**Name: Samarth Jain**

**USN: 4SU20CS081**

**Course: Cybersecurity**

**Trainer: Bharath Kumar**

**Date: 31/08/2023**

**Assignment Details**

Assigned Date: 30/08/2023

Due Date: 31/08/2023

Topic: OWASP Broken Web Applications

**Introduction**

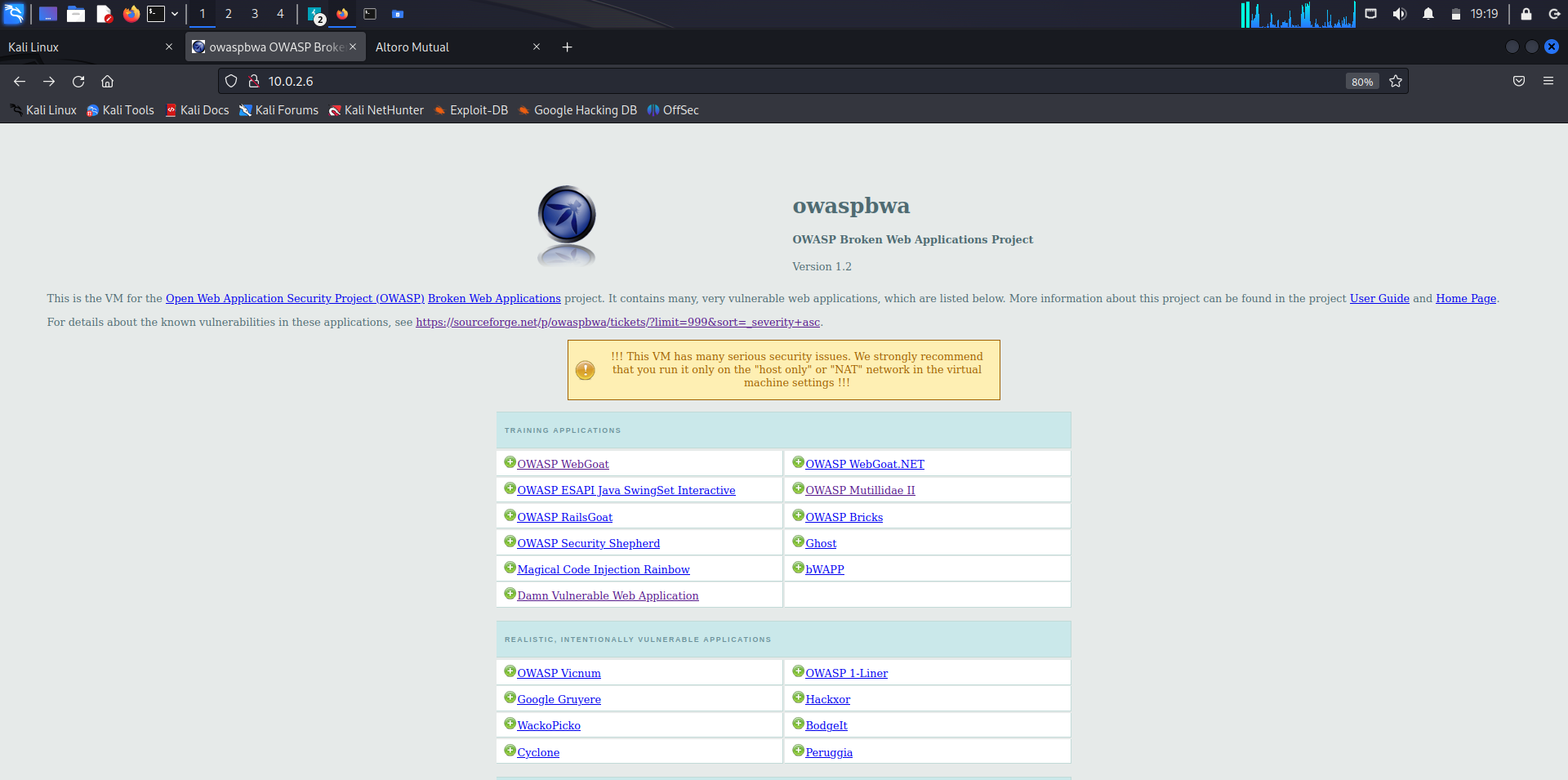
The OWASP Broken Web Applications (OWASP BWA) project is a collection of vulnerable web applications that have been deliberately designed with security flaws. These flaws are intended to serve as educational resources for individuals interested in learning about web application security. The project provides various vulnerable web applications written in different programming languages, frameworks, and technologies, allowing developers, security professionals, and researchers to practice identifying and mitigating security vulnerabilities in a controlled environment.

The main purpose of the OWASP Broken Web Applications project is to promote awareness of common web application vulnerabilities and to provide a hands-on learning experience for individuals looking to improve their skills in secure coding, penetration testing, and security assessments. By exploring and experimenting with these deliberately vulnerable applications, users can gain practical insights into real-world security issues and how to address them effectively.

The OWASP BWA project is part of the Open Web Application Security Project (OWASP), a nonprofit organization focused on improving the security of software. OWASP provides a wealth of resources, tools, and best practices for web application security, and the Broken Web Applications project is just one of the many initiatives they undertake to fulfil their mission.

