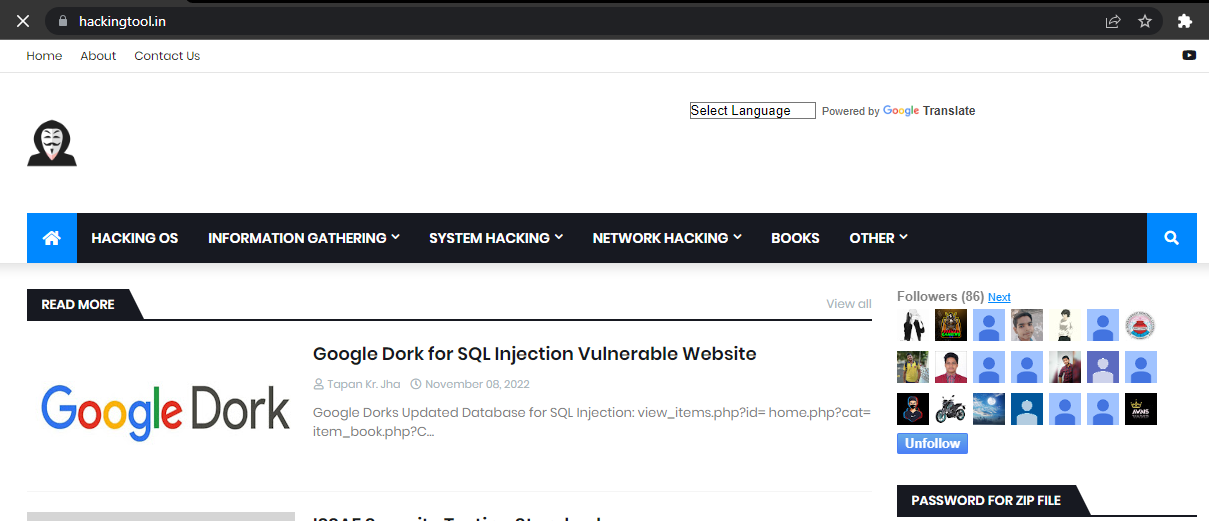
**Web Hacking**

**Lab Description:**

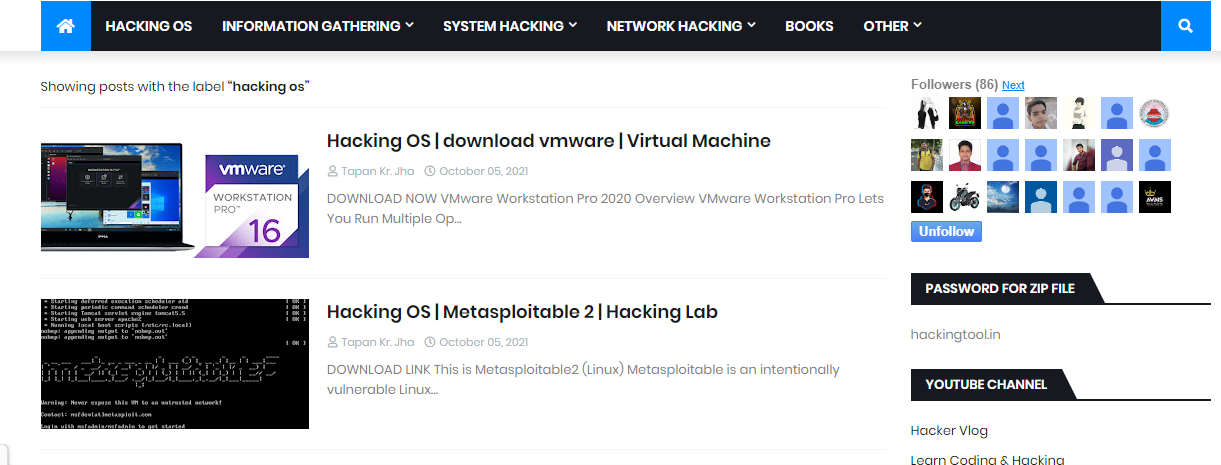
1. **Mutillidae Machine is available on Metasploitable VM. Start the Metasploitable VM and find the IP address of your Metasploitable machine.**

**Download Metasploit:**

* **Visit website** [**https://www.hackingtool.in/**](https://www.hackingtool.in/)

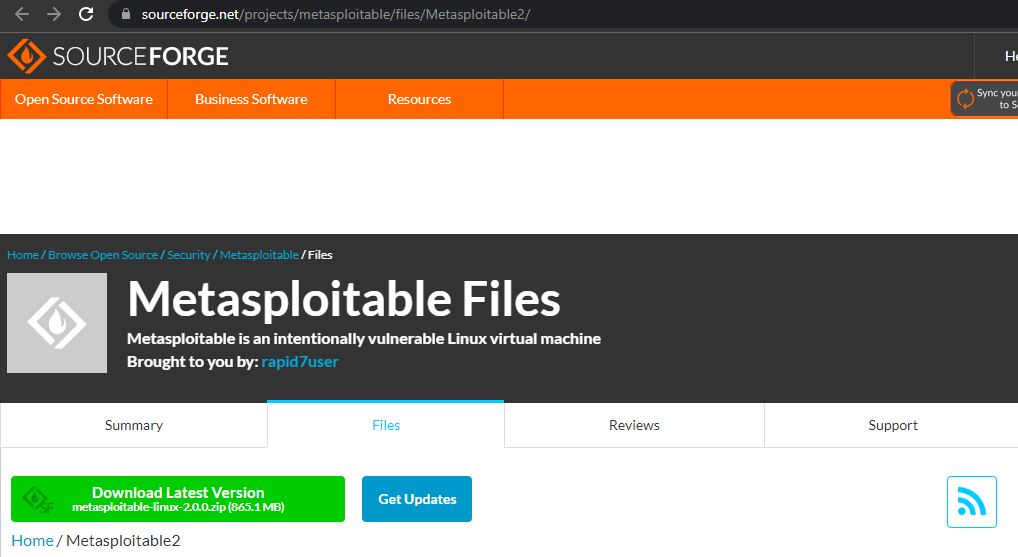
****

* **Click hacking OS**

****

* **Click Metasploitable2 | hacking lab**

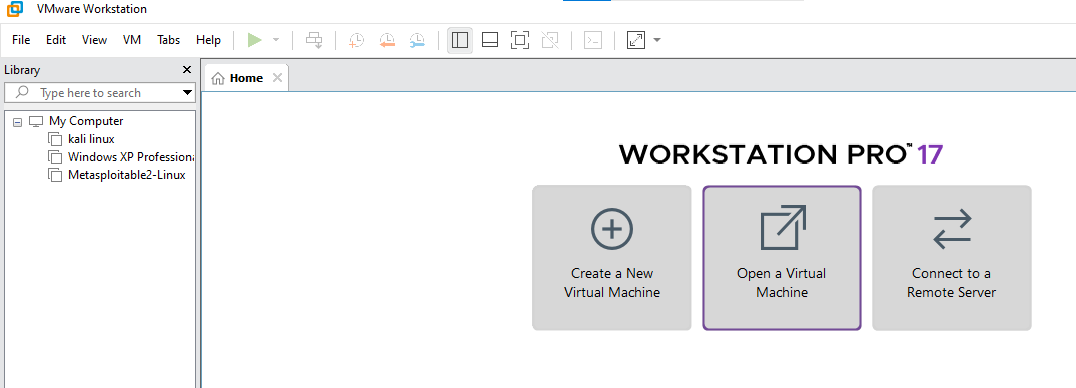
**Reach to the website:**

****

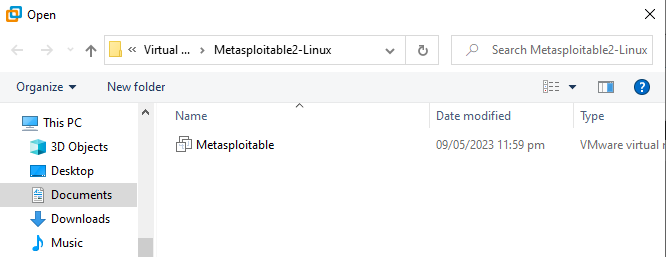
**Download latest version of Metasploitable2 from here.**

