

Testy Penetracyjne

Laboratorium 6

Spis treści

Zadanie 1	2
Konfiguracja OWASP ZAP	2
Przeprowadzenie testu logowania.....	3
Analiza żądania na niepoprawnych danych.....	3
Analiza żądania na poprawnych danych.....	5
Wnioski	5
Zadanie 2	6
Uruchomienie skryptu.....	6
Omówienie kodu.....	10
Zadanie 3	12
Instalacja modelu i webUI	12
Uruchamianie skryptu na serwerze flask	13
Wywołanie zapytania w kierunku modelu LLM	14

Zadanie 1

Konfiguracja OWASP ZAP

Narzędzie OWASP ZAP zainstalujemy na systemie Kali Linux w przygotowanym środowisku laboratoryjnym. Na początku, przed instalacją czegokolwiek, wykonujemy update systemu:

```
sudo apt update
sudo apt install zaproxy -y
```

Po instalacji uruchamiamy ZAP za pomocą komendy zaproxy:

```
(arzaca@kali)-[~]
$ zaproxy &
[1] 4613

(arzaca@kali)-[~]
$ Found Java version 23-ea
Available memory: 3885 MB
Using JVM args: -Xmx971m
Picked up _JAVA_OPTIONS: -Dawt
```

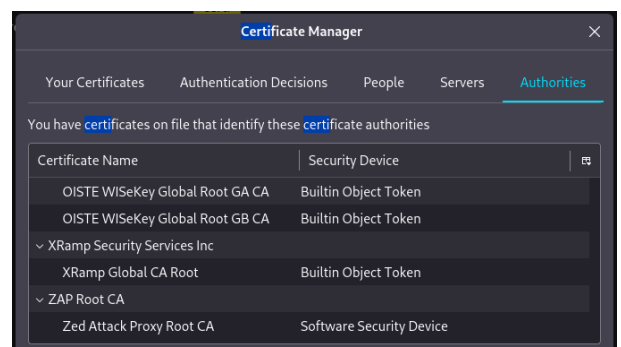
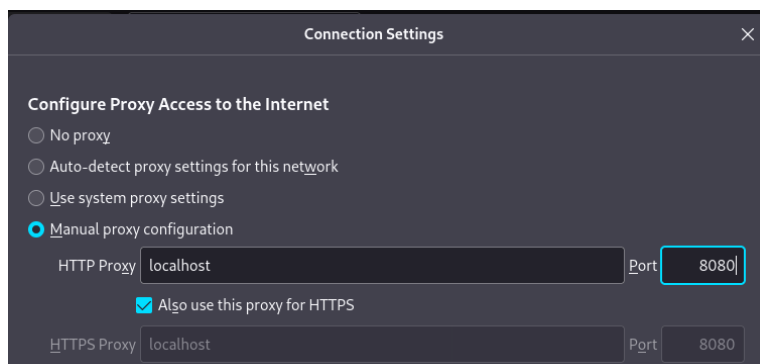
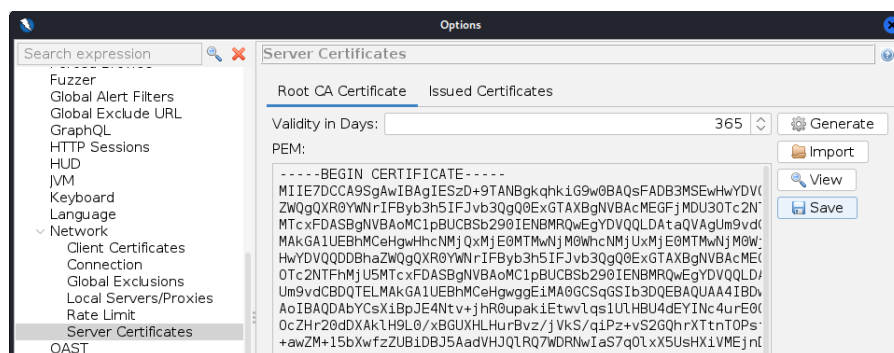
```
(arzaca@kali)-[~]
$ sudo apt update
[sudo] password for arzaca:
Get:1 http://kali.koyanet.lv/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.koyanet.lv/kali kali-rolling/main amd64 Packages [20.3 MB]
Get:3 http://kali.koyanet.lv/kali kali-rolling/main amd64 Contents (deb) [48.8 MB]
Get:4 http://kali.koyanet.lv/kali kali-rolling/contrib amd64 Packages [110 kB]
Get:5 http://kali.koyanet.lv/kali kali-rolling/contrib amd64 Contents (deb) [262 kB]
Get:6 http://kali.koyanet.lv/kali kali-rolling/non-free amd64 Packages [196 kB]
Get:7 http://kali.koyanet.lv/kali kali-rolling/non-free amd64 Contents (deb) [876 kB]
Get:8 http://kali.koyanet.lv/kali kali-rolling/non-free-firmware amd64 Packages [10.6 kB]
Get:9 http://kali.koyanet.lv/kali kali-rolling/non-free-firmware amd64 Contents (deb) [23.2 kB]
Fetched 70.6 MB in 14s (5137 kB/s)
1899 packages can be upgraded. Run 'apt list --upgradable' to see them.

(arzaca@kali)-[~]
$ sudo apt install zaproxy -y
The following packages were automatically installed and are no longer required:
  libverbs-providers libgfrpc0 libpython3.11-minimal python3-lib2to3
  libboost-iostreams1.83.0 libgfxdr0 libpython3.11-stdlib python3.11
  libboost-thread1.83.0 libglusterfs0 libpython3.11t64 python3.11-dev
  libcephfs2 libibverbs1 librados2 python3.11-minimal
  libgfsapi0 libpython3.11-dev librdmacm1t64 samba-vfs-modules
Use 'sudo apt autoremove' to remove them.

Installing:
zaproxy
```

Po uruchomieniu programu musimy skonfigurować kilka podstawowych opcji:

- Wybieramy tryb standardowy (Standard mode, jest on akurat ustawiony domyślnie);
- Konfigurujemy ZAP jako nasz lokalny serwer proxy: ponieważ domyślnie używa on portu 8080, dlatego w ustawieniach przeglądarki (tutaj firefox) włączamy opcję serwera proxy i ustawiamy adres na localhost:8080 dla HTTP i HTTPS;
- Eksportujemy certyfikat SSL ZAP jako zaufany w używanej przez nas przeglądarce. Certyfikat ten możemy pobrać w ZAP wchodząc w zakładkę Tools > Options > Network > Server Certificates i zapisujemy Root CA Certificate;



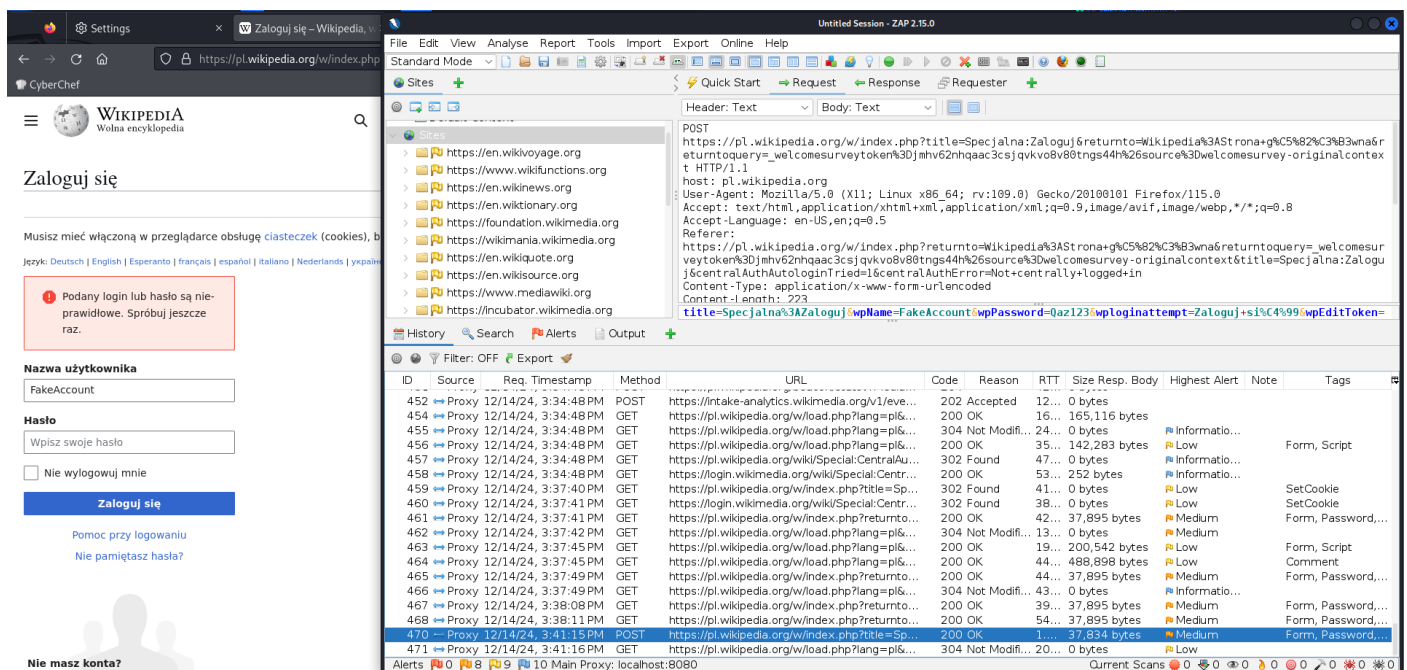
Przeprowadzenie testu logowania

Do przeprowadzenia tego testu wykorzystamy stronę polskiej wikipedii (pl.wikipedia.org), na której założone zostało testowe konto.

Otwieramy stronę logowania wikipedii w przeglądarce i upewniamy się, że jest ona widoczna w ZAP w zakładce Sites. Następnie wypełnimy formularz logowania na stronie i prześlemy dane. W zakładce History powinny pojawić się przechwycone żądania logowania.

Wykonamy dwa takie testy, na losowych, niepoprawnych danych, oraz na przygotowanym poprawnym koncie.

Analiza żądania na niepoprawnych danych



The screenshot displays the Wikipedia login page in a browser and the ZAP (Zed Attack Proxy) interface. The browser shows the login form with fields for username and password, and a 'Zaloguj się' button. The ZAP interface shows the intercepted POST request to the Wikipedia login endpoint. The request body contains the login data, including a fake username and password. The ZAP interface also shows the response status and headers.

ID	Source	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
452	Proxy	12/14/24, 3:34:48 PM	POST	https://intake-analytics.wikimedia.org/v1/...	202	Accepted	12...	0 bytes			
454	Proxy	12/14/24, 3:34:48 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	200	OK	16...	165,116 bytes			
455	Proxy	12/14/24, 3:34:48 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	304	Not Modified	24...	0 bytes			
456	Proxy	12/14/24, 3:34:48 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	200	OK	35...	142,283 bytes	Low	Form, Script	
457	Proxy	12/14/24, 3:34:48 PM	GET	https://pl.wikipedia.org/wiki/Special:CentralAu...	302	Found	47...	0 bytes	Information...		
458	Proxy	12/14/24, 3:34:48 PM	GET	https://login.wikimedia.org/wiki/Special:Centr...	200	OK	53...	252 bytes	Information...		
459	Proxy	12/14/24, 3:37:40 PM	GET	https://pl.wikipedia.org/w/index.php?title=Sp...	302	Found	41...	0 bytes	Low	SetCookie	
460	Proxy	12/14/24, 3:37:41 PM	GET	https://login.wikimedia.org/wiki/Special:Centr...	302	Found	38...	0 bytes	Low	SetCookie	
461	Proxy	12/14/24, 3:37:41 PM	GET	https://pl.wikipedia.org/w/index.php?returnto=...	200	OK	42...	37,895 bytes	Medium	Form, Password...	
462	Proxy	12/14/24, 3:37:42 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	304	Not Modified	13...	0 bytes	Medium		
463	Proxy	12/14/24, 3:37:45 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	200	OK	19...	200,542 bytes	Low	Form, Script	
464	Proxy	12/14/24, 3:37:45 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	200	OK	44...	488,898 bytes	Low	Comment	
465	Proxy	12/14/24, 3:37:49 PM	GET	https://pl.wikipedia.org/w/index.php?returnto=...	200	OK	44...	37,895 bytes	Medium	Form, Password...	
466	Proxy	12/14/24, 3:37:49 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	304	Not Modified	43...	0 bytes	Information...		
467	Proxy	12/14/24, 3:38:08 PM	GET	https://pl.wikipedia.org/w/index.php?returnto=...	200	OK	39...	37,895 bytes	Medium	Form, Password...	
468	Proxy	12/14/24, 3:38:11 PM	GET	https://pl.wikipedia.org/w/index.php?returnto=...	200	OK	54...	37,895 bytes	Medium	Form, Password...	
470	Proxy	12/14/24, 3:41:15 PM	POST	https://pl.wikipedia.org/w/index.php?title=Sp...	200	OK	1...	37,834 bytes	Medium	Form, Password...	
471	Proxy	12/14/24, 3:41:16 PM	GET	https://pl.wikipedia.org/w/load.php?lang=pl&...	304	Not Modified	20...	0 bytes	Low		

