



# Bezpieczeństwo Aplikacji Web

01. Wstęp do bezpieczeństwa aplikacji web





# Bezpieczeństwo Aplikacji Web

## 01. Wstęp do bezpieczeństwa aplikacji web

### Spis treści

Komunikacja	4
HTTP	4
Logi komunikacji HTTP	6
Wysyłanie parametrów i danych	9
Modyfikowanie parametrów zapytania w przeglądarce	10
Modyfikowanie parametrów formularza w przeglądarce	12
Proxy	13
Używanie proxy do modyfikowania żądań	14
Używanie proxy do modyfikowania żądań	18
Log komunikacji proxy	20
Sekrety stron www	21
Automatyczny skan w OWASP ZAP	22
Atak siłowy na parametr	24

## Komunikacja

Większość funkcji aplikacji webowych jest realizowana za pomocą komunikacji pomiędzy klientem a serwerem. **Klient** to program, który pobiera dane z serwera (i czasami wysyła żądania z danymi do przetworzenia). **Serwer** to zdalna aplikacja (bądź urządzenie), które serwuje dane i obsługuje część logiki aplikacji. Konfiguracja składająca się z obydwu powyższych nazywa się **aplikacją webową**.

Klientem może być dowolny program, który wysyła żądania poprzez sieć i prezentuje rezultat użytkownikowi. Zazwyczaj to przeglądarka pełni rolę klienta dla wielu aplikacji webowych, ale czasami używa się dedykowanych klientów. Jeżeli chodzi o serwer, to jest wiele różnych rozwiązań. Najbardziej popularnymi serwerami HTTP są Apache, Nginx, LiteSpeed, ale istnieją również inne rozwiązania (na przykład zbudowane w języku Java na bazie platform takich jak Spring Boot). Serwer zazwyczaj również uruchamia dodatkowe oprogramowanie, jak interpreter PHP albo baza danych, które dodaje dodatkową funkcjonalność.

W tym kursie klientem będzie przeglądarka Mozilla Firefox. Serwerem będzie „czarna skrzynka”, którą będziemy próbować identyfikować i wykrywać podatności.

## HTTP

HTTP jest skrótem od „Hypertext Transfer Protocol” (protokół transferu hipertekstu) i jest najczęściej używanym standardem jeżeli chodzi o obsługę sieciowej wymiany danych.

Na przykład, w momencie otwierania strony logowania do Cyberskillera wysyłane jest żądanie podobne do poniższego:

```
GET /account/login HTTP/1.1
Host: portal.cyberskiller.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13;
rv:72.0) Gecko/20100101 Firefox/72.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Thu, 13 Feb 2020 23:16:13 GMT
If-None-Match: „5e45d8bd-6eb”
Cache-Control: max-age=0
```

Jak widać, przeglądarka oprócz samego adresu strony, którą zostanie wyświetlona, przesyła dużo innych danych.

Pierwsza linia określa trzy rzeczy: **metodę HTTP, adres URL oraz wersję HTTP**. Metoda HTTP to jedna z dostępnych metod, które serwer może obsłużyć. Najczęściej używanymi metodami są GET (do pobierania danych bądź zasobów) oraz POST (do wysyłania danych lub formularzy).

HTTP wspiera następujące metody (wg specyfikacji RFC 7231 i RFC 5789):  
GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

**Adres URL**, który jest wysyłany, może być bezwzględny bądź względny do nagłówka Host, który znajduje się w następnej linii. Pełny URL wyglądałby więc następująco:  
<https://portal.cyberskiller.com/account/login>.

W następnych liniach są wypisywane **nagłówki**. Są to dodatkowe dane opisujące klienta wysyłającego żądanie oraz samo żądanie. Nagłówki mają następującą postać:  
Nazwa-Nagłówka: Wartość

Opcjonalnie wysłane może być również **ciało żądania**, które jest dodawane po pustej linii za nagłówkami. Ciało żądania będzie wysyłane m.in. w przypadku wysyłania formularzy (np. formularzy logowania). Metoda GET nie obsługuje ciał żądań.

Na powyższe żądanie serwer odpowie następująco:

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 17 Feb 2020 06:25:14 GMT
Content-Type: text/html
Content-Length: 1771
Last-Modified: Thu, 13 Feb 2020 23:16:13 GMT
Connection: keep-alive
ETag: „5e45d8bd-6eb”
Accept-Ranges: bytes

