Subject: Security Testing

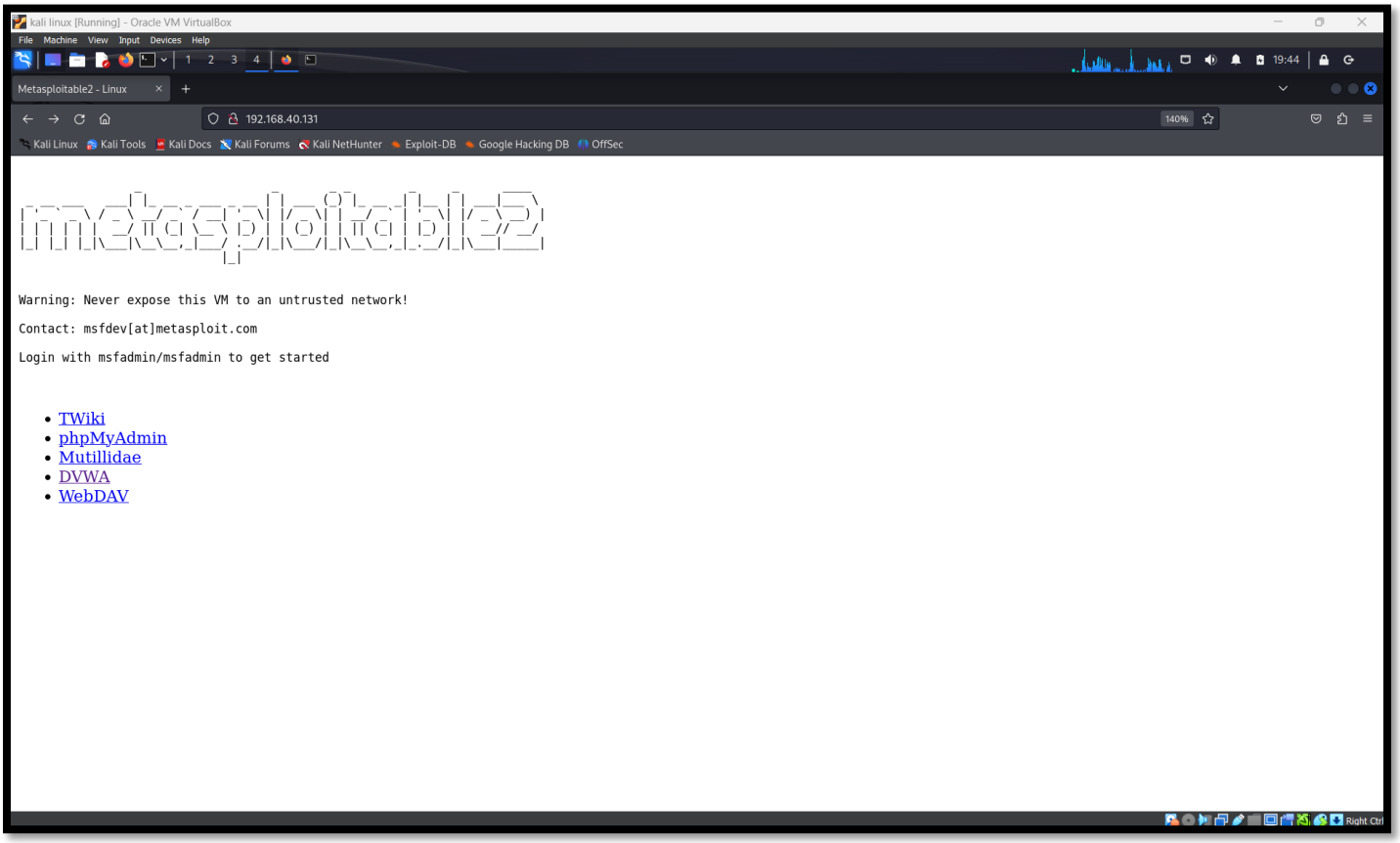
SENG 8061

ASSIGNMENT – 2

Vulnerability Assessment on DVWA application

Name	Student ID
Shivani Varu	8941914
Mohammed Rafique	8954785

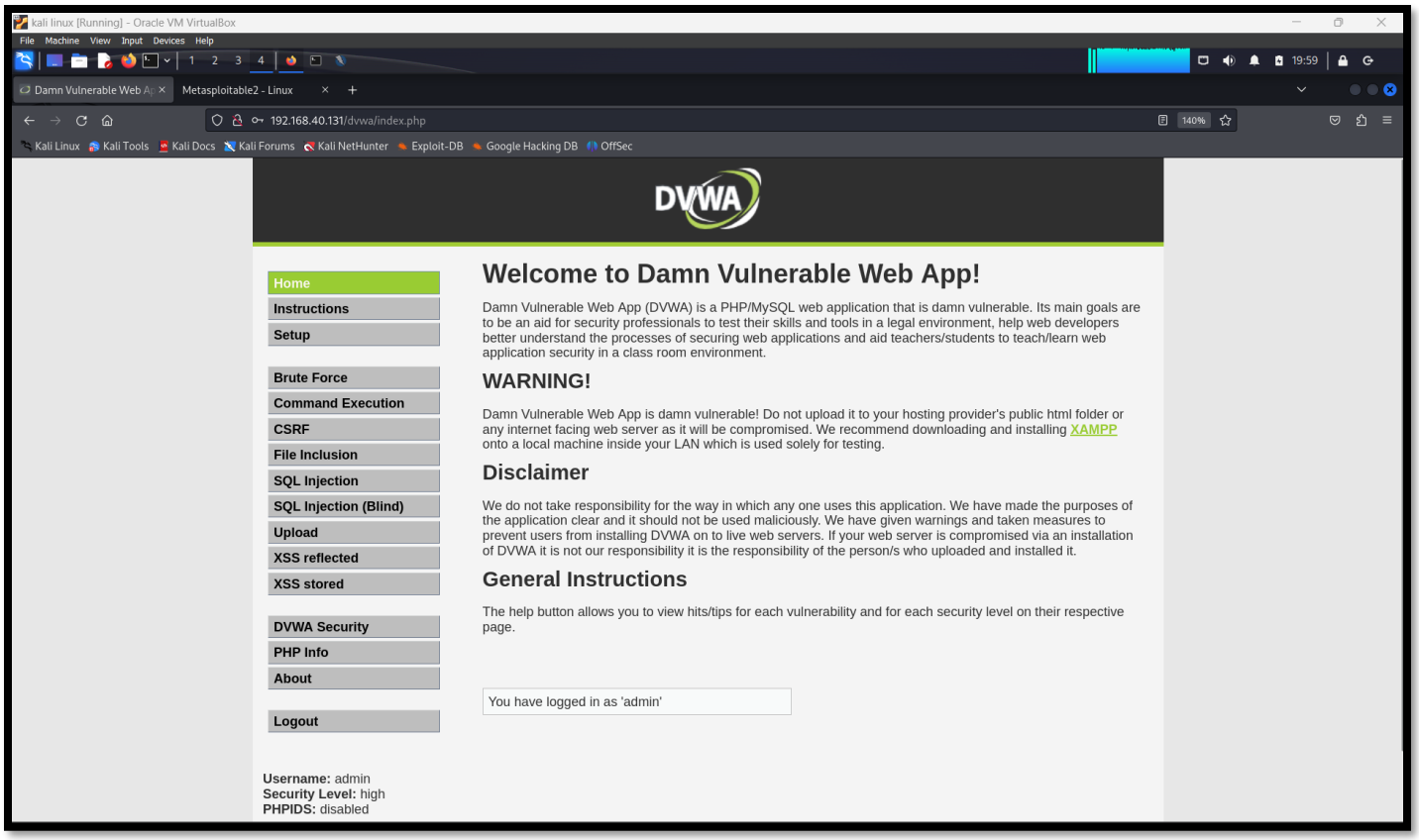
We installed Metasploitable2 on our virtual machine and started it up. Next, we logged into the terminal of Metasploitable2. To identify the IP address, we entered the command 'ifconfig' in the terminal and pressed enter. After obtaining the IP address, we copied it for later use. With the IP address in hand, we opened our web browser and entered the copied IP address into the address bar. This directed us to the website hosting DVWA (Damn Vulnerable Web Application).



Vulnerability #1: No password authentication was found.

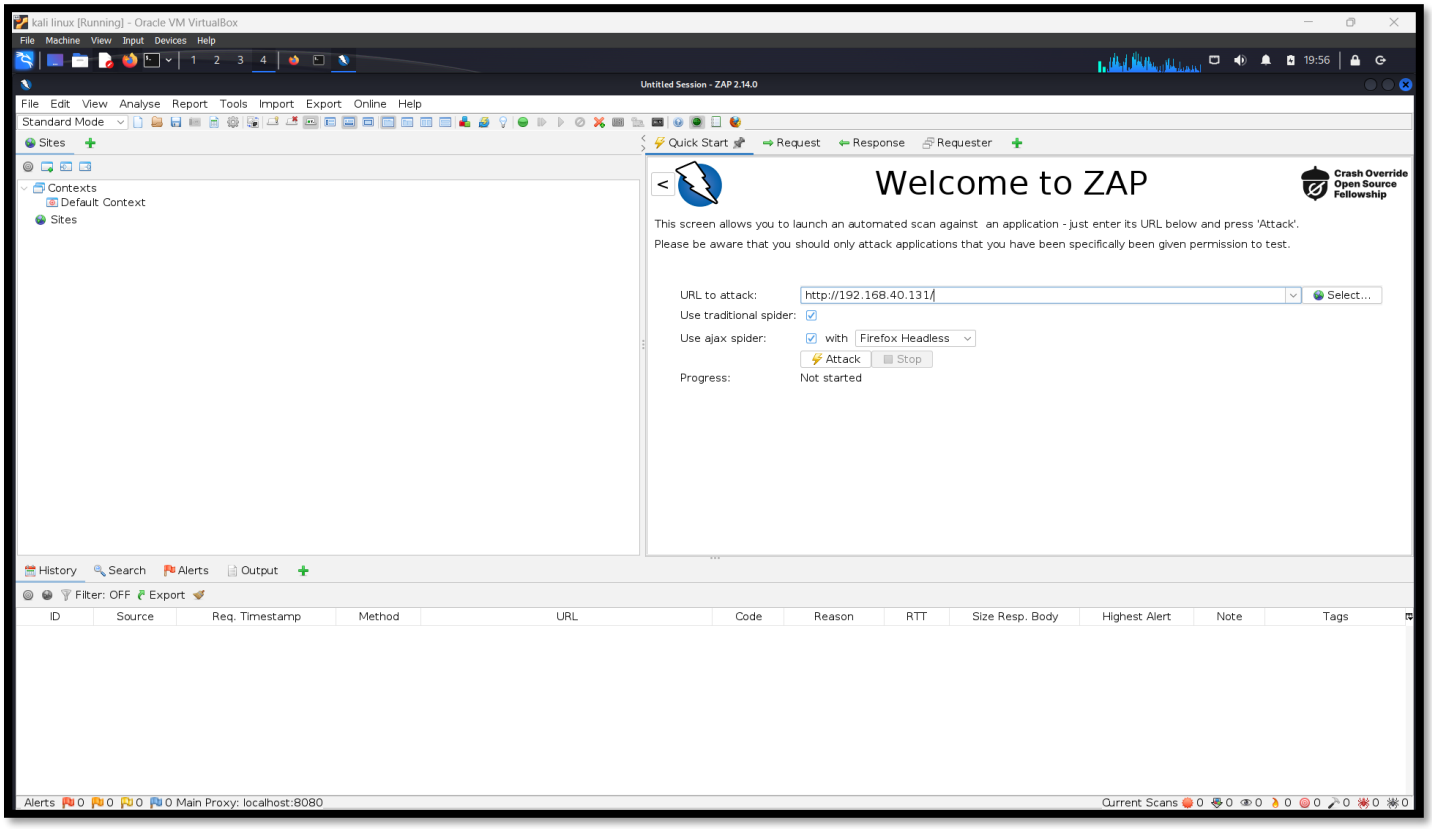
Discovered Vulnerabilities

We found a serious vulnerability when we evaluated the DVWA application. We discovered that there was no password authentication mechanism when we tried to log in as the admin user. This means that anyone can use the application without being required to provide authentic credentials.



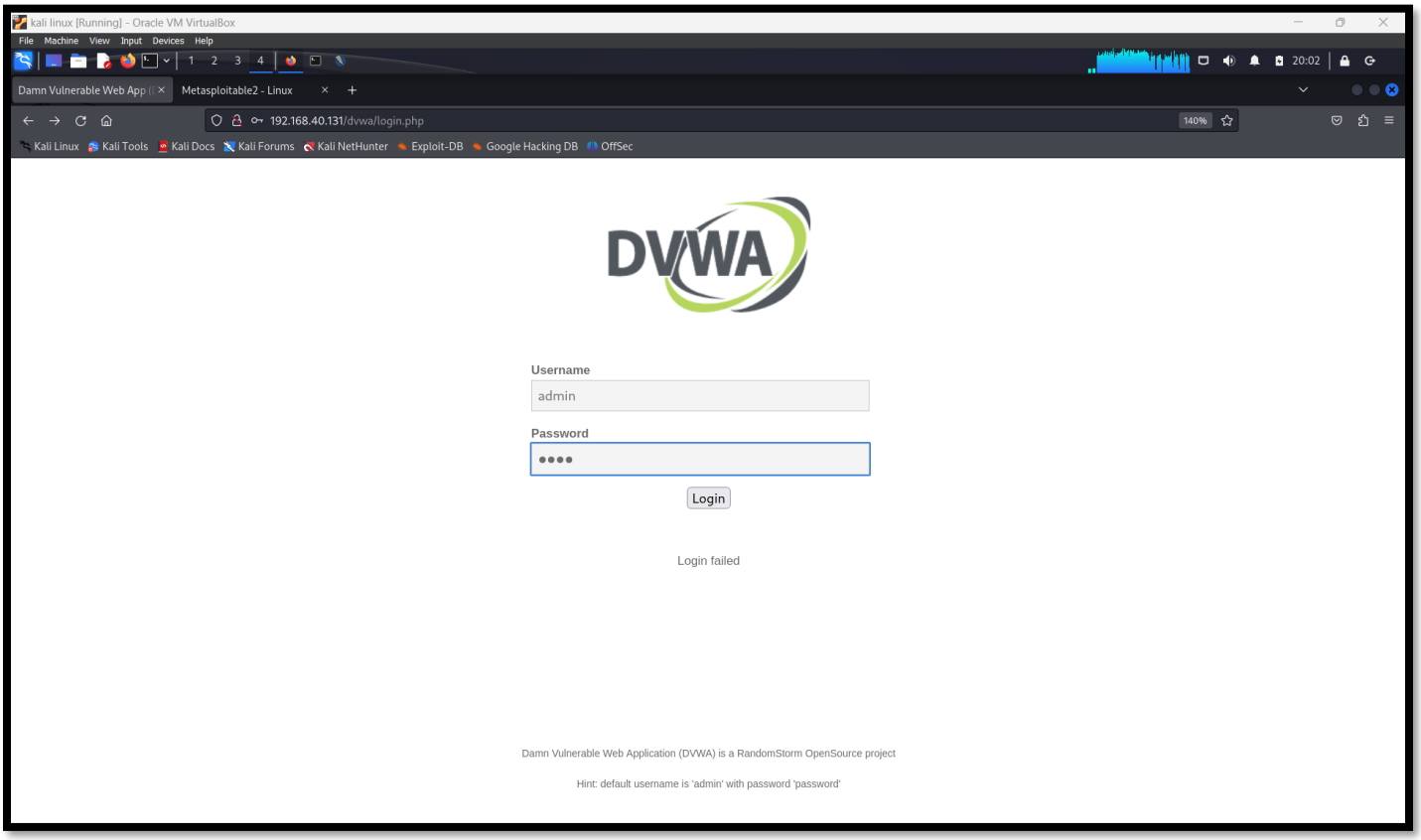
To exploit this vulnerability, we utilized OWASP Zap to execute a brute force attack. With the absence of authentication protections, attackers would have been able to access sensitive areas of the application without authorization.

Our next step was to set up Zap to act like an attack and attempt to bypass the user credentials.

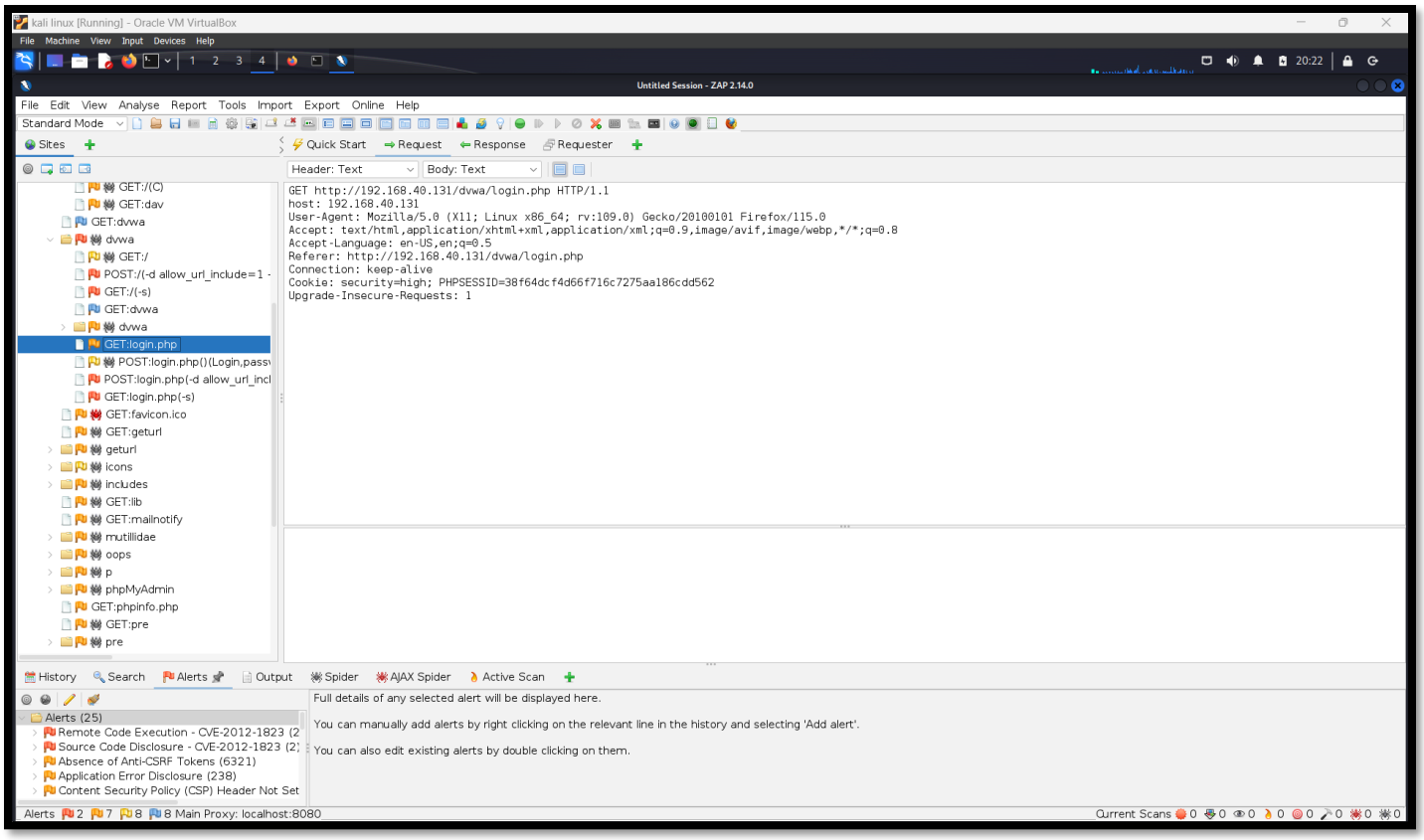


We opened the DVWA application in the Zap browser and entered the IP address for Metaspoitable. Then, we attempted to log in with the username "admin" and the password as SQETA.

