
# PARAMETER EXPLOITATION IN WEB APPLICATIONS & ITS PREVENTION

**A Project Report of Capstone Project - 2**

***Submitted by***

**ANKIT KUMAR**
**(1613101141 / 16SCSE101134)**

***in partial fulfillment for the award of the degree***
***Of***
**Bachelor of Technology**
**In**
**Computer Science and Engineering With Specialization of Computer Networking and Cyber Security**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the supervision of**
**Dr. Sanjeev Kumar Prasad**
**Associate Professor**
**School of Computing Science and Engineering**

**MAY - 2020**


# SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

Certified that this project report "**PARAMETER EXPLOITATION IN WEB APPLICATIONS & ITS PREVENTION**" is the bonafide work of "ANKIT KUMAR (1613101141)" who carried out the project work under my supervision.

**SIGNATURE OF HEAD**
Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
**Professor & Dean,**
**School of Computing Science & Engineering**

**SIGNATURE OF SUPERVISOR**
Dr. Sanjeev Kumar Prasad
Associate Professor
**Professor**
**School of Computing Science & Engineering**

# Abstract

A web application is a computer program that utilizes web browser and web technology to perform tasks over the internet. In recent years, millions of businesses use the internet as an effective communication channel which lets them exchange the information worth their target market and makes fast, secure transactions. However, there are many threats which are arising to gain this valuable information which is mainly done by various kinds of attackers who use different kinds of techniques for data thieves. So, the challenge is to provide security to various web applications and prevent the attacker to gain root shell access and admin passwords As the Internet becomes more and more complex, newly found vulnerabilities continue to develop and through web-based applications, these vulnerabilities are exploited. One of the most common styles of malicious attacks that come to mind is code injection.The unavailability and high price of automated scanners for detecting vulnerabilities in web applications which leads to defacement , hijacking , stealing data from servers creates a security problem for all businesses as well as government people. So making an affordable scanner is important.In this project we will develop a vulnerability scanner for finding vulnerabilities in web applications and provide information about its remediation.Cross site scripting (XSS), SQL Injection, File Inclusions are types of attacks on web pages and account as the unsafe vulnerability existed in web applications. Once the vulnerability is oppressed, an intruder advances intended access of the authenticated user's web-browser and may Perform session-hijacking, cookie-stealing, malicious redirection and malware-spreading. As prevention against such attacks, it is essential to implement security measures that certainly block the third party intrusion. Vulnerabilities of websites are exploited over the network through web request using GET and POST methods. In this project, we are focusing on injection, detection and information about preventing these attacks. The unavailability and high price of automated scanners for detecting vulnerabilities in web applications which leads to defacement, hijacking, stealing data from servers creates a security problem for all businesses as well as government people. So making an affordable scanner is important. In this project we will develop a vulnerability scanner for finding vulnerabilities in web application and provide information about its remediation.

# TABLE OF CONTENTS

# LIST OF TABLE

# List of Figures



## List of Symbols, Abbreviations and Nomenclature

| ABBREVIATION | ILLUSTRATION |
|---|---|
| XSS | CROSS SITE SCRIPTING |
| LFI | LOCAL FILE INCLUSION |
| RFI | REMOTE FILE INCLUSION |
| SQLI | SQL INJECTION |
| | |

# Introduction

**(i) Overall description**

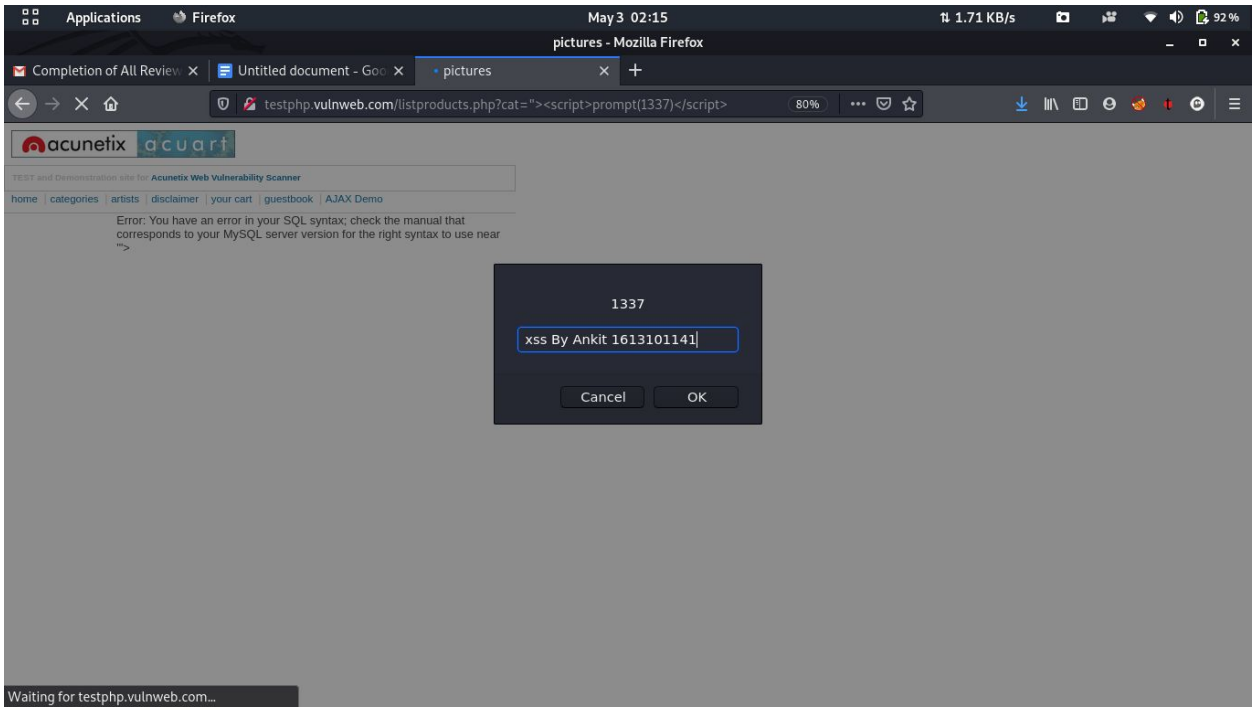
This project covers the following aspects of Web Application security i.e.Web Pentesting some famous Critical Vulnerability analysis and Detection like cross site scripting,local and remote file inclusion, os command injection and sql injection.

