Gilberto Najera-Gutierrez
Juned Ahmed Ansari

Kali Linux

Testy penetracyjne

Wydanie 3

Helion

Packt

Tytuł oryginału: Web Penetration Testing with Kali Linux - Third Edition

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-5123-3

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/kalit3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

4

ROZDZIAŁ

Podatności uwierzytelniania i zarządzania sesjami

Głównym zadaniem aplikacji internetowych jest zapewnienie użytkownikom dostępu do informacji przechowywanych w odległym miejscu i możliwości ich przetwarzania. Czasami takie informacje są publicznie dostępne, ale często zdarza się również, że przetwarzane dane są specyficzne dla jakiegoś użytkownika lub nawet poufne. W takiej sytuacji aplikacje sieciowe zazwyczaj wymagają od użytkowników udowodnienia swojej tożsamości przed uzyskaniem dostępu do danych. Proces weryfikacji tożsamości, który nazywamy **uwierzytelnianiem** (ang. *authentication*), wymaga od użytkownika przedstawienia dowodu tożsamości, którym może być jeden lub więcej poniższych elementów:

- Coś, co użytkownik *zna* — np. nazwa konta użytkownika i tajne hasło.
- Coś, co użytkownik *posiada* — np. karta inteligentna lub specjalny kod wysyłany na telefon użytkownika.
- Coś, co użytkownika *identyfikuje* i co jest jego niepowtarzalną cechą — np.: głos, twarz, odcisk palca lub inny element biometryczny.

W aplikacjach internetowych najczęściej używana jest pierwsza metoda, aczkolwiek istnieją pewne specyficzne zastosowania, takie jak bankowość lub wewnętrzne aplikacje korporacyjne, które mogą wykorzystywać jedną lub więcej pozostałych metod.

HTTP jest protokołem bezstanowym i bezpołączeniowym. Oznacza to, że każde żądanie wysyłane przez klienta do serwera jest traktowane przez serwer jako niezwiązane z żadnymi wcześniejszymi lub przyszłymi żądaniami wysłanymi przez tego lub dowolnego innego klienta. Inaczej mówiąc, gdy użytkownik zaloguje się do aplikacji internetowej, każde następne wysyłane przez niego żądanie jest traktowane przez serwer tak, jakby było pierwsze.

Z tego względu klient z każdym kolejnym żądaniem musiałby wysyłać swoje poświadczenia logowania. Takie rozwiązanie powodowałoby niepotrzebne narażanie poufnych informacji i zbędny narzut na komunikację.

Aby temu zapobiec, opracowano szereg technik umożliwiających aplikacjom internetowym śledzenie działań użytkowników, utrzymywanie stanu aplikacji zgodnie ze zmianami wprowadzanymi w ich własnym środowisku oraz odseparowanie aplikacji od działań innych użytkowników bez proszenia ich o poświadczanie tożsamości przy każdej podejmowanej akcji. Takie rozwiązanie nazywamy **zarządzaniem sesją** (ang. *session management*).

W tym rozdziale zobaczysz, jak uwierzytelnianie i zarządzanie sesją są najczęściej realizowane w nowoczesnych aplikacjach internetowych, oraz dowiesz się, jak zidentyfikować i wykorzystać niektóre z najczęstszych luk bezpieczeństwa w takich mechanizmach.

Schematy uwierzytelniania w aplikacjach internetowych

Zanim zapoznasz się z wybranymi zagadnieniami dotyczącymi przeprowadzania testów penetracyjnych, sprawdzimy, w jaki sposób proces uwierzytelniania odbywa się w nowoczesnych aplikacjach internetowych.

Uwierzytelnianie na poziomie platformy

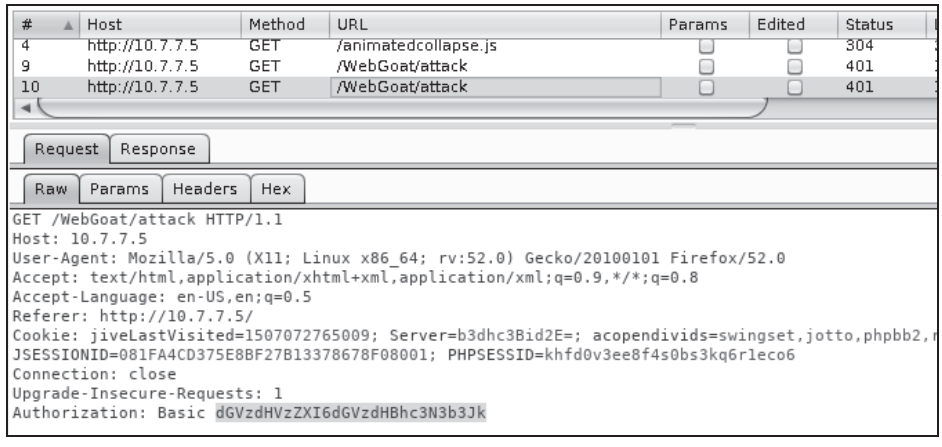
Podczas korzystania z **uwierzytelniania na poziomie platformy** (ang. *platform authentication*) użytkownicy wysyłają swoje poświadczenia w nagłówku każdego żądania, wykorzystując do tego celu pole `Authorization`. Nawet jeżeli użytkownik musi przesłać swoje dane uwierzytelniające tylko raz, przeglądarka lub system przechowuje je i używa ich w razie potrzeby.

Istnieje kilka różnych typów uwierzytelniania na poziomie platformy. Najczęstsze z nich omówiono w kolejnych podrozdziałach.

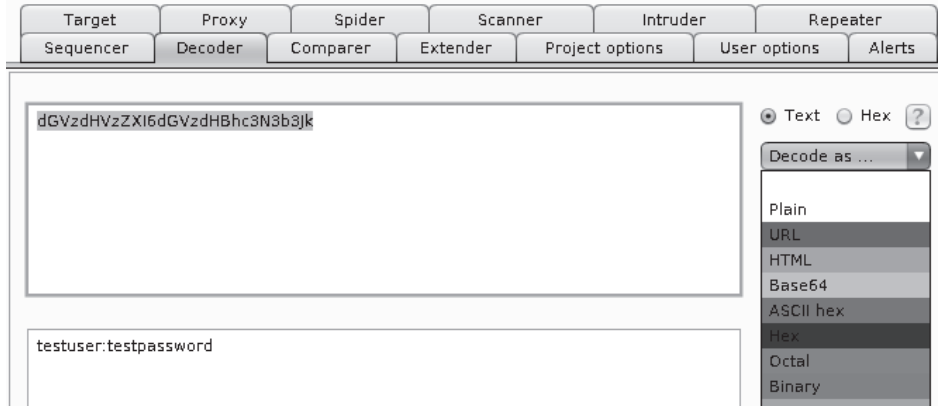
Uwierzytelnianie Basic

W przypadku tego typu uwierzytelniania platformy nazwa użytkownika i hasło są dołączane do nagłówka w polu `Authorization` i kodowane przy użyciu algorytmu base64. Oznacza to, że każdy, kto może przechwycić i zobaczyć nagłówek żądania, jest w stanie rozszyfrować dane uwierzytelniające do postaci jawnego tekstu, ponieważ kodowanie base64 nie jest formatem kryptograficznym.

Na rysunkach poniżej pokazujemy, w jaki sposób poświadczenia logowania są wysyłane w formacie base64 i jak można je dekodować:



Do konwersji tekstu z formatu base64 na ASCII możesz użyć modułu *Decoder* z pakietu Burp Suite:



Uwierzytelnianie Digest

Uwierzytelnianie Digest jest znacznie bezpieczniejsze niż podstawowe uwierzytelnianie typu Basic. Kiedy klient chce uzyskać dostęp do chronionego zasobu, serwer wysyła losowy ciąg znaków, nazywany **nonce**, jako wyzwanie. Następnie klient używa ciągu nonce wraz z nazwą użytkownika i hasłem do obliczenia wartości skrótu MD5 i odsyła tę wartość do serwera w celu weryfikacji.

Uwierzytelnianie NTLM

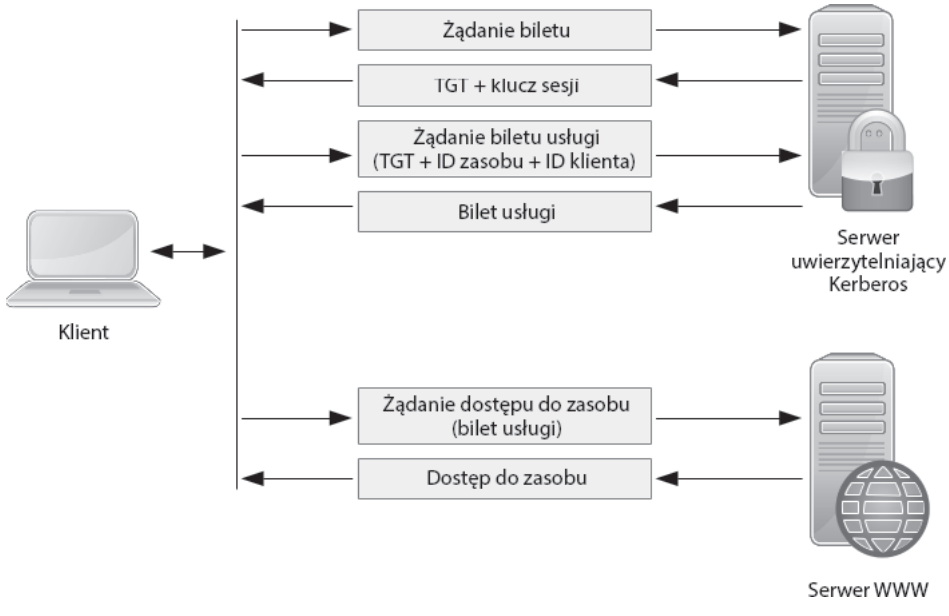
NTLM jest odmianą uwierzytelniania typu Digest, w którym poświadczenia logowania systemu Windows i algorytm haszowania NTLM są używane do wygenerowania komunikatów wyzwania i odpowiedzi. Ten schemat uwierzytelniania wymaga przesyłania wielu sekwencji typu żądanie – odpowiedź, a serwer WWW i wszelkie pośredniczące serwery proxy muszą obsługiwać połączenia trwałe (ang. *persistent connections*).

Uwierzytelnianie Kerberos

Ten schemat wykorzystuje protokół Kerberos do uwierzytelniania użytkownika na serwerze. Podobnie jak w przypadku uwierzytelniania NTLM, nie wymaga podawania nazwy użytkownika i hasła, ale do logowania używa poświadczeń z systemu Windows. Uwierzytelnianie Kerberos wymaga zastosowania dodatkowego serwera, zwanego **serwerem uwierzytelniania** (ang. **AS** — *Authentication Server*), i przebiega w kilku etapach:

1. Klient wysyła nazwę konta użytkownika (ID) do serwera uwierzytelniania (AS).
2. Serwer AS wyszukuje nazwę konta (ID) w swojej bazie danych i używa zahaszowanego hasła do szyfrowania klucza sesji.
3. Serwer AS wysyła do klienta zaszyfrowany klucz sesji oraz bilet **TGT** (ang. *Ticket Granting Ticket*) zawierający identyfikator użytkownika, klucz sesji, datę wygaśnięcia sesji i inne dane zaszyfrowane za pomocą tajnego klucza serwera. Jeżeli podane hasło jest nieprawidłowe, klient nie będzie w stanie odszyfrować klucza sesji.
4. Klient odszyfrowuje klucz sesji.
5. Gdy klient chce uzyskać dostęp do chronionego zasobu na serwerze sieciowym, będzie musiał wysłać bilet TGT oraz identyfikator zasobu w jednej wiadomości, a identyfikator klienta i znacznik czasu zaszyfrować za pomocą klucza sesji w innej wiadomości.
6. Jeżeli serwer jest w stanie odszyfrować odebrane informacje, odpowiada za pomocą biletu usługi zaszyfrowanego przy użyciu tajnego klucza serwera AS i klucza sesji klienta/serwera, dodatkowo zaszyfrowanego przy użyciu klucza sesji klienta.
7. Po otrzymaniu biletu usługi z serwera AS klient może zażądać dostępu do zasobów serwera WWW.

Opisany proces został przedstawiony na poniższym diagramie:



Uwierzytelnianie HTTP Negotiate

Zwane także uwierzytelnianiem systemu Windows (ang. *Windows Authentication*) uwierzytelnianie HTTP Negotiate używa poświadczeń systemu Windows i w zależności od tego, czy protokół Kerberos jest dostępny, wybiera między uwierzytelnianiem Kerberos i NTLM.

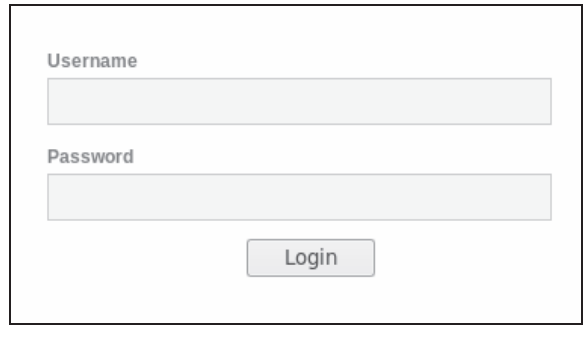
Wady uwierzytelniania na poziomie platformy

Choć schematy Kerberos i NTLM są uważane za bezpieczne, a uwierzytelnianie typu Basic lub Digest może być używane w połączeniach TLS z niskim ryzykiem przechwycenia komunikacji i pozyskania poświadczeń logowania przez złośliwego napastnika, uwierzytelnianie na poziomie platformy pod względem bezpieczeństwa wciąż wykazuje pewne „wrodzone" wady:

- Poświadczenia są wysyłane częściej, w związku z tym ich ekspozycja i ryzyko przechwycenia w ataku typu man-in-the-middle (MITM) są podwyższone, szczególnie w przypadku schematów Basic, Digest i NTLM.
- Uwierzytelnianie na poziomie platformy nie ma opcji wylogowania ani wygaśnięcia sesji. Ponieważ funkcja pojedynczego logowania (ang. **SSO** — *Single Sign On*) podczas korzystania z uwierzytelniania Windows Authentication jest włączona, to gdy użytkownik otworzy stronę główną aplikacji, sesja rozpoczyna się automatycznie, bez pytania o nazwę użytkownika i hasło, a następnie automatycznie odnawia się po wygaśnięciu. Napastnik, który uzyska dostęp do komputera użytkownika lub konta systemu Windows, uzyska więc w takiej sesji natychmiastowy dostęp do aplikacji.
- Uwierzytelnianie na poziomie platformy nie jest odpowiednie dla aplikacji publicznych, ponieważ wymaga znacznie większych nakładów technicznych i administracyjnych do odpowiedniego skonfigurowania i zarządzania niż najbardziej popularne uwierzytelnianie oparte na formularzach.

Uwierzytelnianie oparte na formularzach

Jest to rodzaj uwierzytelniania, z którym jesteśmy bardziej zaznajomieni: formularz HTML zawierający pola nazwy użytkownika i hasła oraz przycisk przesyłania:



Taki sposób uwierzytelniania może działać różnie w różnych przypadkach, ponieważ jego implementacja jest całkowicie zależna od aplikacji. Najczęściej stosowane podejście jest następujące:

1. Użytkownik wypełnia formularz uwierzytelnienia i klika przycisk *Zaloguj*. Klient (przeglądarka internetowa) wysyła następnie żądanie zawierające nazwę użytkownika i hasło do serwera w postaci zwykłego tekstu, chyba że aplikacja wykonuje szyfrowanie po stronie klienta.
2. Serwer otrzymuje przesłane informacje, sprawdza dane użytkownika w swojej bazie danych oraz porównuje zapisane i przesłane hasła (lub ich skróty).
3. Jeżeli użytkownik istnieje i hasło jest poprawne, serwer odpowiada komunikatem potwierdzającym zalogowanie, który może zawierać przekierowanie do strony głównej oraz identyfikator sesji (zwykle jako plik cookie), aby użytkownik nie musiał ponownie wysyłać swoich poświadczeń logowania.
4. Klient otrzymuje odpowiedź, zapisuje identyfikator sesji i przechodzi na stronę główną.

Z perspektywy pentestera zdecydowanie najciekawsza jest metoda uwierzytelniania, ponieważ nie jest w żaden sposób ustandaryzowana (nawet jeżeli istnieją jakieś wytyczne opisujące najlepsze praktyki jej implementacji) i zazwyczaj jest źródłem dużej liczby podatności i luk w zabezpieczeniach spowodowanych niewłaściwym sposobem wdrożenia.

Uwierzytelnianie dwuskładnikowe

Jak wspominaliśmy już wcześniej, aby udowodnić swoją tożsamość aplikacji sieciowej, musisz podać coś, co znasz, coś, co masz, lub coś, czym jesteś. Każdy z tych identyfikatorów nazywany jest **składnikiem** (ang. *factor*). **Uwierzytelnianie wieloskładnikowe** (ang. **MFA** — *Multi-factor Authentication*) polega na zapewnieniu wybranym aplikacjom dodatkowej warstwy zabezpieczeń i zapobieganiu nieautoryzowanemu dostępowi w przypadku, gdy np. hasło dostępu zostało złamane lub wykradzione przez napastnika.