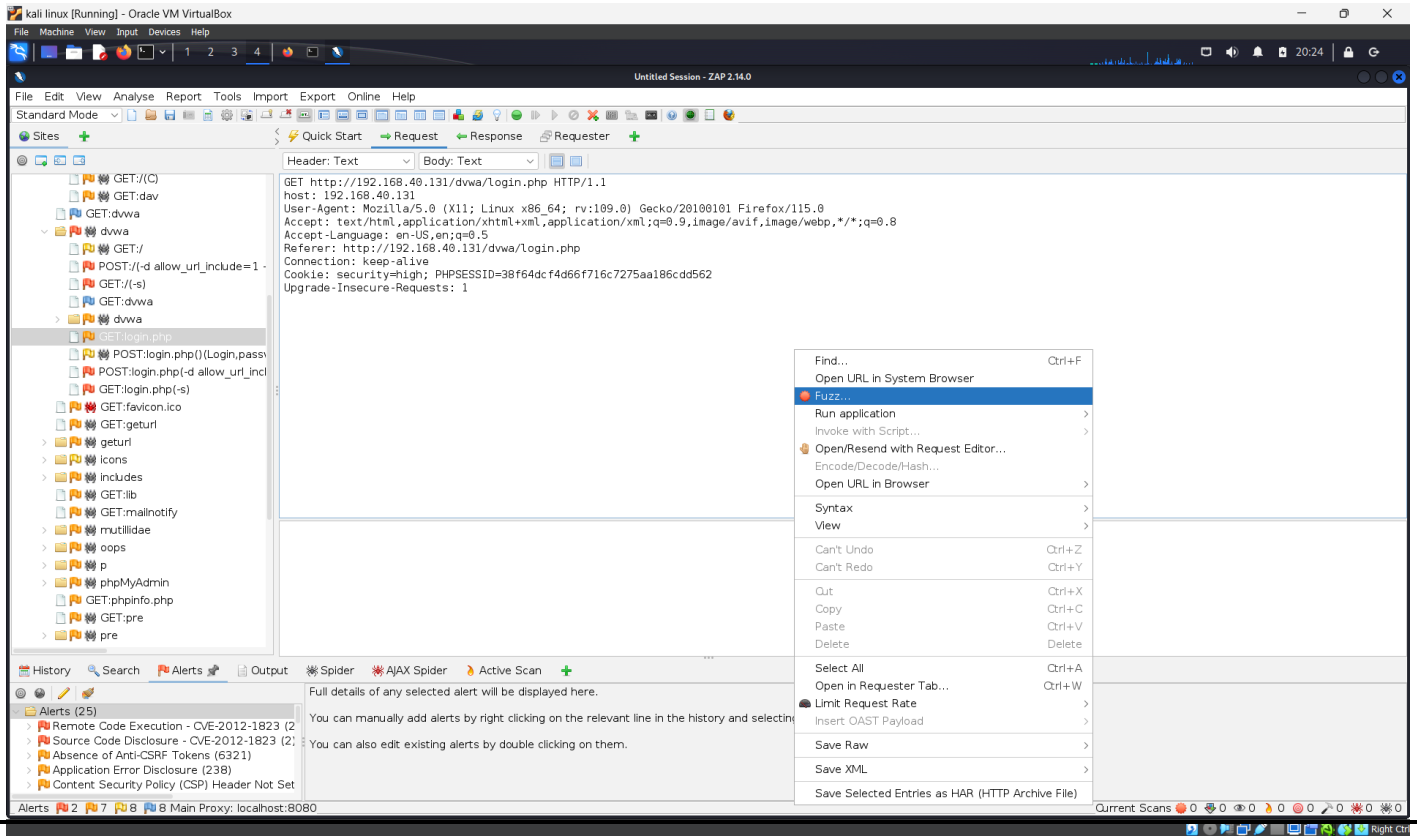
This shows an invalid password now let’s manipulate the password payload and try to enter some random strings which are commonly used passwords in the world.

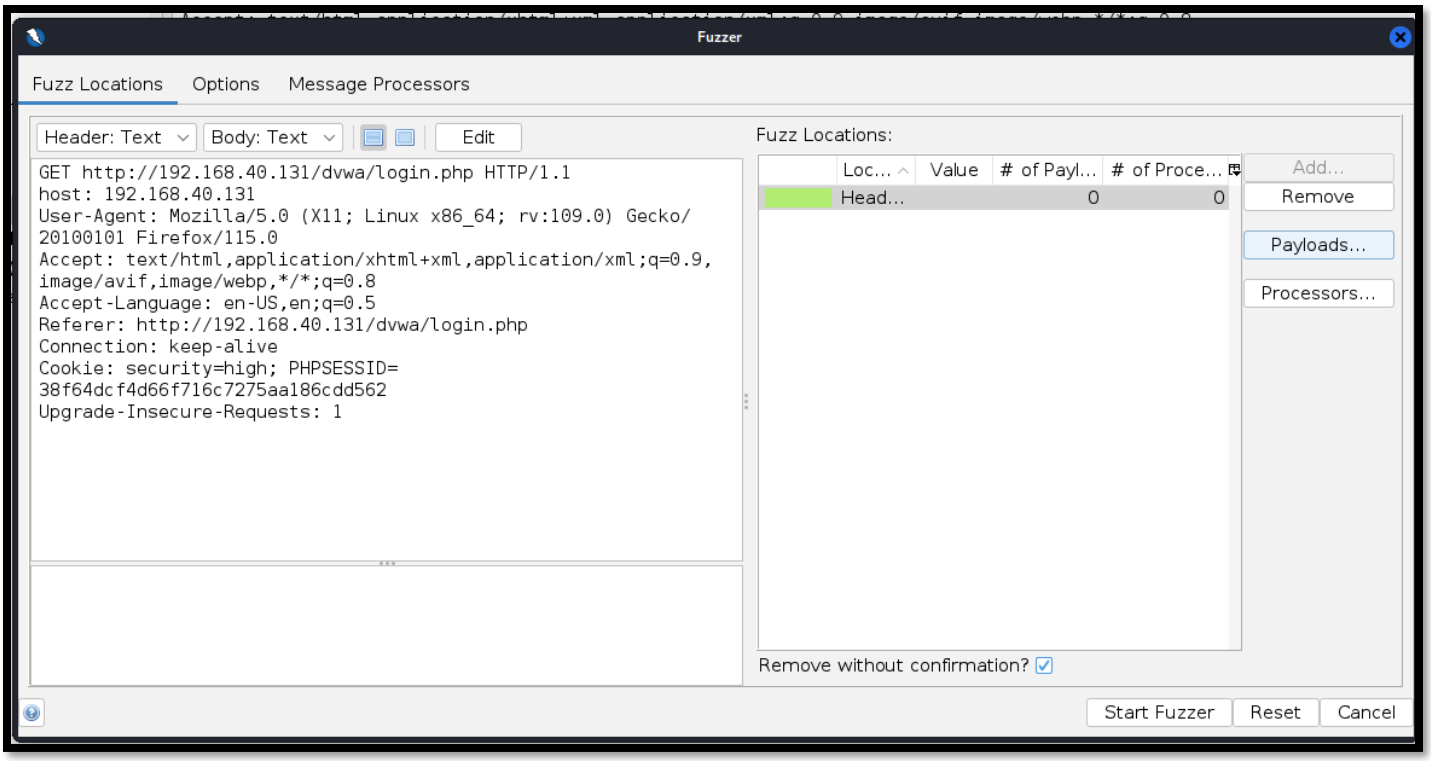


We paste DVWA url path to zap with automated scan option, We clicked on start attack , after attacking completed, we went to sites on left side under the sites we found get login.php, here we clicked on request tab .

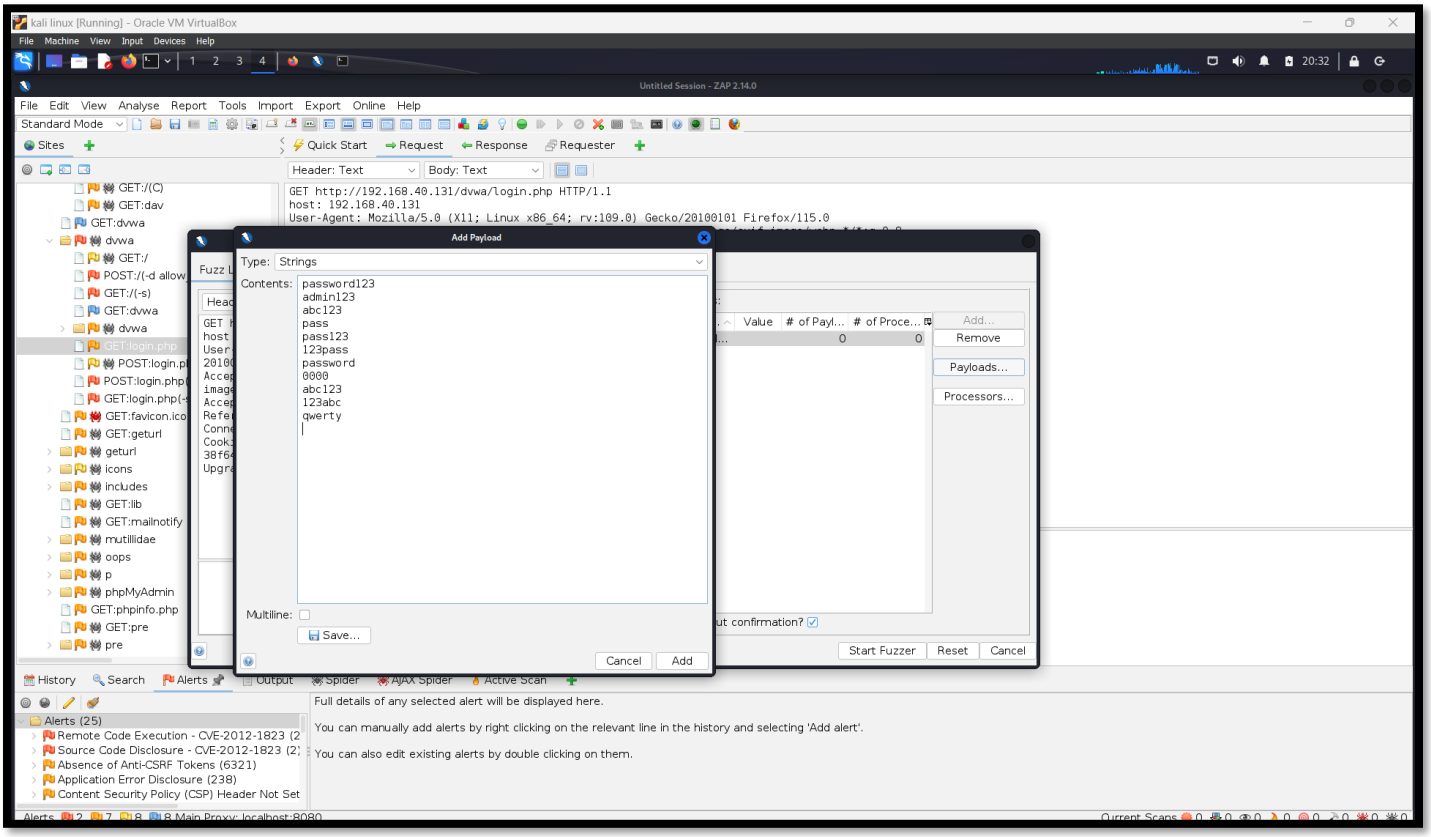


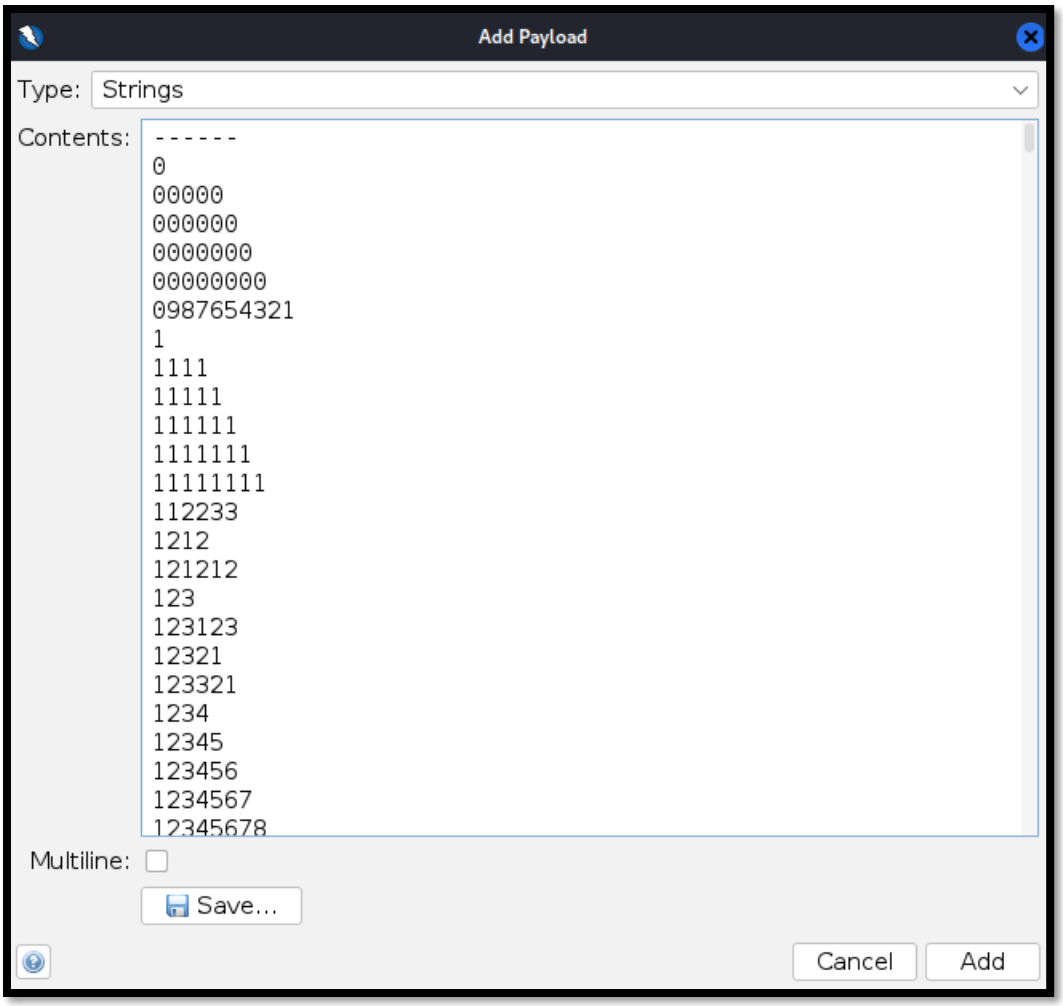
We tried logging in with the given credentials and then right-clicked the page to get to the FUZZ option. We chose fuzzing because it is a security testing technique used to find vulnerabilities in software and code errors. In this case, we used FUZZ to find vulnerabilities in the system. This is done by fuzzing, which involves providing huge amounts of data—often in the form of incorrect or unexpected inputs to test the system for vulnerabilities.