**Accessing Metasploitable2:**

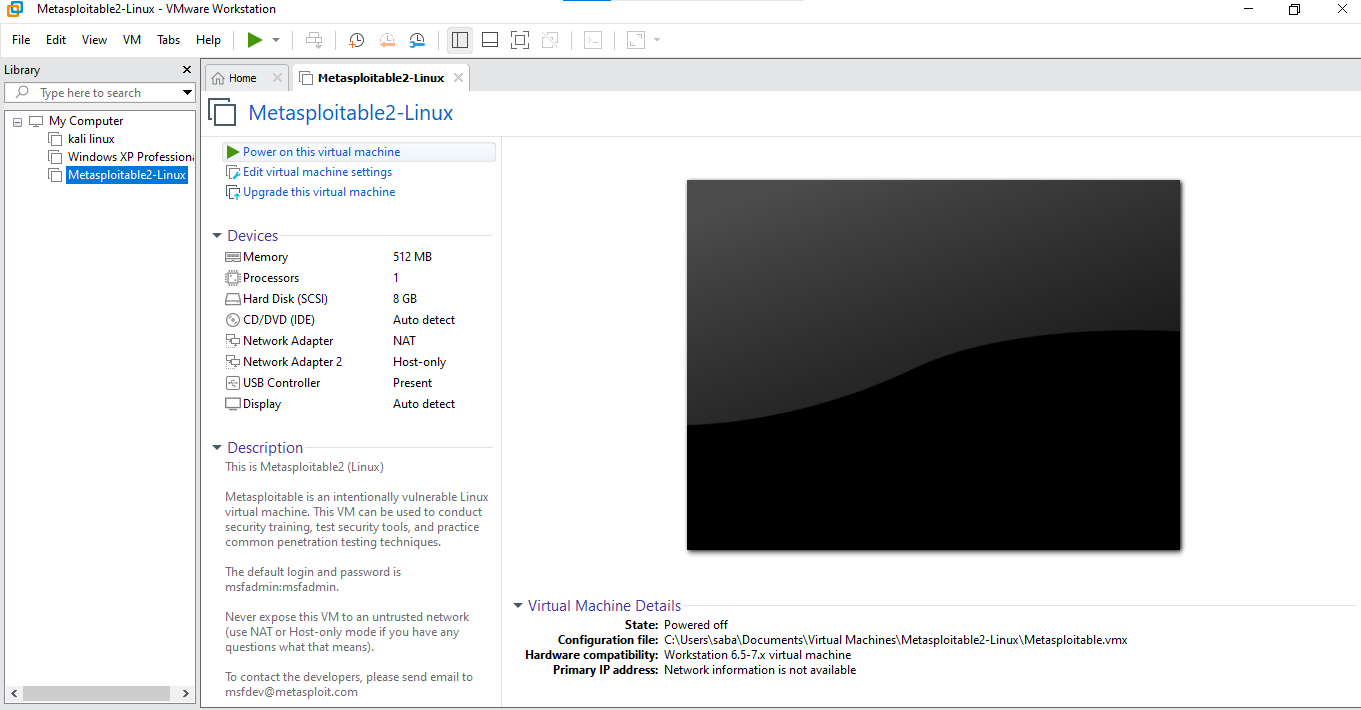
* **Open VMware**

****

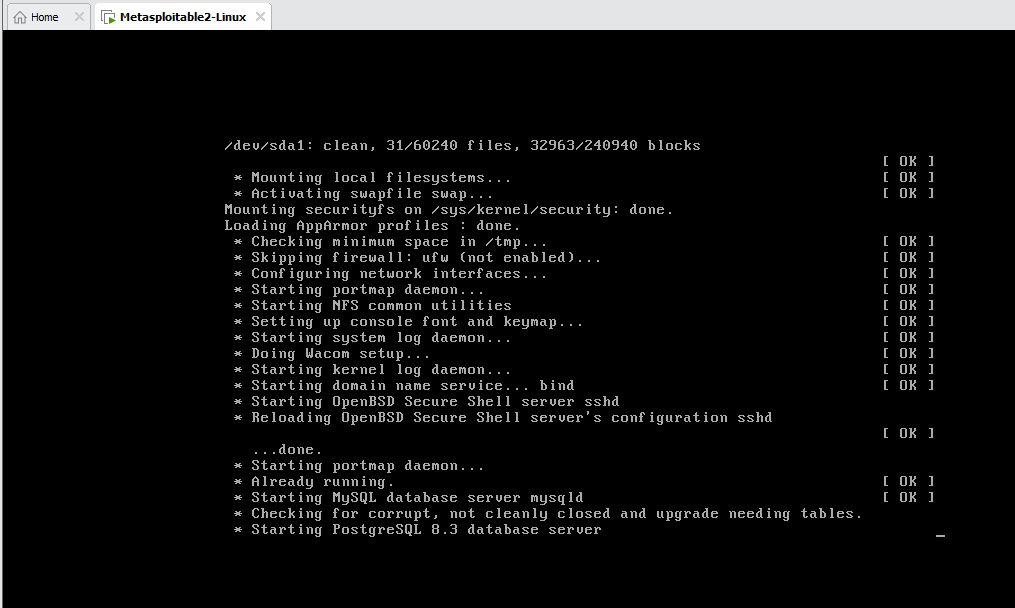
**Open virtual machine**

****

**No need of manual settings.**

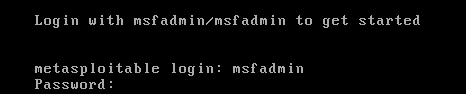
****

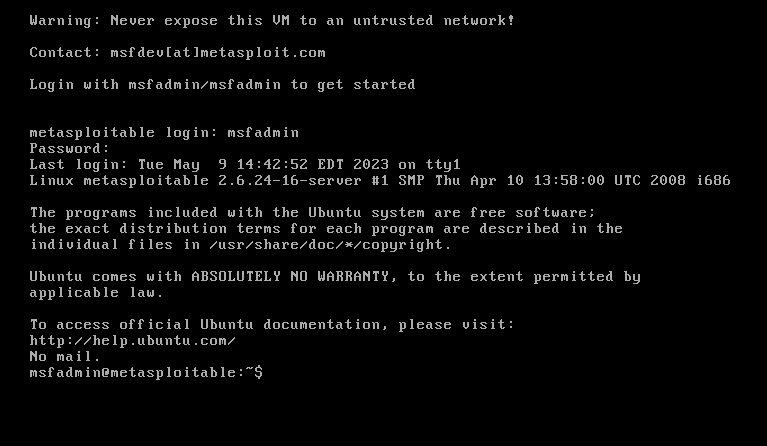
**Power on virtual machine.**

****

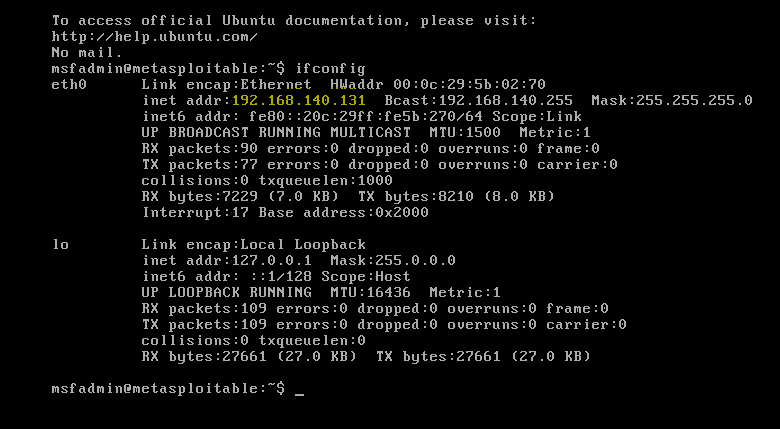
**Starting machine.**

**Add login credentials:**



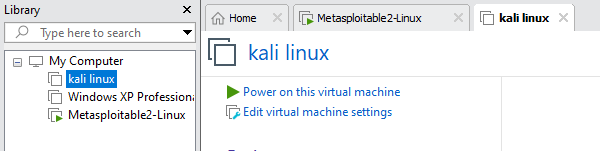


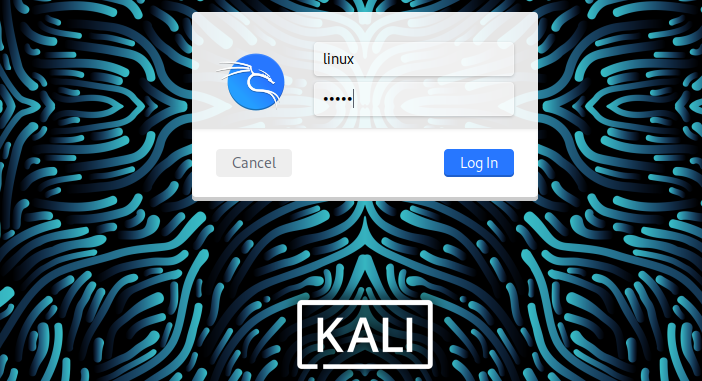
**Using if config find IP address**



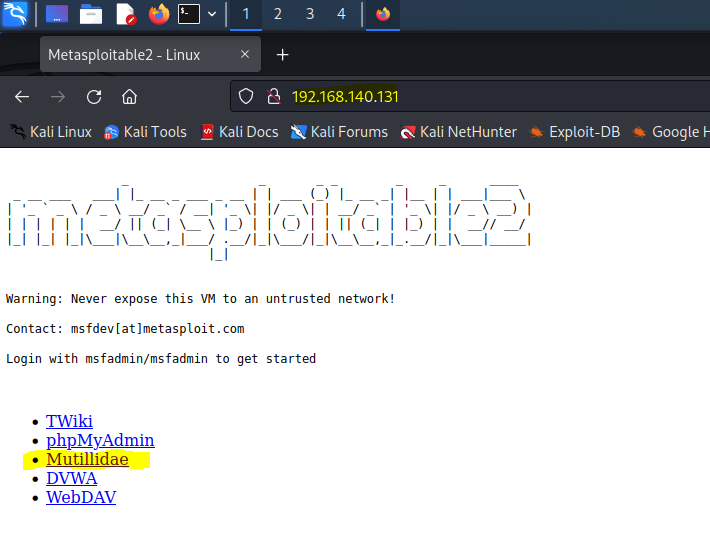
1. **Access the Metasploitable machine from the browser of your Kali Linux machine.**

* **Power on kali linux**

****



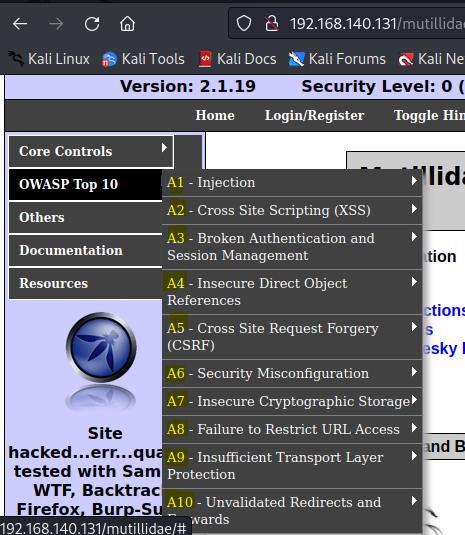
1. **Launch the Mutillidae Machine application and explore different vulnerable features and exercises available.**



**Open Mulillidae**



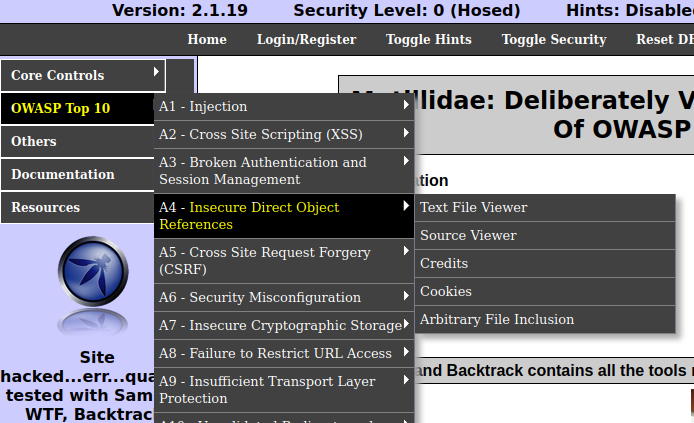
**In OWASP Top 10:**



**4. Write a paragraph about each vulnerability listed below**

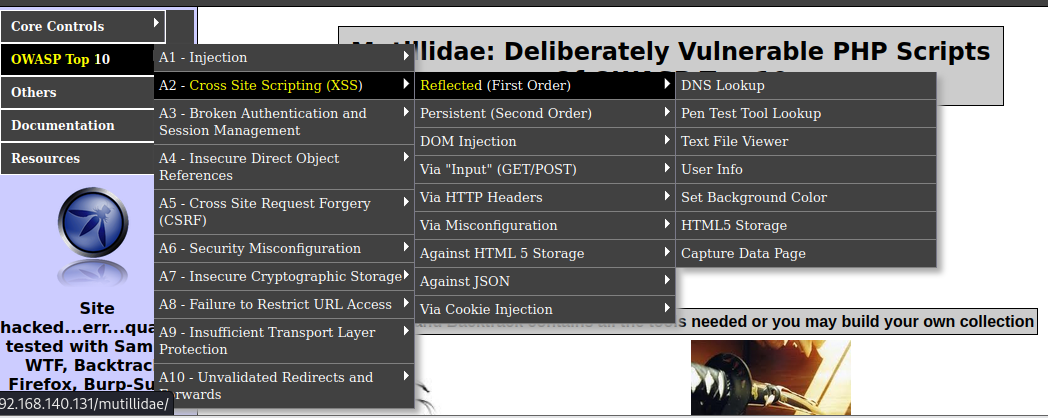
* **Insecure Direct Object Referencing**