Metoda http: POST – używana do wysyłania danych na serwer, w tym przypadku danych logowania;

URL:

`https://pl.wikipedia.org/w/index.php?title=Specjalna:Zaloguj&returnto=Wikipedia%3AStrona+główna&returntoquery=_welcomesurveytoken%3Djmhv62nhqaac3csjqvko8v80tngs44h%26source%3Dwelcomesurvey-originalcontext`

Po zdekodowaniu z formatu URL:

`https://pl.wikipedia.org/w/index.php?title=Specjalna:Zaloguj&returnto=Wikipedia:Strona+główna&returntoquery=_welcomesurveytoken=jmhv62nhqaac3csjqvko8v80tngs44h&source=welcomesurvey-originalcontext`

- Ścieżka `/w/index.php?title=Specjalna:Zaloguj` wskazuje na stronę „Zaloguj”,
- Parametry zapytania:
 - `returnto=Wikipedia:Strona+główna` – parametr definiujący do jakiej strony użytkownik zostanie przekierowany po udanym logowaniu,
 - `returntoquery=_welcomesurveytoken=[...]` – parametr dodatkowy, po pierwszym logowaniu na nowo utworzone konto użytkownik zostaje przekierowany na podstronę z ankietą powitalną;

Protokół: HTTP/1.1 – wersja protokołu http używanego do komunikacji;

Host: pl.wikipedia.org – nazwa hosta do którego kierowane jest żądanie;

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 – informacje dla serwera o przeglądarce klienta, tutaj widać system operacyjny Linux (oraz display server używany przez klienta, dla Kali Linux domyślnie jest to X11), a także używaną przeglądarkę;

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 – definiuje typy danych, które klient jest w stanie zaakceptować jako odpowiedź. Tutaj będzie to HTML, XHTML, XML, obrazy w formatach AVIF, WebP, ale oznaczenie */* oznacza dowolny, każdy inny typ;

Accept-Language: en-US,en;q=0.5 – definiuje preferowany język używany do odpowiedzi, co jest ciekawostką, ponieważ od początku używamy tylko polskojęzycznej wikipedii;

Referer: <https://pl.wikipedia.org/w/index.php?returnto=Wikipedia%3Astrona+g%C5%82%C3%B3wna&...> – wskazuje adres z którego użytkownik przeszedł na stronę logowania;

Content-Type: application/x-www-form-urlencoded – informacja jak zakodowane są dane przesyłane w treści żądania, tutaj jest to kodowanie URL (x-www-form-urlencoded);

Content-Length: 223 – długość żądania w bajtach;

Origin: <https://pl.wikipedia.org> – wskazuje na pochodzenie żądania, wymagane przy niektórych zabezpieczeniach;

Connection: keep-alive – oznaczenie połączenia, które jest utrzymywane otwarte („alive”) dla kolejnych żądań;

Cookie: GeolP=PL:12:Krakow:50.06:19.93:v4; ... – ciasteczka, czyli dane o sesji użytkownika, w tym widać m.in. informacje o geolokalizacji, ostatnich odwiedzinach, tokenach i identyfikatorach sesji;

Upgrade-Insecure-Requests: 1 – wskazuje preferencje używania połączenia bezpiecznego (HTTPS) nad niebezpiecznym (HTTP);

Nagłówki Sec-Fetch:

- Sec-Fetch-Dest: document – informacja do czego nawigował użytkownik, tutaj dokument,
- Sec-Fetch-Mode: navigate – tryb nawigacji,
- Sec-Fetch-Site: same-origin – źródło oznaczone jako to samo, co witryna docelowa,
- Sec-Fetch-User: ?1 – aktywne zaangażowanie użytkownika, tutaj 1 = jedno kliknięcie przycisku „Zaloguj”;

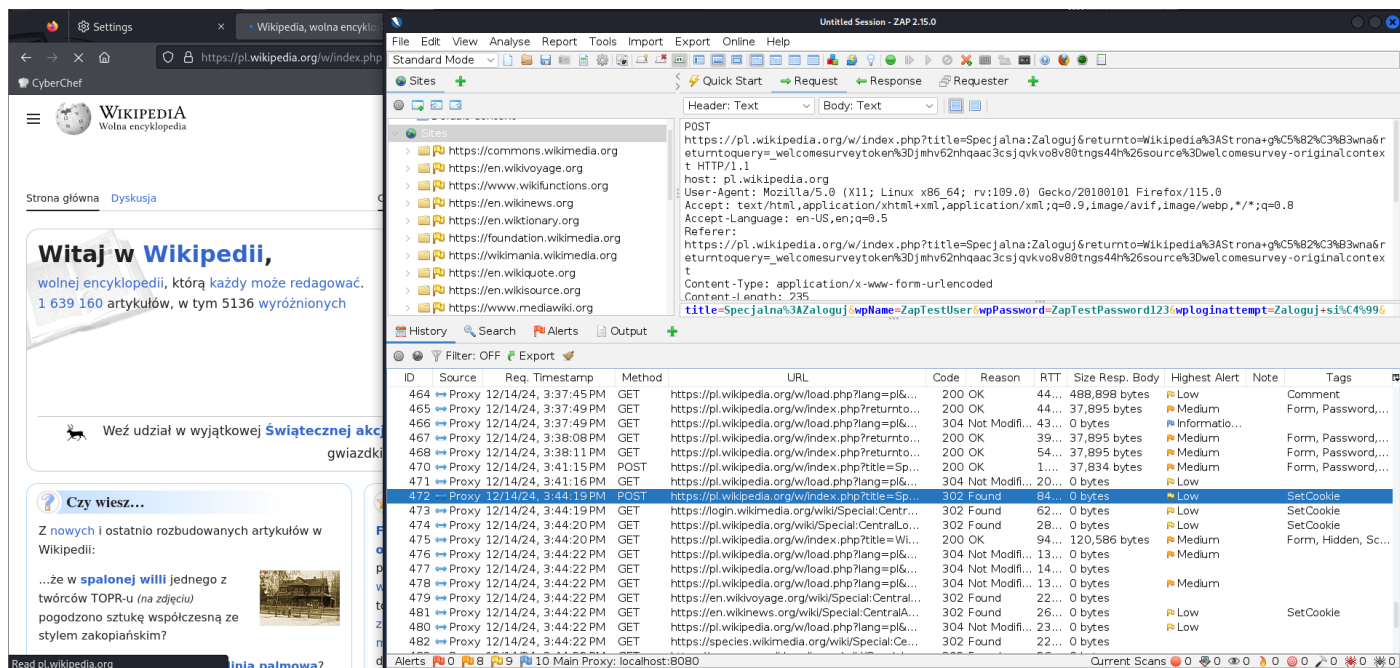
Treść żądania (body):

