Bezpieczeństwo Aplikacji Internetowych

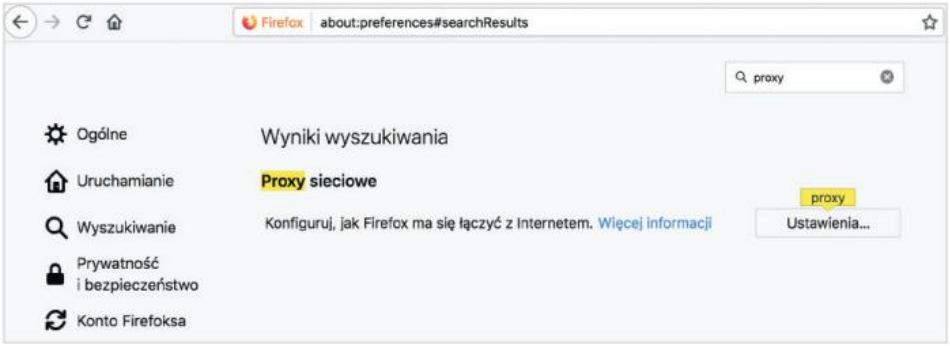
Podstawy Burp Suite Community Edition

CZYM JEST BURP SUITE?

Burp to jedno z dostępnych na rynku narzędzi typu proxy HTTP. Pozwala na przechwytywanie zapytań generowanych przez przeglądarki WWW oraz inne oprogramowanie, wykorzystujące protokół HTTP. Przechwycenie zapytań za pomocą proxy jest jednoznaczne z możliwością ich dowolnej modyfikacji.

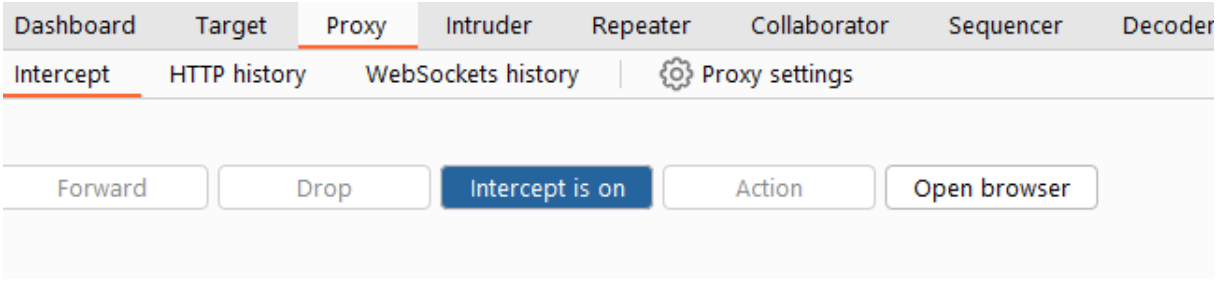
KONFIGURACJA ŚRODOWISKA PRACY

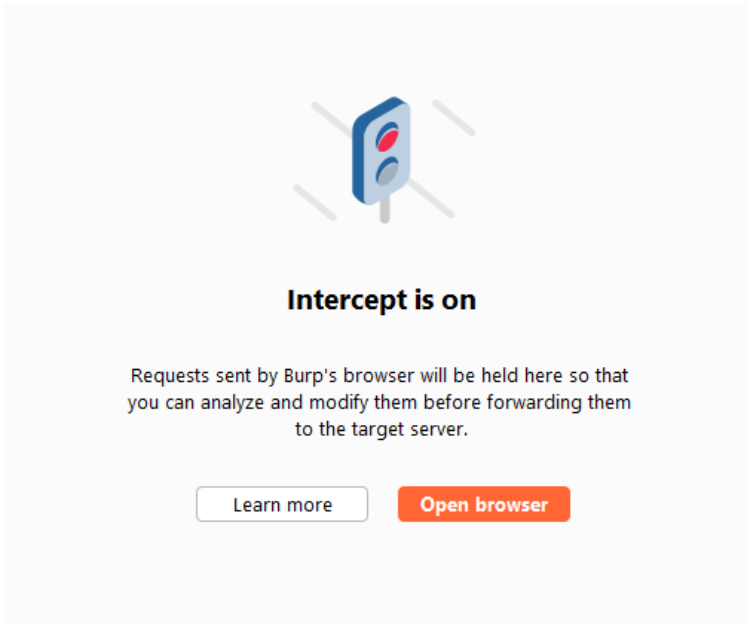
Przeglądarki WWW takie jak Firefox, Chrome czy Internet Explorer nie są domyślnie skonfigurowane do pracy z narzędziami typu Burp. Na potrzeby wykładu skonfigurujemy przeglądarkę Firefox, by zapytania w niej generowane były przesyłane do Burpa, a dopiero później „w świat". W tym celu należy przejść do menu przeglądarki, a następnie wybrać preferencje / opcje. Można też wpisać w pasek przeglądarki adres about:preferences.