**1.1. Cross site scripting:**

Most websites today contain dynamic content which gives its viewers amore interactive and enjoyable experience. Instead of having a classic static website, a dynamic website is generated by two different types of interactivities: client-side scripting (used to change interface behaviors within a specific webpage)and server-side scripting (used to change the supplied page source between pages). In addition to creating a dynamic website, you are making yourself susceptible to a popular and very powerful security vulnerability that plain static websites are not. This threat is called cross-site scripting (XSS). XSS has come about as a result of effectively constructed coding within web based applications. Attackers direct their attention towards these vulnerabilities and insert malicious content into the client-side browser without the user's knowledge which allows an attacker to gain access to the user's personal information. Cross-site scripting (XSS) attack is considered as one of the top 10 web application vulnerabilities of 2013 by the Open Web Application Security Project (OWASP) . According to the Cenzic Application Vulnerability Trends Report (2013) Cross Site Scripting represents 26% of the total population respectively and is considered as the top most first attack. Two recent incidents highlighted the severity of xss vulnerability are Apple Developer Site (July18,2013) and Ubuntu Forums (July 14 and July 0, 2013) . Cross Site Scripting attack carried out using HTML, JavaScript,VBScript, ActiveX, Flash, and other client-side languages

**->Account Hijacking for identity theft**
**->Cookie theft/poisoning to acquire sensitive information**
**-> Conduct phishing attacks**
**-> Gain free access to otherwise paid for content**
**->Spy on users web browsing habits**
**-> Change users settings**
**-> False advertising**

**-> Deface a website**
**-> Insertion of hostile content**




(fig-01)

Reflected xss on ‘cat parameter’

**1.2. Local File Inclusion:**

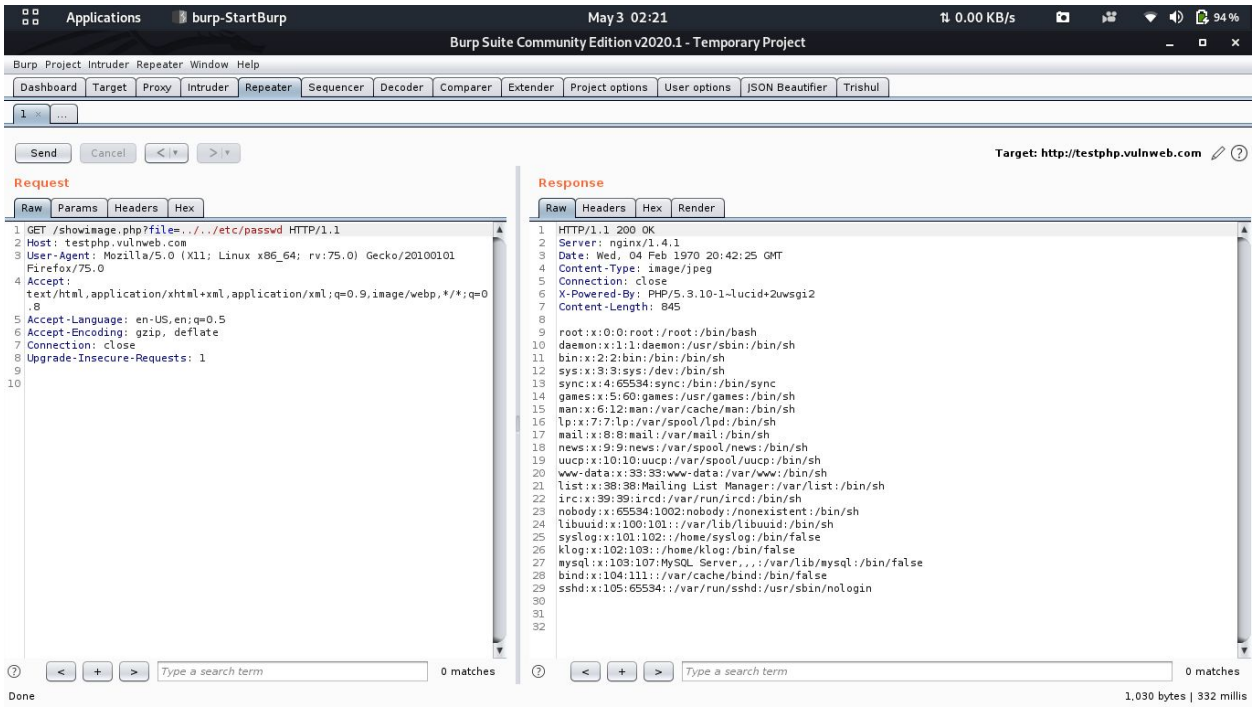
It is the process of including files that are already locally present on the server through the exploitation of vulnerable inclusion procedures implemented in the application. This application occurs ,for example when a page receives as input the path to the file that has to be included and this input is not properly sanitized, allowing directory traversal characters to be injected Since LFI occurs when path passed to “include” statement are

not properly sanitized, in a blackbox testing approach we should look for scripts which take filename as parameter.

http://vulnerable _host /preview.php?file = exp.html

This looks like a perfect place to try for LFI. If an Attacker is lucky enough and instead of selecting the appropriate page from the array by its name, it is possible to include arbitrary files on the server.Typical proof of concept would be to load password file-

http://vulnerable _host /preview.php?file = ../../../../etc/pwd
If the above mentioned conditions not an attacker would see something like the following:-
Root:x:0:0:root:/bin/bash
Bin:x:1:1:bin: /bin : /sbin :/sbin /nologin