**Content**

**Open Web Application Security Project (OWASP) Broken Web Application**

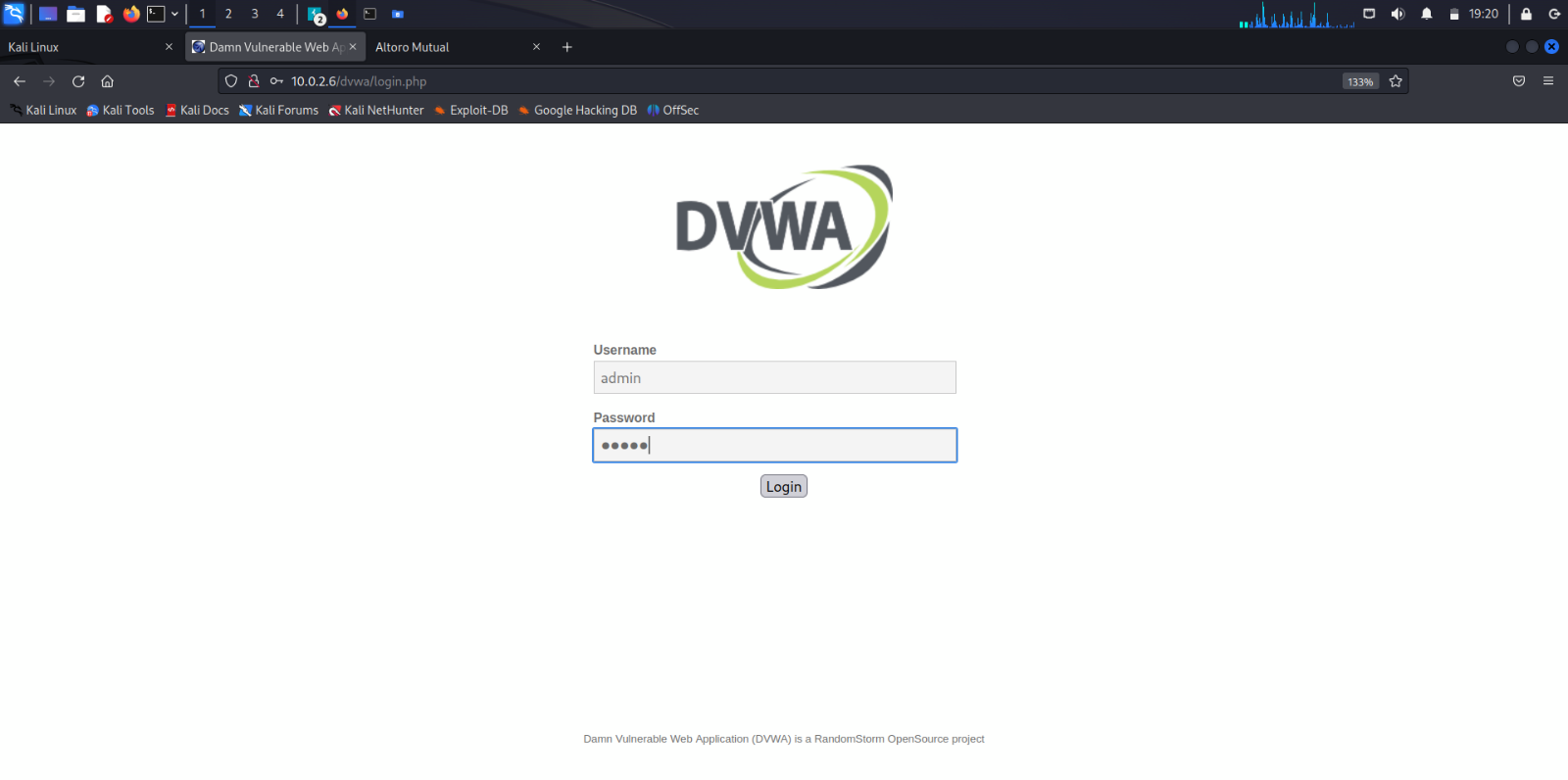


**Damn Vulnerable Web Application (DVWA)**

Login Page

Username: admin

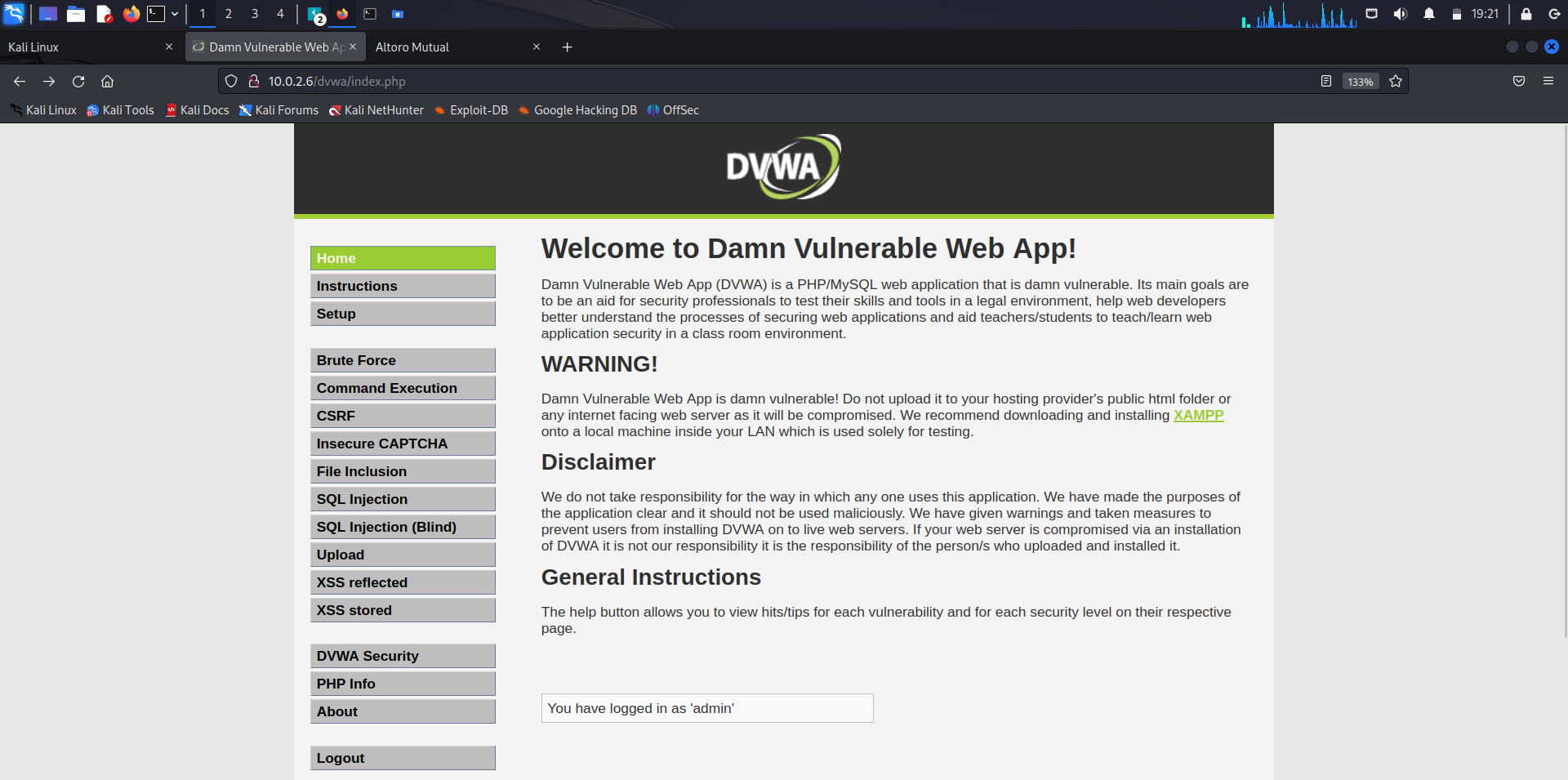
Password: admin



DVWA Home Page

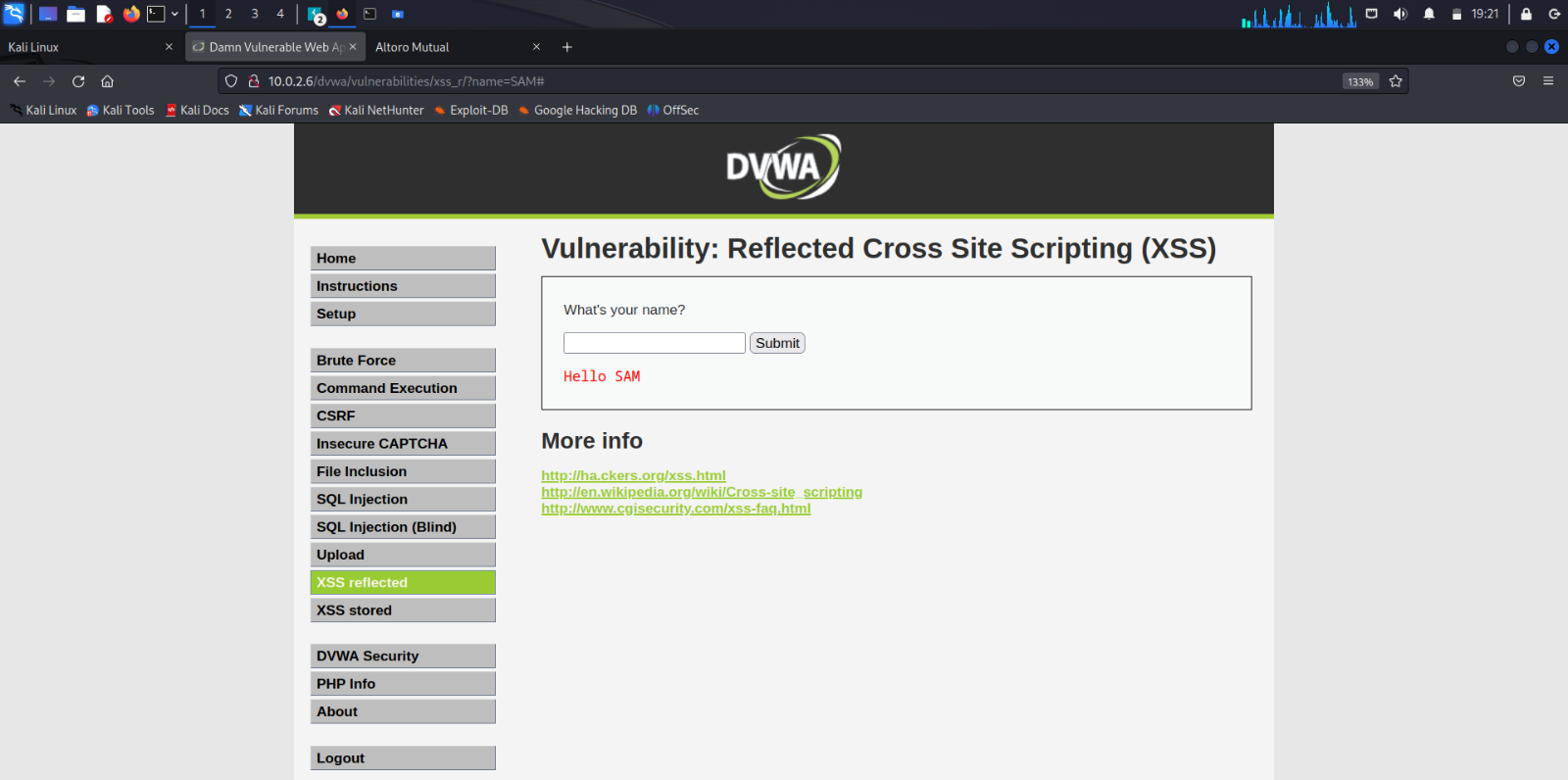
This package contains a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface. Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.



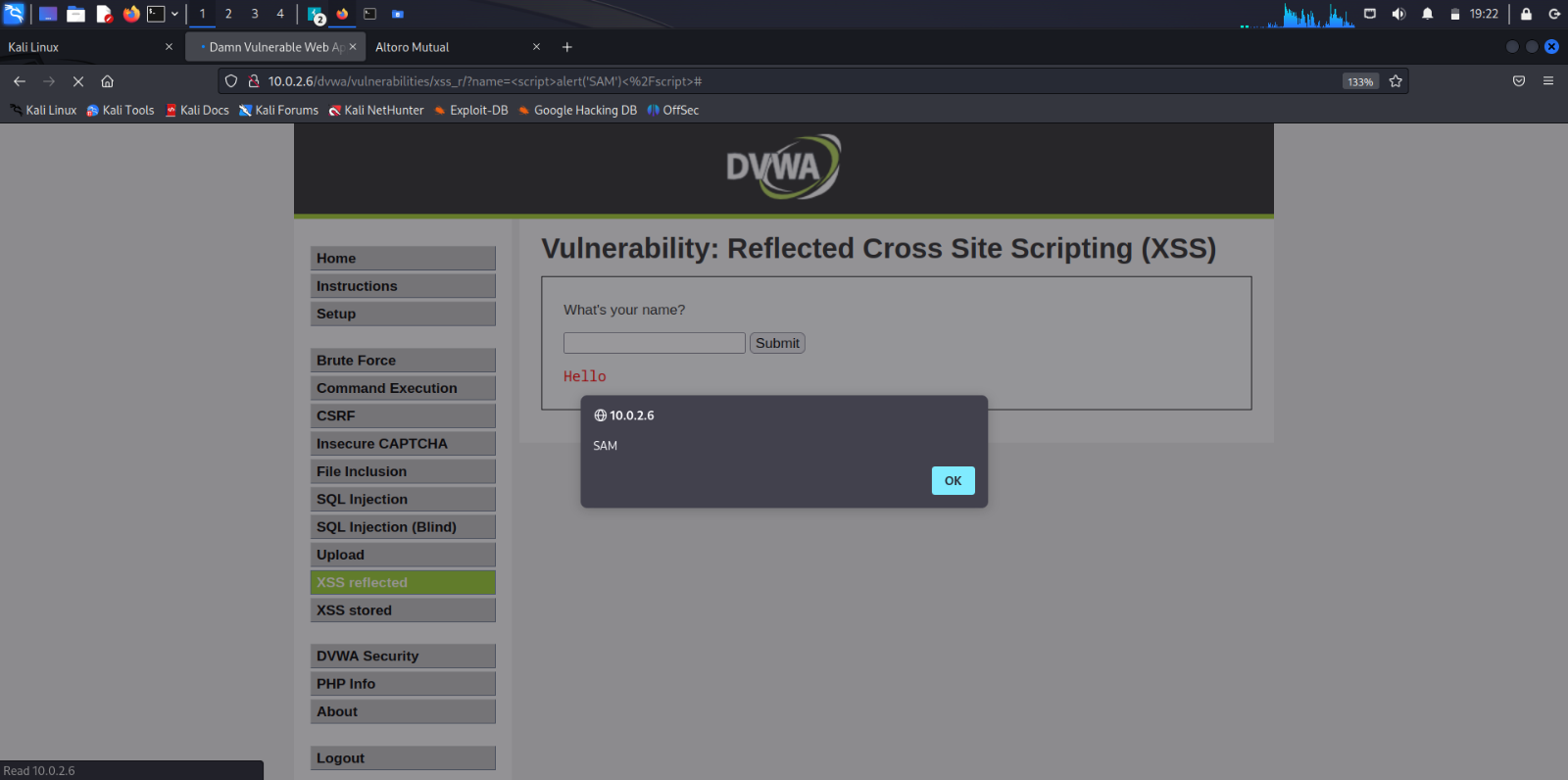
**XSS (Cross Site Scripting) Reflected**

Input: SAM



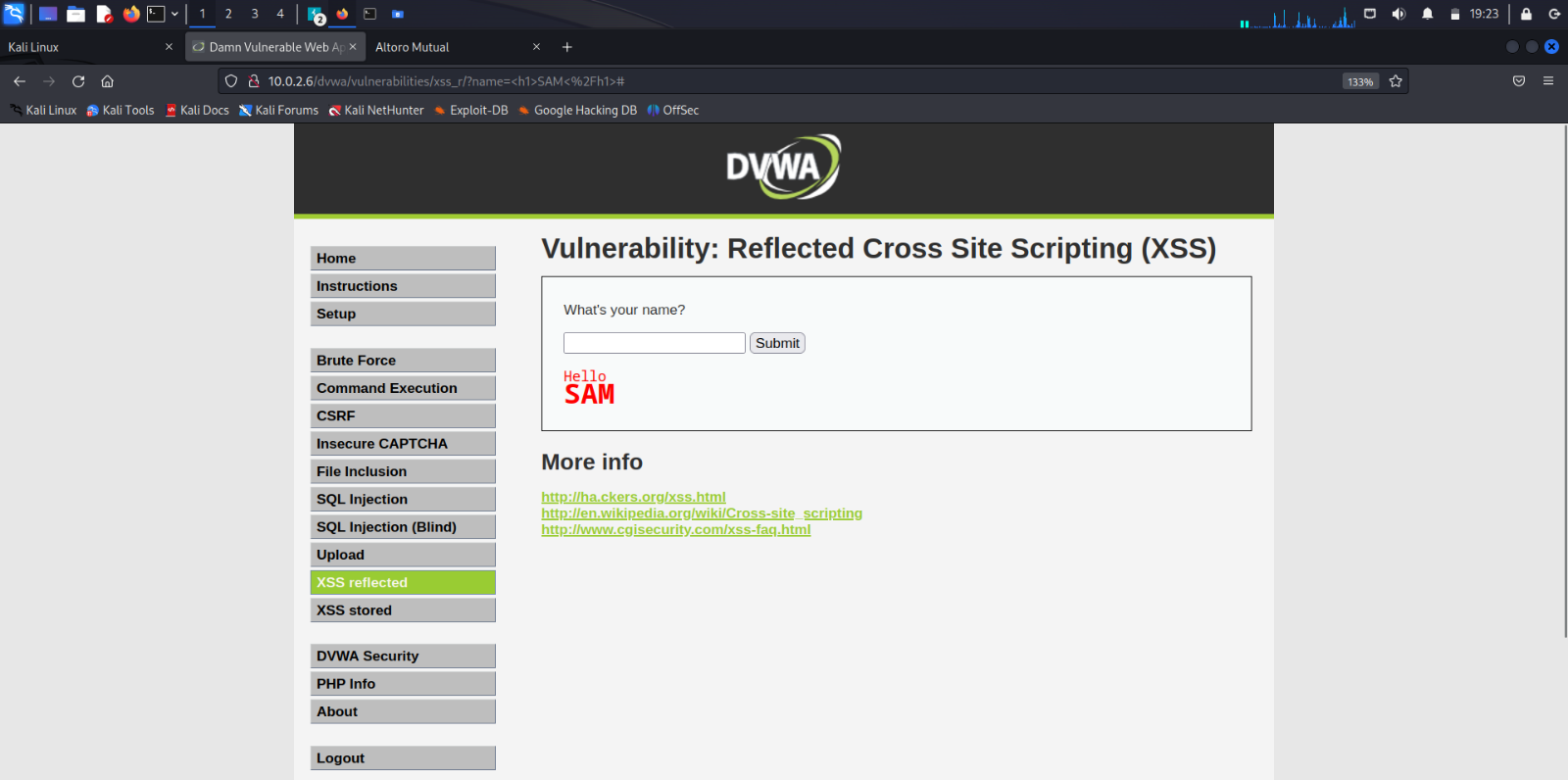
Input: <script>alert('SAM')</script>

//Javascript code to alert the text ‘SAM’