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>CyberSkiller</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-
  -scale=1">
    <link rel="apple-touch-icon" sizes="180x180" href="/assets/
  favicon/apple-touch-icon.png">
  ...
  
```

W odpowiedzi, w ciele żądania, będzie zwrócona strona w formacie HTML. Przeglądarka przetwarza potem tę odpowiedź i wyświetla stronę w czytelny dla użytkownika sposób. Jeżeli dodatkowe zasoby są potrzebne (na przykład obrazki), to przeglądarka wykonuje dodatkowe żądania do serwera, aby je pobrać.

W pierwszej linii serwer wypisuje dwie dane - **używaną wersję HTTP** oraz **kod statusu odpowiedzi**. Kod statusu może być wartością z zakresu od 100 do 599. Kategoryzuje się je względem pierwszej cyfry i istnieją następujące kategorie:

Kategoria	Opis
1xx	kody informacyjne (opisujące stan połączenia)
2xx	kody sukcesu (udane przetworzenie żądania)
3xx	kody przekierowania (informacja o przeniesieniu zasobu)
4xx	błędy klienta (niepoprawnie sformułowane żądanie)
5xx	błędy serwera (problemy wewnętrzne aplikacji)

Pełna lista dostępna jest w dokumentacji protokołu HTTP (np. na stronie <https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>).

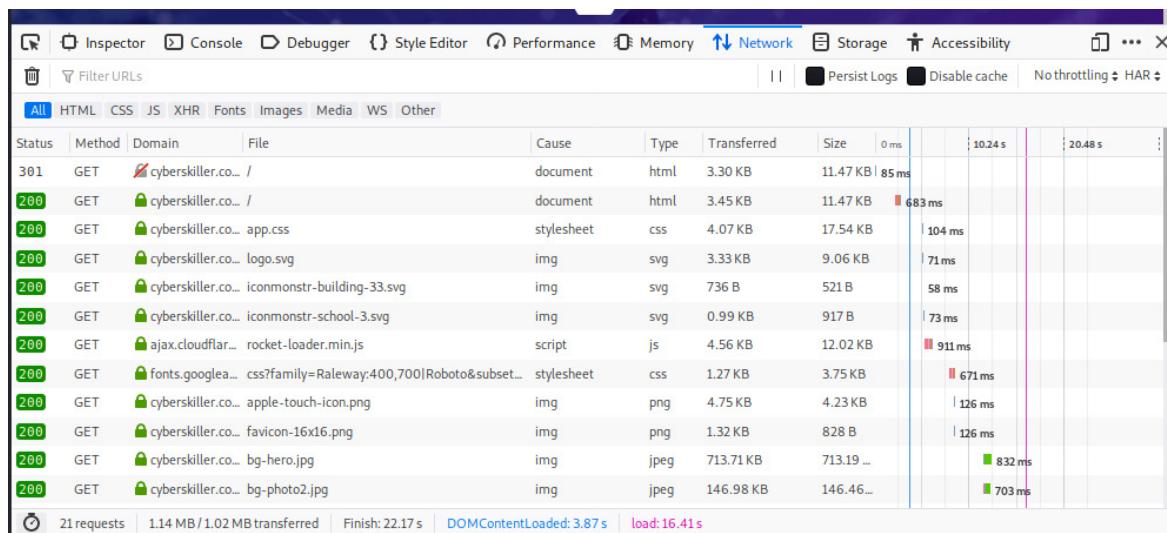
Po kodzie statusu wysyłane są nagłówki z dodatkowymi danymi o serwerze i odpowiedzi.

Potem, na końcu, wysyłane jest ciało odpowiedzi (dane, które zostały zażądane).

## Logi komunikacji HTTP

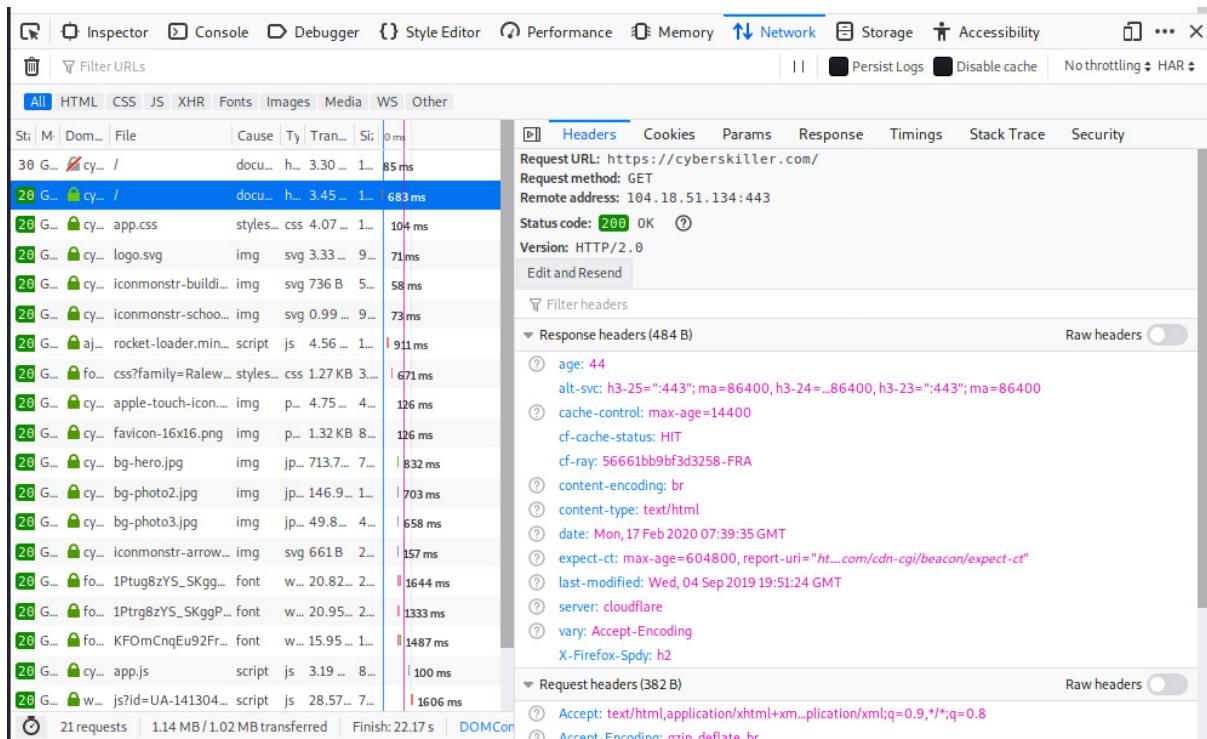
Każda strona internetowa i aplikacja webowa może wysyłać różne dane do serwera, w zależności od potrzeb. Oprócz tego zwracane dane mogą zawierać dużo więcej informacji, niż te, które są prezentowane użytkownikowi. Zarówno żądania, jak i odpowiedzi, mogą być wektorem ataku - może się okazać, że serwer przetwarza dane w taki sposób, że możliwe jest pominięcie „blokad”, które istnieją w kodzie JavaScript strony, ale nie na serwerze. Do analizy, jakie dane są wysyłane do serwera, można użyć przeglądarek.

Przeglądarki zawierają logi komunikacji dla celów debugowania. Logi można wyświetlić w przeglądarce Mozilla Firefox naciskając Ctrl + Shift + E, albo w Chrome naciskając F12 i otwierając kartę „Network”.



Ryc. 1. Logi sieciowe w Mozilla Firefox.

Logi są zbierane tylko wtedy, kiedy to narzędzie jest otwarte. Aby pojawiły się wpisy, należy załadować ponownie stronę.



The screenshot shows the Network tab of a browser's developer tools. It lists 21 requests made to the URL <https://cyberskiller.com/>. The requests include various files like CSS, JS, and images. The right panel provides detailed information for the selected request (the main HTML file). It shows the Request URL, Request method (GET), Remote address (104.18.51.134:443), Status code (200 OK), Version (HTTP/2.0), and Response headers (including age, alt-svc, cache-control, cf-cache-status, cf-ray, content-encoding, content-type, date, expect-ct, last-modified, server, vary, and accept-encoding). It also shows Request headers (Accept and Accept-Encoding).

Ryc. 2. Badanie żądania i odpowiedzi dla dokumentu html.

Aby zbadać załadowaną stronę cyberskiller.com podczas gdy narzędzie „Sieć” jest otwarte, należy odświeżyć stronę.

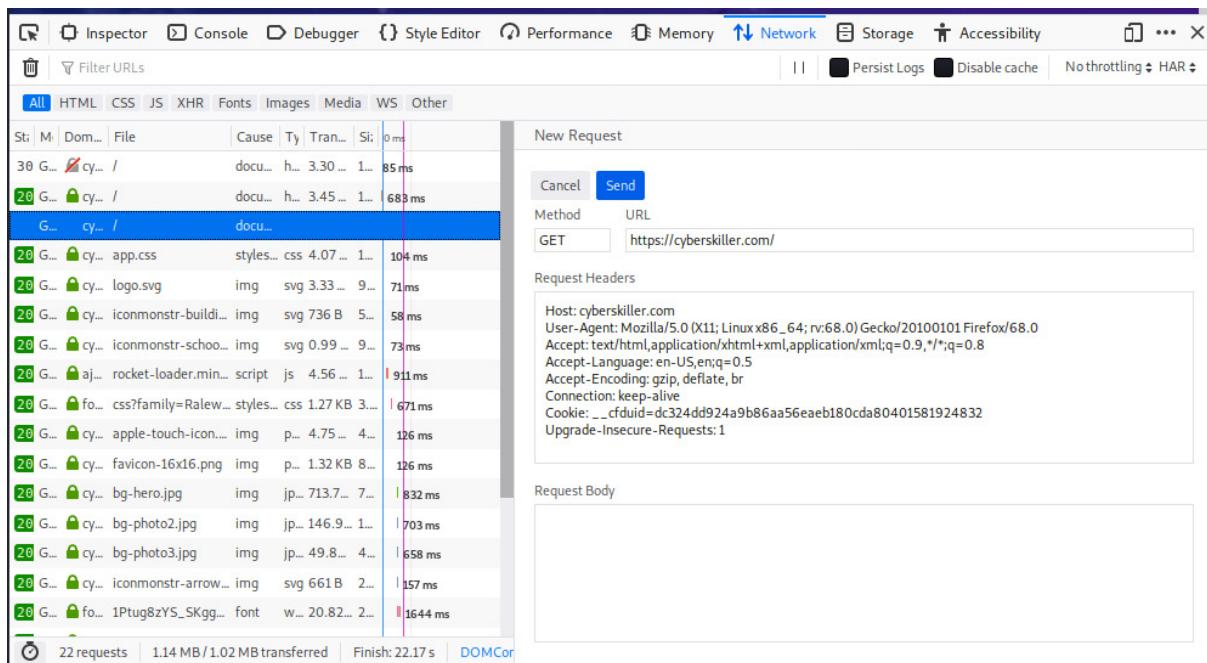
W tych logach można zbadać każde żądanie, które zostało wysłane, i odpowiedź, która została odebrana od momentu załadowania strony. Można kliknąć na pojedynczy element (na przykład dokument html „/”) i zobaczyć dane powiązane z tym zdarzeniem.

W tym oknie wylistowane są nagłówki żądania i odpowiedzi, a także inne szczegóły, które można zbadać - ciasteczka, parametry żądania, odpowiedź.

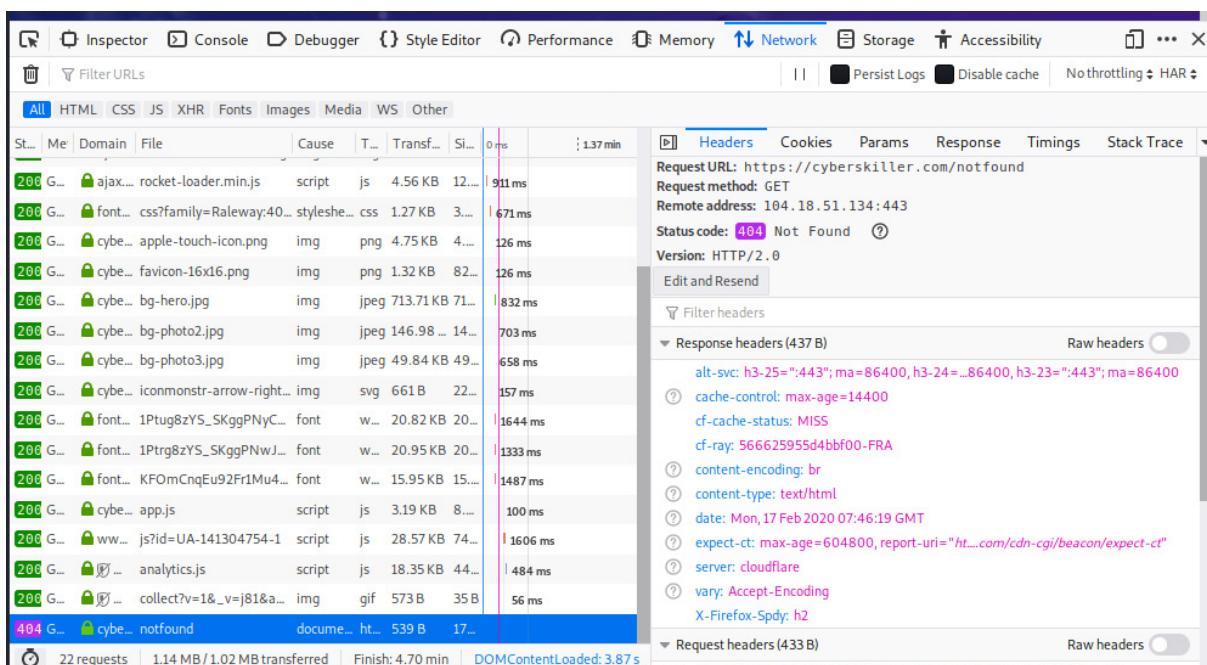
Poprzez zbadanie zdarzenia, można zauważyc, że serwer wysłał w odpowiedzi nagłówek „server: cloudflare”. To przykład informacji, która może się przydać podczas badania możliwych ataków na aplikację webową. W tym przypadku użycie technologii Cloudflare oznacza, że np. atak DDOS byłby nieefektywny.

Żądanie może zostać również edytowane i wysłane ponownie w przeglądarce.

Modyfikacji może podlegać metoda, adres URL, nagłówki i ciało wybranego żądania. Przykładowo można zmienić URL na „<https://cyberskiller.com/notfound>”, a następnie wysłać żądanie przyciskiem „Wyślij” („Send”).

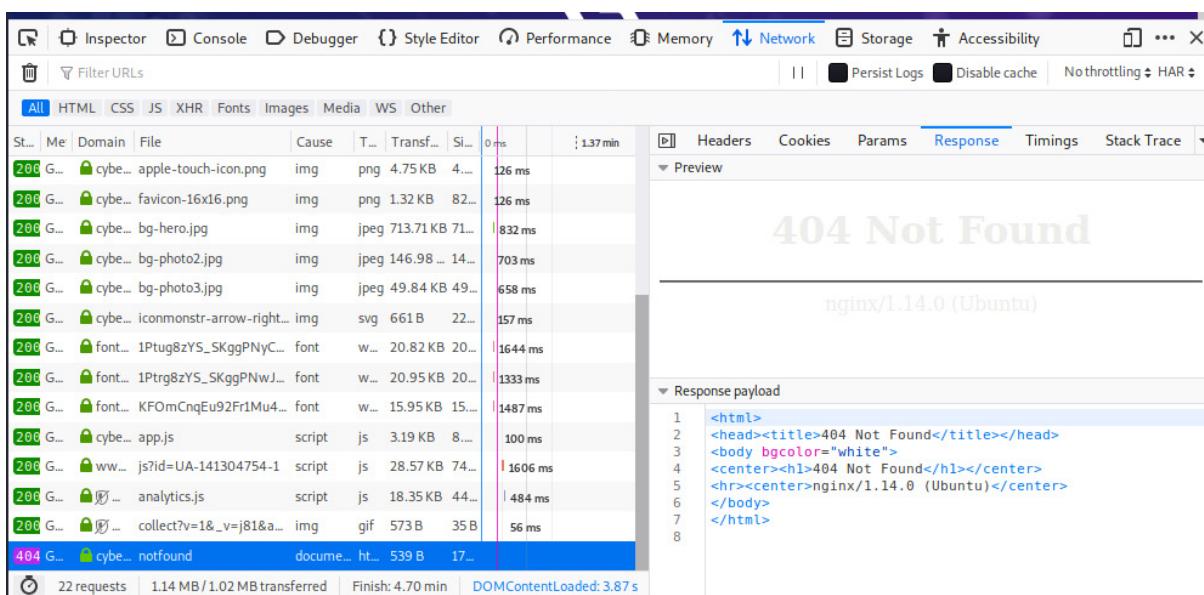


Ryc. 3. „Edytuj i wyślij ponownie” („Edit and Resend”) w Mozilla Firefox.



Ryc. 4. „Edytuj i wyślij ponownie” („Edit and Resend”) w Mozilla Firefox.

Sam błąd nie mówi - jest to błąd, który może wystąpić w wielu przypadkach. Interesujący jest natomiast zapis „nginx/1.14.0 (Ubuntu)” w odpowiedzi. To jest oprogramowanie serwera - serwer **nginx** w wersji **1.14.0** działający w systemie **Ubuntu**. Te dane już mogą zostać użyte do znalezienia i wykorzystania znanych podatności, które może zawierać oprogramowanie używane na serwerze.



Ryc. 5. Podgląd odpowiedzi 404.

Oprócz tego może się okazać, że niektóre aplikacje webowe mają nieprawidłowe mechanizmy zabezpieczeń. Takie błędy w zabezpieczeniach mogą pozwalać na wysyłanie zmodyfikowanych żądań, które zostaną obsłużone tak, aby dać dostęp do zasobów, do których użytkownik nie powinien mieć dostępu.

## Wysyłanie parametrów i danych

Standard HTTP dostarcza dwóch metod wysyłania danych w żądaniu:

- poprzez łańcuch zapytania („query string”),
- poprzez ciało żądania („request body”).

**Łańcuch zapytania (query string)** to parametry, które są dodawane do adresu URL. Początek tego łańcucha jest oznaczany poprzez **?** (znak zapytania) i każdy parametr jest oddzielany za pomocą **&** (ampersand). Parametry wypisywane są tak jak we wzorze:

?param1=wartość&param2=innawartość

Serwer odbierze dwa parametry: param1 (z wartością „wartość”) oraz param2 (z wartością „innawartość”).

Przykładowy adres URL z parametrem „id” ustawionym na „1” będzie wyglądał następująco:

<https://httpbin.org/get?id=1>

Klient zażąda zasobu <https://httpbin.org/get> z parametrem **id=1**. Zazwyczaj zmiana wartości parametru nie powoduje błędu. Parametry łańcucha zapytania mogą być testowane w poszukiwaniu ukrytych zasobów.

**Ciało żądania** (treść żądania) to dodatkowa zawartość, która może zostać wysłana z żądaniem. Jest dodawana po nagłówkach i jej postać zależy od nagłówka **Content-Type**.

Zazwyczaj formularze są wysyłane z nagłówkiem Content-Type: application/x-www-form-urlencoded. Użycie tego typu zawartości (ang. *content type*) pozwala na użycie tego samego formatu, co w łańcuchu zapytania (ale bez znaku zapytania na początku).

Przykładowe żądanie może wyglądać następująco:

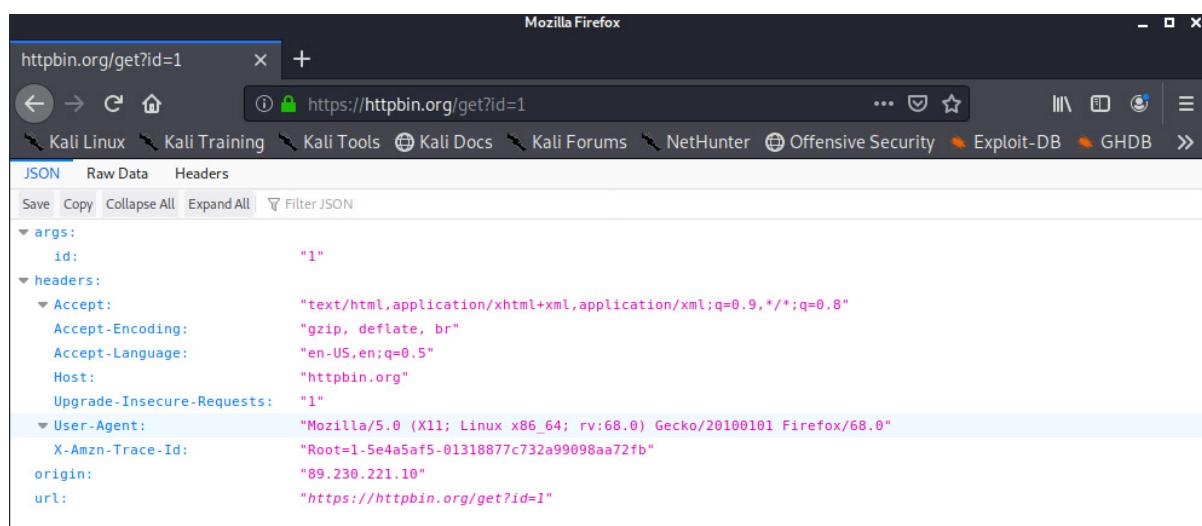
```
POST /post HTTP/1.1
Host: httpbin.org
Content-Type: application/x-www-form-urlencoded
Accept: */*
Cache-Control: no-cache
Host: httpbin.org
Accept-Encoding: gzip, deflate, br
Content-Length: 30
```