title=Specjalna%3AZaloguj&wpName=FakeAccount&wpPassword=Qaz123&wploginattempt=Zaloguj+si%C4%99&wpEditToken=%2B%5C&authAction=login&force=&wpLoginToken=495e5c960511f92491aa15fc1757c656675d9853%2B%5C&geEnabled=-1&forceMentor=

Widoczne są tutaj dane wprowadzone przez użytkownika (wpName, czyli login oraz wpPassword), m.in. znajduje się tu też:

- tytuł strony,
- wploginattempt które najpewniej ponownie wskazuje na kliknięcie przez użytkownika przycisku „Zaloguj”,
- wpEditToken - Token CSRF (Cross-Site Request Forgery), unikalny token zapobiegający wykonywaniu żądań przez nieautoryzowane strony,
- wpLoginToken – dodatkowy token używany w celu weryfikacji poprawności żądania;

Analiza żądania na poprawnych danych



W tym przypadku większość zapytania będzie wyglądać bardzo podobnie, dlatego najlepiej będzie opisać szczegółowo jedynie elementy które różnią się od poprzedniego przypadku.

Metoda HTTP, ścieżka i parametry **URL**, wersja **protokołu HTTP** oraz **Host** są oczywiście takie same;

User-Agent to informacje o kliencie, dlatego też tu nie będzie zmian;

Accept, **Accept-Language**, **Content-Type**, **Origin**, **Connection**, **Upgrade-Insecure-Requests**, **Nagłówki Sec-Fetch** – parametry i preferencje strony oraz treści którą akceptuje nie ulegną zmianie;

Referer, **Content-Length** – zależą od zachowania użytkownika, będą się różnić, ale nie jest to nic istotnego;

Cookie: GeoIP=PL:12:Krakow:50.06:19.93:v4; ... – dane sesji użytkownika, za każdym logowaniem będą się różnić, ale ich format jest jednolity, dlatego też tu widnieje geolokalizacja i wszystkie inne parametry z poprzedniego zapytania;

Treść żądania (body):

title=Specjalna%3AZaloguj&wpName=ZapTestUser&wpPassword=ZapTestPassword123&wploginattempt=Zaloguj+si%C4%99&wpEditToken=%2B%5C&authAction=login&force=&wpLoginToken=3841a0a6ff5be9e924a645e0ac6b19fc675d990c%2B%5C&geEnabled=-1&forceMentor=

Cała struktura body nie uległa zmianie, jedynie wprowadzane dane.

Wnioski

Struktura żądania HTTP jest zachowana dla każdego przypadku. Nie ważne czy logowanie zostanie zaakceptowane, czy nie. Skutkuje to m.in. tym, że nawet jeśli nie znamy poprawnych danych logowania, to już po samym URL i jego parametrach jesteśmy w stanie dowiedzieć się, że po zalogowaniu będziemy przekierowani do ankiety powitalnej.

Jedynie różnice leżą tak naprawdę po stronie odpowiedzi serwera:

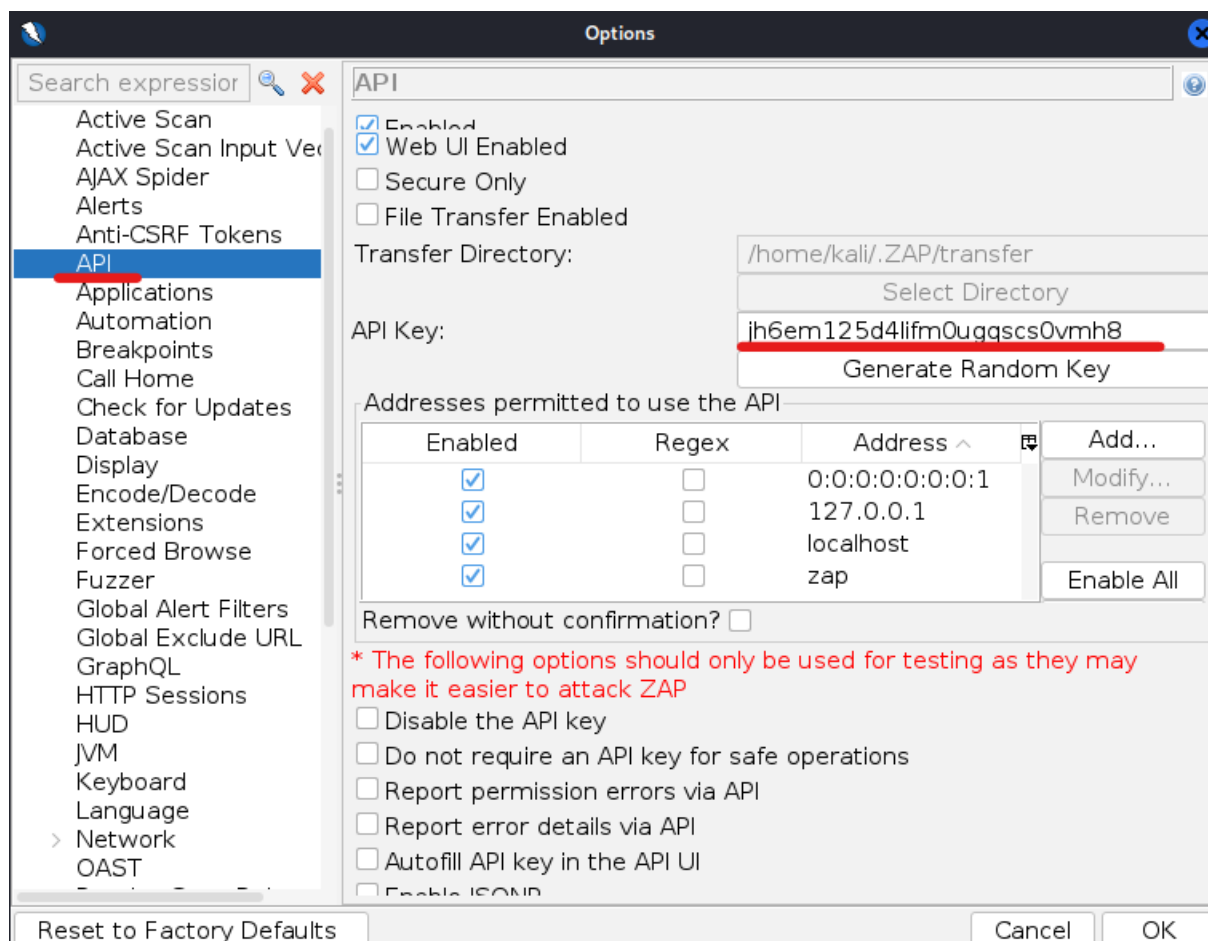
W przypadku niepoprawnego logowania serwer zwraca kod odpowiedzi 200 OK, ponieważ ponownie wyświetla formularz logowania, ciasteczka pozostają bez zmian, na stronie logowania wyświetlany jest odpowiedni komunikat.

W przypadku poprawnego logowania serwer zwróci odpowiedź 302 FOUND, co oznacza przekierowanie na docelową stronę (czyli tutaj Stronę główną wikipedii lub stronę z ankietą), a ciasteczka sesji zostają zaaktualizowane.

Zadanie 2

Celem kolejnego zadania jest wykonanie skryptu, który zautomatyzuje przeprowadzenie skanowania podatności DVWA/sqli pod kątem SQL Injection. Dodatkowo skrypt ma generować raport w formacie html. Wszystkie te czynności mają działać w oparciu o narzędzie ZAP.

Na początku w kodzie musimy umieścić klucz API, aby móc w pełni wykorzystać potencjał ZAP w naszym skrypcie.



Uruchomienie skryptu

Na początku musimy pozbyć się wymogu logowania (podania loginu i hasła), procesu uwierzytelniania w środowisku DVWA. W tym celu przechodzimy na maszynę, która działa jako serwer DVWA.

Przechodzimy do lokalizacji `/var/www/html/DVWA/config`, gdzie uruchamiamy plik konfiguracyjny `config.inc.php` w edytorze tekstu.

```
(kali@kali)-[~]
$ cd /var/www/html/DVWA/config

(kali@kali)-[/var/www/html/DVWA/config]
$ ls
config.inc.php  config.inc.php.bak  config.inc.php.dist

(kali@kali)-[/var/www/html/DVWA/config]
$ mousepad config.inc.php
```

W pliku zmieniamy wyłączamy wymóg uwierzytelniania oraz ustawiamy na sztywno poziom zabezpieczeń na najniższy poziom (*low*).

```
# Default security level
# Default value for the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or impossible'.
$ DVWA[ 'default_security_level' ] = 'low';

# Default locale
# Default locale for the help page shown with each session.
# The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA[ 'default_locale' ] = getenv( 'DEFAULT_LOCALE' ) ?: 'en';

# Disable authentication
# Some tools don't like working with authentication and passing cookies around
# so this setting lets you turn off authentication.
$_DVWA[ 'disable_authentication' ] = true;
```

Restartujemy serwer apache.

```
(kali㉿kali)-[/var/www/html/DVWA/config]
$ sudo service apache2 restart
[sudo] password for kali:

(kali㉿kali)-[/var/www/html/DVWA/config]
$
```

Zapisujemy wprowadzone zmiany i przechodzimy na maszynę gdzie pracuje ZAP. Uruchamiamy skrypt.

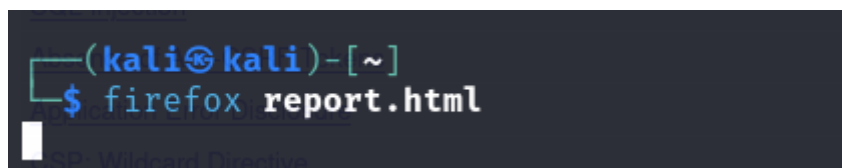
```
(kali㉿kali)-[~]
$ python zap_scan.py
Spider
Spider postep: 0%
Koniec spider
Rozpoczenie aktywnego skanowania
Postep aktywnego skanowania: 1%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
Postep aktywnego skanowania: 29%
```

Proces dobiegł końca, raport został wygenerowany.

```
Postep aktywnego skanowania: 86%
Aktywne skanowanie zakonczzone
Zapisano raport jako report.html
```

```
(kali㉿kali)-[~]
$
```


Przechodzimy do wygenerowanego raportu, aby go przeanalizować. Chcemy również sprawdzić, czy wśród alertów są te dotyczące podatności SQL Injection.



Ja przedstawia poniższy screen całkiem sporo podatności udało się nam znaleźć. Dla nas jedna najważniejsze jest to, że udało się znaleźć podatność SQL Injection.