Input: <h1>SAM</h1>

//HTML code to return the text ‘SAM’ in heading level 1

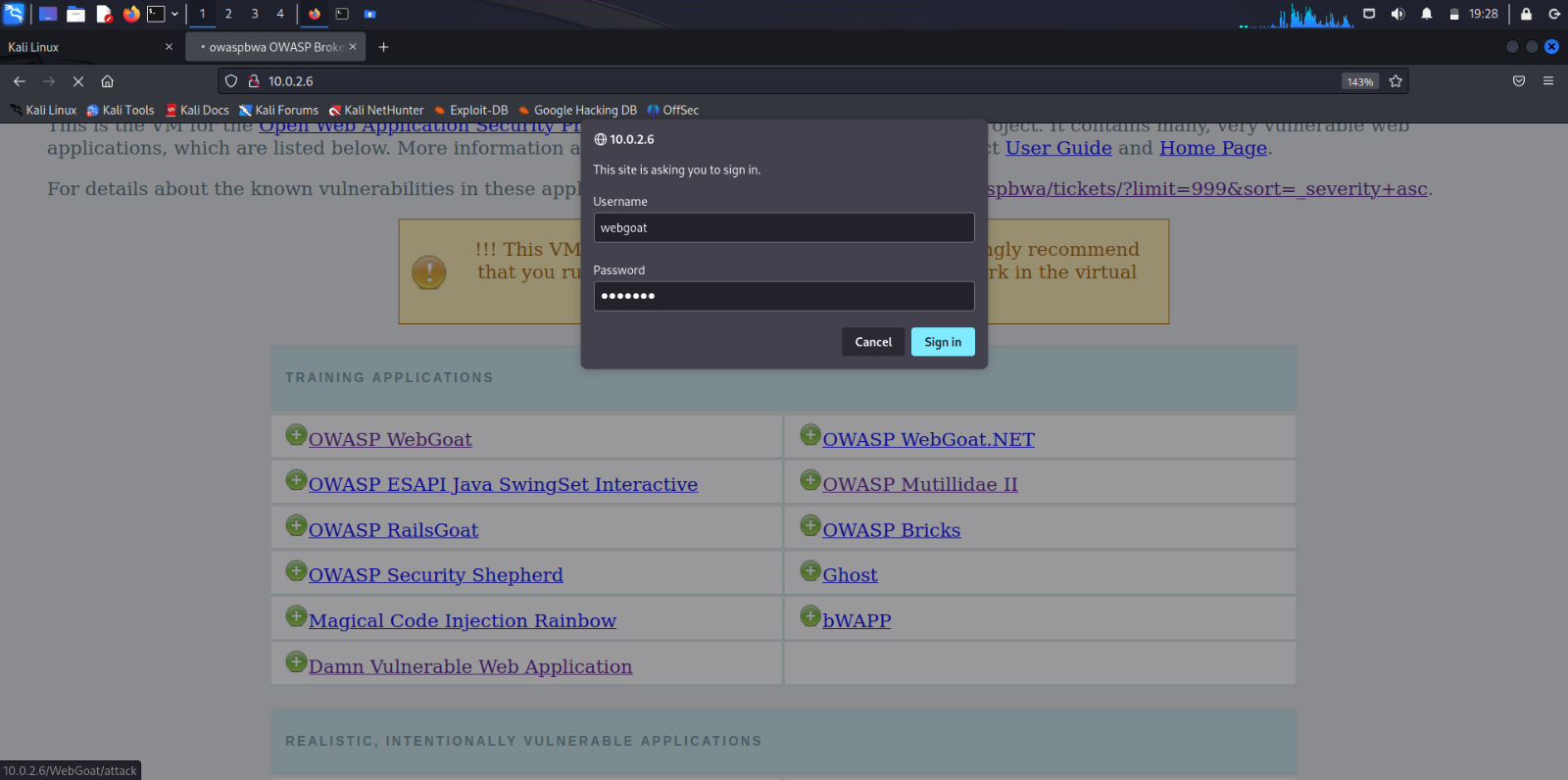


**OWASP WebGoat**

Login Page

Username: webgoat

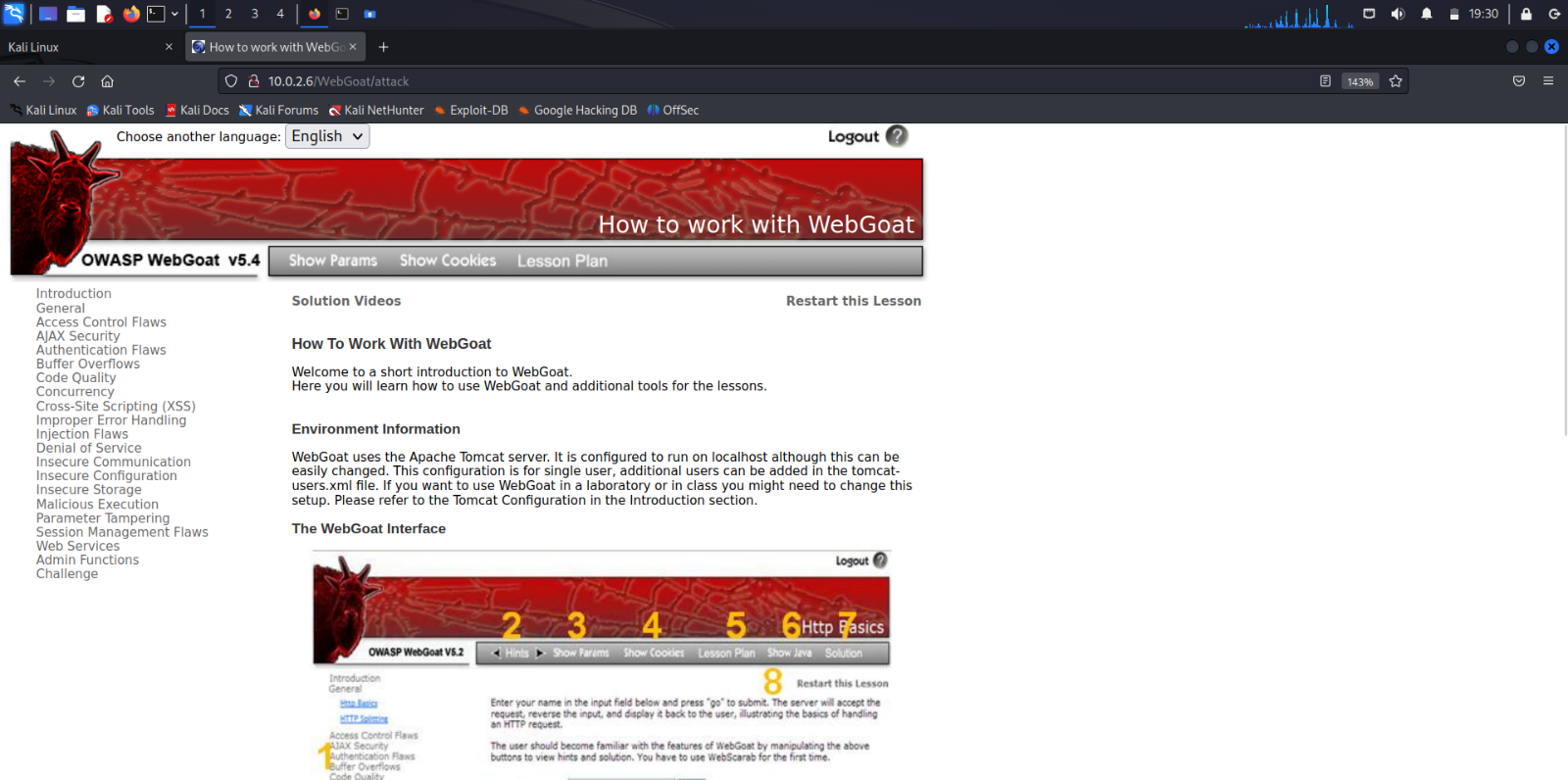
Password: webgoat



Home Page

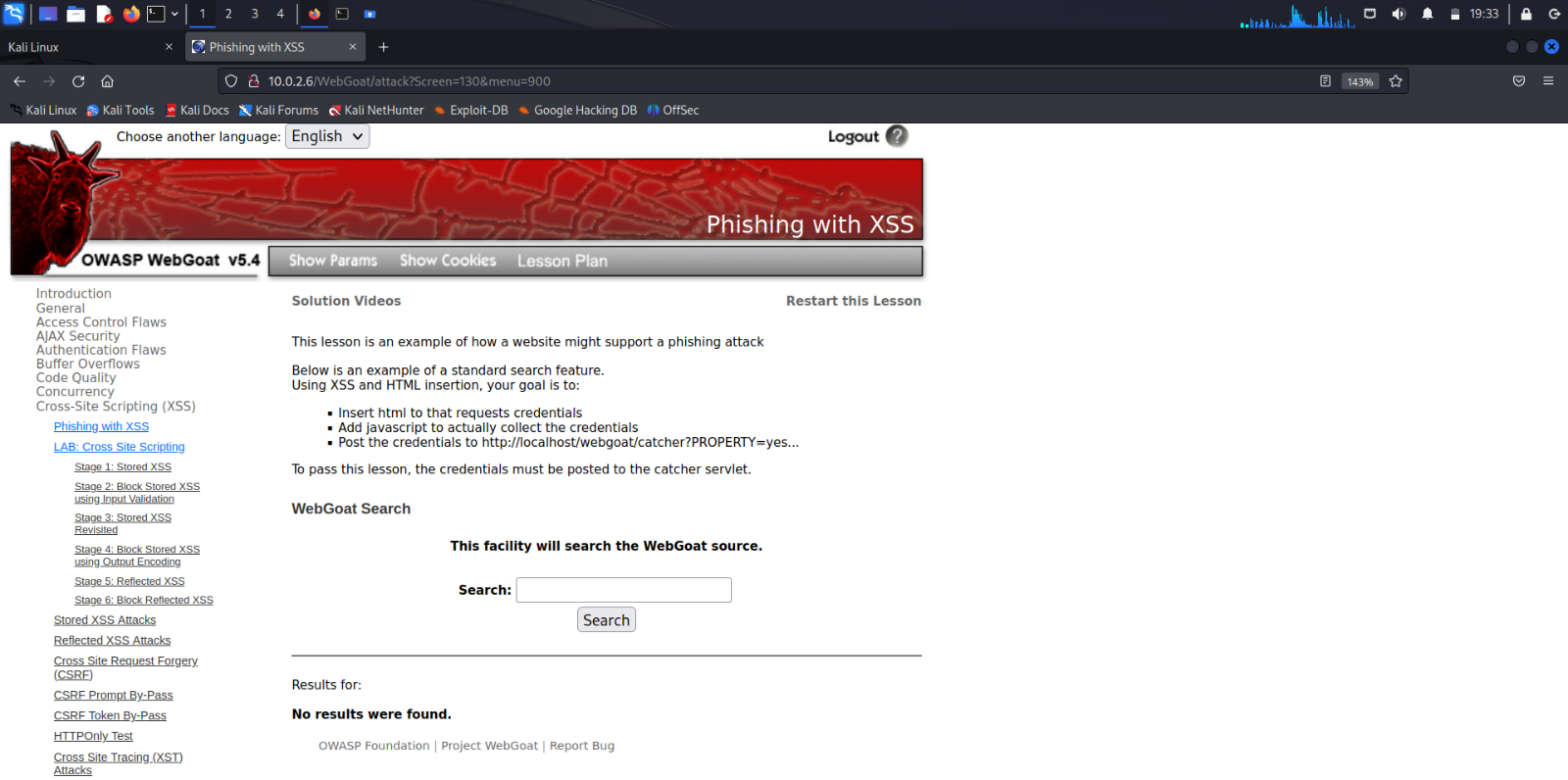
WebGoat is a deliberately insecure application that allows interested developers just like you to test vulnerabilities commonly found in Java-based applications that use common and popular open source components.

WebGoat provides a safe environment to practice exploiting security flaws commonly found in web applications. It covers a wide range of vulnerabilities, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), insecure authentication mechanisms, and more.