```
login=username&password=secret
```

W tym żądaniu wysłane zostały dwa parametry - login (z wartością „username”) i password (z wartością „secret”). Jak widać, nagłówek Content-Type jest obecny i użyta metodą HTTP jest POST.

## Modyfikowanie parametrów zapytania w przeglądarce

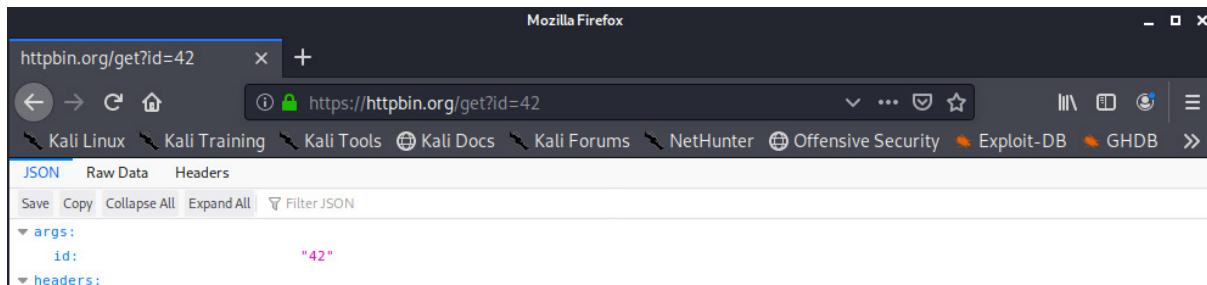
Modyfikacja parametrów może odbywać się z użyciem przeglądarki. Ta umiejętność będzie kluczowa do przeprowadzenia prób odnalezienia wektorów ataku i złamania zabezpieczeń.



Ryc. 6. Przykładowa odpowiedź <https://httpbin.org/get?id=1>.

Na początek zostanie wyświetlona strona <https://httpbin.org/get?id=1>.

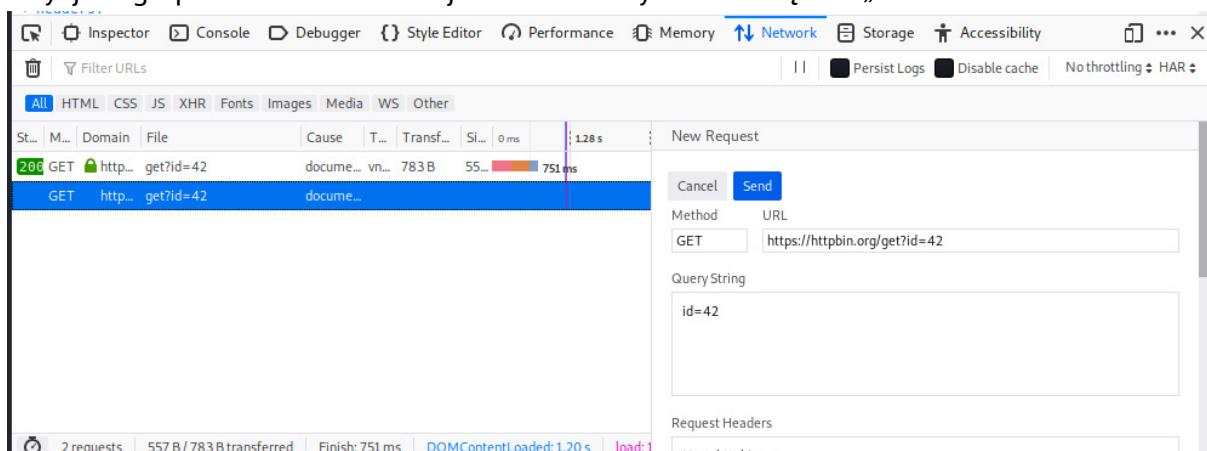
Ta strona będzie odzwierciedlać wysłane parametry i nagłówki w formacie JSON. Przeglądarki zazwyczaj formatują JSON tak, żeby był bardziej czytelny (tak jak na obrazku). Na podstawie tej strony można obserwować, jak modyfikacja żądania wpływa na dane odebrane przez serwer.



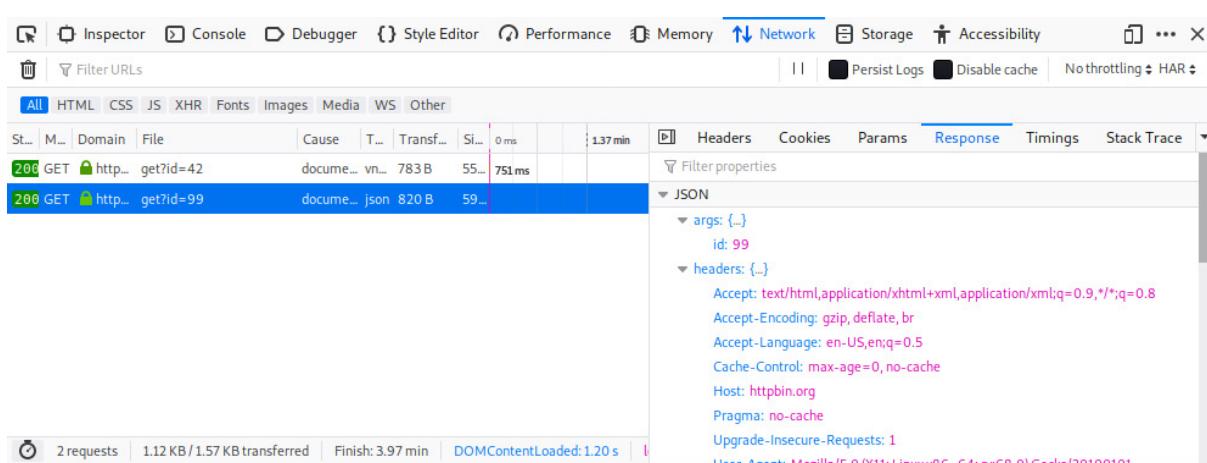
Ryc. 7. Przykładowa odpowiedź <https://httpbin.org/get?id=42>.

Parametr `id=1`, który został dołączony do URL, został wysłany do serwera i poprawnie odebrany, co można zaobserwować w sekcji „args”. Tam jest „`id`” z wartością „1”. Zmiana tego parametru spowoduje również, że inne dane zostaną odebrane przez serwer (np. po zmianie „`id`” na „42”).

Edycja tego parametru możliwa jest także z użyciem narzędzia „Sieć”.



Ryc. 8. Edycja parametryzowanego żądania.



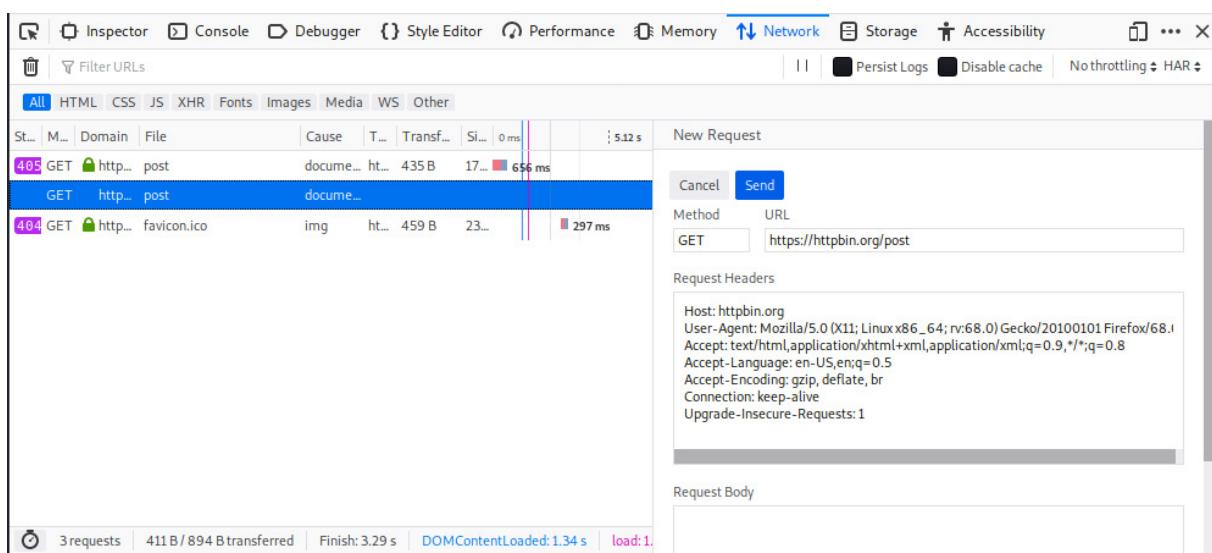
Ryc. 9. Podgląd odpowiedzi na edytowane żądanie.

Podczas edycji żądania można zauważyc, że w edytorze pojawiła się nowa sekcja - Parametry żądania (Query String). Użycie tego pola jest jednoznaczne z edycją parametrów w adresie URL. Przykładowo, można zmienić id=42 na id=99 i nacisnąć „Wyślij” („Send”).

W odpowiedzi na żądanie zostanie wysłany zaktualizowany parametr „id”.

## Modyfikowanie parametrów formularza w przeglądarce

Narzędzie „Sieć” może być również użyte do modyfikowania parametrów formularza. W przypadku braku formularza można go „symulować” używając odpowiednio przygotowanego żądania.



Ryc. 10. Edycja żądania GET <https://httpbin.org/post>.

Mając otwarte narzędzie „Sieć”, należy przejść do <https://httpbin.org/post> i edytować pierwsze żądanie.

Aby żądanie było żądaniem wysłania formularza, należy zmienić poniższe rzeczy:

- Metoda GET musi zostać zmieniona na **POST**
- Musi zostać dodany nagłówek

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

- W treści żądania należy dodać parametry, na przykład:  
`login=user&password=pass`

Następnie należy wcisnąć przycisk „Wyślij” („Send”). Powinien pojawić się nowy wpis w narzędziu „Sieć”, tym razem ze statusem „200 OK”.

Odpowiedź można otworzyć w podglądzie w narzędziu „Sieć” albo dwukrotnie kliknąć na zdarzenie, aby otworzyć odpowiedź w nowej karcie.

W odpowiedzi serwera w sekcji nazwanej „form” powinny być dwa parametry - login oraz password z wartościami, które zostały wysłane w żądaniu.

Method: POST URL: https://httpbin.org/post

Request Headers:

```
Host: httpbin.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
```

Request Body:

```
login=user&password=pass
```

Ryc. 11. Zaktualizowane żądanie https://httpbin.org/post

Narzędzie „Sieć” pozwala również sprawdzić parametry żądania w celu weryfikacji, jakie dane zostały przesłane do serwera.

Kali Linux Kali Training Kali Tools

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
args: {}
data: {}
files: {}
form:
  login: "user"
  password: "pass"
```

Ryc. 12. Wynik wysłania parametrów formularza.

Inspector Console Debugger Style Editor Performance Memory Network Storage Accessibility ...

All HTML CSS JS XHR Fonts Images Media WS Other

St...	M...	Domain	File	Cause	T...	Transf...	Si...	0 ms	1.37 min
405	GET	http...	post	docume...	ht...	435 B	17...	656 ms	
404	GET	http...	favicon.ico	img	ht...	459 B	23...	297 ms	
200	P...	http...	post	docume...	json	984 B	75...		

Headers Cookies Params Response Timings Stack Trace

Filter request parameters

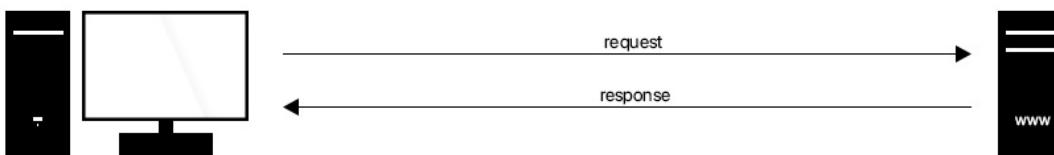
Form data

```
login: user
password: pass
```

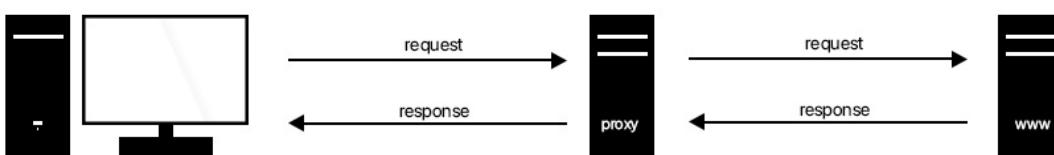
Ryc. 12. Wyświetlanie wysłanych parametrów żądania.