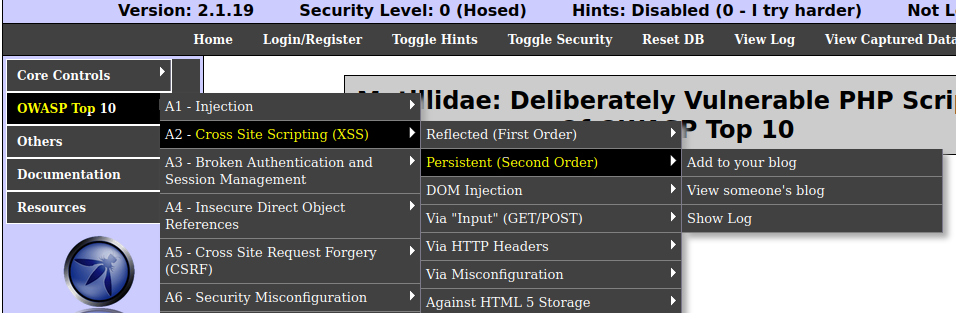
The Mutillidae Machine application provides a variety of vulnerable features and exercises for web application security testing. One common vulnerability found in Mutillidae is Insecure Direct Object Referencing (IDOR), which allows an attacker to access unauthorized resources by manipulating internal object references. Mutillidae has several exercises that demonstrate IDOR, including manipulating URLs to access restricted user information or modify database records.



* **Cross-Site Scripting (Reflected and Stored)**

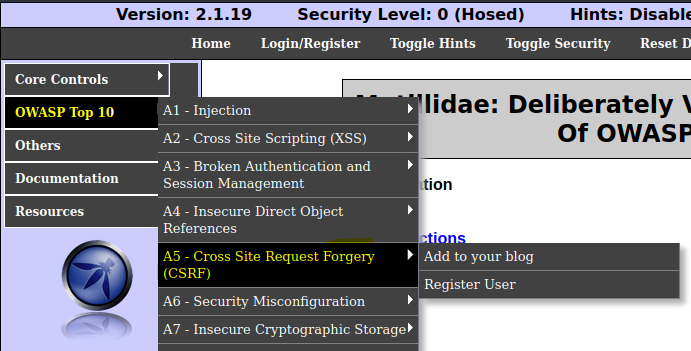
Cross-Site Scripting (XSS) is another vulnerability that Mutillidae highlights, with both Reflected and Stored XSS exercises available. Reflected XSS allows an attacker to inject malicious code into a URL, which is then executed by the user's browser when they access the link. Stored XSS occurs when an attacker injects malicious code into a web application's database, which is then executed by other users who access the compromised data. Mutillidae provides exercises that tell both types of XSS attacks.





