Advanced Detection and Mitigation of Cross-Site Scripting Attacks through Intelligent JavaScript Code Injection Analysis

Ashmit Gupta[1], Neeraj Srivastava[2], Ajit Kumar Srivastava[3]
[1] Student Member IEEE, School of Computer Science and Engineering Vellore Institute of Technology, Vellore, India.
[2] Senior Member IEEE , Associate Professor, Department of Electronics & Communication Engineering, Rustamji Institute of Technology, BSF Academy Tekanpur Gwalior, India.
[3] Associate Professor & Head Department of CSESIS Tec-R, Bhopal, India.

Abstract: Cross-site scripting attack, abbreviated as XSS , has been an incessant problem for Web applications since the early 2000s. It is a code injection attack on the client -side where an attacker injects malicious payload intoavulner able we b applic ation, without client's knowledge. The attacker is often successful in executing the malicious code inadvertently in the browser of an unwary user. Attempts have been made to implement the detection of XS S attacks using Genetic Algorithm, Web Vulnerabilities Finder, Fuzzy Interference Model, but they all come with drawbacks. Implementation URL of the website is collected through an extension and vulnerability is checked by injecting java-script code to the website. If the website is vulnerable, then display a pop -up stating "Website is vulnerable, be aware", else display a pop -up stating " W ebsite is not vulner able ". It is a low cost model which is easy to implement.

Keywords: XSS , F I Model, Genetic Algorithm, Web Vulnerabilities Finder, Cyber Attack.

I.INTRODUCTION

W eb applications and websites are becoming more and more widespread as a result of increased internet usage, which has also led to an increase in cyber attacks on web applications and websites. XS S (Cross -Site Scripting) attacks are one of the most prevalent forms of cyber attack on web applications and websites out of all the other sorts of attacks.

Cross Site Scripting (XS S) is vulnerability in a web application that allows a third party to execute a script in the user's browser on behalf of the web application. Malicious scripts are injected into websites that are normally safe and reputable in Cross-Site Scripting (XS S)attacks. XS S attacks take place when an attacker sends malicious code, typically in the form of a browser side script, to a separate end user using an online application. It allows an attacker to by-pass the origin policy that is designed to segregate different websites from each other.

An attacker who exploits this vulnerability can:
- Read any d ata that the user is able to access
- Capture user's login credentials
- Impersonate or masquerade as the victim user.

Cross-site scripting attacks enable attackers to inject client -side scripts into web pages view ed by the user. The scripts are executed automatically without user's consent, unless manually disabled. Besides, the users don't seem to care about configuring the browsers securely. The number of attacks on users, by exploiting the browser vulnerabilities, has risen at alarming rate.

Already existing attack detecting mechanisms have failed miserably in most scenarios. Additionally, people haven' t paid much attention to configuring their browsers securely with the help of accessible plug-ins and extensions.

The users have to protect themselves from these vulnerabilities and hence the detection of Cross-site scripting attacks is very important .to effectively detect and defend XS S attacks are still one of the most important security issues. Therefore, we need better solutions/ methods for detection of Cross-site Scripting (XS S) attacks.

II.LITERATURE SURVEY

Zhonglin Liu, Yong Fang, Cheng Huang andYijia Xu (2022) - Genetic Algorithm

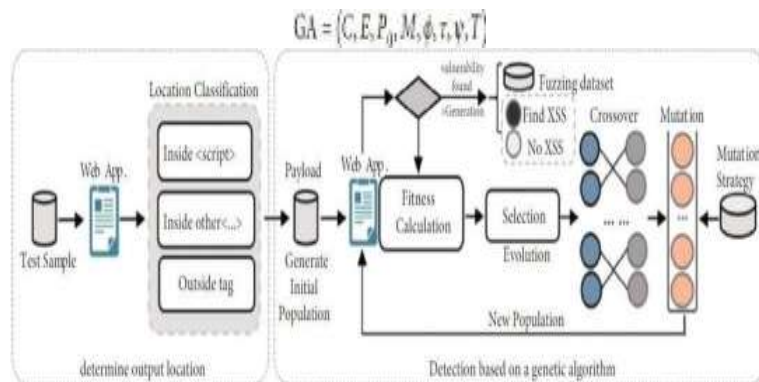$$GA = \{C, E, P_0, M, \phi, \tau, \psi, T\}$$

FIGURE 2: The system architecture of GAXSS.