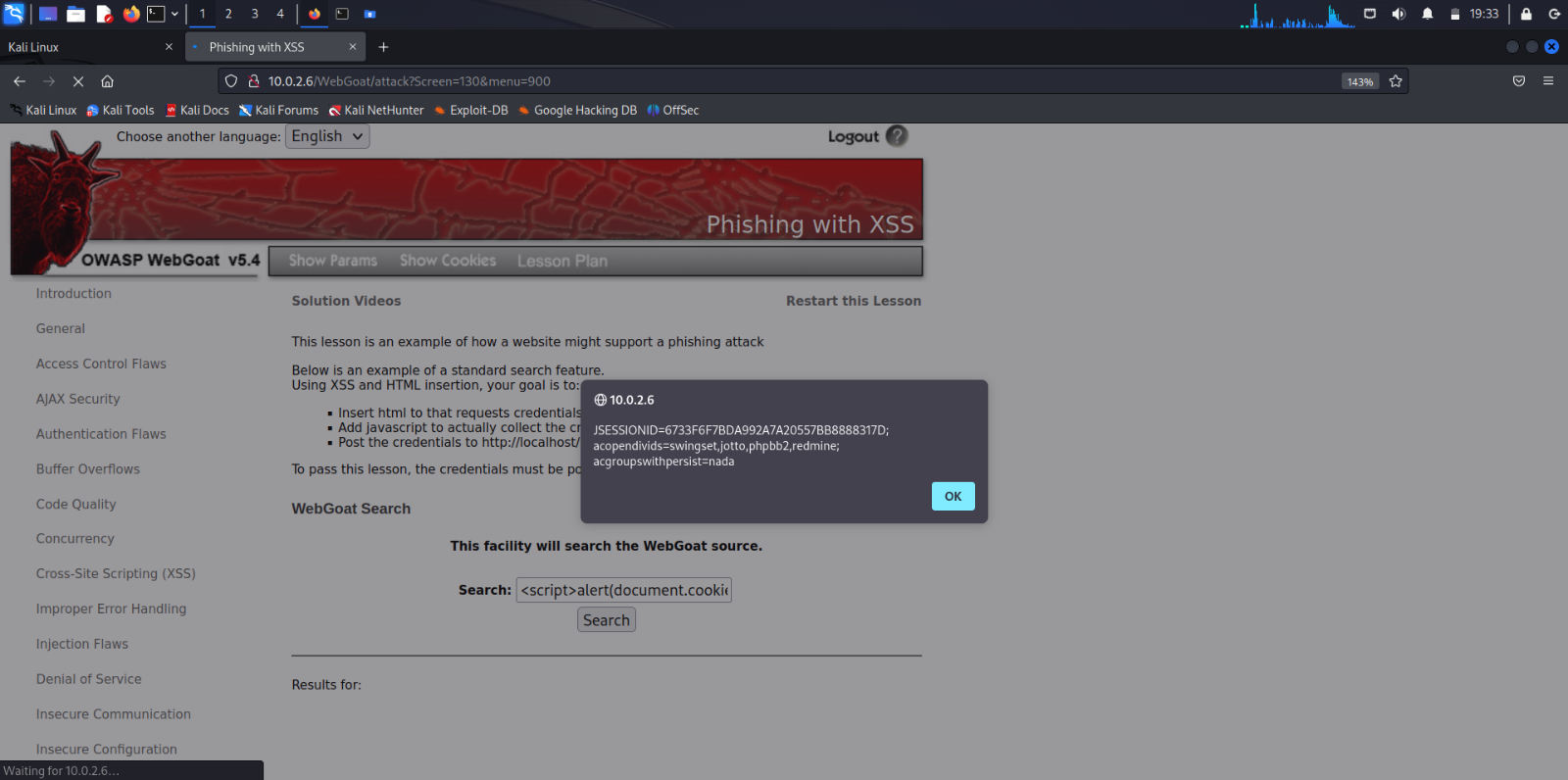
**Cross-Site Scripting (XSS)**

**Phishing with XSS**



Input: <script>alert(document.cookie)</script>

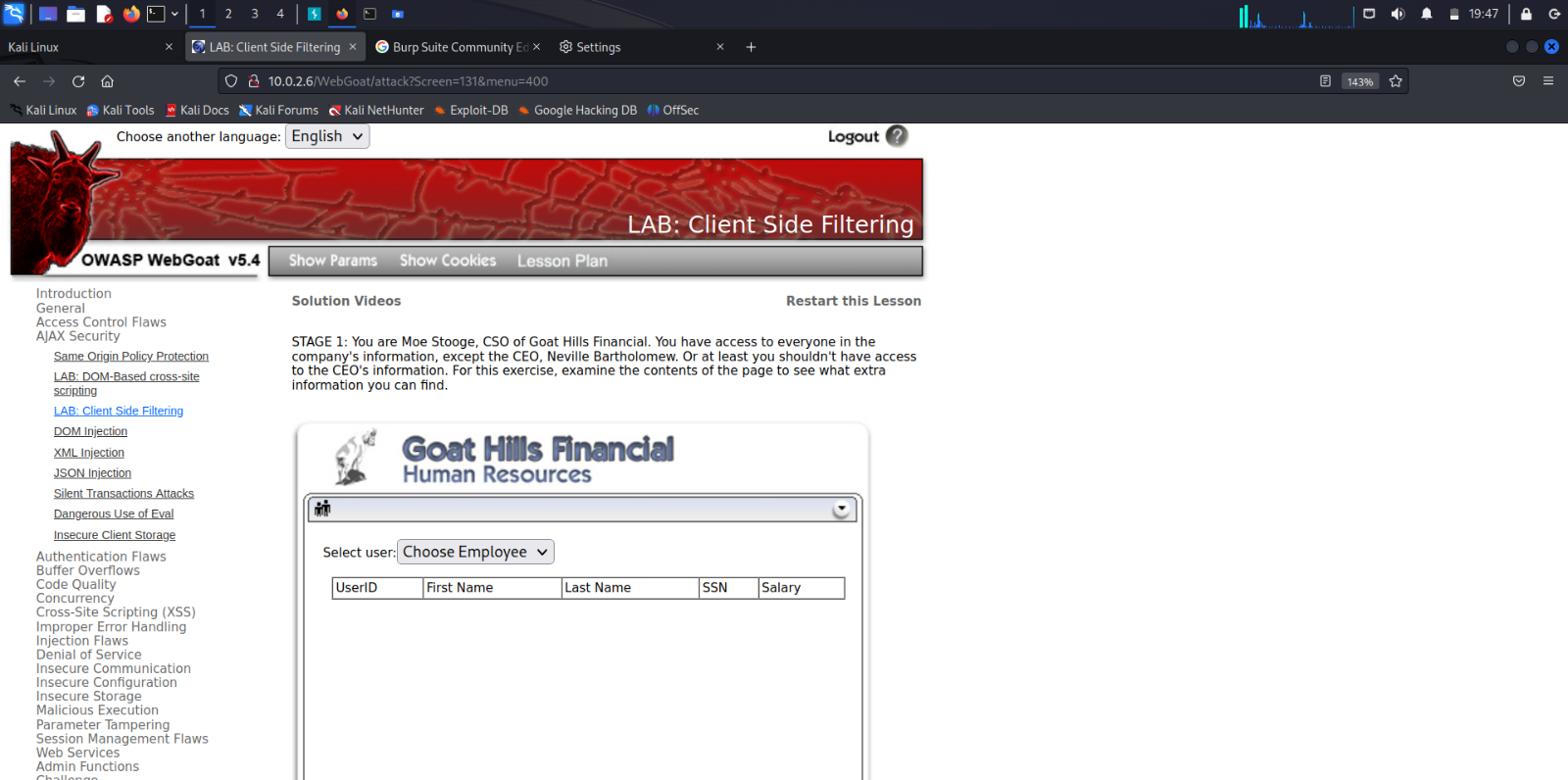
//Javascript code to alert the session cookie



