

# **Security Vulnerability Scanner**

Project report submitted in partial fulfillment of the requirement for the  
degree of Bachelor of Technology

In

**Computer Science and Engineering**

By

Vishal Pant (161257)

Under the supervision of

Mr. Prateek Thakral

to



Department of Computer Science & Engineering and Information  
Technology

**Jaypee University of Information Technology Waknaghat,  
Solan-173234, Himachal Pradesh**

## Candidate's Declaration

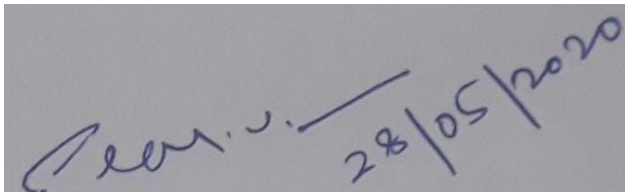
I hereby declare that the work presented in this report entitled **SECURITY VULNERABILITY SCANNER** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2019 to May 2020 under the supervision of **Mr. Prateek Thakral**, Assistant Professor (Grade-II), Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Vishal Pant, 161257

This is to certify that the above statement made by the candidate is true to the best of my knowledge



Mr. Prateek Thakral

Assistant Professor (Grade-II)

Computer Science & Engineering and Information Technology

Dated:

## **Acknowledgment**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to **Mr. Prateek Thakral** for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would like to express our gratitude towards our parents and Jaypee University of Information Technology for their kind cooperation and encouragement which helped us in the completion of this project.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

# TABLE OF CONTENTS

<b>CERTIFICATE.....</b>	<b>i</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>ii</b>
<b>LIST OF FIGURES.....</b>	<b>iv</b>
<b>LIST OF TABLES.....</b>	<b>v</b>
<b>ABSTRACT.....</b>	<b>vi</b>
<b>1. Chapter-1 PROJECT OBJECTIVE</b>	
1.1 Introduction.....	1-8
1.2 Objective .....	9
<b>2. Chapter-2 LITERATURE SURVEY AND METHODOLOGY</b>	
2.1 Literature Survey.....	10-22
2.2 Methodology.....	22-23
<b>3. Chapter-3 SYSTEM DEVELOPMENT</b>	
3.1 Python.....	24-28
3.2 Shell Scripting.....	29
<b>4. Chapter-4 ALGORITHM</b>	
4.1 Algorithm.....	30-31
<b>5. Chapter-5 TEST PLAN</b>	
5.1 Data Set.....	32
5.2 Test Setup.....	32-33
<b>6. Chapter-6 RESULT AND PERFORMANCE ANALYSIS</b>	
6.1.Result.....	34
6.2.Performance Analysis.....	35
<b>7. Chapter-7 CONCLUSIONS</b>	
7.1 Conclusion.....	36
7.2 Future Scope.....	36
<b>REFERENCES.....</b>	<b>37-39</b>
<b>APPENDICES.....</b>	<b>40</b>

## LIST OF FIGURES

<b>Figure</b>	<b>TITLE</b>	<b>PAGE NO.</b>
Figure 1.1	Confidentiality, Integrity, and Availability	3
Figure 1.2	Symbols used for a secure application	9
Figure 2.1	Injection Overview	11
Figure 2.2	Sensitive Information Exposure	12
Figure 2.3	XXE Attack methodology	13
Figure 2.4	Broken Access Control	14
Figure 2.5	XSS POC	15
Figure 3.1	Python version	25
Figure 3.2	Installation of BeautifulSoup library	26
Figure 3.3	How to install requests library	27
Figure 3.4	Explanation of Scrapy framework	28
Figure 3.5	Existence of shell in OS	29
Figure 5.1	How to install python	32
Figure 6.1	Screenshot displays the scanning progress on <a href="https://appsecure.security">https://appsecure.security</a>	34
Figure 6.2	Graph depicting User vs Scanner response time	35

## LIST OF TABLES

<b>Figure</b>	<b>TITLE</b>	<b>PAGE NO.</b>
Table 1.1	List of SANS top 25	20-22

## **ABSTRACT**

In recent years a lot of web applications have been released in the world. At the same time, cyber attacks against web application vulnerabilities have also increased. In such a situation, it is necessary to make web applications more secure. However, checking all web vulnerabilities by hand is very difficult and time-consuming. Therefore, we need a web application vulnerability scanner. In this work, we develop an automated web application security vulnerability scanner which helps security researcher and web developers to identify security vulnerabilities and fix them. The designed tools also identify the hidden directories and files from the server.

# Chapter-1

## PROJECT OBJECTIVE

### 1.1 Introduction - Basics of Information Technology

A fundamental comprehension of data security can assist you with evading pointlessly leaving your product and destinations unreliable and powerless against shortcomings that can be misused for monetary benefit or different malevolent reasons. This report can enable you to realize what you have to know. With this data, you can know about the job and the essential aspect of cybersecurity and past into the sending of your substance.

- **Confidentiality, Integrity, and Availability**

Depicts the essential security targets, which are completely principal to getting security

- **Vulnerabilities**

Characterizes the significant classifications of vulnerabilities and examines the nearness of vulnerabilities in all product

- **Dangers**

Quickly presents significant risk ideas

- **Security Controls**

Characterizes significant classes of security controls and talks about their potential weaknesses



### **1.1.1 Privacy, Integrity, and Availability**

#### **Confidentiality**

Confidentiality is the protecting of information in order to avoid people with malicious intent to obtain or leak sensitive information. We must have the proper tool to counter the issue of anyone disrupting this secure flow of data between systems. A failure to keep this information protected against the assailants can be devastating and must be avoided at all costs. Such attacks on regular basis must be disrupted and can't be restored to normal. The keys and security flags should be adequately protected in the source code. If the sensitive information of an employee are posted so his phone number, address and more private information could go in the hands of a malicious person. A powerlessness to keep this information guaranteed against the aggressors can be wrecking and ought to be sidestepped at all costs. Such attacks on customary reason must be upset and can't be restored to normal. The keys and security pennants should be acceptably made sure about in the source code. If the fragile information of a delegate are posted so his phone number, address and progressively private information could go in the hands of a harmful person

#### **Integrity**

Integrity suggests sheltering information from malicious users. Metaphorically, just individuals who are supported to do so can get to touchy information. The decision to terminate them and operators who are helping Integrity infers guaranteeing the validness of data. Suppose we have a popular E-Commerce site.. That would be a disregard for the reliability considering the sensitive data and the cost of maintaining it could be ravished. Another would be a cybersecurity failure we would face if a vulnerability is exploited leading to thousands of dollars of damage to us. e technique to un-reveal it. On the off chance that your bank records are posted on an open site, everybody can comprehend your cash related balance number, balance, and so on., and that data can't be deleted from their brains, papers, PCs, and different spots. For all intents and purposes all the basic security occasions revealed in the media today consolidate important difficulties of protection. The decision to end them and managers who are helping Integrity determines guaranteeing the validness of data. Accept we have a standard E-Commerce

site.. That would be a carelessness for the immovable quality considering the tricky data and the cost of keeping up it could be abused.

### **Availability**

Availability signifies that the important and essential information is provided to the parties involved. If the hacker isn't able to attack the two aspects of cybersecurity ,they may be able to launch other web vulnerabilities like DDOS attack and severe the link between the servers and the clients and destroy accessibility .

