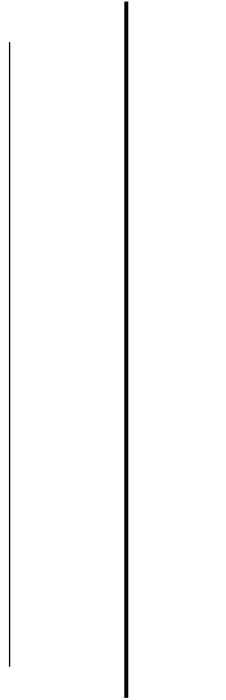


A Thesis for the Fulfilment of Bachelor's Degree
Web Vulnerability scanning using
Python



Submitted By:
SAGAR DHAKAL
STUDENT ID: 17-09-003

Submitted To:
Wakkanai Hokusei Gakuen University
Faculty of Integrated Media
Wakabadai 1-2290-28
Wakkanai, Hokkaido, 097-0013,japan

Supervisors:
Prof. Youjirou Harie

Co-supervisor:
Prof. Shinya Koizumi
Prof. Tomoharu Andoh

Abstract

Web applications security is playing crucial role in this world by securing businesses and their data. By using the web applications and sites, businesses can easily achieve its objective much faster. Web system are accessible anytime, anywhere via any pc with an internet connection. As technology changes, it becomes increasingly challenging for businesses of all types to keep their personal and customer's information on the web secure. If web applications/sites are not safe and secure, those critical business relationships cannot be compromised. A single security breach could be a death-knell for small business. Before publishing the sites and applications we need to check secureness of the web sites/apps. But it is very expensive to learn it with using professional tools, same as with community version tools it's unable to access the wanted information from the community version.

In this research project, I am making web vulnerable scanning scanner using Python programming language. In this project we use object oriented programming language. This project depends on Python and Python's libraries such as beautiful-soup, regular expression and Damn vulnerable web application which including all html resources and its back-end. In this project we scarp data, extract form, scanning link, send payloads to specific form and link to find vulnerability. cross site script is used to inject web application because the payloads is not visible for the browser's XSS filter. User accidentally trigger the payload if they visit the affected page. we can inject manually but it is not convenient due to lots of pages. Our scanner scan automatically in whole web app within limited time. Which helps to a fresh web security learner and motivate them to start web security.

KEYWORDS: Python, Beautiful-soup, data scrapping and web vulnerability.

Acknowledgement

My sincere thanks go to professor Youjiro Harie for guiding this work and his sincerity and encouragement. I will never forget his inspiration to make me one successful student in this bachelor degree. Which would not have been possible without the support of my supervisor Prof. Youjiro Harie and Co-supervisor Prof. Shinya Koizumi and Prof. Ando Tomoharu. I am very thankful for their noble guidance, support with full encouragement and enthusiasm. I would like to express my deepest gratitude to my advisor Prof. Dr. Bishnu Parsad Gautam for his continuous support directly and indirectly in my student carrier. Without their motivation and support it would be impossible for me to complete this path.

Similarly, I would like to thank Wakkanai Hokusei Gakuen University for gave me a golden opportunity to become a student of this college. And whole college team to make special familiar environment in my bachelor carrier.

Most importantly, I am grateful for my family whose constant love and support to keep me motivated and confident. And my friends, siblings who always guided and suggested me to select right path in my entire student career.

Finally, I am forever thankful for all this member to support and motivated in this entire thesis and each step of this path.

Table of Contents

ABSTRACT	2
ACKNOWLEDGEMENT	3
TABLE OF FIGURE	6
INTRODUCTION	7
1.1 VULNERABILITY SCANNER:.....	7
1.2 WEB VULNERABILITY SCANNER:.....	7
1.2.1 ADVANTAGE OF VULNERABILITY SCANNER:	8
1.2.2 DISADVANTAGE OF VULNERABILITY SCANNER:	8
RESEARCH CONCEPT	9
2.1 INTRODUCTION TO RESEARCH CONCEPT:	9
2.1 OBJECT ORIENTED PROGRAMMING LANGUAGE:	9
2.2 PYTHON:	10
2.3 WEB SCRAPPING:	11
2.4 BEAUTIFUL-SOUP:.....	11
2.6 REGULAR EXPRESSION:	12
2.7 REQUESTS:.....	12
2.8 URL PARSE:.....	12
2.9 CROSS SITE SCRIPT(XSS):	13
PROBLEM IDENTIFICATION AND SOLUTION.....	15
3.1 OBJECTIVE OF RESEARCH:.....	15
3.2 PROBLEM IDENTIFICATION:.....	15
3.3 LITERATURE SURVEY:	15
3.4 SOLUTION:.....	16
METHODOLOGY	17
4.1 METHOD:	17
SYSTEM OVERVIEW	20
5.1 HARDWARE AND SOFTWARE:	20
PROCEDURE	21

6.1	INFORMATION GATHERING:.....	21
6.2	SYSTEM DESIGN:	21
6.3	CODING:.....	22
6.4	INTEGRATION AND TESTING:	24
6.5	DEPLOYMENT:.....	32
 <u>IMPLEMENTATION.....</u>		<u>34</u>
 <u>CONCLUSION AND FUTURE WORK.....</u>		<u>35</u>
7.1	CONCLUSION	35
7.2	FUTURE WORK.....	35
 <u>REFERENCE:.....</u>		<u>36</u>

Table of Figure

Figure 1:concept of scanning [Figure-1]	8
Figure 2:Object Oriented programming[Figure2].....	9
Figure 3: Python libraries[Figure3].....	10
Figure 4: web scrapping[Figure4].....	11
Figure 5: Regular expression[figure5]	12
Figure 6: cross site script	13
Figure 7: non-persistent working flow	14
Figure 8:Waterfall model.....	18
Figure 9: Rough sketch of system design.....	21
Figure 10: scrapping data from web app	24
Figure 11:extracting form code.....	24
Figure 12: HTML attributes.....	25
Figure 13: data posting code	26
Figure 14: data posted	27
Figure 15:crawl url	28
Figure 16: Payloads test.....	29
Figure 17:Output of payloads acceptance	30
Figure 18: payloads send through the link	30
Figure 19:sample of output	31
Figure 20:output of the scanner 1 st	32
Figure 21:output of the scanner 2 nd	33

Introduction

1.1 Vulnerability scanner:

Vulnerability scanning is the process of detecting and classifying potential points of exploitation in applications same as it is considered to be the most efficient way to check our site, application, network device and computer systems against a huge list of known vulnerabilities [1]. It is a technically computer programs that search weakness of the systems and breach a network and steal information of the organization, users and install malware.

There are types of cyber security holder known as a hacker and they defined as white hat hacker who works legally and black hat hacker who works illegally. But they both are use vulnerability scanners. Vulnerability scanners are automated tools that allow organizations to check if their networks, systems and applications have security weakness that could expose them to attack.

1.2 Web vulnerability Scanner:

A website vulnerability scanner is an automated software designed to search for security vulnerabilities in web site and application. It scans for web vulnerability scanner which are dynamic and language-independent[2].vulnerability scanner tool test web applications for common security problems such as cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF)[3]. These detect whether the applications have any well-known software vulnerabilities, insecure configurations or insecure code issues.

There are many tools and products in the vulnerability scanning space that cover different types of assets and offer additional features that help companies to implement a complete vulnerability management program the combined processes related to identifying, classifying and mitigating vulnerabilities. This types of tools are expensive, although, it is affordable for big organizations but the student and fresh learner can't afford such tools due to financial problem. But it is possible to do with written programming language. And it is free and affordable to all those person who have knowledge of Object oriented programming. It will be good for fresh learner.

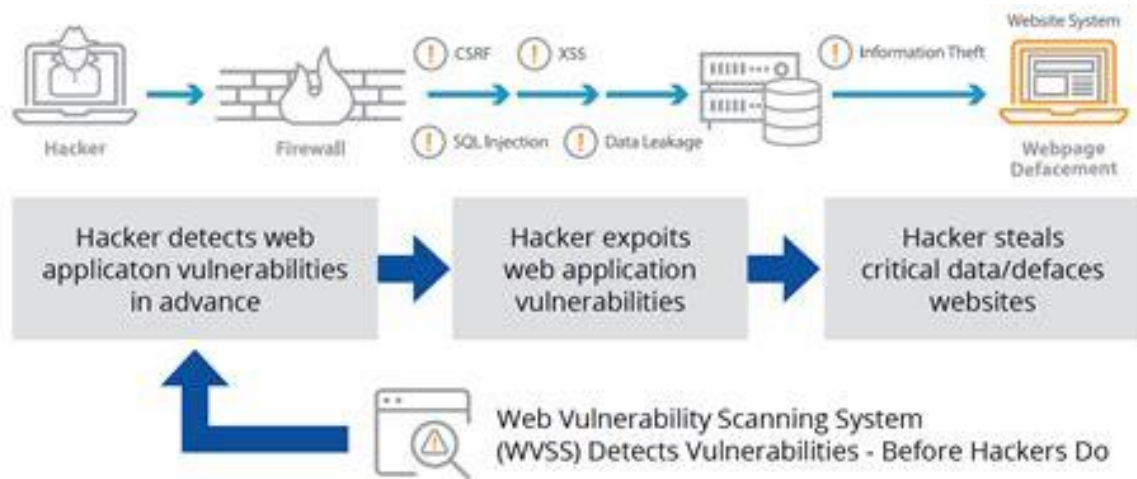


Figure 1:concept of scanning

In figure (1), hacker detects application's vulnerability then they exploit web application using different ideas to steal the critical data of web and user's information.

1.2.1 Advantage of vulnerability scanner:

- Identify known security exposures before attackers find them.
- Create an inventory of all the devices on the network, including purpose and system information. This also includes vulnerabilities associated with a specific device
- Create an inventory of all devices in the enterprise to help with the planning of upgrades and future assessments.
- Defines the level of risk that exists on the network.

1.2.2 Disadvantage of vulnerability scanner:

- Its little bit hard to understand
- Sometime the output will be wrong

Research Concept

2.1 Introduction to research concept:

Research concept is to make vulnerability scanner using Object Oriented Programming language. Web technology change almost every aspect of 21st century. We can do most of thing using web technology such as web banking, shopping, communicating etc. these online platform store customer's personal information which should be secured because All websites and application are prone to get attacked anytime anywhere. Cybercriminals use programs to automatically detect web containing vulnerabilities. These vulnerabilities are used as points of entry to execute an attack and send malicious payloads on that particular website/application. To prevent from cybercriminals we need to learn cyber security. As per my experience it is very hard to learn without professionals tools which is expensive and not affordable to for fresh learner and student. After few research I got Burp suite tools which use for scan vulnerability but it hard to understand due to the extra information as well they wants subscription for the full function similarly, we cannot scan without subscription then I started looking some tutorials of burp suite then started to write script to take same output.

2.1 Object Oriented Programming language:

Object Oriented Programming (OOP) is the programming paradigm that relies on the concept of classes and objects. It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects. There are many object-oriented programming language including JavaScript, C++, Java, and Python[4].