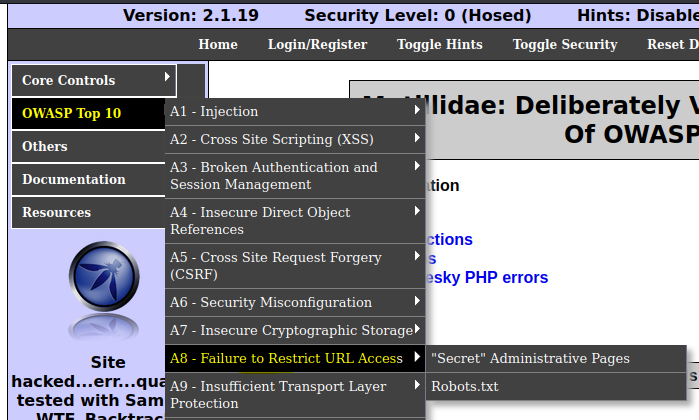
* **Cross-Site Request Forgery**

Cross-Site Request Forgery (CSRF) is a type of attack that tricks a user's browser into performing an unwanted action on a website where they are currently authenticated. Mutillidae has exercises that simulate CSRF attacks, such as manipulating a user's session token to make unauthorized purchases or change their password**.**



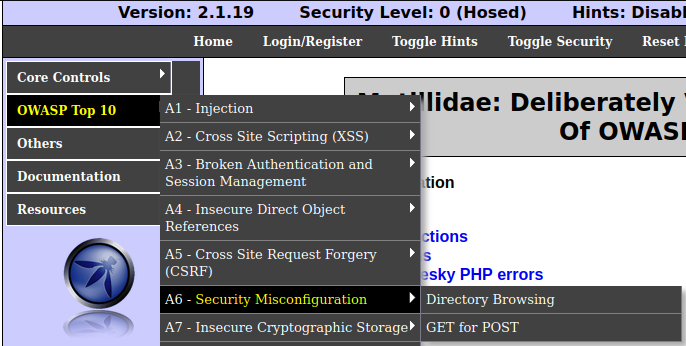
* **Failure to Restrict URL Access**

Failure to Restrict URL Access is a common vulnerability that Mutillidae addresses in several exercises. In these exercises, an attacker can gain access to sensitive data or perform unauthorized actions by simply navigating to a restricted URL without being properly authenticated or authorized.



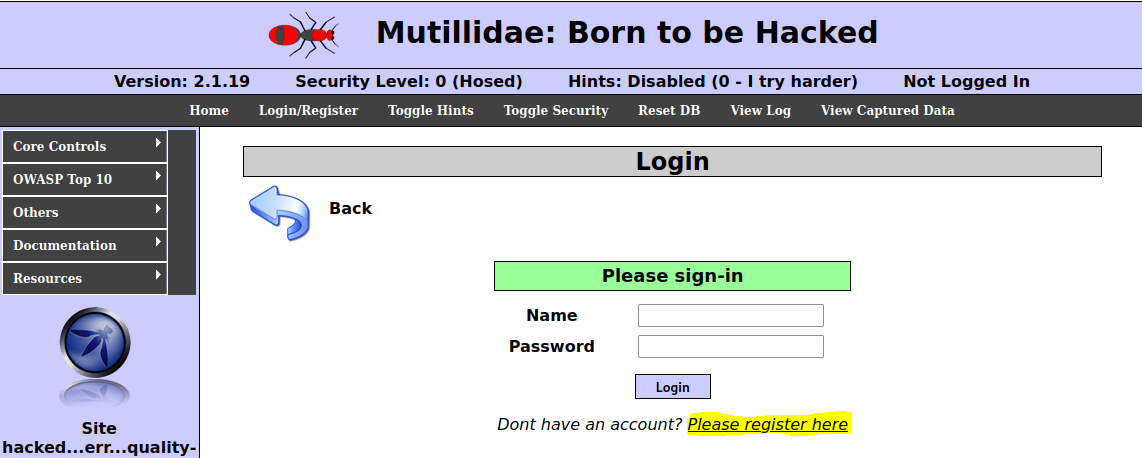
* **Security Misconfigurations**

Security Misconfigurations are also covered in Mutillidae, with exercises that demonstrate how to exploit vulnerabilities such as default passwords, weak encryption settings, and outdated software versions. By providing examples of misconfigurations, Mutillidae helps developers and security professionals understand how these vulnerabilities can be exploited and how to secure their applications against them.

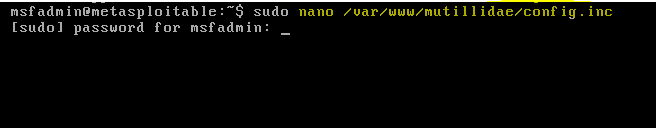


1. **Use Mutillidae Machine to find and exploit common web application vulnerabilities listed above. Document the steps you took to find and exploit the vulnerability, including any tools or techniques you have used.**

