). (2018). *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*. CRC Press.

3. Gupta, B. B., Gupta, S., & Chaudhary, P. (2017). Enhancing the browser-side context-aware sanitization of suspicious HTML5 code for halting the DOM-based XSS vulnerabilities in cloud. *International Journal of Cloud Applications and Computing*, 7(1), 1–31.
4. Gupta, B. B., & Sheng, Q. Z. (eds.). (2019). *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press.
5. Gupta, S., & Gupta, B. B. (2015). BDS: Browser dependent XSS sanitizer. In *Handbook of Research on Securing Cloud-Based Databases with Biometric Applications* (pp. 174–191). IGI Global.
6. Gupta, S., & Gupta, B. B. (2016). JS-SAN: Defense mechanism for HTML5-based web applications against JavaScript code injection vulnerabilities. *Security and Communication Networks*, 9(11), 1477–1495.
7. Gupta, S., Gupta, B. B., & Chaudhary, P. (2018). A client-server JavaScript code rewriting-based framework to detect the XSS worms from online social network. *Concurrency and Computation: Practice and Experience*, 31(21), e4646.
8. Jiang, F., Fu, Y., Gupta, B. B., Lou, F., Rho, S., Meng, F., & Tian, Z. (2018). Deep learning based multi-channel intelligent attack detection for data security. *IEEE Transactions on Sustainable Computing*.
9. Sarmah, U., Bhattacharyya, D. K., & Kalita, J. K. (2018). A survey of detection methods for XSS attacks. *Journal of Network and Computer Applications*, 118, 113–143.
10. Seth, F., Jeremiah, G., Robert, H., Anton, R., & Petko, D. P. (2011). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Elsevier.
11. Stergiou, C., Psannis, K. E., Xifilidis, T., Plageras, A. P., & Gupta, B. B. (2018, April). Security and privacy of big data for social networking services in cloud. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 438–443). IEEE.
12. Taha, T. A., & Karabatak, M. (2018, March). A proposed approach for preventing cross-site scripting. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (pp. 1–4). IEEE.

13. White hat security report. [online] Available at: https://info.whitehatsec.com/rs/675-YBI-674/images/WHS%202017%20Application%20Security%20Report%20FINAL.pdf.
14. XSS filter evasion cheat sheet. [online] Available at: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.
15. XSS prevention cheat sheet. [online] Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.