FIGURE 2: Mamdani fuzzy inference system

- ❑ Genetic algorithm starts from an initial population, through random selection, crossover and mutation.
- ❑ The algorithm generates a group of individuals more suitable for the environment so that the group evolves to better are as in the exploration space.
- ❑ In the process of exploration, obtaining the local optimal solution is not easy.

Drawback: Genetic algorithm is random and hence a large amount of time is consumed in detecting vulnerabilities.

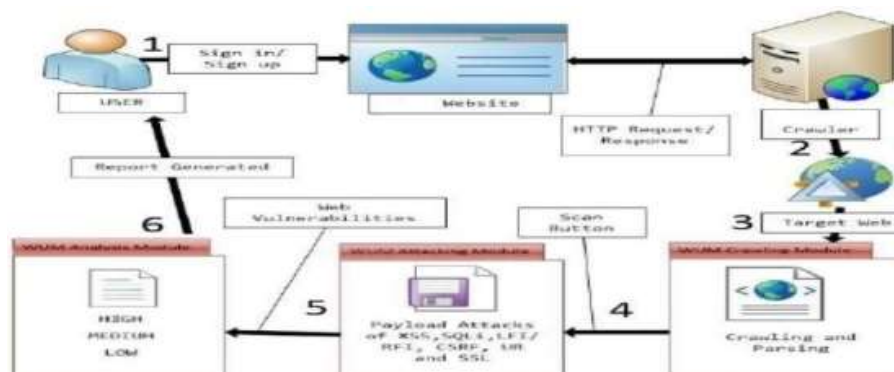**Muhammad Noman Khalid, Muhammad Iqbal, Kamran Rasheed, Malik MuneebAbid (2020) - Web**

Fig.1. The Architecture of WVF scanner

Vulnerabilities Finder (WVF)

- ❑ W eb Vulnerabilities Finder, abbreviated as W VF , is a scanning tool capable of performing efficient penetration tests on php and websites with ".net" domain to identify web vulnerabilities.
- ❑ It is capable of finding hidden S QLi and XS S vulnerabilities.
- ❑ It comprises of : Crawling Module, Attack Module and Analysis Module

Drawback: WVF sometimes generates false positives and false negatives.

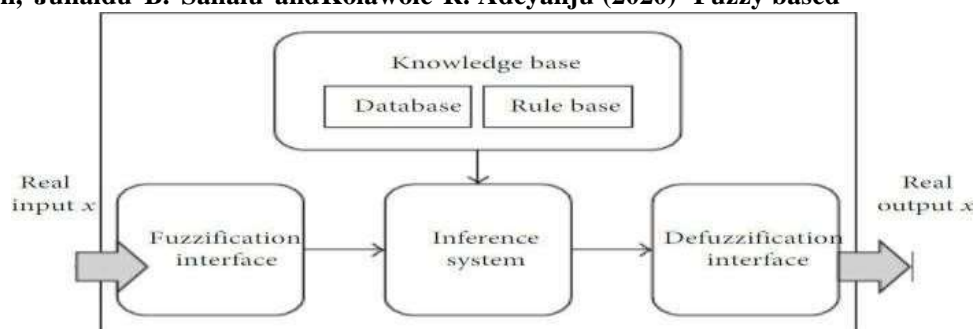**Bakare K. Ayeni, Junaidu B. Sahalu andKolawole R. Adeyanju (2020)- Fuzzy based**

FIGURE 2: Mamdani fuzzy inference system

F uzzy-based appro ach contributes to :
- ❑ S election and implementation of DO M-based features for XS S detection.

❑ Application of the fuzzy logic inferenc e system to detect vulnerabilities in web applications.
❑ Implementation of the UI that gives users an idea about the level of exposure to XS S attacks while visiting a website.
Drawback : In the F uzzy Interferenc e Model, many important categories of vulnerabilities are triggered by unexpected user inputs and can appear anywhere within the application.