MODYFIKOWANIE ZAPYTAŃ HTTP

Aby móc przechwycić zapytanie HTTP należy przejść do zakładki Proxy -> Intercept. Następnie należy uruchomić interceptor.





W kolejnym kroku musimy udać się do przeglądarki (w naszym przypadku firefox) i wpisać adres, pod którym znajduję się strona, którą będziemy sprawdzać pod kątem bezpieczeństwa: ***http://training.securitum.com/burpstarter/***

Uzupełniamy adres email i wysyłamy formularz. W tym momencie nasz Burp powinien przechwycić zapytanie.

Request to http://training.securitum.com:80 [45.56.85.138]

Forward | Drop | Intercept is on | Action | Open browser

Pretty Raw Hex

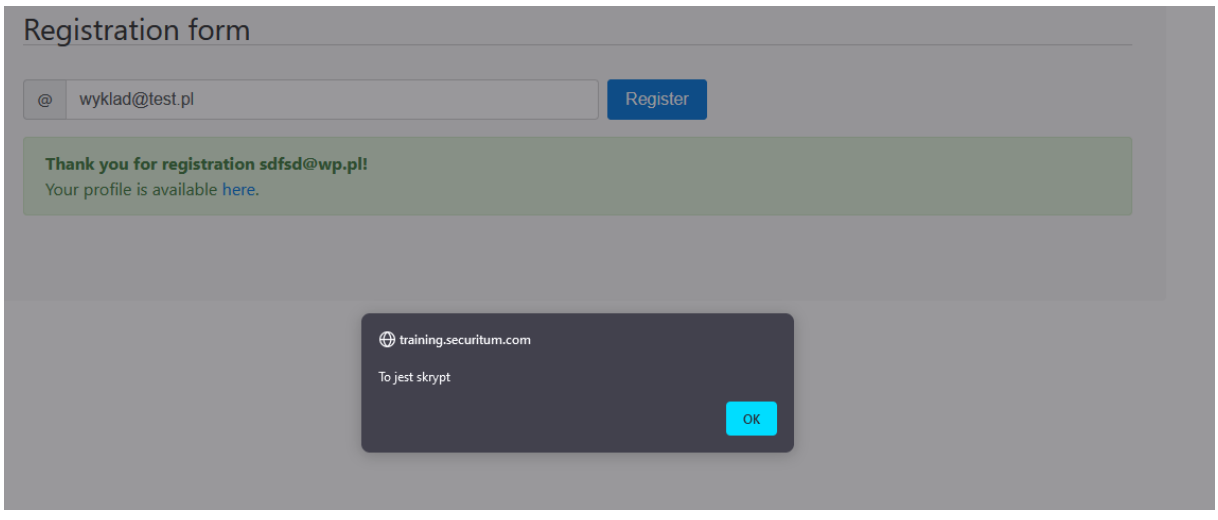
```
1  POST /burpstarter/index.php HTTP/1.1
2  Host: training.securitum.com
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: pl,en-US;q=0.7,en;q=0.3
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 22
9  Origin: http://training.securitum.com
10 Connection: close
11 Referer: http://training.securitum.com/burpstarter/index.php
12 Cookie: PHPSESSID=345497
13 Upgrade-Insecure-Requests: 1
14
15 email=wyklad%40test.pl
```

W tym momencie możemy edytować nasze zapytanie po czym, aby puścić je dalej należy wybrać opcję „Forward”. Wstrzykniemy zatem do parametru email prosty skrypt wyświetlający alert na stronie.

Edited request

Pretty Raw Hex \n

```
1  POST /burpstarter/index.php HTTP/1.1
2  Host: training.securitum.com
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: pl,en-US;q=0.7,en;q=0.3
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 49
9  Origin: http://training.securitum.com
10 Connection: close
11 Referer: http://training.securitum.com/burpstarter/index.php
12 Cookie: PHPSESSID=345497
13 Upgrade-Insecure-Requests: 1
14
15 email=wyk<script>alert("To jest skrypt")</script>
```