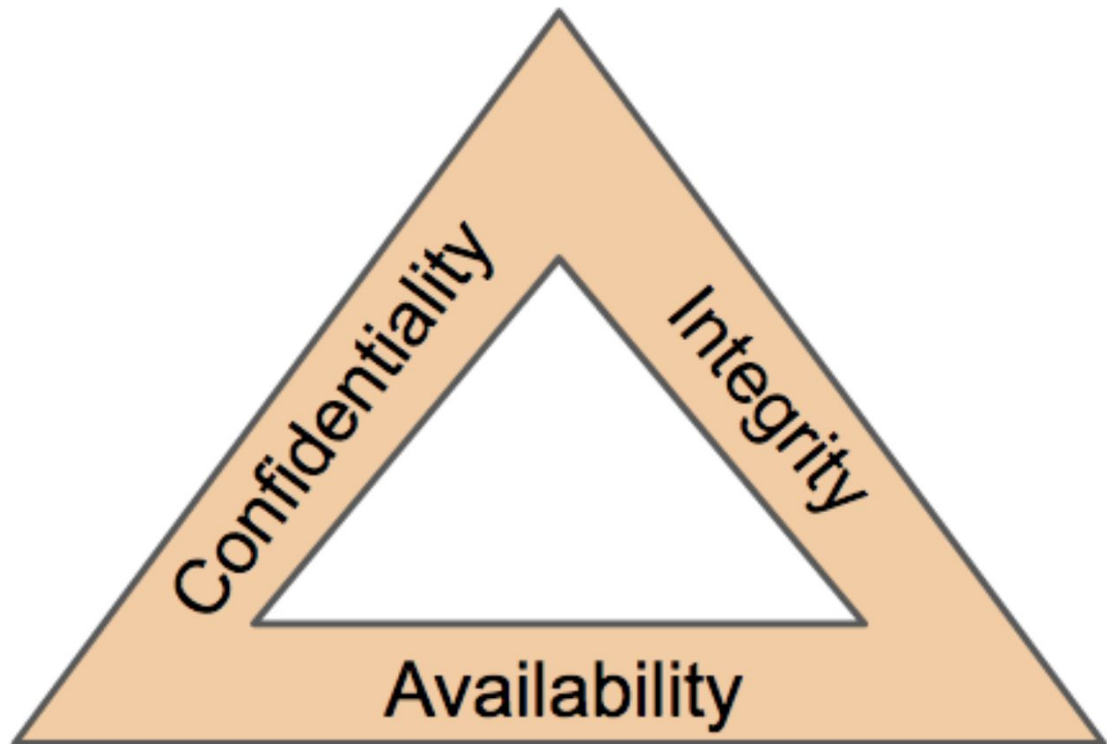


Fig 1.1

### **1.1.2 Vulnerabilities**

A thing flaw shortcoming is accomplished by a unknown mess up in the structure or coding of programming. A model is a data underwriting mess up, for example, the client

gave input not being reasonably assessed for vindictive information and a lot of long qualities related to known ambushes. Another model is a race condition blunder that enables the assailant to play out a particular development with raised favorable circumstances.

A security strategy setting is a piece of a thing's security that can be adjusted through the thing itself. Instances of settings are a working structure offering access to control records that set the preferences that clients have for reports, and an application offering a setting to empower or cripple the encryption .

A security strategy issue weakness joins function of security game plan settings that unfairly sway the protection of the information.

A thing highlight is a helpful breaking point given by programming. A thing highlight abuse weakness is a defenselessness wherein the segment also gives a road to bargain the security of a structure. These vulnerabilities are accomplished by the thing originator making trust questions that license the thing to give significant highlights, while in addition presenting the probability of somebody excusing the trust theories to bargain security.

Programming highlight abuse vulnerabilities are presented during the plan of the product or a segment of the product (e.g., a convention that the product actualizes). Trust suppositions may have been expressed - for instance, an architect monitoring a security shortcoming and discovering that a different security control would make up for it.

In any case, trust suspicions are regularly certain, for example, making an element without first assessing the dangers it would present. Dangers may likewise change over the lifetime of programming or a convention utilized in programming.

For instance, the Address Resolution Protocol (ARP) believes that an ARP answer contains the right mapping between Media Access Control (MAC) and Internet Protocol (IP) addresses. The ARP reserve utilizes that data to give valuable assistance—to empower sending information between gadgets inside a nearby system.

The ARP convention was institutionalized more than 25 years back, and dangers have changed a lot from that point forward, so the trust suspicions characteristic in its structure at that point are probably not going to at present be sensible today.

It might be difficult to separate programming highlight abuse vulnerabilities from the other two classifications. For instance, both programming blemishes and abuse vulnerabilities might be brought about by lacks in programming configuration forms. Be that as it may, programming blemishes are negative—they give no positive advantage to security or usefulness—while programming highlights abuse vulnerabilities happen because of giving extra highlights.

There may likewise be perplexity concerning abuse vulnerabilities for highlights that can be empowered or crippled—as it were, arranged—versus security design issues. The key contrast is that for abuse powerlessness, the arrangement setting empowers or impairs the whole component and doesn't explicitly adjust only its security; for a security design issue defenselessness, the setup setting modifies just the product's security.

For instance, a setting that cripples all utilization of HTML in messages significantly affects both security and usefulness, so powerlessness identified with this setting would be an abuse defenselessness.

### 1.1.3 Threats

A danger source is a reason for risk, for example, an antagonistic digital or physical assault, a human blunder of oversight or commission, a disappointment of association controlled equipment or programming, or other disappointment outside the ability to control of the association. A risk occasion is an occasion or circumstance started or brought about by a danger source that has the potential for causing an unfriendly effect.

System traffic commonly goes through middle PCs, for example, switches, or is persisted unbound systems, for example, remote hotspots. Along these lines, it tends to be captured by an outsider. Dangers against organizing traffic incorporate the accompanying:

Eavesdropping: Data stays unblemished, yet its security is undermined. For instance, somebody could get familiar with your charge card number, record a delicate discussion, or block arranged data.

Tampering: Data in travel is changed or supplanted and afterward sent on to the beneficiary. For instance, somebody could modify a request for merchandise or change an individual's resume.

Impersonation: Data goes to an individual who acts like the expected beneficiary. Pantomime can take two structures:

- Unauthorization: An individual can claim to be another person. For instance, an individual can claim to have the email address `jdoe@example.net`, or a PC can distinguish itself as a site called `www.example.net` when it isn't. This sort of pantomime is known as ridiculing.

- Deception: An individual or association can distort itself. For instance, assume the site `www.example.net` professes to be a furniture store when it is extremely only a site that assumes praise card installments however never sends any merchandise.

### **1.1.4 Security Controls**

Sensitive information should be given the utmost priority in a well structured and planned piece of design plans. The security controls must be given a thought from a out of the box kind of approach and should be thought with two classes. The framework should be able to overcome its own possible exploits that a user can use to achieve a successful attack on the system. given the most outrageous need in a particularly sorted out and orchestrated a couple of structure plans. The security controls must be given thought from an out of the case kind of approach and should be thought with two classes. The framework should have the alternative to vanquish its own latent capacity experiences that a customer can use to achieve a powerful attack on the system.

1. The board controls: The main overhead in a prescribed framework
2. Operational controls: The predefined actions to be executed if a security bug is triggered.
3. Specialized controls: A special set of instructions to be executed when a certain set of conditions are met.

. All of the mentioned above triggers are quite useful for a strong and cemented base for a cyber secure system. A strong security tactic is all the blocks involved should be strong on

their own individually and by using the specialized tools. A union of the security guidelines should provide the best set of instructions for the machine to follow.