## Proxy

Kiedy używany jest serwer proxy, cały ruch sieciowy jest przesyłany przez ten serwer. Proxy wykonuje żądania w imieniu klienta podłączonego do niego i przekazuje odpowiedzi z innych serwerów do klienta - można powiedzieć, że proxy pośredniczy komunikację. Proxy może być używane do filtrowania szkodliwych albo niechcianych stron. W firmach często używany jest firewall, który działa jak proxy i blokuje dostęp do niedozwolonych stron i zasobów.



Ryc. 13. Standardowa komunikacja z serwerem.



Ryc. 14. Pośredniczona komunikacja z serwerem.

Proxy może być też użyte do przeprowadzenia ataku Man-In-The-Middle. Jeżeli atakujący zmusi użytkownika do użycia proxy, to wtedy będzie mógł zobaczyć cały ruch sieciowy danego użytkownika (żądania, odpowiedzi, adresy).

Ruch HTTPS nie może być przeczytany i modyfikowany przez proxy. Szyfrowanie komunikacji i system zaufanych certyfikatów uniemożliwia podgląd żądań i odpowiedzi (w których mogą być hasła), a także modyfikowanie ich przez oprogramowanie „przechwytyjące” ruch sieciowy.

Użytkownik może jednak pozwolić proxy, aby widziało ruch HTTPS - musi wtedy zainstalować certyfikat proxy na swoim urządzeniu. Proxy będzie używała własnego certyfikatu w ramach użycia komunikacji HTTPS, ale ten certyfikat nie jest domyślnie zaufany.

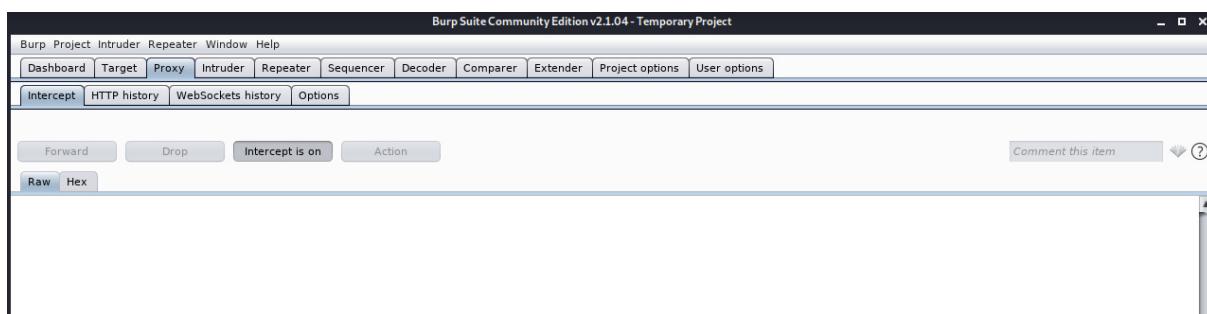
Kiedy proxy próbuje zobaczyć ruch HTTPS, wtedy u użytkownika pojawia się błąd podczas wchodzenia na znane strony takie jak google.com. Proxy nie może rozszyfrować ruchu HTTPS, więc będzie „udawało” oryginalną stronę. Przeglądarka zobaczy, że certyfikat jest błędny i niezaufany. Kiedy użytkownik w przeglądarce zaufa temu certyfikatowi, wtedy proxy będzie mogło przeczytać całą komunikację, ponieważ będzie w stanie ją rozszyfrować.

## Używanie proxy do modyfikowania żądań

Jest wiele dostępnych rozwiązań do tworzenia własnego serwera proxy, który jest w stanie przechwytywać ruch oraz modyfikować żądania i odpowiedzi. Jednym z nich jest proxy dołączone do Burp Suite Community Edition.

Po otwarciu Burp Suite wystarczy stworzyć nowy tymczasowy projekt (temporary project) i przejść do karty „Proxy”.

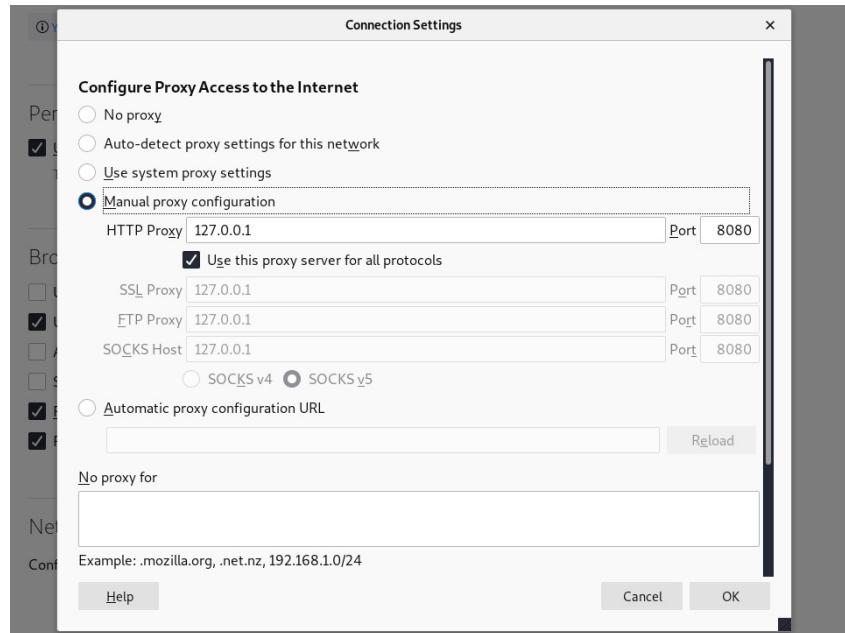
Za każdym razem, gdy wykonywane jest żądanie, będzie ono zatrzymane w proxy Burp Suite.



Ryc. 15. Pusta karta „Proxy” w Burp Suite.

Przeglądarka bądź system musi zostać skonfigurowany tak, aby ruch sieciowy przechodził przez proxy. W Mozilla Firefox należy przejść do Preferencji („Preferences”), przewinąć na sam dół do sekcji ustawień „Sieć” („Network Settings”), a następnie kliknąć na „Ustawienia...” („Settings...”).

Należy wybrać „Ręczna konfiguracja serwerów proxy” („Manual proxy configuration”).



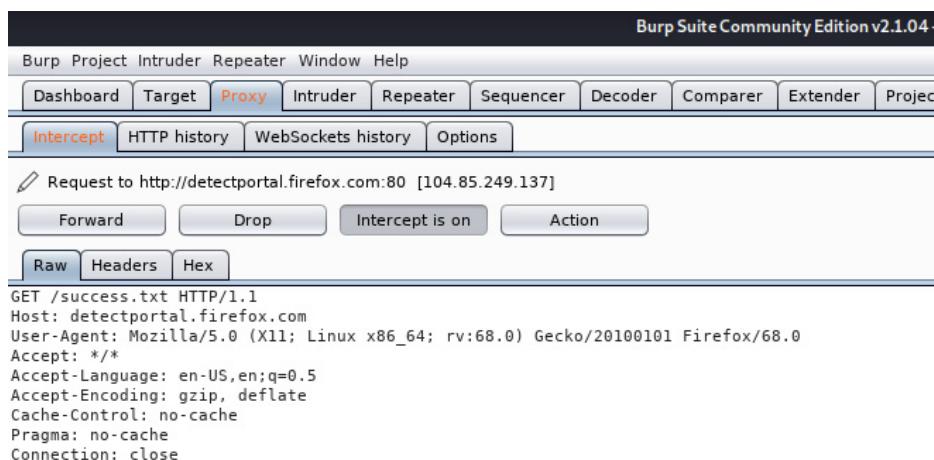
Ryc. 16. Konfiguracja proxy w Mozilla Firefox.

Jako adres proxy HTTP użyć 127.0.0.1 (albo localhost), a jako port 8080. Ten adres zawsze wskazuje na obecny komputer. Port 8080 natomiast jest domyślnym, pod którym Burp Suite tworzy serwer proxy.

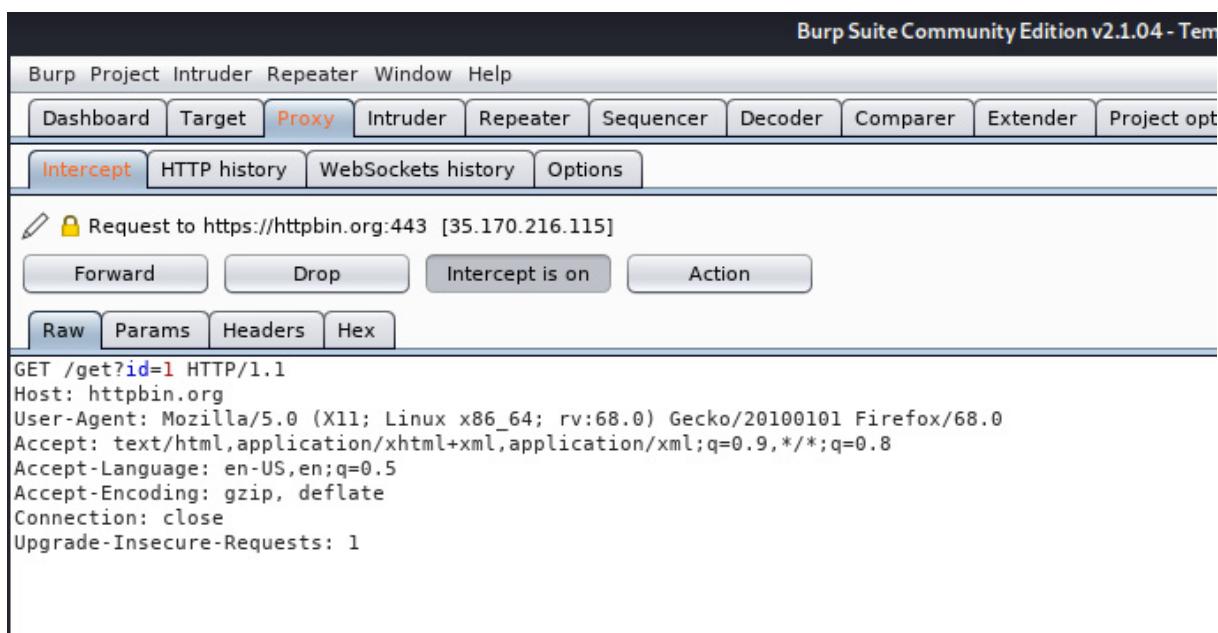
Można również zaznaczyć „Użyj tego serwera proxy także dla FTP i HTTPS” („Use this proxy server for all protocols”) i zatwierdzić za pomocą OK.

Próba przejścia do <https://httpbin.org/get?id=1> powinna zostać zatrzymana w proxy. Firefox może wysłać dodatkowe żądania diagnostyczne do `detectportal.firefox.com:80` i to żądanie może również zostać zatrzymane w Burp Suite.

Jeżeli tak się stanie, należy po prostu pozwolić wysłać przeglądarce to żądanie naciskając na „Forward”. Może się zdarzyć, że będzie wysłane więcej niż jedno żądanie diagnostyczne, które trzeba będzie przekazać dalej.



Ryc. 16. Zatrzymane żądanie diagnostyczne.