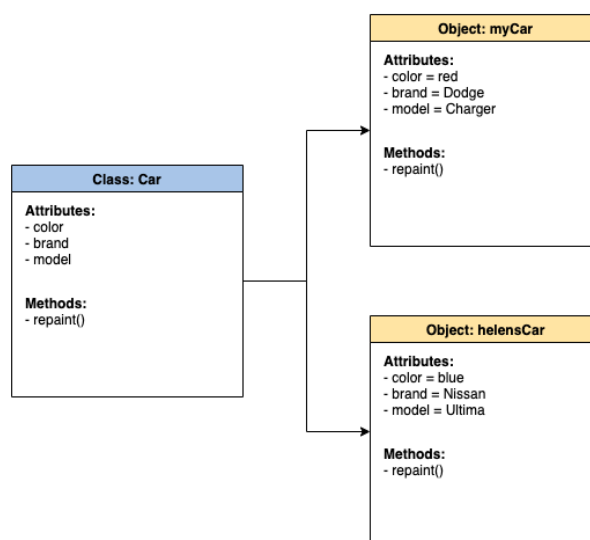


Figure 2: Object Oriented programming

2.2 Python:

In this research, I am using Python programming language and its library. It is a high-level, interpreted, interactive and object-oriented programming language. Python is designed to be highly readable. It has fewer syntactical construction like the English language than other programming languages[5]. It can be used in various different place such as Data Science, Machine learning, Web development, Mobile app development, Computer science education, computer vision. Python is very famous and strong high level programming due to its library. Python's standard library is vast, we can find all the necessary functions we require for any given task. This makes Python independent of external libraries. Although, if we wish to use some external libraries, then with the Python package manage (pip), we can easily import several packages from the massive Python Package Index (PyPi), containing more than 200,000 packages[6].

In this research, I am using Python's library and the library is most important part of my project. A Python library is a reusable chunk of code that is used in program/script using import command. A package is library if it is installable or gets attached to site-package of the folder of Python installation.

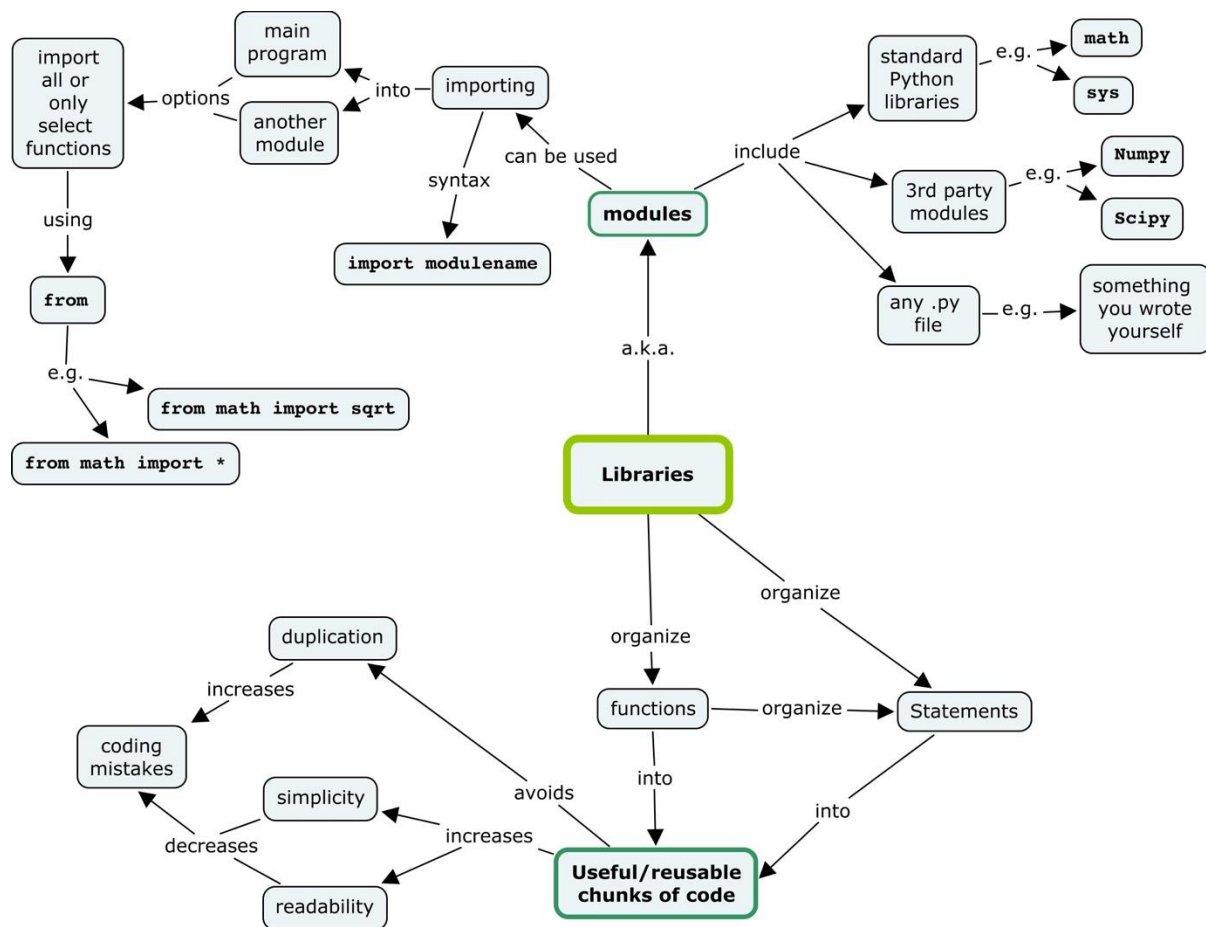


Figure 3: Python libraries

In my project some used library are as: Beautiful-soup, Regular-expression, Requests, URL parse etc which lies in the modules section of figure 3.

2.3 Web scrapping:

In simple terms, Web scraping, web harvesting, or web data extraction is an automated process of collecting large data(unstructured) from websites and other online factors like search engine, company information, news sources, social media etc as illustrated in figure 4. The user can extract all the data on particular sites or the specific data as per the requirement. The data collected can be stored in a structured format for further analysis[8].



Figure 4: web scrapping

2.4 Beautiful-soup:

Beautiful soup is a Python library for getting data out of HTML, XML, and other mark-up languages. Say you've found some webpages that display data relevant to our research, such as date or address information, but that do not provide any way of downloading the data directly . Beautiful Soup helps we pull particular content from a webpage, remove the HTML mark-up, and save the information. It is a tool for web scrapping that helps we clean up and parse the documents we have pulled down from the web[7].

2.6 Regular expression:

A regular expression is sequence of characters that specifies a search pattern. Usually such patterns are used by string-searching algorithms for “find” or “find and replace” operations on strings, or for input validation. It is technique developed in theoretical computer science and formal language theory[9].



Figure 5: Regular expression

2.7 Requests:

Requests is a Python module that we can use to send all kinds of HTTP requests. It is an easy-to-use library with a lot of features ranging from passing parameters in URLs to sending custom header and SSL verifications[10]. Requests allow we to send HTTP/1.1 requests. We can add headers, from data, multi-part files, and parameters with simple Python dictionaries, and access the response data in the same way.

2.8 URL parse:

This module provides a standard interface to break Uniform Resource Locator (URL) strings in components or to combine the components back into a URL string. It also has functions to convert a "relative URL" to an absolute URL given a "base URL.[11]"

2.9 Cross site Script(XSS):

Cross-site scripting (XSS) is a web application vulnerability that permits an attacker to inject code, (typically HTML or JavaScript), into the contents of an outside website. When a victim views an infected page on the website, the injected code executes in the victim's browser. Consequently, the attacker has bypassed the browser's same origin policy and is able to steal private information from a victim associated with the website[1].

Persistent Cross-site Scripting (Stored XSS) attacks represent one of three major types of Cross-site Scripting. The other two types of attacks of this kind are Non-Persistent XSS (Reflected XSS) and DOM-based XSS. In general, XSS attacks are based on the victim's trust in a legitimate but vulnerable web application or website.

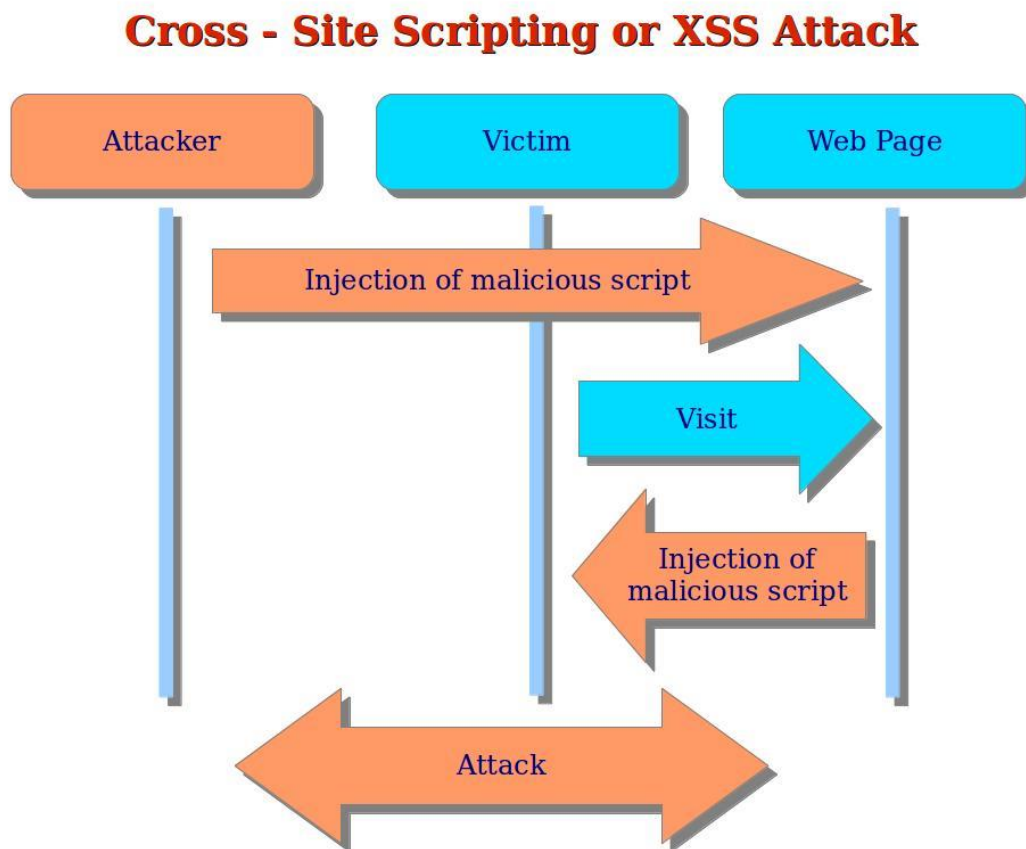


Figure 6: cross site script

In figure (6), attacker inject malicious script in the web page when user is visit in this page the script will execute and user will be hacked.

There are two types of CROSS SITE SCRIPT which are follows:

1. Persistent:

Persistent XSS attacks, also known as stored XSS attacks, occur when an attacker identifies a vulnerability in a web application that allows for script injection. The attacker is able to inject a malicious script into the web.