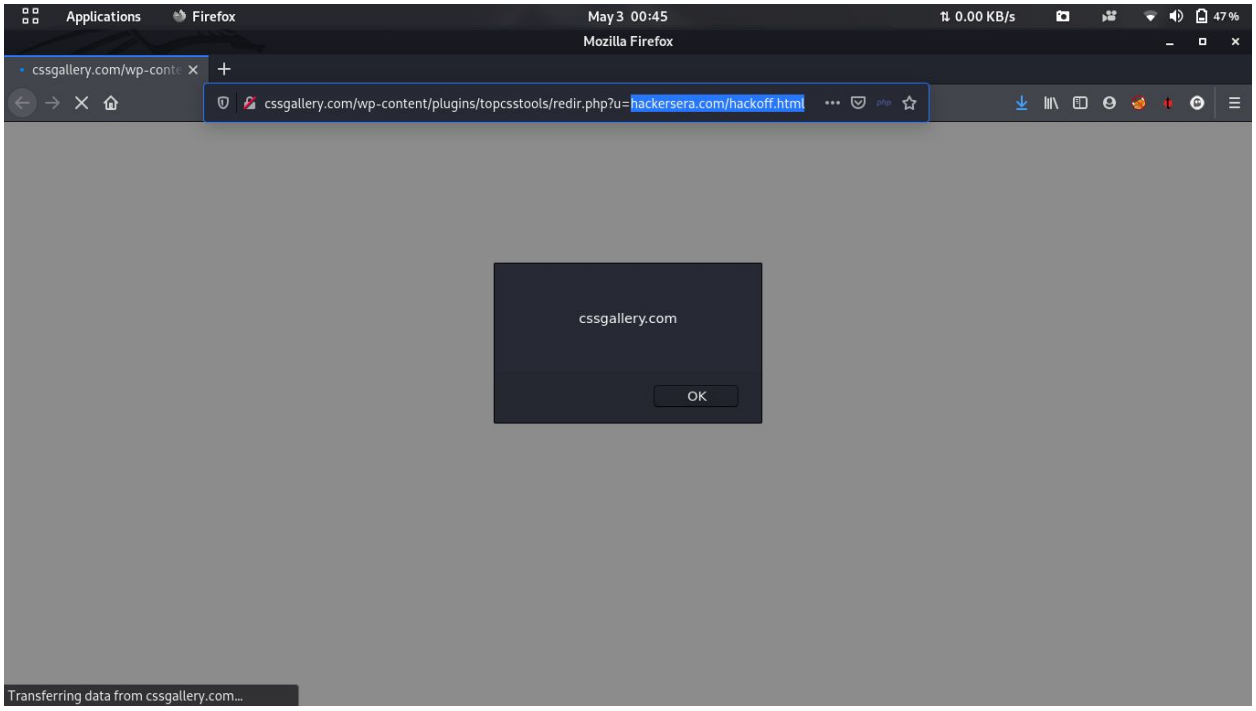



(fig-2)

Local File Inclusion on ‘File’ parameter

### 1.3. Remote File Inclusion (RFI):

RFI is an attack targeting vulnerabilities in web applications that dynamically reference external scripts. The perpetrator's goal is to exploit the referencing function in an application to upload malware (eg. Backdoor shell) from remote URL located within a different domain. In both cases, successful attack results in malware being uploaded to the targeted server. However unlike RFI LFI result aim to exploit insecure local file upload functions that fail to validate user supplies/controlled input.



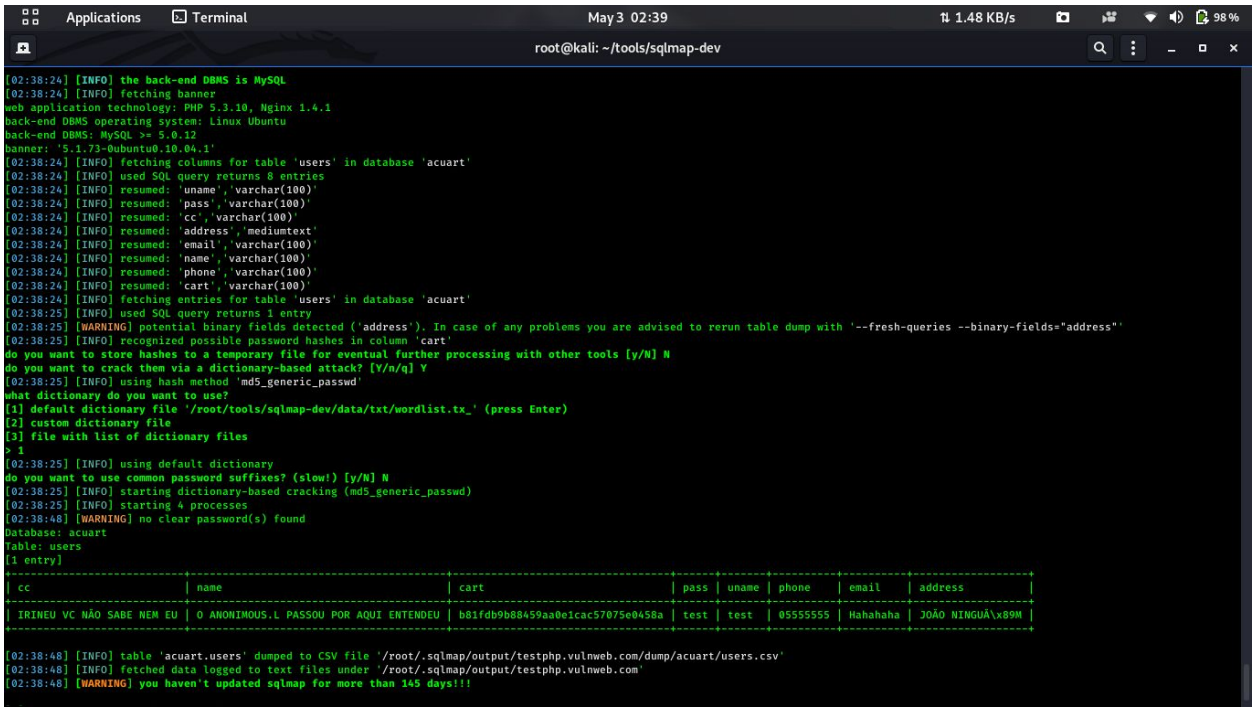

(fig-3)

Remote FIle Inclusion On 'u' parameter

### 1.4. Sql Injection:

SQL Injection is one of the most common threats to a database system in which the attacker adds SQL statements to an application form input box, to gain access the resources or make changes to data stored into the database. Lack of input validation in applications causes attackers to be successful. In an SQL Injection attack, the

attacker injects a string input through the application, which changes or manipulates the SQL statement to the attacker‟s advantage. An SQL Injection attack can harm the database in various ways, such as unauthorized manipulation of the database, or retrieval of sensitive data. It can also be used to execute system level commands that may cause the system to deny service to the application. This issue is very risky because it can cause data loss or misuse of data by parties who are not authorized and as result the functionality and confidentiality are destroyed.