The aggregate of the referenced above triggers are truly significant for a strong and built up base for a computerized secure system. A strong security methodology is all the squares included should be strong in solitude freely and by using the specific mechanical assemblies. A relationship of the security rules should give the best course of action of rules for the machine to follow. The triggers are truly basic for a strong and created base for a motorized secure system. A solid security thinking is all the squares included ought to be solid in division uninhibitedly and by utilizing the particular mechanical parties. A relationship of the security rules should give the best strategy of rules for the machine to follow, including a system based firewall, a host-based firewall, and OS fixing.

## **1.2 Objective**

In the era of information technology, web technologies are spreading at a greater pace. Lots of businesses are going online and more online services come into existence. This will increase the number of vulnerable sites on the internet. The objective of this project is to help developers and web security researchers to discover a web security vulnerability in web applications. The designed tool will identify the basic vulnerability such as Cross-site scripting, hidden directories on the server, Click-jacking, Unencrypted communication, Strict Transport Security enforcement, Denial of Service attack and other vulnerabilities.

It will be helpful in automation of the black box testing of the website and reduce man hours and the ease and maximize the process.



Fig 1.2: Symbols used for a secure application



## **Chapter-2**

### **LITERATURE SURVEY AND METHODOLOGY**

#### **2.1 Literature Survey**

##### **1. OWASP report**

The OWASP is a collection of the most important and lethal online web security vulnerabilities found by web researchers and compiled by them. It casually explains the danger of these exploits to the user. The report is compiled by the people best in the field and who are devoted to online web security research.

The following are the dangers announced in the OWASP report:

##### **A. Injection**

Injection exploits are usually carried out when malicious code is successfully executed on a system like a SQL database and can recover all the sensitive information of users. On the off chance that that structure input isn't appropriately verified, this would bring about that code being executed.

Injection exploits could be avoided by authenticating as well as disinfecting client submitted information. (Approval implies dismissing suspicious-looking information, while sterilization alludes to tidying up potentially dangerous pieces of the information.) moreover, an administrator should limit measure for data infusion assault could uncover.

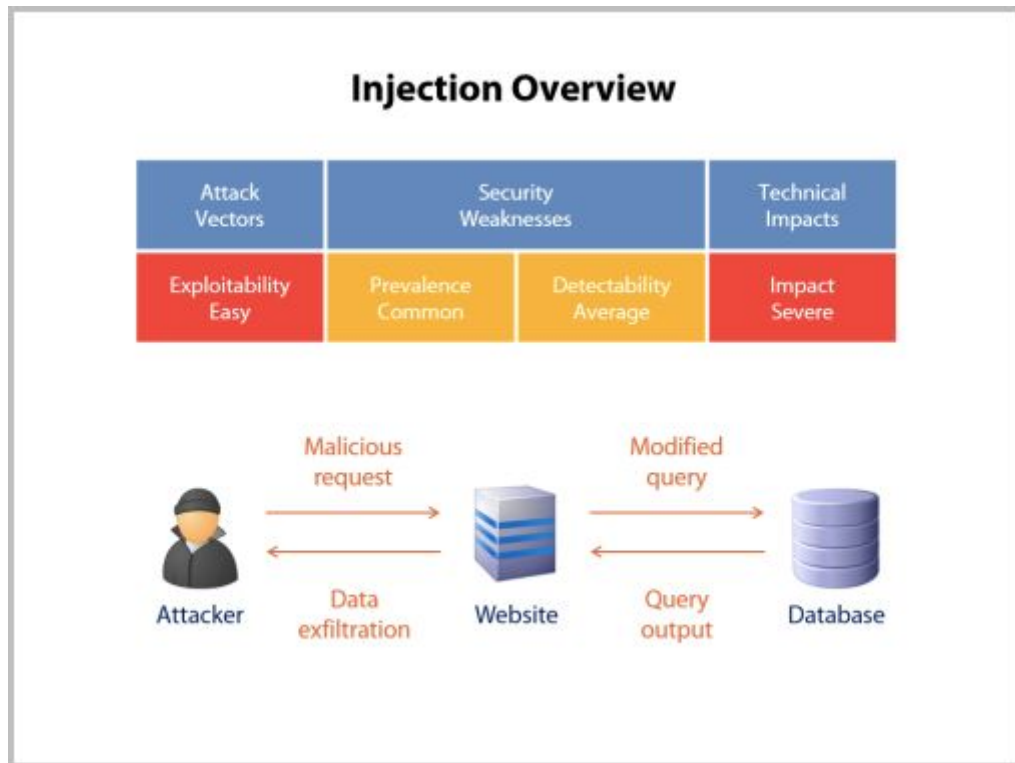


Fig 2.1: Injection Overview

## B. Broken Authentication

Validation and the capacity to bargain a whole framework utilizing an administrator and being able to shield the system from aggressors trying to gain control and gain a lot of information through an information rupture is the goal of a hacker.

An authentication system is usually used to provide more security like a code sent to an EMAIL or a code sent through an SMS .

Endorsement and the capacity to bargain a whole structure utilizing a chief and having the alternative to shield the system from aggressors endeavoring to get control and increment a lot of information through an information break is the target of a software engineer.

## C. Sensitive Data Exposure

If the website doesn't protect sensitive information of the user for example, monetary data and passwords, assailants will leak that sensitive information or could blackmail the users for money and cause damage to the company's image as well.