2. Non-persistent

In these attacks, an attacker passes a malicious script through a query, which is typically within url. Attackers can send this url to unsuspecting users through anonymous emails, forums, or anonymous social media feeds.

In our project, we can scan Non-persistent vulnerability of the website.

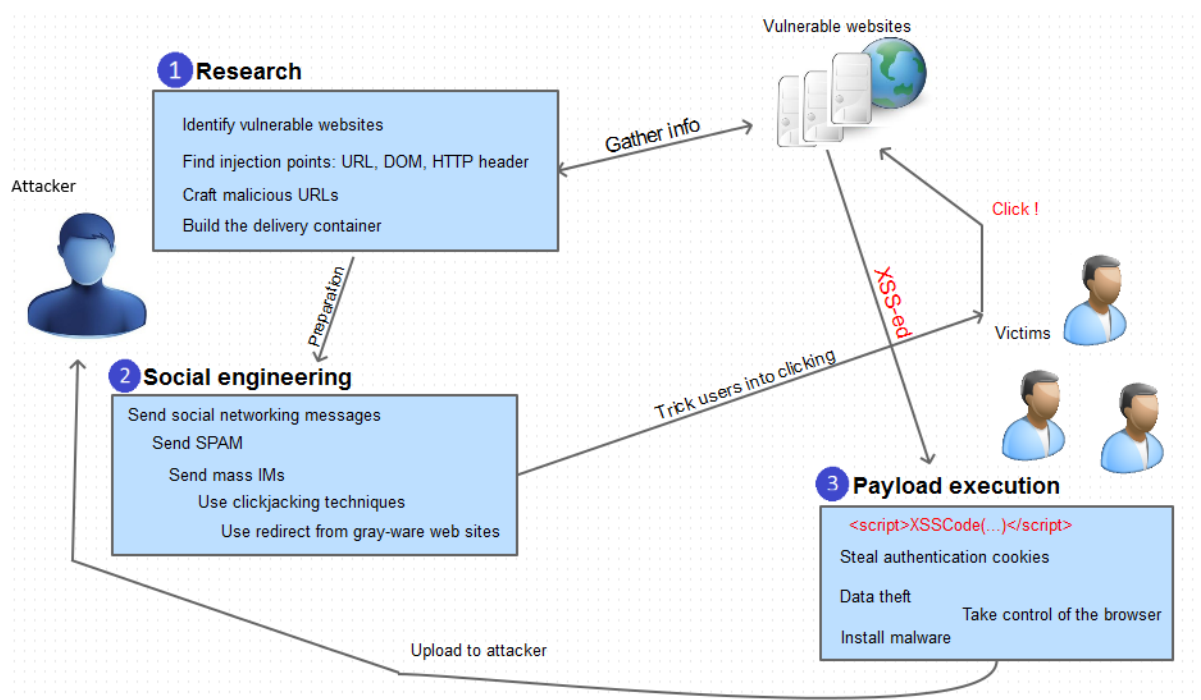


Figure 7: non-persistent working flow

Once a website is identified as being potentially vulnerable, the attackers try to inject script code into the relevant areas and verify if the script is returned in its original form (and executed). This process can either be manual or automatic, depending on the website /web application and the potential injection points found.

Problem identification and solution

3.1 Objective of research:

The main objective of this research is to influence those person who wants to learn security without facing financial and extra heavy information analysis problem in starting phase.

3.2 Problem Identification:

We know, if we are capable to buy there are lot of tools are available in the internet to use and learn web security. Without subscription we cannot access proper function of those tools. Same as it is not easy to understand due to heavy extra information and which demotivate to the person and they stop to learn. If they buy they try to earn money using those tools and they turn into wrong side. I have faced this problem and I started searching the way to learn security. After exploring web for certain period I got some idea, how I can encrypt same output and I can post data through the Python and its library same as that tools. It reduce cost because Python is open source programming language and it motivate to all those person who wants to learn web security. despite all the difficulties there are certain problems that can be solved with web security.

Larger number of population is influenced by the internet due to the continuous proliferation in the Internet facilities, usage, speed, user friendly browsing, global access, etc. At flip side, hackers are also attacking this digital world with new tactics and techniques through exploiting the web application vulnerabilities. The analysis of these vulnerabilities is of paramount importance in direction to secure social digital world. It can be carried out in two ways. First, manual analysis which is error prone due to the human nature of forgiveness, dynamic change in technology and fraudulence attack techniques. Second, through the existing web application vulnerability scanners that sometime may suffer from generating false alarm rate. Depending, there is a need to develop a framework that can detect different levels of vulnerabilities, ranging from client side vulnerabilities, communication side vulnerabilities to server side vulnerabilities

3.3 Literature survey:

The importance of using vulnerability scanner to unveil flaws in web applications before they deployed has been realized by many organizations today. Due to the ever-growing cybercrime, this study has examined some scanners that can be used to detect vulnerabilities which can be easily be missed by developer. If we use scanners tool it performs differently and give different results depending on the program and configuration of the scanner which make confuse to find specific place. It is good for the experienced person because they wants

check fast all over thing in once but it is very hard to understand for fresh learner. Lot of experienced person suggest that for better result, commercial scanners are preferable because of their regular updates as compared to open source scanners[11].

I was motivated by “[The Web Application Hacker’s Handbook](#)” book written by Dafydd Stuttard and his tool [Burp Suite](#) which is available in professional and community version. I didn’t use professional version but I used community version which motivated me to make manual scanner. We cannot scan active or passive link same as we cannot post data that we wanted place in its community version tool.

3.4 solution:

As the view of problem I had have faced extra confusing heavy information and financial problem. Same as, There are lot of person who are in trouble due to the same problem as mine. There are uncountable tools that we can use after subscription but as the view of student and fresh learner they cannot afford neither anyone help them to in starting phase. That’s why I started the see some tutorial of tools and started to learn about tools working method and its output. After, I felt that can be done with Python. Then I started because I have an intermediate knowledge of Python. To make manual scanner we need **data extraction**, which is possible with beautiful-soup and **send payloads** it is normal because its library and it make sense for this journey.

This scanner scan vulnerability in client side of the web application. When a user loads an affected page, the attacker’s scripts will be executed, with which they can steal session tokens and cookies, change the content of the web page through DOM manipulation or even redirect the browser. XSS vulnerabilities typically occur when an application takes user input and outputs it to an unvalidated page.

Here, we scan vulnerability before publishing the web application/site in the market. This scanner check every data insertion part of the web application through the url of the specific page. Same as this scanner scan vulnerability in every url of the web page including insertion. This scanner mainly scan using reflected cross site script to check the site is vulnerable or not. there is hard to check manually every page of the application due to the limited time. This scanner check all connected page of the application in once.

Methodology

4.1 method:

In this project, mentioned listed are implemented in this research.

- Problem identification
- Literature survey
- Solution
- System overview
- Implementation

After starting to learn cyber security there are lot of problem, where should we implement to know about the output same as how to do without particular concept and lot of unknown information. Professor Youjirou Harie helped me by giving some idea, books and source code to help to understand the internal working method of network part in web application . We started to find the specific way to learn cyber security in internet and we got a platform for test for the learner. After it, we tried lot of community version tools watching their tutorials in YouTube and browser but still we were unable to do anything due to community version's tool. Again and again same problem then we started to learn about the output of the professional tools and its working method. It is impossible to make same to same working mechanism due to the lack of knowledge but we were able to done specific working method. I already write the working method in "problem's solution" part. This Which help us to start our journey.

This project has been developed using the **water fall model**[\[12\]](#). It is also referred to as a linear-sequential life cycle model. In this model, each phase must be completed before the next phase can begin and there is no overlapping in this phases. Waterfall model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete.

Here are the phases of waterfall(linear-sequential life cycle) model:

1. **Requirement Gathering and analysis**
2. **System Design**
3. **Coding**
4. **Integration and Testing**
5. **Deployment**
6. **Maintenance**

Waterfall model

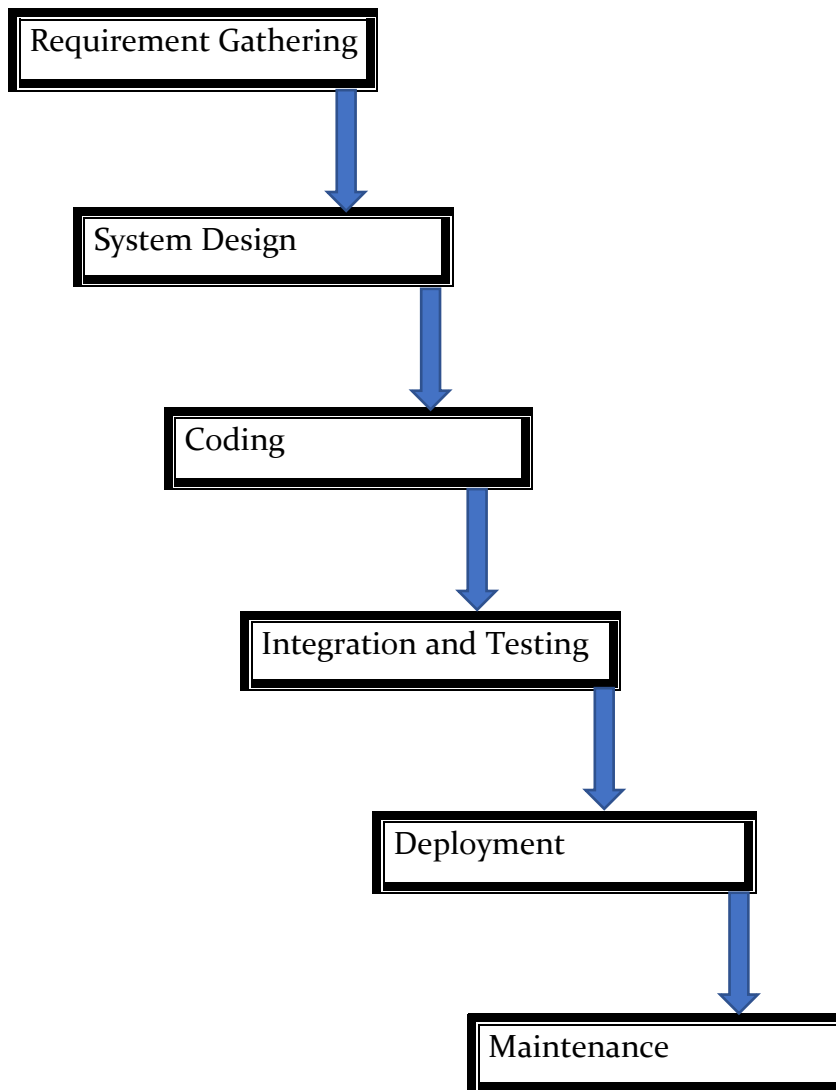


Figure 8: Waterfall model

Working process of waterfall model:

1. Requirement gathering and analysis:

The team gather the possible information which need for system development in first phase of this model. In our content we need to learn some frontend html's code for the design.

2. System design:

After gathering specific possible information or requirements it will be prepared for design. This phase helps us to define overall system architecture.

3. Coding:

Developer implement design according to the requirements, the system is first developed in small programs call units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as unit testing.

4. Integration and Testing:

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

5. Deployment:

Once the functional and non-functional testing is done, the project is deployed in the customer environment or released in real word.