Uwierzytelnianie dwuskładnikowe (ang. **2FA** — *Two-factor Authentication*) w większości aplikacji internetowych oznacza, że użytkownik musi podać nazwę użytkownika i hasło (pierwszy składnik) oraz specjalny kod lub hasło jednorazowe (ang. **OTP** — *One-Time Password*), które jest tymczasowe i losowo generowane przez urządzenie posiadane przez użytkownika (token sprzętowy) lub wysyłane do użytkownika za pośrednictwem wiadomości SMS lub e-mail. Po otrzymaniu hasła jednorazowego użytkownik przesyła je z powrotem do aplikacji. Bardziej zaawansowane aplikacje mogą posiadać zaimplementowane rozwiązania wymagające oprócz podania hasła np. użycia karty inteligentnej (ang. *Smart Card*) lub danych biometrycznych, takich jak odcisk palca czy skan siatkówki oka. Jednak ze względu na fakt, że takie rozwiązania wymagają od użytkownika posiadania dodatkowych, specjalistycznych urządzeń, są znacznie rzadsze.

Warto jednak zauważyć, że takie czy inne rozwiązania MFA zostały już wdrożone w zdecydowanej większości aplikacji bankowych, a w ostatnich latach uwierzytelnianie dwuskładnikowe staje się coraz bardziej popularne nawet w publicznych usługach poczty elektronicznej i w mediach społecznościowych, które zaczęły konsekwentnie promować wśród swoich użytkowników, a nawet wymuszać korzystanie z 2FA.

OAuth

OAuth to otwarty standard uwierzytelniania. Gdy użytkownicy Facebooka lub Google'a zezwalają aplikacjom stron trzecich na dostęp do swoich kont, nie przekazują tym aplikacjom swoich poświadczeń logowania. Zamiast tego usługodawcy (Google, Twitter lub Facebook) udostępniają specjalny token dostępu, który pozwala takim aplikacjom pobierać określone informacje o koncie użytkownika lub uzyskiwać dostęp do określonych funkcji zgodnie z pozwoleniami udzielonymi przez użytkownika.

Mechanizmy zarządzania sesjami

Zarządzanie sesją obejmuje tworzenie lub definiowanie identyfikatorów sesji podczas logowania, ustawianie limitu czasu nieaktywności, wygaśnięcia sesji i unieważniania sesji podczas wylogowywania; może również obejmować sprawdzanie autoryzacji w zależności od uprawnień użytkownika, ponieważ identyfikator sesji musi być powiązany z użytkownikiem.

Sesje oparte na uwierzytelnianiu platformy

Gdy używane jest uwierzytelnianie platformy, najczęściej stosowaną metodą jest wykorzystanie nagłówka zawierającego poświadczenie logowania lub identyfikowanie sesji użytkownika za pomocą mechanizmu wyzwania i odpowiedzi oraz zarządzanie wygaśnięciem sesji i wylogowaniem za pośrednictwem logiki aplikacji. Warto jednak przypomnieć, że bardzo często w aplikacjach korzystających z uwierzytelniania na poziomie platformy nie są stosowane żadne limity czasu sesji, jej wygaśnięcia lub wylogowania.

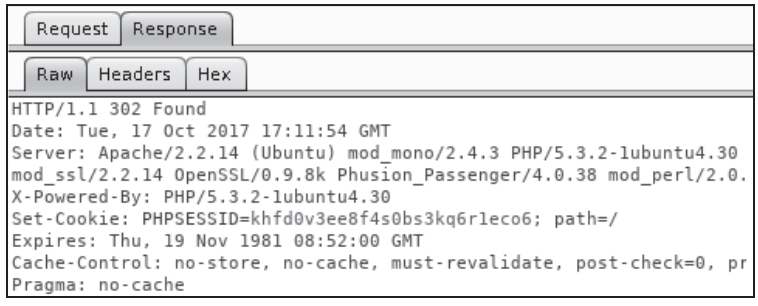
Jeżeli do uwierzytelniania używany jest protokół Kerberos, bilety generowane przez serwer AS zawierają już informacje o sesji i służą do zarządzania taką sesją.

Identyfikatory sesji

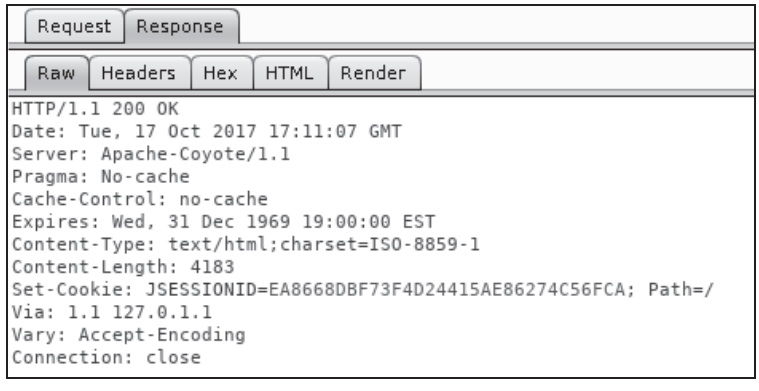
Identyfikatory sesji są bardziej powszechne w rozwiązaniach wykorzystujących mechanizm uwierzytelniania za pomocą formularza, ale mogą występować również, gdy używamy uwierzytelniania platformy. **Identyfikator sesji** (ang. *session ID*) jest unikalnym numerem lub wartością przypisywaną danemu użytkownikowi za każdym razem, gdy inicjuje sesję z aplikacją. Identyfikator sesji musi być inny niż identyfikator użytkownika i hasło, musi być inny za każdym razem, gdy użytkownik loguje się do aplikacji, i musi być dołączany do każdego wysyłanego żądania, tak aby serwer mógł rozróżniać żądania napływające z różnych sesji od różnych użytkowników.

Najczęstszym sposobem przekazywania identyfikatorów sesji między klientem a serwerem są pliki cookie. Gdy serwer otrzymuje poprawne poświadczenie logowania, składające się z nazwy użytkownika i hasła, łączy te informacje logowania z identyfikatorem sesji i przekazuje odpowiedź do klienta, wysyłając takie informacje np. w postaci odpowiednio przygotowanego pliku cookie.

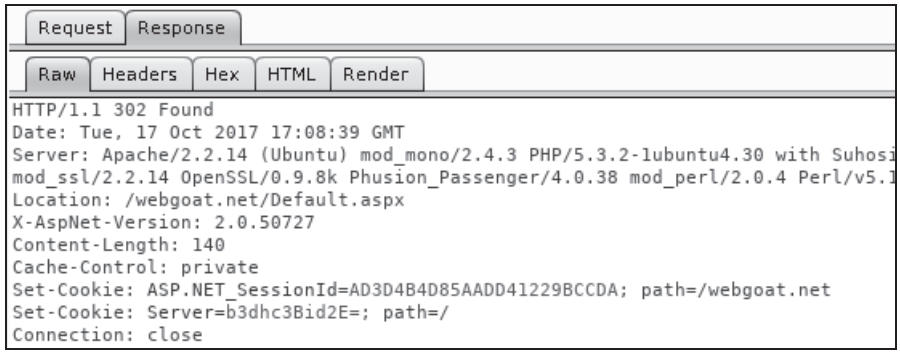
Na poniższych zrzutach ekranu zobaczysz kilka przykładów odpowiedzi serwera, które zawierają pliki cookie sesji:



W powyższym przykładzie aplikacja PHP ustawia plik cookie sesji o nazwie `PHPSESSID`.



W powyższym przykładzie aplikacja Javy ustawia plik cookie sesji o nazwie `JSESSIONID`.



W powyższym przykładzie aplikacja ASP.NET ustawia plik cookie sesji o nazwie `ASP.NET_SessionId`.

Typowe błędy uwierzytelniania w aplikacjach internetowych

Spędziliśmy już trochę czasu na omawianiu działania różnych mechanizmów uwierzytelniania w aplikacjach internetowych. W tej sekcji dowiesz się, jak identyfikować i wykorzystywać niektóre z podatności i luk w zabezpieczeniach spowodowane błędami w implementacji.

Brak uwierzytelnienia lub nieprawidłowa weryfikacja uwierzytelniania

W poprzednim rozdziale pokazywaliśmy, jak korzystać z polecenia DIRB i innych narzędzi pozwalających na wyszukiwanie plików i katalogów, które nie mogą być przywoływane przez żadną stronę na serwerze WWW lub które mogą posiadać uprzywilejowane funkcje, takie jak */admin* czy */user/profile*. Jeżeli możesz bezpośrednio zaglądać do tych katalogów i korzystać z zawartych w nich funkcjonalności bez potrzeby uwierzytelniania lub jeżeli po uwierzytelnieniu jako standardowy użytkownik możesz przeglądać obszary administracyjne aplikacji albo modyfikować profile innych użytkowników, to taka aplikacja z pewnością ma poważne problemy z bezpieczeństwem i funkcjonowaniem swoich mechanizmów uwierzytelniania.

Wyszukiwanie nazw kont użytkowników

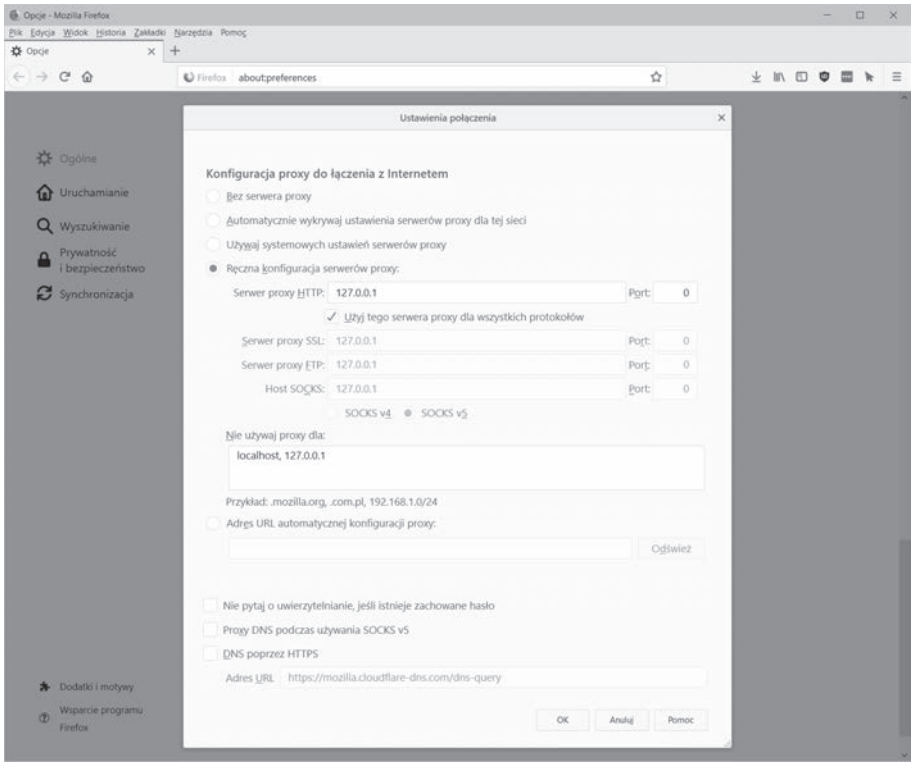
W scenariuszach testów penetracyjnych typu *grey box* lub *black box* próba pozyskania listy nazw kont użytkowników aplikacji może być jednym z pierwszych kroków wykonywanych przez pentestera, szczególnie jeżeli taka aplikacja nie jest komercyjna i nie można liczyć na to, że w sieci internet znajdziemy informacje o domyślnych kontach użytkowników.

Pozyskiwanie nazw kont użytkowników aplikacji internetowych odbywa się poprzez analizę odpowiedzi, gdy nazwy użytkowników są wpisywane w takich miejscach, jak strony logowania, rejestracji czy odzyskiwania hasła. Poniżej przedstawiono typowe komunikaty o błędach, które można zobaczyć podczas przesyłania danych za pośrednictwem formularzy logowania, a które wskazują, że można dokonać próby pozyskiwania nazw kont użytkowników:

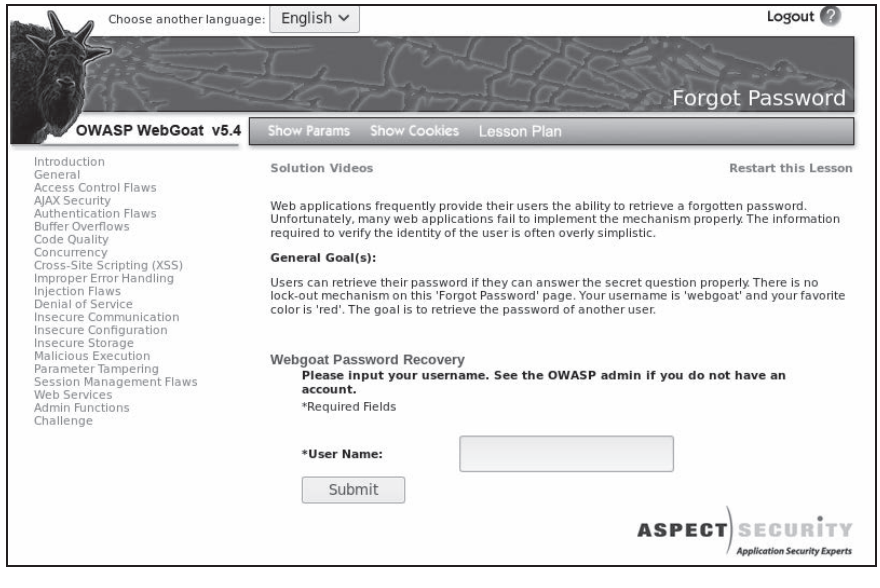
```
User foo: invalid password      (użytkownik foo: nieprawidłowe hasło)
invalid user ID                 (nieprawidłowy identyfikator użytkownika)
account disabled                (konto wyłączone)
this user is not active         (ten użytkownik nie jest aktywny)
invalid user                    (nieprawidłowy użytkownik)
```

Przyjrzyjmy się bardzo prostemu przykładowi pozyskiwania prawidłowych nazw kont użytkowników z aplikacji internetowej, która w przypadku podania nieprawidłowej nazwy użytkownika wyświetla zbyt dużo informacji. Użyjemy do tego celu aplikacji OWASP WebGoat z maszyny wirtualnej Broken Web Applications (BWA), która posiadała adres IP 10.7.7.5.

Najpierw uruchomimy pakiet Burp Suite i skonfigurujemy przeglądarkę tak, aby używała go jako proxy (jeżeli używasz Firefoksa, wybierz *Opcje/Ogólne/Sieć/Ustawienia połączenia*):



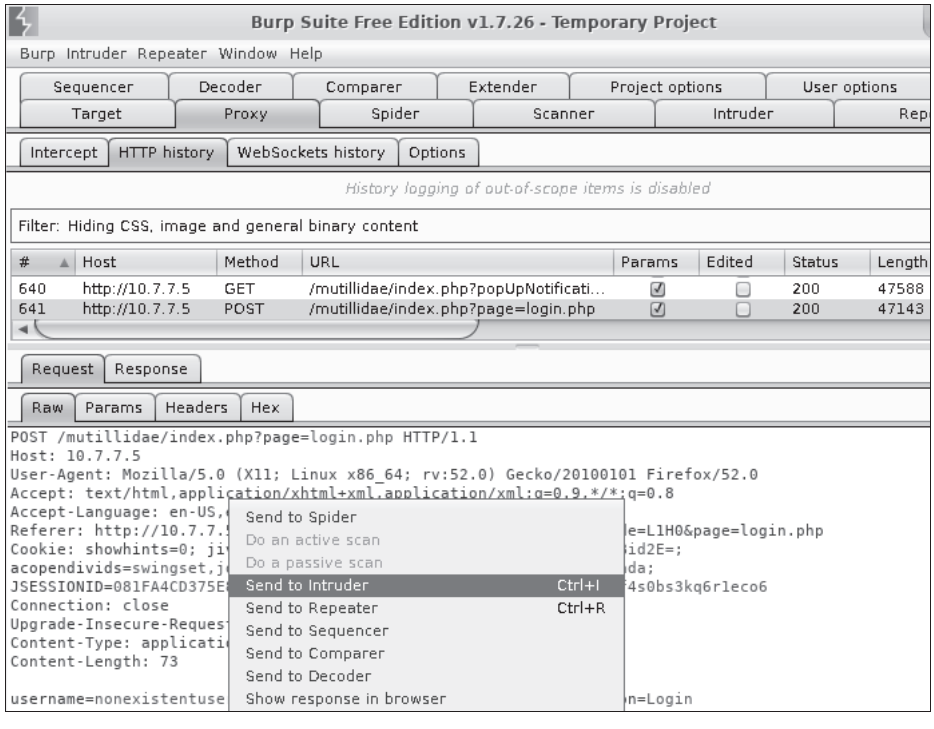
Następnie zaloguj się do aplikacji WebGoat, używając domyślnego konta użytkownika `webgoat` z hasłem `webgoat`, a potem kliknij łącze *Authentication Flaws/Forgot Password* (błędy uwierzytelniania/zapomniałem hasła):



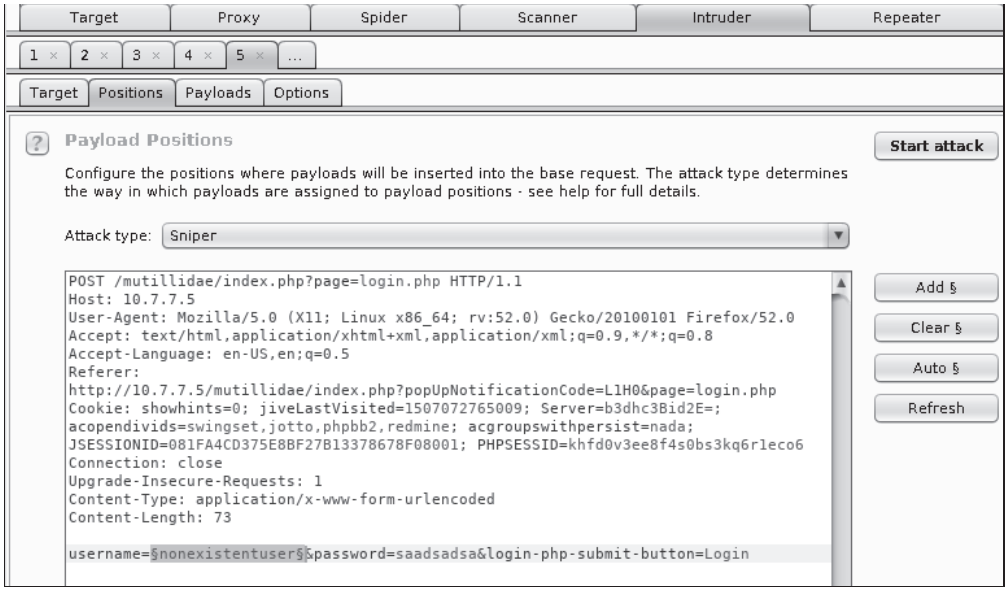
Jest to formularz odzyskiwania hasła, który wymaga podania nazwy użytkownika, aby kontynuować proces odzyskiwania. Wprowadź dowolną nazwę nieistniejącego konta użytkownika, np. `nonexistentuser`, a następnie naciśnij przycisk *Submit* (prześlij), aby zobaczyć rezultat:

Jak widać, nazwa użytkownika jest nieprawidłowa i nie można kontynuować odzyskiwania hasła. Możesz śmiało założyć, że gdy podasz poprawną nazwę konta użytkownika, otrzymasz inną odpowiedź.