**AJAX Security**

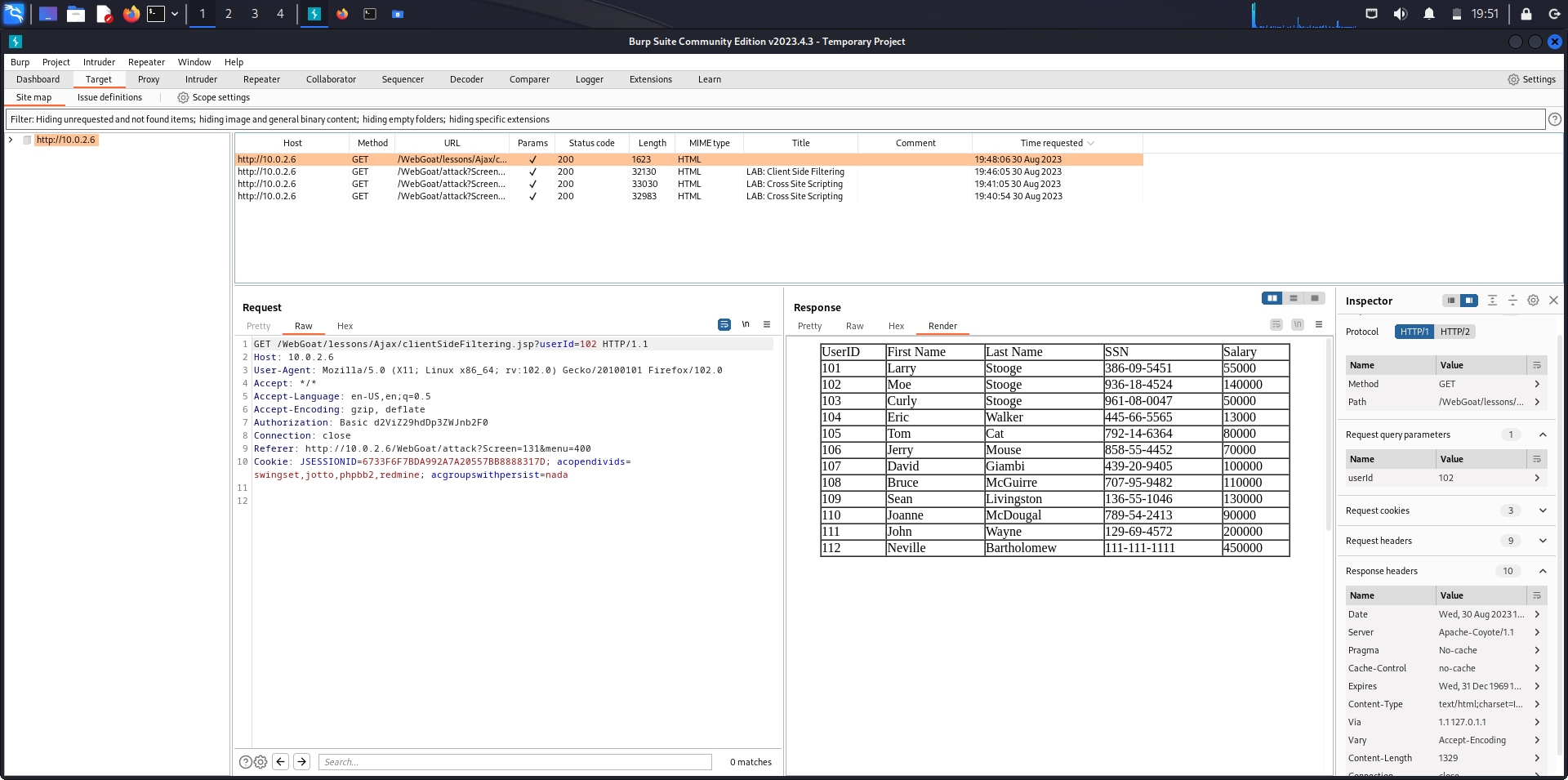
**LAB: Client-Side Filtering**

Problem Statement: Find the salary of the CEO, Neville Bartholomew

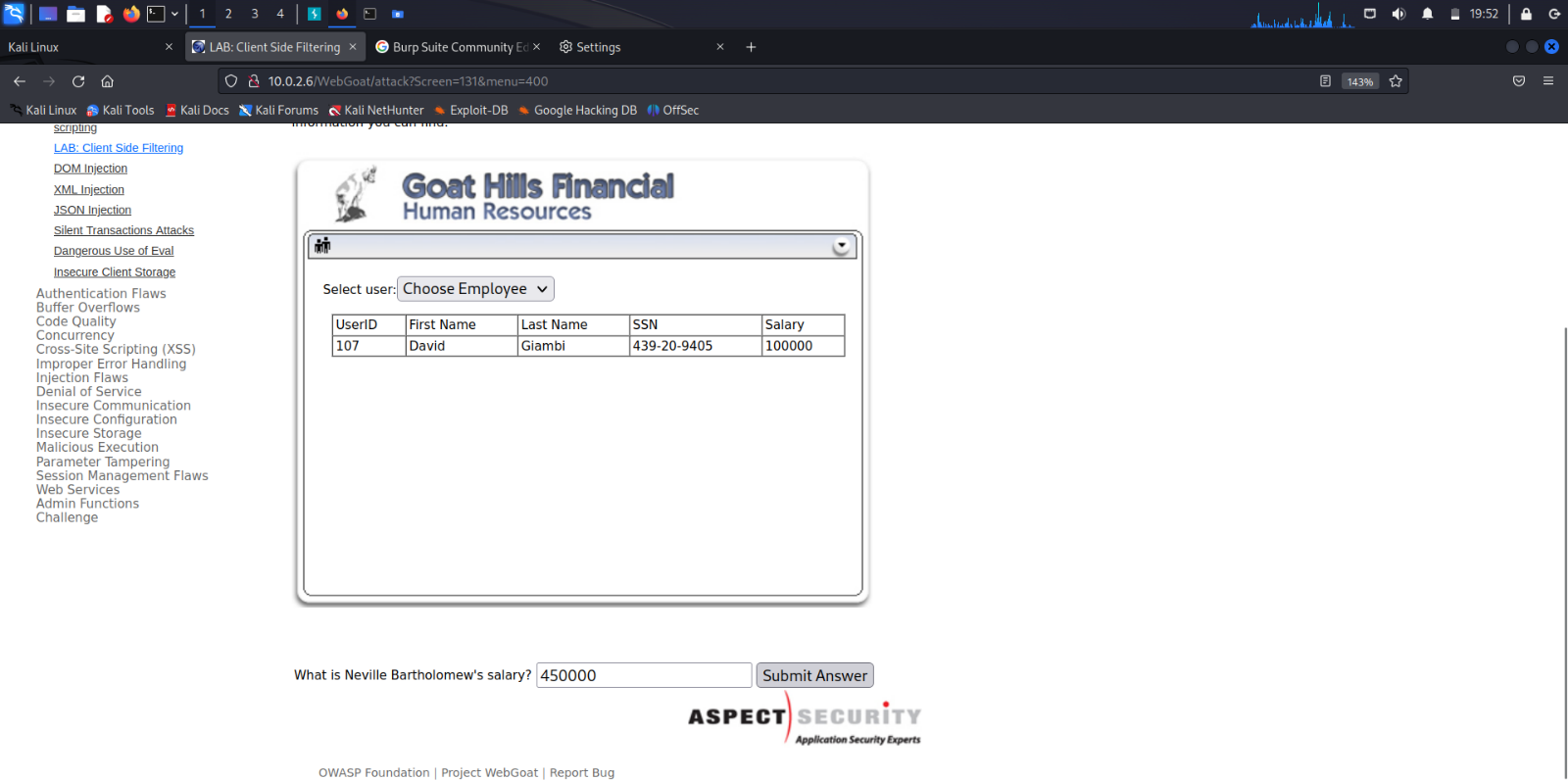


Solution: This information has been filtered due to a process called Client-Side Filtering. Client-side filtering refers to the practice of filtering and processing data directly within the user's web browser or client application, rather than on the server side. In this approach, the logic for filtering and displaying data is implemented using scripting languages like JavaScript within the context of the user's device. Client-side filtering is typically implemented to improve user experience and responsiveness, security professionals often need to bypass or manipulate such filters to test the effectiveness of security controls.

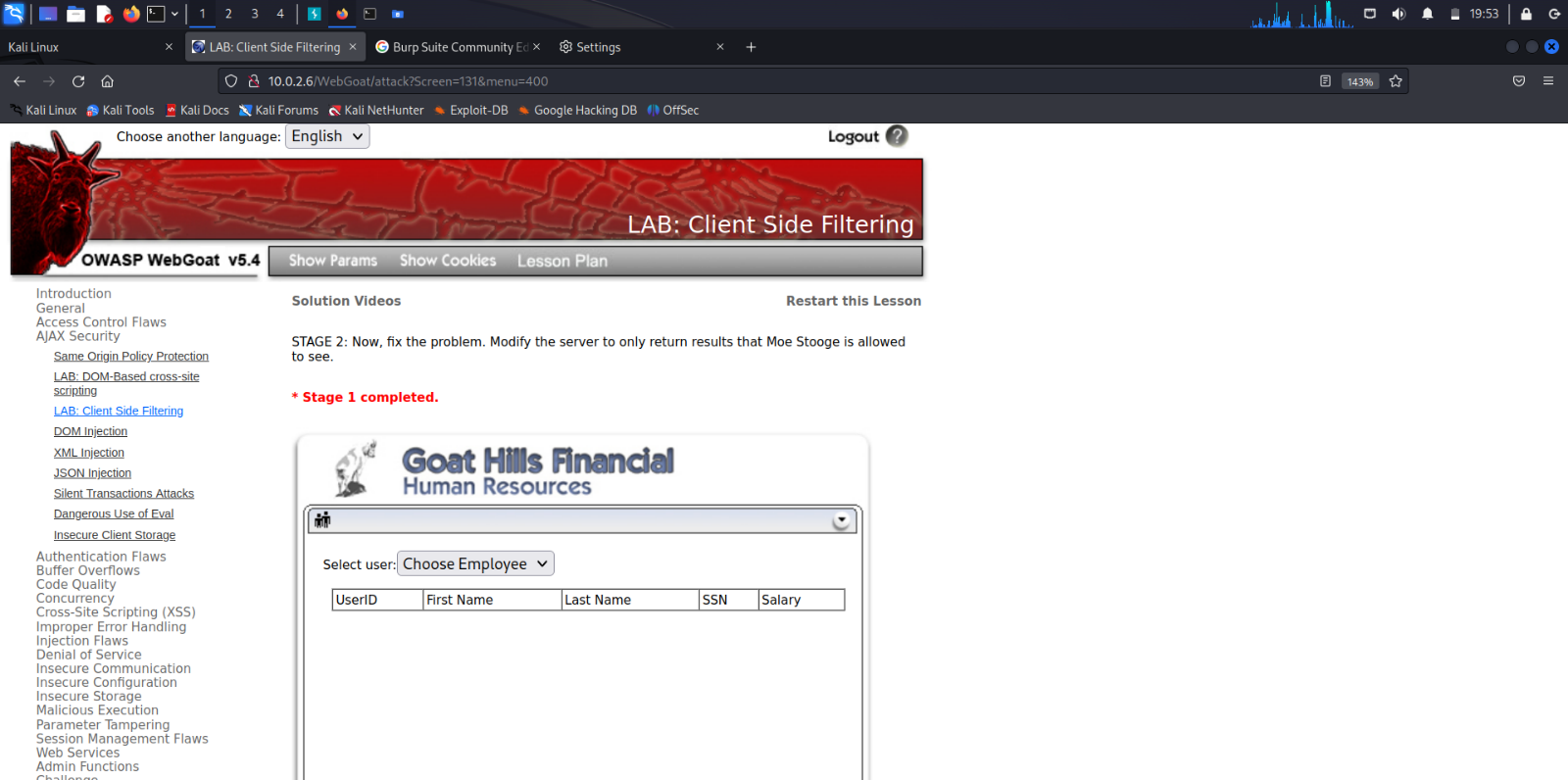
When we render the response in burpsuite, we can see the information about Neville Bartholomew and his salary without any filtering because Burpsuite acts as an intermediary between Server and Client.



The answer to the problem is to be submitted in the textbox below.