6. Maintenance:

There are some issues which come up in the client environment. To fix those issues, and upgrade in future.

System Overview

In this research I have used Object Oriented programming language name as Python and its library and Damn vulnerability web application for the development of proposed system.

This scanner for finding vulnerabilities in web application. It is used for reflected Cross site script scanning. There, we are using all Python and its library to make this scanner. it is not graphic user interface scanner it just command line interface scanner. In this scanner, we need to change url of the web sites for the different website. It's for the fresh web security learner but they must have intermediate Python and web frontend programming knowledge to use this scanner.

It is illegal to try in random web application that's why I choose the damn vulnerable web application and Metasploit machine. This DVWA is used to check tools. Same as it is used to learn hack the web application.

Similarly, for this scanner I use object oriented programming and their library they are as follows:

- Python
- Beautiful-soup
- Requests
- URL parse

Listed definition are already written in introduction part of the thesis. Again I am using this list to demonstrate its use case in my research progress. As I already wrote, Python is open source and can be used it in various different place due to its library. This is the main reason to use this language in this research. Similarly, I have used Beautiful-soup to scrap specific part mark-up language from the web application. Requests, it is used to requests Resource Universal Locator(url) from the web browser of web application. URL Parse is for, parse string value and combining URL components into a URL string.

Internally, this scanner work in two different path, the first one through the http requests using [GET](#) method. The script will be send through the link such as: <http://website.com/news.php?id=1> etc. we can send payloads from the equal sign of this link to check vulnerability of the web application.

Similarly, the second one is [POST](#) method. In post method we use form boxes to find vulnerability .

5.1 Hardware and software:

Used hardware and software for the system development.

- a) Hardware
MacBook Pro (13-inch, mid 2012) processor 2.5 GHz intel core i5 memory 8GB.
- b) Software
Python programming language, Virtual box, Metasploit able machine etc.

Procedure

6.1 Information Gathering:

Begin with defining our goal for success a clear objective will determine what information is relevant and what we can ignore. In this step of the SDLC model we gather information to implement. For this system we need know about the working mechanism of the web applications or sites. After the knowing the mechanism of the sites we need to gather information to use to design for the system. The requirement are as follows:

- Object oriented programming
- Python programming
- Beautiful-soup and Python's library
- Cross site script(XSS)

6.2 System Design:

It is the design that showing the working process of the scanner where the website or user in same device and it requests to server by sending data payloads to server then server send response to user device .

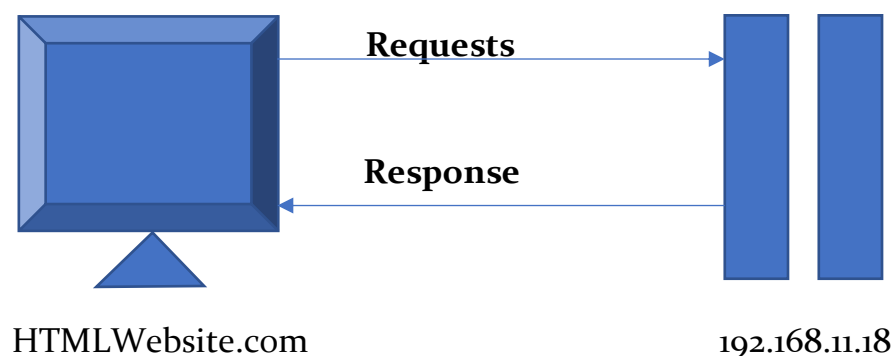


Figure 9: Rough sketch of system

In figure(9), there is end-user name as (htmlwebsite.com) and server. where end-user requests to server by sending data and server responding back to end-user.

Internal process of the design are as follows:

First of all we scrap mark-up language by using beautiful-soup then user sends data through the get or post method to check response of the server. Then we access the every connected link of the web applications. After, user will analyse the output and sends payloads to find vulnerability.

6.3 Coding:

Coding is the act of writing a script in a language that the computer or machine can understand.

Sample of the code:

Crwal3.py

```
import requests
import re
from urllib.parse import urlparse
from bs4 import BeautifulSoup

try:
    import urlparse
except ImportError:
    import urllib.parse as urlparse

class Scanner:
    def __init__(self, url, ignore_links):
        self.target_url = url
        self.target_links = []
        self.session = requests.Session()
        self.links_to_ignore = ignore_links

    def extract_links_from(self, url):
        response = self.session.get(url)
        return re.findall('(?:href=")(.*?)"', response.content.decode('utf-8', 'ignore'))

    def crawl(self, url=None):
        if url is None:
            url = self.target_url
        href_links = self.extract_links_from(url)
        count = 0
        for link in href_links:
            link = urlparse.urljoin(url, link)

            if "#" in link:
                link = link.split("#")[0]
```

```
        if self.target_url in link and link not in self.target_links and link
not in self.links_to_ignore:
            self.target_links.append(link)
            print(link)
            self.crawl(link)
```

Crawl4.py

```
import crawl3

target_url = "http://10.32.168.11/dvwa/"
links_to_ignore = ["http://10.32.168.11/dvwa/login.php"]
data_dict = {"username": "admin", "password": "password", "Login": "submit"}
vuln_scanner = crawl3.Scanner(target_url, links_to_ignore)
vuln_scanner.session.post("http://10.32.168.11/dvwa/login.php",
data=data_dict)
vuln_scanner.crawl()
```

It's little bit rough to write code mid of the thesis that's why I just post here unit of the project code. It is crawl part of the full code. We can extract and explore link of the web application using this code.

It is Python code and its library such as beautiful-soup, requests, urlparse all are including here.

6.4 Integration and Testing:

Implementation phase are integrated into a system after testing of each units. This is the phase where written script's unit is checked. Unit is piece of code. The system is developed by combining all those units.

We can see here all my system's unit and explanations. In system, first we scarp data from the web application using beautiful-soup. Here,

```
</tr>
<tr><td></td></tr>
<tr>
<td class="form-header" colspan="2">Who would you like to do a DNS lookup on?<br/><br/>Enter IP or hostname</td>
</tr>
<tr><td></td></tr>
<tr>
<td class="label">Hostname/IP</td>
<td><input id="idTargetHostInput" name="target_host" size="20" type="text"/></td>
</tr>
<tr><td></td></tr>
<tr>
<td colspan="2" style="text-align:center;">
<input class="button" name="dns-lookup-php-submit-button" type="submit" value="Lookup DNS">
</input></td>
</tr>
<tr><td></td></tr>
<tr><td></td></tr>
</table>
</form>]
Raja@virus project cod %
```

Figure 10: scrapping data from web app

This is the data that extract from the web application using beautiful-soup library. This is the mark-up language of the target web application. there is lot of function use in web applications frontend side but this is form tag or forms function of specific defined link or url.

```
1  #!/usr/bin/env python
2
3  import requests
4  from bs4 import BeautifulSoup
5
6  try:
7      import urlparse
8  except ImportError:
9      import urllib.parse as urlparse
10
11
12
13  def request(url):
14      try:
15          return requests.get(url)
16      except requests.exceptions.ConnectionError:
17          pass
18
19
20
21  target_url = "http://192.168.11.18/mutillidae/index.php?page=dns-lookup.php"
22  response = request(target_url)
23
24  parsed_html = BeautifulSoup(response.content)
25  forms_list = parsed_html.findAll("form")
26  print(forms_list)
```

Figure 11:extracting form code

After completing the first process we need to confirm the html's attributes for the method. The HTML/action Attribute is used to specify where the form data is to be sent to the server after submission of the form. It can be used in the <form> element. Payloads will be send by depending with the get or post method. Same as here we are using parsing library of the Python.

Before starting the process let us know about the get and post method. The HTTP GET request method is used to request a resource from the server. The GET request should only receive data (the server must not change its state)[13]. Same as POST is used to send data to a server to create and update a resource.

Here, the output of the 2nd unit's process:

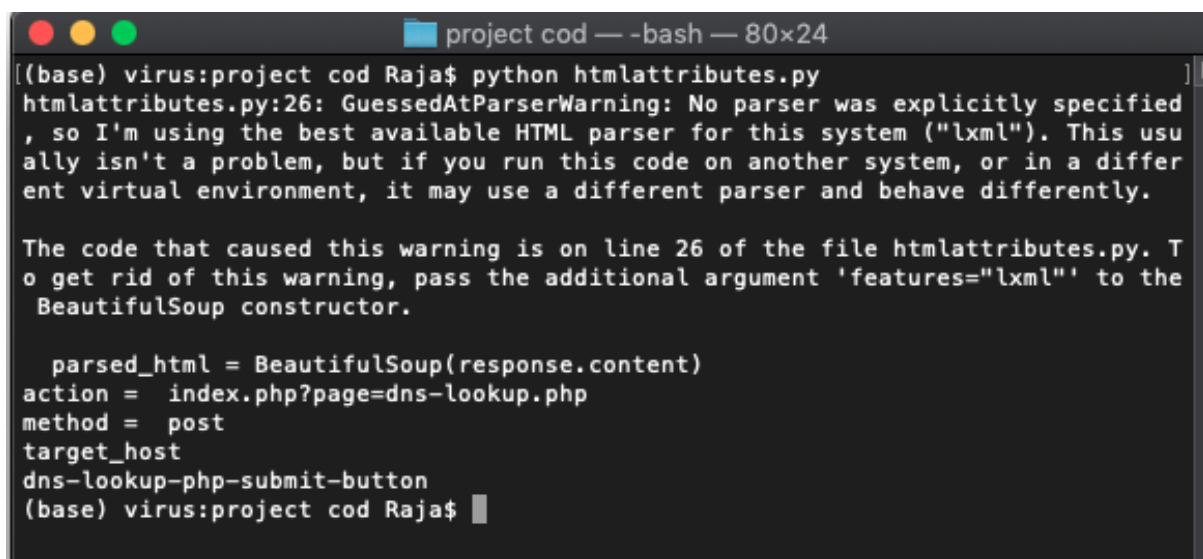
A terminal window titled 'project cod — -bash — 80x24' shows the execution of a Python script. The prompt is '(base) virus:project cod Raja\$'. The command 'python htmlattributes.py' is entered. The output shows a warning from BeautifulSoup about the parser used, followed by a detailed explanation of the warning and a suggestion to use 'features="lxml"'. Below this, the script prints the parsed HTML attributes: 'parsed_html = BeautifulSoup(response.content)', 'action = index.php?page=dns-lookup.php', 'method = post', and 'target_host = dns-lookup-php-submit-button'. The prompt returns to '(base) virus:project cod Raja\$'.

Figure 12: HTML attributes

In this figure, we can see the extract html's attributes which is used send form's data into server. We have action attributes which defining the link of page. The action attribute defines the action to be performed when the form is submitted[14]. Similarly, we can see the method = post which define method is post method. Post method is already defined in the last paragraph.

We already able to extract useable data from the web application by using first and second units. now we try to post data in those method of the application using third unit. This is preparation for sending payloads through the post method. Such as Search box, login box such boxes are post method. In the first unit we extract form element as well using second unit we access the method/action and some information.

In this unit we send data to the server combining first and second unit with this unit.

This is the code and web application to confirm that my scanner is working properly or not.