Teraz użyjmy modułu Intruder z pakietu Burp Suite, aby dokonać próby znalezienia poprawnej nazwy konta użytkownika. Najpierw musimy znaleźć odpowiednie żądanie w historii Burp Proxy i przesłać je do modułu Intruder. Aby to zrobić, naciśnij kombinację klawiszy *Ctrl+I* lub kliknij prawym przyciskiem myszy i z menu podręcznego wybierz polecenie *Send to Intruder* (wyślij do modułu Intruder):



Teraz przejdź na kartę *Intruder*, potem kliknij kartę z numerem żądania i wreszcie przejdź na kartę *Positions* (pozycje). Z pewnością zauważysz, że wszystkie modyfikowalne parametry klienta są domyślnie zaznaczone. Naciśnij przycisk *Clear* (wyczyść), aby usunąć ich zaznaczenie, a następnie wybierz tylko pole `username` i kliknij przycisk *Add* (dodaj):



Moduł Intruder automatyzuje wysyłanie wielu żądań do serwera, zastępując wybrane wartości danymi podanymi przez użytkownika, i rejestruje wszystkie odpowiedzi, dzięki czemu możemy je analizować. Teraz musimy utworzyć listę nazw kont użytkownika, które chcemy wypróbować zamiast tego, które już zostało przesłane.

Burp Intruder pozwala na przeprowadzanie czterech typów ataków, różniących się liczbą i sposobami umieszczania ładunków:

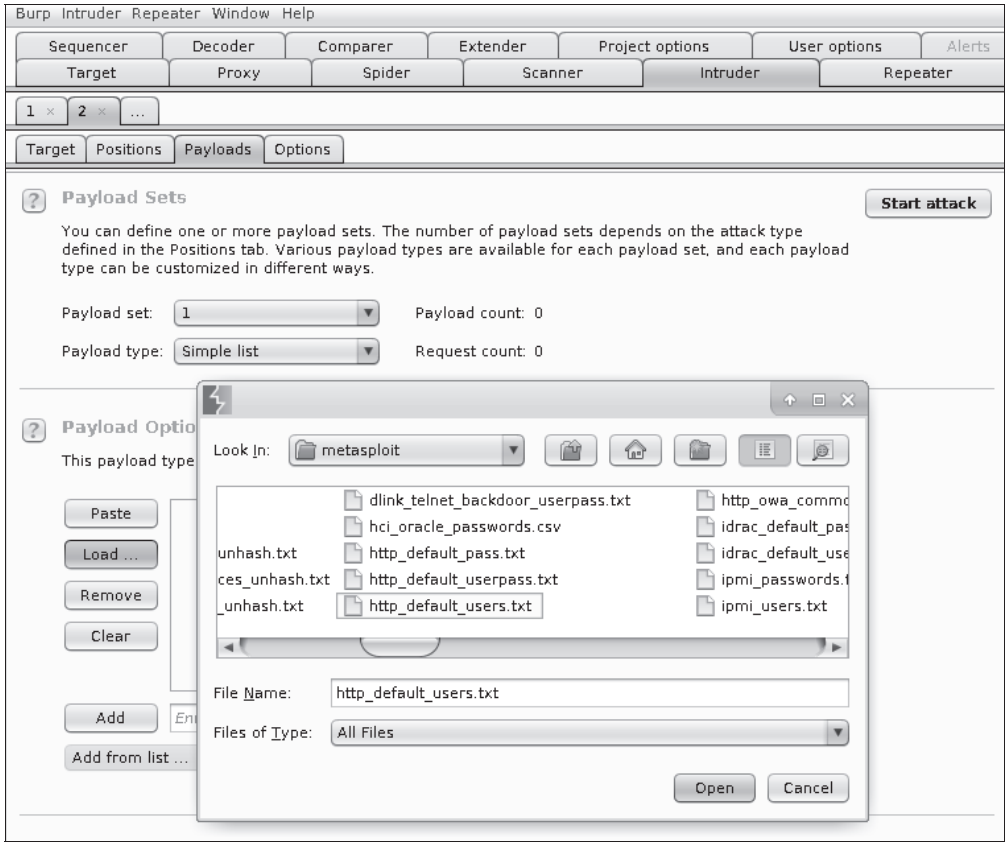
- **Sniper** (snajper) — wykorzystuje ładunek składający się z szeregu wartości; każda z wartości z tego zestawu jest umieszczana po kolei na poszczególnych pozycjach wejściowych, po jednej na raz. Liczba generowanych żądań będzie zatem równa liczbie wartości w zestawie ładunków pomnożonej przez liczbę pozycji wejściowych.
- **Battering ram** (taran) — wykorzystuje ładunek składający się z szeregu wartości; każda z wartości z tego zestawu jest umieszczana jednocześnie na wszystkich pozycjach wejściowych. Liczba żądań będzie równa liczbie wartości w zestawie ładunków.
- **Pitchfork** (widły) — używa wielu pozycji wyjściowych i wymaga utworzenia osobnego zestawu wartości dla każdej pozycji. Po uruchomieniu ataku pobiera po jednej wartości z każdego zestawu i umieszcza na odpowiadających im pozycjach wejściowych w tym samym czasie. Liczba wykonanych żądań będzie równa długości najkrótszego zestawu danych.

- **Cluster bomb** (bomba kasetowa) — jeżeli korzystamy z *n* pozycji wejściowych, poszczególne elementy z pierwszego zestawu danych są kolejno parowane ze wszystkimi elementami zestawu drugiego i pozostałych, tak aby utworzyć wszystkie możliwe kombinacje. Liczbę żądań w takim ataku można określić przez pomnożenie rozmiarów wszystkich zestawów ładunków.

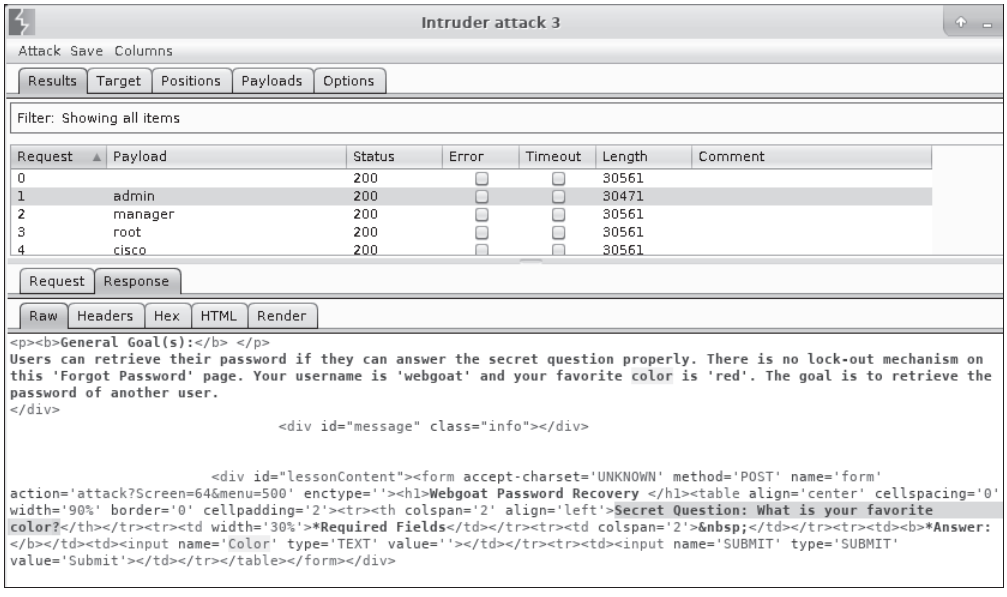
Następnie przejdź na kartę *Payloads* (ładunki) w module Intruder. Pozostaw opcję *Payload set* (zestaw ładunków) niezmienioną i naciśnij przycisk *Load…* (załaduj), znajdujący się w sekcji *Payload Options [Simple List]* (opcje ładunku [prosta lista]), który pozwala na załadowanie pliku zawierającego słowa lub wartości, które chcesz wypróbować. Na szczęście Kali Linux posiada obszerny zbiór słowników i list słów, znajdujący się w katalogu */usr/share/wordlists*.

W naszym przykładzie użyjemy następującego pliku słownika:

/usr/share/wordlists/metasploit/http_default_users.txt



Teraz, gdy masz już przygotowane żądanie ze zdefiniowanymi pozycjami wejściowymi i listę wartości, naciśnij przycisk *Start Attack* (rozpocznij atak):



Jeżeli przyjrzysz się wynikom ataku przedstawionym na powyższym rysunku, z pewnością zauważysz, że wszystkie wypróbowane nazwy kont wygenerowały identyczne odpowiedzi; mówiąc ściślej, wszystkie konta oprócz jednego. Zwróć uwagę, że w przypadku żądania z kontem `admin` nadeszła odpowiedź o innej długości, a kiedy przyjrzysz się zawartości odpowiedzi, zobaczysz, że znajduje się tam pytanie o odzyskanie hasła. Wynika stąd, że `admin` jest w tym scenariuszu prawidłową nazwą konta użytkownika.

> Próby pozyskania listy nazw kont użytkowników można dokonać za każdym razem, gdy aplikacja zwraca różne odpowiedzi dla poprawnych i nieprawidłowych nazw kont. Ponadto niektóre aplikacje podczas rejestrowania nowego konta użytkownika sprawdzają, czy konto o takiej nazwie już istnieje, aby zapewnić jego unikatową nazwę. Jeżeli taka weryfikacja poprawności nazwy zostanie przeprowadzona przed przesłaniem formularza, oznacza to, że w badanym systemie istnieje usługa WWW, która przeprowadza wstępne sprawdzenie nazwy, i można tej usługi użyć do przeprowadzenia próby pozyskania listy kont użytkowników takiego systemu.

Pozyskiwanie haseł za pomocą ataków typu brute force i ataków słownikowych

Po zidentyfikowaniu listy kont użytkowników w aplikacji naturalnym krokiem jest próba znalezienia haseł dla tych użytkowników. Istnieje wiele metod pozyskiwania prawidłowych haseł od użytkowników, np.: podszywanie się pod oryginalną witrynę na innym serwerze i nakłanianie użytkowników za pomocą metod socjotechnicznych do przesyłania swoich informacji, wykorzystywanie źle zabezpieczonych mechanizmów odzyskiwania hasła czy próby odgadywania współużytkowanych haseł.

Brute force to metoda, która polega na wypróbowywaniu wszystkich możliwych kombinacji znaków w celu znalezienia prawidłowego hasła. Taka metoda może się całkiem dobrze sprawdzać w przypadku aplikacji, które zezwalają na stosowanie krótkich haseł, składających się maksymalnie z kilku znaków. Jeżeli takie hasła są dozwolone, to zazwyczaj istnieje spora szansa na to, że korzysta z nich co najmniej jeden użytkownik.

W przypadku dłuższych haseł atak typu brute force jest całkowicie niepraktyczny, ponieważ zazwyczaj będziesz musiał wysłać do aplikacji miliony lub nawet miliardy żądań, zanim odkryjesz jedno prawidłowe hasło, co powoduje, że czas niezbędny do przeprowadzenia takiego ataku jest znacznie dłuższy (ekstremalnie dłuższy) niż standardowe jeden lub dwa tygodnie przeznaczane zazwyczaj na przeprowadzenie całego testu penetracyjnego. W takiej sytuacji powinieneś polegać raczej na przewidywalności elementu ludzkiego — nawet jeżeli teoretycznie liczba możliwych kombinacji haseł składających się z ośmiu lub więcej znaków osiąga wartości naprawdę kosmiczne, my, ludzie, nieco podświadomie używamy tylko niewielkiego podzbioru tych kombinacji, a najpopularniejsze hasła z takiego podzbioru są bardzo często spotykane w praktyce.

Aby wykorzystać ten fakt, tworzone są słowniki, które zawierają takie często spotykane i domyślne hasła, jak również hasła, które wyciekły w poprzednich atakach na popularne witryny. Korzystając z tych słowników, możesz znacząco zminimalizować liczbę prób, które musisz przeprowadzić, aby znaleźć poprawne hasło lub zwiększyć szansę na znalezienie popularnego hasła, które było używane przez wiele osób.

> Od roku 2012 serwis SplashData co roku publikuje listy najczęściej używanych haseł, bazujące na analizach zbiorów haseł, które wyciekły i zostały upublicznione po udanych atakach hakerów na całym świecie. Wyniki z lat 2017 i 2016 można sprawdzić na stronach: *https://www.teamsid.com/worst-passwords-2017-full-list/* oraz *https://www.teamsid.com/worst-passwords-2016/*. Inną listą publikowaną co roku jest lista najpopularniejszych haseł przygotowywana przez deweloperów menedżera Keeper: *https://blog.keepersecurity.com/2017/01/13/most-common-passwords-of-2016-research-study/*.

Ataki na uwierzytelnianie typu Basic za pomocą pakietu THC Hydra

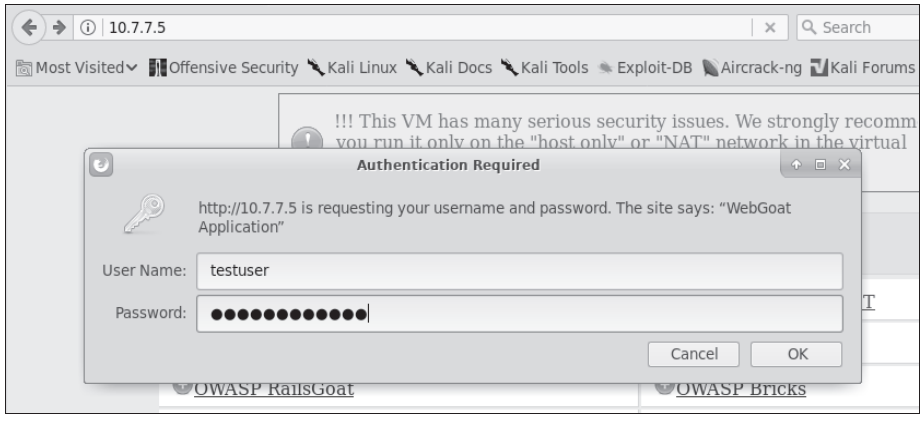
Pakiet **THC Hydra** od dawna jest ulubionym narzędziem do łamania haseł online zarówno wśród hakerów, jak i pentesterów.

Łamanie haseł w trybie online oznacza, że program w każdej iteracji dokonuje próby logowania do usługi sieciowej. Może to generować relatywnie duży ruch sieciowy i powodować pojawianie się ostrzeżeń na serwerze, zwłaszcza gdy jest on wyposażony w różne narzędzia bezpieczeństwa i systemy wykrywania włamań. Z tego powodu podczas przeprowadzania próby pozyskania haseł za pomocą ataku typu brute force lub ataku słownikowego na aplikację lub serwer powinieneś zachować szczególną ostrożność i ustawić parametry ataku tak, aby uzyskać najlepszą możliwą szybkość działania bez przeciążania serwera, generowania alertów lub blokowania kont użytkowników.