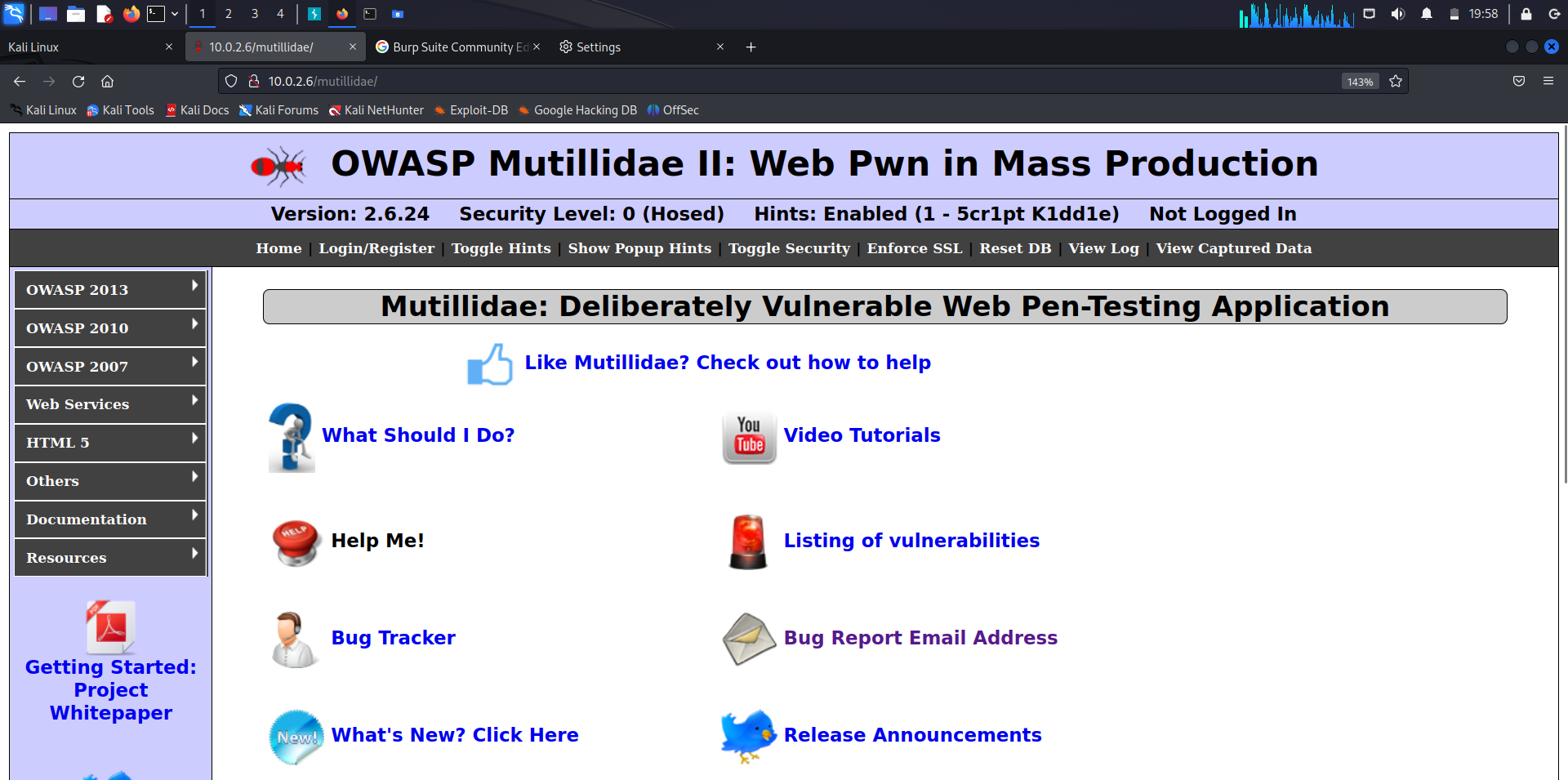


Stage 1 is completed



**OWASP Mutillidae II**

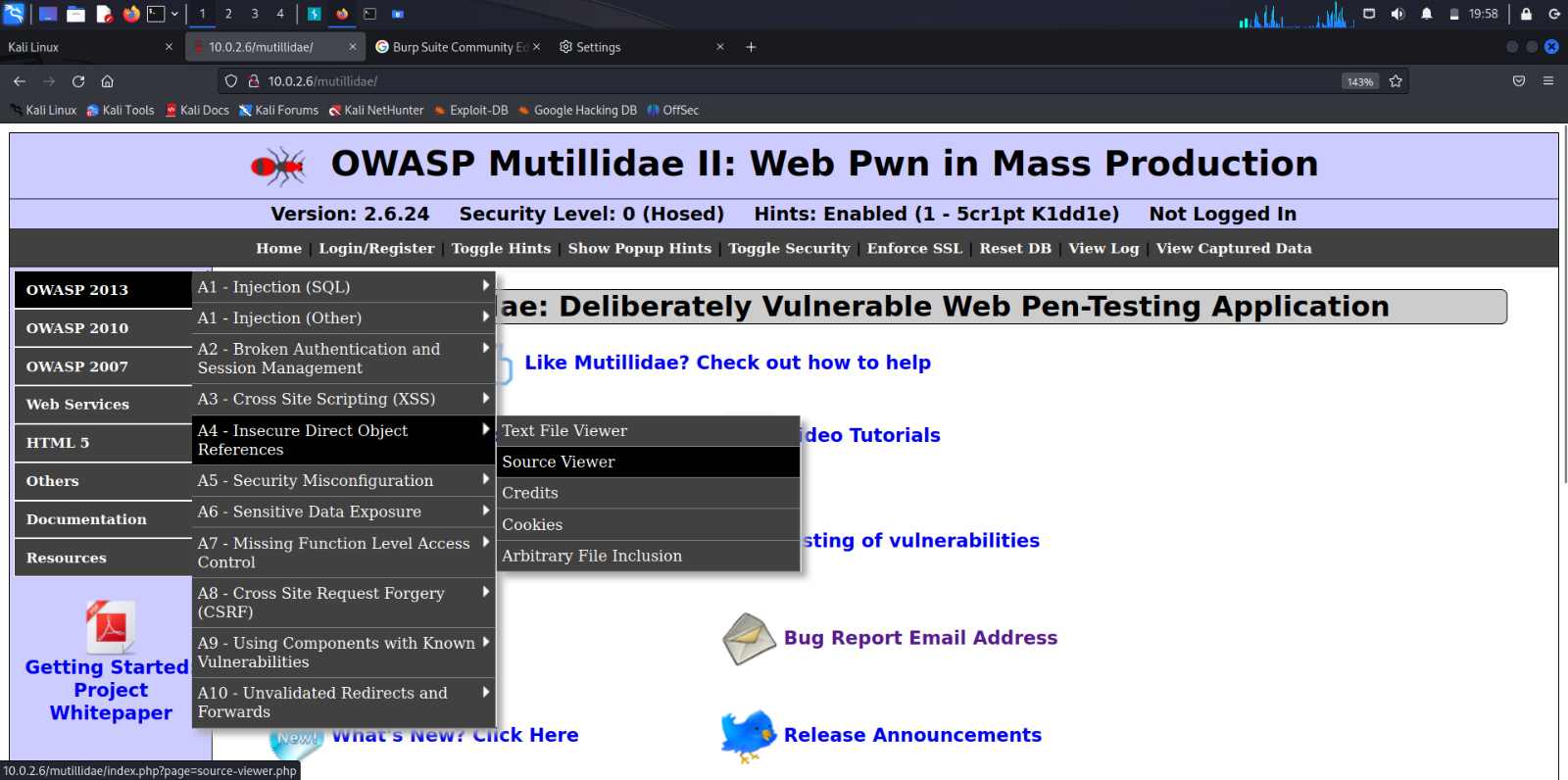
OWASP Mutillidae 2 is a free, open-source web application that is deliberately vulnerable and designed for educational purposes to teach and demonstrate various web security vulnerabilities and attack techniques. The name "Mutillidae" is derived from the family name of ants, and it's meant to symbolize the different vulnerabilities that can exist in web applications, much like the diversity of ants in nature.



**OWASP 2013**

**A4 – Insecure Direct Object References**

**Source Viewer**

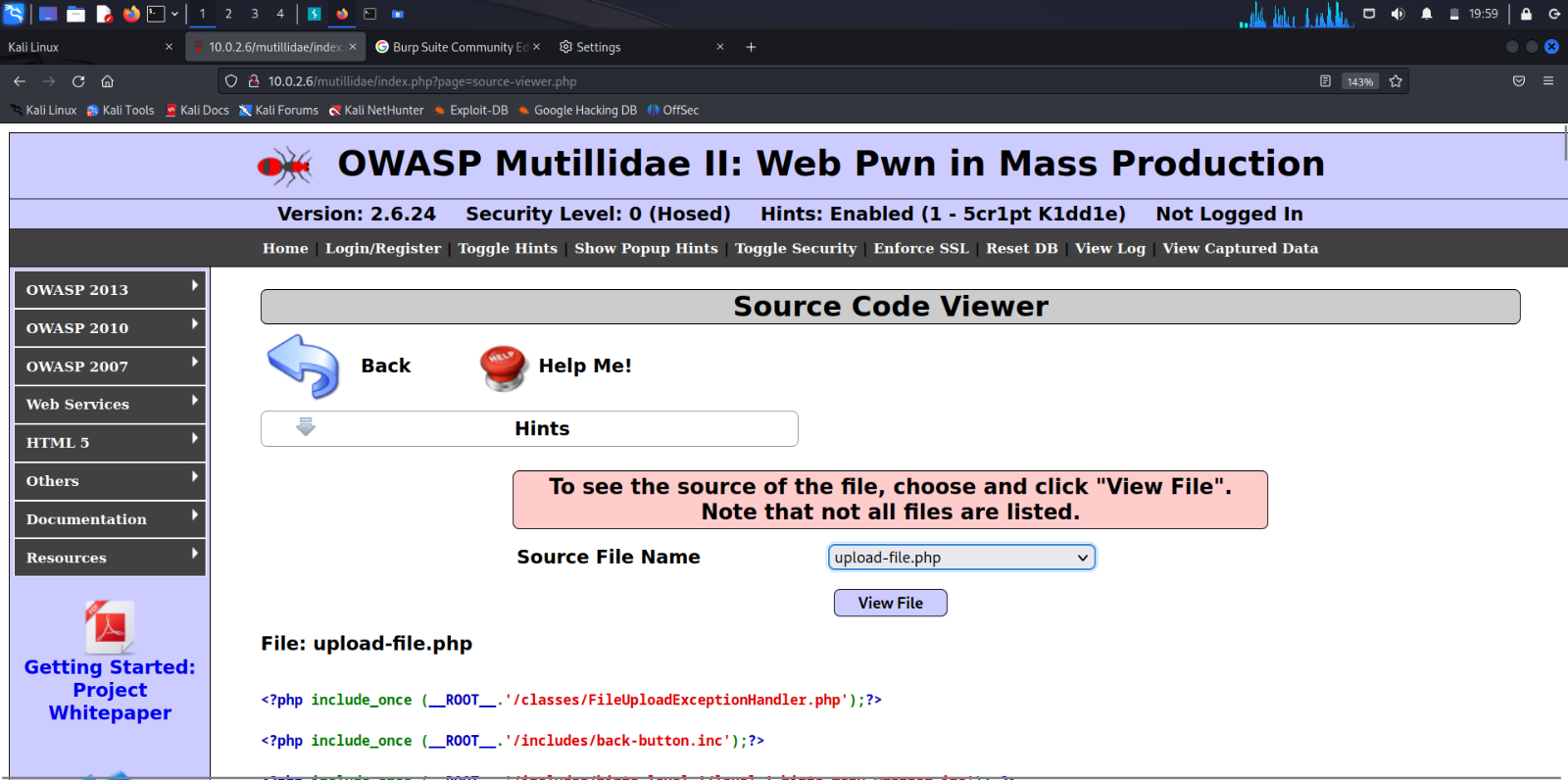


**File Inclusion** is a type of vulnerability in web applications where an attacker is able to include files from the server into the web page being viewed by a user. This can lead to unauthorized access to sensitive files, execution of malicious code, and other security risks. There are two main variations of file inclusion: Local File Inclusion (LFI) and Remote File Inclusion (RFI).

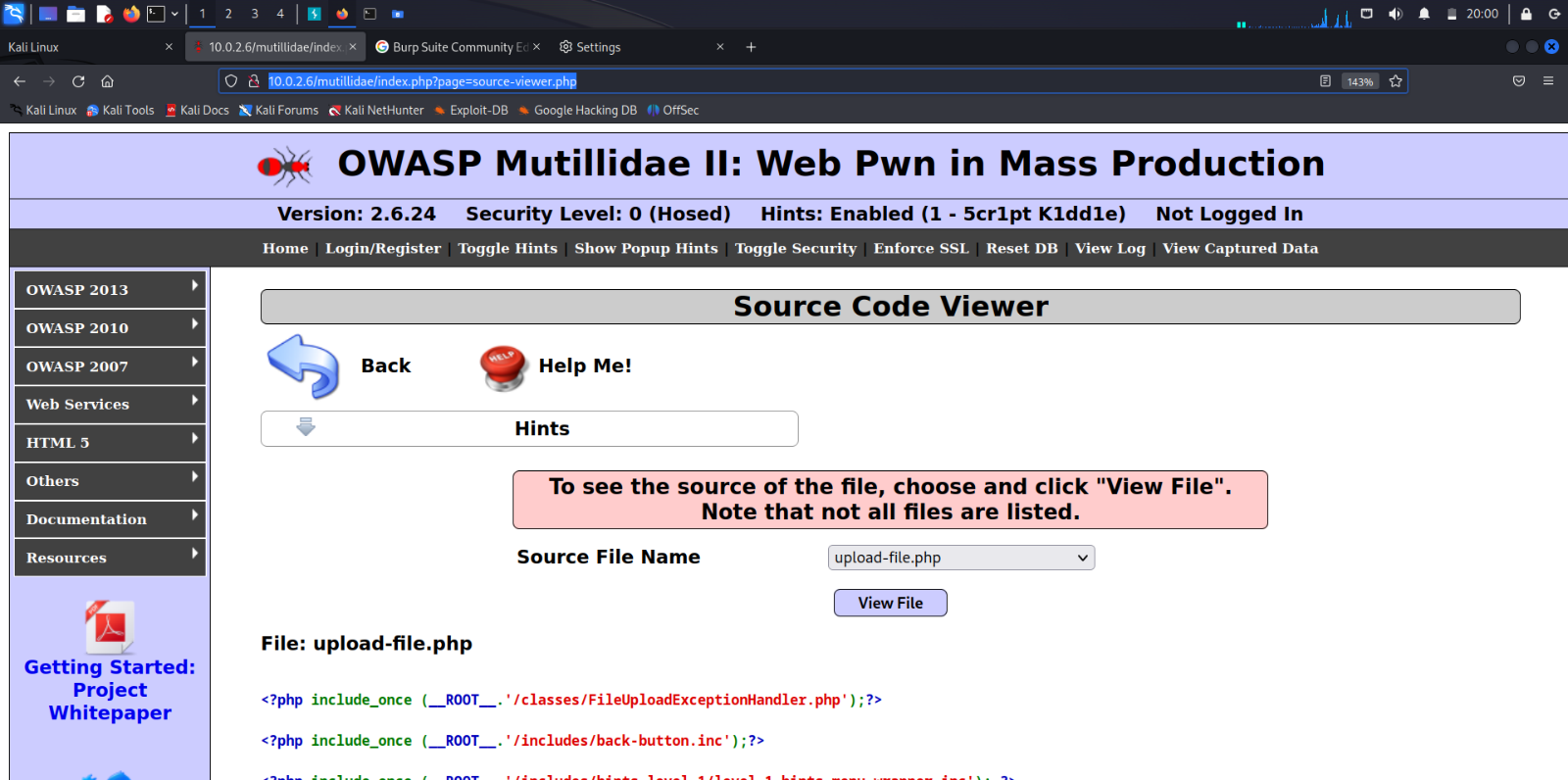
**Local File Inclusion** occurs when an attacker is able to include files that are present on the same server where the web application is hosted. The attacker manipulates input fields or parameters in the application to point to local files on the server.

**Remote File Inclusion** occurs when an attacker is able to include files from a remote server into the web application. This is typically achieved by manipulating input parameters to include URLs pointing to external scripts hosted on the attacker's server.

