6.1 Summary

Now a day Internet has become handy and most advanced useful technology due to use of various electronic gadgets. This online system helps the present human civilization to such a greater extend that life without internet seems to be impossible. Today thousands of web sites which includes customized services become key part of most of the human activities. Thus present generation become thoroughly computer obsessed. Due to its popularity, Internet has started attracting hackers/attackers who keep looking for new techniques to create maliciousness in web application for their personal gain. Web sites are introduced to provide worldwide connectivity, to provide information and services, to make user's work easier, to save transaction time and human efforts. So no one has right to misuse it for personal gain or for harassment.

According to researchers and industry experts, the Cross-Site Scripting (XSS) is the one of the top most vulnerability in the web application which affects victims badly. A cross-site scripting is a one kind of application-layer web attack in which attackers try to inject malicious scripts to perform malicious actions on trusted websites. In cross-site scripting, malicious code executes on the browser side and harms web users.

XSS attacks can be used by criminal to gain financial advantage or just to harass someone. It takes many forms like: take over user' account, spread worms (e.g. Samy XSS worms), Trojan horse, control access of browser, phishing, expose of the user's session cookie, redirect the user to some other page or site, modify presentation of content, bypass restrictions, malware attacks & DoS attack, fake advertisement, click fraud, etc.

XSS is simplest technique for attacker/hacker to meet their desires because for explosion they need only 3 basic things: 1) a JavaScript code that steal confidential data, 2) a web page which receives sensitive information of victim user and 3) a domain space to host malicious web page. XSS exploitation generally occurs due to: improper validation of user's input, output of web page is also not

validated before it is sent to client browser, value manipulation of HTTP header fields, browser does not interpret meaning of given script code.

XSS attacks normally executes when the web page is loaded or associated event is occur. XSS attack can be usually fired by embedding vulnerable code in JavaScript and HTML form. But it can be also embedded through any other browser supported executable scripting language and mark-up language. Furthermore, attacker/hacker exploits XSS attack vectors using encoding methods like: HTML, Hex, Decimal, Octal, UTF-8, Base64, etc. So identifying XSS attack vectors is challenging task.

Stored XSS, reflected XSS, DOM-based XSS, XSSR/CSSR, XSSQLI and XSRF/CSRF are various forms of XSS attacks which utilize by attacker/hacker to create vulnerabilities in web application. Stored XSS usually occurs when user' input contains XSS vectors and is stored in database on the target server. Reflected attacks are those where the injected script is reflected off the web server, for example in search result, an error message, or any other response that contains user' input sent to the server the request part. DOM XSS is type of cross site scripting attack which arises due to improper handling of data related to DOM (Document Object Model) present in HTML page. XSSR or CSSR stands for Cross Site Script Redirection is used to redirect a victim to another page unwillingly for example mouse over event redirect user to some malicious page. XSSQLI is an amalgamation of Cross Site Scripting attack and SQL Injection attack, where an unaware victim visits a crafted link containing SQL Injection commands for a region of the website which requires rights that other user doesn't have. XSRF or CSRF (sometimes referred to as C-Surf) stands for Cross Site Request Forgery which is used to send automated input via the user to the target site.

Generally, XSS data payload is formatted as hyperlink i.e. using SRC and HREF attributes embedded with vulnerable content and is distributed over the Internet. Attacker/hacker injects malicious code using HTML tags, events handlers or DOM methods.

Usually web application developers are not aware of various types of vulnerabilities which occur in web page and their consequences. So for them it may

difficult to provide security to protect and defend against such kind of vulnerabilities and their consequences. Moreover to make changes in existing web site in order to provide security is also time consuming and expensive process. So dealing with such kind of vulnerability in web application is very difficult job. Therefore this research work is conducted to propose security mechanism which protects web application from such kind of Cross-Site Scripting attacks and its consequences.

This research work presents detailed study of:

- Cross-Site Scripting attacks
- Various forms of cross-site scripting attacks
- Sub-categories of cross-site scripting attacks
- Root cause of attacks
- Role of different actor in occurrence of attack scenario in web application
- Various attack injection points
- Analysis of various security reports
- Problems in current web scenario

In this research area, various past studies have been carried out. To study them, many literatures are reviewed which helps to analyze past and present scenario of different trends and technologies work towards defending such kind of web vulnerabilities cause through cross-site scripting attacks which harm web application as well as its authorized user.