Dobrym podejściem do przeprowadzania ataków online, gdy środowisko celu jest monitorowane lub gdy atakowane konto jest automatycznie blokowane po wykonaniu określonej liczby nieudanych prób logowania, jest rozpoczęcie ataku z trzema lub czterema hasłami na użytkownika, ewentualnie z liczbą prób mniejszą niż próg blokady. Na początek wybierz najbardziej oczywiste lub często używane hasła (*password*, *admin* czy *12345678*), a jeżeli taka próba się nie powiedzie, wróć do etapu rozpoznania, postaraj się uzyskać więcej szczegółowych informacji o celu i wybrać lepszy zestaw haseł, a następnie spróbuj ponownie po kilku minutach lub godzinach.

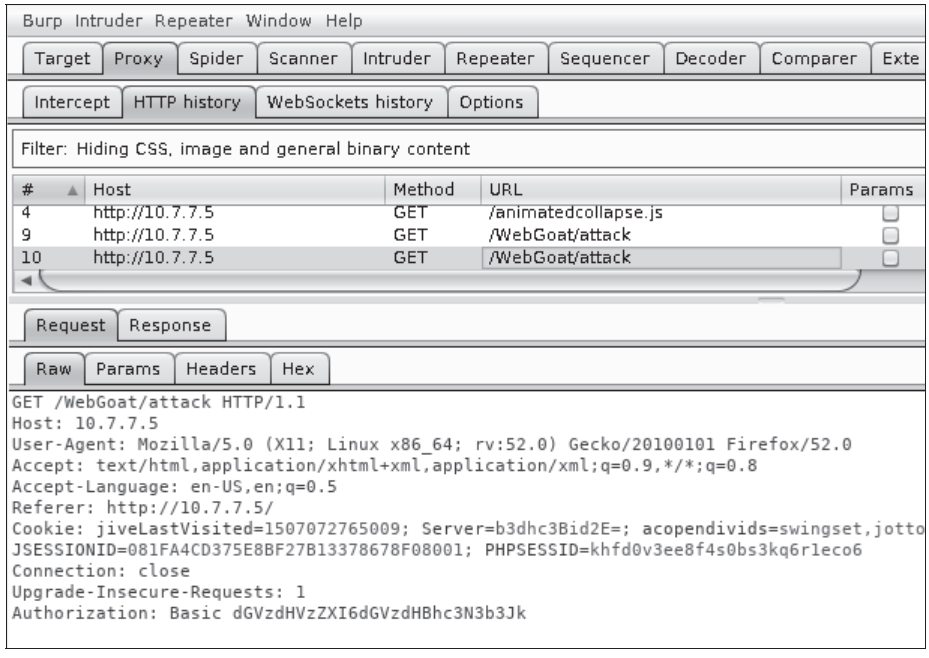
Pakiet THC Hydra ma możliwość łączenia się z szeroką gamą usług sieciowych, takich jak FTP, SSH, Telnet czy RDP. Wykorzystamy go do ataku słownikowego na serwer HTTP, który używa uwierzytelnienia typu Basic.

Najpierw musisz poznać adres URL, który faktycznie przetwarza dane logowania. Uruchom swoją maszynę z systemem Kali Linux, otwórz pakiet Burp Suite i skonfiguruj przeglądarkę tak, by używała go jako proxy. W naszym przykładzie użyjemy podatnej na atak maszyny wirtualnej i aplikacji WebGoat. Podczas próby uzyskania dostępu do WebGoat pojawia się okno dialogowe z prośbą o podanie danych logowania. Jeżeli przesyłasz dowolną, losową nazwę i hasło, ponownie otrzymasz to samo okno dialogowe:



Nawet jeżeli próba logowania się nie powiodła, żądanie jest już zarejestrowane w Burp Proxy. Następnie wyszukaj żądanie, w którym znajduje się nagłówek `Authorization: Basic` (patrz zrzut na następnej stronie).

Teraz wiesz, że adres URL odpowiadający za logowanie do aplikacji to *http://10.7.7.5/WebGoat/attack*. Jest to wystarczająca ilość informacji do uruchomienia Hydry, ale najpierw musisz przygotować listę możliwych nazw kont użytkowników i drugą, zawierającą hasła. W rzeczywistym scenariuszu listy potencjalnych nazw użytkowników i haseł zależą od organizacji, aplikacji i wiedzy użytkowników. W naszym przykładzie dla aplikacji WebGoat możesz użyć następującej listy prawdopodobnych nazw kont użytkowników:




```
adminwebgoat
administrator
user
test
testuser
```

Jeżeli chodzi o listę haseł, możesz wypróbować niektóre z najbardziej popularnych haseł oraz dodać różne warianty nazwy aplikacji:

```
123456
password
Password1
admin
webgoat
WebGoat
qwerty
123123
12345678
owasp
```

Zapisz listę użytkowników jako plik *users.txt*, a listę haseł jako plik *passwords.txt*. Najpierw uruchom polecenie `hydra` bez żadnych parametrów, aby wyświetlić opis składni i opcji wywołania polecenia:

```
root@kali:~# hydra
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations,
 or for illegal purposes.

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M F
ILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [servic
e://server[:PORT][/OPT]]

Options:
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -U        service module usage details
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)

Supported services: adam6500 asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get|post}
 http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md
5][s] mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres radmin2 rdp redis
rexec rlogin rpcap rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak teln
et[s] vmauthd vnc xmpp

Hydra is a tool to guess/crack valid login/password pairs. Licensed under AGPL
v3.0. The newest version is always available at http://www.thc.org/thc-hydra
Don't use in military or secret service organizations, or for illegal purposes.

Example:  hydra -l user -P passlist.txt ftp://192.168.0.1
```

Widać, że aby zdefiniować plik zawierający listę potencjalnych nazw kont użytkowników, musimy użyć opcji `-L`, do określenia pliku listy haseł musimy użyć opcji `-P`, a protokół, adres serwera i dodatkowe informacje możemy zdefiniować w następujący sposób: `protocol://server:port/optional`. Uruchom poniższe polecenie:

```
hydra -L users.txt -P passwords.txt http-get://10.7.7.5:8080/WebGoat/attack
```

```
root@kali:~# hydra -L users.txt -P passwords.txt http-get://10.7.7.5:8080/WebGoat/attack
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service or
 or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-10-19 12:26:41
[DATA] max 16 tasks per 1 server, overall 16 tasks, 60 login tries (l:6/p:10), ~4 tries pe
[DATA] attacking http-get://10.7.7.5:8080//WebGoat/attack
[8080][http-get] host: 10.7.7.5   login: webgoat   password: webgoat
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-10-19 12:26:42
```

Po uruchomieniu polecenia szybko się przekonasz, że w tym przykładzie do serwera możemy się zalogować, używając konta użytkownika o nazwie `webgoat` i hasła `webgoat`.

Jedną z bardzo użytecznych opcji wywołania programu Hydra jest `-e` z modyfikatorami `n`, `s` lub `r`, które mogą zmieniać proces logowania, wysyłając puste hasło (n), używając nazwy użytkownika jako hasła (s), odwracając nazwę użytkownika i używając jej jako hasła (r). Ciekawą i bardzo przydatną opcją jest również `-u`, która powoduje, że Hydra najpierw pobierze pierwsze hasło z pliku haseł i spróbuje się zalogować po kolei na wszystkie podane konta użytkowników, następnie pobiera drugie hasło i próbuje się zalogować po kolei na wszystkie konta itd., aż do wyczerpania się pliku haseł. Takie postępowanie może pomóc Ci uniknąć zablokowania przez niektóre mechanizmy obronne środowiska celu.

Ataki na uwierzytelnianie oparte na formularzach

Ponieważ proces logowania za pośrednictwem formularza nie jest w żaden sposób ustandaryzowany, a aplikacje internetowe są znacznie bardziej elastyczne pod względem sprawdzania poprawności i zapobiegania atakom, ataki typu brute force na systemy wykorzystujące uwierzytelnianie oparte na formularzach stanowią szczególne wyzwanie, ponieważ:

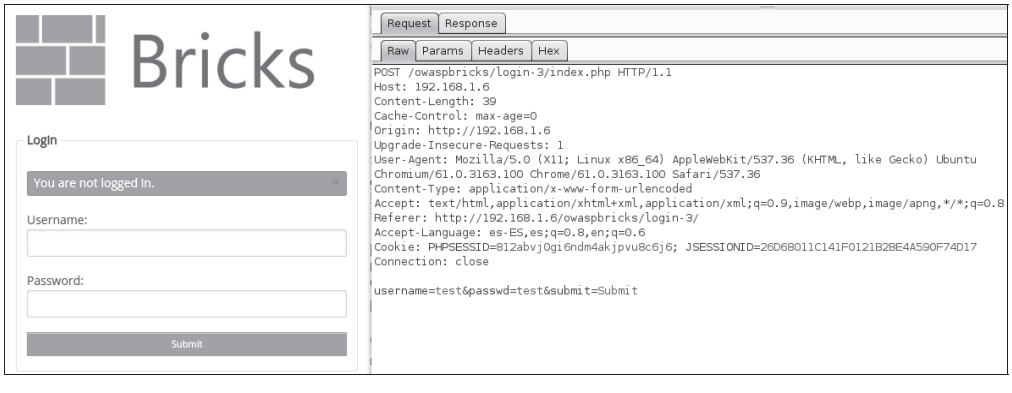
- Parametry, za pomocą których przekazywane są nazwy kont użytkowników oraz hasła, nie mają żadnej standardowej nazwy, pozycji ani formatu.
- Nie istnieje żaden standard, który określałby, jak powinna wyglądać standardowa negatywna i pozytywna odpowiedź na próbę logowania.
- Kontrole poprawności danych po stronie klienta i po stronie serwera mogą zapobiegać niektórym rodzajom ataków lub wielokrotnemu powtarzaniu przesyłanych żądań.
- Uwierzytelnienie można wykonywać w więcej niż jednym kroku; np. żądając podania nazwy konta użytkownika na jednej stronie i hasła na następnej stronie.

Szczęśliwie dla pentesterów większość aplikacji wykorzystuje podstawowy wzór formularza HTML, wysłany przez żądanie `POST`, gdzie nazwa użytkownika i hasło są przekazywane jako parametry, po pomyślnym zalogowaniu się następuje przekierowanie do strony głównej, a w przypadku niepowodzenia wyświetlany jest komunikat o wystąpieniu błędu lub następuje ponowne przekierowanie na stronę logowania. Teraz przeanalizujemy dwie metody, których można używać do przeprowadzenia ataku słownikowego na tego rodzaju formularz. Ta sama zasada odnosi się do prawie wszystkich mechanizmów uwierzytelniania opartych na formularzach, z pewnymi modyfikacjami sposobu interpretacji odpowiedzi i wymaganych do przesłania parametrów.

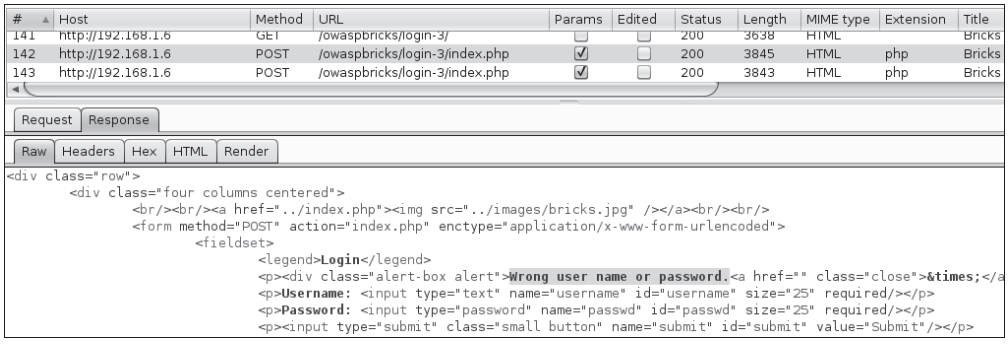
Zastosowanie modułu Burp Intruder

Podobnie jak w przypadku ataku na uwierzytelnianie typu Basic, aby przeprowadzić poprawny atak na formularz, musimy zidentyfikować żądanie, które realizuje faktyczne uwierzytelnienie, i sprawdzić jego parametry.

Na poniższym zrzucie ekranu po lewej stronie przedstawiono fragment formularza logowania aplikacji OWASP Bricks (w menu głównym maszyny wirtualnej BWA wybierz polecenie *Bricks/Login pages/Login #3* (Bricks/strony logowania/login numer 3)), a po prawej stronie fragment okna Burp Proxy z wyświetlonym żądaniem przesłanym za pomocą metody `POST`. Zwróć uwagę, że parametry `username` i `passwd` są wysyłane w treści żądania, a w jego nagłówku nie ma pola `Authorization`:

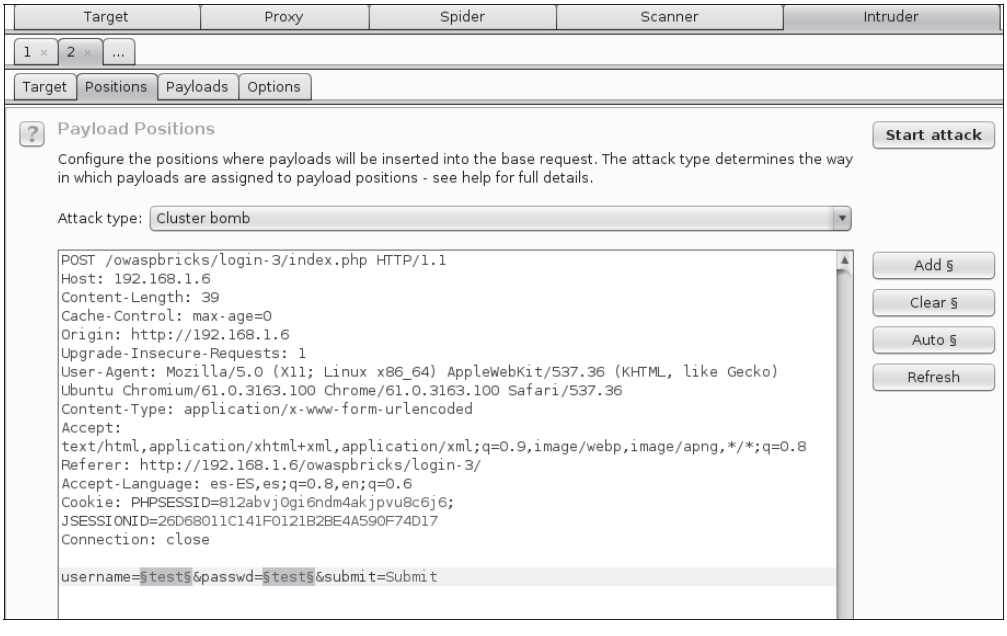


Aby wykonać atak słownikowy na tej stronie logowania, musimy najpierw przeanalizować otrzymaną odpowiedź, by sprawdzić, co odróżnia nieudane logowanie od udanego:

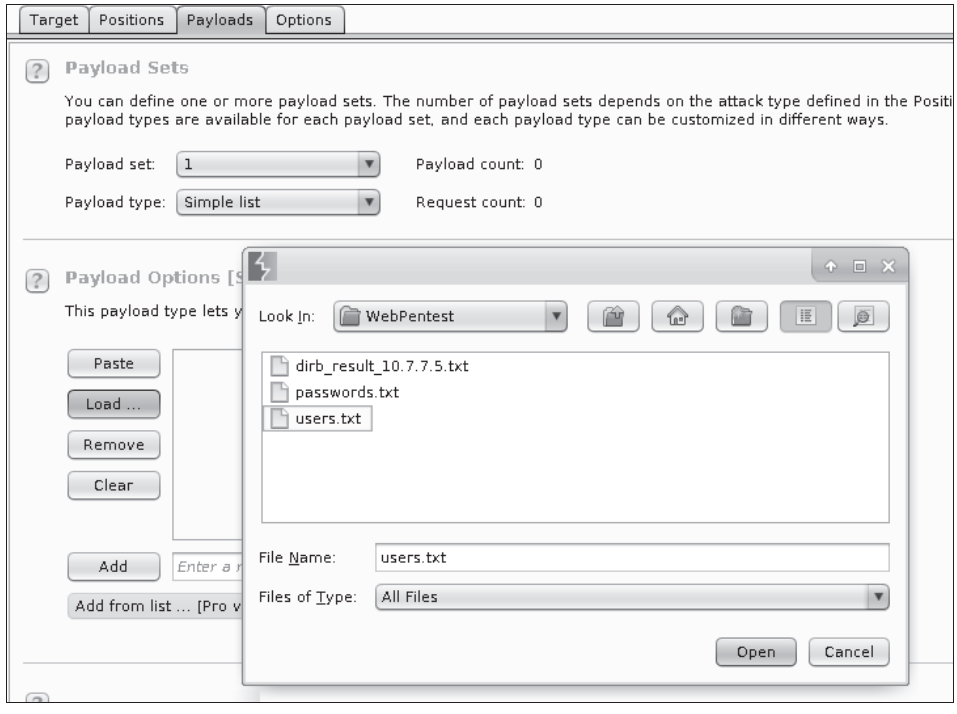


Na zrzucie ekranu można zauważyć, że w przypadku niepowodzenia logowania odpowiedź zawiera komunikat *Wrong user name or password* (niepoprawna nazwa konta użytkownika lub hasło). Możemy śmiało założyć, że takiego komunikatu nie zobaczymy w odpowiedzi potwierdzającej pomyślne zalogowanie się do aplikacji.