* **Login into Mutillidae**



**Open Metasploitable**

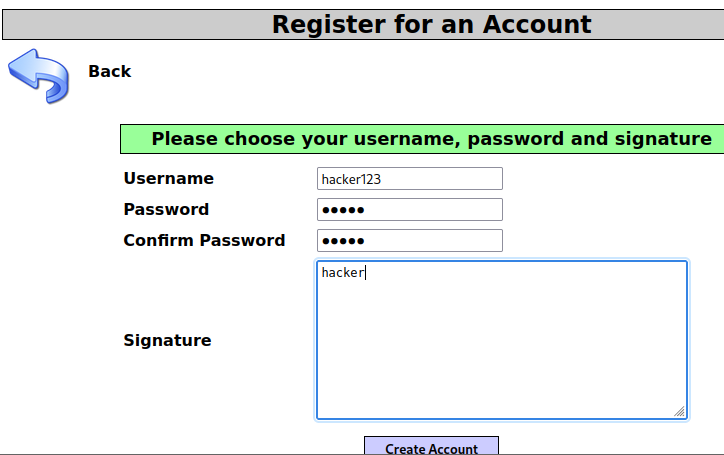




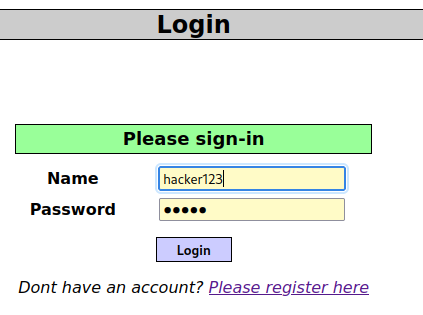
**Change database:**



**Registering for an account:**

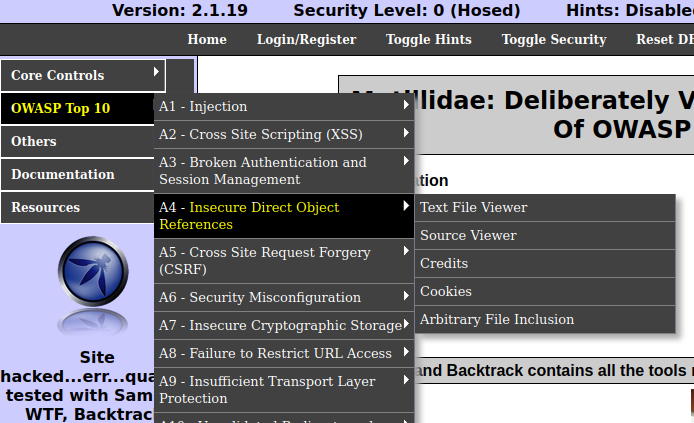


**Login into an account:**

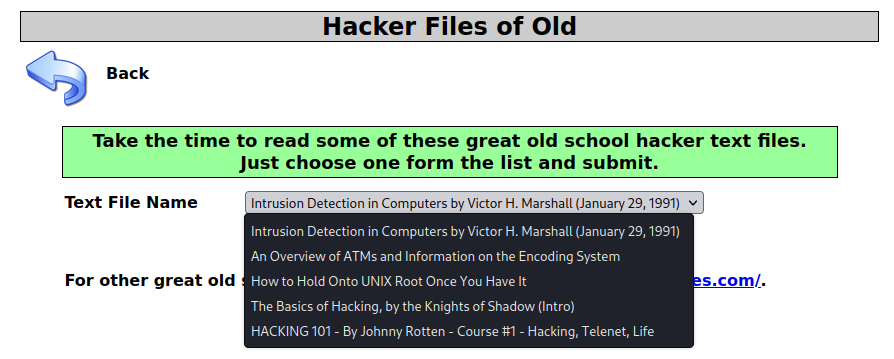


* **Insecure Direct Object Referencing**

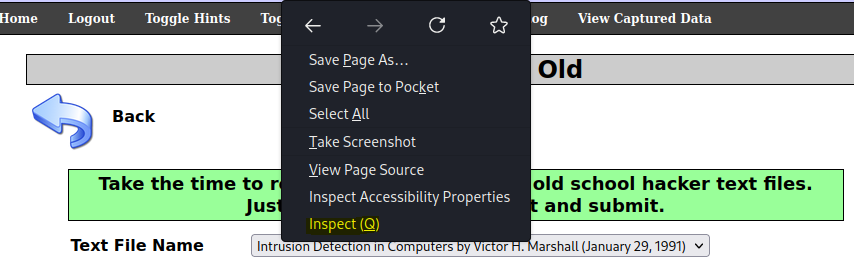
Let’s explore IDOR vulnerability



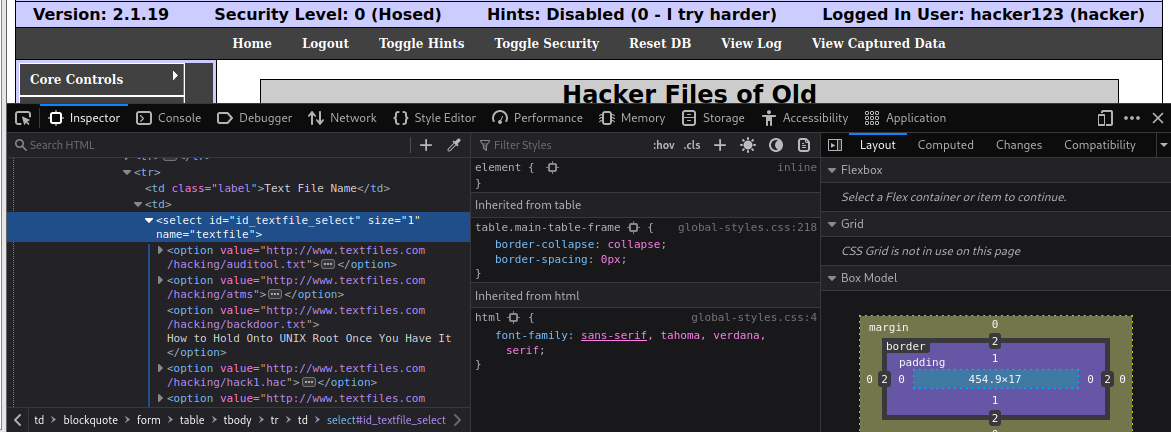
Different types of files we can access:



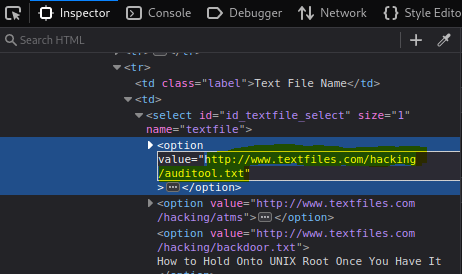
Go to inspect of that page



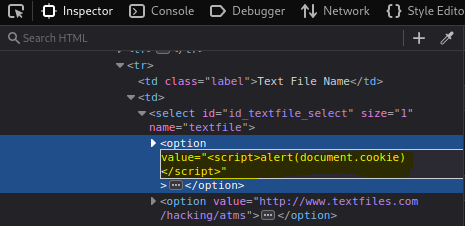
Here is inspect page



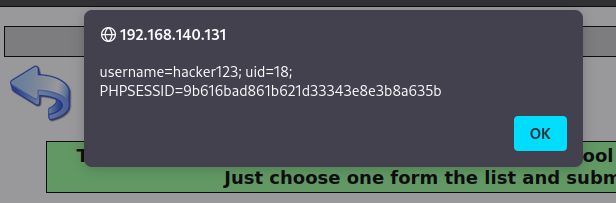
Showing address of the page



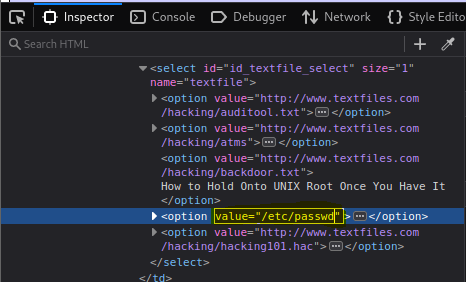
Editing page ( write script on its place )



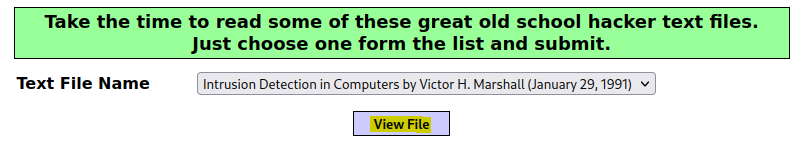
Extract cookies of current user:



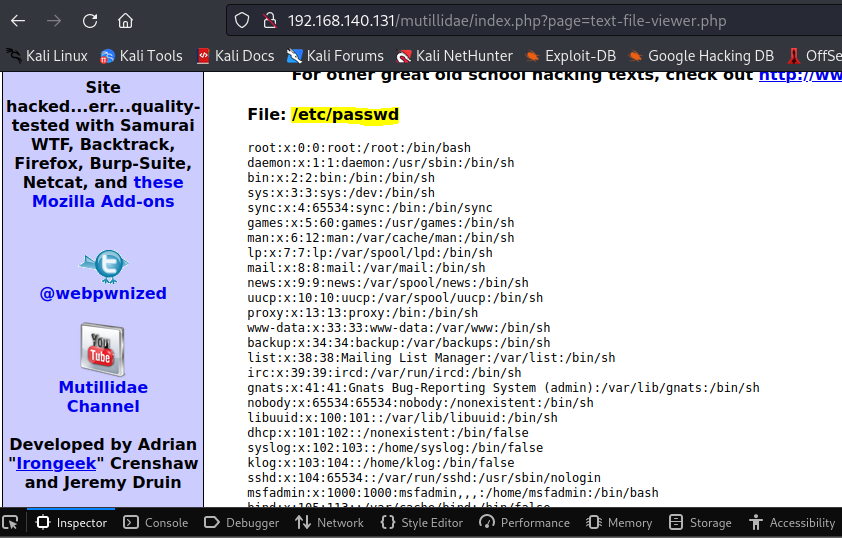
Change value for accessing password directory



Click view file

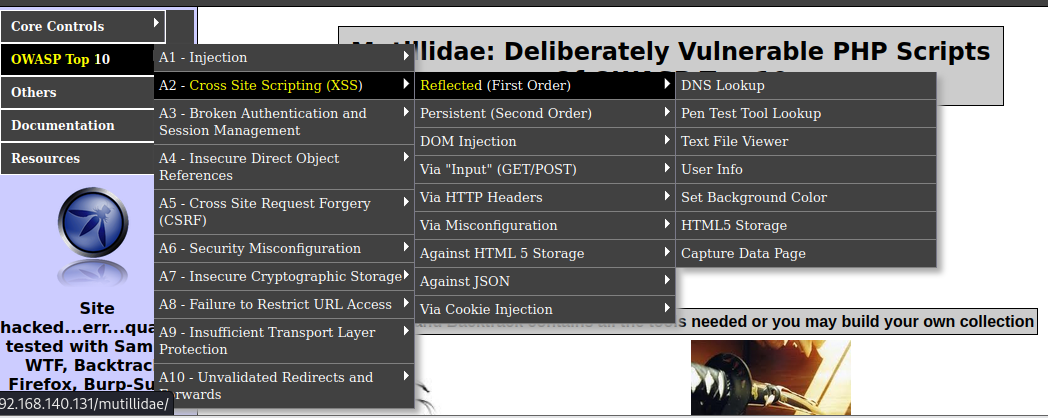


Here is the list of passwords and we **successfully Exploit IDOR** vulnerability.