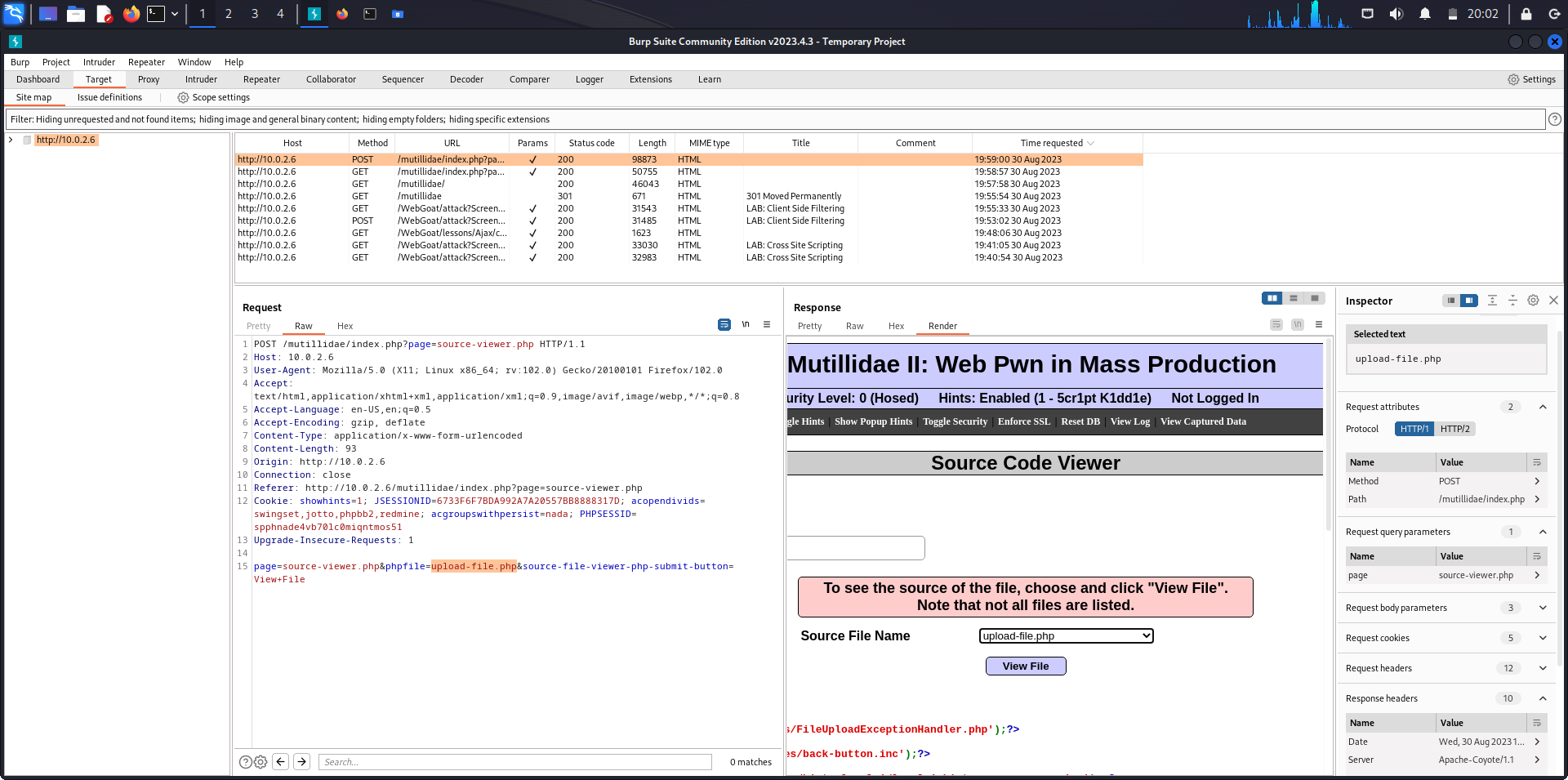
Source Code Viewer



View any file by clicking the ‘View File’ button.



The HTTP request can be seen in the Burpsuite. We can see the file which has been requested (upload-file.php). The Request in the raw form is sent to the Repater by clicking on ‘Send to Repeater’ button in the list.

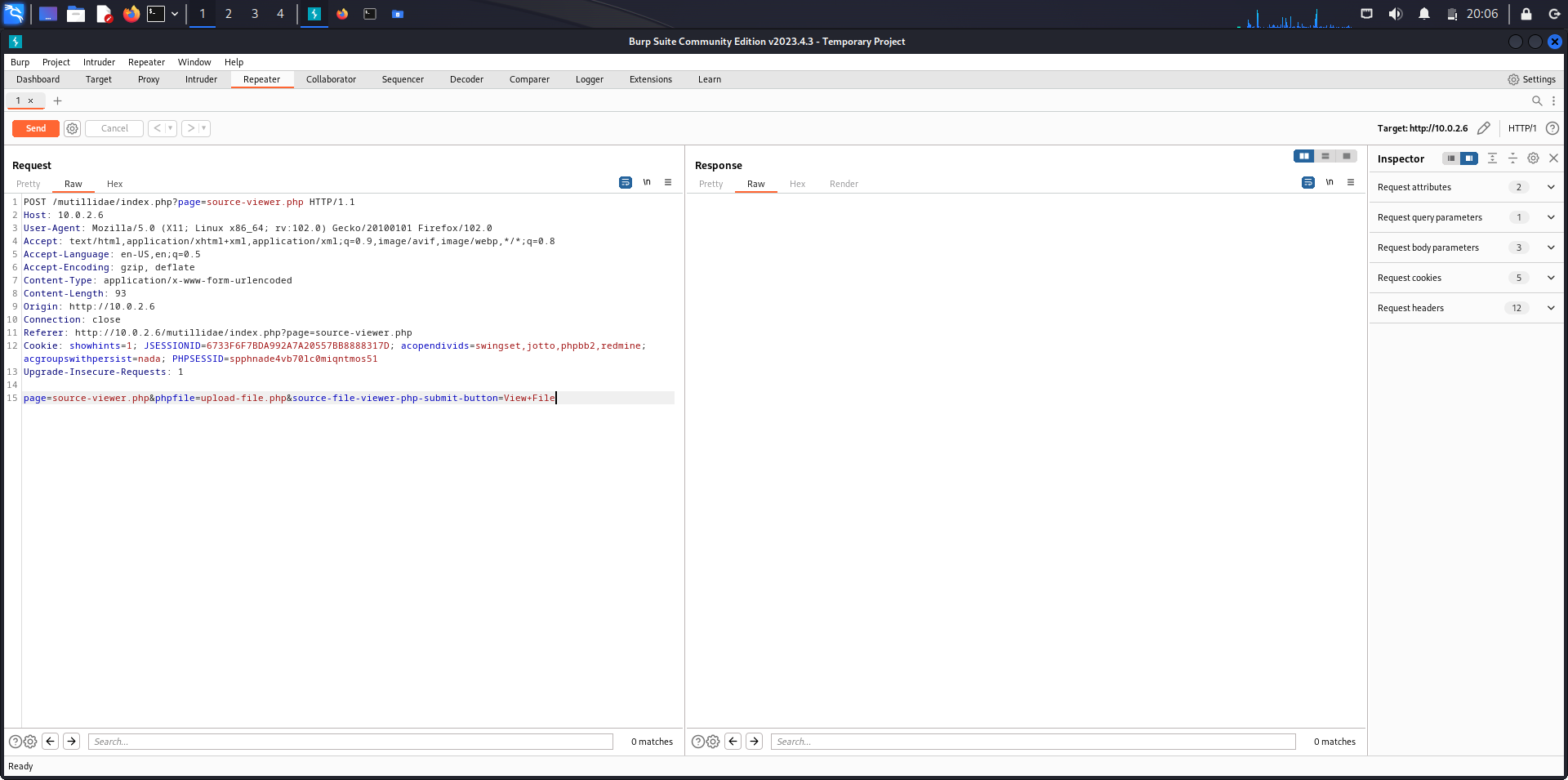


Repeater tab

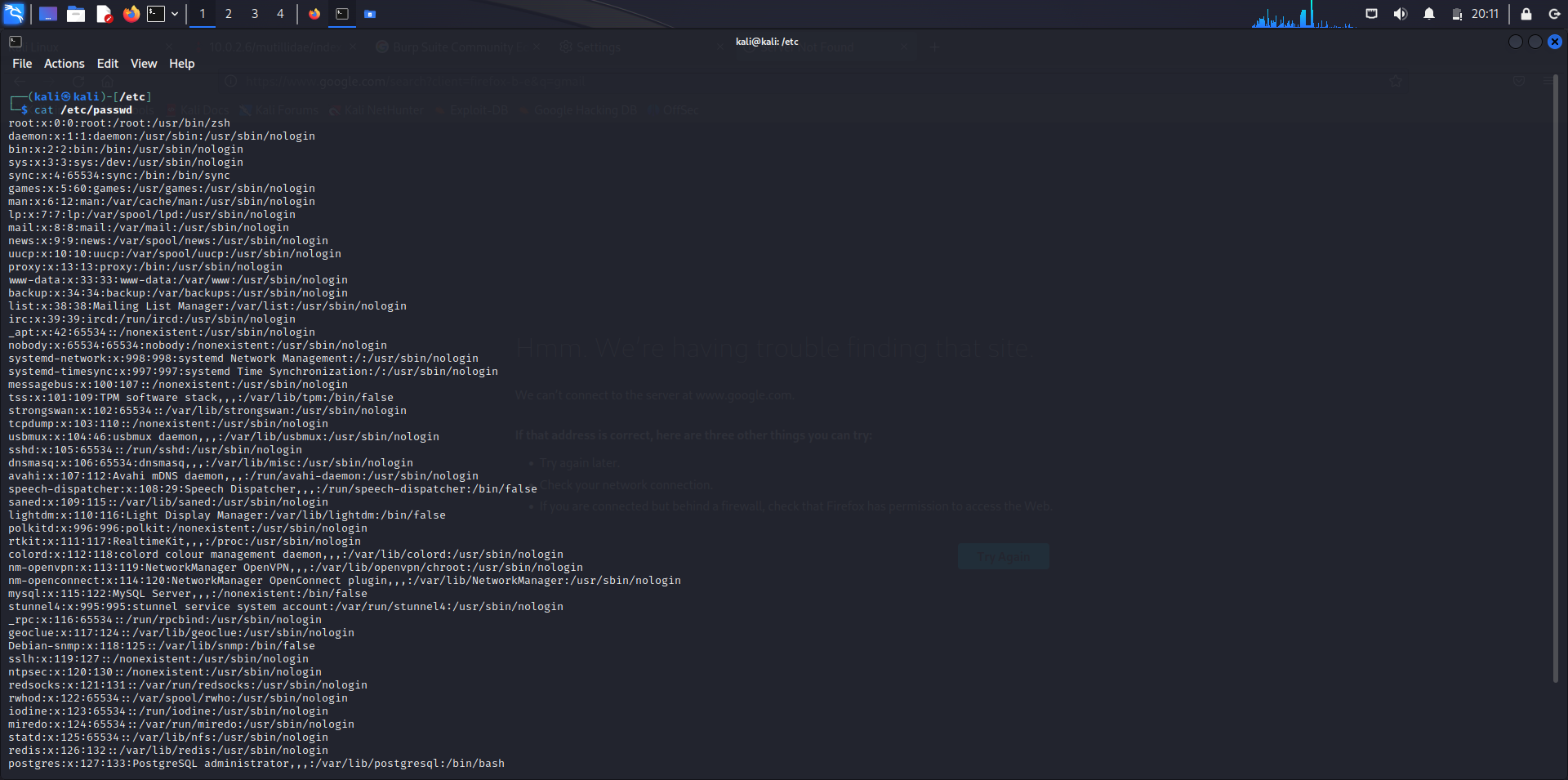
The raw Request is now available in Repeater tab. The Request format can now be manipulated.

The "Repeater" tab is a feature that allows security testers and web developers to manually modify and resend HTTP requests and observe the corresponding responses. Burp Suite is a widely used web vulnerability scanner and security testing tool.