Piece of code and testing application:

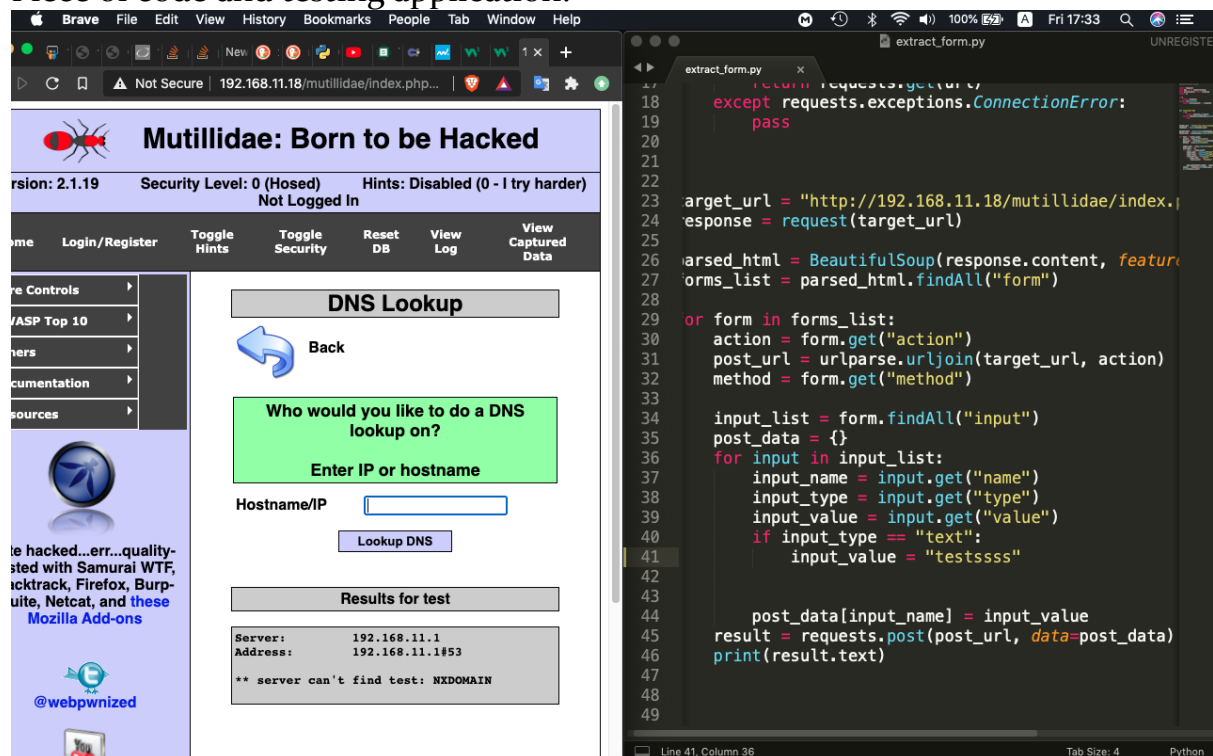
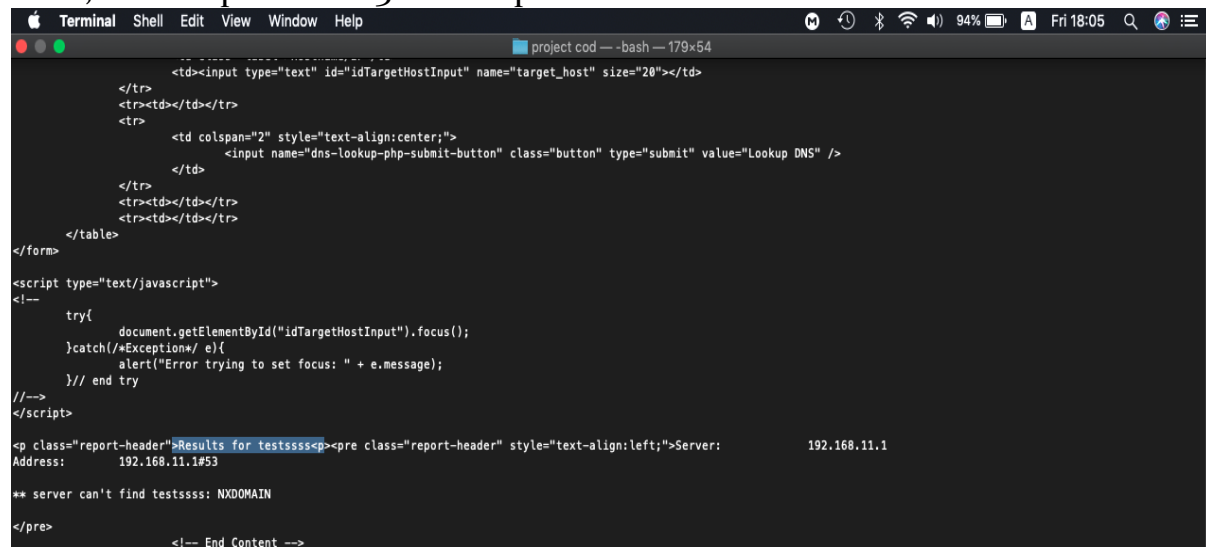


Figure 13: data posting code

There is different value posted in hostname of the website and similarly, posted value in line 41 of the code side.

Mutillidae is a free, open source web application provided to allow security enthusiasts to pen-test and hack a web application. Mutillidae contains dozens of vulnerabilities and hints to help the user exploit them; providing an easy-to-use web hacking environment deliberately designed to be used as a hack-lab for security enthusiasts, classroom labs, and vulnerability assessment tool targets. Mutillidae has been used in graduate security courses, in corporate web security training courses, and as an "access and the accessor" where access is the process of pulling data from the web site and accessor is person who access the data targeting for vulnerability software[15].

Here, the Output of the 3rd unit's process:

A screenshot of a macOS Terminal window. The title bar shows 'Terminal' and 'Shell Edit View Window Help'. The window content displays HTML code for a form with a text input and a submit button. Below the form, there is a JavaScript script that attempts to focus the input field. At the bottom, a pre-formatted HTML report header shows 'Results for testsssss' and 'Server: 192.168.11.1'. The report body indicates that the server cannot find 'testsssss' (NXDOMAIN).

```
<td><input type="text" id="idTargetHostInput" name="target_host" size="20"></td>
</tr>
<tr>
<td colspan="2" style="text-align:center;">
<input name="dns-lookup-php-submit-button" class="button" type="submit" value="Lookup DNS" />
</td>
</tr>
</table>
</form>
<script type="text/javascript">
<!--
    try{
        document.getElementById("idTargetHostInput").focus();
    }catch(Exception e){
        alert("Error trying to set focus: " + e.message);
    }
    // end try
-->
</script>
<p class="report-header">Results for testsssss</p>
<pre class="report-header" style="text-align:left;">Server: 192.168.11.1
Address: 192.168.11.1#53
** server can't find testsssss: NXDOMAIN
</pre>
<!-- End Content -->
```

Figure 14: data posted

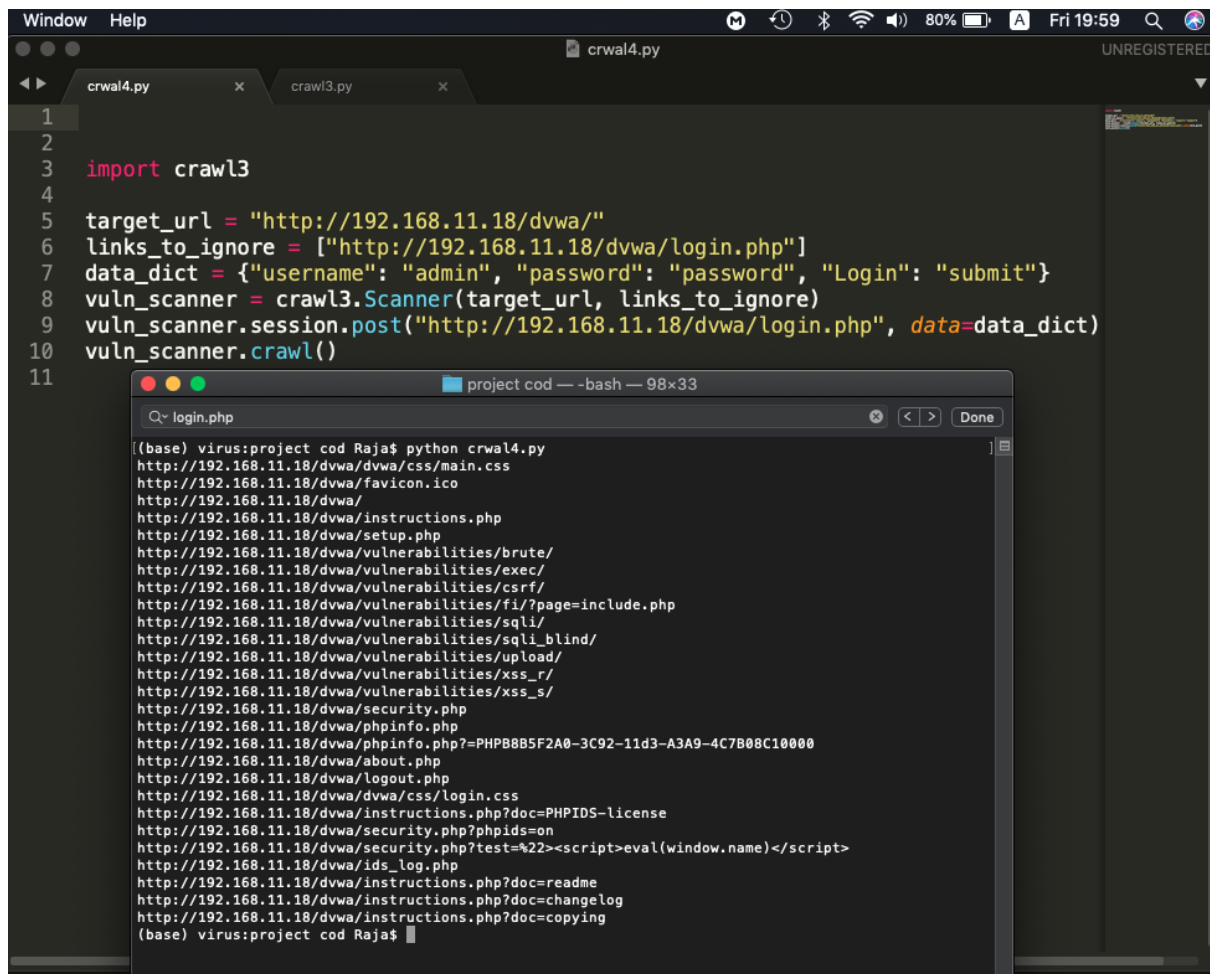
That web application reflect the insert data. We can see in figure (13) when I insert value as test in web application it response us “results for test” similarly, when I send value through the written program’s output are same. Here, we are able to send value to web application through the post method of the form. For the get method we need to extract the link to use get method. To extract the get method we need to use crawl technique.

Crawl technique, Web crawling and web scraping are two different but related concepts. Web crawling is a component of web scraping, the crawler logic finds URLs to be processed by the scraper code.