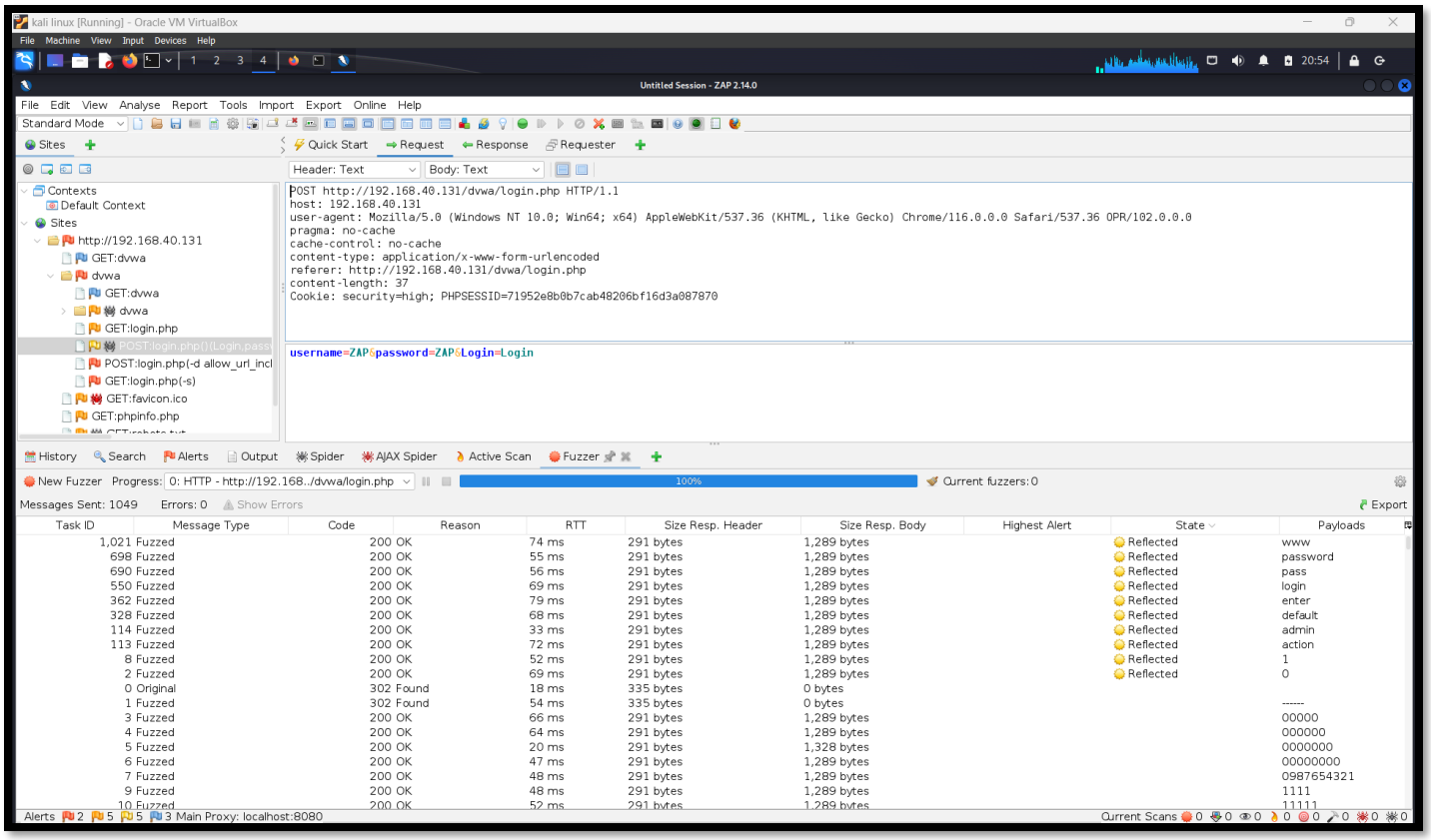


Then we click on payload button where we add many commonly used password copied from econestoga best1050.txt file in our course content.

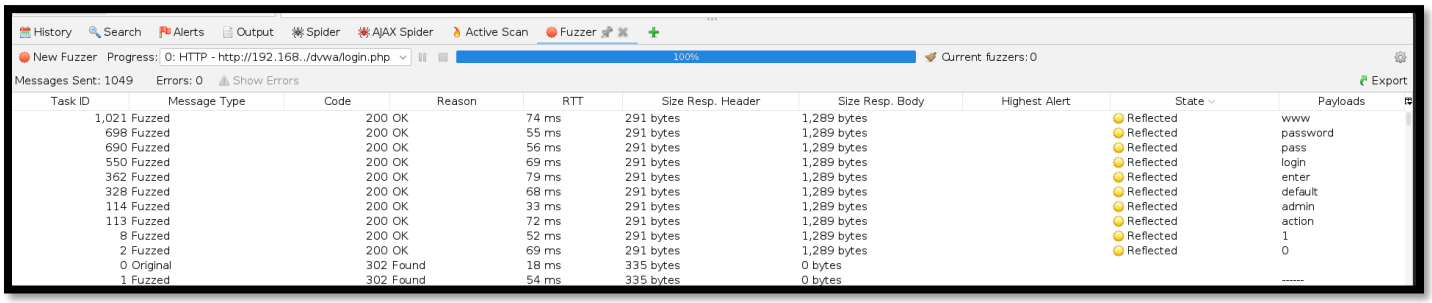




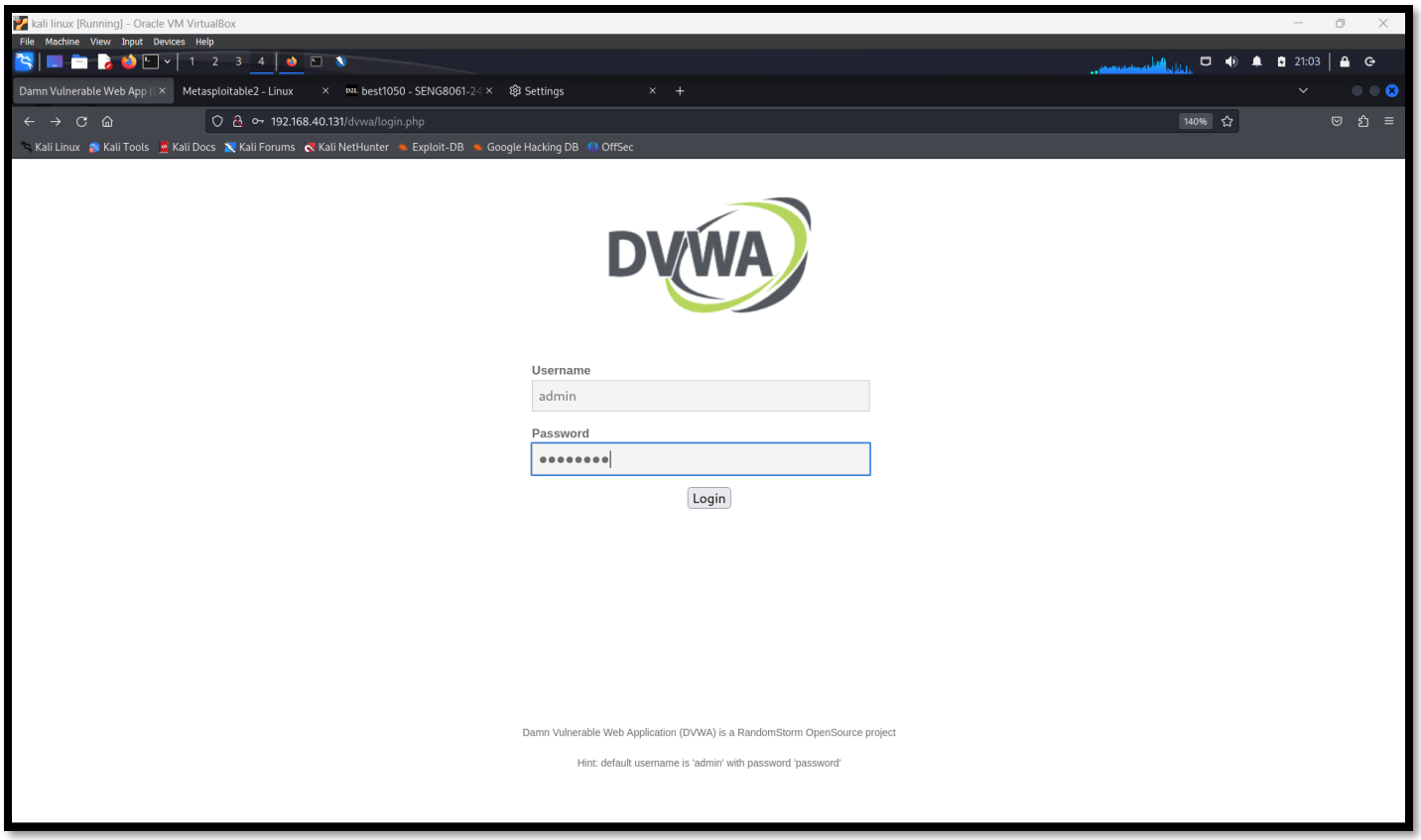
Then we click ok button, then we click on start fuzzer button. Using OWASP Zap's Fuzzer, we discovered that the page responded to a variety of password strings. We went further and logged into the application to confirm the authenticity of these passwords.

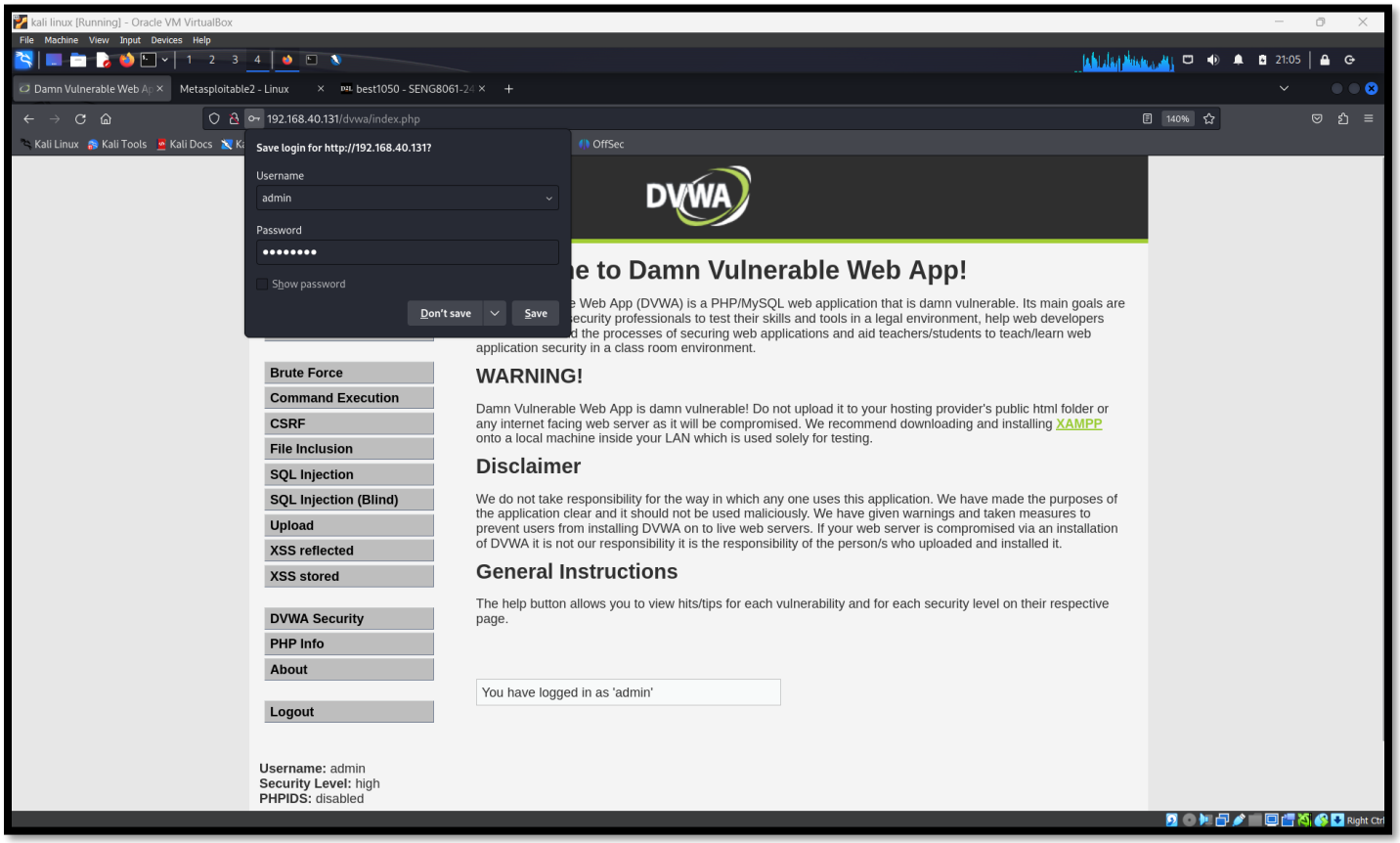


After completing fuzzer, we filter out state by clicking on it, we got some response.



we use those response as our input to admin password.





Finally, we have successfully bypassed the authentication and logged in as admin with password = password., without any secure authorization.

This vulnerability shows that there are weak authentication processes in the DVWA application, which gives attackers the chance to obtain unauthorized access through brute force attacks.

Recommendations/ Mitigations:

- **Put CAPTCHA into the application:** This will allow the application to authenticate that a human user is attempting to log in, preventing automated brute-force attacks.
- **Limit The Number of Logins:** To prevent attackers from continually guessing passwords, set a limit on the number of failed logins attempts within a specific time range.
- **Implement Two-Factor Authentication:** The security of the login process can be greatly improved by requiring an additional method of authentication, such as a one-time password given by SMS or produced by an authentication app such Microsoft Authenticator, Google Authenticator

Risk Analysis

For the DREAD analysis we have use rank criteria as below:

DREAD Ranking:

High (40-50),

Medium (30-39),

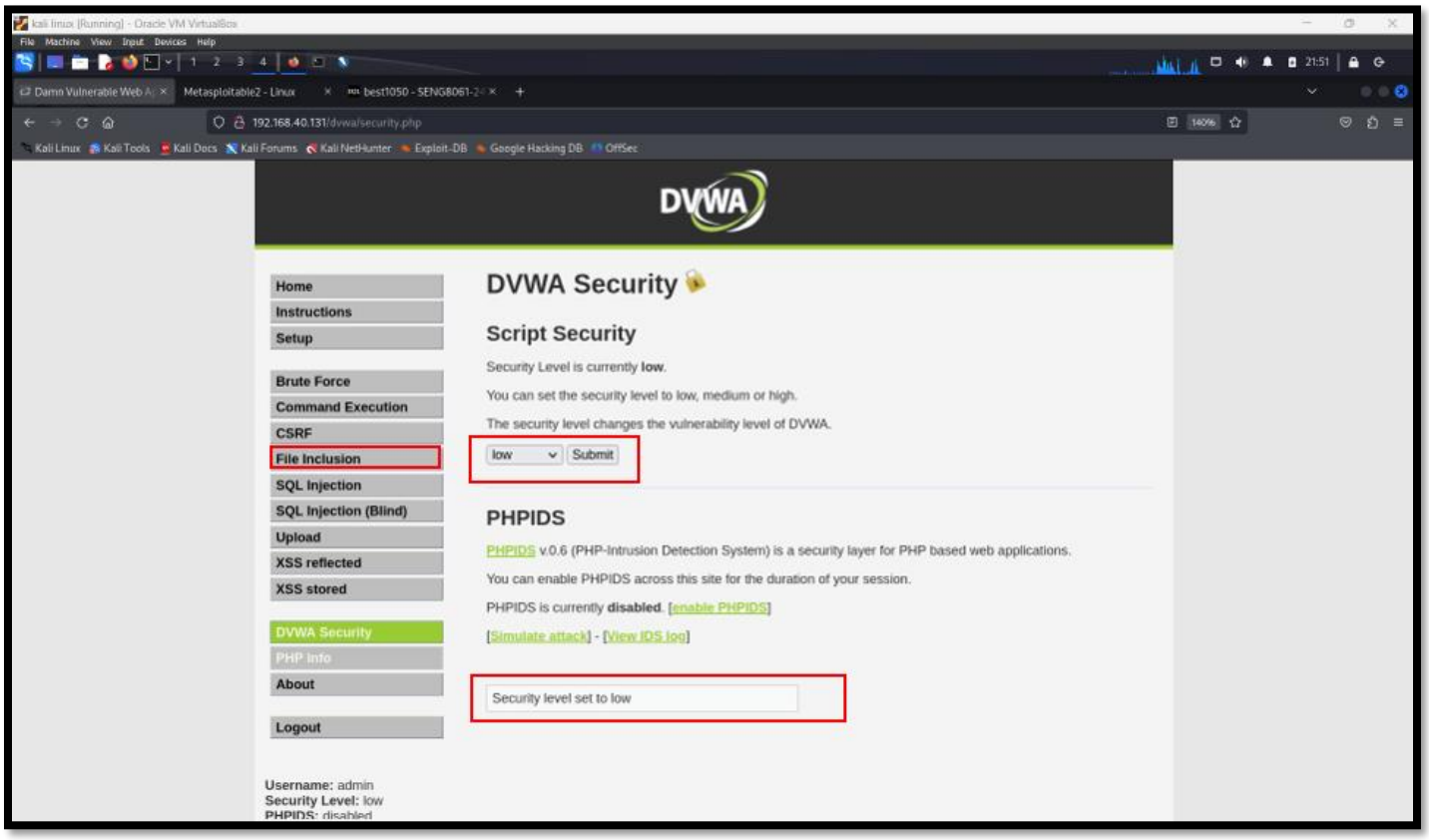
Low (20-29)

DREAD	Detail		Rank
Damage	This vulnerability can cause damage as it allows unauthorized access to critical areas of the application, ultimately leading to data breaches, modification of sensitive information, and unauthorized actions.		9/10
Reproducibility	The vulnerability appears to be highly reproducible because it is built within the application, allowing anyone to access it without authorization.		10/10
Exploitability	Exploiting this vulnerability is relatively easy, as demonstrated by using OWASP Zap to perform a brute force attack, which successfully bypassed the authentication mechanism.		10/10
Affected Users	This vulnerability potentially affects all users of the DVWA application since it compromises system security and allows unauthorized access.		10/10
Discoverability	This vulnerability is easy to find because it was found using basic testing and exploitation methods like brute force attacks using frequently used passwords.		10/10
DREAD SCORE	**(Risk Level: HIGH)**	Overall score	49/50

