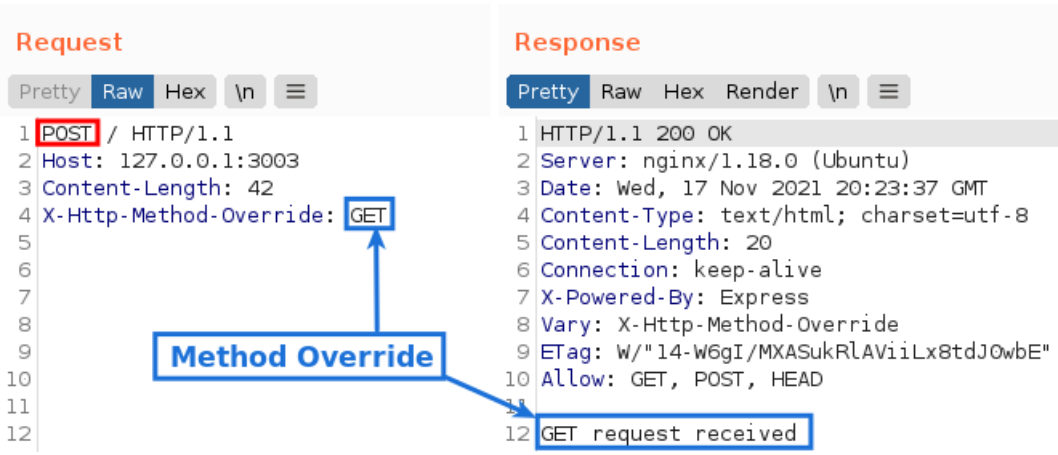
2 Nagłówki protokołu HTTP

Nagłówek X-HTTP-Method-Override

Nagłówek `X-HTTP-Method-Override` pozwala w żądaniach typu `GET` lub `POST` przemycić inną metodę `HTTP`. Gdy żądanie dotrze do serwera docelowego, oryginalna metoda zostanie nadpisana metodą pobraną z nagłówka `X-HTTP-Method-Override` i w tej formie trafi do obsłużenia w aplikacji.

Przykład:

Rysunek 10: Żądanie bez nagłówka `X-HTTP-Method-Override`

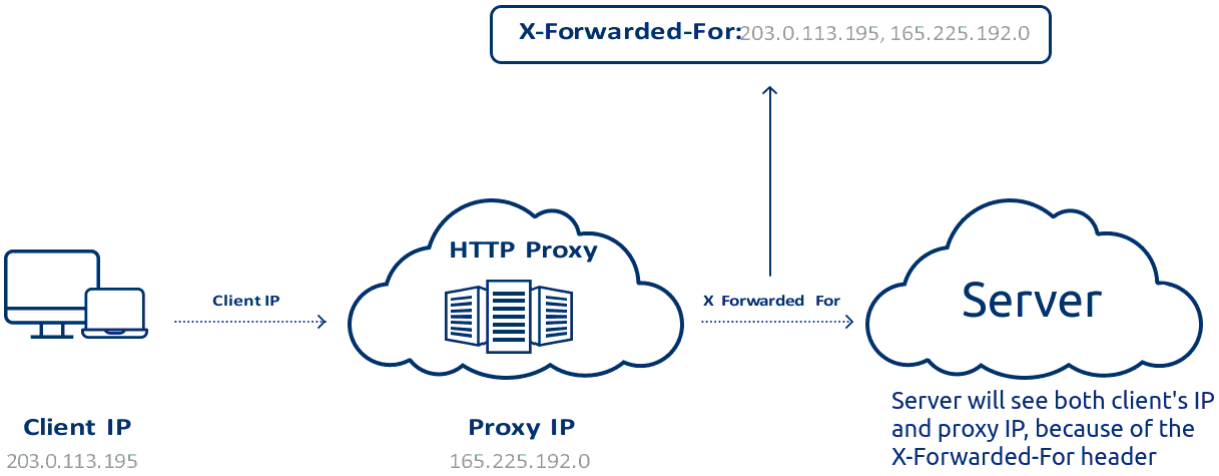


Rysunek 11: Żądanie z nagłówkiem `X-HTTP-Method-Override`

Nagłówek X-Forwarded-For

`X-Forwarded-For (XFF)` to niestandardowy nagłówek HTTP, który identyfikuje adres IP klienta dla oryginalnego żądania, które zostało dostarczone przez serwer proxy lub load balancer, dzięki czemu aplikacja na drugim końcu wie, z kim ma do czynienia.