A web crawler starts with a list of URLs to visit, called the seed. For each URL, the crawler finds links in the HTML, filters those links based on some criteria and adds the new links to a queue.

Similarly, In this research project we used crawl technique to extract link. In this technique we can filter urls. Like, if we want to ignore the some unnecessary url from the output we can.

We can see in our scanner output.

The image shows a screenshot of a computer screen with a code editor and a terminal window. The code editor at the top has a dark theme and shows a Python script named 'crwal4.py'. The script imports 'crawl3', sets a 'target_url' to 'http://192.168.11.18/dvwa/', defines a 'links_to_ignore' list containing 'http://192.168.11.18/dvwa/login.php', and creates a 'data_dict' with login credentials. It then uses 'crawl3.Scanner' to scan the target URL, ignoring the specified link, and posts the login data. The terminal window below shows the execution of 'python crwal4.py', which outputs a list of URLs found on the DVWA application, excluding the login.php page as specified in the script. The terminal output includes URLs for various files like 'main.css', 'favicon.ico', 'instructions.php', 'setup.php', and several vulnerability endpoints like 'brute/', 'exec/', 'csrf/', 'fi/?page=include.php', 'sqli/', 'sqli_blind/', 'upload/', 'xss_r/', and 'xss_s/'. It also lists security-related pages like 'security.php', 'phpinfo.php', and 'about.php', and ends with 'ids_log.php' and 'instructions.php' with different document types like 'readme' and 'changelog'.

```
1
2
3 import crawl3
4
5 target_url = "http://192.168.11.18/dvwa/"
6 links_to_ignore = ["http://192.168.11.18/dvwa/login.php"]
7 data_dict = {"username": "admin", "password": "password", "Login": "submit"}
8 vuln_scanner = crawl3.Scanner(target_url, links_to_ignore)
9 vuln_scanner.session.post("http://192.168.11.18/dvwa/login.php", data=data_dict)
10 vuln_scanner.crawl()
11
```

```
(base) virus:project cod Raja$ python crwal4.py
http://192.168.11.18/dvwa/css/main.css
http://192.168.11.18/dvwa/favicon.ico
http://192.168.11.18/dvwa/
http://192.168.11.18/dvwa/instructions.php
http://192.168.11.18/dvwa/setup.php
http://192.168.11.18/dvwa/vulnerabilities/brute/
http://192.168.11.18/dvwa/vulnerabilities/exec/
http://192.168.11.18/dvwa/vulnerabilities/csrf/
http://192.168.11.18/dvwa/vulnerabilities/fi/?page=include.php
http://192.168.11.18/dvwa/vulnerabilities/sqli/
http://192.168.11.18/dvwa/vulnerabilities/sqli_blind/
http://192.168.11.18/dvwa/vulnerabilities/upload/
http://192.168.11.18/dvwa/vulnerabilities/xss_r/
http://192.168.11.18/dvwa/vulnerabilities/xss_s/
http://192.168.11.18/dvwa/security.php
http://192.168.11.18/dvwa/phpinfo.php
http://192.168.11.18/dvwa/phpinfo.php?PHPBB5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://192.168.11.18/dvwa/about.php
http://192.168.11.18/dvwa/logout.php
http://192.168.11.18/dvwa/dvwa/css/login.css
http://192.168.11.18/dvwa/instructions.php?doc=PHPIOS-license
http://192.168.11.18/dvwa/security.php?phpids=on
http://192.168.11.18/dvwa/security.php?test=%22<script>eval(window.name)</script>
http://192.168.11.18/dvwa/ids_log.php
http://192.168.11.18/dvwa/instructions.php?doc=readme
http://192.168.11.18/dvwa/instructions.php?doc=changelog
http://192.168.11.18/dvwa/instructions.php?doc=copying
(base) virus:project cod Raja$
```

Figure 15:crawl url

Here, we have new function as data_dict to submit authentication information. As I said before this is the url of the web application that extract from the web using crawl technique.

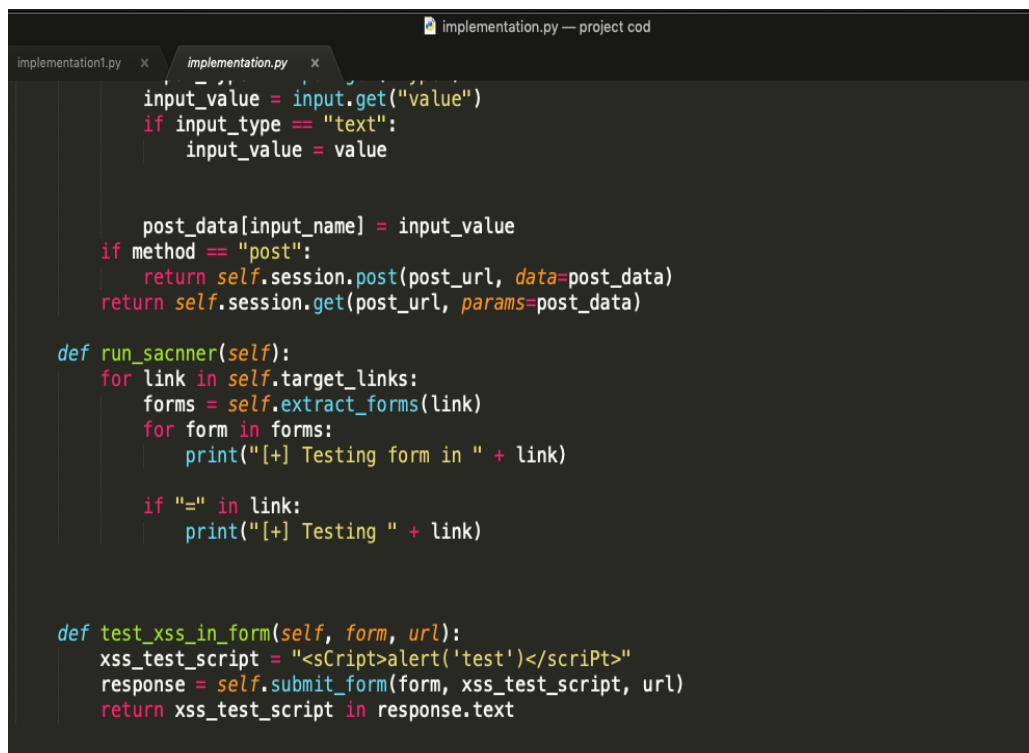
Before, we had done all work without login system. We didn't use authentication system in figure 11's web application. we can use crawl system in every types of web application and sites. Here, figure 11 and figure 13 is different web application because in figure 11's application don't have user login system to send authentication but in figure 13's we send user authentication data defining as "data_dict" variable.

I already said we can filter URLs using this technique where I am ignoring authentication's link because I already send its data and I don't want to check anything in this link. We can see in figure 13, in the output of the terminal I try to find the login.php link which is not available in whole output because we have defined it in our system to ignore this link. Tested web is made using php language that's why we using .php form. we can ignore and define such page or link depending on the web application's language. These all are the possible links which are vulnerable in this application.

We are able to login and crawl link together through the our unit. We are to connect with the front-end side of the application but what about the server side, we need to do same process that before we did in different web applications.

In that application, username and user password is not necessary to enter in application but in figure 13's username and password are required to enter in the application. likewise, we cannot to scrap data without login web application.

Here, we send data again using process figure (13). if the sent payloads is accepted in specific form the response will be true otherwise the response will be false or none. Let's see in figure (16) with specific part of the code,

A screenshot of a code editor window titled 'implementation.py - project cod'. The editor shows Python code for a web vulnerability scanner. The code includes a function to handle form submissions, a main scanning loop, and a specific function for testing XSS payloads. The code is as follows:

```
input_value = input.get("value")
if input_type == "text":
    input_value = value

post_data[input_name] = input_value
if method == "post":
    return self.session.post(post_url, data=post_data)
return self.session.get(post_url, params=post_data)

def run_sacnner(self):
    for link in self.target_links:
        forms = self.extract_forms(link)
        for form in forms:
            print("[+] Testing form in " + link)

            if "=" in link:
                print("[+] Testing " + link)

def test_xss_in_form(self, form, url):
    xss_test_script = "<script>alert('test')</script>"
    response = self.submit_form(form, xss_test_script, url)
    return xss_test_script in response.text
```

Figure 16: Payloads test

We can see here, we have test_cross_site_form in last function to check that the data is acceptable in form or not. We must not be confused with the submit form which already shown this code in figure 11. We have used Boolean in the test_cross_site_form method which output will be true or false. If the payloads is accepted the output will be true if not output will be false. I have used response.text and response.content for the output. It depend in output content depending on our machine.

I already mentioned about cross site script earlier in our paper. In figure 14's last method we have xss_test_script variable to define XSS which is sample xss code for the test our scanner .

```

1 import implementation
2
3 target_url = "http://192.168.11.18/dvwa/"
4 links_to_ignore = ["http://192.168.11.18/dvwa/login.php"]
5 data_dict = {"username": "admin", "password": "password", "Login": "submit"}
6 vuln_scanner = implementation.Scanner(target_url, links_to_ignore)
7 vuln_scanner.session.post("http://192.168.11.18/dvwa/login.php", data=data_dict)
8 forms = vuln_scanner.extract_forms("http://192.168.11.18/dvwa/vulnerabilities/xss_r/")
9 print(forms)
10 response = vuln_scanner.test_xss_in_form(forms[0], "http://192.168.11.18/dvwa/vulnerabilities/xss_r/")
11 print(response)

```

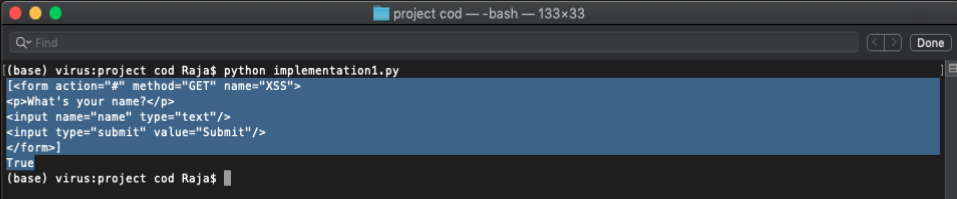
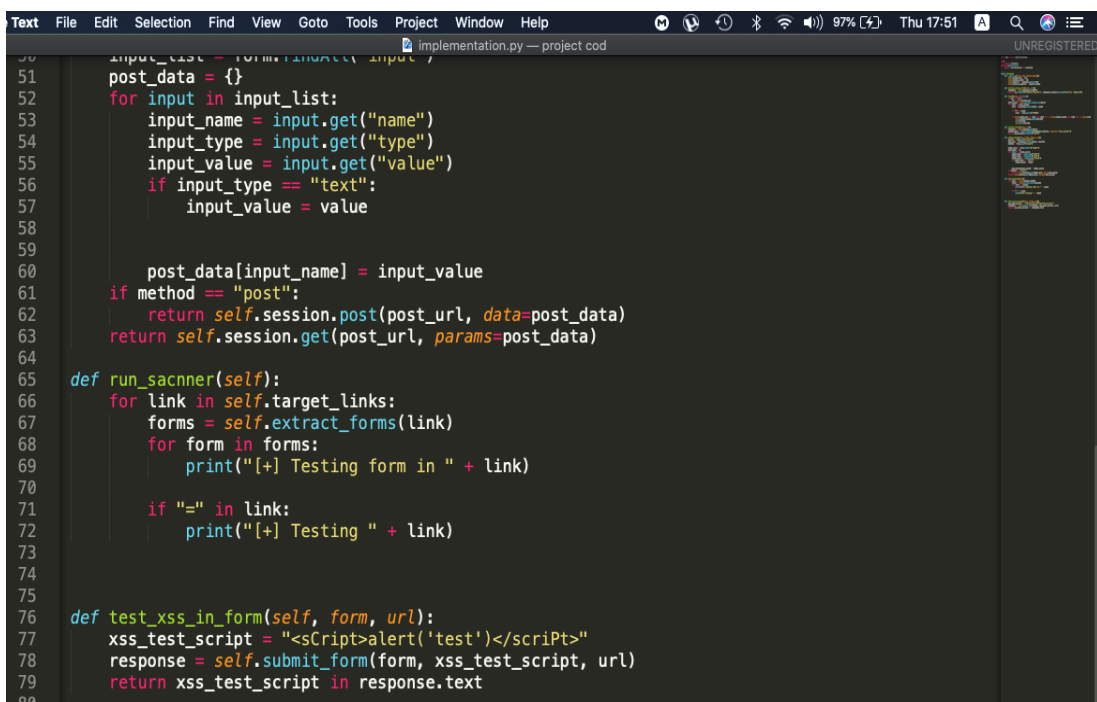