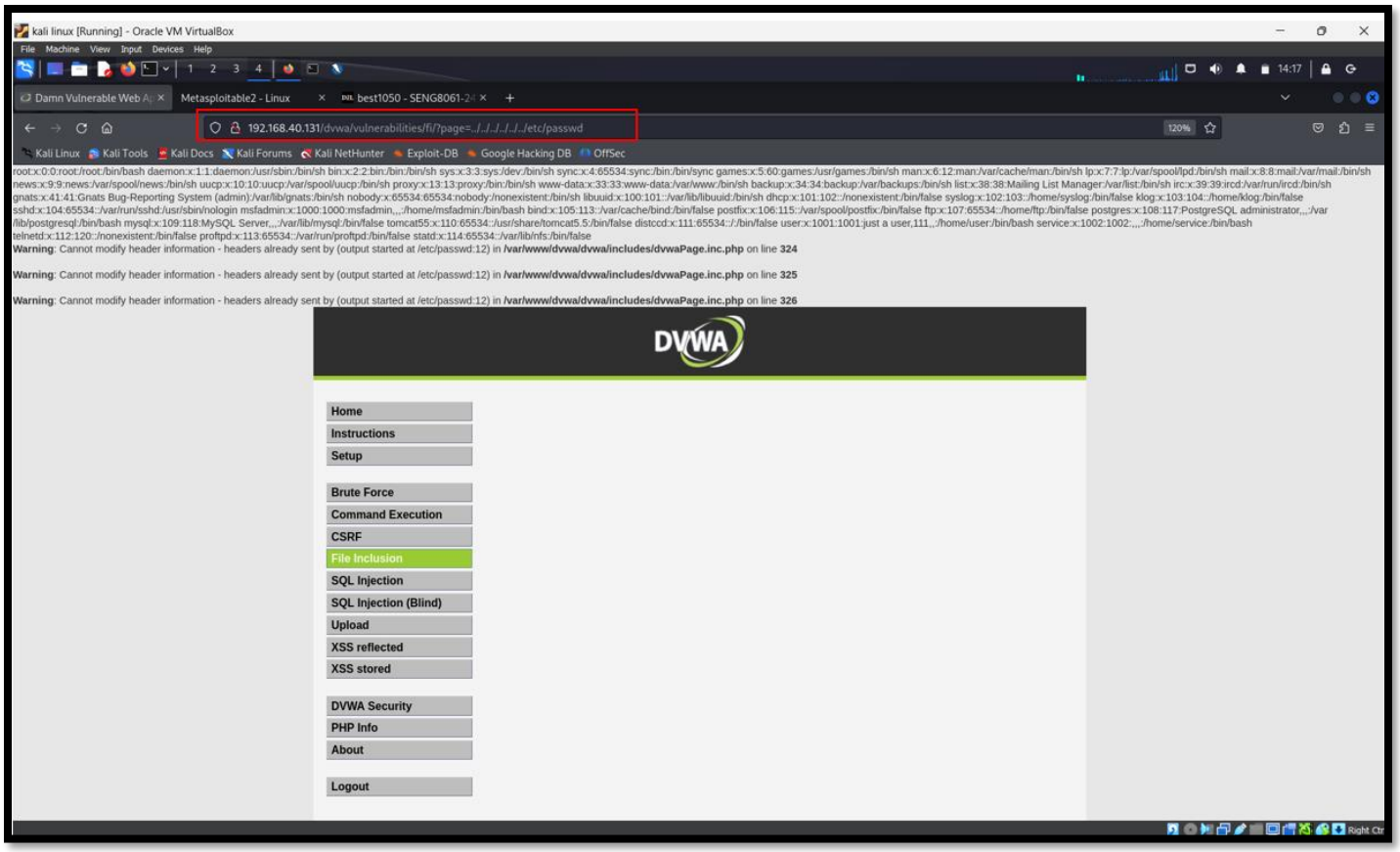
Vulnerability #2: File Inclusion

Discovered Vulnerabilities

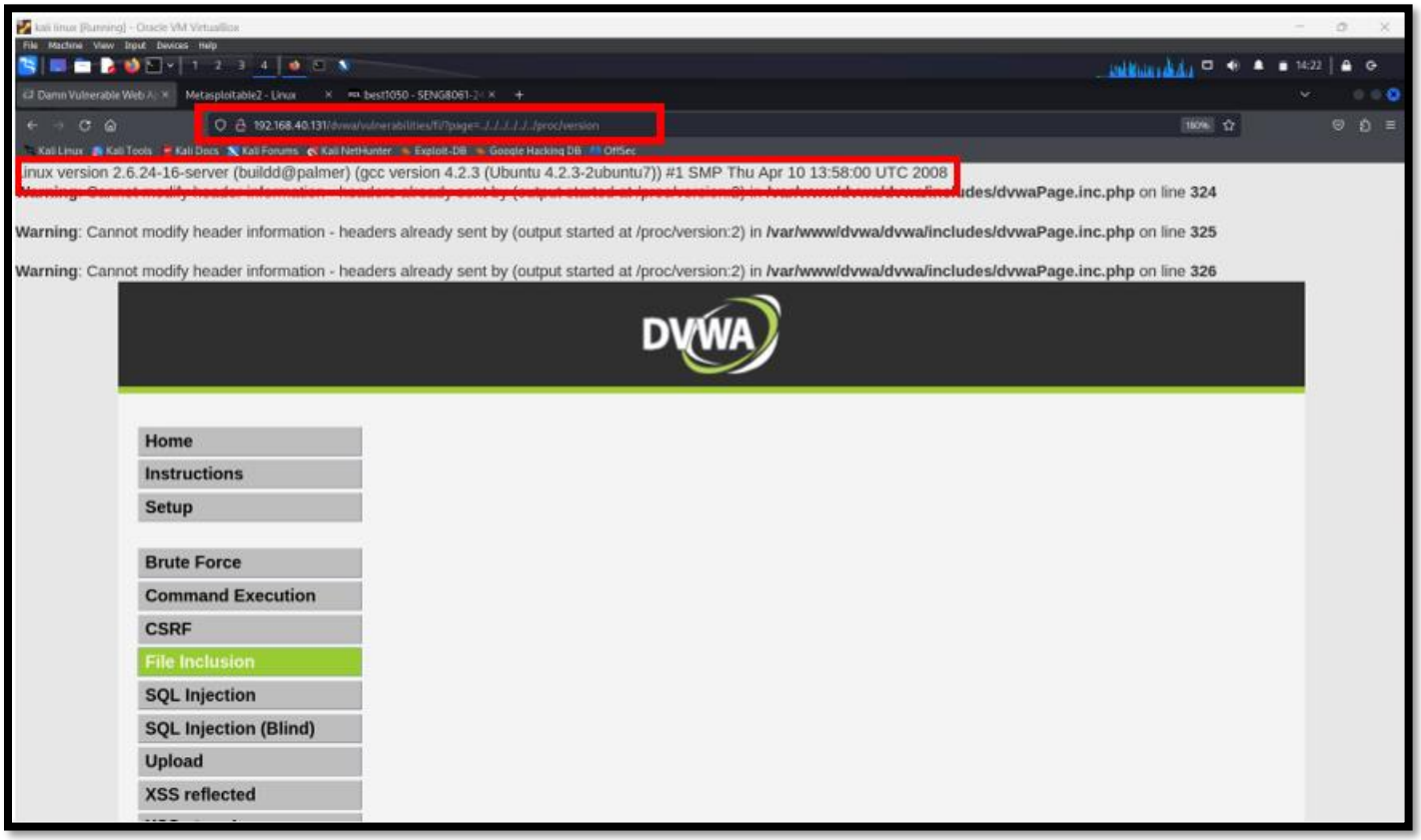
We logged in as admin to DVWA page, then we went to DVWA security, where we select 'LOW' as DVWA vulnerability level and submit the button. Then we went to file inclusion tab.



Then we typed on URL ../../../../../../etc/passwd which is useful for finding user's information in Linux, then we run the browser. Here we saw that the file included as a php script.



Then we typed change that url path to these path ../../../../../../proc/version , we saw find version of the web browser on the page.



Recommendations/ Mitigations:

- Validate and sanitize user input before using it to include files.
- Maintain a whitelist of acceptable inputs.
- When including files, use absolute paths rather than user-controlled relative paths.
- Ensure that the web server process has minimal permissions and can only access necessary files.

Risk Analysis

For the DREAD analysis we have use rank criteria as below:

DREAD Ranking:

High (40-50),

Medium (30-39),

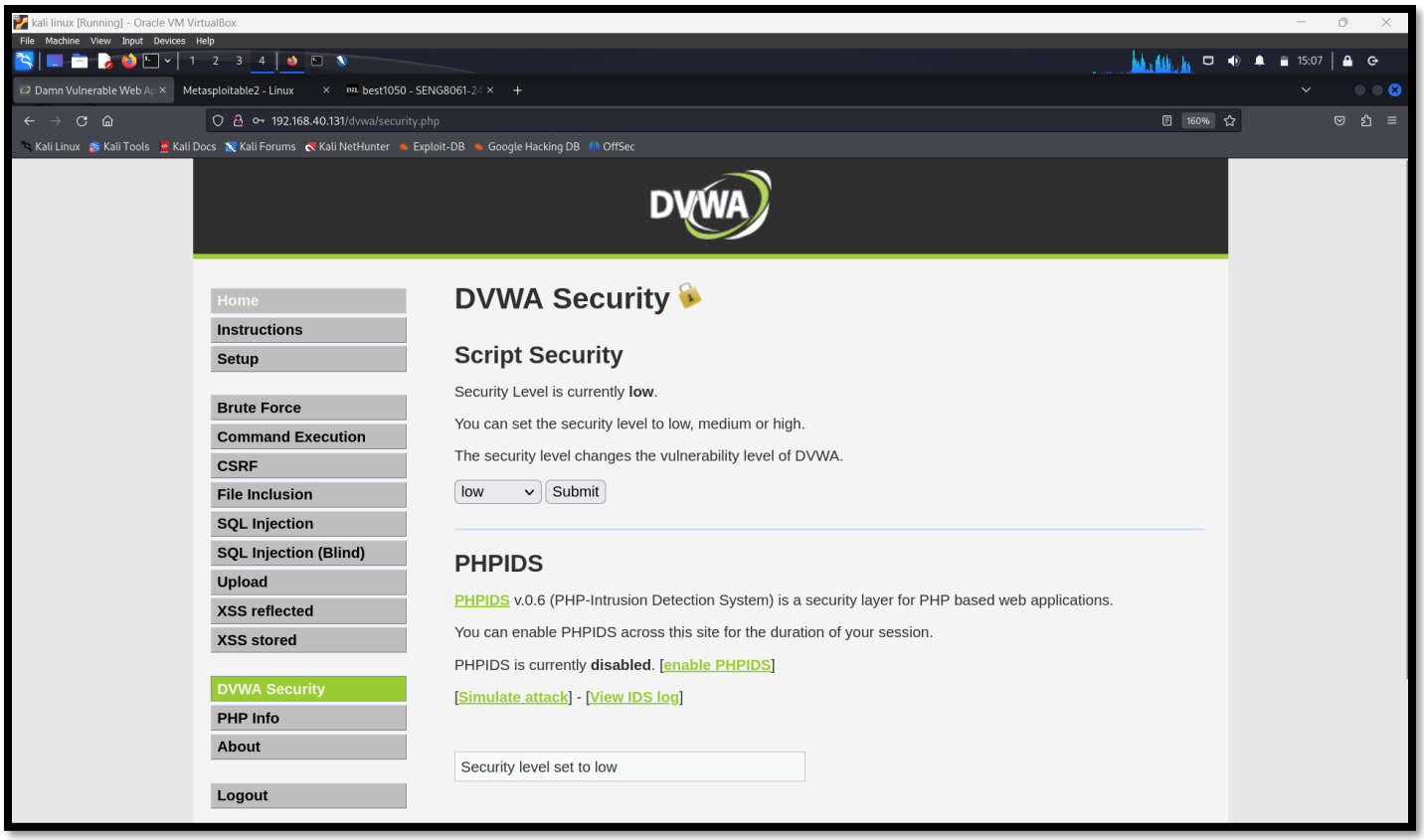
Low (20-29)

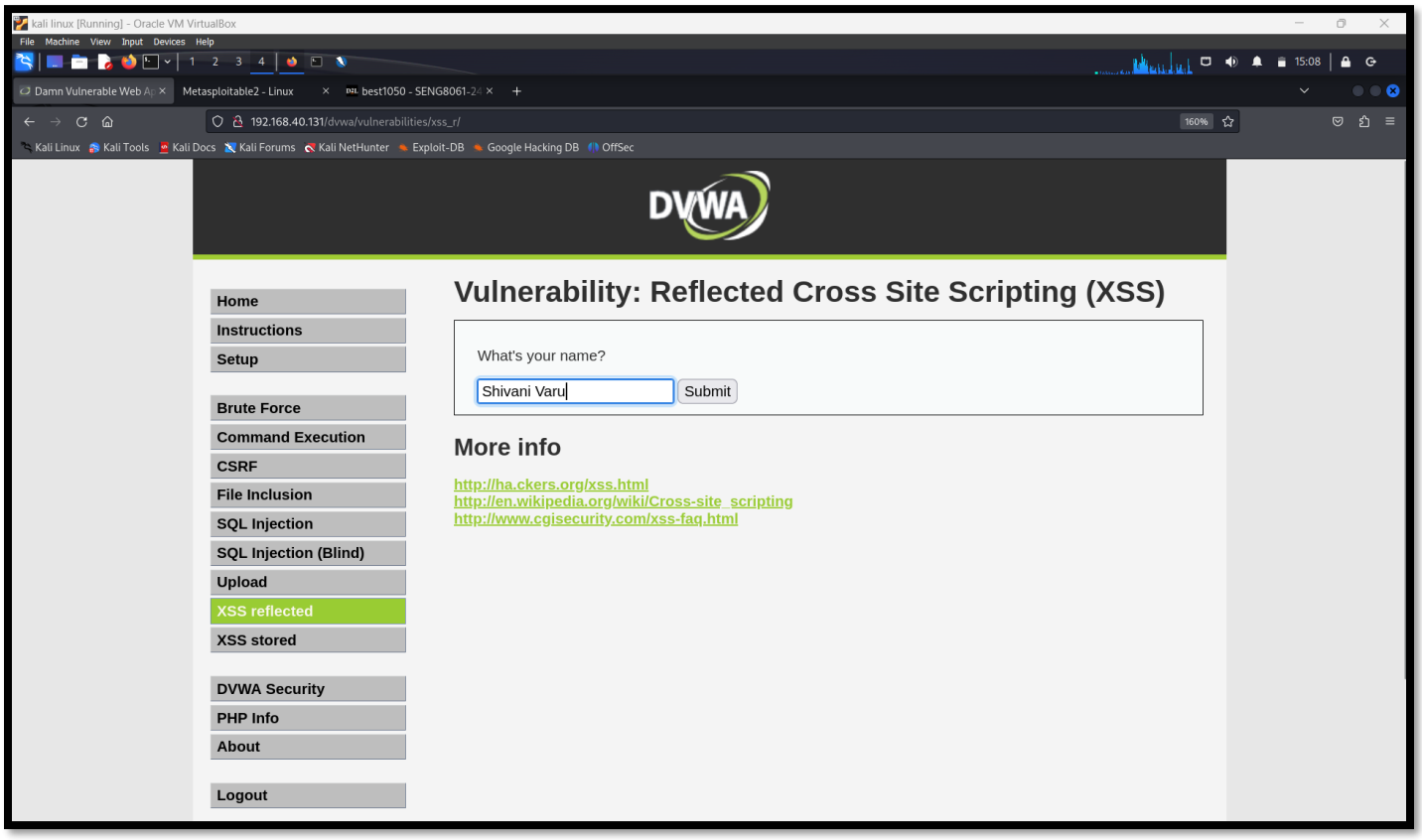
DREAD	Detail		Rank
Damage	The potential damage of this vulnerability is significant as it allows an attacker to access sensitive system files like /etc/passwd and /proc/version, which can reveal system and user information.		7/10
Reproducibility	The vulnerability appears to be highly reproducible as it can be exploited by manipulating the URL to include arbitrary files.		9/10
Exploitability	Exploiting this vulnerability seems relatively straightforward as demonstrated by accessing sensitive system files through URL manipulation.		7/10
Affected Users	All users accessing the vulnerable file inclusion feature are potentially affected, as the vulnerability allows attackers to access system files regardless of user authentication.		7/10
Discoverability	Discovering this vulnerability might require some knowledge about file system structure and URL manipulation techniques, but once understood, it's relatively easy to exploit.		9/10
DREAD SCORE	**(Risk Level: Medium)**	Overall score	39/50

Vulnerability #3: Cross-Site Scripting (XSS) Reflection

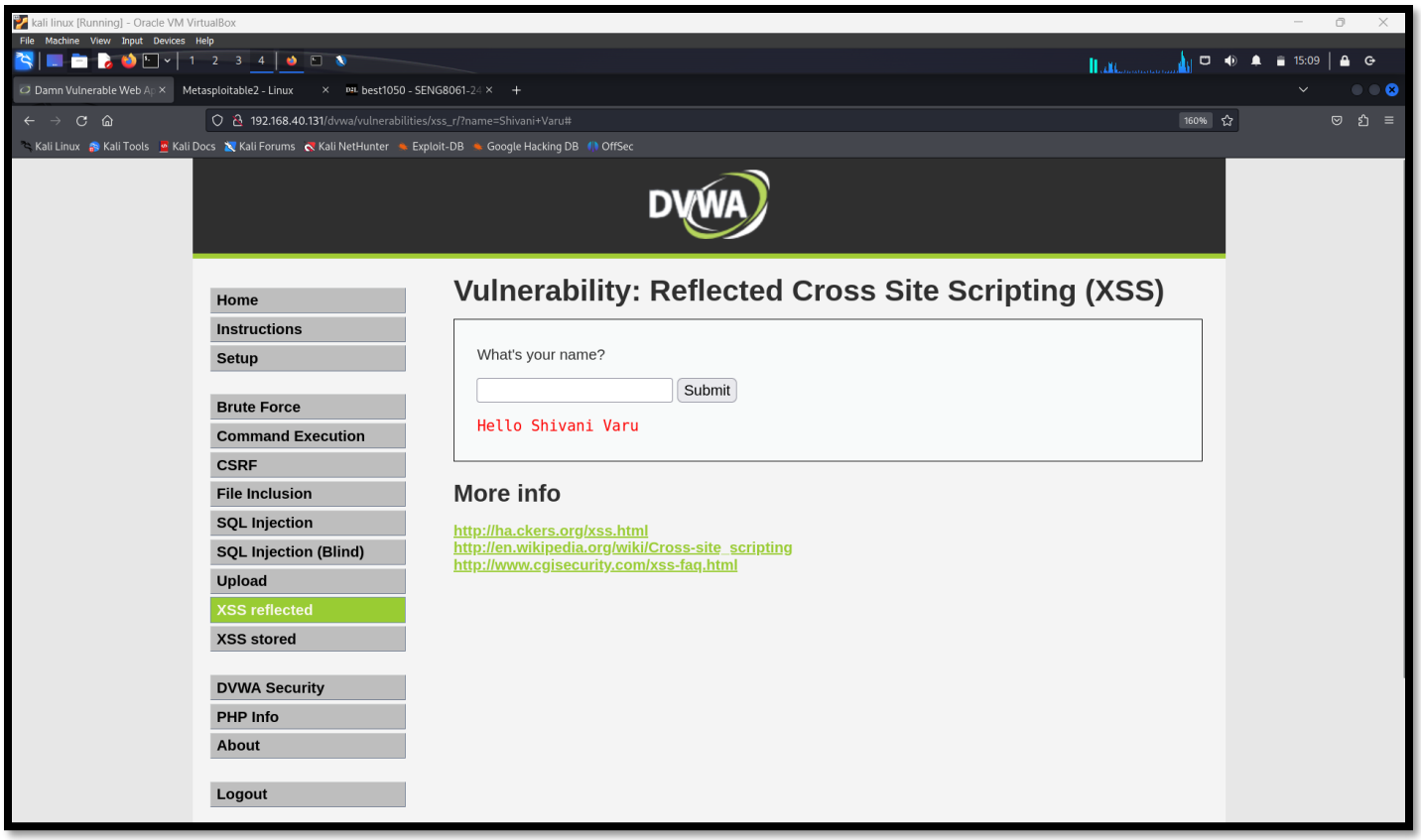
Discovered Vulnerabilities

We opened the DVWA site and set the difficulty level to low from the DVWA security tab.