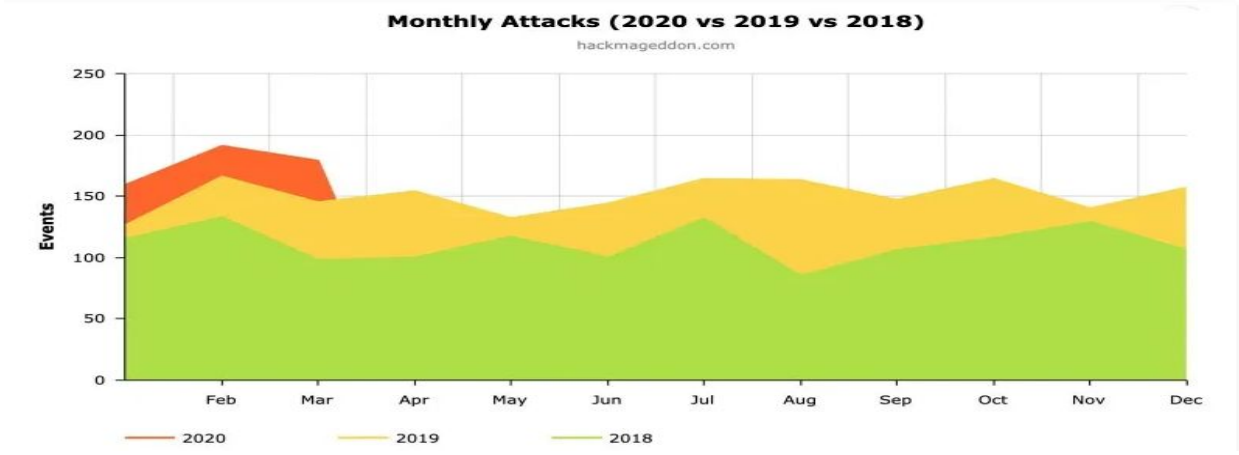

(fig-4)
Sql injection on ‘cat’ parameter

### 1.2. Purpose:

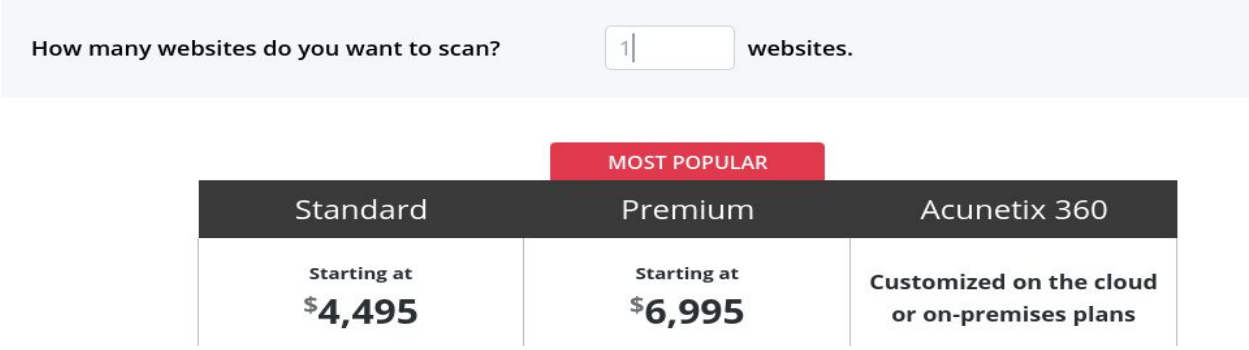
The unavailability and high price of automated scanners for detecting vulnerabilities in web applications which leads to defacement , hijacking , stealing data from servers creates a security problem for all businesses as well as government people. So making an affordable scanner is important.In this project we will develop vulnerability scanner for finding vulnerabilities in web applications and provide information about its remediation.

### 1.3.Motivations and scope:

Developers, product owners, AppSec, and security engineers can use this information to better understand application security threats, adjust security controls, and improve their security posture. Through reading this report on a monthly cadence, AppSec teams can gain a better understanding of the possible types and origins of attacks and attackers.




(fig-5)




(Fig -6)
highly expensive price of tools

# LITERATURE SURVEY

There are a number of researches done on various web vulnerabilities which come under Semantic URL, Cross-Site scripting, Cross-Site Request forgery, etc. In this our project comes under Semantic URL means such attacks involve a user modifying the URL discovery mode to perform various actions which are not originally planned to be handled by a server. We studied various vulnerabilities such as RFI, LFI, SQLi, Cross-Site Scripting.

The web application users enter tags in the input field along with other data for formatting purposes. The attackers can misuse it to enter malicious content like viruses, Trojans and worms. There are two possible solutions for prevention from XSS which includes Filtering and Validation of user inputs:
• Negative security model which filters inputs that match with the blacklist.
• Positive security model that specifies the expected inputs from the user

**3. Research Methodology:**

| Sr.No. | Traditional Research Methodology | 5-P-Model |
|---|---|---|
| 1 | Research Design | Proof Of Concept |
| 2 | Hypothesis Formulation | Point Of View |
| 3 | Method of Data Collection | Proof Of Concept |
| 4 | Scope Of Study | Point Of View |
| 5 | Testing Of Hypothesis | Pilot |

**Table Literature Review**

- **Point Of View/Analysis:** In the first stage of prototype development functionality of the subject was defined and the hierarchy of steps to be followed was outline.