Figure 17: Output of payloads acceptance

As I mentioned, the output will be true or false. We can see true in output if the submitted data has been accepted by application. I think there is more form in this specific url but we have send data in first form, index 0 of the list. We can see tested form as well condition's output.

We are able to send payloads in form as well we can send payloads through the url too.



```

51 post_data = {}
52 for input in input_list:
53     input_name = input.get("name")
54     input_type = input.get("type")
55     input_value = input.get("value")
56     if input_type == "text":
57         input_value = value
58
59     post_data[input_name] = input_value
60 if method == "post":
61     return self.session.post(post_url, data=post_data)
62 return self.session.get(post_url, params=post_data)
63
64
65 def run_scanner(self):
66     for link in self.target_links:
67         forms = self.extract_forms(link)
68         for form in forms:
69             print("[+] Testing form in " + link)
70
71         if "=" in link:
72             print("[+] Testing " + link)
73
74
75
76 def test_xss_in_form(self, form, url):
77     xss_test_script = "<script>alert('test')</script>"
78     response = self.submit_form(form, xss_test_script, url)
79     return xss_test_script in response.text
80

```

Figure 18: payloads send through the link

Same as we can send same XSS script through the link. As all we know about the website, if site going to post anything through or change content it is defined through the "=" this sign like, as we can see in figure :

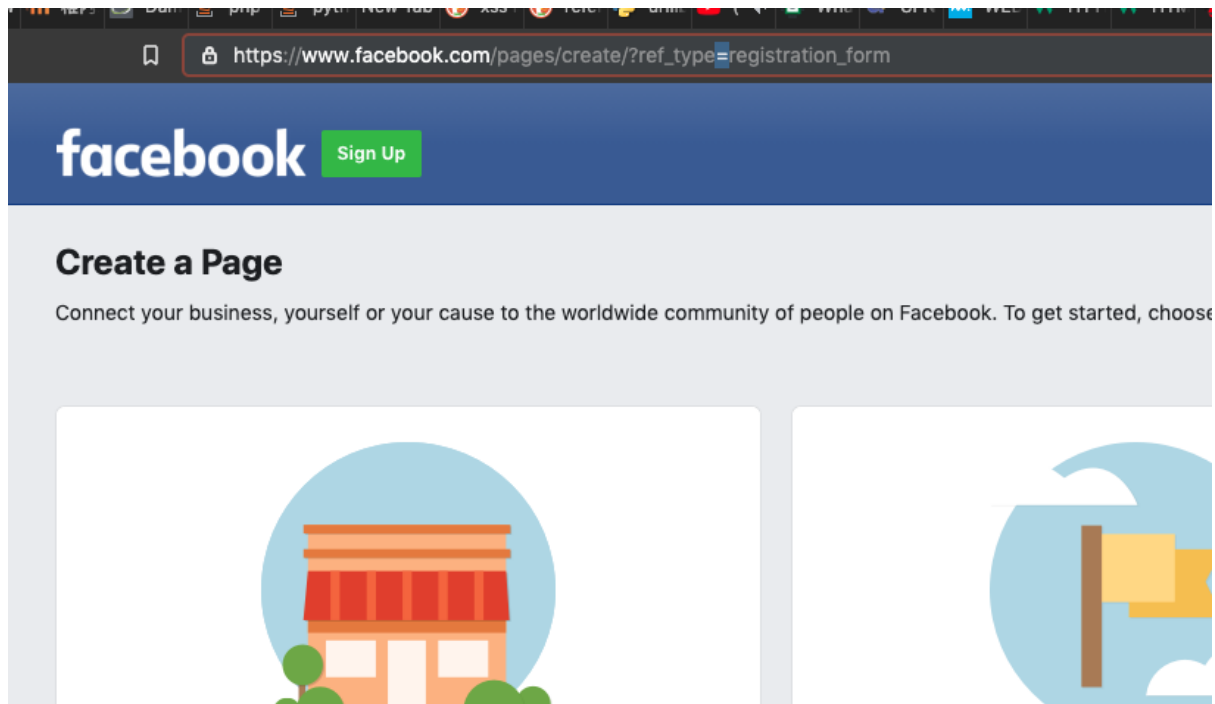


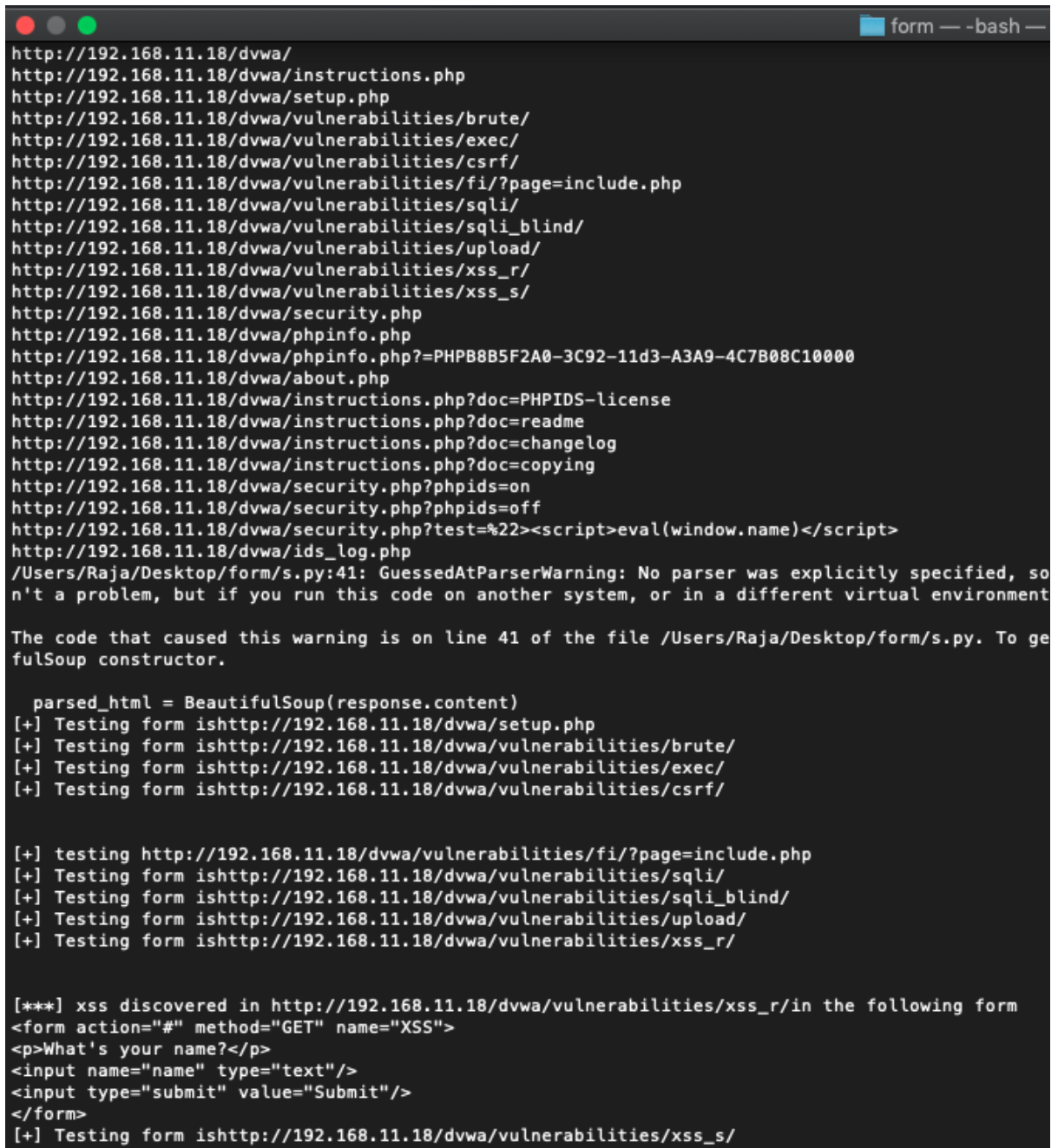
Figure 19:sample of output

We can see in figure there is “=” sign if we insert data in then it will go for register in server of the fakebook and save in fakebook’s database. For the extra information Database injection also done through the same place. But in this research I just use it for reflection of the data.

We decided to check vulnerability form the link using same method if we want to try in figure 17’s link through our unit, it start replace from the equal “=” sign same written in our figure 16’s code of the second last method. Where equals is equal after concatenate with xss_test_script.