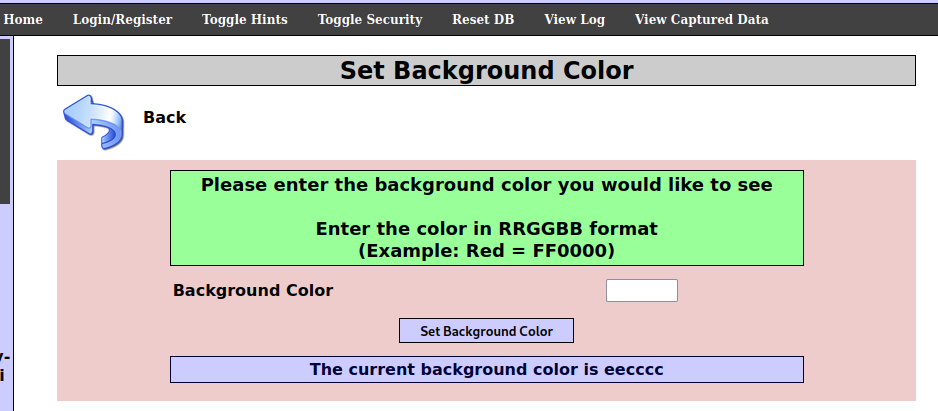


* **Cross-Site Scripting (Reflected)**

Let’s explore cross-site Scripting vulnerability



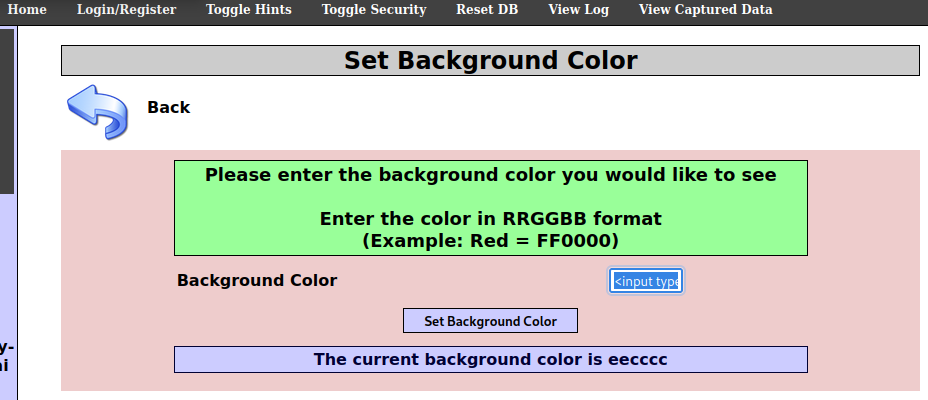
Open set background option.



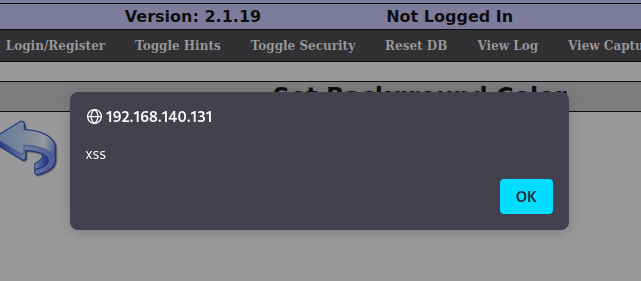
**Script to exploit xss vulnerability:**

<input type="text" name="colour" value="FF0000"><script>alert("xss")</script> " />

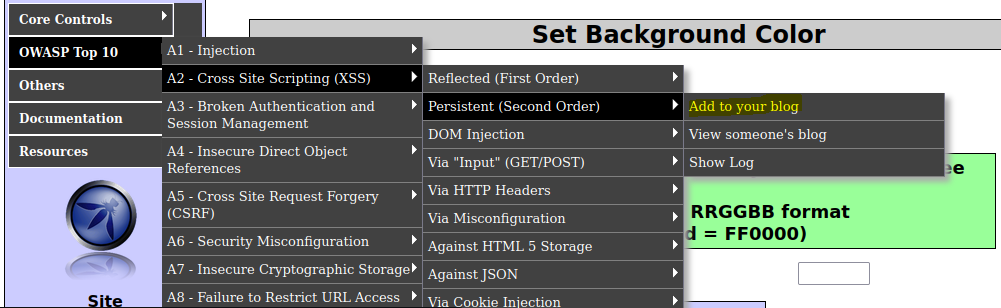
* Paste this script in input field.



This message shows that script is runing.



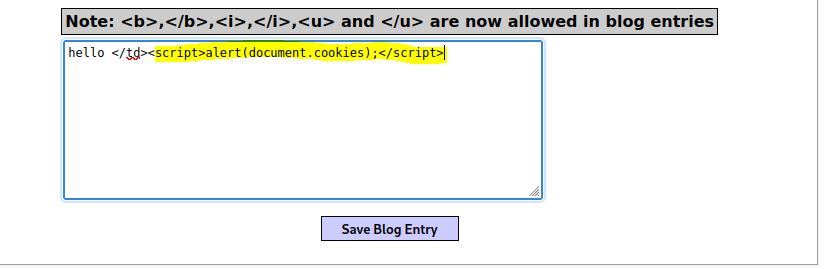
**Cross-Site Scripting (stored)**



**Script for exploiting xss vulnerability:**

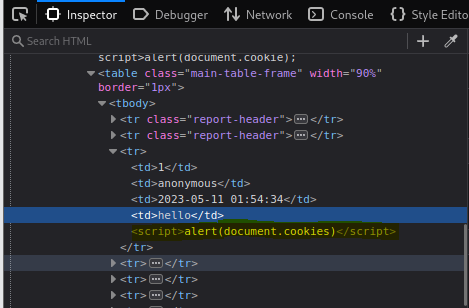
hello </td><script>alert(document.cookies);</script>

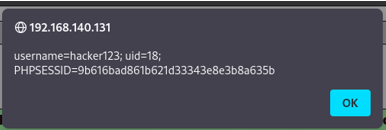
* Paste Script in input field.



Click save blog entry to execute script and save this script in database.

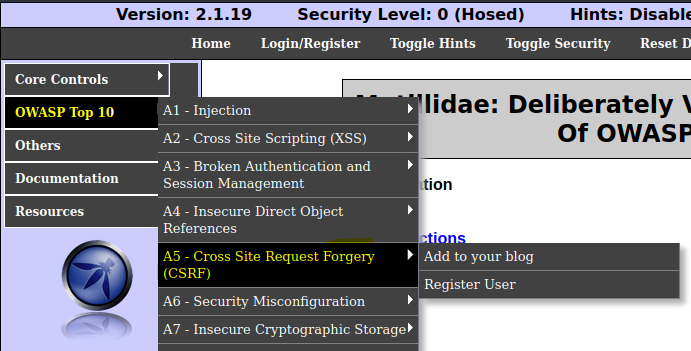
Refresh page and see in inspect that script in saved.

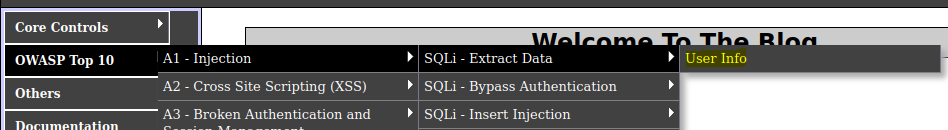


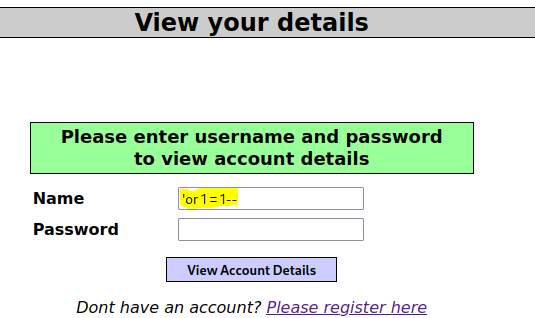


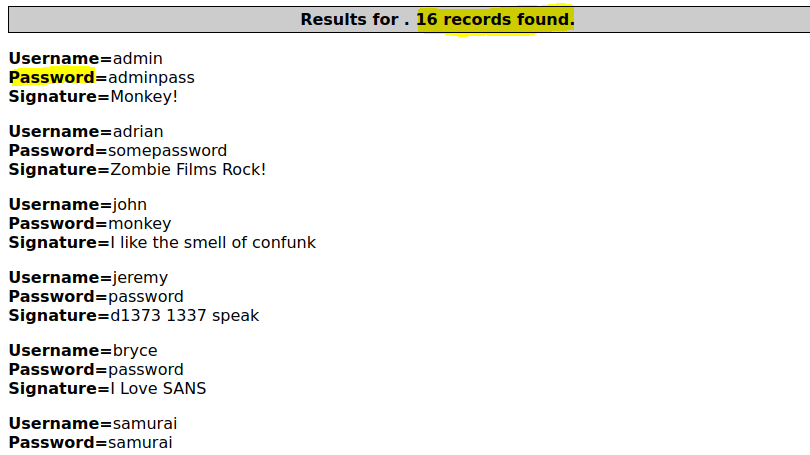
So, we successfully launch attack and always we get cookie.