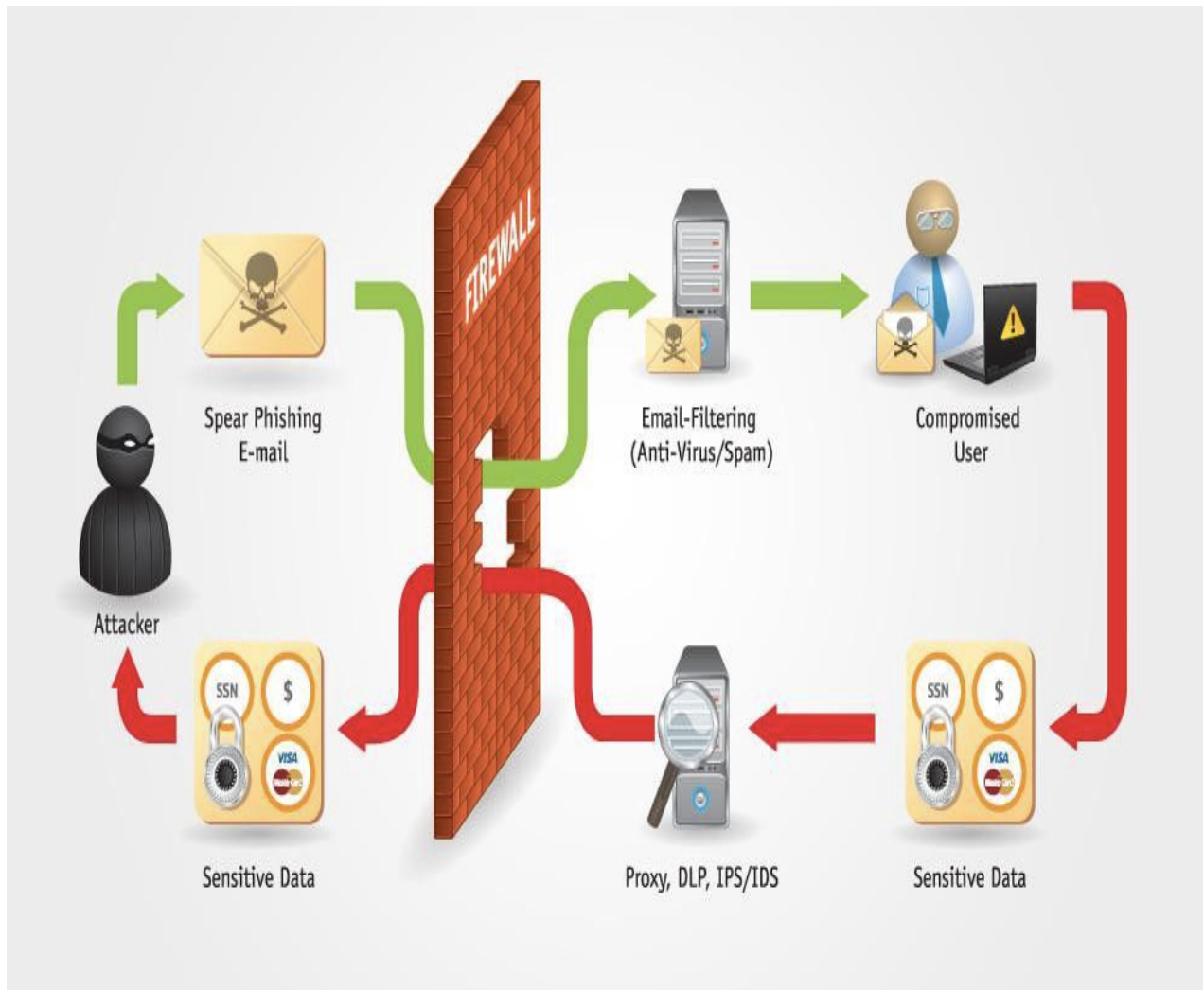


Fig 2.2: Sensitive Information Exposure

#### D. XML External Entities (XEE)

A web assault against a online code that uses XML input. This information would exploit an outside element, endeavoring to misuse helplessness. An 'outside substance' setting is

set for a fixed issue, for example, a HDD. An XML code can be hoodwinked into exploiting information for an unapproved outside substance, it could leak sensitive and private information of the client unknowingly to an aggressor.

(JSON) is a basic, intelligible documentation frequently used to transmit information over the web. In spite of the fact that it was initially made and can deciphered by a wide range of programming dialects.

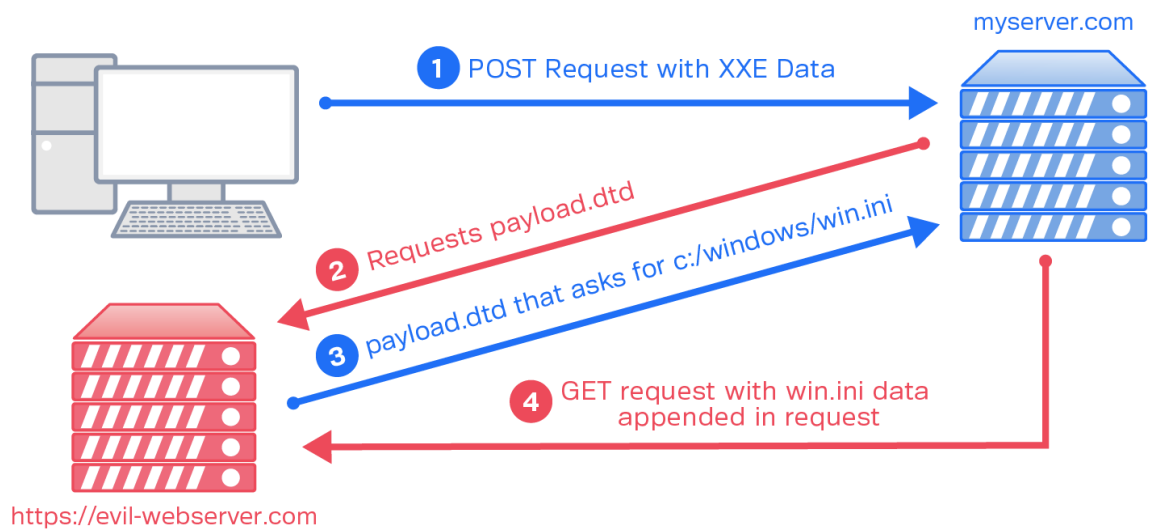


Fig 2.3: XXE Attack methodology

## E. Improper Access

For example, a web client could empower a customer to change which account they are marked in as just by changing a bit of a URL, with no other affirmation.

Access controls can be verified through guaranteeing that a website utilizes approval systems and is keen on them.

\*Many administrations control approval key when clients sign in. Each advantaged solicitation ;client causes will necessitate that the approval token is available. This is a safe method to guarantee .

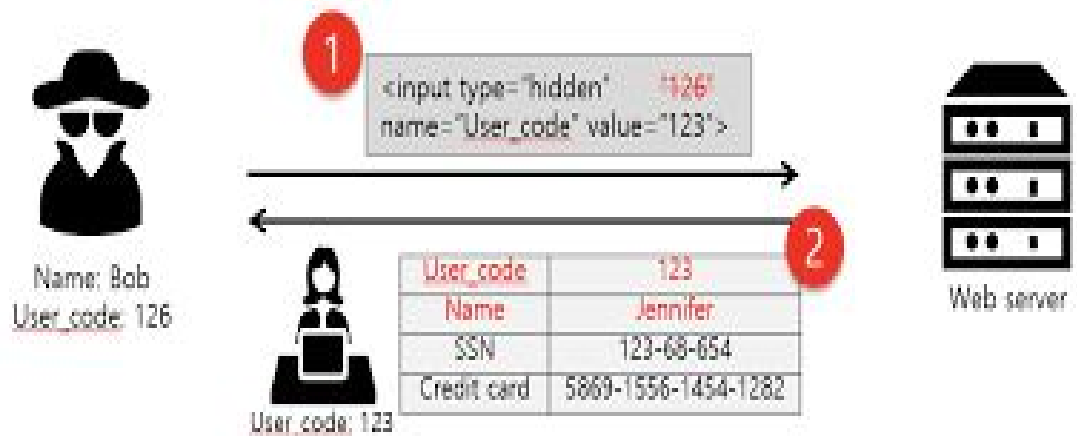


Fig 2.4: Broken Access Control

## F. Security Misdiagnosis

Security misdiagnosis can be widely recognized defenselessness rundown and should be frequently the consequence for utilizing arrangements or showing too many blunders. For example, an application could show a client can be generally perceived lack of protection once-over and ought to be much of the time the ramification for using courses of action or indicating an excessive number of botches.

## G. XSSs

Cross-site scripting generally occurs when a hacker can run malicious code on the client side of the website and steal the cookies of a user or can also execute scripts which can trigger further events vulnerabilities code on an unfortunate casualty's program. an application could show a customer can be commonly seen absence of assurance once-finished and should be a significant part of the time the consequence for utilizing game-plans or demonstrating an unreasonable number of bungles.

Relief techniques for cross-site scripting incorporate getting away untrusted HTTP demands just as approving or potentially purifying client produced content.