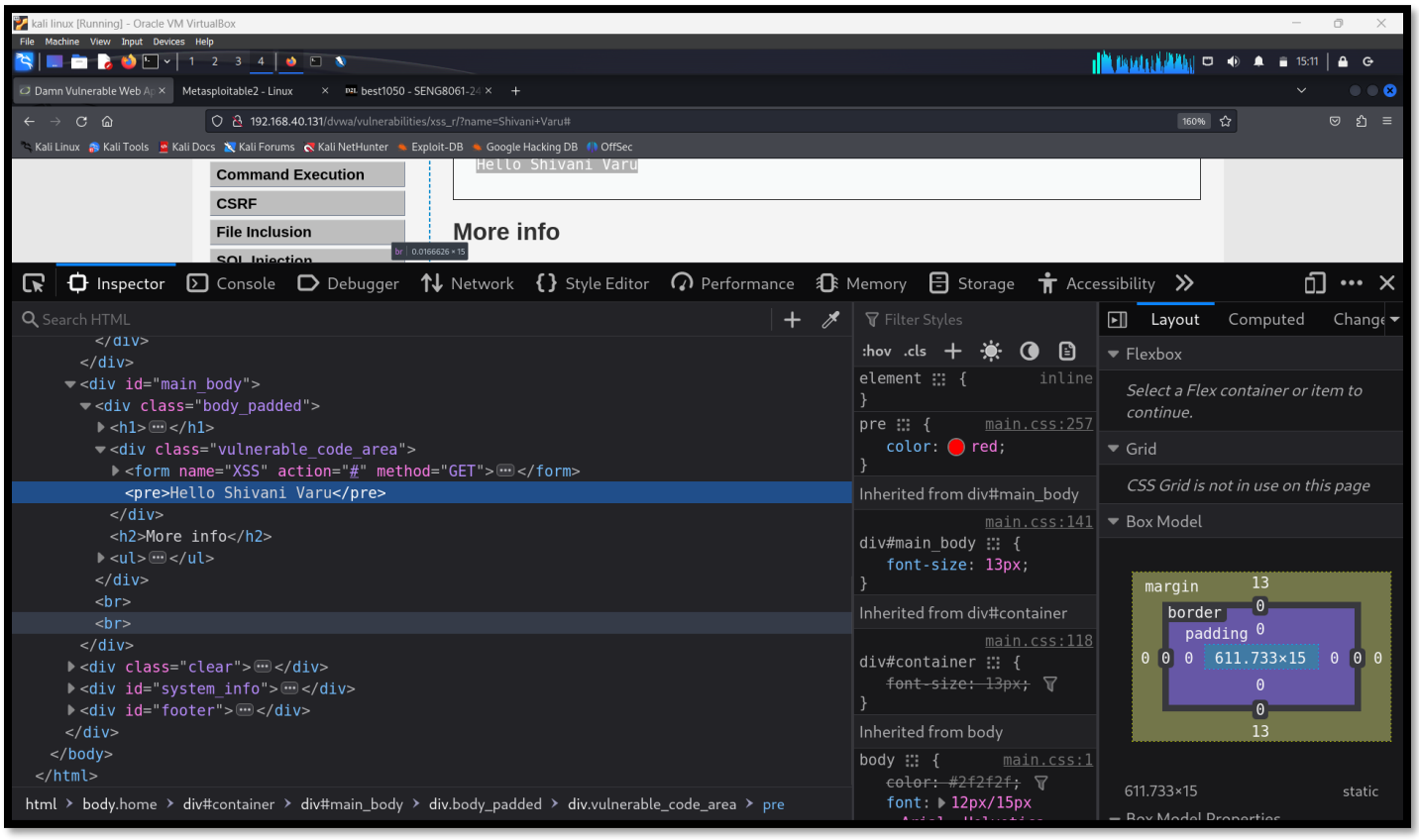




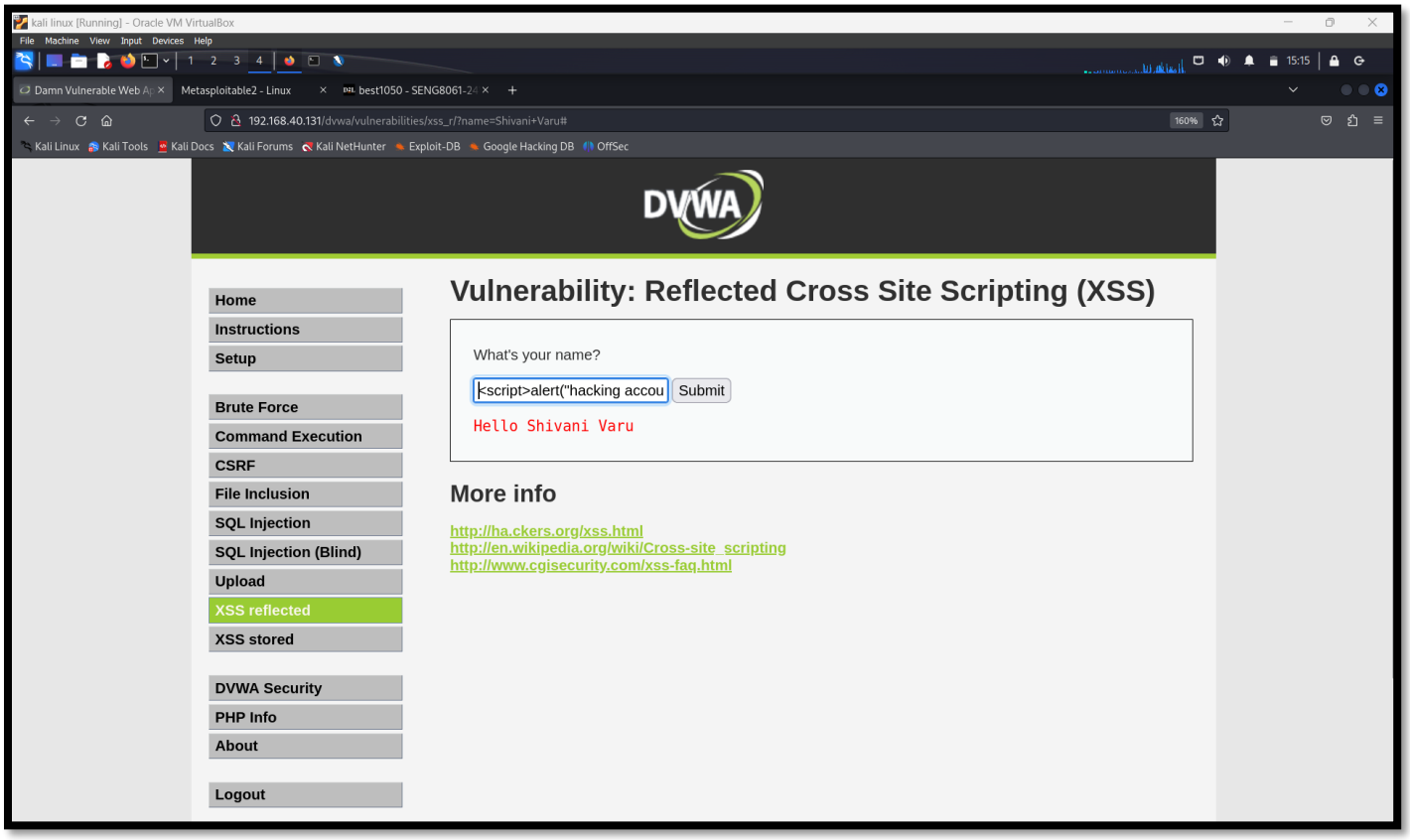
Then, we entered some value in the search field and clicked on submit.



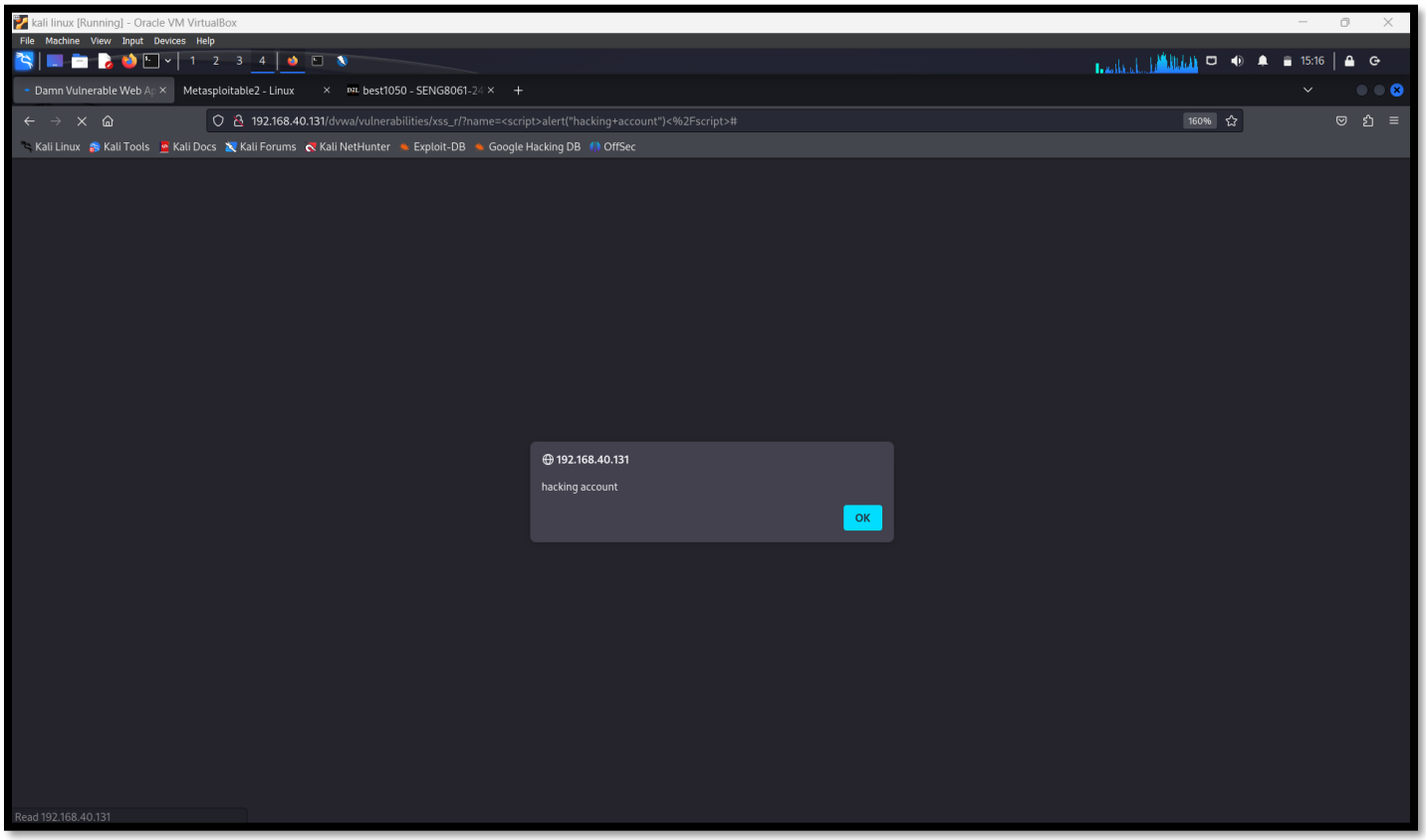
We saw the entered value in the source page.



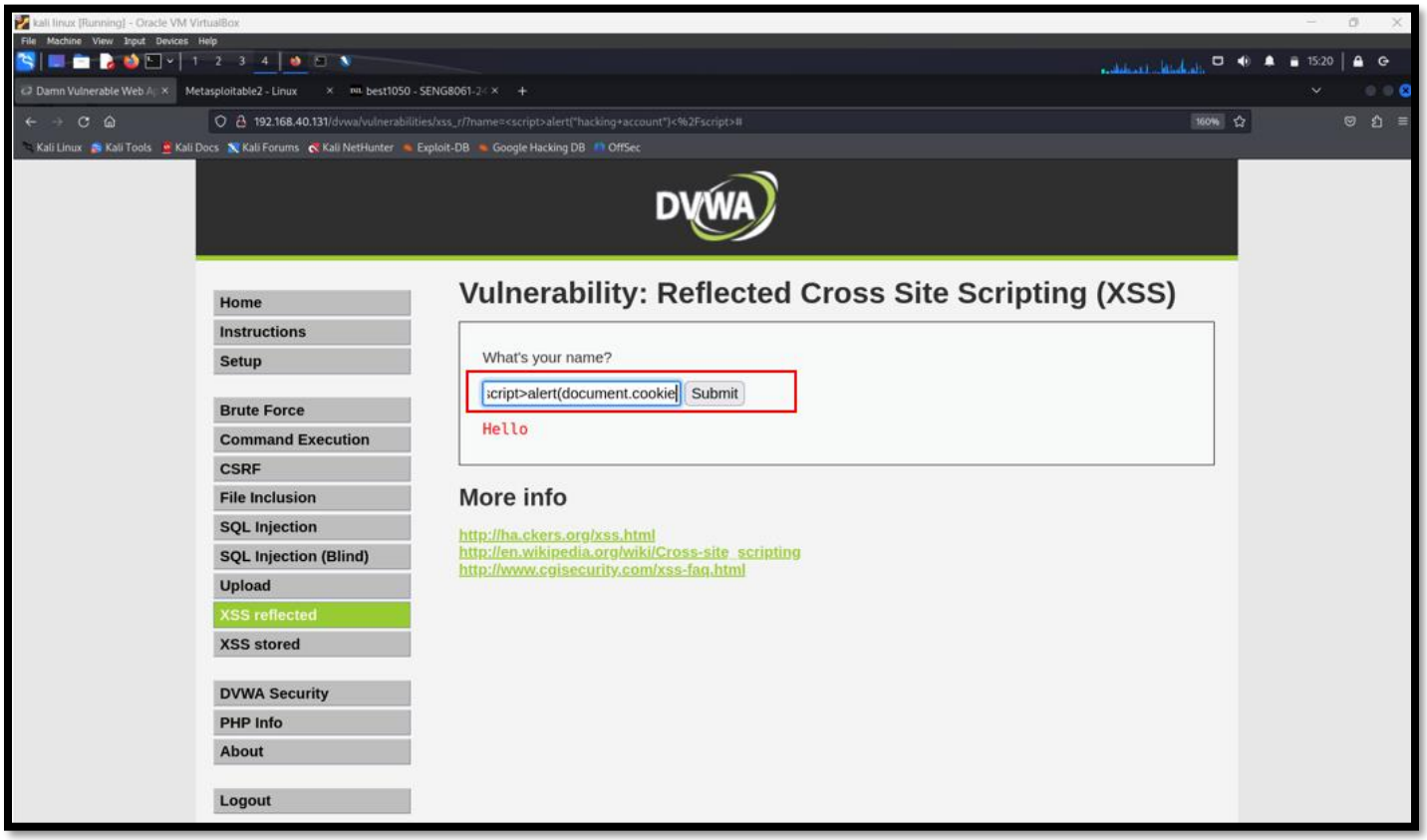
We provided the script <script> alert("hacking account") </script> in the field and clicked on submit.