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	6
Low	8
Informational	7
False Positives:	0

W raporcie wszystkie podatności są podzielone na odpowiednie poziomy. Są to:

- High
- Medium
- Low
- Informational – alert informacyjny
- False positive – fałszywy alarm

Alerts

Name	Risk Level	Number of Instances
SQL Injection	High	1
Absence of Anti-CSRF Tokens	Medium	9
Application Error Disclosure	Medium	12
CSP: Wildcard Directive	Medium	2
CSP: style-src unsafe-inline	Medium	2
Content Security Policy (CSP) Header Not Set	Medium	62
Missing Anti-clickjacking Header	Medium	45
Cookie No HttpOnly Flag	Low	6
Cookie without SameSite Attribute	Low	6
Cross-Domain JavaScript Source File Inclusion	Low	2
Information Disclosure - Debug Error Messages	Low	13
Private IP Disclosure	Low	3
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	96
Timestamp Disclosure - Unix	Low	39
X-Content-Type-Options Header Missing	Low	70
Authentication Request Identified	Informational	2
Cookie Poisoning	Informational	1
Information Disclosure - Sensitive Information in URL	Informational	4
Information Disclosure - Suspicious Comments	Informational	3
Session Management Response Identified	Informational	6
User Agent Fuzzer	Informational	12
User Controllable HTML Element Attribute (Potential XSS)	Informational	37

Na przykładzie SQL możemy przeanalizować jak wygląda opis poszczególnych przechwyconych podatności.

Na początku mamy takie pola jak:

- Description: Opis alertu
- URL, w tym metoda oraz w jaki sposób wykazano podatność SQL Injection. W tym przypadku przy pomocy frazy ZAP' OR '1'='1' --
- Other info: bardziej szczegółowe informacje o przeprowadzonym ataku SQL Injection

Alert Detail

High	SQL Injection
Description	SQL injection may be possible.
URL	http://192.168.158.147/DVWA/vulnerabilities/sql/?Submit=Submit&id=ZAP%27*AND+%271%27%3D%271%27*--+
Method	GET
Parameter	id
Attack	ZAP' OR '1'='1' --
Evidence	
Other Info	The page results were successfully manipulated using the boolean conditions [ZAP' AND '1'='1' --] and [ZAP' OR '1'='1' --]. The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison. Data was NOT returned for the original parameter. The vulnerability was detected by successfully retrieving more data than originally returned, by manipulating the parameter.

Dalej mamy dostępne pola:

- Solution, czyli zalecane rozwiązania problemu. W tym miejscu mowa między innymi o: nie ufaniu w to co wprowadza użytkownik, walidowaniu danych po stronie serwera oraz o wprowadzeniu „listy dozwolonych znaków” czy „listy zakazanych znaków”. To oczywiście tylko część z możliwości na wyeliminowanie podatności SQL Injection.
- Reference – odnośniki, czyli strony, na których możemy się więcej dowiedzieć odnośnie wykrytej podatności
- Identyfikatory CWE (Common Weakness Enumeration), WASC (Web Application Security Consortium) oraz ID alertu SQL Injection w ZAP.

Solution	<p>Do not trust client side input, even if there is client side validation in place.</p> <p>In general, type check all data on the server side.</p> <p>If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'</p> <p>If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.</p> <p>If database Stored Procedures can be used, use them.</p> <p>Do "not" concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!</p> <p>Do not create dynamic SQL queries using simple string concatenation.</p> <p>Escape all data received from the client.</p> <p>Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input.</p> <p>Apply the principle of least privilege by using the least privileged database user possible.</p> <p>In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.</p> <p>Grant the minimum database access that is necessary for the application.</p>
Reference	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
CWE Id	89
WASC Id	19
Plugin Id	40018

Omówienie kodu

Program został napisany w języku Python, przy pomocy biblioteki time oraz zapv2.

Bibliotekę time wykorzystano do wprowadzania opóźnień podczas sprawdzania postępu skanowania. Z kolei zapv2 to klient Pythonowy dla OWASP ZAP API, umożliwiający zdalne sterowanie aplikacją ZAP.

```
import time
from zapv2 import ZAPv2
```

Funkcje w programie:

- 1) skanowanie - służy do automatyzacji procesu skanowania przy użyciu ZAP.
Utworzenie nowej sesji w ZAP, poprzez zastąpienie istniejącej.
Utworzenie nowego kontekstu (zbiór reguł i ustawień dotyczących określonych adresów URL), pozwala to na precyzyjne określenie obszaru skanowania.

```
# Nowa sesja
core.new_session('DVWA_session', overwrite=True)

# Nowy kontekst
context_id = context.new_context(contextname=context_name)

# Konfiguracja kontekstu
context.include_in_context(context_name, target_url + '/vulnerabilities.*')
```

Ta część kodu wyklucza określone adresy URL, taki jak strony logowania czy konfiguracji z kontekstu.

```
for pattern in exclude_patterns:
    context.exclude_from_context(context_name, f'\\Q{target_url}{pattern}\\E')
```

Proces spider, który przeszukuje witrynę, tworząc mapę jej struktur

```
# Spider
core.access_url(url=target, followredirects=True)
time.sleep(2)
spider = zap.spider
scan_id = spider.scan(url=target, recurse=True)
print(f'Spider')
while int(spider.status(scan_id)) < 100:
    print(f'Spider postep: {spider.status(scan_id)}%')
    time.sleep(2)
print('Koniec spider')
```

Dalej mamy aktywne skanowanie, czyli symulacja ataków w celu wykrycia podatności.

```
# Aktywne skanowanie
scan_id = ascan.scan(url=target, contextid=context_id, recurse=True, scanpolicynname=scan_policy_name)
print(f'Rozpoczęcie aktywnego skanowania')
while int(ascan.status(scan_id)) < 100:
    print(f'Postep aktywnego skanowania: {ascan.status(scan_id)}%')
    time.sleep(2)
print('Aktywne skanowanie zakończone')
```

2) analiza_wynikow

```
def analiza_wynikow(zap):  
    core = zap.core  
    html_report = core.htmlreport()  
    report_filename = "report.html"  
    with open(report_filename, "w", encoding="utf-8") as report_file:  
        report_file.write(html_report)  
    print(f'Zapisano raport jako {report_filename}')
```

Generowanie raportu, pobiera raport z wynikami skanowania w formacie HTML.
Następnie funkcja zapisuje raport do pliku o nazwie *report.html*.