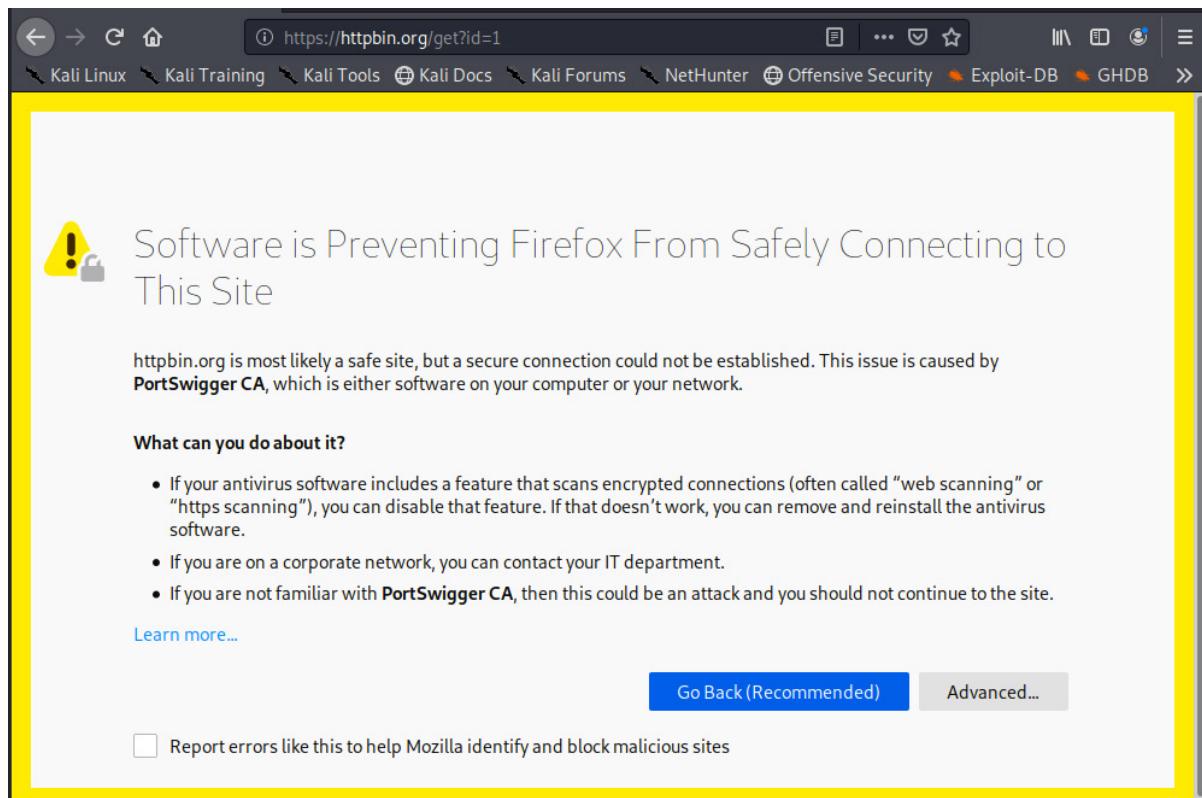
The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A request to `https://httpbin.org:443` is listed, showing the status as 'Request to https://httpbin.org:443 [35.170.216.115]'. The 'Intercept' button is highlighted in orange, indicating it is active. Below the request, the raw HTTP message is displayed:

```
GET /get?id=1 HTTP/1.1
Host: httpbin.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
Upgrade-Insecure-Requests: 1
```

Ryc. 17. Zatrzymane żądanie do `https://httpbin.org/get?id=1`.

Firefox powinien również wyświetlić błąd, który mówi o tym, że nie można połączyć się bezpiecznie ze stroną.

Spowodowane jest to tym, że Burp Suite próbuje podsłuchać ruch HTTPS i zamienia prawdziwy certyfikat na własny. Firefox zauważa niepasujący certyfikat i informuje, że podejrzewa atak Man-in-The-Middle.



The screenshot shows a Firefox browser window with a yellow border around the main content area. The address bar shows `https://httpbin.org/get?id=1`. The page content displays a warning message:

 Software is Preventing Firefox From Safely Connecting to This Site

httpbin.org is most likely a safe site, but a secure connection could not be established. This issue is caused by **PortSwigger CA**, which is either software on your computer or your network.

**What can you do about it?**

- If your antivirus software includes a feature that scans encrypted connections (often called "web scanning" or "https scanning"), you can disable that feature. If that doesn't work, you can remove and reinstall the antivirus software.
- If you are on a corporate network, you can contact your IT department.
- If you are not familiar with **PortSwigger CA**, then this could be an attack and you should not continue to the site.

[Learn more...](#)

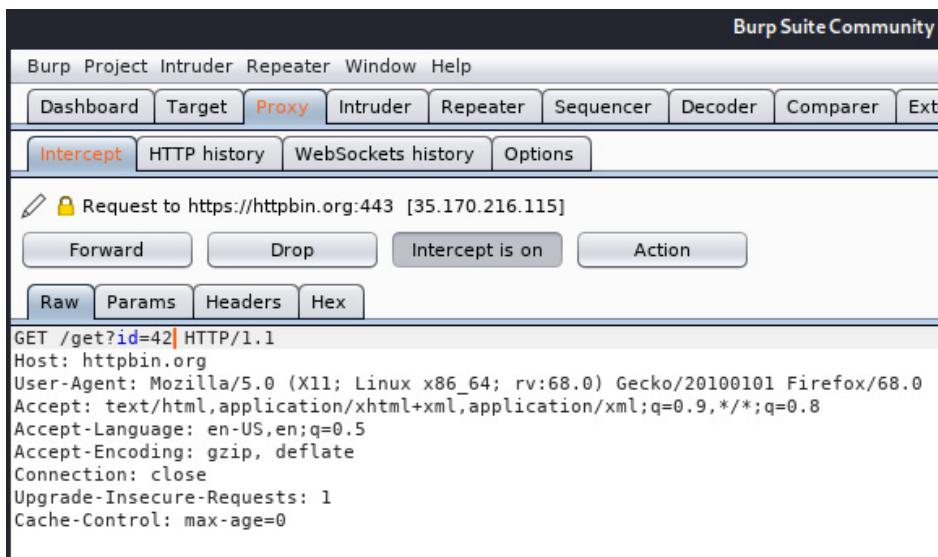
[Go Back \(Recommended\)](#) [Advanced...](#)

Report errors like this to help Mozilla identify and block malicious sites

Ryc. 18. Błąd wyświetlany podczas próby połączenia ze stroną HTTPS podczas używania proxy Burp Suite.

W tym momencie rozsądny użytkownik powinien zrezygnować z używania tego serwera proxy bądź sieci. Dla celów szkoleniowych, aby umożliwić wyświetlenie strony, należy kliknąć na „Zaawansowane...” („Advanced...”) i „Akceptuję ryzyko, kontynuuj” („Accept the Risk and Continue”).

W Burp Suite powinno się pojawić nowe żądanie.



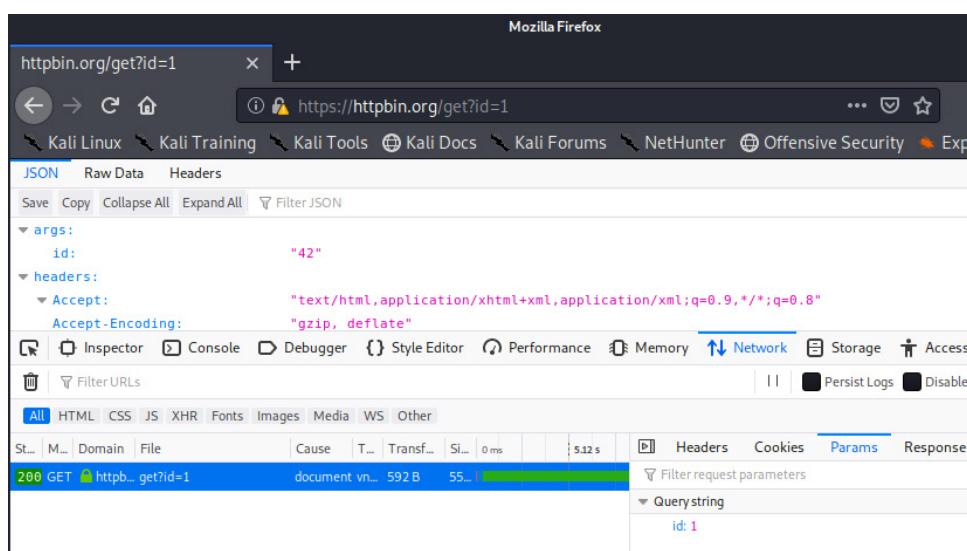
Ryc. 19. Modyfikacja żądania https://httpbin.org/get?id=1.

Po przesłaniu tego żądania dalej strona w końcu się załaduje.

Aby podjąć próbę modyfikacji żądania bez wiedzy przeglądarki, należy odświeżyć stronę. Nowe żądanie powinno zostać zatrzymane w proxy Burp Suite.

W przykładowym żądaniu zostanie zmieniony parametr **id** na 42.

Po powrocie do przeglądarki można zauważyć, że adres strony to `https://httpbin.org/get?id=1`, ale serwer odpowiedział tak, jakby zostało otwarte `https://httpbin.org/get?id=42`.



Ryc. 20. Odpowiedź na zmodyfikowane żądanie.

Przeglądarka nie wie, że żądanie było zmieniane w jakikolwiek sposób. W narzędziu „Sieć” będzie wpis, że został wysłany parametr „id” mający wartość „1”. Serwer, jednakże, odebrał id=42, ponieważ proxy zmieniło żądanie.

## Używanie proxy do modyfikowania żądań

Proxy może również modyfikować odpowiedzi.

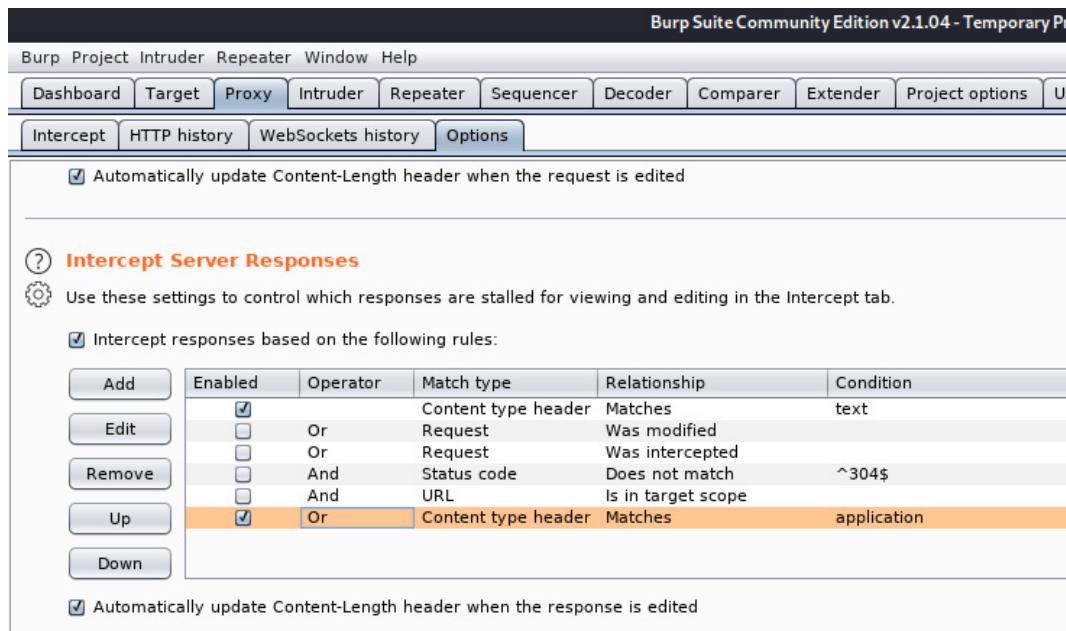
Dla atakującego zaletą modyfikowania odpowiedzi jest też to, że serwer nie zarejestruje zmienionego żądania. Czasami ze względów bezpieczeństwa wszystkie żądania są rejestrowane, a następnie analizowane w celu sprawdzenia, czy nie ma żadnej podejrzanej aktywności.

Oprócz tego atakujący może zmodyfikować odpowiedź, aby oszukać użytkownika - na przykład: dodając ostrzeżenie na stronie, że muszą zalogować się na fałszywej stronie banku (atak typu phishing). Możliwe jest też wstrzyknięcie złośliwego skryptu JavaScript, który spowoduje wykradnięcie danych przechowywanych w przeglądarce, bądź użyje komputera ofiary jako „koparki” kryptowaluty.

Aby włączyć przechwytywanie odpowiedzi w Burp Suite, należy przejść do „Options” i włączyć opcję „Intercept responses based on the following rules”.

Warto również dodać nową zasadę, która umożliwi przechwytywanie odpowiedzi, które nie są tylko w formacie HTML. Należy kliknąć na „Add” i ustawić następujące parametry:

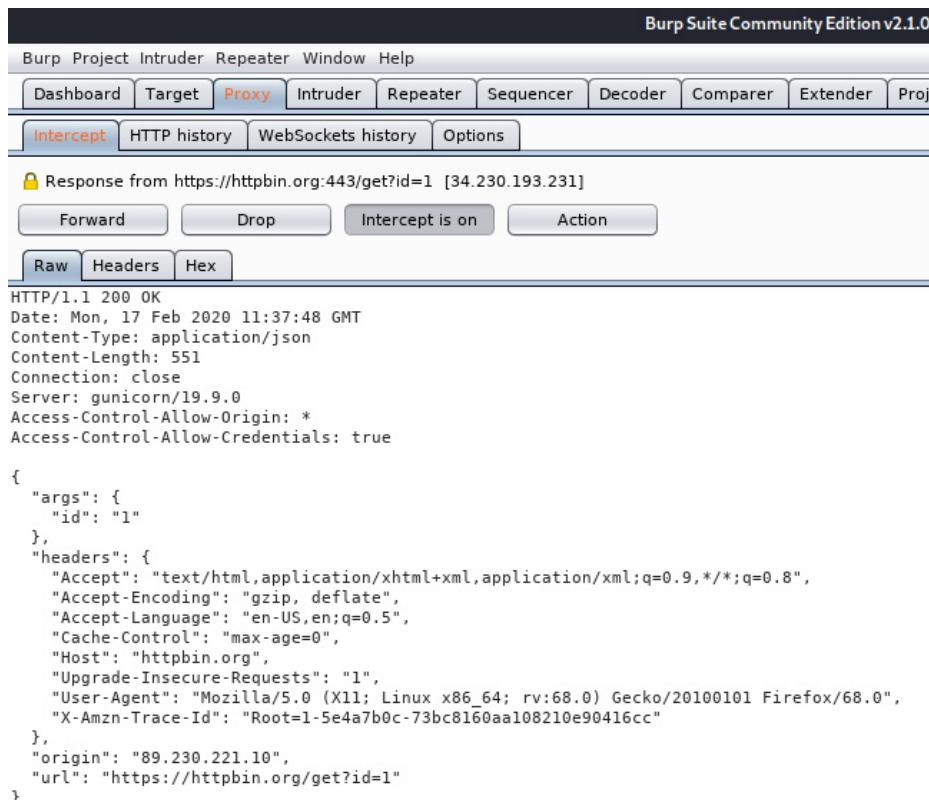
- operator „Or”,
- „match type” „Content type header”
- „match relationship” „Matches”
- „match condition” „application”.