Jingchi Zhang, Yu-Tsern Jou, Xiangyang Li (2019)
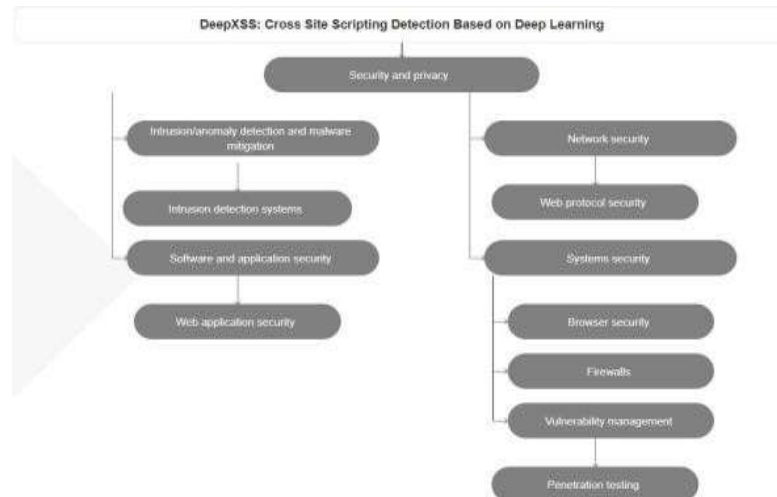
❑ A probabilistic model, Gaussian mixing model, posits that all of the data points were produced by combining a limited number of Gaussian distributions with unknowable parameters.
❑ The expectation maximization (EM) approach for fitting a mixture of Gaussian models is implemented by the Gaussian Mixture o bje ct.
❑ A Gaussian Mixture Model can be learned from train data using the Gaussian Mixture .fit technique.
❑ Using the Gaussian Mixture predict technique and test data, it may assign each sample the Gaussian that it most likely belongs to.

Yong Fang, Yang Li , Liang Liu, Cheng Huang(2021) – Deep XSS
❑ Deep XS S is to detect XSS attacks based on deep learning.
❑ It uses wor d2v ec in ord er to extra ct the word order inform ation fro m XS S payloads and map each payload to a feature vector.
❑ According to experimental findings, the proposed deep learning-based XS S detection model obtains a precision rate of 99.5% and a rec all rate of 97.9 % in real datasets.
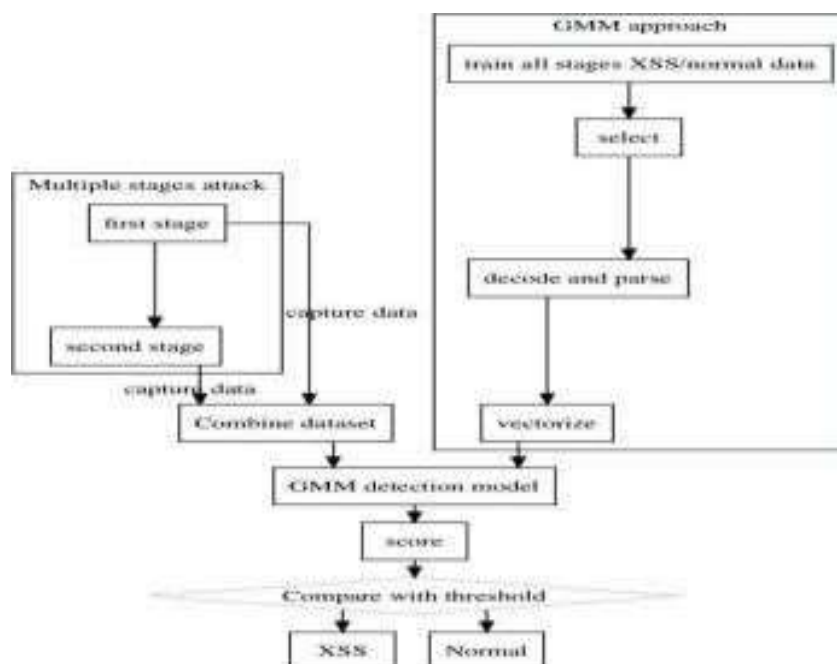
III.PROPOSED SYSTEM

All of the XS S prevention tools come into scenario, when an XS S attack is being performed on a website or has been successfully executed. The draw backs of this are:

❑ Loss on mon ey
❑ Loss of time
❑ Loss of res our ces
❑ Dent to company's reputation
❑ Loss of data
❑ Threat to integrity and confidentiality

Approach (What Did We Do?)

❑ S o, you can call our approach as "prev ention is better than cure".
❑ Basically, our approach is to make a XS S detection extension which can be added to any browser and it identifies all the forms from that web page using web scrapping and get all those forms which has input type as text and submit method as post or get requests.
❑ Now we will inject a random qu ery into the inputs.
❑ As we kno w that for ev er y requ est to the server there is a response from it.
❑ If the response is negative then we can say that the website is not vulnerable.
❑ If the response is positive, it means that the website is vulnerable and we will display to the user, the exact form in which there is vulnerability with parameters including:

-Request method
-Input type
-Res pons e fr o m se rve r

IV.METHODOLOGY

We have build an extension as a json file and have implemented detection of XS S attack using python and node.js code. Using the extension we find out all the inputs types and check if it contains malicious code. If yes then we print the malicious sites in the extension window. This will tell the user that this website is malicious and the user will not fill any details in the website. In this way our extension will prevent the user from getting attacked by cross site scripting attack.