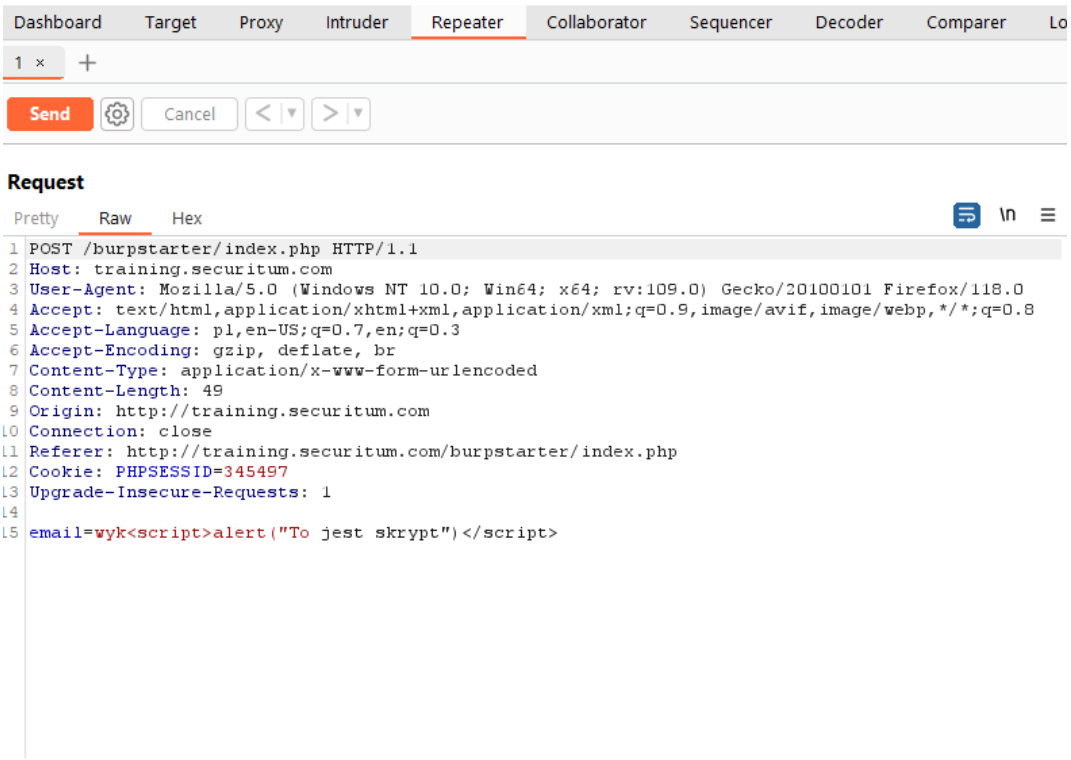
REPEATER

Do tej pory przechwytywaliśmy zapytania po to, by je podejrzeć lub zmodyfikować, a następnie przesłać do serwera. Cała operacja wymagała wyzwolenia zapytania w przeglądarce, a później wykonania wymaganych operacji w proxy. Co jednak, jeżeli chcemy określone zapytanie wysłać do serwera kilkukrotnie, bez konieczności ciągłego wspomagania się przeglądarką? Odpowiedź znajdziemy w zakładce repeater.

Przygotujmy środowisko pracy. W tym celu w zakładce http history kliknijmy prawym przyciskiem myszy na pozycję, która zawiera modyfikowane przez nas zapytanie POST**.**

108	http://detectportal.firefox.com	GET	/canonical.html
107	http://training.securitum.com	POST	/burpstarter/index.php
106	http://detectportal.firefox.com	GET	/success.txt?ipv6

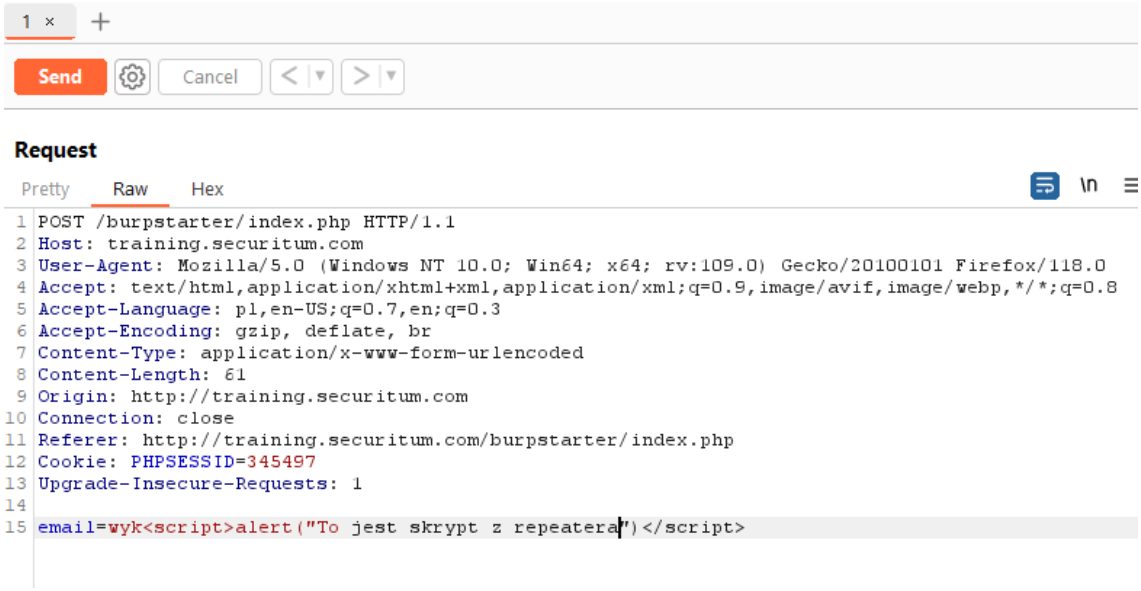
Następnie z menu wybierzmy opcję send to repeater. Po wykonaniu tej czynności możemy w Burpie przejść do zakładki repeter**.**



Główne okno tego narzędzia podzielone jest na dwie części. W pierwszej prezentowana jest treść zapytania, w drugiej będziemy mieli dostęp do odpowiedzi, jaką zwraca serwer.