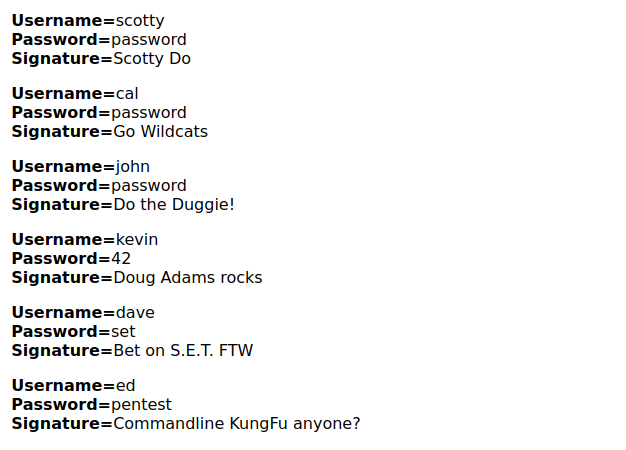
* **Cross-Site Request Forgery**





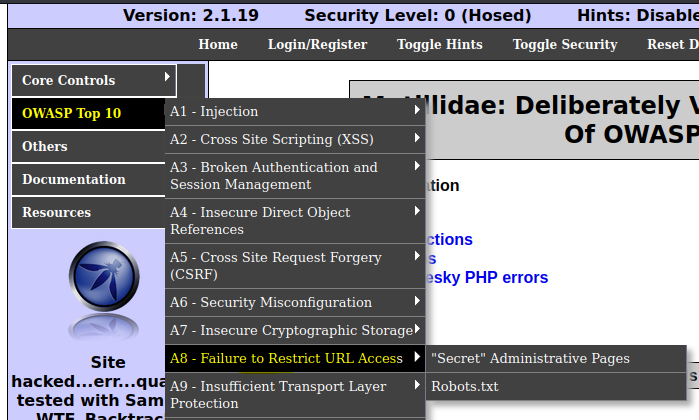


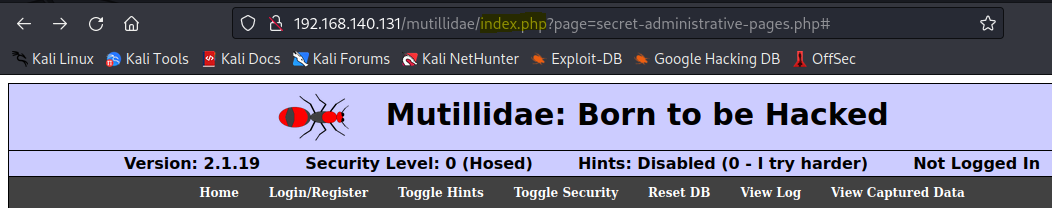


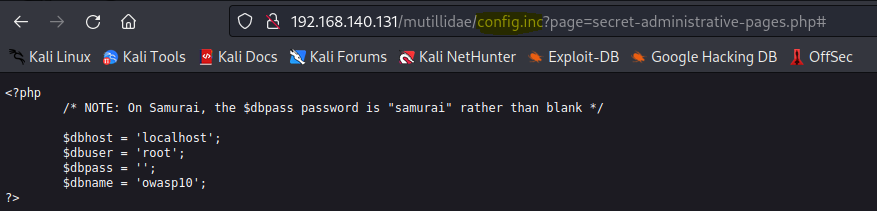


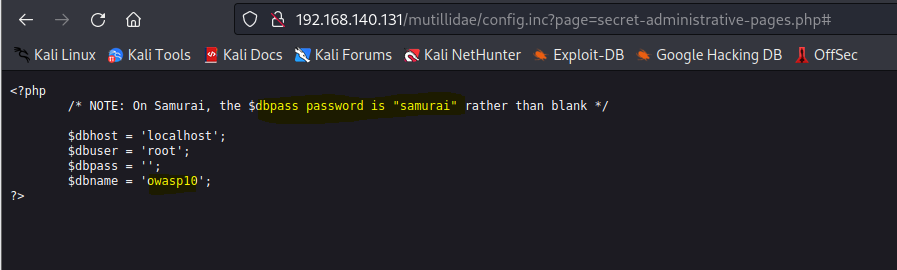
So, we successfully exploit Cross-Site Request Forgery vulnerability.

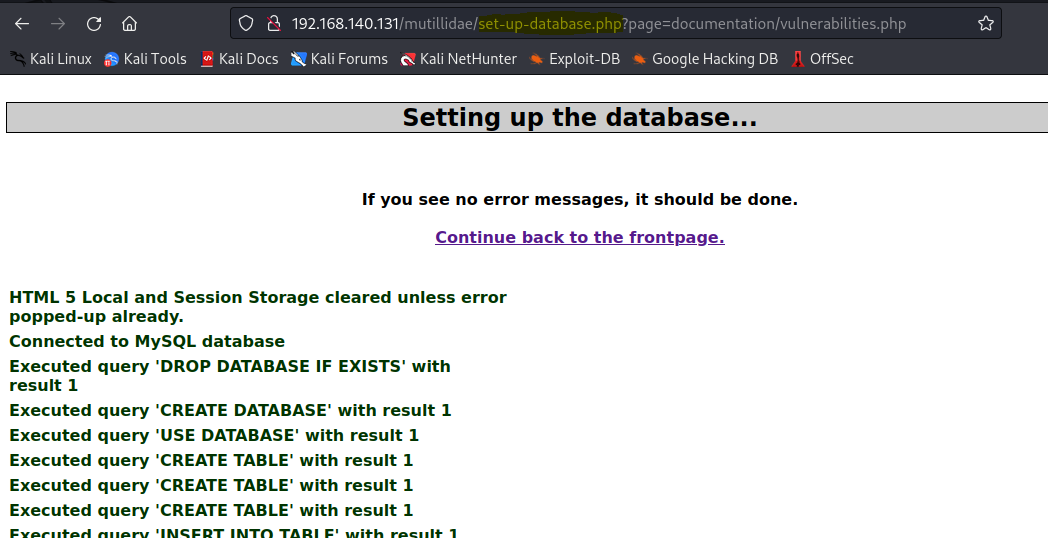
**Failure to Restrict URL Access**







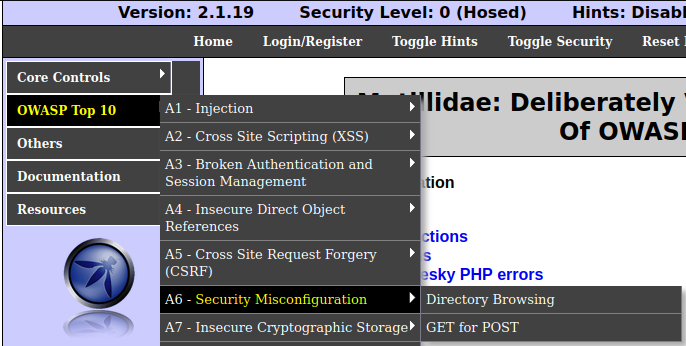


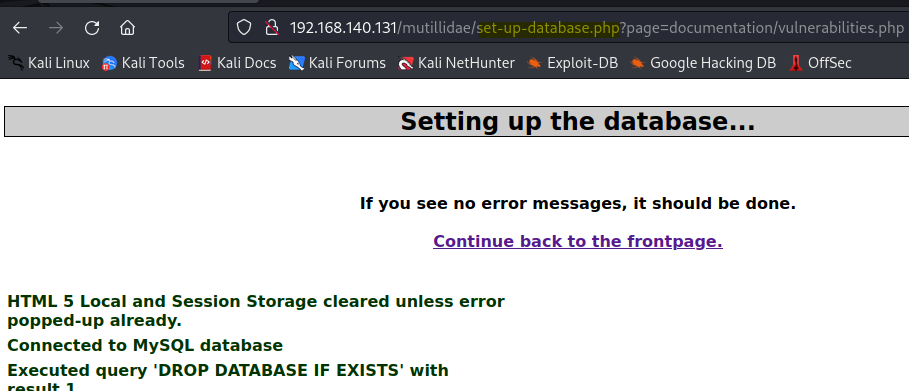


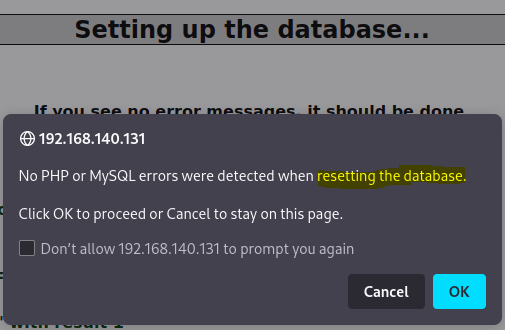
So, we successfully exploit failure to Restrict URL Access