Nagłówek ten jest jednym z najważniejszych nagłówków, które mogą mieć wpływ na bezpieczeństwo, ponieważ za jego pomocą (poprzez jego sfałszowanie) możliwe jest ominięcie zasad bezpieczeństwa aplikacji internetowej. Co więcej, brak lub niepoprawne ustawienie tego nagłówka powoduje, że aplikacja zobaczyłaby tylko adres IP serwera proxy, co jest sytuacją niepożądaną. Z tego powodu serwery stojące za serwerami proxy muszą wiedzieć, które z nich są godne zaufania właśnie dzięki poprawnej interpretacji wartości tego nagłówka.

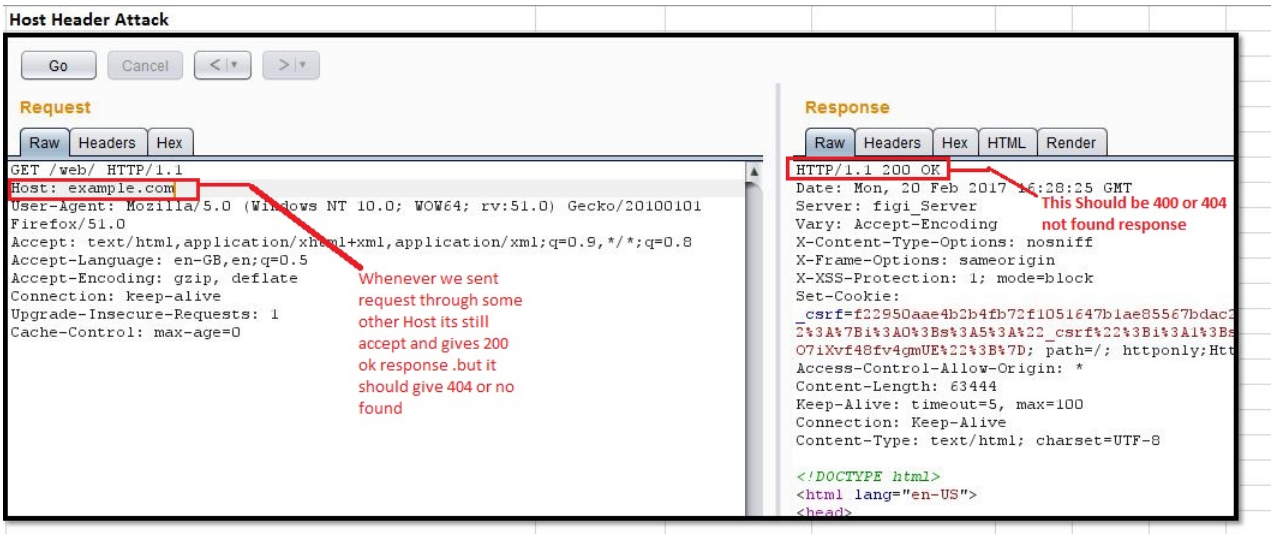


Rysunek 12: Działanie nagłówka `Nagłówek X-Forwarded-For`

Nagłówek Host

Nagłówek Host jest jednym z najważniejszych nagłówków w komunikacji HTTP. Informuje on serwer, którego wirtualnego hosta ma użyć, pod jaki adres chcemy wysłać zapytanie oraz określa, która aplikacja powinna przetwarzać przychodzące żądanie HTTP. Nagłówek ten, wprowadzony w HTTP/1.1, to trzecia z najważniejszych informacji, której można użyć, oprócz adresu IP i numeru portu, w celu jednoznacznego zidentyfikowania serwera aplikacji lub domeny internetowej.

Wartość nagłówka Host w protokole HTTP jest dowolnym tekstem kontrolowanym i przekazywanym przez klienta, ale czasami traktuje się go tak, jakby był bezpieczną zmienną - a nią nie jest. Jeśli konfiguracja aplikacji webowej (a także serwera WWW) jest nieprawidłowa, atakujący jest w stanie wstrzyknąć dowolny nagłówek HTTP typu Host i przepisać adresy na każdej stronie tak, aby kierowały do jego spreparowanych zasobów (np. szkodliwego oprogramowania).