Spróbujmy teraz ponownie zmodyfikować zapytanie, tym razem właśnie w Repeaterze. Edycja treści zapytania odbywa się w taki sam sposób jak poprzednio. Pole tekstowe pozwala nam modyfikować treść zapytania tak jak tekst w dowolnym edytorze. Aby sprawdzić, jak Repeater zachowuje się w praktyce, zmieńmy wartość parametru email na dowolny inny tekst. Akcję wysyłania do serwera wyzwalamy przyciskiem send.

Po chwili w drugiej części okna powinniśmy zauważyć odpowiedź, jaką wygenerował serwer (rysunek 20). Korzystając z pola wyszukiwania danych, możemy sprawdzić, czy na pewno wprowadzony przez nas ciąg znaków w parametrze email pojawił się w odpowiedzi.

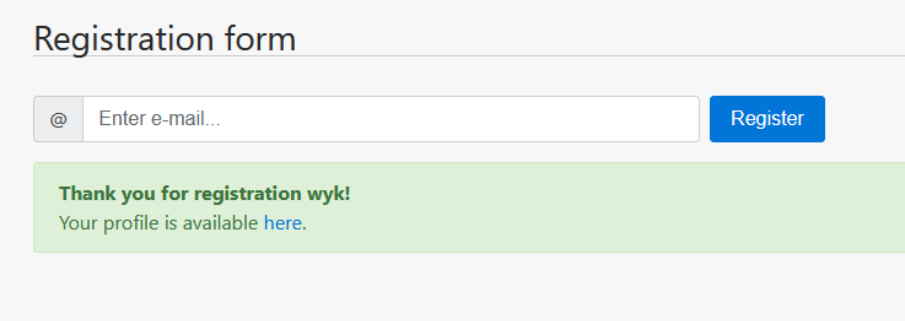



```
<form class="form-inline" method="POST" action="/burpstarter/index.php">
  <div class="input-group mb-2 mr-sm-2 mb-sm-0">
    <div class="input-group-addon">
      @
    </div>
    <input type="email" autocomplete="off" size=52 name="email" class="
    form-control" placeholder="Enter e-mail...">
  </div>
  <button type="submit" class="btn btn-primary">
    Register
  </button>
</form>
<div class="alert alert-success" role="alert">
  <strong>
    Thank you for registration wyk<script>
      alert("To jest skrypt z repeatera")
    </script>
    !
  </strong>
  <br/>
  Your profile is available <a href="index.php?user=2">
    here
```

INTRUDER

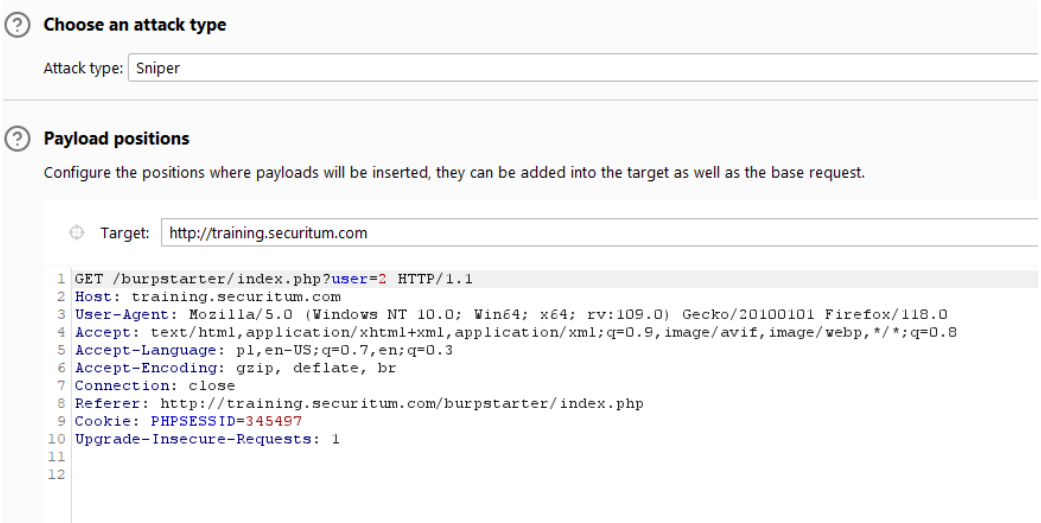
Wyobraźmy sobie, że musimy wielokrotnie wysłać określone zapytanie do serwera. Możemy to robić ręcznie, wspomagając się narzędziem Repeater, ale takie podejście jest możliwe do zaakceptowania tylko w sytuacji, gdy liczba zapytań do wykonania mieści się w granicach kilku lub kilkunastu. Co zrobić, jeżeli są ich tysiące?