1. User has to login to the account.
2. User has to download and install XS S _Vulnerability_Extension on the web browser.
3. User must open any webpage after installation of the extension.
4. Extension captures the URL of currently opened tab.
5. URL is sent to the created backend.
6. URL is extracted in the backend and is passed to the scanner python file.
7. The Beautiful S oup library in the python code detects forms on the web page.
8. A dummy java-script is created to inject into the web page.
9. All the forms present on the webpage are traversed thoroughly.
10. F orm details like their action, method and inputs are extracted.
11. F orm is submitted using all the collected details and java-script is inserted in the text input field.
12. If the script gets inserted to the web page then, it is vulnerable, else it is not vulnerable.

V.BLOCK DIAGRAM
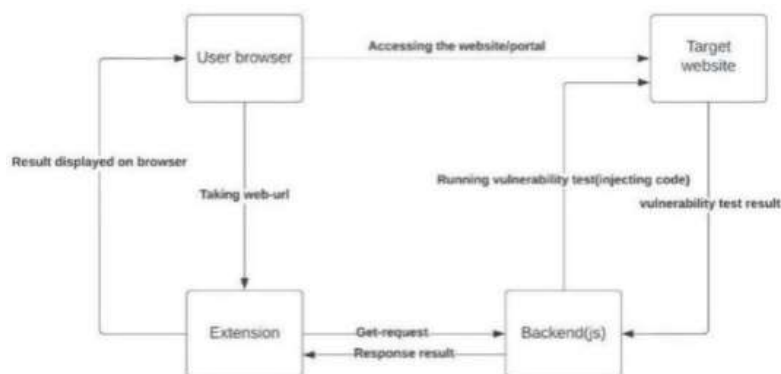
VI.FLOW CHART
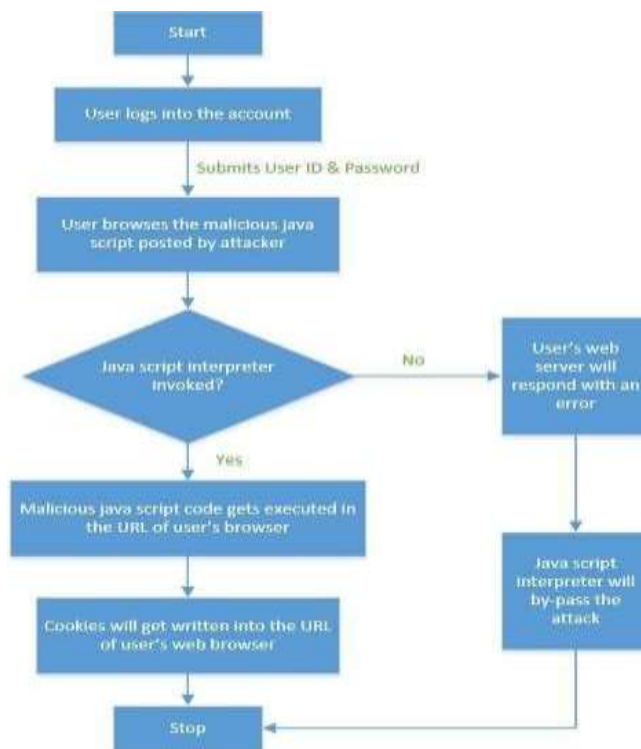
Python script for detecting the vulnerability
• **Libraries needed**

```
import sys
sys.path.insert(0, C:/Users/LEGION/AppData/Local/Programs/Python/Python39/Lib/site-packages )
from pprint import pprint
from bs4 import BeautifulSoup as bs
from urllib.parse import urljoin
```

Set the path to the directory where all paths have been downloaded.