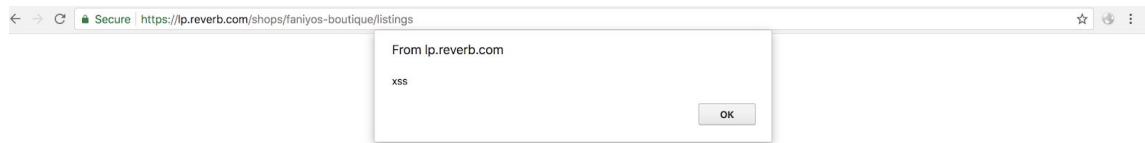


Fig 2.5: XSS POC

## H. Insecure Deserialization

This danger focuses on the vulnerabilities and issues often leak and endanger information., for example, putting away the information to the plate or spilling it. Deserialization is the polar opposite: changing over serialized information once more into objects the application can utilize. Serialization is similar to storing furniture a process where data is recycled and used resembles unloading the cases and gathering the furniture after the move. A shaky deserialization assault resembles having the movers mess with the substance of the containers before they are unloaded reused and used takes subsequent to exhausting the cases and gathering the furniture after the move. A temperamental deserialization snare takes in the wake of having the movers play with the substance of the compartments before they are exhausted.

## I. Using Components With Known Vulnerabilities

Numerous cutting edge security engineers do parts, for example, structures in their applications. All parts units of programming to assist engineers with maintaining a strategic distance regular model incorporates front-end structures like React and littler libraries that used to include share symbols or a/b testing.

Part designers regularly provide fixes and updates, however, cybersecurity engineers generally fixed or latest adaptations with segments executing as planned. For testing the danger of running segments with known issues, designers expel underutilized parts from undertakings, just guaranteeing .

## J. Inadequate Surveilling

A numerous number of websites are not finding a way to recognize information ruptures. For normal revelation time for a break is for at least 6 months. An assailant takes a many great deal of time to cause harm .Experts prescribes that backend engineers should execute 1 observing just occurrence reaction intends to guarantee that they are made mindful of assaults on their applications.

## 11. SANS TOP 25

The SANS Institute is a helpful research and instruction association. IT is a rundown of the most across the board and basic blunders that can prompt genuine exploits in programming (it would be ideal if you note: not all helplessness types apply to all programming dialects). The exploits incorporate uncertain cooperation between parts, dangerous asset the executives, and permeable guards.

Tears can identify 24 out of the be distinguished by static examination programming, causes you rapidly find them in your application, and gives point by point data on the best way to fix the blunders.

Rank	CWE	Name
1	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
2	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
3	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

4	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
5	306	Missing Authentication for Critical Function
6	862	Missing Authorization
7	798	Use of Hard-coded Credentials
8	311	Missing Encryption of Sensitive Data
9	434	Unrestricted Upload of File with Dangerous Type
10	807	Reliance on Untrusted Inputs in a Security Decision
11	250	Execution with Unnecessary Privileges
12	352	Cross-Site Request Forgery (CSRF)
13	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
14	494	The download of Code Without Integrity Check
15	863	Incorrect Authorization
16	829	Inclusion of Functionality from Untrusted Control Sphere
17	732	Incorrect Permission Assignment for Critical Resource
18	676	Use of Potentially Dangerous Function



20	131	Incorrect Calculation of Buffer Size
22	601	URL Redirection to Untrusted Site ('Open Redirect')
23	134	Uncontrolled Format String
24	190	Integer Overflow or Wraparound
25	759	Use of a One-Way Hash without a Salt

Table 2.1: List of SANS top 25

## 2.2 Methodology

Here are six of the types of website vulnerabilities which can be detected using this automated tool:

1. Hidden Directories Exploit - Recon
2. Click-jacking
3. Unencrypted Communication
4. DOS - Denial of Service attack
5. Strict Transport security not enforced
6. Cross-Site Scripting
7. Missing Essential Security header

The project is divided into two different phases:

1. Hidden directory finder on server
2. Web Application static analysis

### **1. Hidden directory & file finder**

Hidden directory and file finder application help a security researcher/engineer to find hidden files and directories on the application server. Hidden directory or file on the application server sometimes discloses the user information, hidden credentials or API key and server information. Exposes such information may be beneficial for an attacker to carry out further attacks.

### **2. Web application static analysis**

Static analysis of the web application consists of the analysis of the response header, response time and source code.

This will help developers in finding clickjacking, Cross-site scripting, denial of service vulnerabilities. Also, this will alert developers if any sensitive credential, API key or secret is leaked in the source code.

## Chapter 3

### SYSTEM DESIGN

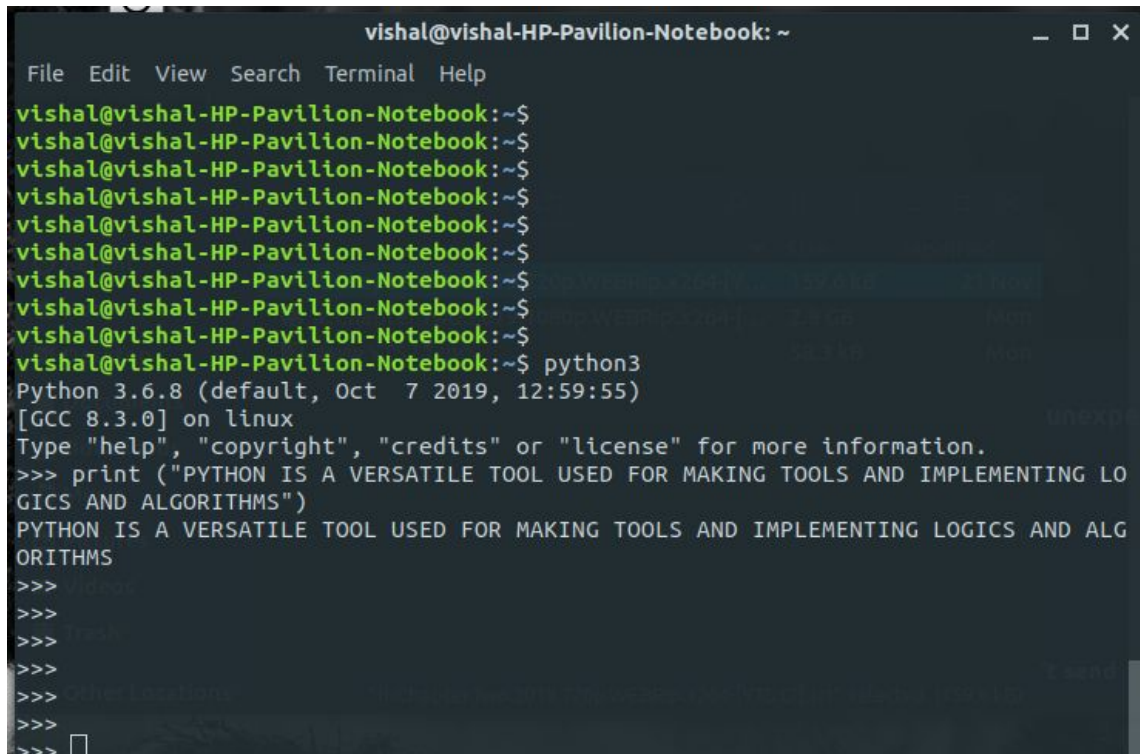
#### 3.1. PYTHON

Python is a very simple yet powerful programming language. It is very linear in its approach and yet retains its roots of object oriented programming. Python's lightweight and plethora of built in scientific-oriented inbuilt functions are quite useful. It's the perfect language for parsing and scripting.

. Python bolsters modules and bundles, which supports program measured quality and code reuse. The Python mediator and the broad standard library are accessible in source or paired structure without charge for every single significant stage, and can be uninhibitedly conveyed.