- Proof of Concept/Design: This was the second stage of prototype building and once the foundation was laid by the function matrix, a functional architecture was built on that foundation to define active flow.
- Prototype/Develop: With the function Matrix and functional architecture in place a prototype was developed in this stage.
- Pilot/Testing: This stage was marked by the testing of the prototype and refining, to produce a more scalable software solution.
- Package/Release: Release of the software to the market with a predefined business model and earnin of the revenue from the same

**4. Software Requirement Specification:**

**4.1 PROJECT SCOPE:**

• The software will scan the full web application.
• The main focus will be on finding vulnerabilities on parameter based URLs.
• It will show information about vulnerable URL and its vulnerability and gives the solution to fix it.

**4.2 USER CLASSES AND CHARACTERISTICS:**

●Users have websites URL, RAM, and have software installed.
●Users have power to start scan and stop scan.
●Users have different priorities so they have been given different options to scan.

**4.3 Assumptions and Dependencies:**

●User must require the computer system
●User has to install the application on his system.
●Users must require high speed internet connection.
●Software automatically scans all the webpages.

**5 FUNCTIONAL REQUIREMENT**

**5.1 SYSTEM FEATURE:**

❖Continuous Crawling:
Continuous crawling improves the performance for faster discovery of web pages.

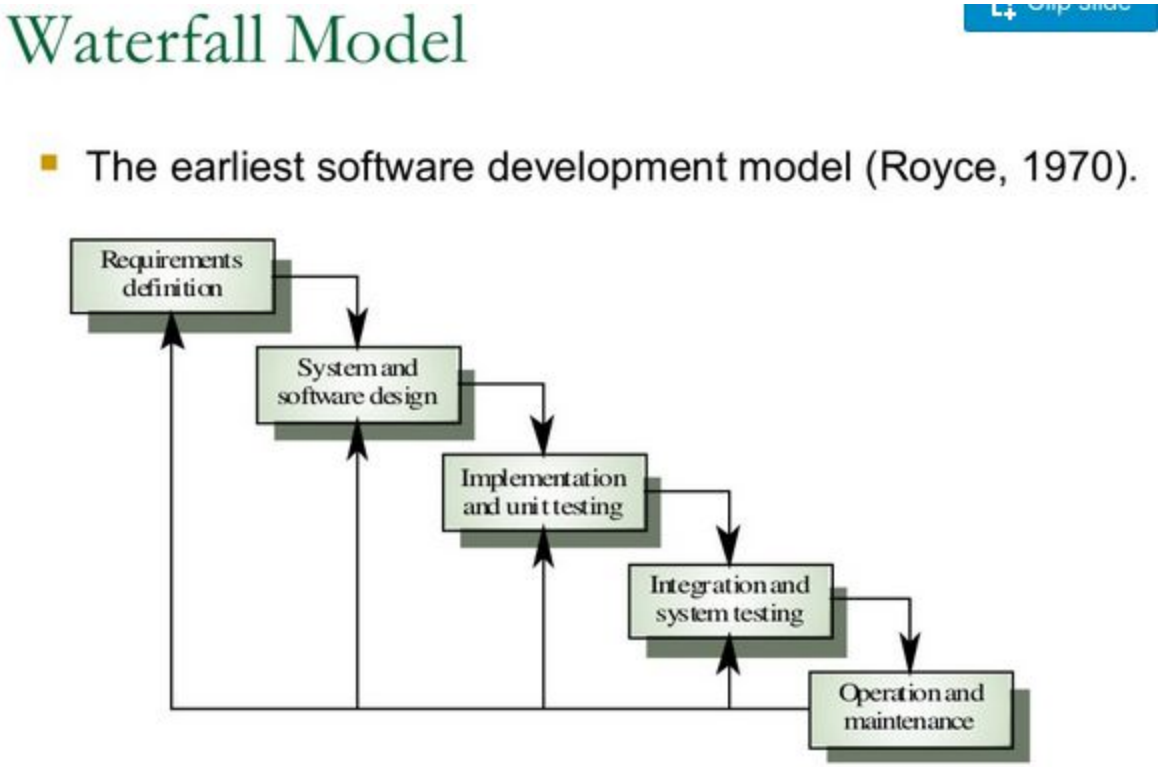
❖ Scanning options:
canning options help users to scan a website for a particular vulnerability or he can use a full scan option to scan for all types of vulnerabilities.

❖ Logs:

Logs help users to find out which payloads where tested and which URLs are tested. Vulnerable URLs appear in different colors.

**6. ANALYSIS MODELS:**

We are using a waterfall model for our project.




(fig-7)

Waterfall SRS model

**6.1. Requirement gathering and analysis**:

In this step of waterfall we identify what are various requirements are need for our projects such as software and hardware required, database, and interfaces.

### 6.2. System Design:

In this system design phase we design the system which is easily understood for end user i.e. user friendly. We design some UML diagrams and data flow diagrams to understand the system flow and system module and sequence of execution.

### 6.3. Implementation:

In the implementation phase of our project we have implemented various module required of successfully getting expected outcomes at the different module levels. With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

### 6.4. Testing:

The different test cases are performed to test whether the project modules are giving expected outcomes in assumed time. All the units developed in the implementation phases are integrated into a system after testing each unit. Post integration the entire the system is tested for any faults and failures.