In this research work, brief explanation about various techniques and methodologies introduced and implemented by researchers in order to detect and prevent different kinds of cross-site scripting attacks is also cited. Various mitigation techniques, replace functions, content filtering, cryptography, blacklist filtering, whitelist filtering, plug-ins, black-box testing, web page crawling, IP defending, hash mapping, sanitization process, pattern matching using regular expressions, analyzing depth of URL, output filtering, encoding dynamic contents, working of DOM parser, features extraction, analyze static as well as dynamic tags, HTTP header observation, browser modification, client site or server site proxy, maintaining

attack vector repositories, handling false positive as well as false negative rates, tokenization, vulnerability scanners, query handling, etc. techniques are presented by researchers. Here, comparative analysis of various reviewed models is also carried out which depict which model works on which type of attacks and whether it only detects or also provides prevention mechanism against such kind of attacks.

To protect web application from different types of Cross-Site Scripting vulnerabilities, many research objectives are designed for undertaken research work. Identify all possible patterns and aspects of XSS attacks, find out malicious script injected point in web application, prevent against cookies and session stealing using XSS that leak victim's sensitive information, protects the victim who is redirected to another page unwillingly or their inputted data is sent automatically to target site, protects web application from persistent and non-persistent XSS attacks, DOM-based XSS attacks, XSSQLI attacks, etc. are key objectives of this research work.

To check the need of proposed detection and prevention mechanism against cross-site scripting attacks and its various forms, a survey has been conducted of various companies who are engaged with web development activities and web developers. This survey contains information about how web application security testing is conducted by these companies and web developers. To find cross-site scripting vulnerabilities, whether they test security checks against it manually, using some tools or both. During this test, if cross-site scripting vulnerabilities are found then how they fix such exploits. This survey includes 100 samples from 26 web development companies and 74 web developers in response to this subject. In this 100 samples, all of them perform only security testing manually or using some tools, no one is utilizing any prevention mechanism. Based on 100 samples survey result, 69% testing work is done manually and 29% testing is done using various tools, whereas 2% of them perform testing manually as well as using tools. In this survey, a question was asked to web development companies and web developers about how they perform security testing for detecting cross-site scripting vulnerabilities in their web applications. With response to this question 20 companies said that they are performing XSS vulnerability test manually, 5 companies use some testing tools and 1 company does not perform any security testing against cross-site scripting vulnerabilities. With response to the same question 25 web developers said that they

are performing vulnerability test manually, 46 web developers use some testing tools and 3 web developers do not perform any security testing against cross-site scripting vulnerabilities

Based on various past studies of research carried out so far, we analyze that there are eight different approaches which are commonly utilized by researchers to prevent cross-site scripting vulnerabilities. The commonly used approaches are: 1) user input validation, 2) web scanner, 3) firewall, 4) hash mapping, 5) browser modifications, 6) static analysis, 7) tokenization and 8) IP tracking.

- User input validation approach only verified predefined signatures of attacks by setting policy rule set or whitelist or blacklist which requires regular updation to incorporate newly introduced attack signature.
- Web scanner simply detects attacks, does not provide protection mechanism to resolve these attacks. If bandwidth is low, web scanner may slowdown the given web site or entire web server on which this web site is host.
- Sometimes firewall gives popup to user for valid packet of legal site which means it continuously requires user's interaction. Some firewalls only allow predefined URLs and it needs updation for filter rules continuously.
- Hash mapping technique increases storage space because hash value is also required to be stored with its original value and if attacker fetches this hashed code then number of readymade tool as well as web sites are available which decrypt hash code. One cannot makes any changes directly in rendering process of browser easily.
- Modifications in browser requires lots of efforts with accuracy and consistency.
- Static analysis approach manually checks information flow in web page. It neither respond against changes made by dynamic contents nor provides any protection mechanism.
- In tokenization approach, every time random unique token is required to be created. Maintaining these tokens, comparing it to verify check points increase lots of overheads of server.

- IP tracking technique gets failed if attacker is connected through VPN then fetched IP location is fake. What if this blocked IP is spoofed by attacker??? Here authorized user having this IP is not aware of IP spoofing and its misusing then user is became victim. Genuine user is not able to access Internet services.