Frequently, software engineers begin to look all starry eyed at Python as a result of the expanded efficiency it gives. Since there is no assemblage step, the alter test-troubleshoot cycle is unimaginably quick. Troubleshooting Python programs is simple: a bug or terrible info will never cause a division deficiency. Rather, when the mediator finds a mistake, it raises a special case. At the point when the program doesn't get the special case, the translator prints a stack follow.

A source-level debugger permits examination of the neighborhood and worldwide factors, assessment of self-assertive articulations, setting breakpoints, venturing through the code a line at once, etc. The debugger is written in Python itself, vouching for Python's thoughtful power. Then again, regularly the snappiest method to investigate a program is to add a couple of print explanations to the source: the quick alter test-troubleshoot cycle makes this straightforward methodology exceptionally successful.



```
vishal@vishal-HP-Pavilion-Notebook: ~
File Edit View Search Terminal Help
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$
vishal@vishal-HP-Pavilion-Notebook:~$ python3
Python 3.6.8 (default, Oct 7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("PYTHON IS A VERSATILE TOOL USED FOR MAKING TOOLS AND IMPLEMENTING LOGICS AND ALGORITHMS")
PYTHON IS A VERSATILE TOOL USED FOR MAKING TOOLS AND IMPLEMENTING LOGICS AND ALGORITHMS
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

Fig 3.1: Python version

### 3.1.1 Python Library

A **Python library** is a reusable lump of code that you might need to remember for your projects/ventures. Contrasted with dialects like C++ or C, Python libraries don't relate to a particular setting in Python. Here, a 'library' freely depicts an assortment of center modules.

Every library in Python contains an enormous number of valuable modules that you can import for your consistent programming.

### Beautiful Soup



Scrapy is an open-source python system constructed explicitly for web scratching by Scrapinghub fellow benefactors Pablo Hoffman and Shane Evans. You may be asking yourself, "I'm not catching that's meaning?"

It implies that Scrapy is a completely fledged web scratching arrangement that takes a great deal of the work out of the building and designing your bugs, and the best part is that it consistently manages edge cases that you most likely haven't thought of yet.

Close to introducing the system, you can have a completely working bug scratching the web. Out of the case, Scrapy arachnids are intended to download HTML, parse and process the information and spare it in either CSV, JSON or XML record groups.

There is likewise a wide scope of inherent augmentations and middlewares intended for dealing with threats and sessions just as HTTP highlights like pressure, confirmation, reserving, client specialists, robots.txt and creep profundity confinement. Scrapy additionally makes it exceptionally simple to reach out through the advancement of custom middlewares or pipelines to your web scratching ventures which can give you the particular usefulness you require.

Perhaps the greatest bit of leeway of utilizing the Scrapy system is that it is based on Twisted, a nonconcurrent organizing library. This means Scrapy bugs don't need to hold on to make demands each in turn. Rather, they can make numerous HTTP demands in parallel and parse the information as it is being returned by the server. This fundamentally builds the speed and proficiency of a web scratching arachnid.

One little downside about Scrapy is that it doesn't deal with JavaScript straight out of the crate like Selenium. In any case, the group at Scrapinghub has made a Splash, a simple to-incorporate, lightweight, scriptable headless program explicitly intended for web scratching.

The expectation to absorb information to Scrapy is somewhat more extreme than, for instance, figuring out how to utilize BeautifulSoup. In any case, the Scrapy venture has

amazing documentation and a very dynamic biological system of designers on GitHub and StackOverflow who are continually discharging new modules and helping you investigate any issues you are having.

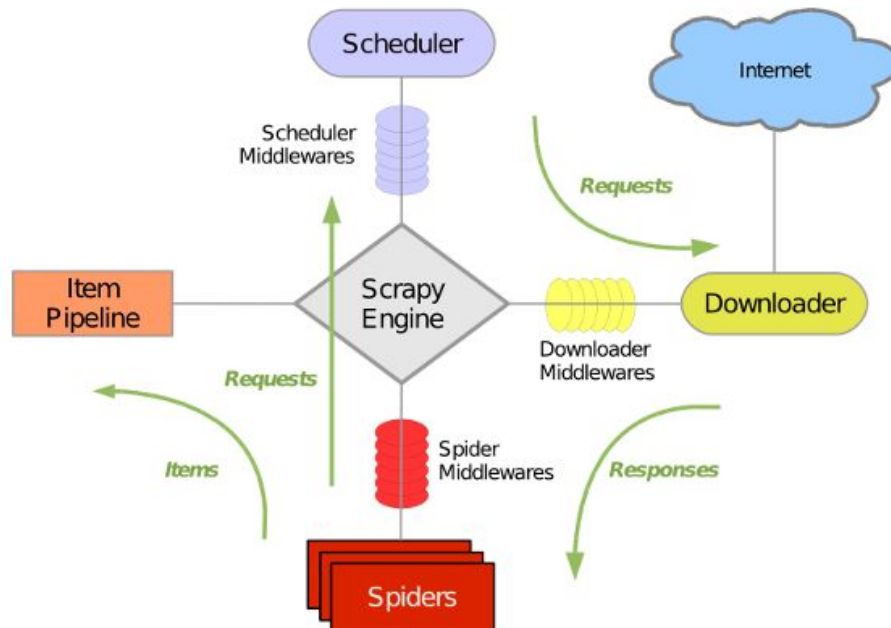


Fig 3.4: Explanation of Scrapy framework

### 3.2. SHELL SCRIPTING

Shell scripting is composing a progression of order for the shell to execute. It can join extensive and dreary successions of directions into a solitary and straightforward content, which can be put away and executed whenever. This lessens the exertion required by the end client.

Let us understand the steps in creating a Shell Script

1. **Create a file using** a vi editor(or any other editor). Name script file with **extension .sh**
2. **Start** the script with **#!/bin/sh**
3. Write some code.
4. Save the script file as filename.sh

5. For **executing** the script type **bash filename.sh**

A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

When you run the terminal, the Shell issues a command prompt (usually \$), where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal.

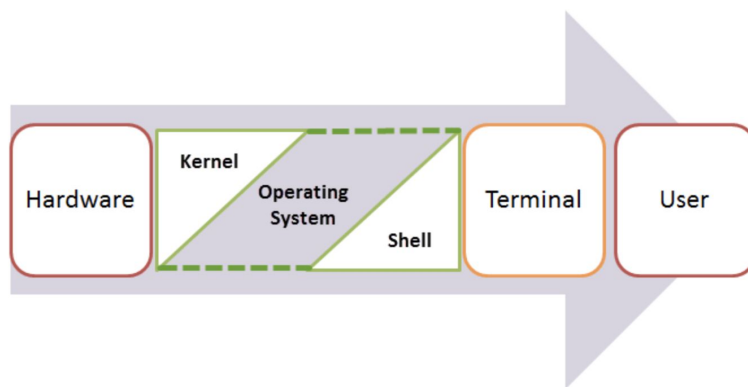


Fig 3.5: Existence of shell in O



## Chapter-4

### ALGORITHM

#### 4.1 Algorithm

```
//python Pseudocode
#import required python libraries

Import beautifulsoup
Import requests
Import urllib.requests

# Check for hidden directories
# file will contains most commonly directories used by developer and web servers
file=open("file","r")
Content = f.readline
content_array=[]
for x in content:
    content_array.append(x)
print(content_array)
domain=input("enter url")
for i in range(0,len(content_array)):
    url=domain+"/"+content_array[i]
    response=requests.head(url, allow_redirects=false)
    if response.status_code < 400:
        if response.status_code >=300:
#check redirection url using location header value
        print(response.status_code,url, response.headers(location))
    elif response .status_code >=200:
        print(response.status_code,url)
response=requests.head(domain,allow_redirects=false)
```