### 6.5. Deployment of System:

Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

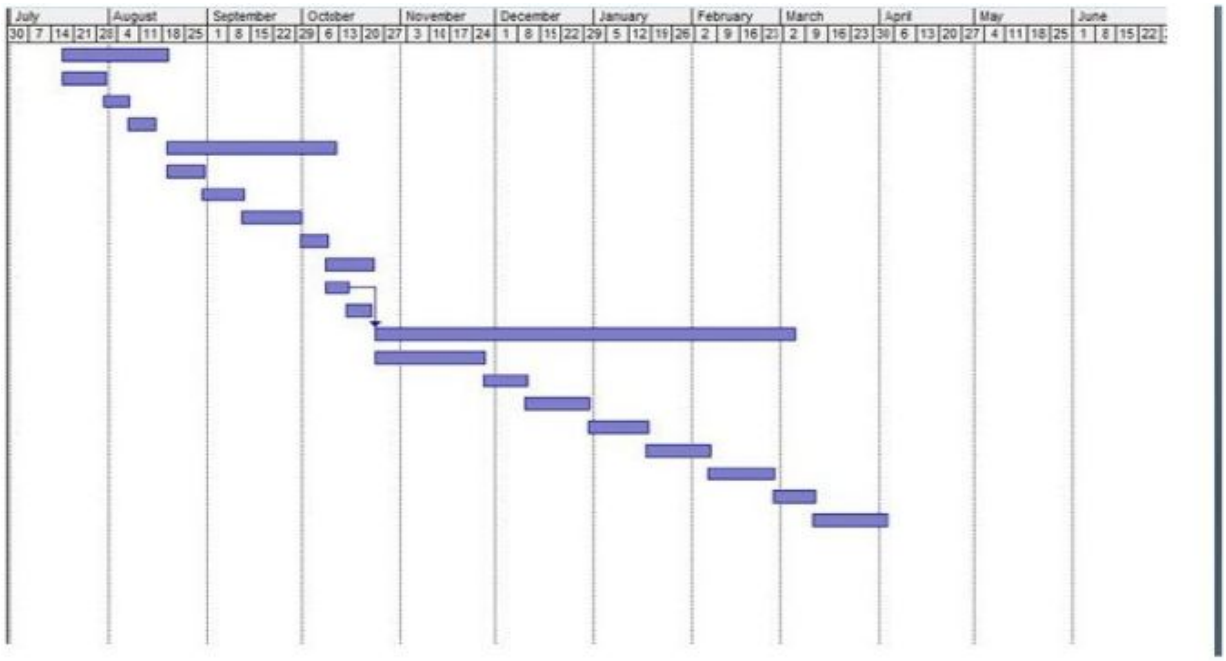
### 6.6. Maintenance:

There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. All these phases are cascaded to each other in which progress is seen as flowing steadily downwards like a waterfall through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and i

## 7 Architecture Diagrams:

### 7.1. Time Chart:

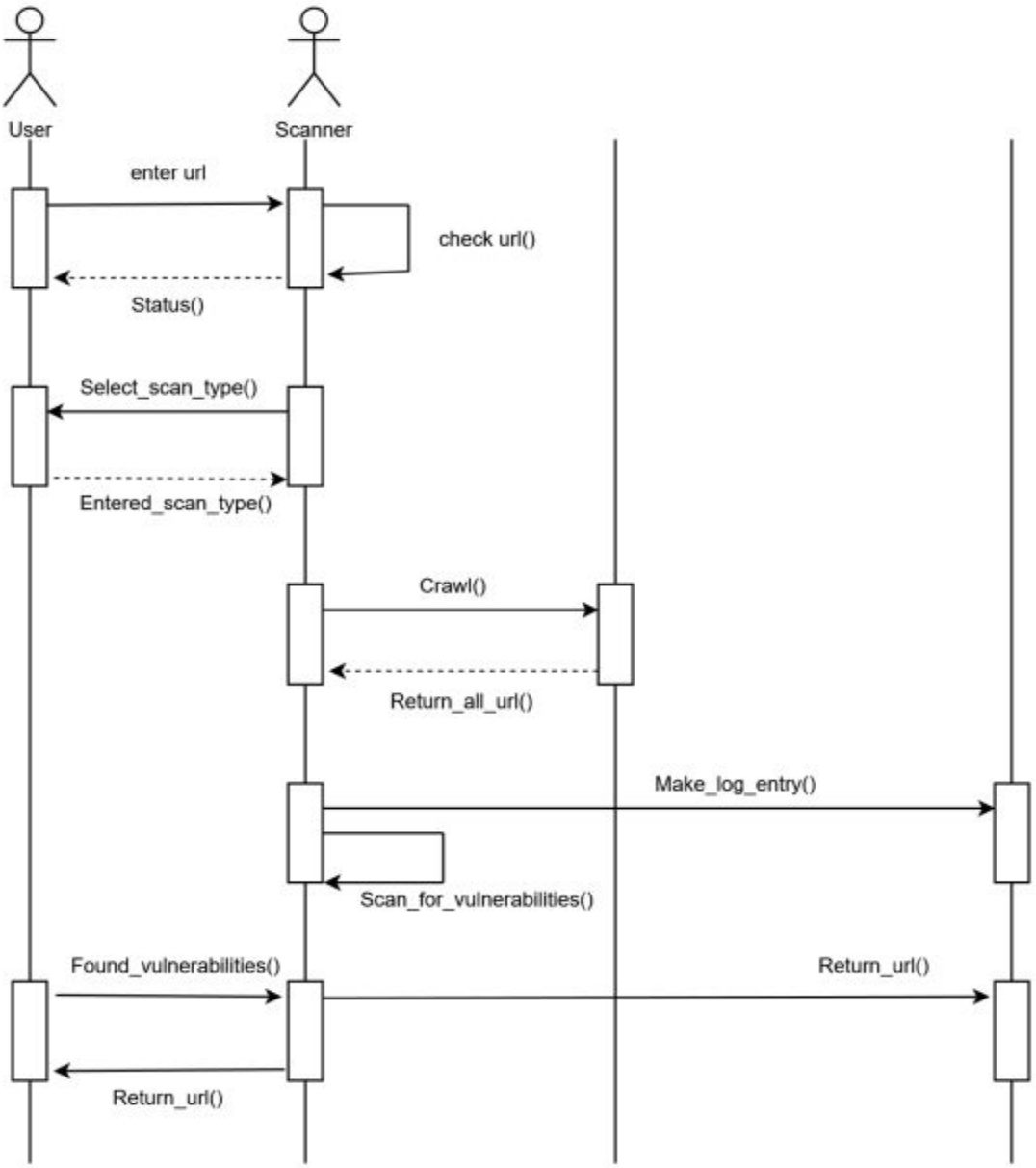
A *time chart* is a statistical display used to examine trends in data over time . Time charts show time on the *x*-axis (for example, by month, year, or day) and the values of the variable being measured on the *y*-axis (like time to first iteration, total unit testing, or population size). Each point on the time chart summarizes all the data collected at that particular time.