To meet the needs of research objectives, the framework called web watch is proposed. This framework monitors scripts (client-site as well as server-site) of web application to detect XSS injected script that directly reflect to HTML and DOM API and also harms the victim. This proposed framework contains following seven main modules:

1) XSS Detector
2) Token Analyzer
3) DOM Tracker
4) XSSQLI Pattern Extractor
5) XSSR & XSRF Evaluator
6) Attack Handler
7) Report Monitoring

XSS detector module's algorithm can be implemented as web service routine. When user provides inputs in given web page, web developer need to call this web service routine to detect vulnerable script attack by providing user input as parameter to this web service routine. This algorithm for XSS detector module detects stored cross-site scripting attacks as well as reflected cross-site scripting attacks. Attacker/hacker may provide attack vectors as inputted data using encoding charset by using some readymade functions. If it is encoded then decode the given inputted string so that the attack signatures can be easily identified. Once given user input is decoded, the next step is identifying attack vectors of stored cross-site scripting and reflected cross-site scripting in user provided input.

Token analyzer module's algorithm can be implemented at web development phase. When user provides login details in given web page, web developer need to

set and verify user session for authorized user. This algorithm helps web developer to protect session and cookies against XSS attacks that leaks sensitive information of victim. To do so, web developer needs to create protected session. For this, user id, browser details (like name of browser and browser version) and IP address needs to be stored as session variables. Encrypt these values of user session. Web developer also needs to check certain information with every web page request. Session id needs to be changed repeatedly. Apart from algoritham and to provide better security, web developer needs to set some security options as configuration setting in the web configuration file of given web site.

The DOM XSS attack is difficult to detect by server-side attack detection and prevention tools, because usually the malicious payload does not get to the server and hence cannot be sanitized in the server-side code, like in the case of other XSS attacks. So DOM Tracker module's algorithm can be implemented as client-site solution (like plug-ins or browser modification) wherein DOM properties can be modified either directly or via accessing HTML. Before requested web page is appear or visible in client browser, browser's rendering engine creates render engine which is combination of DOM tree and CSSOM tree. To fix up DOM attacks, render tree needs to be traverse and verify properly before it started drawing on browser's screen. Here, SiteURLTable is data table, which contains list of URL supported by given web site; HTML tag is considered as an element node; attribute of HTML tag is considered as an attribute node; if web page contains comment, it is considered as comment node. This algorithm checks comment node and attribute node only. Text node is not verified here as it contains contents of web page. In overall scenario of DOM based attack, attacks are generally occurred by: changing URL redirection, modifying current HTML element nodes and add new HTML element node. This can be done by using either document object, location object, windows object or using element directly. In all above cases, attacker/hacker mostly tries to redirect victim to some malicious web page. All the modifications of attacker/hacker are interpreted and resulting elements are added to render tree which in turn is painted on browser's screen. So in above algorithm, only those attribute nodes are veriferd which contains 'href' or 'src' attribute.

Reflected XSS attacks and Stored XSS attacks are also known as standard/classic XSS. The difference among the standard XSS attacks and DOM based XSS attacks are measured by using certain characteristics like: their root cause; location where they actually found; nature of page where these type of attacks can be occurred; detection and prevention mechanisms used for these type of exposure.

For XSSQLI attacks, it is important to note that whenever web developer needs to fetch and process the user provided data then avoid direct use of such data in query statement. After fetching user's input parameters, check whether these parameters contain special characters or not by extracting its pattern. If it contains special characters/tags then pass IP detail to attack handler module. Perform data encryption before start any processing (i.e. processing like data comparison or data storing in data table). By performing encryption before any processing, one can avoid XSSQLI attacks for login bypass because if input parameters contain attack vectors then form of attack vectors are changed by encryption as static string and thus it does not evaluated directly. And if attacker tries to expose advanced SQL injection then data value becomes useless for him because it is in encryption form and difficult to decrypt without having unlock key or using readymade decryption tools. To perform encryption and decryption of given piece of data, the proposed module's algorithm can be implemented as web service which contains two methods for encryption and decryption. To encrypt and decrypt data, this web service utilize tripleDES cryptographic service provider. Here, each and every block of user provided data is separately encrypted by using ECB mode of DES. Using ECB mode, if error occurs during encryption of one block then it does not affect any other block of same string data. Here, it is important to note that perform encryption only on sensitive data because it have some limitations like: Storing of data in encrypted form increase storage space of database; It slow down system's response time; and It is complex task to perform.