```
#check for clickjacking, strict transport security enforcement, unencrypted
communication, XSS
if response.headers('X-Content-Type') != "deny","same-origin":
    print("Vulnerable to clickjacking")
# Similarly check for other headers and analysis of the response

#check for DOS attack

response=requests.head(url, allow_redirects=false)
init_responsecode=response.status_code
For i in range (0,1000):
response=requests.head(url, allow_redirects=false)
final_responsecode=response_code
    if init_responsecode != final_responsecode:
        print("vulnerable to xss")
```

## Chapter-5

### TEST PLAN

#### 5.1 Data Set

For enumerating the directories on the web server, we use a list of all possible directories used by developers around the world and web servers. The scanner will scan the web server according to the provided lists of directories and files and will search for possible exposed directories.

The list used in this project is taken from OWASP project dir-buster. OWASP top 10 is considered as top organisation in the security field. And all security researchers follow OWASP top 10 guidelines while searching for security issue in a web application.

LIST:

<https://github.com/dustyfresh/dictionaries/blob/master/DirBuster-Lists/directory-list-2.3-small.txt>

#### 5.2 Test Setup

1. Install python on your machine

A. Windows



Fig 5.1: How to install python

B. Ubuntu/Kali Linux

Use these commands in terminal:

```
$ sudo apt-get update
```

```
$ sudo apt-get install python3.6
```

C. Mac O

Step 1: Install homebrew <http://brew.sh/>

Step 2: In Terminal use this command: *brew install python3*

2. Install python libraries/dependencies:

A. Requests

To install Requests, simply run this simple command in your terminal of choice:

```
$ pipenv install requests
```

B. BeautifulSoup

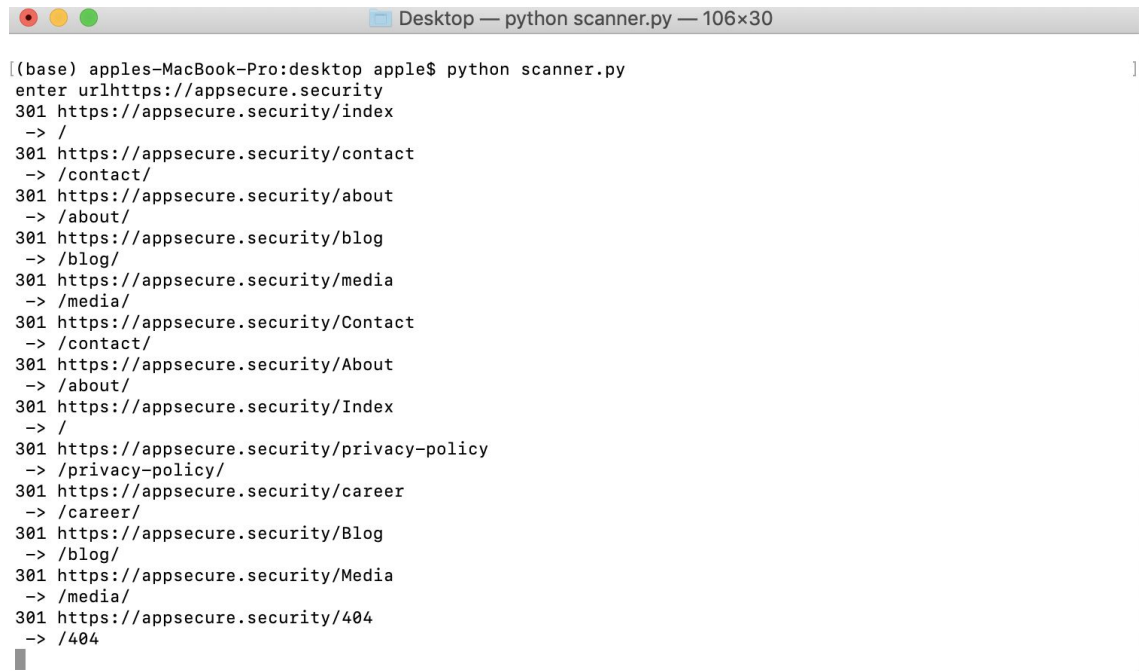
3. Run the developed python script in terminal and enter your site for scanning.

## **Chapter-6**

### **RESULT AND PERFORMANCE ANALYSIS**

#### **6.1 Result**

In seventh semester, half of the project is completed. The scanner is capable of nicely scanning the hidden directories and files on the web servers using website address.

A screenshot of a terminal window on a Mac. The window title is "Desktop — python scanner.py — 106x30". The terminal shows the following output:

```
[(base) apples-MacBook-Pro:desktop apple$ python scanner.py  
enter urlhttps://appsecure.security  
301 https://appsecure.security/index  
-> /  
301 https://appsecure.security/contact  
-> /contact/  
301 https://appsecure.security/about  
-> /about/  
301 https://appsecure.security/blog  
-> /blog/  
301 https://appsecure.security/media  
-> /media/  
301 https://appsecure.security/Contact  
-> /contact/  
301 https://appsecure.security/About  
-> /about/  
301 https://appsecure.security/Index  
-> /  
301 https://appsecure.security/privacy-policy  
-> /privacy-policy/  
301 https://appsecure.security/career  
-> /career/  
301 https://appsecure.security/Blog  
-> /blog/  
301 https://appsecure.security/Media  
-> /media/  
301 https://appsecure.security/404  
-> /404
```

Fig 6.1: Screenshot display the scanning progress on <https://appsecure.security>





## Performance Analysis

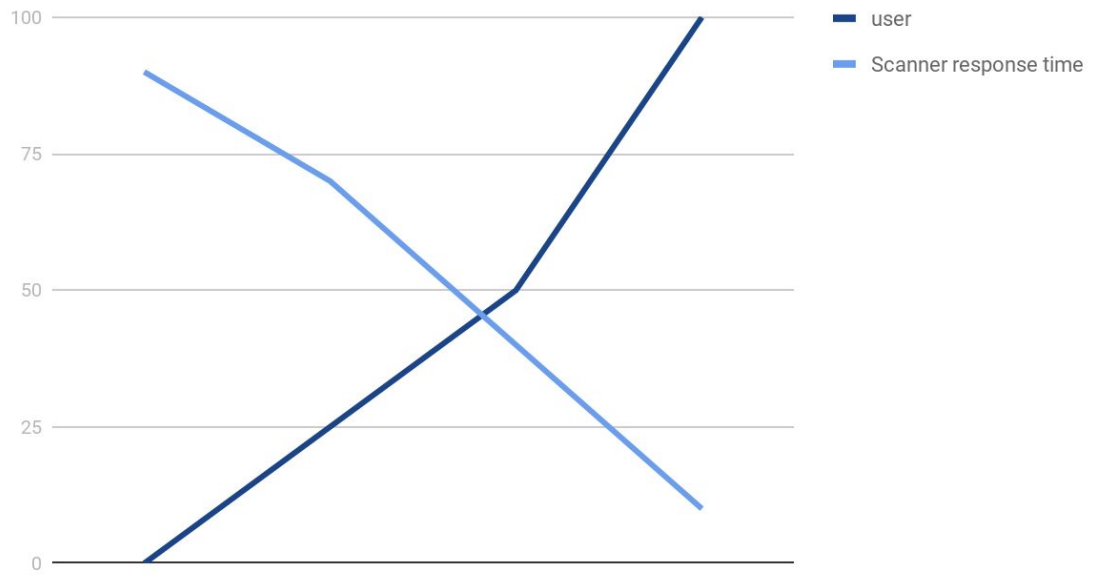


Fig 6.2: Graph depicting User vs Scanner response time



## **Chapter-7**

### **CONCLUSION**

#### **7.1 Conclusion**

Dangers to uprightness and privacy of data and assets are expanded. To remain ensured, developers and security engineers perform security testing of their web application to check the security stance of the framework. As we have experienced the writing overview of vulnerability detection techniques, it is discovered that there are different vulnerabilities leaving the web application prone to attack. Attackers finding better approaches to sidestep security systems so new vulnerabilities are advancing which should be tended to.

#### **7.2 Future Scope**

With more and more start of online businesses and services, the web applications are growing. This will create more attack surface for an attacker. With the help of this automated vulnerability scanner, may website can be protected from some basic security attacks. And will also create hindrance for the large attack, if this basic security vulnerability is fixed.

## REFERENCES

- [1] OWASP top 10  
[<https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/owasptop10/OWASP%20Top%2010%20-%202013.pdf>]
- [2] SANS top 25 [<https://www.sans.org/top25-software-errors/>]
- [3] Directory Buster  
[[https://www.owasp.org/index.php/Category:OWASP\\_DirBuster\\_Project](https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)]
- [4] Port Swigger web security [<https://portswigger.net/web-security>]
- [5] Microsoft Corporation. Microsoft .NET Framework Development Center.  
<https://msdn.microsoft.com/netframework/>, 2005.
- [6] Microsoft Corporation. System.Reflection Namespace.  
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfssystemreflection.asp>, 2005.
- [7] David Cruwys. C Sharp/VB - Automated WebSpider / WebRobot.  
<http://www.codeproject.com/csharp/DavWebSpider.asp>, March 2004.
- [8] <http://www.codeproject.com/csharp/DavWebSpider.asp>, March 2004.
- [9] David Endler. The Evolution of Cross-Site Scripting Attacks. Technical report, iDEFENSE Labs, 2002.
- [10] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. Fundamentals of Software Engineering. Prentice-Hall International, 1994.
- [11] Yao-Wen Huang, Fang Yu and Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, and Sy-Yen Kuo. Securing web application code by static analysis and runtime protection. In 13th ACM International World Wide Web Conference, 2004.
- [12] Yao-Wen Huang, Shih-Kun Huang, and Tsung-Po Lin. Web Application Security Assessment by Fault Injection and Behavior Monitoring. 12th ACM International World Wide Web Conference, May 2003.
- [13] Insecure.org. NMap Network Scanner. <http://www.insecure.org/nmap/>,
- [14] Shah. Sugandh, B.M. Mehtre, "A Modern Approach to CyberSecurity Analysis Using Vulnerability Assessment and Penetration Testing" in NCRTCST 2013, Hyderabad (A.P), India, Nov. 2013.

- [15] Shah Sugandh, B. M. Mehtre, "A Reliable Strategy for Proactive Self-Defence in Cyber Space using V APT Tools and Techniques", *Computational Intelligence and Computing Research (ICCIC)*, 2013.
- [16] S. Shah, B.M. Mehtre, "An automated approach to Vulnerability Assessment and Penetration Testing using Net-Nirikshak 1.0", *Advanced Communication Control and Computing Technologies (ICACCCT) 2014 International Conference on*, pp. 707-712, 8–10 May 2014.
- [17] Kumar Kranthi, K. Srinivasa Rao, "A Latest Approach to Cyber Security Analysis using Vulnerability Assessment and Penetration Testing", *International Journal of Emerging Research in Management & Technology*, vol. 3, no. 4, ISBN 2278–9359.
- [18] Urmi Chhajed, Ajay Kumar, "A Critical Review on Detecting Cross-Site Scripting Vulnerability", *International Journal of Innovative Research in Science Engineering and Technology*, vol. 3, no. 4, April 2014, ISBN 2319–8753.
- [19] A. Kieyzun, P.J. Guo, K. Jayaraman, M.D. Ernst, "Automatic creation of SQL Injection and cross-site scripting attacks", *Software Engineering 2009. ICSE 2009 IEEE 31st International Conference on*, vol. 199, no. 209, 16–24 May 2009.
- [20] Yadav Sushilkumar et al., "Survey: Secured Techniques for Vulnerability Assessment and Penetration Testing", *(IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 5, no. 4, pp. 5132-5135, 2014.
- [21] I. Yusof, A.-S.K. Pathan, "Preventing persistent Cross-Site Scripting (XSS) attack by applying pattern filtering approach", *Information and Communication Technology for The Muslim World (ICT4M) 2014 The 5th International Conference on*, vol. 1, no. 6, 17–18 Nov. 2014.
- [22] J. Bau, E. Bursztein, D. Gupta, J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing", *Security and Privacy (SP) 2010 IEEE Symposium on*, vol. 332–345, 16–19 May 2010.
- [23] M.E. Ruse, S. Basu, "Detecting Cross-Site Scripting Vulnerability Using Concolic Testing", *Information Technology: New Generations (ITNG) 2013 Tenth International Conference on*, vol. 633–638, 15–17 April 2013.
- [24] T.S. Rocha, E. Souto, "ETSSDetector: A Tool to Automatically Detect Cross-Site Scripting Vulnerabilities", *Network Computing and Applications (NCA) 2014 IEEE 13th International Symposium on*, vol. 306–309, 21–23 Aug. 2014.

- [25] Singh Tejinder, "Detecting and Prevention Cross-Site Scripting Techniques", *IOSR Journal of Engineering* 2.4 (2012), pp. 854-857.
- [26] "CWE -CWE List Version 2.9", Feb 2016, [online] Available: [Cwe.mitre.org](http://cwe.mitre.org).

## APPENDICES

### Code

```
import urllib.request
import requests
f = open("raw.txt","r")
content=f.readlines();
content_array=[]
for x in content:
    content_array.append(x)
#len(content_array)
domain=input("enter url")
for i in range(0,len(content_array)):
    url=domain+"/"+content_array[i]
    response=requests.head(url, allow_redirects=False)
    if response.status_code < 400:
        if response.status_code >=300:
            print(response.status_code,url , "->",
response.headers["Location"])
        elif response .status_code >=200:
            print(response.status_code,url)
print("Scanning done")
```

## Security Vulnerability Scanner 3

### ORIGINALITY REPORT

<b>17%</b>	<b>9%</b>	<b>5%</b>	<b>15%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>www.guru99.com</b> Internet Source	<b>3%</b>
<b>2</b>	<b>Submitted to University of Greenwich</b> Student Paper	<b>3%</b>
<b>3</b>	<b>developer.mozilla.org</b> Internet Source	<b>3%</b>
<b>4</b>	<b>Yuma Makino, Vitaly Klyuev. "Evaluation of web vulnerability scanners", 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015</b> Publication	<b>1%</b>
<b>5</b>	<b>Submitted to University of Bedfordshire</b> Student Paper	<b>1%</b>
<b>6</b>	<b>Submitted to University of Wales Institute, Cardiff</b> Student Paper	<b>1%</b>
<b>7</b>	<b>Submitted to Asia Pacific University College of Technology and Innovation (UCTI)</b>	<b>1%</b>

Pen. J. — 28/05/2020