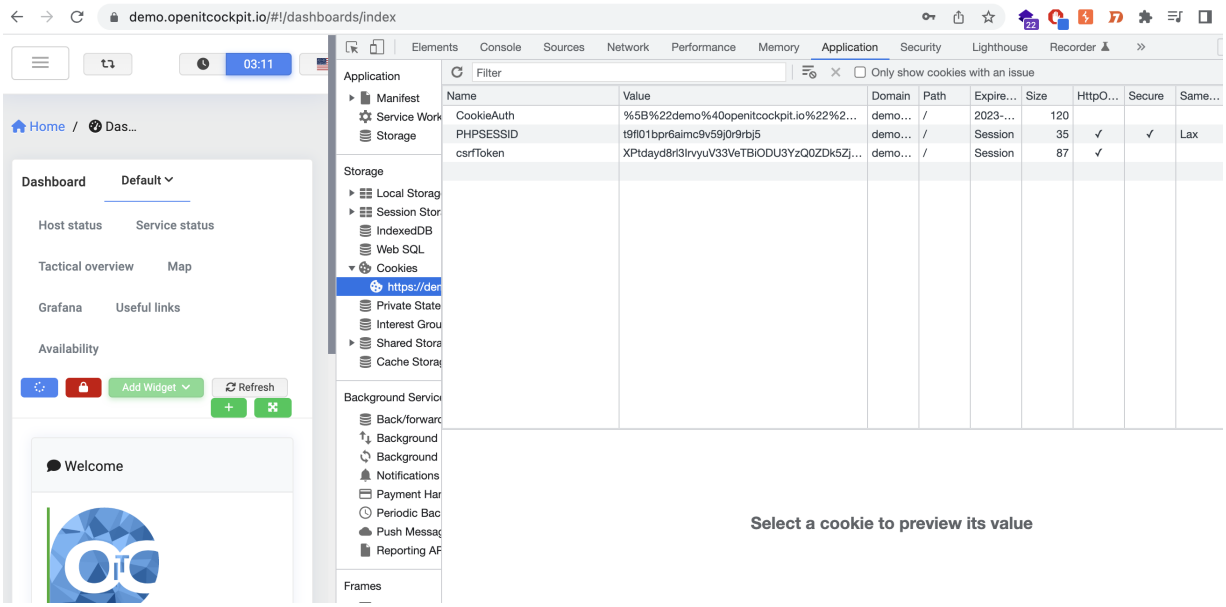
Rysunek 13: Atak przy użyciu nagłówka `Host`

Nagłówek Cookie

Jak wiemy, protokół HTTP jest bezstanowy, tzn nie 'przechowuje' informacji o tym, co działo się wcześniej. Jest to oczywisty problem w większości przypadków, kiedy korzystamy z narzędzi wymagających zalogowania się - informacja o tym, jaki użytkownik jest zalogowany musi być przechowywana w jakiś sposób. Rozwiązaniem stosowanym obecnie na szeroka skalę są tzw. ciasteczka - pierwotnie mające postać plików tekstowych w formacie klucz=wartość, obecnie przechowywane w wewnętrznej bazie danych przeglądarki. Formalnie ciasteczka (ang. cookies) to zbiór par klucz-wartość przypisanych do danej domeny które są wysyłane do serwera z każdym zapytaniem. Oczywiście ze względów bezpieczeństwa nie przechowuje się w nich danych użytkownika, do tego służą tzw. sesje, czyli kolekcje danych przechowywane po stronie serwera. W ciasteczkach najczęściej zapisuje się tzw. klucz sesji - unikalny identyfikator, na podstawie którego serwer ma możliwość powiązania jednego ze zbiorów danych które przechowuje z wysyłającym zapytanie klientem. Oczywiście większość współczesnych serwerów ma dodatkowe zabezpieczenia, sprawdza np. adres IP klienta powiązanego z sesją, żeby upewnić się że nie ma miejsca próba przechwycenia sesji (tzw. atak typu Session Hijacking). Ogólnie jednak ciasteczka i jedna zapisana w nich wartość jest podstawą wystarczającą do wprowadzenia stanowości w aplikacjach webowych.

Flagi nagłówka cookie HTTP (ang. HTTP Cookie Flags) to specjalne atrybuty dodawane do nagłowków cookie HTTP, które pozwalają na kontrolowanie zachowań i ograniczeń tych wartości nagłowków cookie w przeglądarkach internetowych. Flagi te pomagają zwiększyć prywatność, bezpieczeństwo i kontrolę nad danymi przesyłanymi za pomocą cookie. Oto kilka popularnych flag nagłówka cookie HTTP:

- `Secure` Flagę Secure ustawia się, gdy plik cookie jest używany tylko w połączeniach zabezpieczonych przez protokół HTTPS. Oznacza to, że plik cookie będzie przesyłany tylko wtedy, gdy strona jest ładowana przez bezpieczne połączenie, co pomaga w ochronie danych użytkowników przed atakami typu man-in-the-middle.
- `HttpOnly`: Ta flaga uniemożliwia dostęp do plików cookie za pomocą skryptów JavaScript. Pozwala to na zabezpieczenie niektórych informacji cookie przed atakami XSS (Cross-Site Scripting).
- `Max-Age` i `Expires`: Flagi te pozwalają określić, przez jak długi czas plik cookie powinien być przechowywany w przeglądarce użytkownika. `Max-Age` określa czas w sekundach, a `Expires` określa konkretną datę i godzinę wygaśnięcia cookie.
- ... `https://www.invicti.com/learn/cookie-security-flags/`



Rysunek 14: Sprawdzenie zawartości nagłówka Cookie

Linki

https://www.sidechannel.blog/en/http-method-override-what-it-is-and-how-a-pentester-can-use-it/

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For

https://dev.to/nathan20/all-you-should-know-about-http-host-header-injection-18il

https://www.sobyte.net/post/2022-03/http-host-header-attack/

https://sekurak.pl/jak-czasem-mozna-latwo-oszukac-mechanizm-przypominania-hasla-vulnz/

https://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html

https://portswigger.net/web-security/host-header

Zadania

2.1 Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/bsk-book-p1-ch2-ex21:latest`), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 `127.0.0.1` na porcie TCP o numerze `2001` działa serwer obsługujący protokół HTTP w wersji 1.1. Serwer w odpowiedzi na żądanie, zwraca nazwę metody HTTP, którą otrzymał w żądaniu. W przypadku odebrania żądania z metodą HTTP `OPTIONS`, serwer zwraca zawartość pliku `password21.txt`. Spróbuj wysłać do serwera żądanie, tak, aby otrzymać zawartość pliku `password21.txt`. Zadanie możesz rozwiązać za pomocą narzędzi: `telnet`, `cURL`, `BurpSuite`, ...

2.2 Rozwiąż zadanie **2.1** za pomocą skryptu w języku Python. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów. Odebrane hasło zapisz do pliku `password21.txt`.

2.3 Poniżej znajduje się request protokołu HTTP/1.1 w zapisie szesnastkowym. Wiedząc, że znak `\r` w zapisie szesnastkowym to `0x0d`, natomiast `\n` to `0x0a` oraz wiedząc, że w zapisie szesnastkowym jedna cyfra reprezentuje 4 bity, oraz, że kod ASCII jest kodem 8-bitowym, napisz program w języku Python, w którym sprawdzisz, jakie nagłówki protokołu HTTP zostały wysłane do serwera.

```
47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a
48 6f 73 74 3a 20 68 74  74 70 62 69 6e 2e 6f 72
67 0d 0a 58 2d 48 54 54  50 2d 4d 65 74 68 6f 64
2d 4f 76 65 72 72 69 64  65 3a 20 50 55 54 0d 0a
0d 0a
```

2.4 Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/bsk-book-p1-ch2-ex24:latest`), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 `127.0.0.1` na porcie TCP o numerze `2004` działa serwer obsługujący protokół HTTP w wersji 1.1. Serwer w odpowiedzi na żądanie, zwraca zawartość pliku `password24.txt`. Jednakże, do pobrania pliku uprawniony jest jedynie komputer z adresem IPv4 `192.168.200.200`. W przypadku odebrania żądania pochodzącego od innego komputera, serwer zwraca adres IP, z którego przyszło żądanie. Spróbuj wysłać do serwera odpowiednie żądanie, tak, aby otrzymać zawartość pliku `password24.txt` z komputera z dowolnym adresem IP. Zadanie możesz rozwiązać za pomocą narzędzi: `telnet`, `cURL`, `BurpSuite`, ...

2.5 Rozwiąż zadanie **2.4** za pomocą skryptu w języku Python. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów. Odebrane hasło zapisz do pliku `password24.txt`.

2.6 Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/bsk-book-p1-ch2-ex26:latest`), uruchom serwer HTTP. Po uruchomieniu obrazu, pod adresem IPv4 `127.0.0.1` na porcie TCP o numerze `2006` działa serwer obsługujący protokół HTTP w wersji 1.1. W odpowiedzi na żądanie `GET` klienta, serwer zwraca prostą stronę z adresem, do którego połączył się klient. Serwer posiada również endpoint `http://127.0.0.1/whoami`. Wyślij do serwera żądanie w taki sposób, aby zamiast odsyłać stronę ze swoim adresem, serwer wysłał do klienta dowolną stronę (działającą w Internecie, wskazaną przez klienta). Zadanie możesz rozwiązać za pomocą narzędzi: `telnet`, `cURL`, `BurpSuite`, ...