Wróćmy jeszcze do naszej testowej aplikacji. Po wpisaniu poprawnego adresu e-mail i wybraniu opcji register wyświetla ona link do utworzonego profilu użytkownika



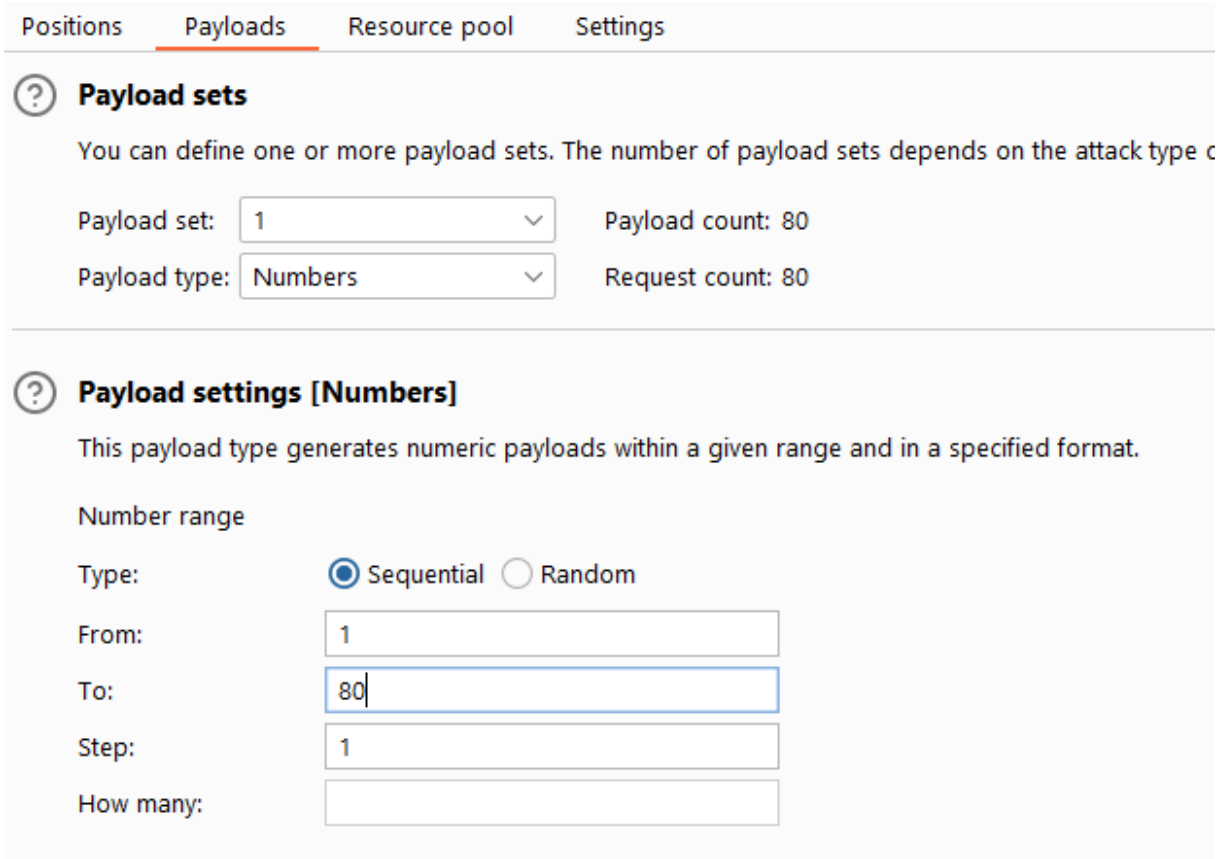
Po kliknięciu w link zostajemy przekierowani na adres:
http://training.securitum.com/burpstarter/index.php?user=2

Łatwo możemy dostrzec parametr user, który jest liczbą z czego możemy wywnioskować, że jest to swego rodzaju identyfikator zarejestrowanego użytkownika. W tym momencie możemy przeprowadzić atak podstawiający kolejne liczby w parametrze user, aby sprawdzić czy możemy dostać się do nieautoryzowanych danych. Wybieramy z historii adres wymieniony wyżej i wysyłamy go do Intrudera.



Zaznaczamy tutaj parametr, który chcemy modyfikować i wybieramy opcje Add (dostępna z prawej strony).