(fig-8)

**7.2 Sequence Diagrams:**

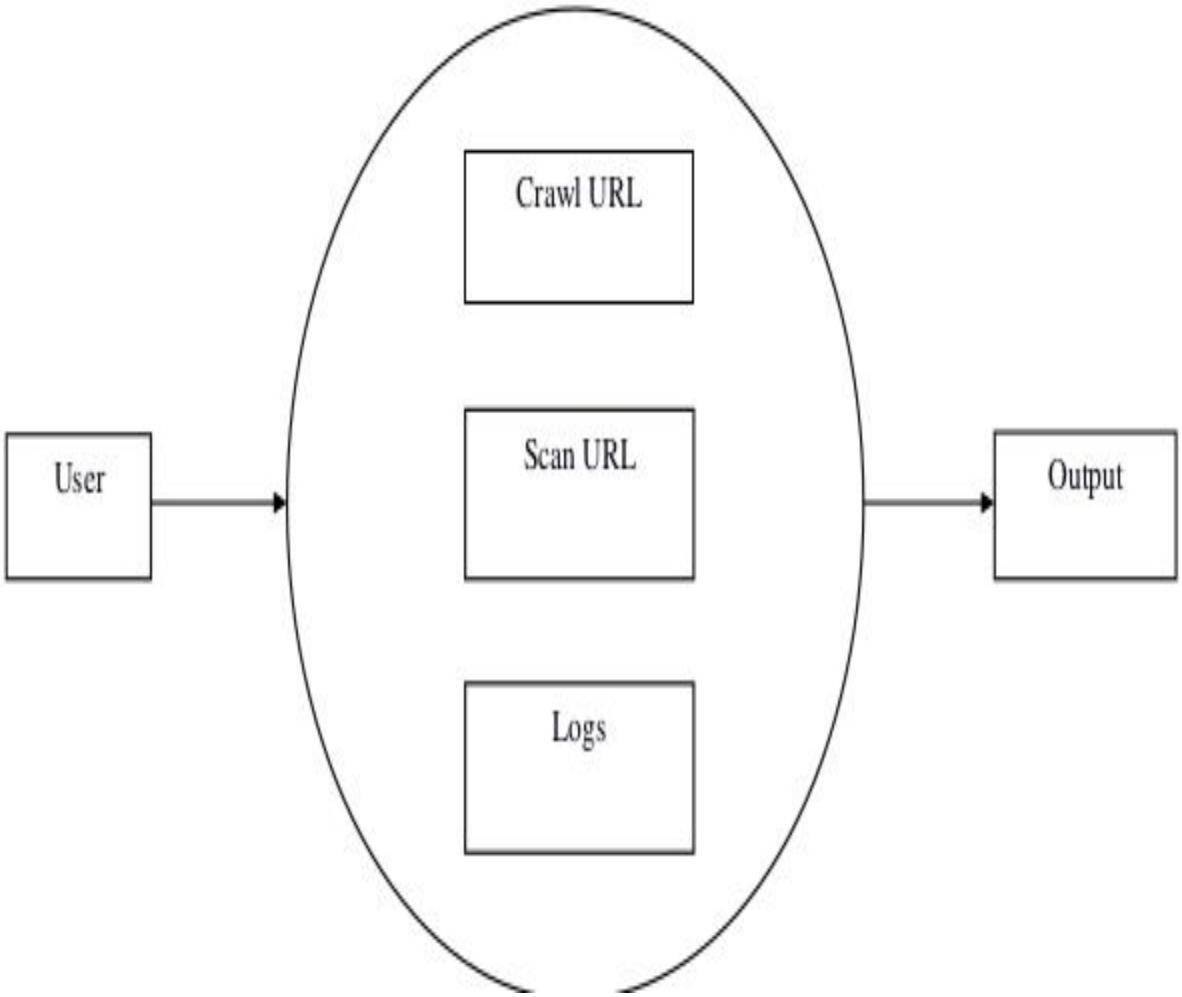
A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function



(fig-9)

**7.3 Data Flow Diagram:**

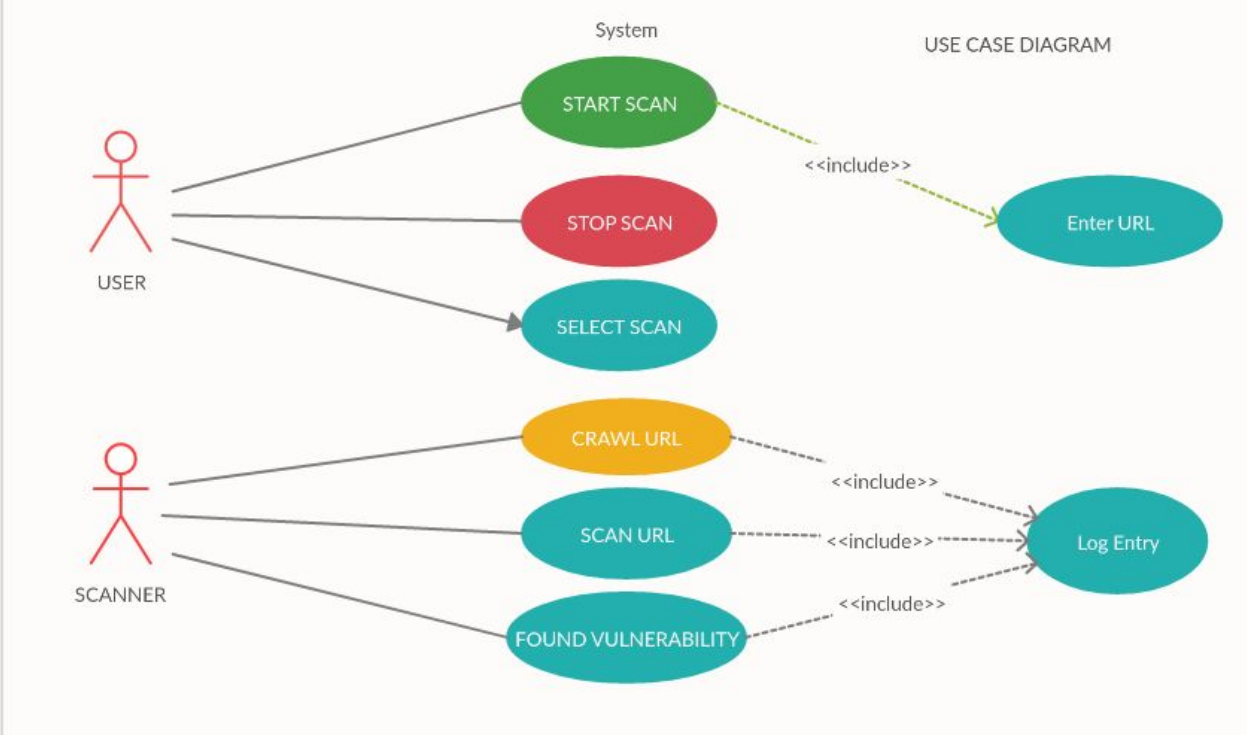
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.