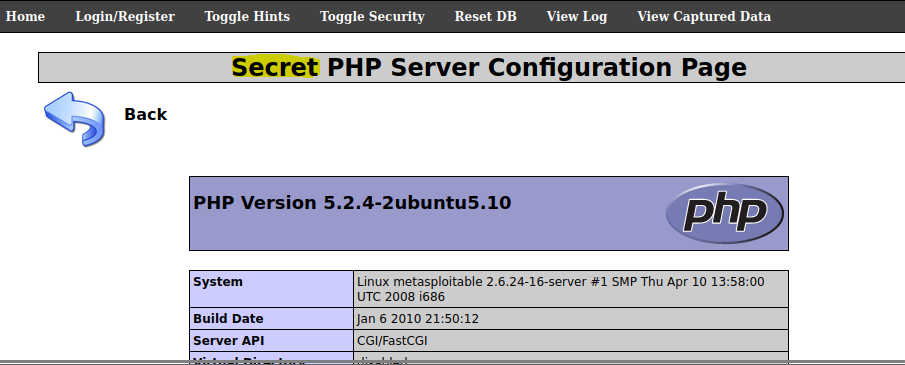
vulnerability.

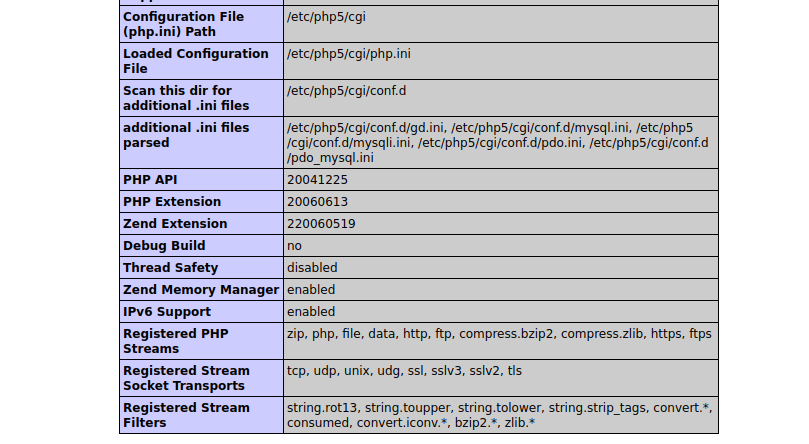
**Security Misconfigurations**









So, we successfully exploit Security Misconfigurations vulnerability.

1. **Once you have successfully exploited the vulnerability, document how you would recommend the vulnerability be mitigated and provide a brief explanation of why this mitigation technique is effective.**
2. **Insecure Direct Object Referencing:**

* Occurs when an application allows direct access to an object without proper authorization
* Attackers can exploit this vulnerability to access and manipulate sensitive data
* **Mitigation techniques:**
  + Implement access controls based on user roles and permissions
  + Use authentication to verify the user's identity
* **Remediation table:**

|  |  |
| --- | --- |
| **Mitigation Technique** | **Explanation** |
| **Implement access controls** | Implement access controls to restrict access to specific objects based on the user's role and permissions. This can help to prevent unauthorized access and manipulation of sensitive data. |
| **Use authentication** | Use authentication to verify the user's identity before granting access to sensitive objects. This can help to ensure that only authorized users are able to access the objects. |

1. **Cross-Site Scripting (XSS):**

* Reflected XSS occurs when user input is reflected back to the user in a way that allows malicious code to be executed
* Stored XSS occurs when user input is stored on the server and executed when a user views the affected page
* Attackers can use XSS vulnerabilities to steal user data, perform actions on behalf of the user, or deface the website
* **Mitigation techniques:**
  + Implement input validation to ensure that user input does not contain any malicious code
  + Use input filters and sanitization techniques to remove any potentially harmful characters or scripts
* **Remediation table:**

|  |  |
| --- | --- |
| **Mitigation Technique** | **Explanation** |
| **Implement input validation** | Implement input validation to ensure that user input does not contain any malicious code. This can help to prevent XSS attacks by detecting and rejecting malicious input. |
| **Use input filters and sanitization techniques** | Use input filters and sanitization techniques to remove any potentially harmful characters or scripts from user input. This can help to prevent XSS attacks by removing the ability for attackers to inject malicious code. |

1. **Cross-Site Request Forgery (CSRF):**

* Occurs when an attacker tricks a user into performing an action on a website without their knowledge or consent
* Attackers can use CSRF vulnerabilities to perform actions on behalf of the user, such as changing passwords or making unauthorized purchases
* **Mitigation techniques:**
  + Implement CSRF tokens to validate requests and prevent unauthorized actions
  + Use CAPTCHA’s to prevent automated attacks
* **Remediation table:**

|  |  |
| --- | --- |
| **Mitigation Technique** | **Explanation** |
| Implement CSRF tokens | Implement CSRF tokens to validate requests and prevent unauthorized actions. CSRF tokens are unique tokens that are generated for each session and are used to validate requests. |
| Use CAPTCHA’s | It prevents automated attacks by requiring users to solve a puzzle or enter a code before performing certain actions. Make it more difficult for attackers to automate the attack. |

1. **Failure to Restrict URL Access:**

* Occurs when an application allows unauthenticated users to access sensitive pages or resources
* Attackers can use this vulnerability to access sensitive data or perform unauthorized actions
* **Mitigation techniques:**
  + Implement access controls to restrict access to sensitive pages and resources
  + Use authentication to verify the user's identity
* **Remediation table:**

|  |  |
| --- | --- |
| **Mitigation Technique** | **Explanation** |
| **Implement access controls** | Implement access controls to restrict access to sensitive pages and resources based on the user's role and permissions. This can help to prevent unauthorized access to sensitive data or actions. |
| **Use authentication** | Use authentication to verify the user's identity before granting access |

1. **Security Misconfigurations**

* Security Misconfiguration can happen due to a variety of reasons, including weak passwords, outdated software, unsecured network ports, unnecessary features enabled, or incorrect permission settings.
* Attackers can exploit these vulnerabilities to gain unauthorized access, steal data, or perform other malicious activities.
* Mitigating Security Misconfiguration involves ensuring that the application, system, or software is configured securely, with secure default settings, secure passwords, and up-to-date software.
* Regular security audits and vulnerability scanning can help detect any misconfigurations and provide recommendations for remediation.
* Using password managers, multi-factor authentication, and implementing password policies that require regular password changes and disallow reused passwords can help in mitigating the risks of weak passwords.
* Keeping software up-to-date by regularly applying patches and updates, using vulnerability scanning tools, and disabling or removing unused software or services can help in mitigating the risks of outdated software.
* Regular security audits, monitoring and logging tools, and incident response plans can help in quickly identifying and responding to security incidents.

|  |  |
| --- | --- |
| **Mitigation Technique** | **Remediation Technique** |
| **Use secure default settings** | Review and update the default settings for the application or software |
|  | Disable unnecessary features or services |
|  | Ensure that sensitive data is encrypted at rest and in transit |
| **Use secure passwords** | Ensure that strong passwords are enforced for all users |
|  | Use password managers and multi-factor authentication |
|  | Implement password policies that require regular password changes and disallow reused passwords |
| **Keep software up-to-date** | Regularly apply software patches and updates |
|  | Use vulnerability scanning tools to detect and remediate vulnerabilities |
|  | Disable or remove unused software or services |
| **Regular security audits** | Conduct regular security audits and penetration testing to identify and remediate vulnerabilities |
|  | Use monitoring and logging tools to detect suspicious activity |
|  | Implement incident response plans to respond to security incidents |