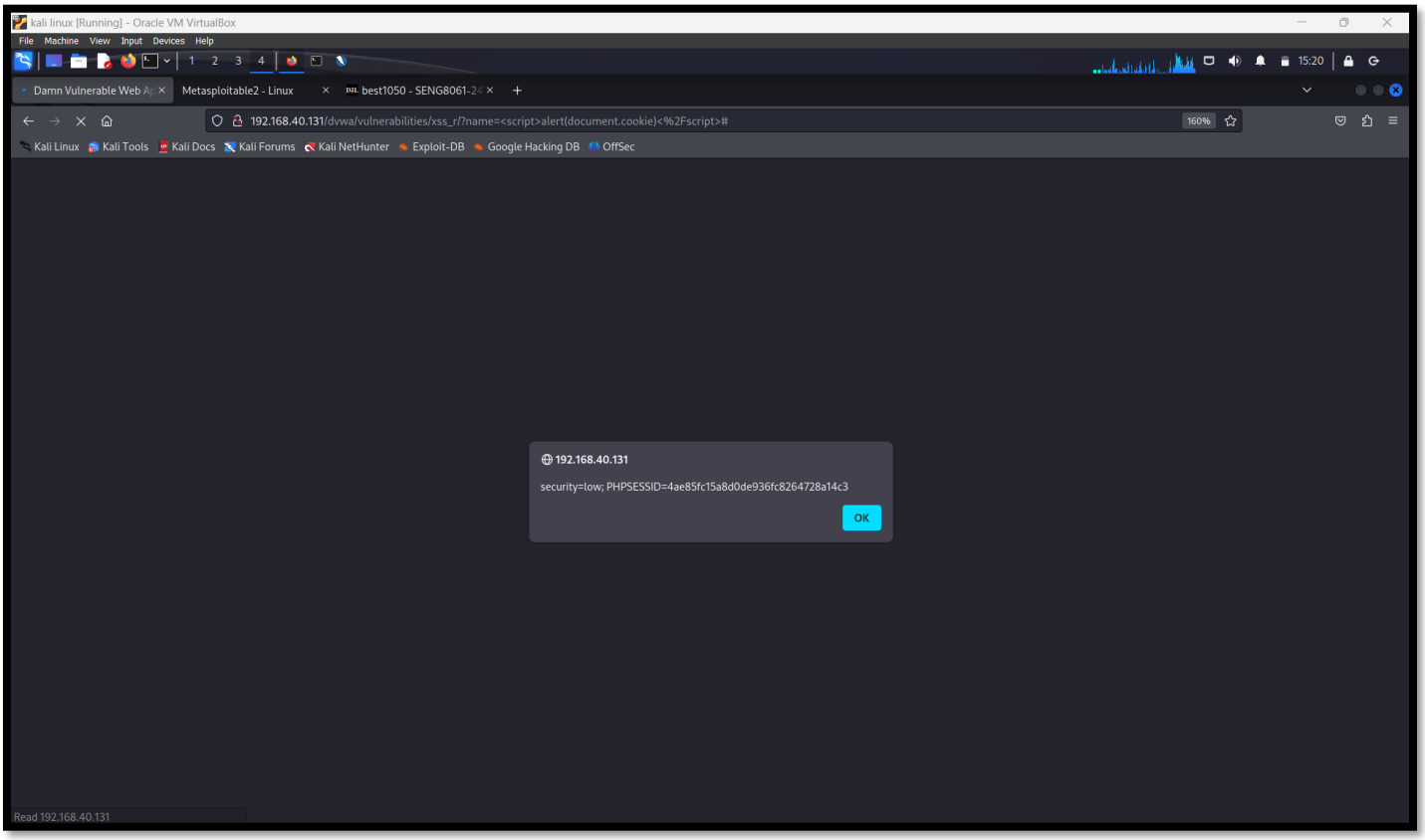
As a result, a black pop-up with the given information appeared, representing exploitation.



Then, we also provided the script <script> alert(document.cookie) </script> to find the session ID.



We observed that the session ID was retrieved. We can see the session ID is retrieved.



Recommendations/ Mitigations:

- Implement strict input validation and sanitization to ensure that user-supplied data is properly filtered, removing or encoding any potentially dangerous characters or script tags.
- Use output encoding when returning user input in HTML to ensure that any potentially malicious characters are properly encoded.
- Implement a Content Security Policy (CSP) to restrict the sources from which scripts can be loaded.
- Use HTTPOnly cookies to prevent scripts from accessing sensitive cookie data.

Risk Analysis

For the DREAD analysis we have use rank criteria as below:

DREAD Ranking:

High (40-50),

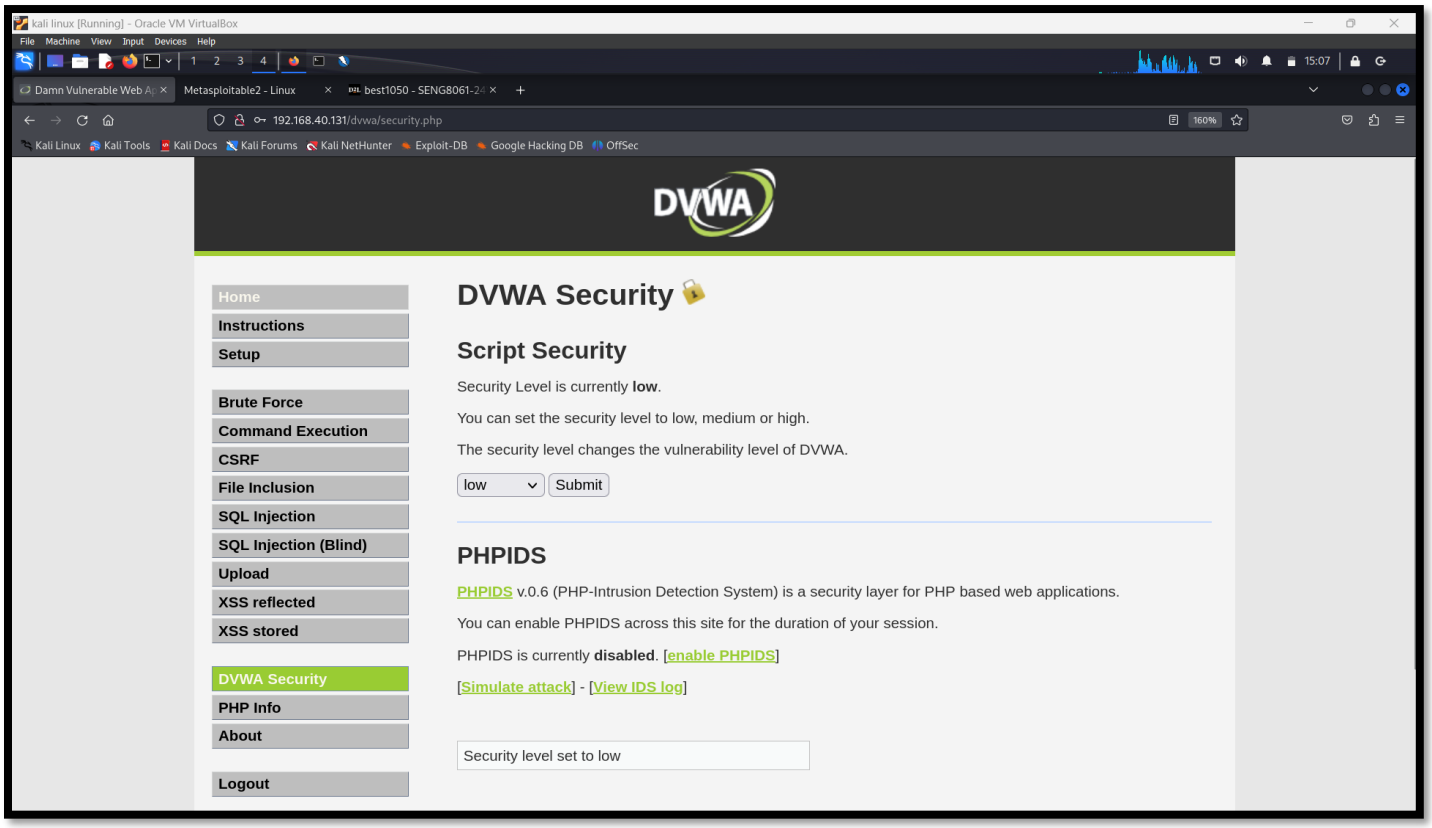
Medium (30-39),

Low (20-29)

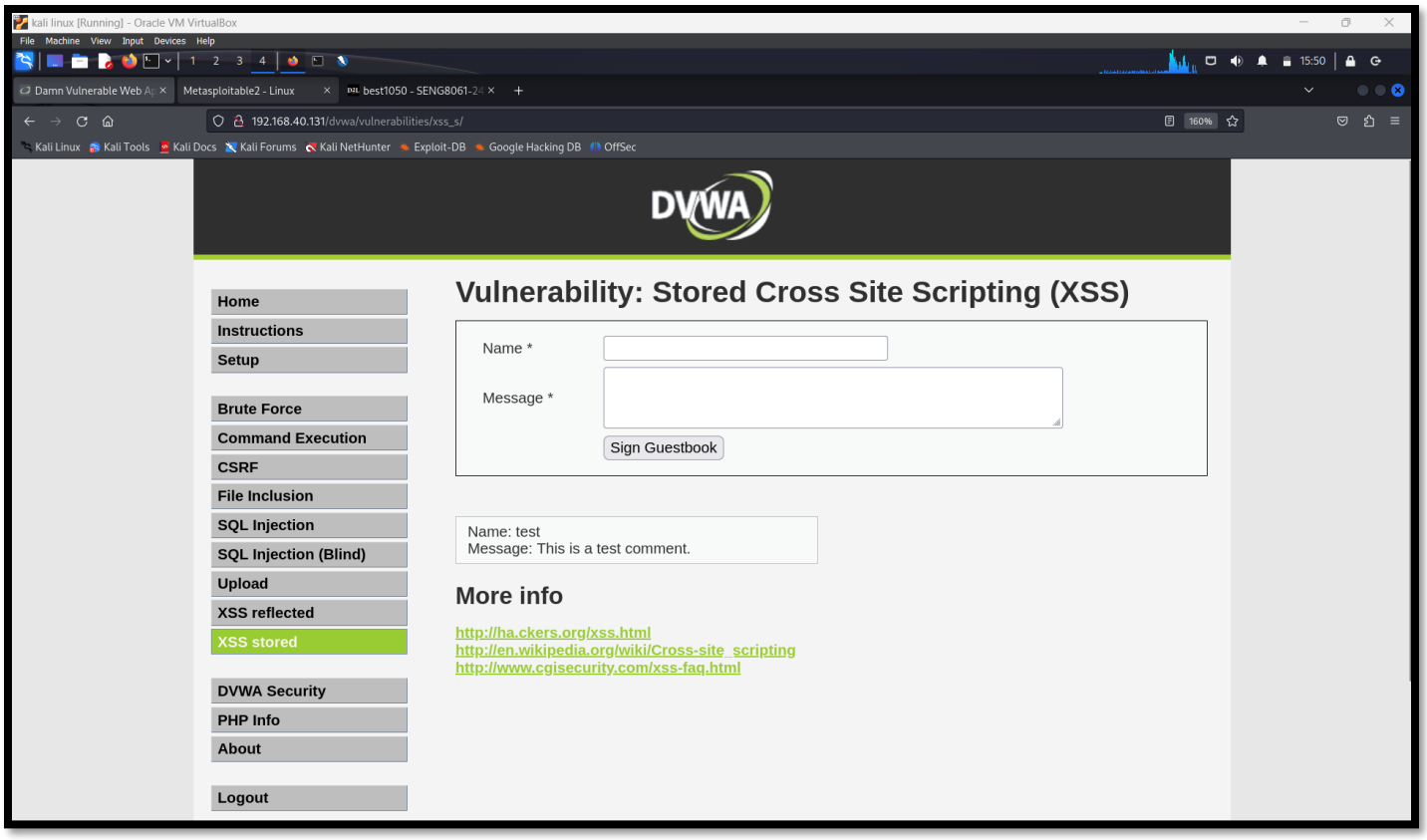
DREAD	Detail		Rank
Damage	An attacker could be able to run arbitrary scripts within the sessions of other users, which makes the XSS vulnerability extremely dangerous. This could result in various consequences, such as stealing sensitive information (such as session IDs or cookies), performing on the user's behalf, or damaging the website.		9/10
Reproducibility	The vulnerability is highly reproducible, as it appears to occur regularly whenever user-supplied data is sent to the web page without enough validation and sanitization.		10/10
Exploitability	Simple scripts such as alert boxes can be injected to exploit this vulnerability, which appears to be quite straightforward. It can be used for more malicious activities in the future.		9/10
Affected Users	All users who interact with the vulnerable function can be affected, especially if they provide input that is reflected, they return to the page.		8/10
Discoverability	This vulnerability appears to be very straightforward to discover, having been discovered through manual testing by entering scripts into the input field and observing the results.		8/10
DREAD SCORE	**(Risk Level: High)**	Overall score	44/50

Vulnerability #4: Cross-Site Scripting (XSS) Stored

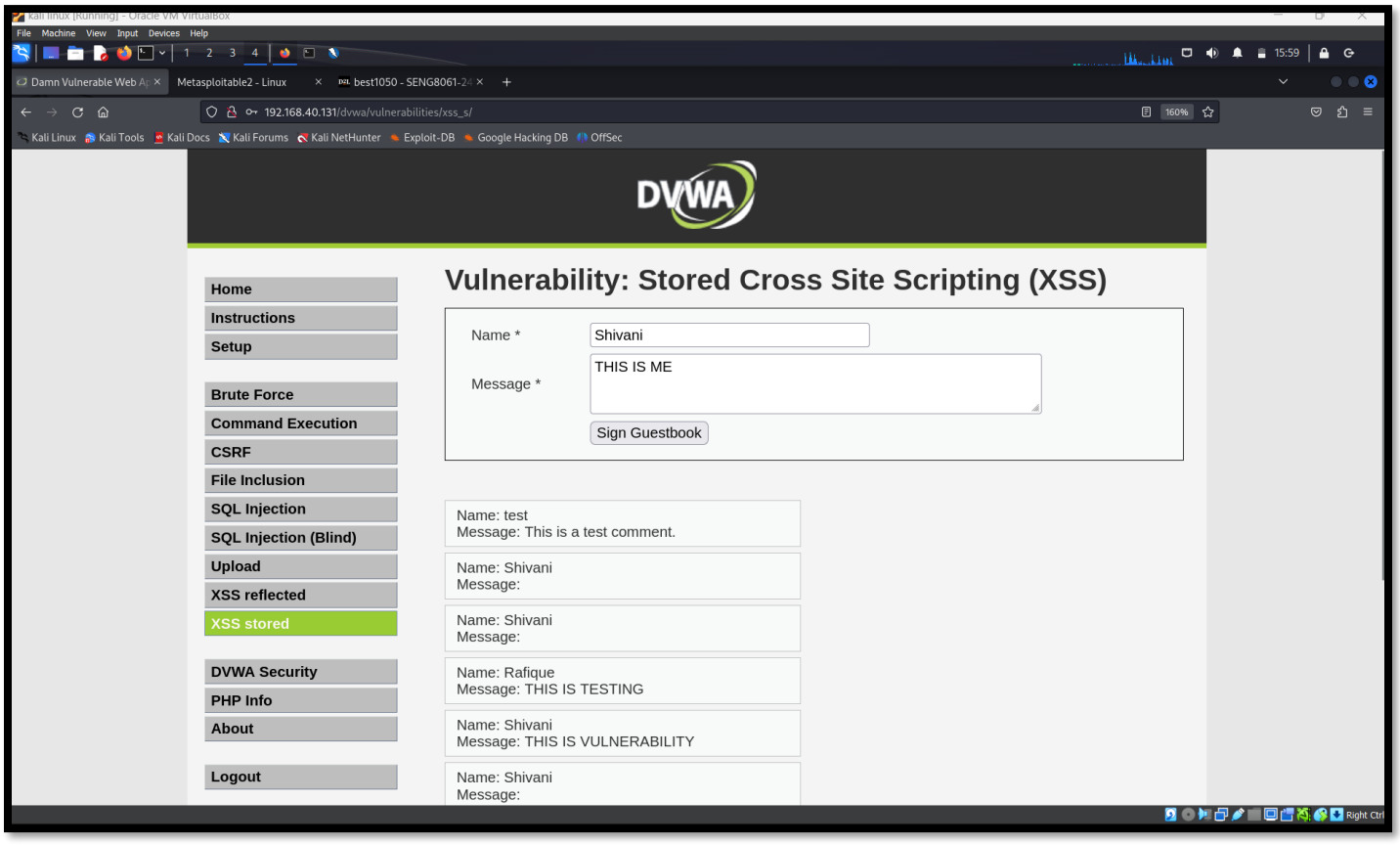
We opened the DVWA site and set the difficulty level to low.