(fig-10)

### 7.4 Use Case Diagram:

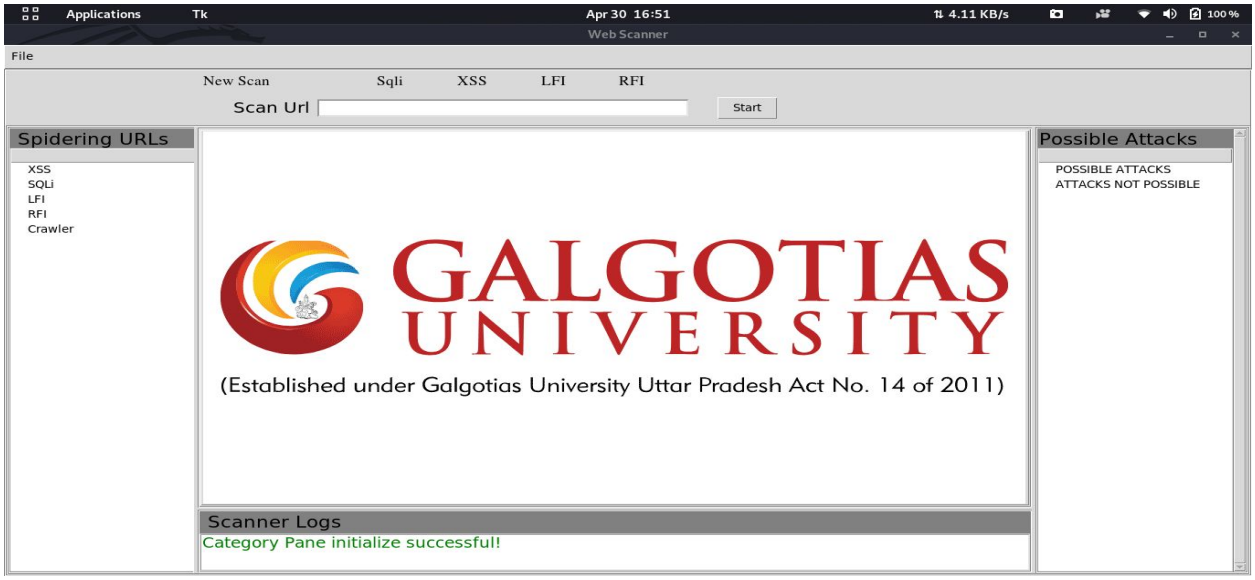
A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.




**(fig-11)**

## OUTPUT:

## USER INTERFACE OF TOOL:




**(fig-12)**

**Crawler Result Of Tool:**


**(fig-13)**

## CONCLUSION AND FUTURE WORK:

This report discussed web application security principles and fundamental information that can help us to prevent web exploits in our system. Web applications are considered the most exposed and least protected, thereafter vulnerable because the standards somehow are not focused on security but more on the server's needed functionality. Security threats are more common than before because the internet has become today's economy the most valuable tool for everyone. So there is indeed a need to protect our resources, data and user privacy information. As technology moves forward and brings new strategies, tools, models and methods to increase security levels, hackers will be part of this never end game.

The proposed system is developed to detect the vulnerabilities like SQLi, XSS, LFI in web applications and it will also provide information about remediation of vulnerable URLs and its vulnerability. Our formalization goes at the heart of the problem and captures seemingly different types of above mentioned vulnerabilities. The simple and effective strategy is meant to be cost effective and is openly targeted toward large commercial applications. The results of the proposed systems are satisfactory.

## References


- The Open Web Application Security Project, “OWASP Top ten project” https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- A. S. Christensen, A. Møller and M. I. Schwartzbach, “Precise analysis of string expression”, In proceedings of the 10th international static analysis symposium, LNCS, Springer-Verlag, vol. 2694, pp. 1-18.
- Afasana Begum and Md. Maruf Hasan,”RFI and SQLi based Local File Inclusion Vulnerabilities in Web Applications”,International Workshop on Computational Intelligence(IWCI),12-13 Dec2016.
- MirSamanTajbakhsh and JamshidBagherzadeh,”A sound framework for Dynamic prevention of Local File Inclusion ”, 7 th International Conference on Information and Knowledge Technology,2015.
- Shahriar, H.; Zulkernine, M., "S2XS2: A Server Side Approach to Automatically Detect XSS Attacks," IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011, pp.7-14, Dec. 2011
- https://www.owasp.org/images/1/19/OTGv4.pdf
- Karl D'silva,J.Vanajakshi,KNManjunath,SrikanthPrabhu"An Effective Method for Preventing SQL Injection Attack and Session Hijacking"Electronic ISBN: 9781-5090-3704-9
- Chen Ping"A second-order SQL injection detection method"Electronic ISBN: 9781-5090-6414-4
- Anastasios Stasinopoulos, Christoforos Ntantogian, Christos Xenakis “Commix : Detecting and Exploiting Command Injection Flaws” Department of Digital Systems,University of Piraeus
- P. Bisht and V. N. Venkatakrishnan, “XSS-GUARD: Precise dynamic prevention of Cross-Site Scripting Attacks,” In Proceeding of 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, LNCS, vol. 5137, (2008), pp.
- Mir Saman Tajbakhsh; Jamshid Bagherzadeh“A sound framework for dynamic prevention of Local File Inclusion” ISNB: 978-1-4673-7485-9
- P.S.Sadaphule, Priyanka Kamble,Sanika Mehre,Utkarsha Dhande,Rashmi Savant ”Prevention of Website Attack Based on Remote File Inclusion-A survey” InternationalJournal of Advanced Engineering and Research Development” e-ISSN : 2348-4470.