6.5 Deployment:

We have tested all unit which ready for work in real scenario we extract form and crawl link send payloads in different form of the application. finally we are able to do all work which we need to do for one reflect vulnerability scanning tool. We scan every link and every form of the page. If the form or link reflect the data the link and form will be vulnerable. We can see the whole systems output in this figure.



```
http://192.168.11.18/dvwa/
http://192.168.11.18/dvwa/instructions.php
http://192.168.11.18/dvwa/setup.php
http://192.168.11.18/dvwa/vulnerabilities/brute/
http://192.168.11.18/dvwa/vulnerabilities/exec/
http://192.168.11.18/dvwa/vulnerabilities/csrf/
http://192.168.11.18/dvwa/vulnerabilities/fi/?page=include.php
http://192.168.11.18/dvwa/vulnerabilities/sqli/
http://192.168.11.18/dvwa/vulnerabilities/sqli_blind/
http://192.168.11.18/dvwa/vulnerabilities/upload/
http://192.168.11.18/dvwa/vulnerabilities/xss_r/
http://192.168.11.18/dvwa/vulnerabilities/xss_s/
http://192.168.11.18/dvwa/security.php
http://192.168.11.18/dvwa/phpinfo.php
http://192.168.11.18/dvwa/phpinfo.php?PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://192.168.11.18/dvwa/about.php
http://192.168.11.18/dvwa/instructions.php?doc=PHPIDS-license
http://192.168.11.18/dvwa/instructions.php?doc=readme
http://192.168.11.18/dvwa/instructions.php?doc=changelog
http://192.168.11.18/dvwa/instructions.php?doc=copying
http://192.168.11.18/dvwa/security.php?phpids=on
http://192.168.11.18/dvwa/security.php?phpids=off
http://192.168.11.18/dvwa/security.php?test=%22><script>eval(window.name)</script>
http://192.168.11.18/dvwa/ids_log.php
/Users/Raja/Desktop/form/s.py:41: GuessedAtParserWarning: No parser was explicitly specified, so
n't a problem, but if you run this code on another system, or in a different virtual environment

The code that caused this warning is on line 41 of the file /Users/Raja/Desktop/form/s.py. To ge
fulSoup constructor.

    parsed_html = BeautifulSoup(response.content)
[+] Testing form ishttp://192.168.11.18/dvwa/setup.php
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/brute/
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/exec/
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/csrf/

[+] testing http://192.168.11.18/dvwa/vulnerabilities/fi/?page=include.php
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/sqli/
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/sqli_blind/
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/upload/
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/xss_r/

[***] xss discovered in http://192.168.11.18/dvwa/vulnerabilities/xss_r/in the following form
<form action="#" method="GET" name="XSS">
<p>What's your name?</p>
<input name="name" type="text"/>
<input type="submit" value="Submit"/>
</form>
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/xss_s/
```

Figure 20:output of the scanner 1st

Contd;

```
<form action="#" method="GET" name="XSS">
<p>What's your name?</p>
<input name="name" type="text"/>
<input type="submit" value="Submit"/>
</form>
[+] Testing form ishttp://192.168.11.18/dvwa/vulnerabilities/xss_s/

[***] xss discovered in http://192.168.11.18/dvwa/vulnerabilities/xss_s/in the following form
<form method="post" name="guestform" onsubmit="return validate_form(this)">
<table border="0" cellpadding="2" cellspacing="1" width="550">
<tr>
<td width="100">Name *</td> <td>
<input maxlength="10" name="txtName" size="30" type="text"/></td>
</tr>
<tr>
<td width="100">Message *</td> <td>
<textarea cols="50" maxlength="50" name="mtxMessage" rows="3"></textarea></td>
</tr>
<tr>
<td width="100"></td>
<td>
<input name="btnSign" onclick="return checkForm();" type="submit" value="Sign Guestbook"/></td>
</tr>
</table>
</form>
[+] Testing form ishttp://192.168.11.18/dvwa/security.php

[+] testing http://192.168.11.18/dvwa/phpinfo.php?PHPBB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

[+] testing http://192.168.11.18/dvwa/instructions.php?doc=PHPIDS-license

[+] testing http://192.168.11.18/dvwa/instructions.php?doc=readme

[+] testing http://192.168.11.18/dvwa/instructions.php?doc=changelog

[+] testing http://192.168.11.18/dvwa/instructions.php?doc=copying
[+] Testing form ishttp://192.168.11.18/dvwa/security.php?phpids=on

[+] testing http://192.168.11.18/dvwa/security.php?phpids=on
[+] Testing form ishttp://192.168.11.18/dvwa/security.php?phpids=off

[+] testing http://192.168.11.18/dvwa/security.php?phpids=off
[+] Testing form ishttp://192.168.11.18/dvwa/security.php?test=%22<script>eval(window.name)</script>

[+] testing http://192.168.11.18/dvwa/security.php?test=%22<script>eval(window.name)</script>
[+] Testing form ishttp://192.168.11.18/dvwa/ids_log.php
(base) virus:form Raja$
```

Figure 21:output of the scanner 2nd

We can see in figure 18, lots of link same as form where scanner vulnerability is checked in every form that available in web application writing as testing form and link of the page. And there is link which is tested. Tested link are written as “testing with link”. We have output “testing form” is for web form.

Similarly, we have “ testing ” for link, This is the whole output of the system. If we scanner detect the vulnerability in any form or link it will show we as “xss discovered in specific link”

We can see in both output where written as “xss is discovered in (specific url) in the following form. And it show form also in same row of the output.

Implementation

This is used for scanning of reflected vulnerability of the web application. This is not application nor software. This is a written Python's script to scan vulnerability of the web application. we can check vulnerability in both GET and POST method of the web application.

Now, it just work for reflected cross site script in both method. To implement this scanner we don't need any cost But all of we must have Python's programming knowledge to edit normal thing depend on the web application. We need to learn little bit more for both web and object oriented programming to become one script editor and user of this tool but it reduce cost and unnecessary extra information for the fresh student and we will be able to start the journey of security without facing defined problem. As the view of system require for the scanner all of we must have installed Python version 3 and its library in our device.

Similarly, we must have lot of web application's mechanism knowledge to recognize method and working flow.

In web sites there are lots of web security weakness which can be accessed by third person in order to access the data and sale it on black markets. There are a lot of platforms emerging to minimize the risks such as hacker one bounty program. In order to find the vulnerability the basic and the first thing to do as a security specialist to check for the loopholes using cross site scripting. My research is based on finding the vulnerabilities in web apps by sending payloads and scanning responses for the possible vulnerability.

Conclusion and future work

7.1 Conclusion

This vulnerability scanner is developed to scan web vulnerability by using Python programming language and its library such as beautiful-soup, regular expression and universal resource locator parse etc. it works as scanning tool. The main purpose of this scanner is to scan reflected cross site script vulnerability of the web application. This system is developed to motivate those student who wants to learn security. Without facing financial and heavy informative problem. we can use it any web application but we need change little bit information depend on the web application.

I believe with my work this can motivate to all those student like me who face lot of trouble while start to learn security. Implementing this tool would make easy and satisfying to all those person.

7.2 Future work

Till today web vulnerability scanner is developed. It is one part of the non-persistent attack. In this scanner, we scan reflected vulnerability in post/get method. This can be limitation of the project. We have confusions at starting while thinking about project because we started hacking and leaved because of cybercrime. At that time we didn't have the application and professional tools to test. Then we changed our project into scanning tool we started making tools. we succeed to scan vulnerability but Due to limitation of time we are unable to make data stored cross site script tool. As my future work, scanner will be more advance including data stored cross site script scanner as well database injection.

Reference:

1. Online(accessed on May 19): <https://portswigger.net/burp/vulnerability-scanner/guide-to-vulnerability-scanning>
2. Online(accessed on May 19): <https://www.zeguro.com/blog/what-is-a-website-vulnerability-scanner-and-why-should-you-use-one>
3. Online(accessed on May 19): <https://portswigger.net/burp/vulnerability-scanner/guide-to-vulnerability-scanning>
4. Online(accessed on May 26): <https://www.educative.io/blog/object-oriented-programming>
5. Online(accessed on May 26): <https://www.educba.com/is-Python-open-source/>
6. Online(accessed on May 29): <https://intellipaat.com/blog/advantages-and-disadvantages-of-Python/>
7. Online(accessed on Jun 3): <https://programminghistorian.org/en/lessons/intro-to-beautiful-soup>
8. Online(accessed on Jun 4): <https://www.analyticsvidhya.com/blog/2021/06/web-scraping-with-Python-beautifulsoup-library/>
9. Online: (accessed on Jun 12): https://en.wikipedia.org/wiki/Regular_expression#Expressive_power_and_compactness
10. Online: (accessed on Jun 13):<https://www.edureka.co/blog/Python-requests-tutorial/>
11. Online(accessed on Jun 17): <https://www.imperva.com/learn/application-security/reflected-xss-attacks/>
12. Online(accessed on Jun 26):<http://sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-unified-list.html>
13. Online(accessed on Jun 29): https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
14. Online(accessed on July 3): <https://reqbin.com/Article/HttpGet>
15. Online(accessed on July 5): https://www.w3schools.com/html/html_forms_attributes.asp
16. Online(accessed on July): <https://www.irongeek.com/i.php?page=mutillidae/mutillidae-deliberately-vulnerable-php-owasp-top-10>

Appendix:

Main.py

```
import requests
import re
from urllib.parse import urlparse
from bs4 import BeautifulSoup

try:
    import urlparse
except ImportError:
    import urllib.parse as urlparse

class Scanner:
    def __init__(self, url, ignore_links):
        self.session = requests.Session()
        self.target_url = url
        self.target_links = []
        self.links_to_ignore = ignore_links

    def extract_links_from(self, url):
        response = self.session.get(url)
        return re.findall('(?:href=")(.*?)"', response.content.decode('utf-8',
'ignore'))

    def crawl(self, url=None):
        if url is None:
            url = self.target_url
        href_links = self.extract_links_from(url)
        for link in href_links:
            link = urlparse.urljoin(url, link)

            if "#" in link:
                link = link.split("#")[0]

            if self.target_url in link and link not in self.target_links and link
not in self.links_to_ignore:
                self.target_links.append(link)
                print(link)
                self.crawl(link)

    def extract_forms(self, url):
        response = self.session.get(url)
        parsed_html = BeautifulSoup(response.content)
        return parsed_html.findAll("form")

    def submit_form(self, form, value, url):
```

```

        action = form.get("action")
        post_url = urlparse.urljoin(url, action)
        method = form.get("method")

        input_list = form.findAll("input")
        post_data = {}
        for input in input_list:
            input_name = input.get("name")
            input_type = input.get("type")
            input_value = input.get("value")
            if input_type == "text":
                input_value = value

            post_data[input_name] = input_value

        if method == "post":
            return self.session.post(post_url, data=post_data)
        return self.session.get(post_url, params=post_data)

    def run_scanner(self):
        for link in self.target_links:
            forms = self.extract_forms(link)
            for form in forms:
                print("[+] Testing form is" + link)
                is_vulnerable_to_xss = self.test_xss_in_form(form, link)
                if is_vulnerable_to_xss:
                    print("\n\n[***] xss discovered in " + link + "in the
following form")

                    print(form)

                if "=" in link:
                    print("\n\n[+] testing " + link)
                    is_vulnerable_to_xss = self.test_xss_in_link(link)
                    if is_vulnerable_to_xss:
                        print("[***] discovered XSS in " + link)

    def test_xss_in_link(self, url):
        xss_test_script = "<sCriPt>alert('test')</scriPt>"
        url = url.replace("=", "=" + xss_test_script)
        response = self.session.get(url)
        return xss_test_script in response.text

    def test_xss_in_form(self, form, url):
        xss_test_script = "<sCriPt>alert('test')</scriPt>"
        response = self.submit_form(form, xss_test_script, url)
        return xss_test_script in response.text

```

Address.py

```
import Main

target_url = "http://192.168.11.18/mutillidae/"
links_to_ignore = ["http://192.168.11.18/dvwa/logout.php"]

target_url = "http://192.168.11.18/dvwa/"
data_dict = {"username": "admin", "password": "password", "Login": "submit"}

vuln_scanner = Main.Scanner(target_url, links_to_ignore)
response = vuln_scanner.session.post("http://192.168.11.18/dvwa/login.php",
data=data_dict)
vuln_scanner.crawl()
vuln_scanner.run_scanner()
```