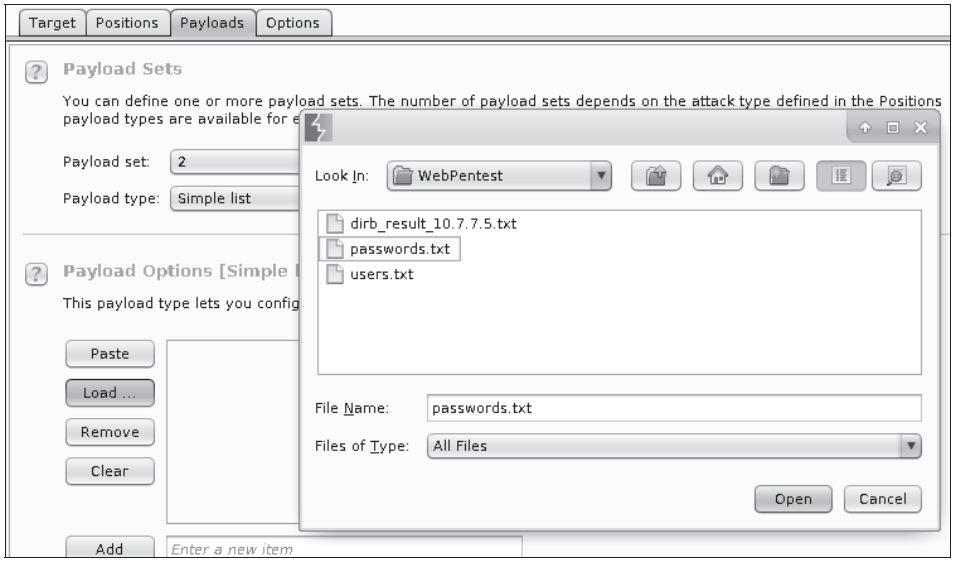
Następnie przekazujemy to żądanie do modułu Intruder, definiujemy parametry `username` oraz `passwd` jako dane wejściowe i jako typ ataku wybieramy *Cluster bomb*:



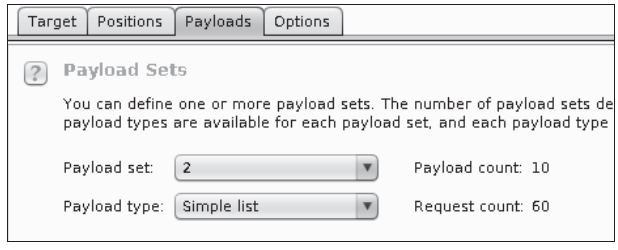
Następnie przechodzimy na kartę *Payloads* (ładunki), z listy rozwijanej *Payload Set* (zestaw ładunków) wybieramy ładunek numer 1 i ładujemy plik *users.txt* zawierający nazwy użytkowników, z którego korzystaliśmy wcześniej:



Jako zestaw ładunków numer 2 użyjemy pliku haseł *passwords.txt*, którego również używaliśmy w poprzednim ćwiczeniu:



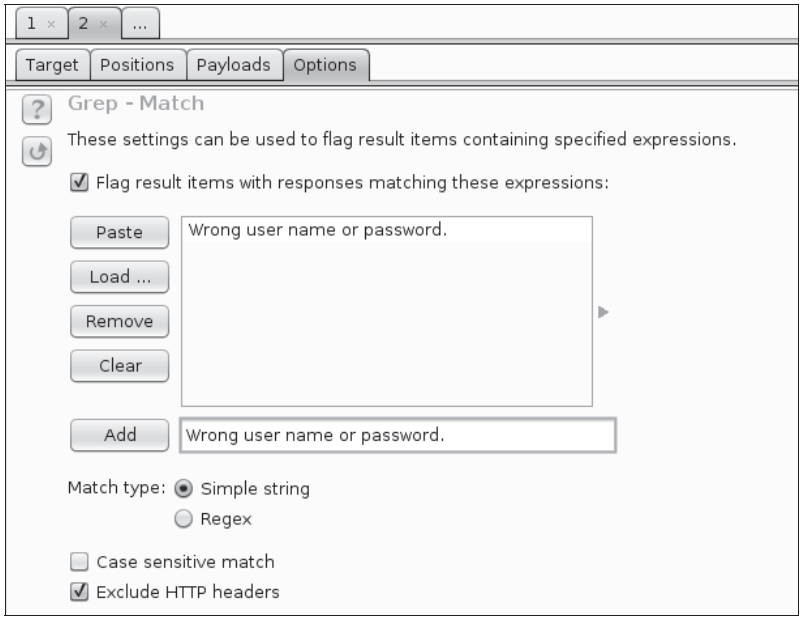
Jak widać na kolejnym rysunku, do serwera zostanie wysłanych 60 żądań, ponieważ w naszych przykładowych plikach znajdowało się 6 nazw kont użytkowników i 10 możliwych haseł:



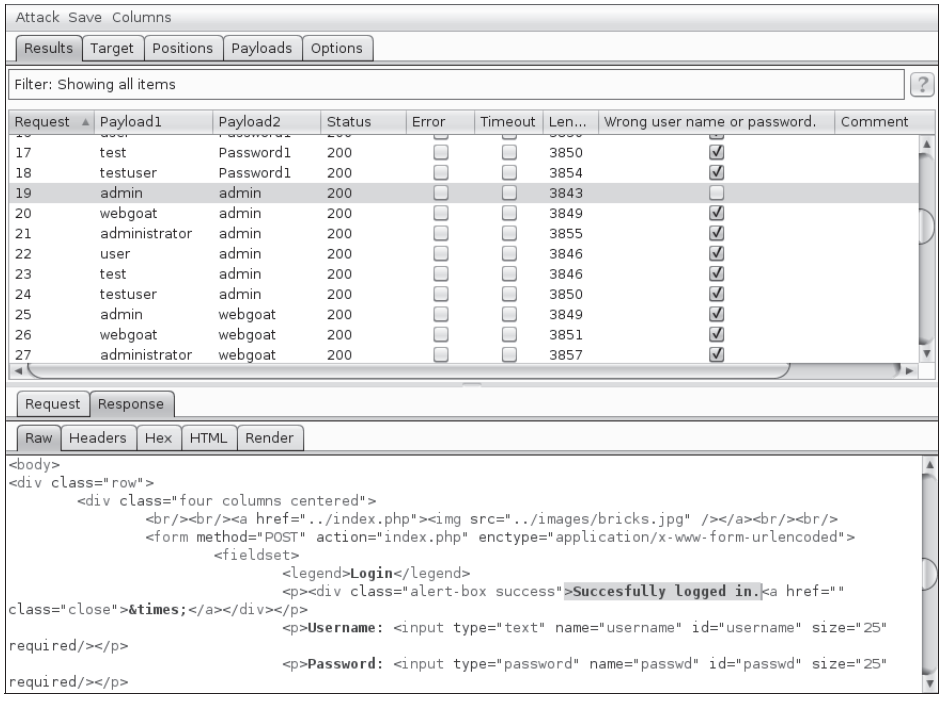
W tym momencie możesz już rozpocząć atak, przeanalizować otrzymane odpowiedzi i sprawdzić, czy jakaś kombinacja logowania zakończyła się powodzeniem. Moduł Burp Intruder ma jeszcze kilka innych funkcji, które mogą znacząco ułatwić Ci życie, nie tylko w prostych scenariuszach takich jak ten, ale także podczas atakowania złożonych, rozbudowanych aplikacji w rzeczywistym środowisku celu. Aby się o tym przekonać, przejdź na kartę *Options* (opcje), a następnie wybierz opcję *Grep — Match* (wyszukiwanie z użyciem wyrażenia grep), która powoduje automatyczne wyszukiwanie podanego tekstu w otrzymanych odpowiedziach, dzięki czemu możemy łatwo zidentyfikować żądania logowania, które zakończyły się powodzeniem. Zaznacz opcję *Flag result items with responses matching these expressions* (oznacz żądania, których zawartość pasuje do podanych wyrażeń), wyczyść bieżącą listę wyrażeń i w polu *Enter a new item* (wprowadź nowy element) wpisz następujące wyrażenie:

```
Wrong user name or password
```

Naciśnij klawisz *Enter* lub kliknij przycisk *Add* (dodaj). Moduł Intruder automatycznie wyróżni wszystkie odpowiedzi zawierające podany komunikat, w związku z tym te, które nie zostaną oflagowane, mogą oznaczać pomyślne logowanie. Jeżeli wiesz, jaki komunikat jest przesyłany w odpowiedzi w przypadku poprawnej próby logowania, możesz wprowadzić go zamiast komunikatu użytego powyżej i w ten sposób bezpośrednio zidentyfikować poprawny zestaw poświadczeń:



Rozpocznij atak i poczekaj na wyniki:



Wygląda na to, że udało nam się znaleźć przynajmniej jedną prawidłową kombinację nazwy konta użytkownika i hasła dostępu.

Zastosowanie pakietu THC Hydra

Wśród wielu protokołów obsługiwanych przez pakiet Hydra znajdziemy takie jak `http-get-form`, `http-post-form`, `https-get-form` i `https-post-form`, które są formularzami logowania HTTP i HTTPS przesyłanymi za pomocą metod odpowiednio `GET` i `POST`. Korzystając z informacji z poprzedniego ćwiczenia, możemy zatem uruchomić atak słownikowy z użyciem pakietu Hydra za pomocą następującego polecenia:

```
hydra 10.7.7.5 http-form-post
"/owaspbricks/login-3/index.php:username=^USER^&passwd=^PASS^&submit=Submit
:Wrong user name or password." -L users.txt -P passwords.tx
```

```
root@kali:~/WebPentest# hydra 10.7.7.5 http-form-post "/owaspbricks/login-3/index.php:username=^USER^
&passwd=^PASS^&submit=Submit:Wrong user name or password." -L users.txt -P passwords.txt
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations
, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-10-26 14:16:36
[DATA] max 16 tasks per 1 server, overall 16 tasks, 60 login tries (l:6/p:10), ~4 tries per task
[DATA] attacking http-post-form://10.7.7.5:80//owaspbricks/login-3/index.php:username=^USER^&passwd=^
PASS^&submit=Submit:Wrong user name or password.
[80][http-post-form] host: 10.7.7.5   login: admin   password: admin
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 5 final worker threads did not complete until end.
[ERROR] 5 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2017-10-26 14:16:41
```

Z pewnością zauważyłeś, że w tym przypadku składnia polecenia wywołującego program Hydra jest nieco inna niż poprzednio. Przyjrzyjmy się temu bliżej:

1. Rozpoczynamy od nazwy polecenia `hydra` i adresu IP hosta docelowego (`hydra 10.7.7.5`).
2. Następnie podajemy protokół lub usługę, którą chcemy przetestować (`http-form-post`).
3. Dalej dodajemy parametry specyficzne dla protokołu, które umieszczamy w cudzysłowach prostych (`""`) i rozdzielamy dwukropkami:
   - Adres URL (`/owaspbricks/login-3/index.php`).
   - Treść żądania ze wskazaniem miejsca `^USER^`, gdzie Hydra powinna umieścić nazwy użytkowników, i `^PASS^` dla miejsca, w którym powinny się znaleźć hasła.
   - Nieudany komunikat logowania (`Wrong user name or password.`).
   - Na końcu podajemy nazwy plików z listą użytkowników (opcja `-L`) oraz haseł (opcja `-P`).

Mechanizm resetowania hasła

Innym, powszechnie spotykanym i często podatnym na atak elementem aplikacji internetowych jest implementacja mechanizmu odzyskiwania i resetowania hasła.

Ponieważ aplikacje sieciowe muszą być przyjazne dla użytkownika, a zdarza się, że niektórzy użytkownicy zapominają swoje hasła, aplikacje muszą zawierać sposób, który będzie umożliwiał im resetowanie lub odzyskiwanie haseł. Opracowanie bezpiecznego sposobu wykonania takiej operacji nie jest łatwym zadaniem i w wielu przypadkach kończy się pozostawieniem w aplikacji słabego ogniwa, które może wykorzystać pentester lub napastnik.

Odzyskiwanie hasła zamiast resetowania

Kiedy deweloper aplikacji staje w obliczu sytuacji, że użytkownik nie pamięta swojego hasła, może wybrać jedno z dwóch najczęściej stosowanych rozwiązań:

1. Pozwolić użytkownikowi na odzyskanie starego hasła.
2. Pozwolić użytkownikowi na zresetowanie starego hasła i ustawienie nowego.

Warto jednak zauważyć, że sam fakt, iż aplikacja pozwala użytkownikowi na odzyskanie starego hasła, implikuje istnienie pewnych oczywistych luk bezpieczeństwa w projekcie aplikacji:

- Zamiast używania jednokierunkowego algorytmu haszującego, który jest najlepszym sposobem na przechowywanie haseł, są one zapisywane w bazie danych w sposób umożliwiający ich odzyskanie.
- Po stronie serwera hasło użytkownika może być odzyskane przez administratora systemu lub innego użytkownika posiadającego odpowiednie uprawnienia (np. pracownika działu wsparcia technicznego). Zdeterminowany napastnik może również być w stanie to zrobić poprzez inżynierię społeczną lub wykorzystanie potencjalnych podatności i luk w systemie zabezpieczeń.

- Odzyskane hasło jest potencjalnie narażone na przechwycenie przez osoby trzecie podczas przekazywania użytkownikowi, co zwykle odbywa się za pośrednictwem poczty elektronicznej, telefonicznie lub przez wyświetlenie na stronie internetowej. Istnieje wiele sposobów, w których ktoś znajdujący się w tej samej sieci lub nawet osoba postronna może przechwycić takie informacje.

Typowe błędy w mechanizmach resetowania hasła

Bardzo popularną metodą, która umożliwia użytkownikom odzyskiwanie lub resetowanie haseł, jest zadawanie jednego lub więcej pytań, na które odpowiedź powinien znać tylko prawowity użytkownik. Obejmuje to pytania o miejsce urodzenia, pierwszą szkołę, imię pierwszego zwierzaka czy panieńskie nazwisko matki. Problemy pojawiają się, gdy pytania zadawane przez aplikację nie są tajemnicą dla potencjalnego napastnika, a lawinowo narastają, jeżeli użytkownik jest osobą o wysokim stopniu rozpoznawalności, np. celebrytą lub politykiem, gdy wiele szczegółów dotyczących jego życia jest publicznie dostępne.

Kolejny sposób ochrony polega na tym, że aplikacja nie daje bezpośredniego dostępu do funkcji resetowania hasła, ale wysyła wiadomość e-mail lub SMS z łączem do strony pozwalającej na resetowanie hasła. Takie rozwiązanie sprawdza się, o ile aplikacja posiada odpowiednie adresy e-mail bądź numery telefonów zapisane w swojej bazie. Jeżeli jednak źle zaprojektowana aplikacja podczas procesu odzyskiwania bądź resetowania hasła prosi o podanie odpowiedniego adresu czy numeru telefonu i nie porównuje wprowadzonych danych z danymi zapisanymi w bazie podczas rejestracji konta, istnieje prawdopodobieństwo, że złośliwy użytkownik może sfałszować te informacje, zastąpić numer bądź adres użytkownika swoim i w nieuprawniony sposób odzyskać hasło użytkownika.

Jeżeli adres e-mail lub numer telefonu są poprawnie zweryfikowane i nie można ich sfałszować, nadal istnieje ryzyko, że łącze prowadzące do strony resetowania hasła nie zostanie poprawnie wdrożone. Czasami takie łącza zawierają parametr wskazujący identyfikator konta (np. jego numer) lub nazwę użytkownika, którego hasło ma zostać zresetowane. W takim przypadku wszystko, co musisz zrobić, to wygenerować nowe łącze do resetowania hasła za pomocą konta użytkownika, które kontrolujesz, a następnie zmienić odpowiedni parametr reprezentujący konto na nazwę użytkownika, którego hasło chcesz zresetować.

Innym często spotykanym błędem jest to, że takie wygenerowane łącze używane do resetowania hasła nie jest unieważniane po pierwszym zgodnym z prawem użyciu. W takim przypadku, jeżeli złośliwy napastnik uzyska dostęp do takiego łącza, będzie mógł w dowolnej chwili ponownie z niego skorzystać i zresetować hasło użytkownika.