XSSR & XSRF Evaluator module is divided into two parts: part 1 for simple redirection and part 2 for request forgery. To protect against simple redirection attacks (part 1) and request forgery attacks (part 2), the proposed module's

algorithms can be implemented at web development phase. For part 1 attacks, URL of ‘href’ or ‘src’ attribute is fetched and check against data entries of SiteURLTable. For part 2 attacks, when user provides login details in given web page, web developer need to set and verify user session for authorized user as mentioned below to protect against attack.

Attack handler module records attack details into related data tables. When attack is identified by various modules then each module passes attack details to this attack handler module. All this information is properly categorized and stored into database for the purpose of analyzing attacks. With each web page request, IP address of user must be fetched and verified. This IP address passes by each and every module which detects attack’s signature. This module also fetch information about given IP address of device used by attacker for attack explosion. The information related IP address includes country name, city name, region or state name, zip or postal code, latitude, and longitude. To record IP address related information, this module utilize readymade API provided by IPInfoDB which contains open-source version of database regarding IP address geolocations. The information about attacker’s IP address is then recorded into related data tables for reporting purpose.

Report monitoring becomes essential part to track various vulnerabilities of categorized attacks as well as to analyze these attacks. This module generates various types of scheduled reports, key indicator reports, demand reports, exception reports and drill down reports. These reports are send to users who are listed in UserTable according to their priority set as level. Report monitoring module sends alert about various reports to user groups like: Level1 users get alert reports on daily basis, Level2 users get alert reports on weekly basis and Level3 user get alert reports on monthly basis. These reports details can be send via e-mail and text message.

To track and analyze attack related information, the database called AttackWatchDb is also created and maintained which contains following data tables:

- SiteURLTable that contains list of URLs supported by given web application.
- AttackRecordTable that contains information about various categorized attacks.
- IpLocTable that contains information about geolocation of given IP address.
- UserTable that contains contact details of user to whom attack information is need to be transfer periodically.

The methodology described is implemented and results are obtained for the purpose of data analysis and interpretation. The methodology is implemented which is containing seven modules. The vulnerability test is performed using sites in two folds. The first fold is pertaining to the access of sites without applying the developed algorithm. The second fold is pertaining to the access of sites by applying the developed algorithm.

For the purpose of vulnerability and performance evaluation test, 120 sites randomly selected which are highly ranked commercial sites based on the popularity index of various popularity search rank index. The vulnerability and performance test (VPT) is observed as a whole as well as on individual module based.

The test sites are grouped into four groups using random selection approach. Each group is tested for both cases. As a result, total four groups consisting of 30 websites were assessed. The groups are named as from Group-A to Group-D.

Six vulnerability test case are listed as follow. The VPT results are depicted in the following list and their results obtained are shown in graphical forms.

V001: Vulnerable to XSS

V002: Vulnerable to Sessions & Cookies

V003: Vulnerable to DOM based attacks

V004: Vulnerable to SQL injection

V005: Vulnerable to divert to other webpage or sending

data to other site

V006: Vulnerable to database

The VPT test conducted on Group-A. The result depicts those 70% sites of Group-A observed to be vulnerable to XSS. 73.33% sites of Group-A were observed to be vulnerable to Session and Cookies. In case of DOM based attacks, 86.67% sites of Group-A were observed to be vulnerable. Vulnerability to SQL injection was observed to be 56.67% in case of sites of Group-A. Vulnerability to divert to other webpage or sending data to other site was observed to be 76.67% and vulnerable to database was observed to be 66.67% in case of sites of Group-A.