2.7 Rozwiąż zadanie **2.6** za pomocą skryptu w języku Python. Podczas pisania kodu związanego z operacjami sieciowymi, nie korzystaj z dodatkowych bibliotek, wykorzystaj jedynie gniazda. Zadbaj o prawidłową obsługę błędów.

2.8 Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/bsk-book-p1-ch2-ex28:latest`), uruchom aplikację `rdiffweb`. Po uruchomieniu obrazu, pod adresem IPv4 `127.0.0.1` na porcie TCP o numerze `2008` działa serwer obsługujący aplikację `rdiffweb` służącą do zarządzania backupami Linuksa przez interfejs webowy. Wiedząc, że aplikacja jest podatna na atak Open redirect z wykorzystaniem `Host header injection`, wykonaj atak, który przekieruje użytkownika na dowolną stronę. Zadanie możesz rozwiązać za pomocą `BurpSuite`.

2.9 Rozwiąż zadanie **2.8** za pomocą skryptu w języku Python. Podczas pisania kodu związanego z operacjami sieciowymi, możesz skorzystać z gniazd, lub modułu `requests`. Zadbaj o prawidłową obsługę błędów.

2.10 Korzystając z przygotowanego obrazu Dockerowego (`mazurkatarzyna/bsk-book-p1-ch2-ex210:latest`), uruchom aplikację `memos`. Po uruchomieniu obrazu, pod adresem IPv4 `127.0.0.1` na porcie TCP o numerze `2010` działa serwer obsługujący aplikację `memos`. Sprawdź jakie flagi ma ustawiony nagłówek `Cookie` przesłany przez serwer, a następnie spróbuj wykorzystać tę informację do dodania nowego użytkownika do aplikacji za pomocą narzędzia `cURL`. **Uwaga:** aby rozwiązać zadanie możesz wysłać tylko jeden request za pomocą narzędzia `cURL`.

Nagłówki protokołu HTTP

2.1 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/bsk-book-p1-ch2-ex21:latest
docker run -dp 2001:2001 mazurkatarzyna/bsk-book-p1-ch2-ex21
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: `http://127.0.0.1:2001` i podejrzyj, jak wygląda żądanie i odpowiedź HTTP.

2.2 Wykorzystaj gniazda TCP:

```python
#!/usr/bin/env python3
import socket

if __name__ == '__main__':

        sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sockIPv4.close()
```

2.3 Możesz skorzystać z funkcji: ascii.

2.4 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/bsk-book-p1-ch2-ex24:latest
docker run -dp 2004:2004 mazurkatarzyna/bsk-book-p1-ch2-ex24
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: `http://127.0.0.1:2004` i podejrzyj, jak wygląda żądanie i odpowiedź HTTP.

2.5 Wykorzystaj gniazda TCP:

```python
#!/usr/bin/env python3
import socket

if __name__ == '__main__':

        sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sockIPv4.close()
```

2.6 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/bsk-book-p1-ch2-ex26:latest
docker run -dp 2006:2006 mazurkatarzyna/bsk-book-p1-ch2-ex26
```

Działanie serwera możesz przetestować za pomocą przeglądarki. W swojej przeglądarce otwórz stronę: `http://127.0.0.1:2006` i podejrzyj, jak wygląda żądanie i odpowiedź HTTP.

2.7 Wykorzystaj gniazda TCP:

```python
#!/usr/bin/env python3
import socket

if __name__ == '__main__':

        sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sockIPv4.close()
```

2.8 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/bsk-book-p1-ch2-ex28:latest
docker run -dp 2008:8080 mazurkatarzyna/bsk-book-p1-ch2-ex28
```

Do połączenia za pomocą narzędzia `telnet` użyj polecenia: `telnet 127.0.0.1 2008`. W swojej przeglądarce otwórz stronę: `http://127.0.0.1:2008` i podejrzyj, jak wygląda żądanie i odpowiedź HTTP.

2.9 Wykorzystaj gniazda TCP:

```python
#!/usr/bin/env python3
import socket

if __name__ == '__main__':

        sockIPv4 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sockIPv4.close()
```

2.10 Serwer możesz uruchomić za pomocą poniższych poleceń:

```
docker pull mazurkatarzyna/bsk-book-p1-ch2-ex210:latest
docker run -dp 2010:5230 mazurkatarzyna/bsk-book-p1-ch2-ex210
```