Luki w implementacjach mechanizmu 2FA

Najbardziej rozpowszechnioną formą mechanizmu uwierzytelniania wieloskładnikowego (ang. **MFA** — *Multi-factor Authentication*) w aplikacjach internetowych jest użycie losowo wygenerowanej liczby (składającej się najczęściej z czterech do ośmiu cyfr) używanej jako hasło jednorazowe (ang. **OTP** — *One-Time Password*), którą użytkownik otrzymuje ze specjalnego urządzenia, z aplikacji mobilnej (takiej jak Google Authenticator, Authy, 1Password czy LastPass Authenticator) lub za pośrednictwem wiadomości SMS czy e-mail wysłanej przez serwer na żądanie.

Podczas przeprowadzania testu penetracyjnego możemy wykryć i wykorzystać niektóre podatności i luki w zabezpieczeniach, które wkradły się na etapie wdrożenia takiego rozwiązania. Aby to się udało, muszą być spełnione następujące warunki:

- Hasła jednorazowe nie są całkowicie losowe i można je przewidzieć lub obliczyć.
- Hasła jednorazowe nie są połączone z użytkownikiem, do którego są przypisane. Oznacza to, że możesz wygenerować OTP dla jednego użytkownika i używać go z innym.
- To samo hasło lub token mogą być użyte wiele razy.
- Nie ma ograniczeń dotyczących liczby prób przesłania hasła jednorazowego. Taki błąd w implementacji otwiera możliwość ataków typu brute force, których szansa powodzenia jest całkiem spora, ponieważ hasła OTP są zwykle krótkimi ciągami liczb.
- Przed wysłaniem hasła jednorazowego za pośrednictwem wiadomości e-mail lub SMS dane użytkownika nie są weryfikowane, co pozwala napastnikowi na podszywanie się pod adres e-mail lub numer telefonu.
- Czas wygaśnięcia hasła jednorazowego jest zbyt długi w stosunku do potrzeb aplikacji. Taka sytuacja powoduje nadmierne rozszerzenie okna czasowego, w którym napastnik może pozyskać lub przechwycić prawidłowy nieużywany token.
- Utworzenie nowego hasła jednorazowego nie powoduje automatycznego unieważnienia poprzednich haseł. Przykładowo w sytuacji, kiedy użytkownik ponownie prosi o token lub hasło, ponieważ poprzednia operacja się nie powiodła, napastnik może użyć wcześniej wygenerowanego hasła OTP do wykonania innej operacji, nawet po tym, jak legalnie przeprowadzana przez użytkownika operacja została już zakończona.
- Do uwierzytelniania MFA powinno się używać tego samego urządzenia, z którego uzyskuje się dostęp do aplikacji. W dzisiejszych czasach ludzie mają na smartfonach różne aplikacje bankowe, osobistą pocztę e-mail, aplikacje sieci społecznościowych, służbową pocztę e-mail i wiele innych aplikacji, dlatego powinieneś się dwa razy zastanowić, zanim do otrzymywania bądź generowania haseł jednorazowych użyjesz tego samego smartfona, za pośrednictwem którego łączysz się ze swoim bankiem lub odbierasz służbową pocztę.

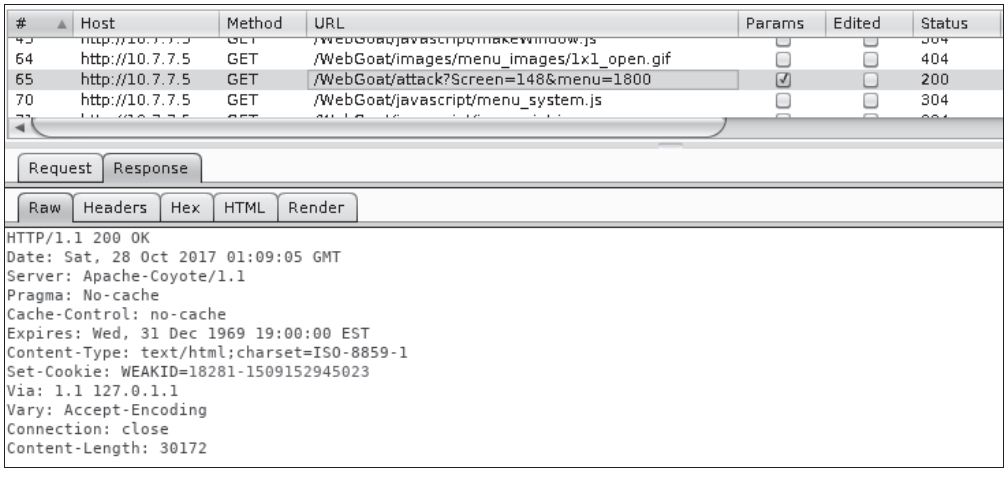
Wykrywanie i wykorzystywanie niewłaściwego zarządzania sesjami

Jak już wcześniej wspomniano, zarządzanie sesjami pozwala aplikacji śledzić aktywność użytkowników i sprawdzać warunki autoryzacji bez konieczności dołączania danych uwierzytelniających użytkownika do każdego żądania. Oznacza to, że jeżeli zarządzanie sesją nie jest prawidłowo zaimplementowane, użytkownik może uzyskać dostęp do informacji innych użytkowników lub wykonywać działania wykraczające poza swój poziom uprawnień, a jeżeli może to zrobić użytkownik, to równie dobrze może taki dostęp uzyskać napastnik zewnętrzny.

Zastosowanie modułu Burp Sequencer do oceny jakości identyfikatorów sesji

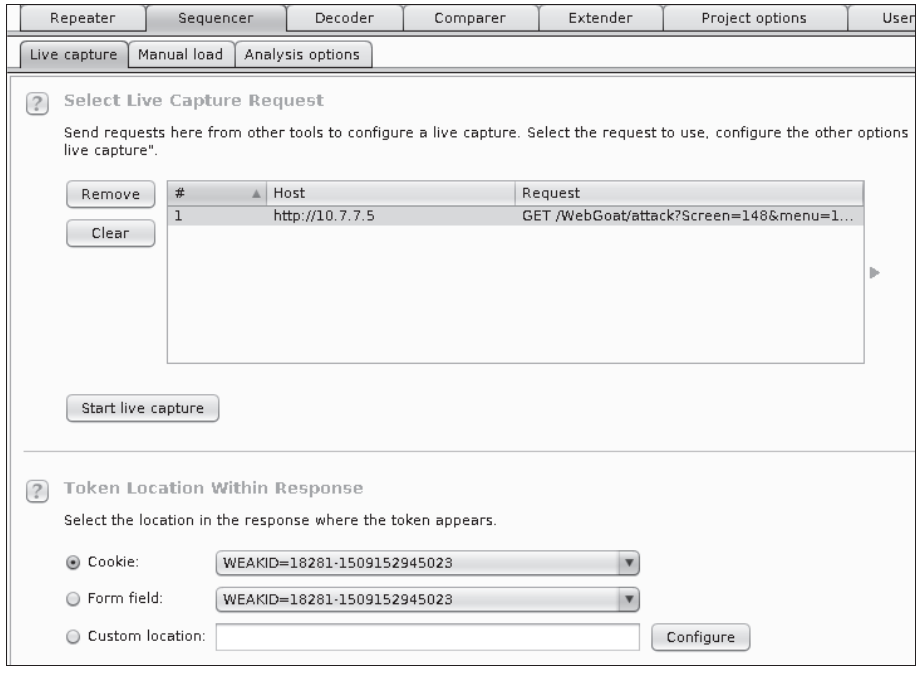
Burp Sequencer to narzędzie do analizy statystycznej, które umożliwia zbieranie dużej ilości danych, takich jak identyfikatory sesji, i wykonywanie na nich obliczeń w celu oceny, czy są one generowane losowo, czy też są tworzone lub kodowane w inny sposób. Jest to przydatne, gdy mamy do czynienia ze złożonymi plikami cookie sesji, ponieważ daje wyobrażenie o tym, jak takie pliki są generowane i jaki istnieje sposób na przewidywanie ich zawartości i wykorzystanie ich do przeprowadzenia ataku.

Aby korzystać z modułu Burp Sequencer, najpierw musisz znaleźć odpowiedź, która ustawia plik cookie sesji. Zwykle jest to odpowiedź na udane logowanie, która w nagłówku posiada ustawione pole `Set-Cookie`. Na poniższym zrzucie ekranu pokazujemy odpowiedź, która ustawia plik cookie sesji (`WEAKID`) w ćwiczeniu ilustrującym przechwytywanie sesji w aplikacji WebGoat — aby to zobaczyć na własne oczy, wybierz z menu opcję *WebGoat/Session Management Flaws/Hijack a Session* (WebGoat/luki w zarządzaniu sesjami/przechwytywanie sesji):

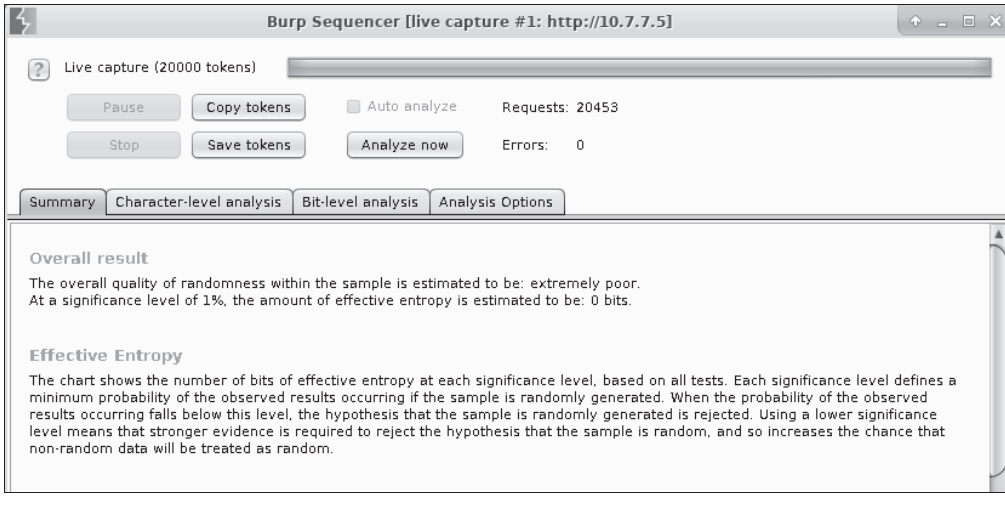


Na pierwszy rzut oka wartość odpowiedzi może się wydawać unikatowa i trudna do odgadnięcia. Pierwsza część wygląda jak identyfikator, a druga wydaje się być jakimś znacznikiem czasu, być może czasem wygaśnięcia ważności podanym w nanosekundach. A przecież odgadnięcie, w której dokładnie nanosekundzie kończy się sesja, powinno być bardzo trudne, prawda? Jak się za chwilę przekonasz, nie jest to najlepsze podejście.

Odszukaj tę odpowiedź w historii pakietów przechwyconych przez moduł Burp Proxy i kliknij ją prawym przyciskiem myszy. Na ekranie pojawi się menu podręczne, w którym znajdziesz opcję *Send to Sequencer* (wyślij do sekwencera). Po przejściu do modułu Burp Sequencer powinieneś wskazać, na której części odpowiedzi ma się skupić:

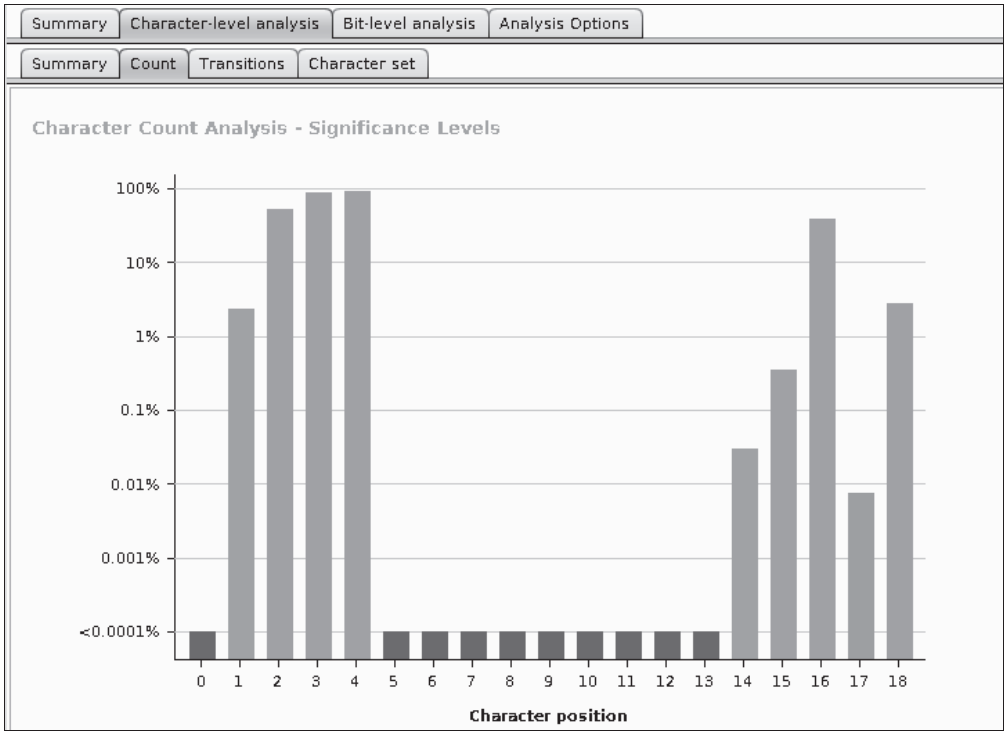


W zależności od potrzeb możesz przeanalizować plik cookie, pole formularza lub inną, dowolnie wybraną część odpowiedzi. W naszym przypadku wybierz ciasteczko `WEAKID` i naciśnij przycisk *Start live capture* (uruchom przechwytywanie na żywo). Program rozpocznie wysyłanie żądań do serwera w celu przechwycenia jak największej liczby różnych wartości cookie. Po zakończeniu przechwytywania naciśnij przycisk *Analyze now* (analizuj teraz), aby rozpocząć analizę statystyczną zebranych danych. Po zakończeniu analizy Sequencer wskaże, czy analizowana wartość jest wystarczająco losowa i czy jest dobrym wyborem jako identyfikator sesji. Jak widać, w naszym przykładzie ciasteczko `WEAKID` jest słabe i łatwe do przewidzenia:



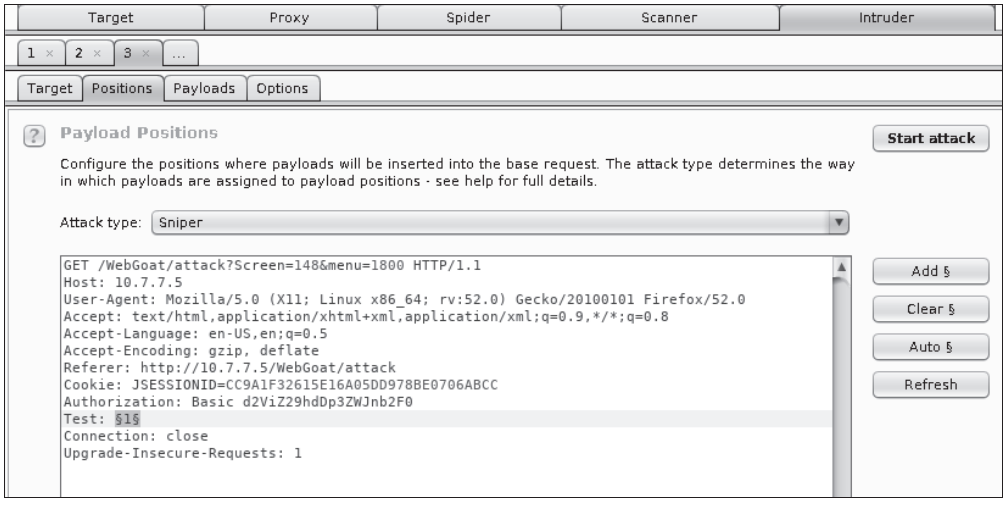
Entropia jest miarą poziomu losowości określonych danych. Wyniki naszego przykładu pokazują, że wartość `WEAKID` ma zerową losowość, co oznacza, że jest całkowicie przewidywalna i nie jest dobrym wyborem jako identyfikator sesji. Moduł Sequencer dostarcza również bardziej szczegółowych informacji na temat dystrybucji i znaczenia każdego bajta i bitu w łańcuchach.

Na kolejnym zrzucie ekranu, zamieszczonym poniżej, zobaczysz wykres analizy znaków. Widać, że znaki na pozycjach `3`, `4`, `15`, `16` i `18` zmieniają się znacznie częściej niż znaki na pozycjach `0` lub `5` do `13`, które w ogóle się nie zmieniają. Znaki na pozycjach od `0` do `4` sugerują obecność licznika lub liczbę rosnącą, ponieważ ostatni znak zmienia się znacznie częściej niż poprzednie itd. Zweryfikujemy to w następnej sekcji.