Out of six VPT, it was observed that two sites were matching to all six modules. 50% of sites were vulnerable to five modules and 20% of sites were vulnerable to four modules. It is also important to observe that more than 76.66% sites were observed to be vulnerable to more than four vulnerable test modules. It is also important to note that 16.67%, 3.33% and 3.33% of total tested sites were found vulnerable to three, two and one vulnerable modules respectively. Here, highest vulnerability was observed in case of V003 VPT and lowest was observed in case of V004 which is measured as 86.67% and 56.67% respectively. It is observed that highest vulnerability was found in case of DOM based attacks and lowest was observed in case of SQL injection attacks.

The std. deviation is observed to be 0.8165, 1.2909, 1.2583, 0.9574, 1.2909 and 1.2909 for the VPT performance test ranging from V001 to V006 respectively. The range analysis result shows that for case of V001 and V004 the results are observed within the range interval of 2 whereas the results obtained for V002, V003, V005 and V006 are observed within the range interval of 3.

To analyze the pattern and ways of malicious script injected points, an approach called MIP-VPT (Malicious script Injected VPT) is used. It is observed that there are broadly three possible ways of SQL injection. Malicious script Injected Points are tested by considering the observed total attacks, out of total 522 attacks

observed, 73 attacks were of category SQLI. In other word, 13.98% of attacks were based on SQLI. The major three categories of SQLI are having sub categories. Inband SQLI having sub category based on Error and Union. The second category of SQLI is Internal SQLI which is further subcategorized as blind boolean based and blind time based. The third category is of type Outband SQLI. Majority of SQLI VPT observations were observed are of type Inband SQLI. It is observed to be 51% of total SQLI VPT observations. The Internal SQLI VPT observations were observed to be of 38% of total observations and Outband SQLI observations are observed to be of 11%.

Further, it is observed that Error based SQLI share the highest vulnerability among all SQLI attacks. It is observed to be 32.88% of total SQLI vulnerability. Blind-boolean based SQLI is observed to be 21.92%. Union Based, Outband SQLI and Blind Time based SQLI are observed to be 17.81%, 10.96% and 16.44% respectively.

Considering all four groups and the observations obtained depicts that total 82 observations found to be Vulnerable in case of Cookies and Session stealing out of 120 observations. This shows that 68.33% of tested sites were found vulnerable to Cookies and Session Stealing. The observation also shows that among the four Groups, the range is observed to be 3 and having median value of 20.5 and Standard Deviation observed to be 1.291.

The observations of cookies and Session Stealing made for all four groups shows that the SideJacking is observed to be highest compared to other two categories. The result also shows that the SideJacking is observed to be 59.76%, fixing session and predictable SessionId is observed to be 24.39% and 15.85% of total observed results. It is observed that majority of Cookies & Session Stealing vulnerability falls in the category of SideJacking which is nearly 60% of total Cookies & Session Stealing Analysis observations. Similarly Fixing sessi on and Predictable SessionId are observed to be 24% and 16% respectively.

Vulnerability of Redirections and Database are also tested for two broad categories: Persistent or stored XSS vulnerability and Non-Persistent this is also known as reflected XSS type. As per the observations obtained using VPT, it is observed that diverting to other webpage or sending data to other site is observed in case of 75% of observations. Similarly, 71.76% of results are observed where the Database was found to be vulnerable. It is observed that Persistent type of vulnerability in case of diverting to other webpage was observed to be 43.33% cases; whereas in case of Non-Persistent Type of vulnerability is observed in 56.67% cases. Vulnerable to database attack was observed among 86 websites from the sample tested sites out of 120 datasets. This is 71.67% vulnerability observations obtained. For Database VPT test, the Rights violations were observed to be in case of 15.12% cases from the observations. Patching was observed in case of 22.09% observations and Blocking Bypass was observed in case of 62.79% observations.

It is important to assess the association ship and correlation among the six vulnerability performances. These associations are required to assess the correlation among various tests and their performance analysis. It is observed that strong Association is observed among the VPT1 and VPT3 which is measured as 63.33%. This depict that in 63.33% cases, when Vulnerability observed for VPT1, it was also in case of VPT3. It is also observed that association ship among the VPT2 and VPT3 is observed to be 60.00%. It can be depicted as in 60% of cases, when Vulnerability is observed for VPT2, it is also vulnerable in case of VPT3. It is also observed that association ship among the VPT3 and VPT5 is observed to be 70.00%. It can be depicted as in 70% of cases, when Vulnerability is observed for VPT3, it is also vulnerable in case of VPT5. Here, one more observation can be highlighted that all categories have their strongest association ship with VPT3. Among all vulnerabilities, the strongest was observed with VPT3 which was observed to be 53.33% and 63.33% with VPT4 and VPT6 respectively. The negative strong correlation was observed among the VPT1 and VPT4. It means that when VPT1 is increased, there is less possibility of having VPT4.