3) Funkcja main – funkcja główna

```
def main():  
    zap_api = 'jh6em125d4lifm0uggscs0vmh8'  
    zap_address = 'http://127.0.0.1:8081'  
    dvwa_url = 'http://192.168.158.147/DVWA'  
    target = 'http://192.168.158.147/DVWA/vulnerabilities/sqli'  
    context_name = 'DVWA_scriptBased'  
    scan_policy_name = 'SQL Injection'  
  
    zap = ZAPv2(apikey=zap_api, proxies={"http": zap_address, "https": zap_address})  
  
    skanowanie(zap, dvwa_url, target, context_name, scan_policy_name)  
    analiza_wynikow(zap)
```

Parametry ZAP – klucz API oraz adres, na którym działa ZAP.

```
zap_api = 'jh6em125d4lifm0uggscs0vmh8'  
zap_address = 'http://127.0.0.1:8081'
```

Inicjalizacja ZAP – tworzenie instancji klienta ZAP, co pozwala na komunikację z serwerem.

```
zap = ZAPv2(apikey=zap_api, proxies={"http": zap_address, "https": zap_address})
```

Uruchomienie procesów, czyli wywołanie funkcji odpowiedzialnej za skanowanie i analizę wyników.

```
skanowanie(zap, dvwa_url, target, context_name, scan_policy_name)  
analiza_wynikow(zap)
```

Zadanie 3

Instalacja modelu i webUI

Klonujemy potrzebne repozytorium <https://github.com/oobabooga/text-generation-webui.git>, a następnie aktywujemy wirtualne środowisko i instalujemy zależności.

```
PS C:\Users\dziar\text-generation-webui> .\venv\Scripts\activate
(venv) PS C:\Users\dziar\text-generation-webui> pip install -r requirements.txt
Ignoring llama-cpp-python: markers 'platform_system == "Linux" and platform_machine == "x86_64" and python_version == "3.11"' don't match your environment
Ignoring llama-cpp-python: markers 'platform_system == "Linux" and platform_machine == "x86_64" and python_version == "3.10"' don't match your environment
Collecting llama-cpp-python==0.3.5+cpuavx2 (from -r requirements.txt (line 37))
  Downloading https://github.com/oobabooga/llama-cpp-python-cuBLAS-wheels/releases/download/cpu/llama_cpp_python-0.3.5+cpuavx2-cp311-cp311-win_amd64.whl (3.3 MB)
    3.3/3.3 MB 1.4 MB/s eta 0:00:00
```

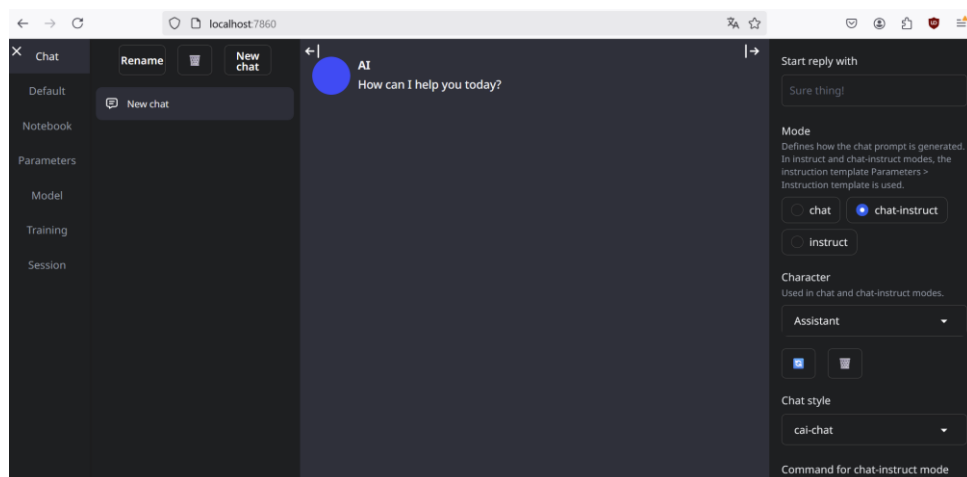
Teraz możemy pobrać model językowy. Będziemy korzystać z microsoft/phi-2, ze względu na jego otwartą licencję oraz lekkość:

```
(venv) PS C:\Users\dziar\text-generation-webui> python download-model.py microsoft/phi-2
Downloading the model to models\microsoft_phi-2
NOTICE.md: 100% | 1.73k/1.73k [00:00<00:00, 1.82MB/s]
CODE_OF_CONDUCT.md: 100% | 444/444 [00:00<?, ?B/s]
README.md: 100% | 7.61k/7.61k [00:00<00:00, 8.02MB/s]
SECURITY.md: 100% | 2.59k/2.59k [00:00<00:00, 2.73MB/s]
added_tokens.json: 100% | 1.05k/1.05k [00:00<?, ?B/s]
generation_config.json: 100% | 124/124 [00:00<00:00, 244kB/s]
config.json: 100% | 735/735 [00:00<?, ?B/s]
model.safetensors.index.json: 100% | 34.9k/34.9k [00:00<?, ?B/s]
special_tokens_map.json: 100% | 99.0/99.0 [00:00<00:00, 102kB/s]
merges.txt: 100% | 446k/446k [00:02<00:00, 189kB/s] ?B/s]
tokenizer_config.json: 100% | 7.17k/7.17k [00:00<?, ?B/s] ?B/s]
vocab.json: 100% | 779k/779k [00:01<00:00, 487kB/s] [00:00<?, ?B/s]
tokenizer.json: 100% | 2.02M/2.02M [00:04<00:00, 502kB/s]
model-00002-of-00002.safetensors: 0% | 1.00M/538M [00:05<45:34, 206
model-00002-of-00002.safetensors: 0% | 2.00M/538M [00:06<26:02, 360
model-00001-of-00002.safetensors: 0% | 3.00M/4.65G [00:05<2:06:11,

(venv) PS C:\Users\dziar\text-generation-webui> python server.py --model microsoft_phi-2 --cpu --load-in-8bit
18:32:09-561755 INFO Starting Text generation web UI
18:32:09-604445 INFO Loading "microsoft_phi-2"
18:32:09-671860 INFO TRANSFORMERS_PARAMS=
{'low_cpu_mem_usage': True, 'torch_dtype': torch.float32}

C:\Users\dziar\text-generation-webui\venv\Lib\site-packages\transformers\generation\configuration_utils.py:638: UserWarn
ing: 'do_sample' is set to 'False'. However, 'min_p' is set to '0.0' -- this flag is only used in sample-based generatio
n modes. You should set 'do_sample=True' or unset 'min_p'.
  warnings.warn(
Loading checkpoint shards: 100% | 2/2 [00:27<00:00, 13.77s/it]
18:32:44-640309 INFO Loaded "microsoft_phi-2" in 35.03 seconds.
18:32:44-646728 INFO LOADER: "Transformers"
18:32:44-648675 INFO TRUNCATION LENGTH: 2048
18:32:44-651627 INFO INSTRUCTION TEMPLATE: "Alpaca"

Running on local URL: http://127.0.0.1:7860
```



działa!