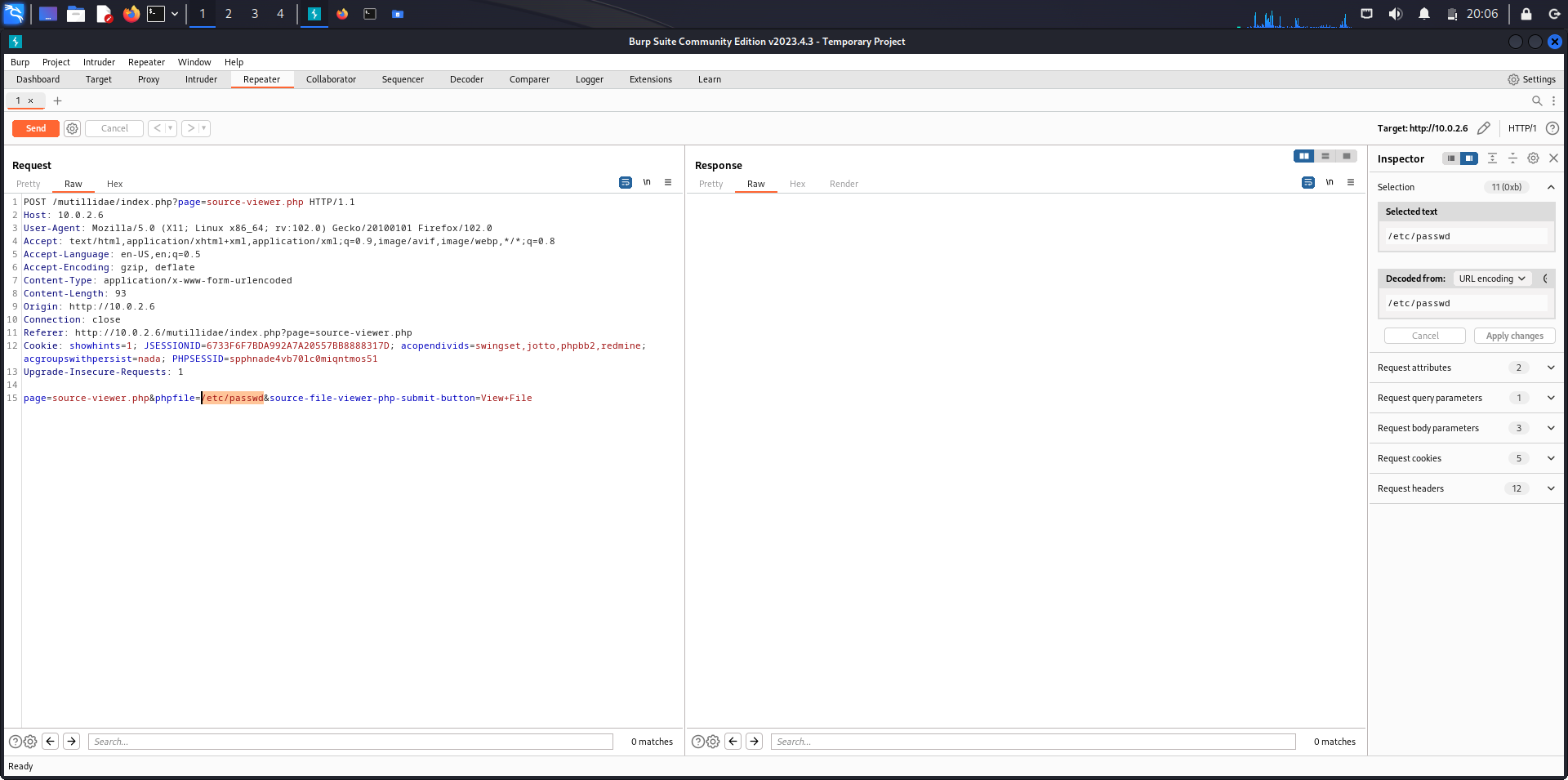
Repeater is also used for various purposes like Request Analysis and Modification, Parameter Manipulation, Session Management Testing, Response Analysis, Brute-force and Fuzzing, Vulnerability Confirmation.



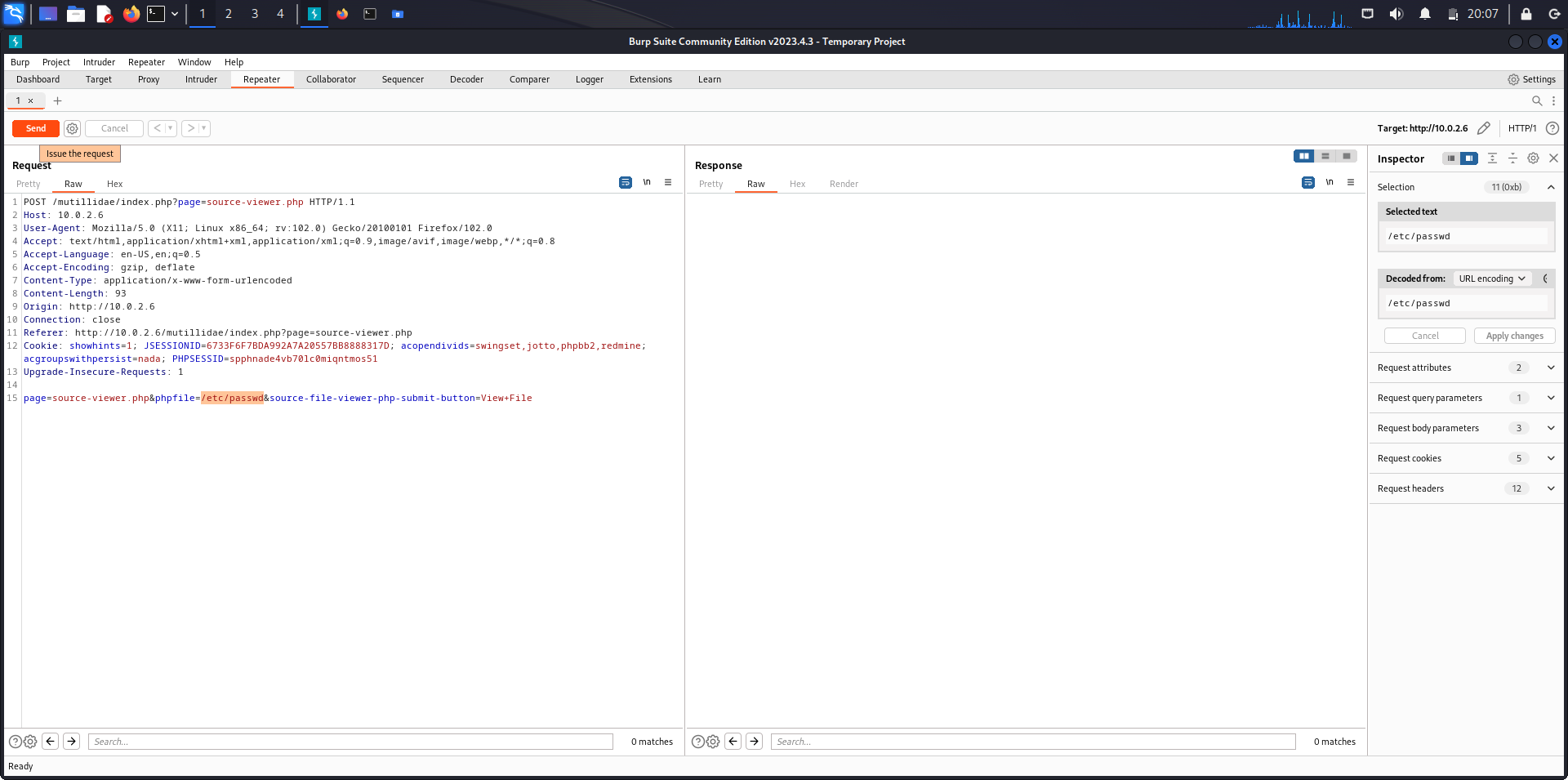
The file to be included is passwd. It is present inside the /etc/ directory.



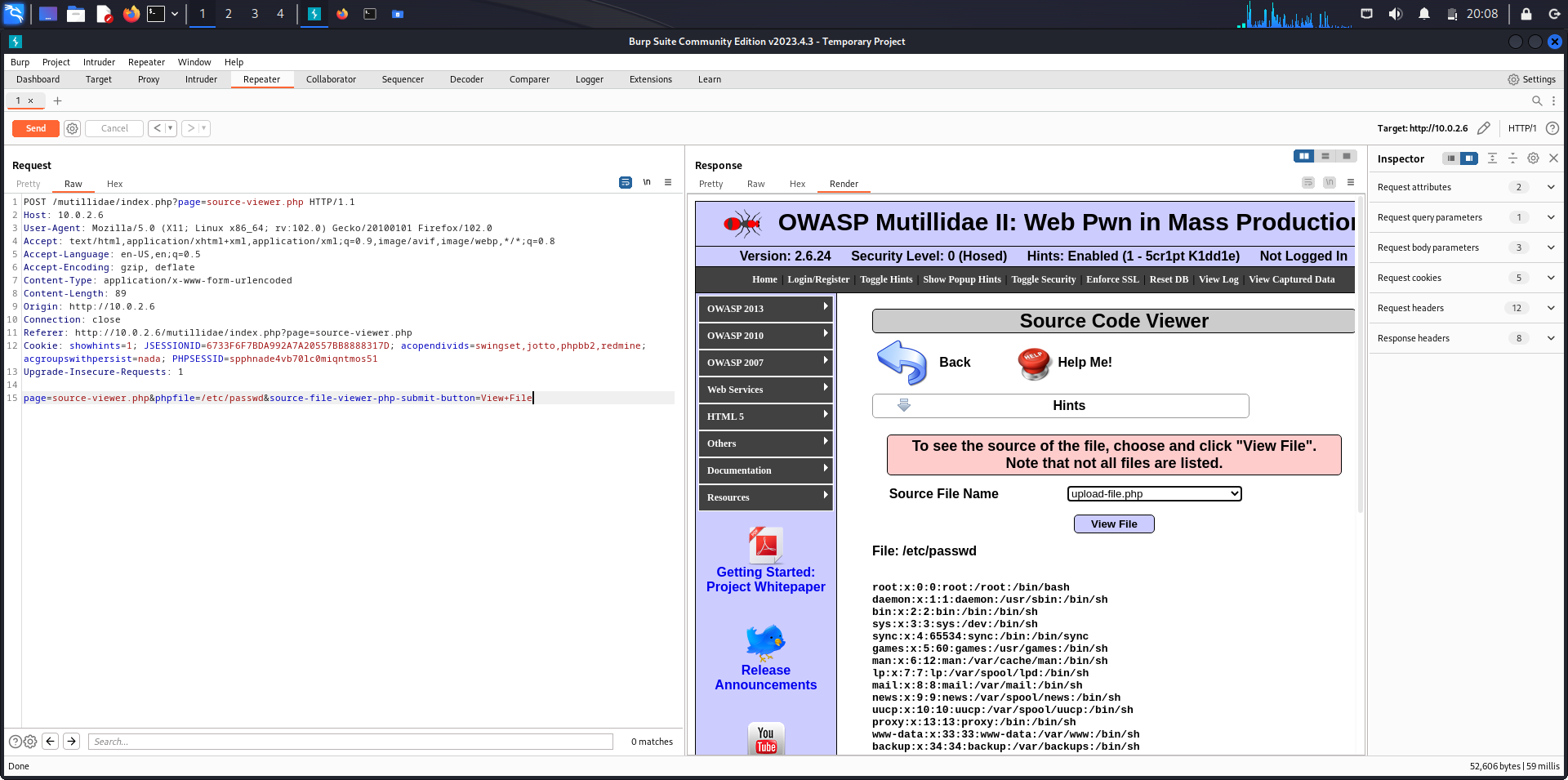
The value of the key phpfile is changed from ‘upload-file.php’ to ‘/etc/passwd’.



To issue the modified request, Send button is clicked.



When the Response page is rendered, we can see that the file has been locally changed. This is called as Local File Inclusion.



**Analysis**

In the context of the OWASP Broken Web Application Project, an analysis was conducted on the project's handling of Cross-Site Scripting (XSS) vulnerabilities. By employing Burp Suite, the project's client-side filtering mechanisms were examined and found to be insufficient in preventing malicious input. The Burp Suite tools facilitated the discovery of a bypass technique that successfully circumvented the filtering, thus highlighting a critical weakness in the application's defence against XSS attacks. Additionally, the investigation revealed a susceptibility to file inclusion vulnerabilities within the OWASP Broken Web Application, underscoring the significance of addressing multiple security aspects to ensure a robust and resilient web application.

**Conclusion**

In conclusion, the assessment of the OWASP Broken Web Application project showcased the complexity of securing web applications against common vulnerabilities. The utilization of Burp Suite to expose shortcomings in client-side filtering underscored the importance of implementing robust input validation and output encoding practices. The successful bypass of filtering mechanisms demonstrated the need for a layered defence strategy that encompasses both client-side and server-side security measures. Furthermore, the identification of file inclusion vulnerabilities emphasizes the necessity of continuous testing and remediation efforts to maintain a strong security posture. Ultimately, this analysis reinforces the critical role of security awareness and proactive measures in building resilient web applications.