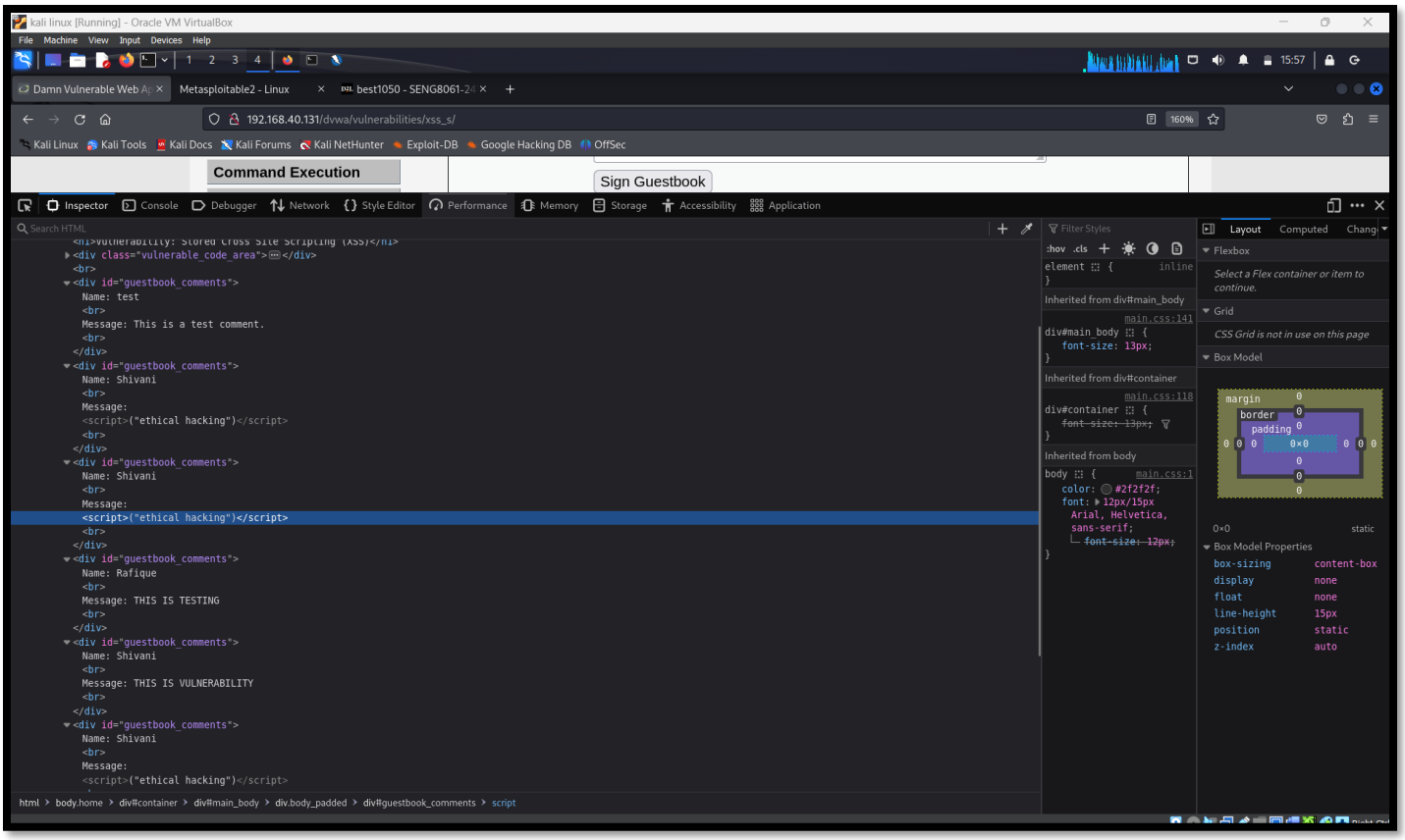
After that,we went to XSS stored from navigation bar,



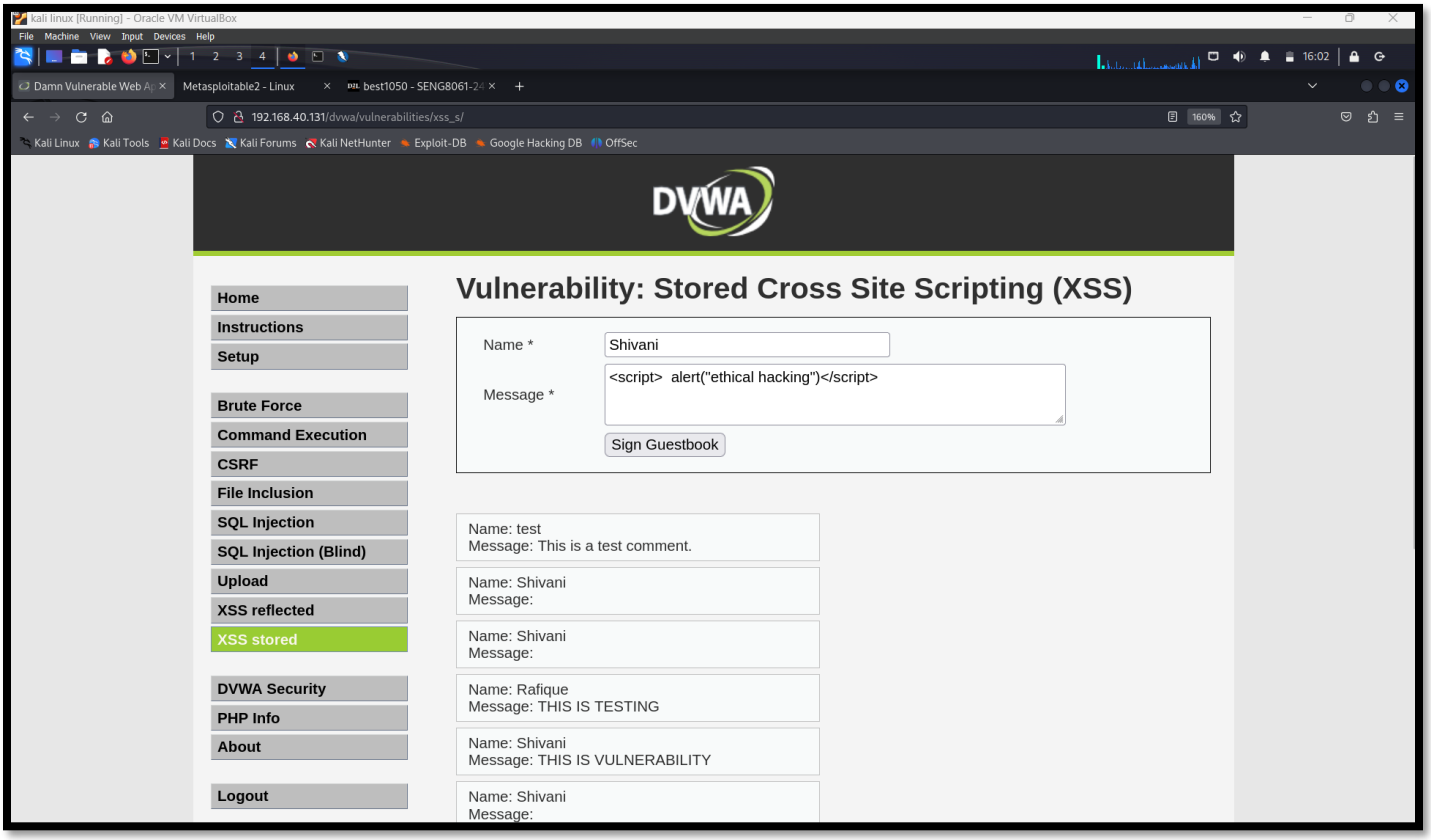
where we entered some values in the name and message fields and clicked on submit.



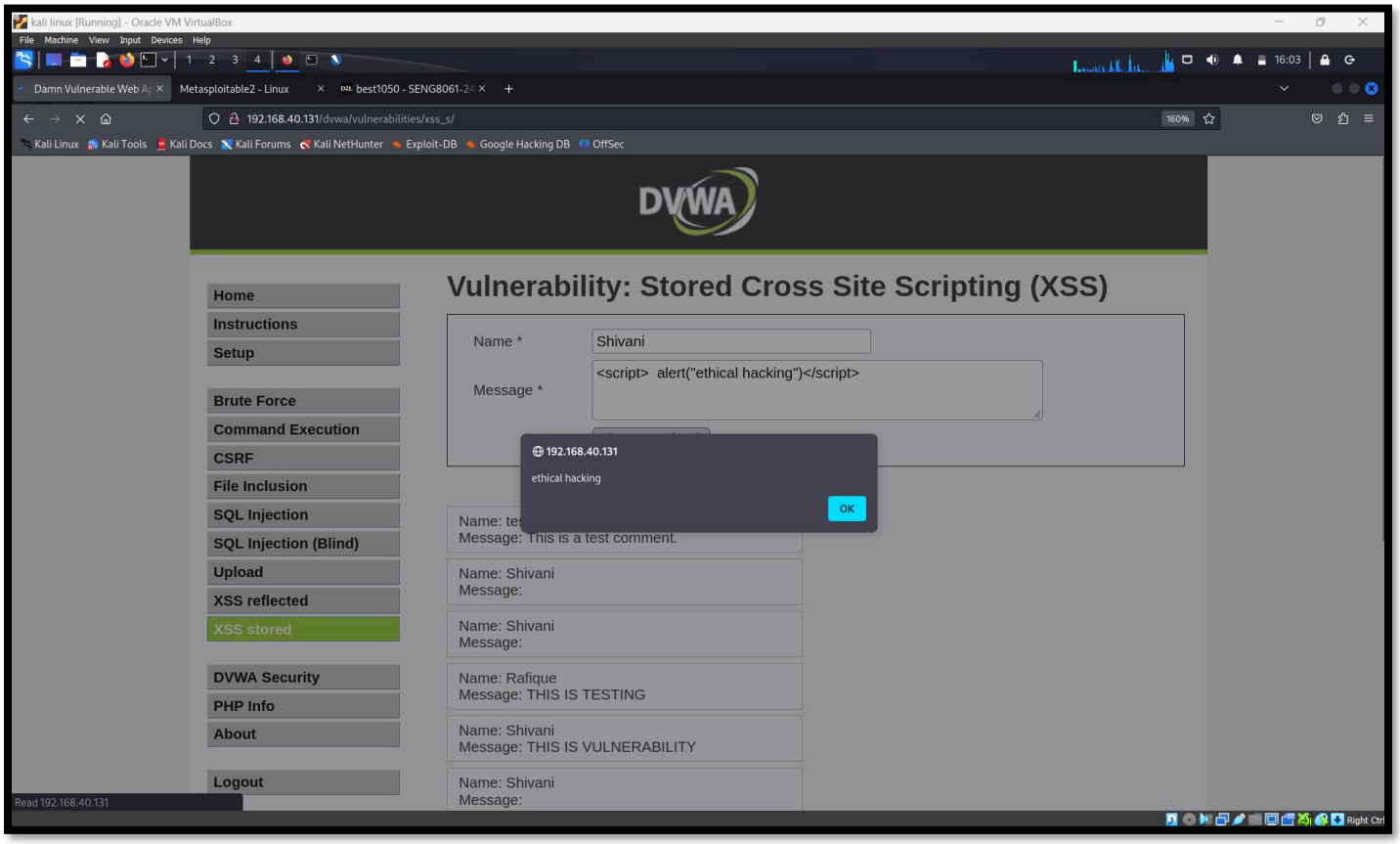
Upon inspecting the source code, we confirmed that the entered data was indeed present.



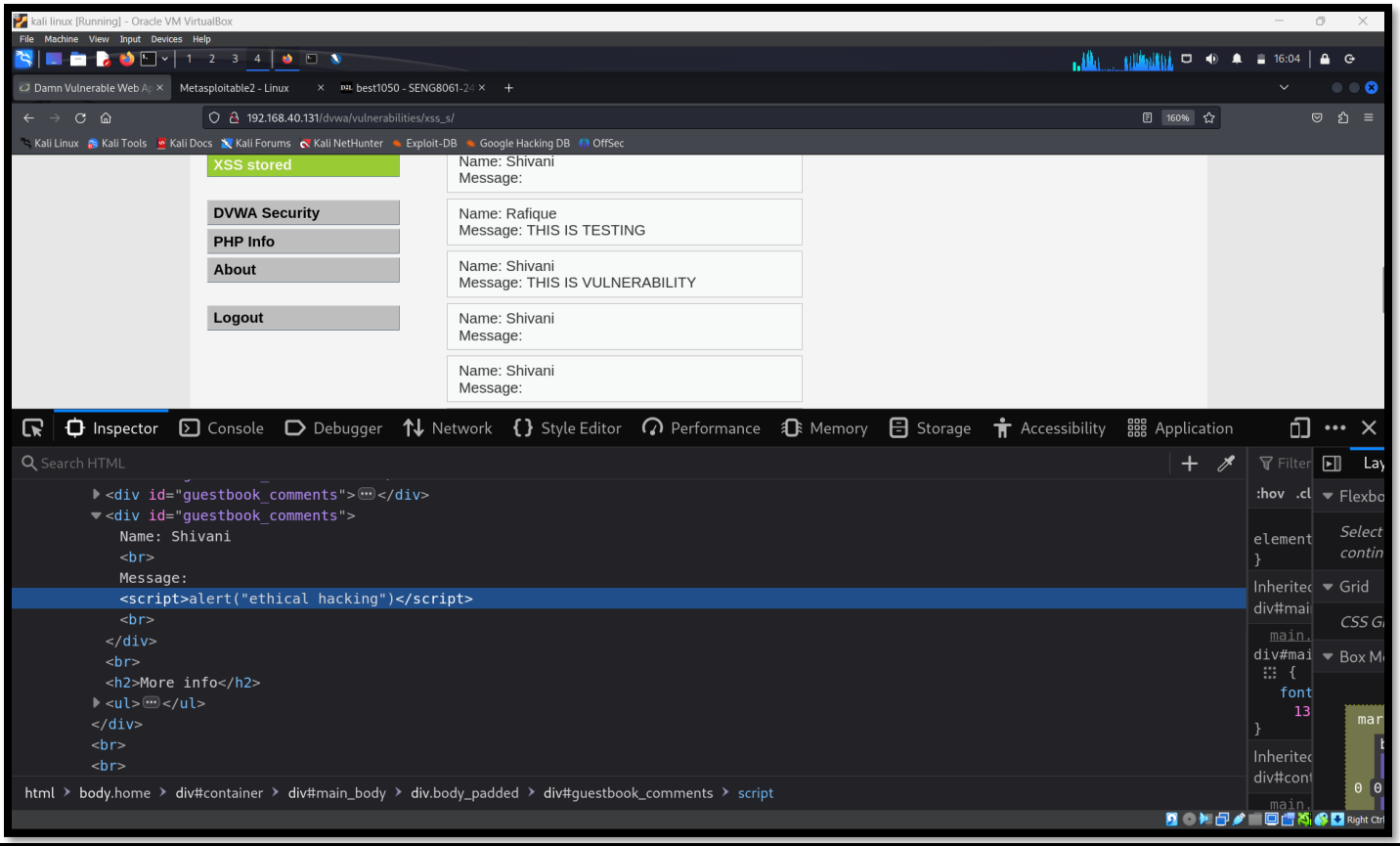
Subsequently, we entered the script <script> alert("ethical hacking") </script> in the message field and clicked on submit.



Instantly, a pop-up emerged, indicating that exploitation had been successful.



Furthermore, upon examining the source code, we could see that the script we entered was indeed present in the file.



Recommendation/Mitigation:

- Organization should always encrypt the output our application sends to a browser when it uses user input so that the browser interprets it as data rather than as HTML or JavaScript code. This stops the execution of any script.
- When a user types <script> alert("ethical hacking") </script>, for example, the browser should display the text exactly as it was typed, without interpreting it as a script.
- Encoding converts HTML characters with meanings into character entities, which the browser displays as characters rather than code.

Risk Analysis

For the DREAD analysis we have use rank criteria as below:

DREAD Ranking:

High (40-50),

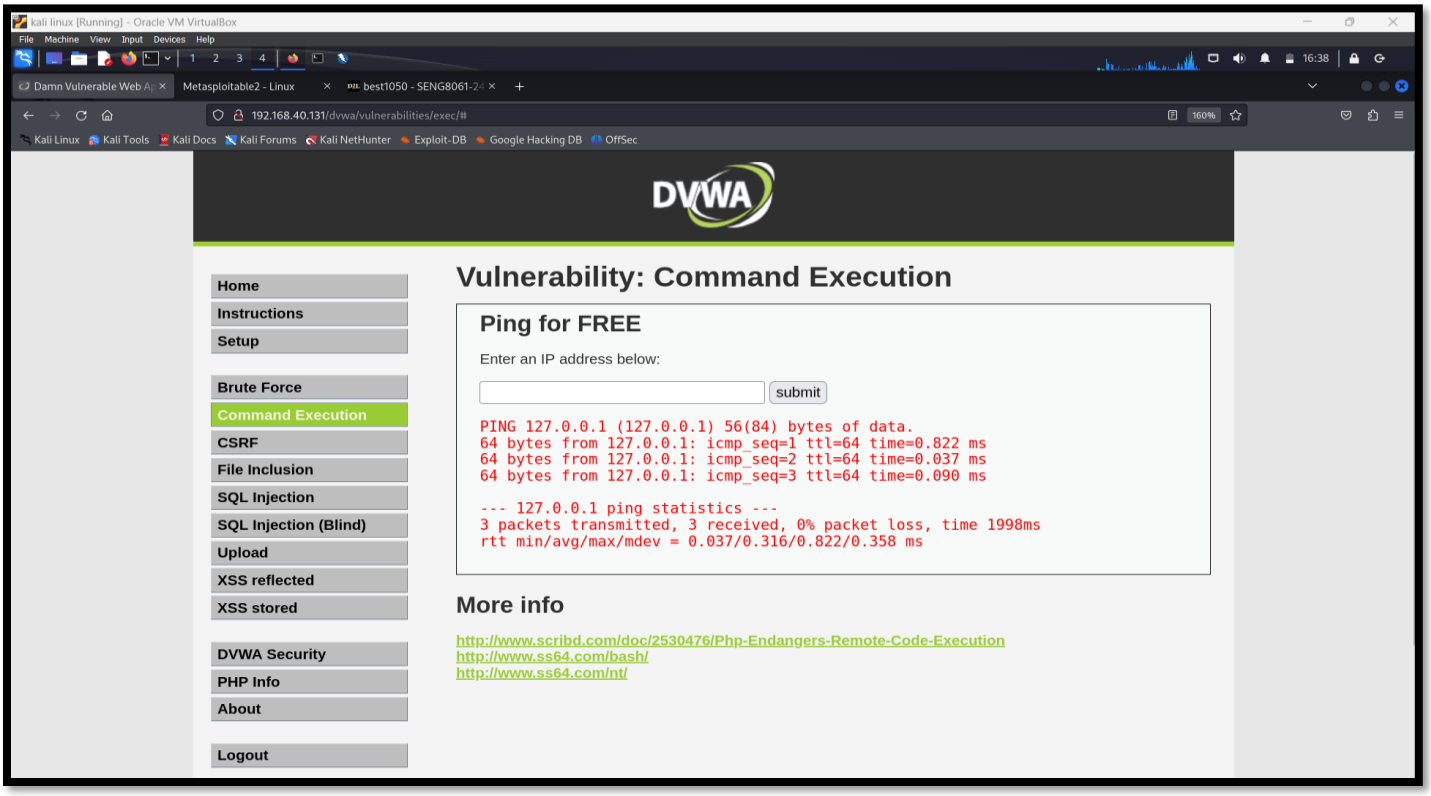
Medium (30-39),

Low (20-29)