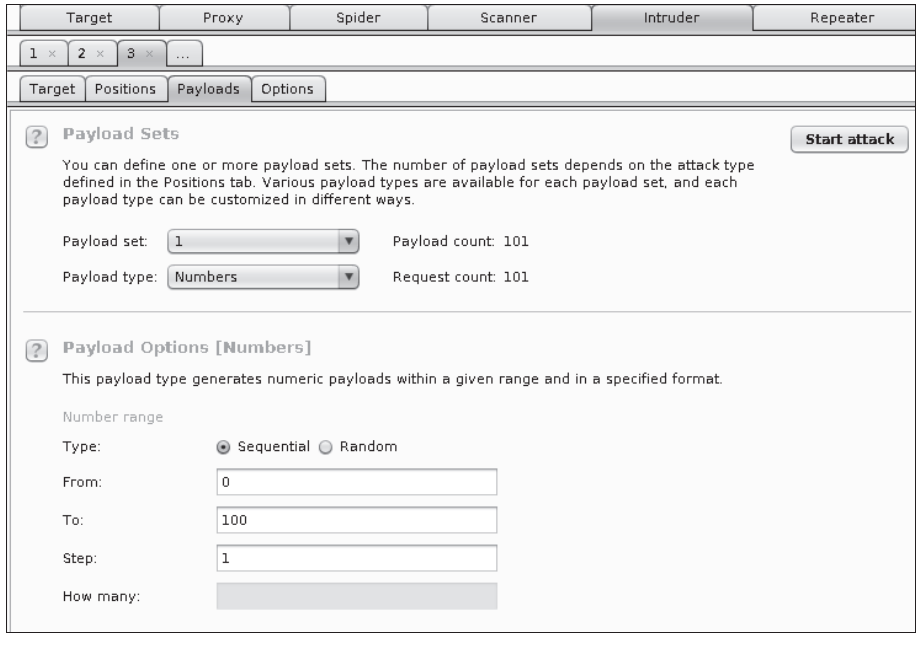


Przewidywanie wartości identyfikatorów sesji

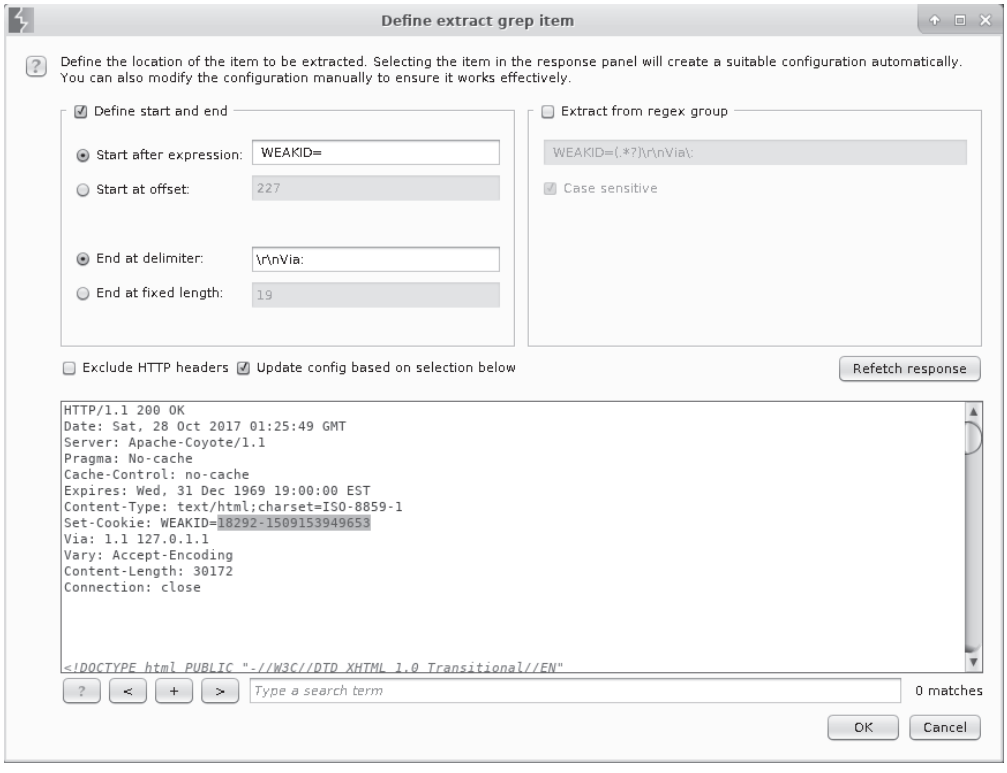
Zidentyfikowaliśmy identyfikator sesji, który wydaje się przewidywalny. Teraz spróbujmy znaleźć prawidłową sesję. Aby to zrobić, weźmiemy to samo żądanie, które ustawiało nasz plik cookie, i przekażemy je do modułu Intruder. W takim przypadku wystarczy powtórzyć takie żądanie kilka razy. Aby jednak moduł Intruder mógł działać prawidłowo, musi mieć zdefiniowane punkty wstawiania. Dodaj zatem do żądania pole (`Test: 1`) i jako pozycję wstawiania wskaż jego wartość:



W tym przykładzie wyślemy 101 żądań, zatem z listy *Payload type* (typ ładunku) wybierz opcje *Numbers* (liczby), a następnie zaznacz opcję *Type: Sequential* (typ sekwencyjny) i jako zakres wartości wybierz od 0 do 100 z krokiem 1 (kolejno pola: *From*, *To*, *Step*):

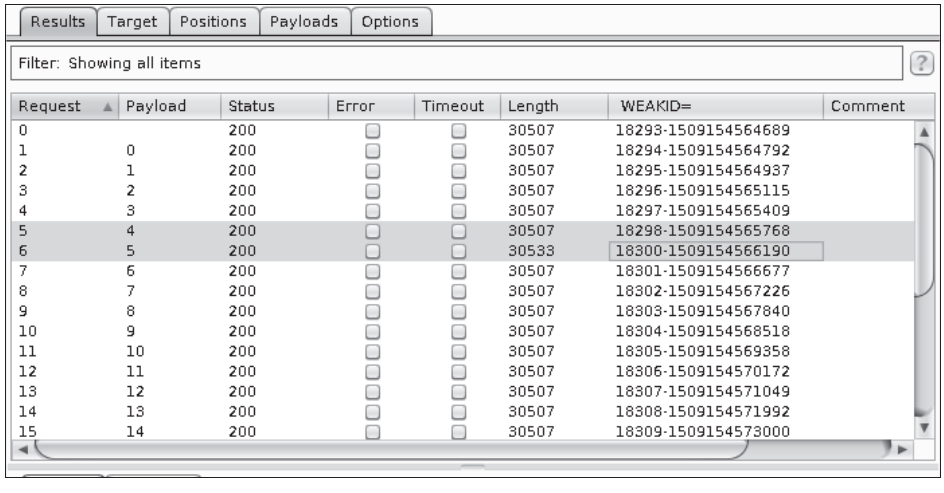


Teraz przejdź na kartę *Options* (opcje) i w sekcji *Grep-Extract* (wyodrębnianie z użyciem wyrażenia grep) dodaj jedno wyrażenie. Upewnij się, że opcja *Update config based on selection below* (aktualizuj konfigurację na podstawie poniższego wyboru) jest włączona, i zaznacz tylko wartość tego pliku cookie:



Naciśnij przycisk *OK*, a następnie *Start attack* (rozpocznij atak).

Teraz możesz zobaczyć wartość `WEAKID` w tabeli wyników działania modułu Intruder i sprawdzić, czy pierwsza część wartości cookie jest liczbą sekwencyjną, a druga część zawsze rośnie. Zależy to od czasu otrzymania żądania przez serwer. Jeżeli przyjrzysz się poniższemu zrzutowi ekranu, z pewnością zauważysz, że w sekwencji wartości są pewne luki:



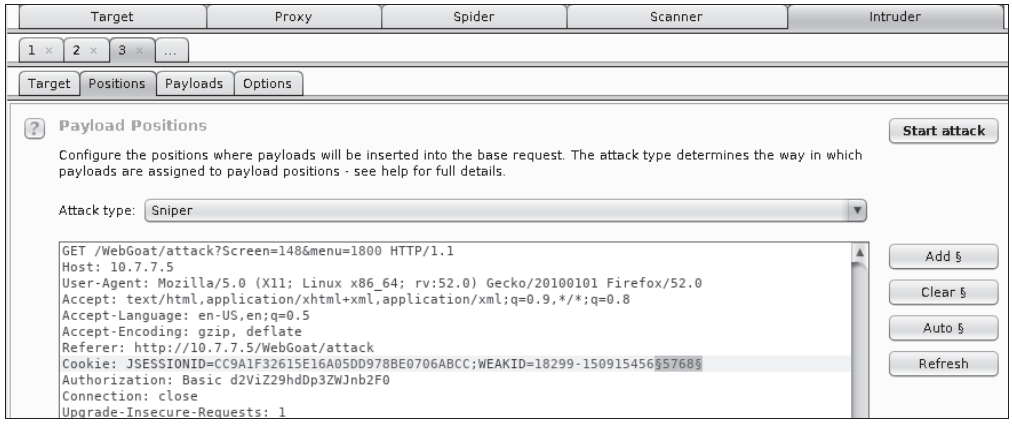
Request	Payload	Status	Error	Timeout	Length	WEAKID=	Comment
0		200			30507	18293-1509154564689	
1	0	200			30507	18294-1509154564792	
2	1	200			30507	18295-1509154564937	
3	2	200			30507	18296-1509154565115	
4	3	200			30507	18297-1509154565409	
5	4	200			30507	18298-1509154565768	
6	5	200			30533	18300-1509154566190	
7	6	200			30507	18301-1509154566677	
8	7	200			30507	18302-1509154567226	
9	8	200			30507	18303-1509154567840	
10	9	200			30507	18304-1509154568518	
11	10	200			30507	18305-1509154569358	
12	11	200			30507	18306-1509154570172	
13	12	200			30507	18307-1509154571049	
14	13	200			30507	18308-1509154571992	
15	14	200			30507	18309-1509154573000	

Pierwsza część identyfikatora bieżącej aktywnej sesji to 18299. Wiemy o tym, ponieważ nie otrzymaliśmy tej wartości z serwera, i wiemy, że jest inkrementowana z każdym żądaniem. Wiemy też, że druga część to znacznik czasu i że zależy również od czasu przypisania ciasteczka sesyjnego. Wynika stąd, że druga część poszukiwanej wartości musi się znajdować pomiędzy dwiema wartościami, które już znamy: 1509154565768 i 1509154566190. Ponieważ różnica między tymi liczbami jest mała (422), możemy z łatwością użyć Intrudera do próby znalezienia tej wartości metodą brute force.

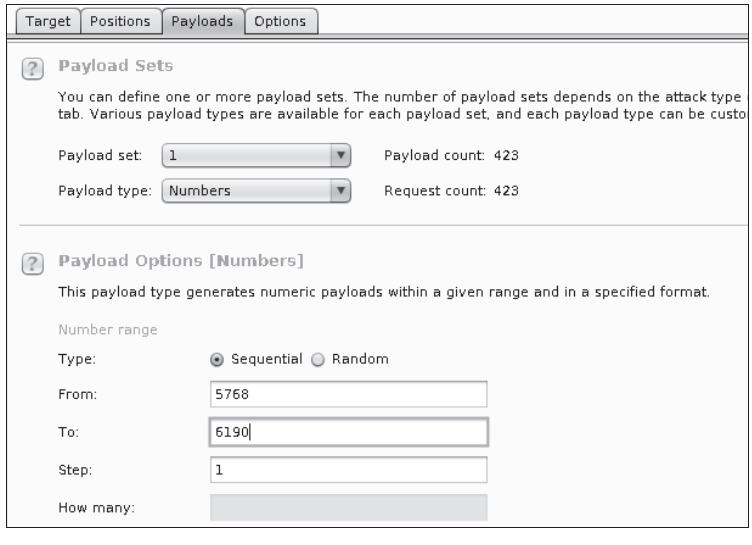
Aby to zrobić, weźmiemy pierwotne żądanie i przekażemy je ponownie do modułu Intruder. Tym razem dodamy do niego plik cookie. Bezpośrednio po wartości pola `JSESSIONID` dodaj następujące elementy (pamiętaj, aby odpowiednio dopasować wartości do swoich wyników):

```
; WEAKID = 18299-1509154565768
```

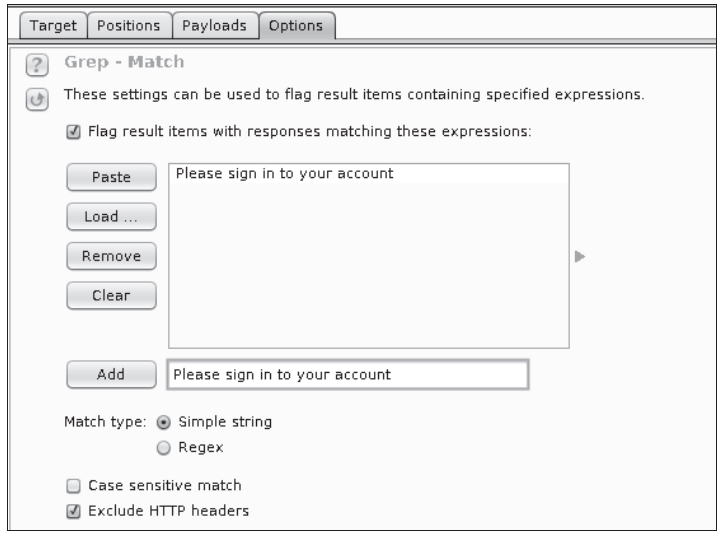
Zaznacz cztery ostatnie znaki i ustaw tam znacznik pozycji:



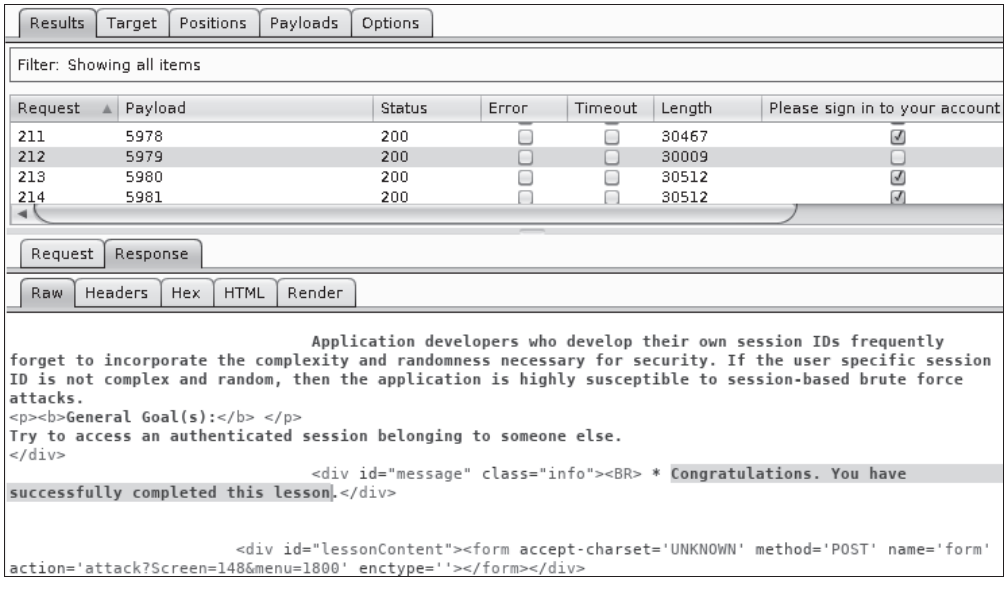
Przejdź teraz na kartę *Payloads* (ładunki) i przekonaj się, że w ataku zostaną użyte liczby od 5768 do 6190:



Na końcu dodaj wyrażenie grep, które zasygnalizuje Ci, kiedy osiągniesz sukces. W tym momencie znamy tylko komunikat, który otrzymuje nieuwierzytelniony użytkownik. Zakładamy tutaj, że uwierzytelniony użytkownik (czyli użytkownik, który posiada ważny plik cookie sesji) nie będzie musiał się logować:



Uruchom atak i poczekaj, aż działanie Intrudera przyniesie jakieś wyniki:



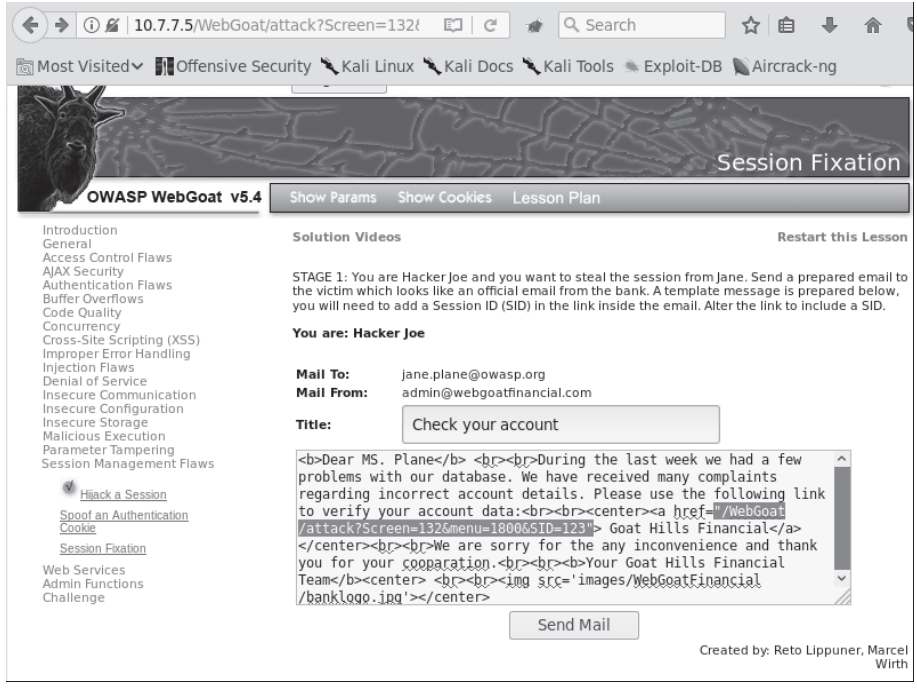
Masz teraz poprawny identyfikator sesji. Aby z niego skorzystać, wystarczy wymienić wartość pliku cookie sesji na tę, którą właśnie znalazłeś, a następnie odwiedzić stronę i przejąć kontrolę nad sesją innego użytkownika. Zostawimy Ci to jednak do samodzielnego przetestowania.