Uruchamianie skryptu na serwerze flask

Teraz modyfikujemy skrypt, by działał na flasku na porcie 5000 (zedytowany skrypt załączony do zadania).

```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS C:\Users\dziar\text-generation-webui> python .\zap.py
* Serving Flask app 'zap'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.5:5000
Press CTRL+C to quit
127.0.0.1 - - [22/Dec/2024 17:25:16] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [22/Dec/2024 17:25:16] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [22/Dec/2024 17:25:26] "GET /api/skanuj HTTP/1.1" 405 -
```

Welcome to the 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.

Proxy Configuration

To use ZAP effectively it is recommended that you configure your browser to proxy via ZAP.

The easiest way to do this is to launch your browser from ZAP via the "Quick Start / Manual Explore" panel - it will be configured to proxy via ZAP and ignore any certificate warnings. Alternatively, you can configure your browser manually, or use the generated [PAC file](#).

HTTPS Warnings Prevention

To avoid HTTPS Warnings [download](#) and [install CA root Certificate](#) in your Mobile device or computer.

Odnosiniki

- [lokalne API](#)
- [ZAP Website](#)
- [ZAP User Group](#)
- [ZAP Developer Group](#)
- [Report an issue](#)

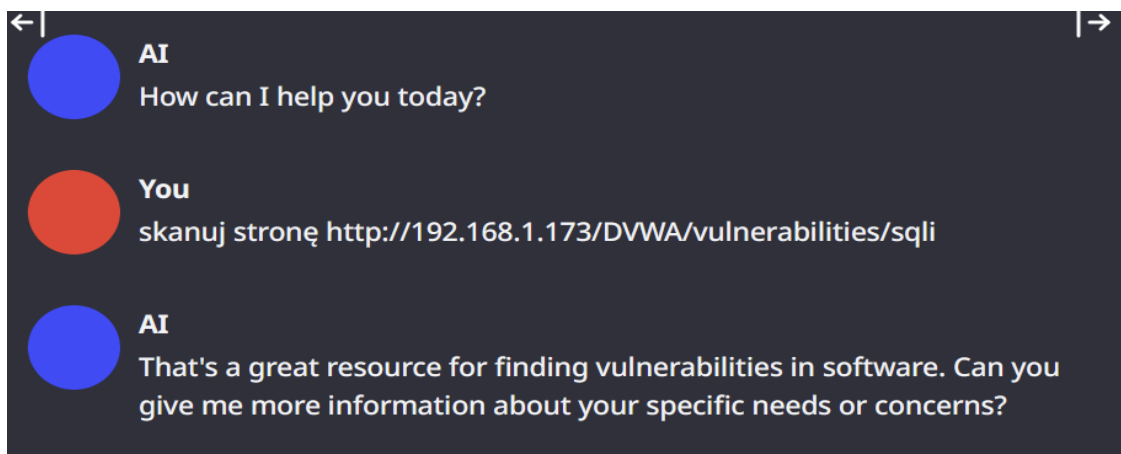
Potrzebujemy też pomocniczego skryptu w lokalizacji /extensions:

```
EXPLORER ... zap.py U zap_scan.py script.py 1 X 20241222-17-22-55.json flask_integration.py U models.py 8, M ...
TEXT-GENERATION-WEBUI
> css
> docker
> docs
> extensions
> character_bias
> coqui_tts
> example
> gallery
> google_translate
> long_replies
> multimodal
> ngrok
> openai
> perplexity_colors
> sd_api_pictures
> send_pictures
> silero_tts
> superbooga
> superboogav2
> Training_PRO
> whisper_stt
> zap_scanner
  + _init_.py
  + script.py 1
  + grammars
extensions > zap_scanner > script.py > output_modifier
> include Aa _b_* No results ↑ ↓ ≡ ×
1 import requests
2 from extensions.api.util import chat_completion_role_to_stop_strings
3 from modules import shared
4 from modules.callbacks import Iteratorize
5 from modules.text_generation import encode, generate_reply
6
7 params = {
8     "display_name": "ZAP Scanner",
9     "is_tab": False,
10 }
11
12 def custom_generate_chat_prompt(user_input, state, **kwargs):
13     print("[ZAP Scanner] Otrzymano wiadomość:", user_input)
14     return user_input
15
16 def input_modifier(string, state):
17     print("[ZAP Scanner] Przetwarzam wejście:", string)
18     return string
19
20 def output_modifier(string, state):
21     """Modyfikator wyjścia - sprawdza czy jest komenda do skanowania"""
22     print("[ZAP Scanner] Przetwarzam wyjście:", string)
23
24     lower_string = string.lower()
25     trigger_phrases = [
26         "zeskanuj strone",
27         "skanuj strone",
```

Wyznacza on m.in. triggery - frazy, w przypadku znalezienia których wykona konkretne polecenie, w tym wypadku, wykonanie skanu

Wywołanie zapytania w kierunku modelu LLM

Niestety, tutaj nasza passa sukcesów się kończy:



Model nie chce wysłać zapytania do skanera, zamiast tego dopytuje o szczegóły zadania - tak jakbyśmy nie skonfigurowali go odpowiednio :/ Niestety, pomimo godzin troubleshootingu, próbowania edytowania extensions, używania różnych frameworków do stawiania serwera, i różnych konfiguracji (jedna vs kilka maszyn, adresy lokalne, publiczne), nie udało się doprowadzić zadania do skutku :(