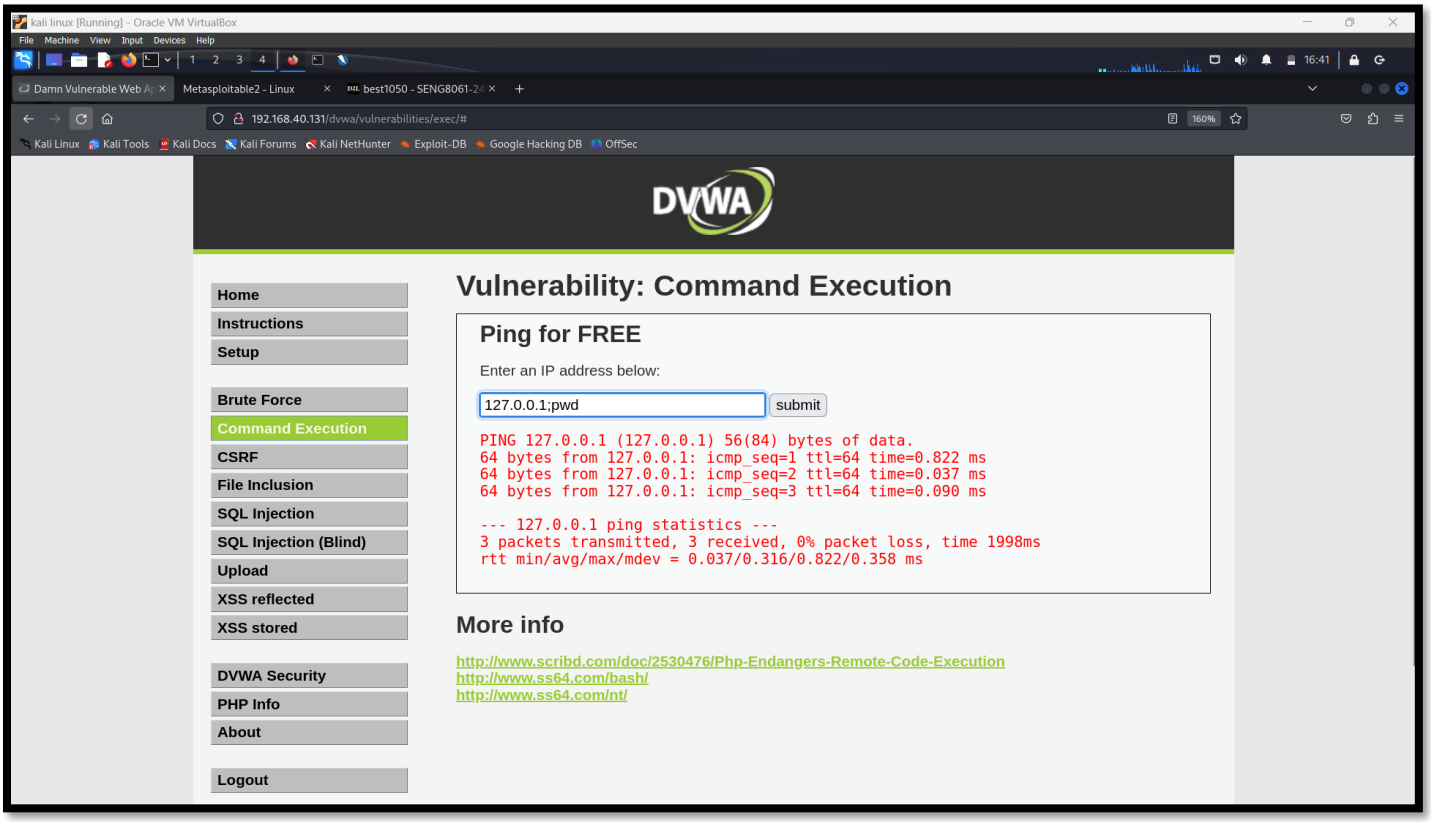
DREAD	Detail		Rank
Damage	XSS vulnerabilities can potentially lead to various types of damage, including stealing user credentials, session hijacking, defacement of the website, or spreading malware. In this case, the attacker was able to execute arbitrary scripts on the webpage, demonstrating a high potential for damage.		8/10
Reproducibility	The vulnerability appears to be easily reproducible, as the attacker was able to inject a script and trigger it successfully.		8/10
Exploitability	The exploitability of XSS vulnerabilities is typically high, as attackers can easily inject malicious scripts through input fields and trick other users into executing them. In this case, the attacker was able to exploit the vulnerability by simply inputting a script into the message field.		9/10
Affected Users	This depends on the number of users of the vulnerable application. If it's a widely used application, the number of affected users could be high.		7/10
Discoverability	XSS vulnerabilities are commonly discovered through manual testing or automated scanning tools. In this case, the vulnerability was discovered during testing, indicating moderate discoverability.		6/10
DREAD SCORE	**(Risk Level: Medium)**	Overall score	38/50

Vulnerability #5: Command Execution

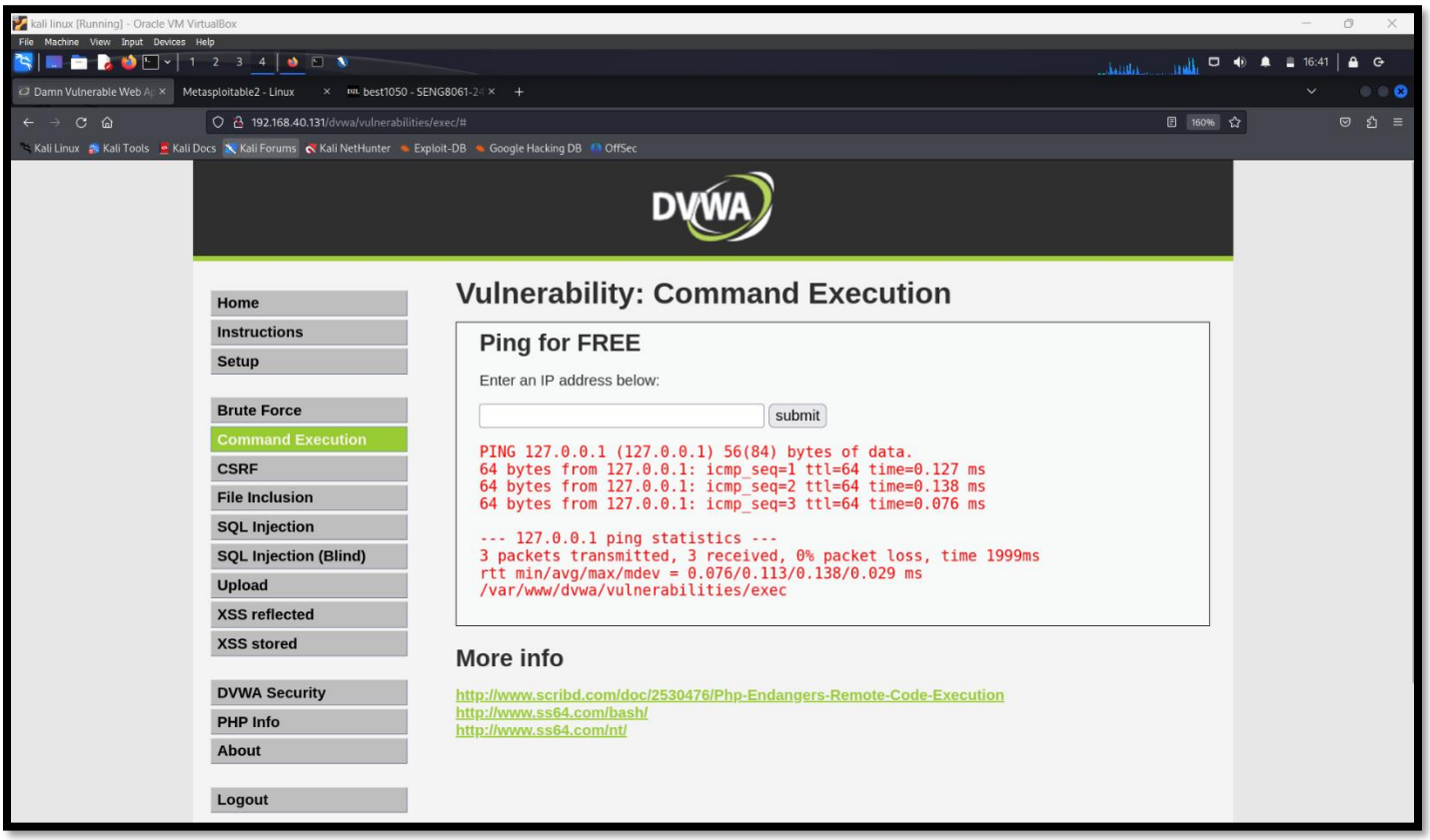
We opened the DVWA site and set the difficulty level to low. Then we went to Command Execution in the left navigation bar. We put 127.0.0.1 as input in the text field.



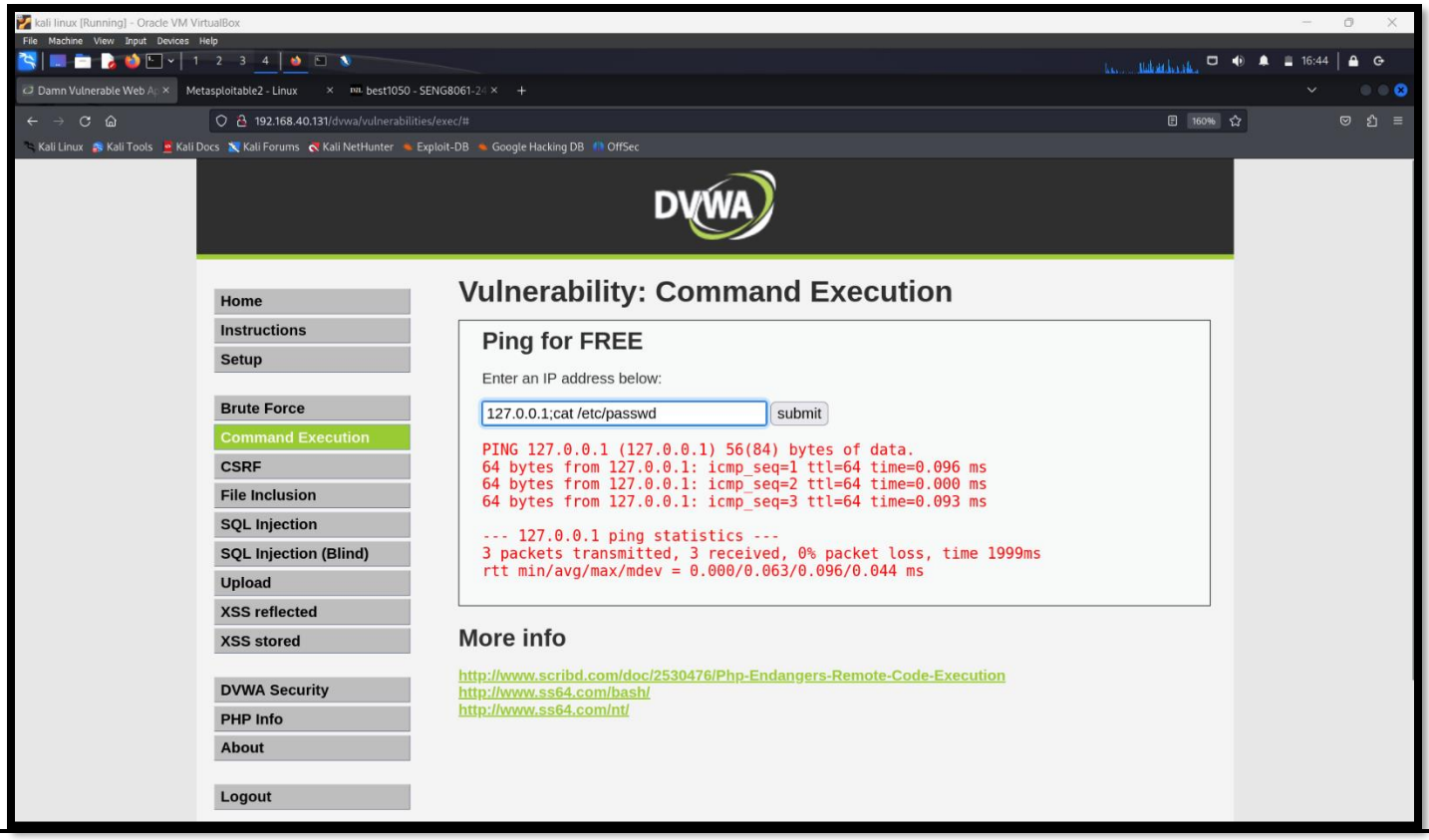
Here, we got some response. After that we put pwd at the end of localhost URL and hit the submit button.



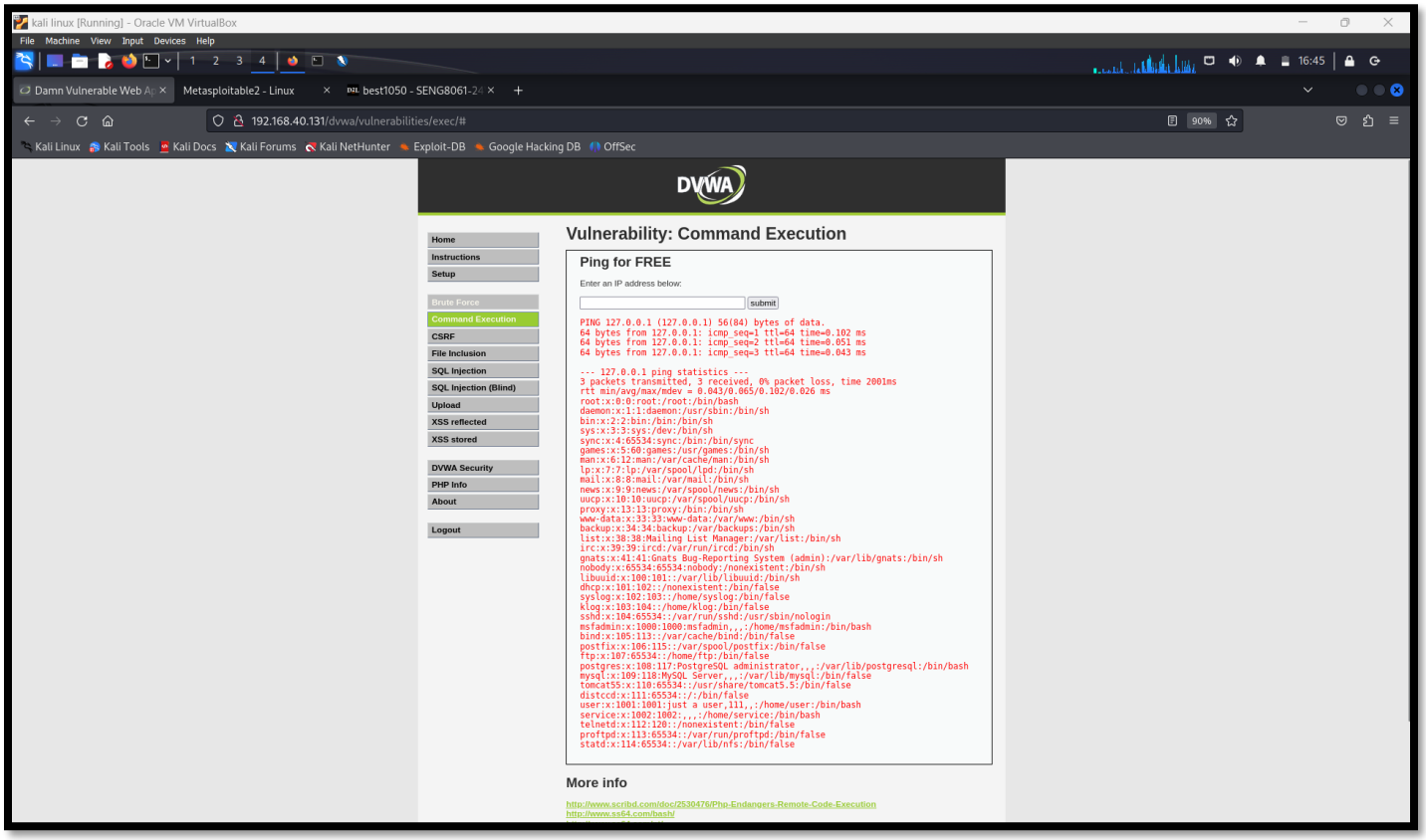
We got the directory of this machine /var/www/dvwa/vulnerabilities/exec.



To make this attack a little more forward, we put ;cat /etc/passwd to see what happened.



And we got the directory, which was ATC password files, so this was the vulnerability in low security.



Recommendation/Mitigation:

To fix such types of vulnerabilities, developer needs to follow below:

- Check all user inputs to ensure they follow the expected formats and ranges. Reject any input that appears to have malicious intent or special characters or commands.
- Sanitize user input by eliminating or encrypting special characters that could be interpreted as commands. To prevent injection attacks, use input filtering libraries or custom sanitization techniques.
- To minimize the potential damage of successful attacks, restrict the privileges of web application components. Run server processes with restricted permissions to avoid unwanted access to private information.
- Use secure APIs and libraries when performing commands or interacting with system resources. Keep the application safe from command injection vulnerabilities by avoiding using system calls or shell execution functions.

Risk Analysis

For the DREAD analysis we have use rank criteria as below:

DREAD Ranking:

High (40-50),

Medium (30-39),

Low (20-29)

DREAD	Detail		Rank
Damage	This vulnerability allows attackers to execute commands on the server, which leads to data breaches, system compromise, and unauthorized access. Disclosing system files like "/etc/passwd" is a serious threat to security and privacy.		9/10
Reproducibility	An attacker can easily reproduce the vulnerability. They can put malicious commands into the input areas of the application and observe how the server responds. Because the vulnerability is built into the application's architecture and behavior, it is likely to be exploited consistently.		10/10
Exploitability	Exploiting this issue needs minimal effort and knowledge. Attackers simply require basic command injection knowledge and access to the vulnerable application. There are further tools and scripts available that can automate the exploitation process.		9/10
Affected Users	This vulnerability can cause damage to all users of the affected application. Any user interacting with the application's input fields—such as form inputs or search bars—has a chance to unintentionally cause the server to execute malicious commands.		8/10
Discoverability	The vulnerability is relatively easy to discover through manual testing or automated scanning techniques. Security researchers, penetration testers, and malicious actors can identify the vulnerability by inspecting the application's input validation mechanisms and observing its response to unexpected inputs		8/10
DREAD SCORE	**(Risk Level: High)**	Overall score	44/50

Video Recording Link:

https://youtu.be/ClqeEhyQlFI