• **Getting the forms**

```
def get_all_forms(url):
    soup = bs(requests.get(url).content, "html.parser")
    return soup.find_all("form")

def get_form_details(form):
    details = {}
    action = form.attrs.get("action").lower()
    method = form.attrs.get("method", "get").lower()
    inputs = []
    for input_tag in form.find_all("input"):
        input_type = input_tag.attrs.get("type", "text")
        input_name = input_tag.attrs.get("name")
        inputs.append({"type": input_type, "name": input_name})
    details["action"] = action
    details["method"] = method
    details["inputs"] = inputs
    return details
```

Select the forms having input type as text and methods as P OS T and GE T

```python
def submit_form(form_details, url, value):

    target_url = urljoin(url, form_details['action'])
    inputs = form_details['inputs']
    data = {}
    for input in inputs:
        if input['type'] == 'text' or input['type'] == 'search':
            input['value'] = value
        input_name = input.get('name')
        input_value = input.get('value')
        if input_name and input_value:
            data[input_name] = input_value

    if form_details['method'] == 'post':
        return requests.post(target_url, data=data)
    else:
        return requests.get(target_url, params=data)
```

Injecting the JS query into the inputs and posting it

```python
def scan_xss(url):
    forms = get_all_forms(url)
    js_script = "<Script>alert('hi')</scripT>"
    is_vulnerable = False
    for form in forms:
        form_details = get_form_details(form)
        content = submit_form(form_details, url, js_script).content.decode()
        if js_script in content:
            pprint(form_details)
            is_vulnerable = True
    return is_vulnerable
```

This function runs all above function in sequence

```python
if __name__ == "__main__":
    url = sys.argv[1]
    print(str(scan_xss(url)))
```

Main F unction Acc epts website URL as input

- **Server**

```javascript
const express = require('express')
const { spawn } = require('child_process')
const {PythonShell} = require('python-shell')
const app = express()
const cors = require('cors')
const port = 3000

app.use(function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');
  next();
})

app.get('/', async (req, res) => {

  let options={
    args: [req.query.link]
  }

  console.log(req.query.link)
  var dataTosend
  const python = await spawn('python3', ['scanner.py', req.query.link.toString()])
  python.stdout.on('data', async data=> {
    dataTosend = await data.toString()
    console.log(dataToSend)
  });
  python.stderr.on('data', data => {
    console.log('None type returned from the object')
  });
  python.on('close', (code) => {
    console.log(`child process close all stdio with code ${code}`);
    res.send(dataTosend)
  });
})
app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

- **Index.js**

- **Extension of the js file**

VIII.RESULTS

a. **Link-**
http://drra nurek ha .blog spot.com/ 20 1 8/ 07 / sliding
-window-protocol.html

As displayed by the extension, this webp age is vulnerable to Cross-site S cripting attack

b. **Link-** http://testhtml5 .vulnweb.com/ #/popula r
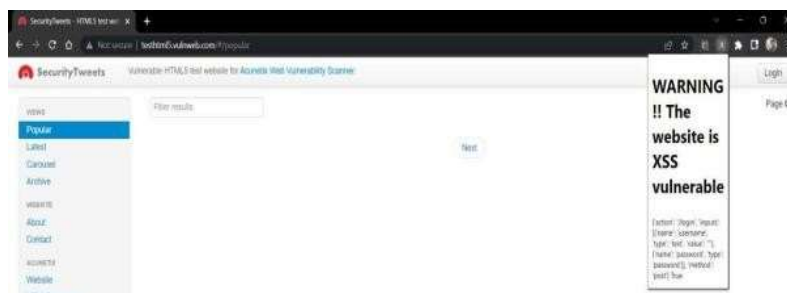
As displayed by the extension, this webp age is vulnerable to Cross-site S cripting attack

c.Link - http://testphp.vulnweb.com/

As warned by the extension, this webpage is vulnerable to XS S attack

d.Link - https://www.youtube.com/

YouTube is a secure w ebsite and hence is not vulnerable to XS S

e. Link - https://xss-game. appspot.com/level1/frame

As warned by the extension, this webpage is XS S vulnerable

REFERENCES

1. GA XSS: Effective Payload Generation Method to Detect XSS Vulnerabilities Based on Genetic Algorithm – https://www.hindawi.com/journals/scn/2022/2031924/
2. Web Vu lnerability Finder(W VF): Automated Black-Box Web Vulnerab ility Scanner-https://www.mecs-pres.org/ijitcs/ijitcs-v12-n4/v12n4-5.html
3. Detecting Cross-Site Scripting in Web
4. Applications Using Fuzzy Inference System–https://www.hindawi.com/journals/jcnc/201 8/8159548/
5. Cross-Site Scripting(XSS) Detection Integrating Evidences in Multiple Stages- https://dblp.org/rec/conf/hicss/hangJL19/ht ml
6. DeepXSS: Cross Site Scripting DetectionBased on Deep Learning –https://dl.acm.org/doi/abs/10.1145/3194452.3194469