Datasets observed for the vulnerability are tested using the framed algorithm in different phases. The algorithm implementation was applied in three phases.

Without implementing the algorithm at any level out of the six levels, the result depicts as observed with count of 21 sites in case of group-1 which measures to be 70% vulnerability and 30% level of accuracy at VPT1 level, count of 22 sites in case of group-1 which measures to be 73.33% vulnerability and 26.67% level of accuracy at VPT2 level, count of 26 sites in case of group-1 which measures to be 86.67% vulnerability and 13.33% level of accuracy at VPT3 level, count of 17 sites in case of group-1 which measures to be 56.67% vulnerability and 43.33% level of accuracy at VPT4 level, count of 22 sites in case of group-1 which measures to be 73.33% vulnerability and 26.67% level of accuracy at VPT5 level and count of 20 sites in case of group-1 which measures to be 66.67% vulnerability and 33.33% level of accuracy at VPT6 level. Performance pertaining to vulnerability is significantly improved after implementing algorithm.

Analysis is based on the script results observed and obtained from four groups without implementing algorithm and by applying algorithm. The Comparison yield the performance analysis. Comparative observations for all four groups among WA (Without Applying Algorithm) and AA (Applying Algorithm). In case of XSS attack, 14 scripts out of 82 scripts were detected in case of WA which is having 17.07% rate. However, in case of AA it is observed that, 80 scripts were detected out of 82 and detection rate is 97.56%. In case of CSS based script the WA detection rate is 7 out of 37 scripts, whereas in case of AA, the performance rate is 36. This shows 18.91% of detection rate for WA compared to 97.29% detection performance in case of AA. In case of HTML based script the WA detection rate is 7 out of 19 scripts, whereas in case of AA, the performance rate is 19. This shows 36.84% of detection rate for WA compared to 100% detection performance in case of AA. Results of JavaScript Parsing based script shows that WA detection rate is 0 out of 14 scripts, whereas in case of AA, the performance rate is 13. This shows 0% of detection rate for WA compared to 92.86% detection performance in case of AA. Others categories of XSS attack based scripts shows that WA detection rate is 0 out of 12 scripts, whereas in case of AA, the performance rate is 12. This shows 0% of detection rate for WA compared to 100% detection performance in case of AA. XSS using HTML quote Encapsulation based scripts shows that WA detection rate is 0 out of 24 scripts, whereas in case of AA, the performance rate is 22. This shows 0% of detection rate for WA compared to 91.22% detection performance in case of AA.

URL String Avoidance based scripts shows that WA detection rate is 0 out of 14 scripts, whereas in case of AA, the performance rate is 13. This shows 0% of detection rate for WA compared to 92.85% detection performance in case of AA.

To analyze the performance of implementation of Algorithm compared to non-implementation of algorithm based on its performance outcome on Brower. For the reason of this purpose, five different Browsers are used. On these all browsers, 120 scripts which are vulnerable to XSS, Session & Cookies, DOM based attack, SQLi, Divert to other webpage and vulnerable to Database are assessed. Out of total 120 observations, without implementation of Algorithm, the observations for individually availed for every Browser type can be seen as: 35% of vulnerability identification rate in IE Browser, 26.83% of vulnerability identification rate in Firefox Browser, 49.17% of vulnerability identification rate in Chrome Browser, 24.17% of vulnerability identification rate in Opera Browser and 4.17% of vulnerability identification rate in NetScape Browser.

After implementing algorithm, it is also observed that out of 180 scripts execution observations found based on six categories of vulnerabilities, for each Browsers, following observations are` made: 98.89% of vulnerability identify in IE, observed to be 177 blocked scripts found in case of FireFox Browser. This is 98.33% of vulnerability identify in FireFox, 99.44% of vulnerability identify in Chrome, 96.11 % of vulnerability identify in opera and 96.87 % of vulnerability identify in NetScape.