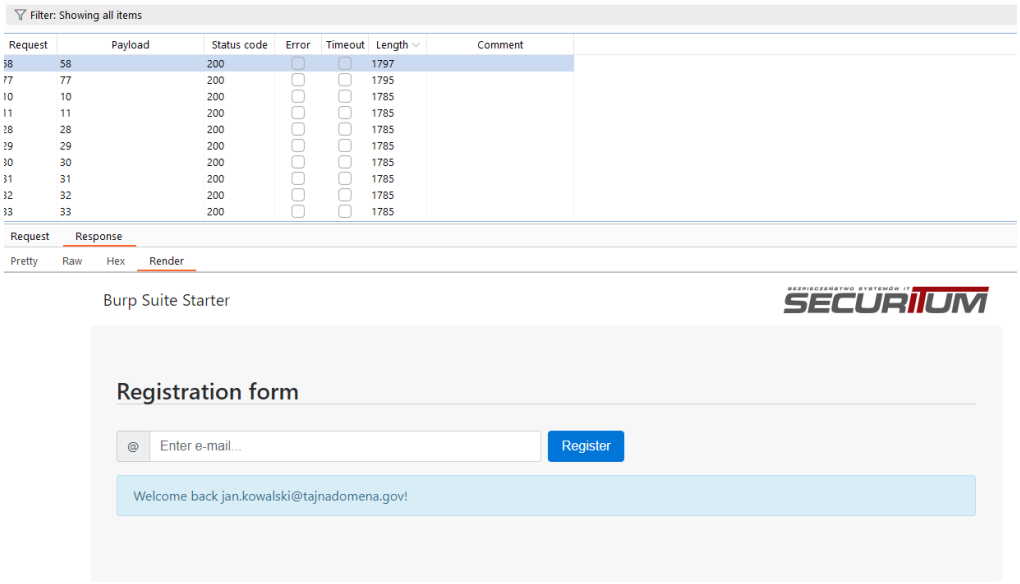
Następnie ustawiamy w zakładce payloads jakiego typu dane mają wystąpić w zapytaniu.

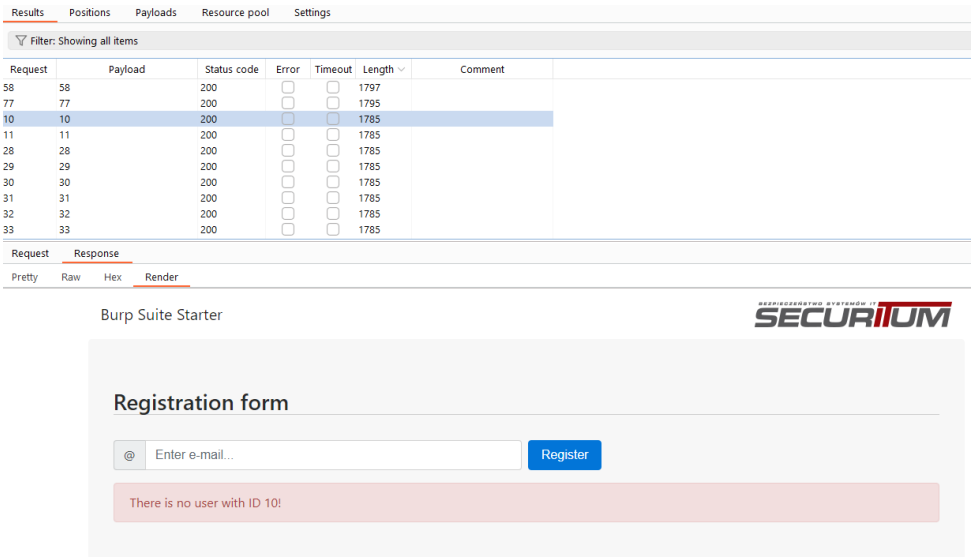


Na koniec wybieramy opcję start atak i czekamy na rezultaty ataku. Po przeprowadzonym ataku należy przeanalizować odpowiedzi serwera np. pod kątem rozmiaru zwróconej odpowiedzi.

Request	Payload	Status code	Error	Timeout	Length	Comment
58	58	200	☐	☐	1797	
77	77	200	☐	☐	1795	
10	10	200	☐	☐	1785	
11	11	200	☐	☐	1785	
28	28	200	☐	☐	1785	
29	29	200	☐	☐	1785	
30	30	200	☐	☐	1785	
31	31	200	☐	☐	1785	
32	32	200	☐	☐	1785	
33	33	200	☐	☐	1785	