Ryc. 21. Włączanie przechwytywania odpowiedzi.

Po odświeżeniu strony w przeglądarce można zauważyć, że w Burp Proxy zostanie zatrzymane zarówno żądanie, jak i odpowiedź.

Dla przykładu zostanie również zmieniona wartość w odpowiedzi "id": "1" na "id": "100".

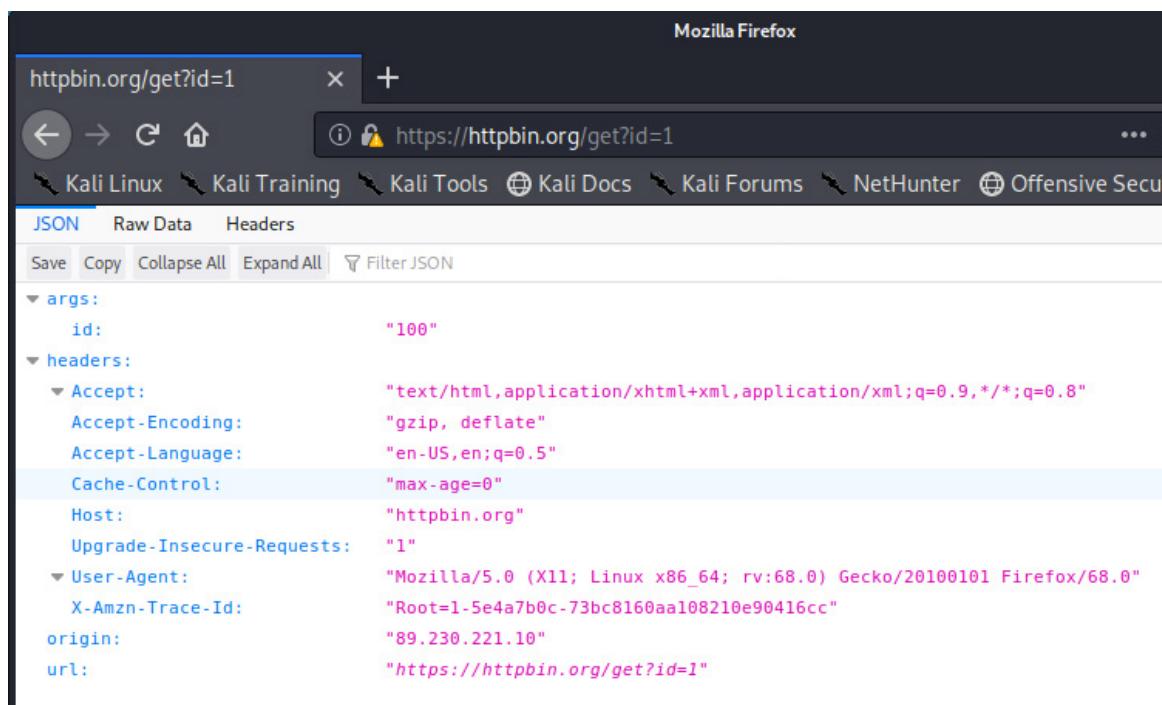


The screenshot shows the Burp Suite interface with the "Intercept" tab selected. A response from <https://httpbin.org/get?id=1> is displayed. The response body is a JSON object with the "args" field containing an "id" value of "1". The "headers" field includes standard HTTP headers like Accept, Accept-Encoding, Accept-Language, Cache-Control, Host, Upgrade-Insecure-Requests, User-Agent, X-Amzn-Trace-Id, Origin, and URL. The "Raw" tab is selected at the bottom.

```
HTTP/1.1 200 OK
Date: Mon, 17 Feb 2020 11:37:48 GMT
Content-Type: application/json
Content-Length: 551
Connection: close
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

{
  "args": {
    "id": "1"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "en-US,en;q=0.5",
    "Cache-Control": "max-age=0",
    "Host": "httpbin.org",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0",
    "X-Amzn-Trace-Id": "Root=1-5e4a7b0c-73bc8160aa108210e90416cc"
  },
  "origin": "89.230.221.10",
  "url": "https://httpbin.org/get?id=1"
}
```

Ryc. 22. Zatrzymana odpowiedź <https://httpbin.org/get?id=1>.



The screenshot shows the Mozilla Firefox developer tools network tab for the URL <https://httpbin.org/get?id=1>. The response is displayed in JSON format. The "args" field now contains an "id" value of "100". The "headers" field includes the same set of standard HTTP headers as in the Burp Suite screenshot. The "Raw Data" tab is selected at the top.

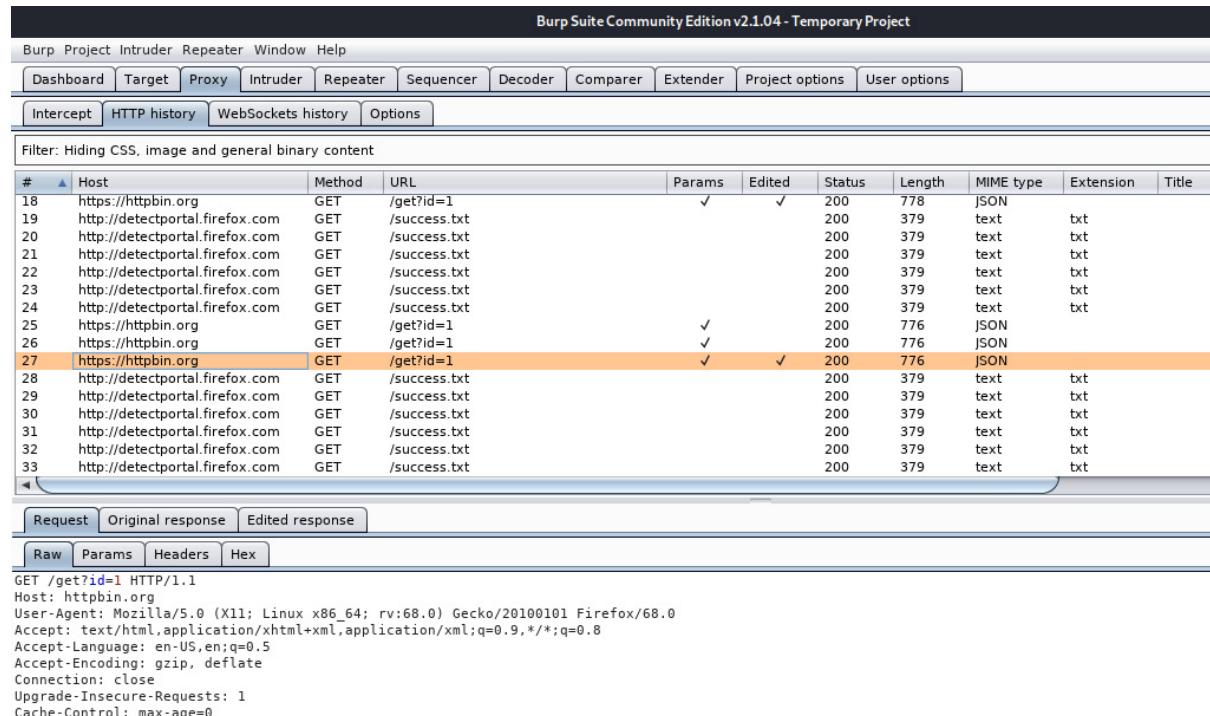
```
args:
  id: "100"
headers:
  Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
  Accept-Encoding: "gzip, deflate"
  Accept-Language: "en-US,en;q=0.5"
  Cache-Control: "max-age=0"
  Host: "httpbin.org"
  Upgrade-Insecure-Requests: "1"
  User-Agent: "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
  X-Amzn-Trace-Id: "Root=1-5e4a7b0c-73bc8160aa108210e90416cc"
  origin: "89.230.221.10"
  url: "https://httpbin.org/get?id=1"
```

Ryc. 23. Zmodyfikowana odpowiedź <https://httpbin.org/get?id=1>.

Po załadowaniu, będzie można zauważyc, że odpowiedź odebrana przez przeglądarkę została zmodyfikowana - dokładnie tak jak opisano wyżej. Parametr `id` powinien mieć wartość „100”, a `url` - niezmienioną wartość „<https://httpbin.org/get?id=1>”.

## Log komunikacji proxy

Proxy Burp Suite pozwala na zobaczenie historii pośredniczonych żądań, a także oryginalne oraz edytowane wiadomości.



#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
18	https://httpbin.org	GET	/get?id=1			200	778	JSON		
19	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
20	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
21	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
22	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
23	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
24	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
25	https://httpbin.org	GET	/get?id=1			200	776	JSON		
26	https://httpbin.org	GET	/get?id=1			200	776	JSON		
27	https://httpbin.org	GET	/get?id=1			200	776	JSON		
28	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
29	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
30	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
31	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
32	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
33	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	

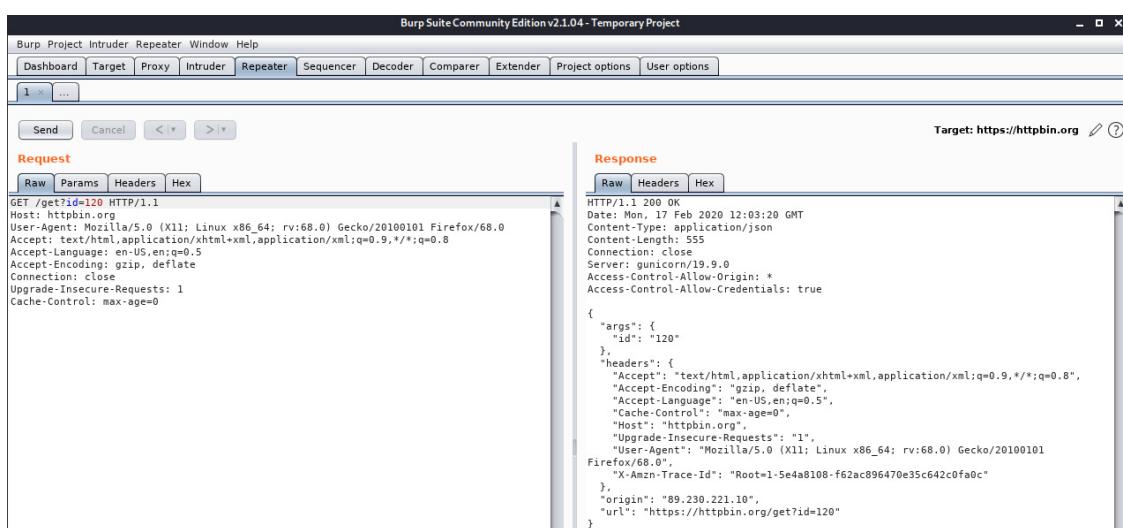
Request: <https://httpbin.org/get?id=1>

```
GET /get?id=1 HTTP/1.1
Host: httpbin.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Ryc. 24. Historia pośredniczonych żądań i odpowiedzi w Burp Suite.

Logi komunikacji są dostępne w karcie „HTTP history”.

Historia HTTP pozwala nie tylko na przejrzenie przeszłych żądań i odpowiedzi, a także na powtórzenie lub porównanie ich.



Request:

```
GET /get?id=120 HTTP/1.1
Host: httpbin.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
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
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 17 Feb 2020 12:03:20 GMT
Content-Type: application/json
Content-Length: 555
Connection: close
Server: Apache/2/Ubuntu/19.0.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
{
  "args": {
    "id": "120"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "en-US,en;q=0.5",
    "Cache-Control": "max-age=0",
    "Host": "httpbin.org",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0",
    "X-Amzn-Trace-Id": "Root=1-5e4a8108-f62ac896470e35c642c0fa0c"
  },
  "origin": "89.230.221.10",
  "url": "https://httpbin.org/get?id=120"
}
```

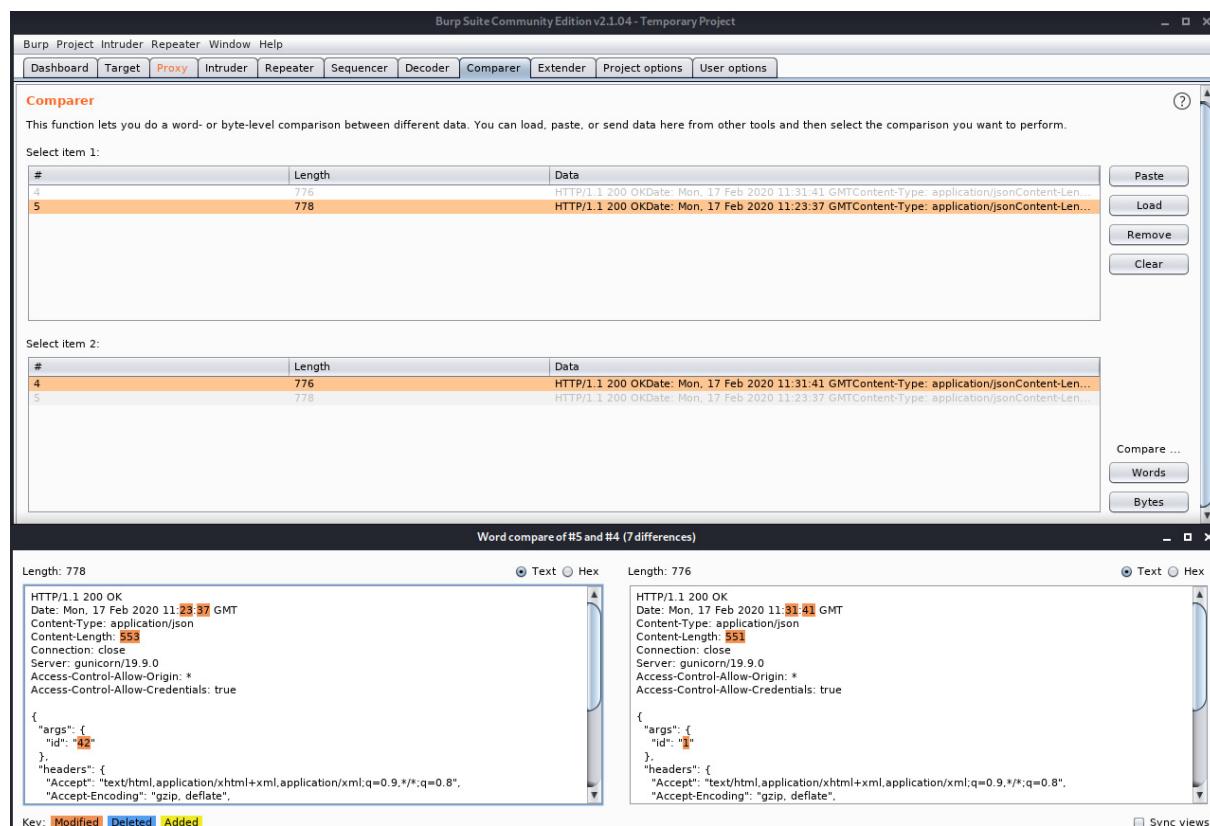
Ryc. 25. Ponowne wysyłanie żądania w Burp Suite.

Klikając na ostatnie żądanie do <https://httpbin.org> można wybrać „Send to Repeater”. Odpowiednie żądanie zostanie przekazane do karty „Repeater”.

Wybrane żądanie będzie tutaj dostępne do edycji, a także do ponownej wysyłki. Tym razem parametr id zostanie zmieniony na 120, a następnie żądanie zostanie wysłane po kliknięciu „Send”.

Odpowiedź otrzymana od serwera powinna być adekwatna do parametrów wysłanych w Repeaterze.

W historii HTTP można także wysłać dwa wybrane żądania lub odpowiedzi do narzędzia „Comparer”. Tam można je porównywać po słowach lub przeczytanych przez proxy bajtach.



Ryc. 26. Porównywanie odpowiedzi w Burp Suite.

## Sekrety stron www

Niektóre strony ukrywają „sekretnie” pliki albo zasoby poprzez używanie nazw lub ścieżek, które nie są nigdzie linkowane. Przeciętny użytkownik może nie wiedzieć, że taki plik istnieje, ponieważ nie ma do niego żadnego linku.

Załóżmy, że istnieje strona mająca adres „<http://sitewithsecrets.com>”. Administrator umieścił plik na serwerze i tylko on wie, że ten plik jest dostępny pod adresem „<http://sitewithsecrets.com/admins-private-files.zip>”. Jest mało prawdopodobne, że ktoś znajdzie ten plik - użytkownicy nie wiedzą, że istnieje, a wyszukiwarki również nie odnalały pliku. Można powiedzieć, że ten plik jest ukryty.

Ale sposób użyty przez administratora jest niepoprawnym sposobem ukrywania zasobów przed nieuprawnionym dostępem. Jeżeli ktoś odgadły adres, dostanie się do prywatnych plików administratora. Nie ma żadnej weryfikacji, że osoba, która próbuje pobrać plik, jest osobą, która powinna mieć do niego dostęp. Proces odnajdywania ukrytych zasobów może być nawet zautomatyzowany - jest wiele narzędzi, które automatycznie próbują „odgadnąć” każdą kombinację słów i znaków.

Parametry również mogą być podatne na wyżej przedstawiony atak. Założymy, że strona „<http://wordpresswithsecrets.com/>” działająca na WordPressie ma ukrytą stronę „<http://wordpresswithsecrets.com/?p=1029>”. Można bardzo łatwo ją odnaleźć, wiedząc, że strony na WordPressie mają linki w formacie <http://wordpresswithsecrets.com/?p=x>, gdzie x jest liczbowym identyfikatorem strony. Atakujący może więc po prostu sprawdzać po kolei ?p=1, ?p=2, ?p=3 itd. aż w końcu odnайдzie ukrytą stronę.

## Automatyczny skan w OWASP ZAP

Jednym z najbardziej popularnych narzędzi do skanowania stron jest OWASP ZAP. Pozwala na przeprowadzenie zautomatyzowanego skanu strony, albo ręcznie ustawić parametryzowane skanowanie. Posiada również dużo użytecznych narzędzi do badania stron (takich jak wbudowane proxy).

W Kali Linux OWASP ZAP może nie być domyślnie zainstalowany. Aby go zainstalować, należy uruchomić następujące komendy w terminalu:

```
sudo apt update  
sudo apt install zaproxy
```

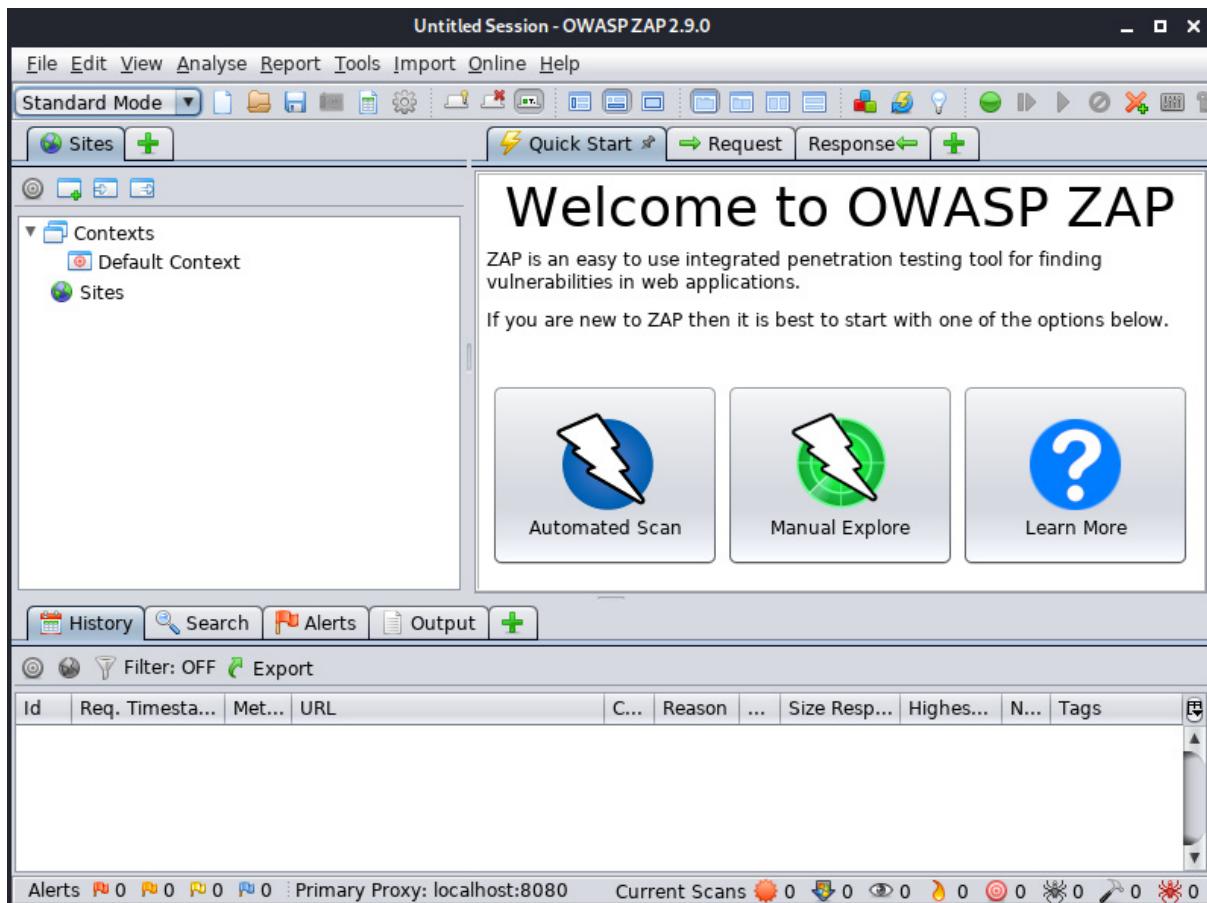
W OWASP ZAP można na początek wybrać automatyczny skan („Automated Scan”) i wpisać adres URL.

Przykładowo można wpisać <https://httpbin.org/get?id=1> i kliknąć na „Attack”.

Po chwili automatyczny skan powinien się zakończyć. W dolnej części okna powinny być dwie nowe karty - „Spider” i „Active Scan”. Powinna także pojawić się nowa strona w „Sites” w lewym panelu.

**Spider** to narzędzie, które próbuje odnaleźć pliki i foldery używając linków dostępnych na stronie i najczęściej używanych wzorów.

**Active Scan** to automatyczny skan, który szuka parametrów w adresach i próbuje znaleźć często występujące podatności.



Ryc. 27. Ekran główny OWASP ZAP.

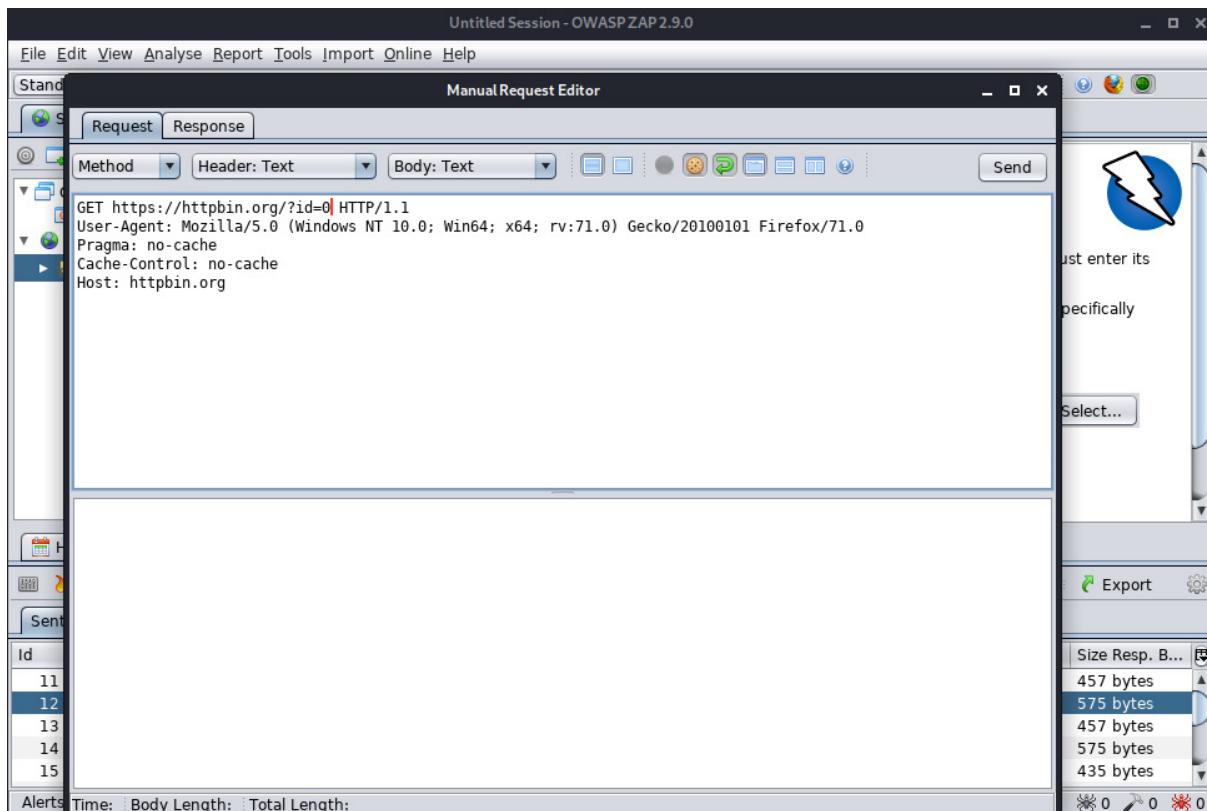
This screenshot shows the OWASP ZAP interface after performing an active scan on the site <https://httpbin.org>. The left sidebar shows the context 'Default Context' and the site 'https://httpbin.org' with several listed endpoints: GET:deny, GET:get(id), GET:robots.txt, and GET:sitemap.xml. The main pane displays the raw response for the 'GET:deny' endpoint, showing the HTTP headers and the JSON payload. The response body contains a JSON object with fields 'args', 'id', 'headers', 'Cache-Control', 'Content-Length', 'Host', and 'Pragma'. The status bar at the bottom indicates 'Alerts 0 1 4 1 Primary Proxy: localhost:8080 Current Scans:0 : Num requests: 125 : New Alerts:1 : Export'.

ID	Req. Timestamp	Resp. Timestamp	Met...	URL	Co...	Reason	R...	Size Resp...	He...	Size Resp. B...
11	2/17/20, 7:51:5...	2/17/20, 7:51:5...	GET	<a href="https://httpbin.org/get?id=c%3A%2FW...">https://httpbin.org/get?id=c%3A%2FW...</a>	200	OK	6...	230 bytes		457 bytes
12	2/17/20, 7:51:5...	2/17/20, 7:51:5...	GET	<a href="https://httpbin.org/get?id=..%2F..%2F...">https://httpbin.org/get?id=..%2F..%2F...</a>	200	OK	1...	230 bytes		575 bytes
13	2/17/20, 7:51:5...	2/17/20, 7:51:5...	GET	<a href="https://httpbin.org/get?id=c%3A%5CW...">https://httpbin.org/get?id=c%3A%5CW...</a>	200	OK	1...	230 bytes		457 bytes
14	2/17/20, 7:51:5...	2/17/20, 7:51:5...	GET	<a href="https://httpbin.org/get?id=..%5C..%5C...">https://httpbin.org/get?id=..%5C..%5C...</a>	200	OK	1...	230 bytes		575 bytes
15	2/17/20, 7:51:5...	2/17/20, 7:51:5...	GET	<a href="https://httpbin.org/get?id=%2Fetc%2F...">https://httpbin.org/get?id=%2Fetc%2F...</a>	200	OK	1...	230 bytes		435 bytes

Ryc. 28. Przeglądanie wyników Active Scan w OWASP ZAP.

## Atak siłowy na parametr

Atak na dany parametr żądania może być ustawiony ręcznie. Na początku należy przygotować żądanie do ataku. Przed rozpoczęciem zautomatyzowanego ataku OWASP ZAP powinien mieć wzór żądania, które będzie używane. Aby edytować żądanie należy kliknąć prawym przyciskiem myszy ma „<https://httpbin.org>” w sekcji „Sites” i wybrać „Open/Resend with Request Editor...”



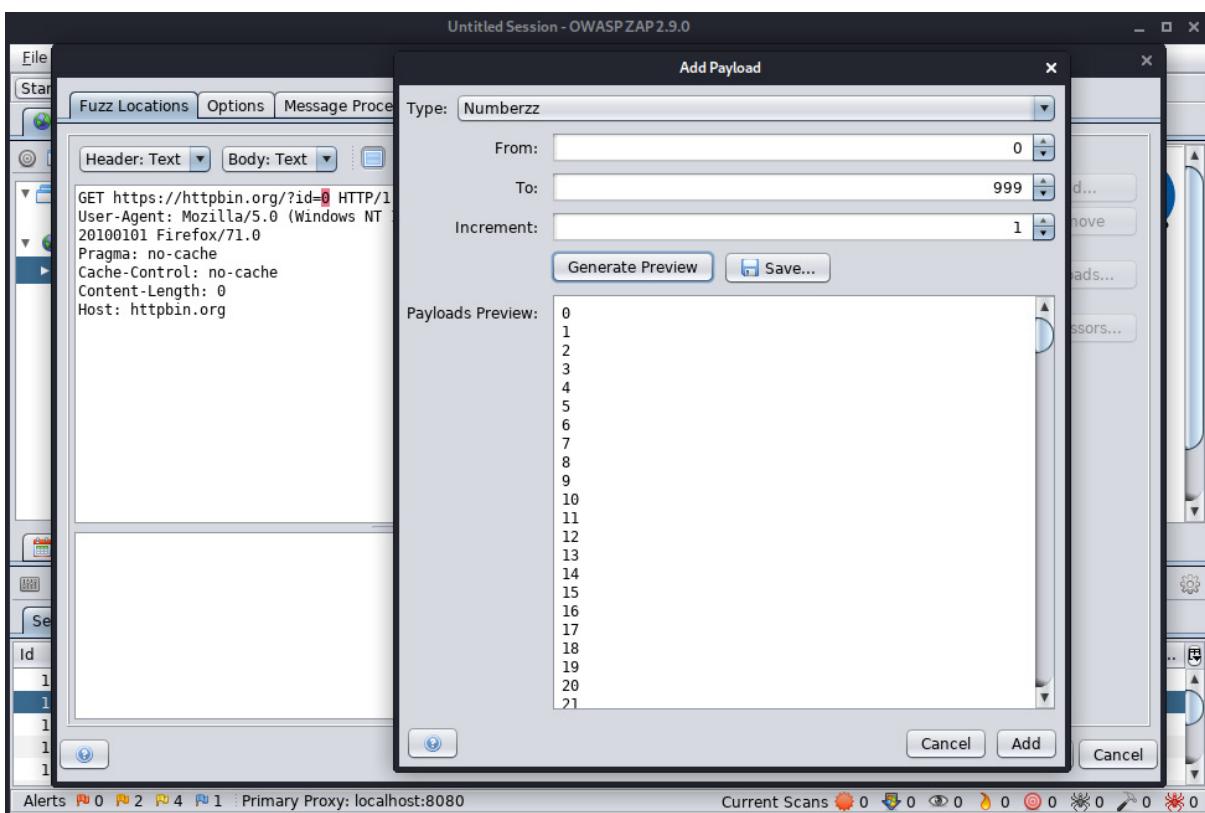
Ryc. 29. Edycja żądania w OWASP ZAP.

Należy przygotować żądanie tak, aby miało parametr **id** w adresie URL. Po skończeniu edycji należy kliknąć na „Send” i zamknąć okno.

Mając przygotowane żądanie, można rozpocząć przygotowywania ataku. Należy kliknąć prawym przyciskiem myszy na „<https://httpbin.org>” i wybrać „Attack” > „Fuzz....”

W oknie, które się pojawiło, należy części żądania, która będzie podlegała zmianom podczas kolejnych prób ataku. W przypadku przygotowanego żądania do httpbin.org będzie to wartość parametru „id”. Należy zaznaczyć tę wartość i kliknąć na „Add....”

Okno, które się pojawiło, pozwala na zdefiniowanie zestawów wartości, które będą użyte do ataku. Aby dodać zestaw wartości, należy kliknąć w tym oknie na „Add....”



Ryc. 30. Konfiguracja ataku Fuzz.

Wygenerowane wartości liczbowe można dodać wybierając typ „Numberzz” oraz docelowy zakres wartości. Przykładowo OWASP ZAP może wygenerować liczby od 0 do 999. Cały ten zakres będzie używany jako wartość parametru „id” w kolejnych żądaniach.

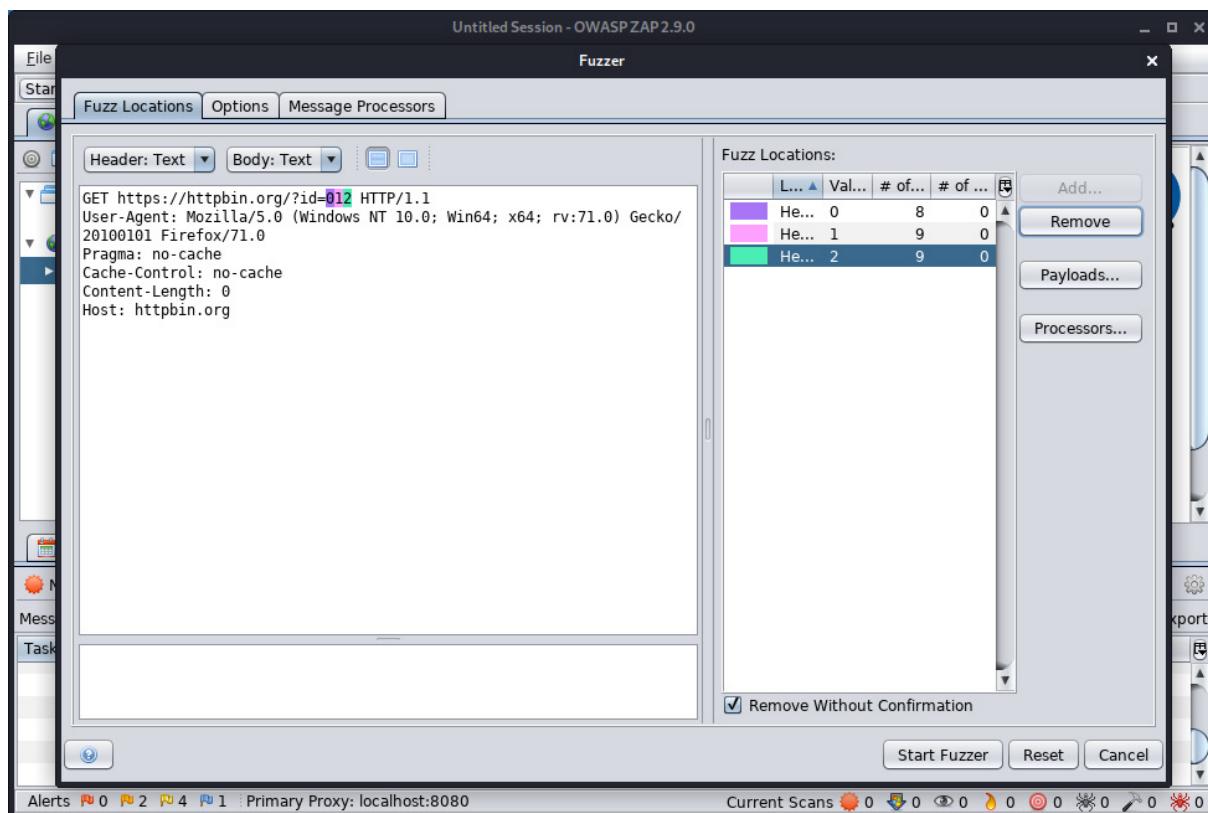
Po zapisaniu zestawów wartości, atak Fuzz jest gotowy.

Aby rozpocząć atak należy kliknąć na „Start Fuzzer”. Atak może chwilę potrwać.

Wyniki mogą być sortowane po kodzie odpowiedzi (Code field), czasie odpowiedzi (RTT) albo wielkości odpowiedzi (Size Resp. Body). Te dane mogą być przydatne podczas prawdziwego ataku, ponieważ:

- Serwer może odpowiedzieć błędem, jeśli parametr jest błędny, ale za to kodem 200 OK, kiedy odkryto ukryty zasób
- Odpowiedź może zająć serwerowi więcej czasu, gdy atak się powiedzie (z powodu dodatkowego zapytania do bazy danych, albo ładowania dodatkowych plików)
- Serwer może odpowiedzieć z ciałem o innej wielkości, kiedy atak się powiedzie (ponieważ serwer może zwrócić dodatkowe dane zamiast strony z błędem)

Są także inne rodzaje ataku Fuzz, które pozwalają na użycie tekstu albo parametrów z pliku. W żądaniu można również zdefiniować więcej niż jedno miejsce ataku. W tym przypadku zostaną przetestowane wszystkie możliwe kombinacje.



Ryc. 31. Użycie trzech Fuzzów do atakowania parametru.



Train for your career in computer science  
anytime, anywhere.



email: [contact@cyberskiller.com](mailto:contact@cyberskiller.com)  
[cyberskiller.com](http://cyberskiller.com)