It is observed that for IE, the increase in % identification rate is 63.89 and it shows enhancement of performance by 282.54%. Similarly, in case of Firefox, the increase in % identification rate is 72.5 and performance enhancement is shown to be 380.68%. Chrome performance is observed to be increased by 50.27 and performance enhancement is observed to be 202.24%. Opera performance is enhanced by 71.94 and performance enhancement is observed to be 397.64%. Finally, Netscape performance which was observed to be lowest among all is enhanced by 82.5 and performance enhancement is observed to be 682.22%.

Based on the observations obtained and shown in Chapter-4, the analysis is made and results are discussed as follows. The discussion is pertaining to the objectives of the research. Result is analyzed based upon various vulnerability performance test.

Considering the analysis of results obtained on applying the algorithm and the results obtained without applying the algorithm establish the overall scenario and significance of algorithm. The perspective of algorithm can be seen in preview of the framed objectives of the research.

- **Objective 1:** Based on the observations the result can be depicted that highest detection rate is in case of HTML based scripts by browsers without implementation of algorithm. It is also important to note that no detections are observed in case of JavaScript parsing based and other types of XSS attacks.

- **Objective 2:** Module-4 of algorithm is based on identifying the malicious script injected points. It identifies the possible threat and detects as well as restricts the injected scripts. On applying Algorithm it is observed that very high avoidance of vulnerability is observed. The accuracy level result is observed 100%.

- **Objective 3:** Cookies and session stealing using XSS which leak victim's sensitive information is prevented and controlled when the Module-2 is implemented with the accuracy of 100%.

- **Objective 4:** Considering observations of all Groups, implementation of Module-5 of algorithm protects the victim who is redirected to another page unwillingly or their inputted data is sent automatically to target site which has been restricted with 100% accuracy.

- **Objective 5:** Vulnerability to different attacks are prevented using the modules applied in appropriate sequence. Module-1 is preventing the

Persistent and Non-Persistent XSS attacks. DOM-based attacks are prevented using Module-3 which try to identify the attacks which are DOM based. XSSQLi pattern attacks are prevented through implementation of Module-4. It extracts parameters of SQL queries and detect XSS based SQLinjection. Application of these modules and observed results shows the effective implementation of modules. Considering all four group analysis, it is observed that the algorithm % vulnerability was observed to be 60.83% when the algorithm is not implemented. In case of implementing the algorithm, it is observed that the vulnerability is reduced to 1.37%.

6.2 Recommendations

Following are some counter-measures which should be taken in consideration to protect against various cross-site scripting attacks:

- Web developers need to set some security options as configuration setting in the web configuration file of given web site. These setings are define as follow:
    - Restrict sketching of HTTP request so that cross site tracing attack can be prevented.
    - Enable HttpOnly option for cookie with secure flag.
    - Setting of X-Frame-Options with value SAMEORIGIN helps to defend against clickjacking.
    - Apply protection against cross-site scripting attacks by setting X-XSS-Protection to mode-block.
    - Restrict all web request from HTTP 1.0 because it has security breaches of session hijacking. Use modrewite module to allow HTTP 1.1 only.
    - Always set session timeout option.
    - Use SSL services with SSL cipher wich transfer data in encrypted form.

- Don’t allow directory listing throgh the browser.
- Limit HTTP methods. Disable methods like options, put, trace, delete, connect, etc.
- Change value of session every time with random values so that it can prevent from guessing attack techniques or brute fource attacck.

- If web page contains form which demands input from user then web devlopers should have to restrict data type and size of each field of form, specially file upload field so that the attacker/hacker is unable to insert malicious script as input. And they also provide proper validations for critical data.

- To protect against various forms of cross-site scripting attacks, web devlopers have to call web service routine on form submit event and have to implement suggested algorithms as part of web page code.

- Web users have to install plug-ins to protect against DOM-based XSS attacks.

- Web users must update their softwares time to time because software upgradation always contains solution for bugs which provides protection against vulnerabilities.

- Make habit to check URL string of particular web site in address bar of browser while surfinng to prevent from phishing.

- Always avoid clicking on suspicious links which is generally received via e-mails. It may contain hidden embedded malwares which gives control of your system to attacker/hacker.