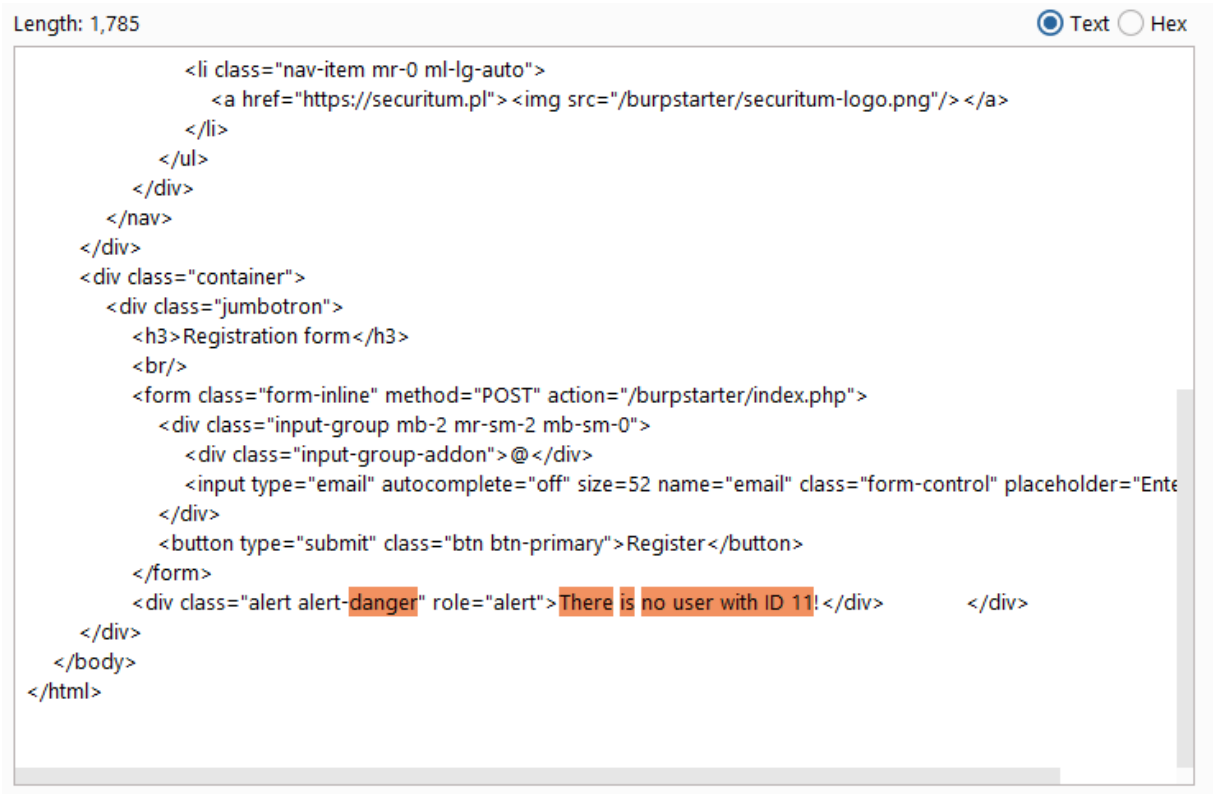
W naszym przypadku posortowaliśmy odpowiedzi serwera po rozmiarze odpowiedzi. Widzimy, że w przypadku id 58 oraz 77 rozmiar ten jest inny. Należy zatem przyjrzeć się temu czym odpowiedź dla tych id różni się od pozostałych.