6.3 Limitations of Proposed Framework

In this research work almost all major aspects of Cross-Site Scripting attacks have been covered. Still there are certain limitations of projected algorithms which have further scope of improvement. Following section lists some shortcomings of this proposed framework:

- These modules cannot fix up attacks performed through the use of plug-ins, ActiveX or flash object.
- It cannot detects DOM attacks injected in text node of DOM tree. Text node contains contents which are going to display on a web page. So it is difficult to recognize attack vector in text node.
- To protect data from unauthorized access, data needs to be stored in encrypted mode in database which increases storage space. It also slows down system's response time.
- This system is not able to fetch valid IP location information if attacker is connected through VPN. As stated earlier, attacker is situated at Pakistan's city and through the VPN, IP location indicates some location of US City.
- Here this system is not able to protect against XSS shell i.e. some malicious file which is uploaded through some file upload control of form.
- It does not protects against reflected XSS attacks which are delivered via email.
- Proposed framework only deals with XSS attacks and its various forms. Since new web application vulnerabilities and attacks can be cope up any point of time. In future addressing them will be obstacle.

6.4 Future Scope

As technologies are changing rapidly, the broad scope of futuristic work for the security aspect of WWW is always open. There are many hidden aspects of this research area that need to be further studied and explored.

- Graphical (GUI) editor can be introduce to insert new attack vector signature for validation check and also provides action to perform against it.

- JavaScript attacks that make use of the language's features such as pop-up windows for every mouse events or display multiple alert messages or redirecting user are annoying. Proposed framework can be further determine how many times a particular method call being invoked.

- IP tracking verification is incorporated which will verify fake IP addresses used through VPN and tries to locate exact and valid IP location of attacker.

6.5 Conclusion

The web world has become critical and integral part of a man's day to day life. Every service and product is now available at the clicks of your, which makes present generation technology obsessed. This online system helps the present human civilization to such a great extent that life without internet seems to be impossible. So the World Wide Web (WWW) becomes one of the valuable resources which provides communication channel and online facilities to the entire globe. These online systems are closely monitored and may be exploited by attackers/hackers for their personal gain. They are continuously looking for new techniques to find loop holes in the system using which they can create vulnerabilities in web applications. Cross-Site Scripting attacks are easy, simple but highly-damaging way for an attacker/hacker to perform malicious activities in web application to fulfill their desires. Cross-Site Scripting is listed as one of the top most vulnerabilities in the web application. Generally web application developers are not aware of various types of vulnerabilities which occur in web page and their consequences. So for them it may be difficult to provide security to protect and defend against such kind of vulnerabilities and their consequences. Moreover to make changes in existing web site in order to provide security is also time consuming and expensive process. So dealing with such kind of vulnerability in web application is very difficult task.

Therefore this research work proposes security mechanism which protects web application from such kind of Cross-Site Scripting attacks and its consequences. To conduct this research work, the framework called “Web Watch” is introduced which contains seven different modules namely: XSS Detector, Token Analyzer, DOM Tracker, XSSQLI Pattern Extractor, XSSR & XSRF Evaluator, Attack Handler and Report Monitoring. These modules protect web pages which are vulnerable to stored XSS attacks, reflected XSS attacks, DOM-based XSS attacks, XSSR attacks, XSSQLI attacks and CSRF attacks.

The methodology described is implemented and results are obtained for the purpose of data analysis and interpretation. Various vulnerability test are conducted from dataset which shows that 13.98% web applications contain malicious injection points, 68.33% web applications are vulnerable to cookies and session stealing, 75% web applications are vulnerable to web page redirection attacks and 71.76% web applications are having database vulnerabilities. From this vulnerability test, vulnerability performance association is measured which depicts that strong association ship between VPT3 and VPT5 is observed to 70.00%. All vulnerability categories have their strongest association ship with VPT3 which was observed to be 53.33% and 63.33% with respect to VPT4 and VPT6.

The performance of proposed methodology is verified using four measures namely: detection rate, accuracy, script based analysis and browser based analysis. This methodology is highly significance which gives detection rate from 96.67% to 100%, script based analysis gives 95.11% result, and browser wise analysis gives results from 96.11% to 99.44%. Accuracy of model is increased from 33.33% to 89.17%.