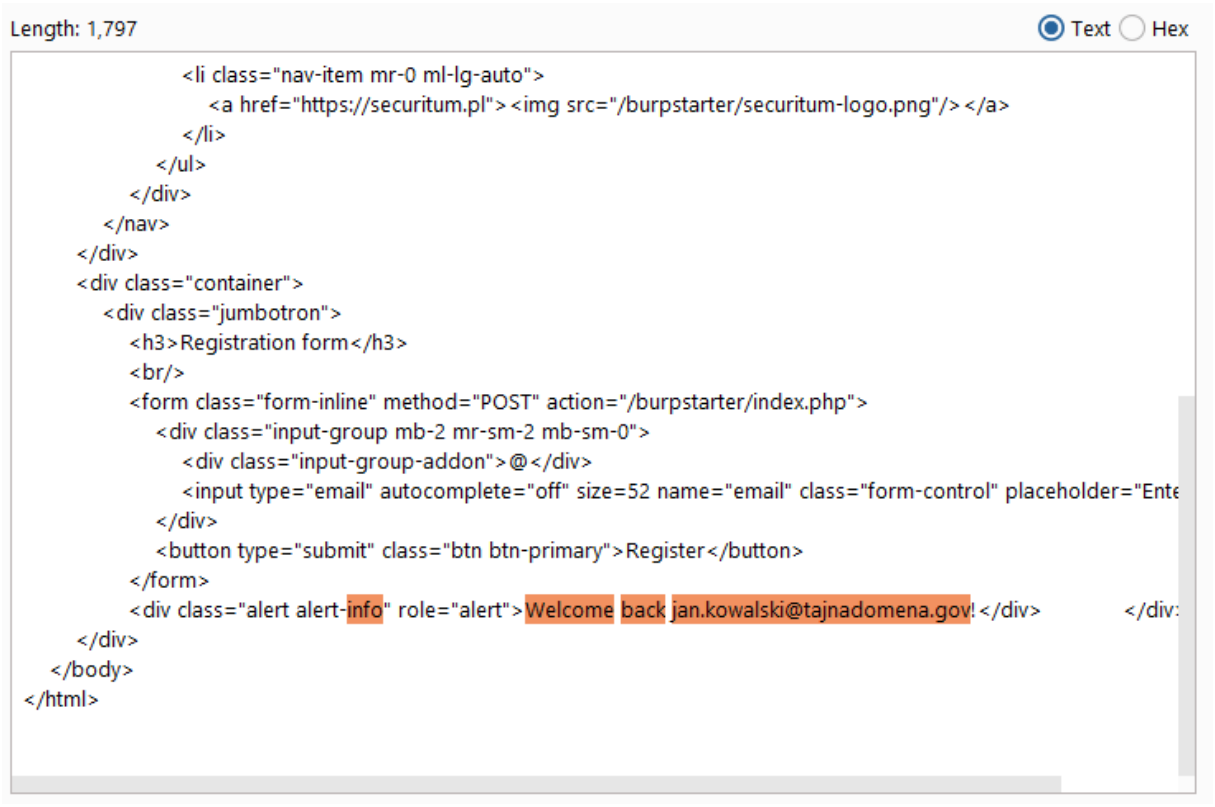




COMPARER

Narzędzie to pozwala nam porównać ze sobą dwa zapytania bądź odpowiedzi z serwera.



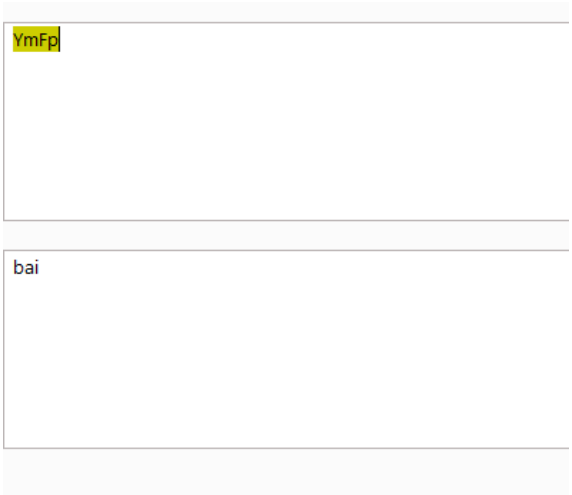


DECODER

Decoder w narzędziu Burp Suite jest używany do analizy i manipulacji danych zakodowanych w różnych formatach, które mogą być przesyłane między klientem a serwerem podczas testowania aplikacji webowej. Służy on do rozkodowywania i kodowania danych, co jest przydatne w wielu przypadkach podczas testowania penetracyjnego oraz analizy bezpieczeństwa aplikacji internetowych. Oto kilka zastosowań dekodera w narzędziu Burp Suite:

- Analiza żądań HTTP: Decoder pozwala na analizę zawartości żądań HTTP, włączając w to ciała żądań i parametry przesyłane jako część zapytania. Można używać go do zdekodowania zakodowanych danych, takich jak base64, aby zrozumieć, co jest przesyłane między klientem a serwerem.
- Testowanie wstrzykiwania danych: W trakcie testowania aplikacji webowej często próbuje się wstrzyknąć błędne lub złośliwe dane. Decoder może być używany do zakodowania takich danych, aby przekształcić je w odpowiedni format, który jest akceptowany przez aplikację.
- Analiza i modyfikacja ciasteczek (cookies): Decoder pozwala na analizę i manipulację ciasteczkami w przeglądarce. Można go użyć do zdekodowania zawartości ciasteczek, edycji ich wartości i ponownego zakodowania, zanim zostaną przesłane do serwera.
- Manipulacja parametrami URL: Czasami parametry przesyłane w adresach URL są zakodowane lub zaszyfrowane. Decoder umożliwia odkodowanie tych parametrów, ich modyfikację, a następnie ponowne zakodowanie przed przesłaniem żądania.
- Obsługa różnych formatów kodowania: Decoder obsługuje różne formaty kodowania, takie jak Base64, URL-encoding, HTML-encoding, Gzip i wiele innych. Dzięki temu można radzić sobie z różnymi rodzajami zakodowanych danych.

Poniżej przykład rozkodowania tekstu zakodowanego w base64.



LITERATURA

Bezpieczeństwo Aplikacji Webowych - Michał Bentkowski / Artur Czyż / Rafał 'bl4de' Janicki / Jarosław Kamiński Adrian 'vizzdoom' Michalczyk / Mateusz Niezabitowski / Marcin Piosek Michał Sajdak / Grzegorz Trawiński / Bohdan Widła - ISBN: 978-83-954853-2-9 - Kraków 2020