Ataki typu Session Fixation

Czasami do generowania identyfikatora sesji wykorzystywane są informacje dostarczone przez użytkownika lub, co gorsza, informacje dostarczone przez użytkownika *stają się* identyfikatorem sesji. W takim przypadku napastnik może zmusić użytkownika do użycia predefiniowanego identyfikatora, a następnie monitorowania aplikacji i sprawdzania, kiedy ten użytkownik rozpocznie sesję. Taki rodzaj złośliwego działania nazywamy atakiem z wykorzystaniem spreparowanych stałych identyfikatorów sesji lub w skrócie **atakiem typu Session Fixation**.

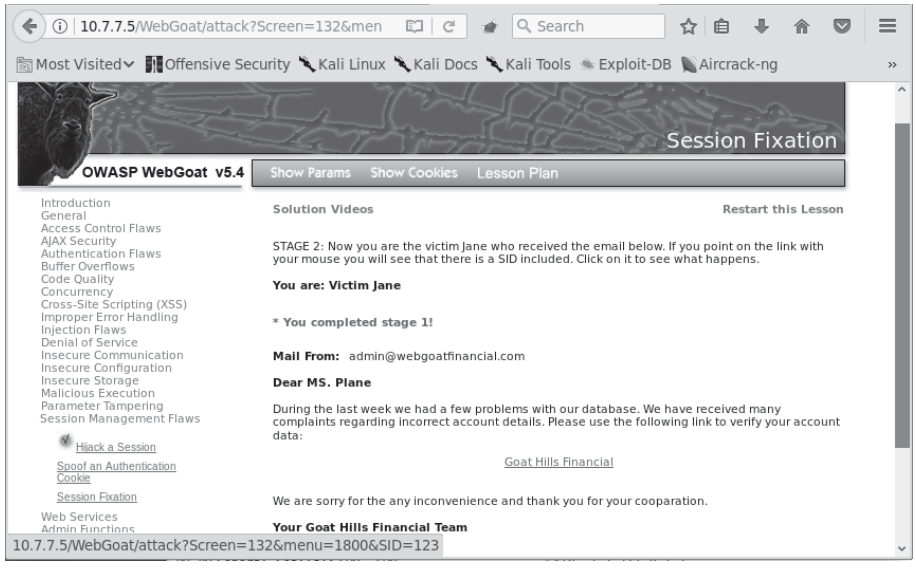
W aplikacji WebGoat znajdziemy dosyć uproszczoną, ale bardzo obrazową demonstrację tej luki (z menu aplikacji *WebGoat* wybierz opcję *Session Management Flaws/Session Fixation* (luki w zarządzaniu sesjami/Session Fixation)). Wykorzystamy tę opcję, aby zilustrować, w jaki sposób można przeprowadzić taki atak.

1. W pierwszym kroku wcielasz się w rolę napastnika. Musisz utworzyć wiadomość e-mail zawierającą dowolnie wybraną wartość identyfikatora sesji (np. `SID=123`) w linku Goat Hills, który wyślesz do ofiary:



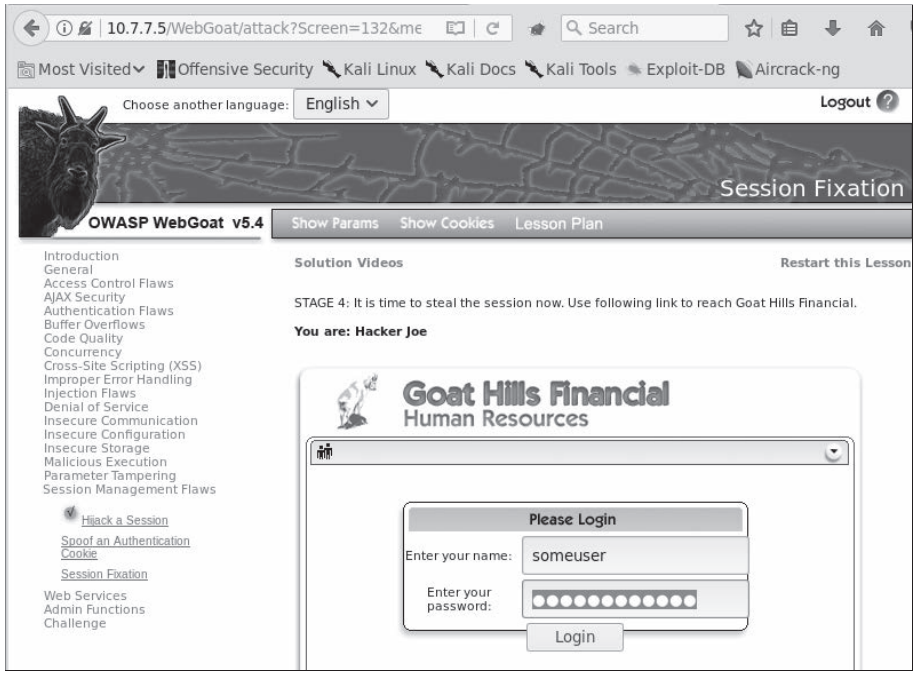
   Napastnik odkrył, że witryna *Goat Hills Financial* używa parametru `GET` do definiowania identyfikatorów sesji, i wysłał przygotowaną wyżej wiadomość phishingową do klienta tej instytucji.

2. Na tym etapie ćwiczenia działasz jako ofiara, która otrzymuje wiadomość e-mail od napastnika:

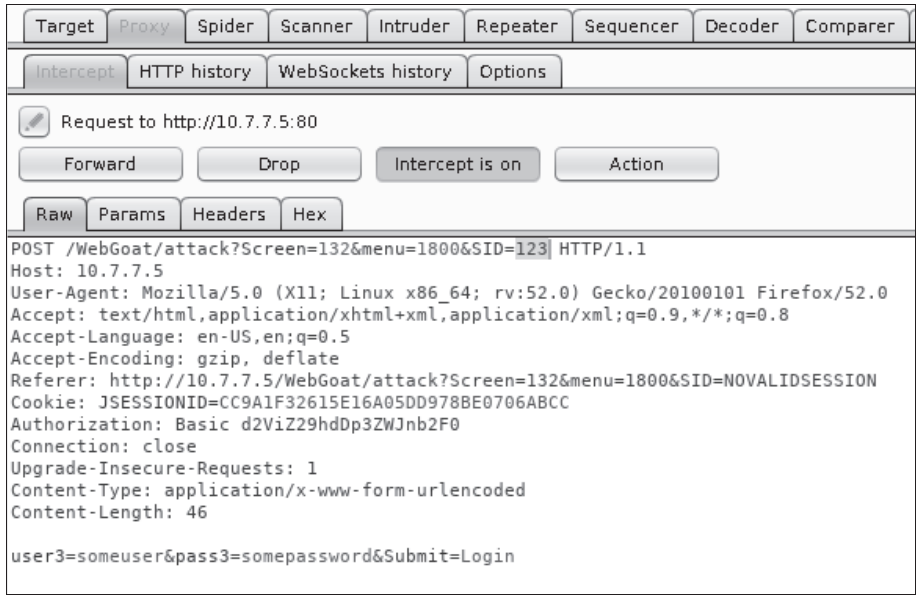


Ponieważ wiadomość e-mail nadeszła z adresu *admin@webgoatfinancial.com*, istnieje duże prawdopodobieństwo, że „ofiara" weźmie go za dobrą monetę, postąpi zgodnie z sugestią „admina" i kliknie załączony w wiadomości link, co spowoduje przejście do strony logowania i zalogowanie użytkownika. W tym momencie zostanie utworzona ważna sesja, która używa parametru wysłanego przez atakującego.

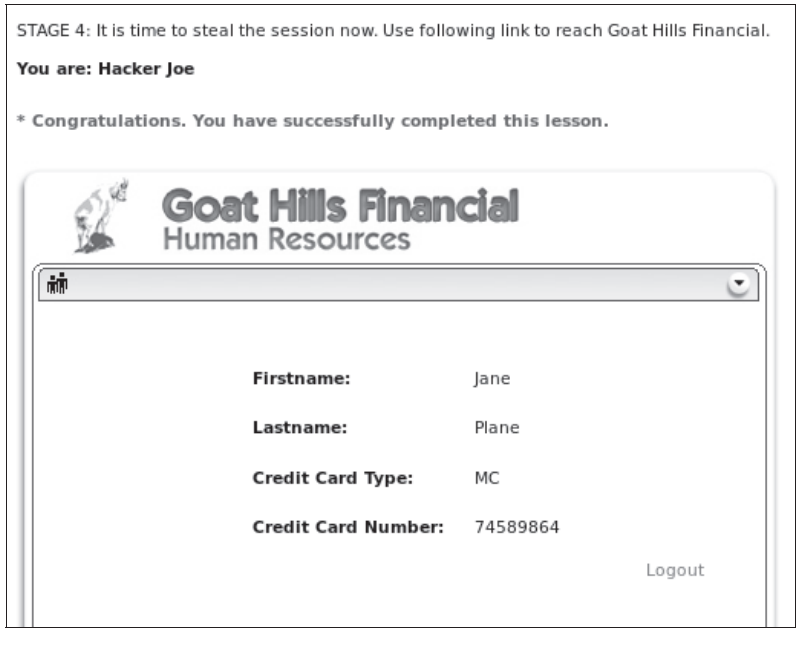
3. Następny etap ataku wymaga, aby napastnik zalogował się do tej samej witryny co ofiara:



Przechwytujemy żądanie za pomocą Burp Proxy i edytujemy je tak, aby uwzględnić parametr `SID`, którego ofiara użyła do zalogowania się:



4. Po wysłaniu zmodyfikowanego żądania napastnik otrzymuje dostęp do profilu ofiary:



W tym przykładzie zaprezentowaliśmy dwa główne błędy w procesie zarządzania identyfikatorami sesji:

1. Po pierwsze, identyfikatory sesji są generowane na podstawie informacji podanych przez użytkownika, co ułatwia intruzowi zidentyfikowanie prawidłowych wartości i powiązanie ich z istniejącymi kontami użytkowników.
2. Po drugie, identyfikator nie zmienia się po uruchomieniu uwierzytelnionej sesji (kiedy ofiara zaloguje się do aplikacji), dlatego mówimy, że jest to atak z wykorzystaniem spreparowanych stałych identyfikatorów sesji (ang. *Session Fixation*), ponieważ napastnik może dowolnie ustawić wartość identyfikatora sesji dla ofiary, a następnie wykorzystać tę samą wartość do przejęcia sesji uwierzytelnionej ofiary.

Zapobieganie atakom na uwierzytelnianie i sesje

Uwierzytelnianie w aplikacjach internetowych jest trudnym zagadnieniem, a co gorsza, do tej pory nie znaleziono uniwersalnego rozwiązania. Z tego powodu zapobieganie lukom w tym obszarze aplikacji jest w dużej mierze zależne od umiejętności deweloperów aplikacji, którzy muszą znaleźć równowagę pomiędzy użytecznością a bezpieczeństwem zgodnie z przeznaczeniem aplikacji i profilem użytkowników, z którymi będą mieli do czynienia.

To samo można powiedzieć nawet o całym zarządzaniu sesją, ponieważ obecnie używane metody nadal stanowią tylko obejście niedociągnięć protokołu HTTP. Prawdopodobnie wraz z pojawieniem się HTML5 i WebSockets lub podobnych technologii będziemy mieć lepsze alternatywy do pracy w przyszłości.

Nie zmienia to jednak w niczym faktu, że możliwe jest zdefiniowanie ogólnych wytycznych dotyczących zarówno uwierzytelniania, jak i zarządzania sesjami, które pomogłyby programistom podnieść poprzeczkę zabezpieczeń i utrudnić napastnikom życie, a których pentesterzy mogą użyć jako punktu odniesienia przy poszukiwaniu usterek i wydawaniu zaleceń klientom.

Wytyczne dotyczące uwierzytelniania

Poniżej zamieszczamy listę wskazówek dotyczących uwierzytelniania:

- Nazwy kont lub identyfikatory użytkowników muszą być unikatowe dla każdego użytkownika i nie powinny być zależne od wielkości liter (nazwy `użytkownik` oraz `UżytkowniK` odnoszą się do jednego i tego samego konta).
- Aplikacja powinna wymuszać tworzenie odpowiednio silnych haseł oraz uniemożliwiać używanie następujących ciągów znaków jako haseł:
  - Nazwa użytkownika jako hasło.
  - Krótkie hasła (czyli mniej niż osiem znaków).

  - Hasła pisane jedną wielkością liter (czyli tylko małymi lub tylko wielkimi literami).
  - Hasła składające się wyłącznie ze znaków z jednego zestawu (np. wyłącznie liczby albo wyłącznie litery) oraz hasła bez znaków specjalnych.
  - Sekwencje kolejnych liczb (np. `123456`, `9876543210`).
  - Imiona, nazwiska bądź tytuły znanych gwiazd, programów telewizyjnych, filmów lub fikcyjnych postaci (np.: `Superman`, `Batman`, `Star Wars`).
  - Hasła znajdujące się w publicznych słownikach najczęściej używanych haseł (np. 25 najczęściej używanych haseł).
- Do przesyłania danych związanych z uwierzytelnianiem aplikacja powinna zawsze używać bezpiecznych protokołów, takich jak TLS.
- W komunikatach o błędach lub kodach odpowiedzi aplikacja nie powinna nigdy ujawniać informacji o istnieniu lub ważności konta użytkownika (np. nie powinna odpowiadać kodem `404`, gdy użytkownik nie zostanie znaleziony).
- Aby uniknąć ataków typu brute force, aplikacja powinna posiadać zaimplementowaną tymczasową blokadę konta aktywowaną po pewnej liczbie nieudanych prób logowania; np. pięć prób logowania to dobrze dobrana liczba — inaczej mówiąc, konto użytkownika, który pięć razy z rzędu poda niepoprawne hasło, powinno zostać zablokowane na pewien czas, powiedzmy 20 lub 30 minut.
- Jeżeli funkcja resetowania hasła jest zaimplementowana, aplikacja powinna zapytać o nazwę użytkownika oraz odpowiedź na dodatkowe pytanie ochronne (o ile jest dostępne). Następnie powinna przesyłać jednorazowy link resetujący na adres e-mail lub telefon komórkowy użytkownika, które zostały podane podczas tworzenia konta. Link resetujący hasło powinien tracić ważność po zresetowaniu hasła przez użytkownika lub po upływie określonego czasu, np. kilku godzin, nawet jeżeli użytkownik w tym czasie z niego nie skorzysta.
- Podczas implementowania usługi MFA należy korzystać z zewnętrznych i sprawdzonych frameworków, takich jak Google Authenticator czy Authy, jeżeli używasz aplikacji mobilnych, lub RSA czy urządzeń Gemalto, jeżeli wymagany jest fizyczny token lub karta inteligentna.
- Należy unikać implementowania w aplikacji niestandardowych lub własnych algorytmów kryptografii i generowania liczb losowych; korzystaj ze standardowych algorytmów z dobrze znanych bibliotek i frameworków.
- W przypadku wykonywania szczególnie wrażliwych zadań, takich jak zmiany uprawnień dla użytkowników, usuwanie danych poufnych lub zmiana globalnych ustawień konfiguracji, aplikacja powinna prosić o ponowne uwierzytelnienie.

Na stronie projektu OWASP znajdziesz krótki przewodnik po sprawdzonych metodach wdrażania uwierzytelniania w aplikacjach internetowych: *https://www.owasp.org/index.php/Authentication_Cheat_Sheet*.

Wskazówki dotyczące zarządzania sesjami

Poniżej zamieszczamy listę wskazówek dotyczących zarządzania sesjami:

- Bez względu na zastosowany mechanizm uwierzytelniania zawsze wdrażaj zarządzanie sesjami i sprawdzaj poprawność sesji na każdej stronie i dla każdego żądania.
- Używaj długich, losowych i unikatowych identyfikatorów sesji. Korzystaj z mechanizmów już wdrożonych w najważniejszych językach programowania, takich jak ASP.NET, PHP i J2EE.
- Generuj nowe identyfikatory sesji dla użytkowników podczas logowania i wylogowywania. Trwale unieważniaj wykorzystane identyfikatory.
- Unieważniaj sesje i staraj się wylogowywać użytkowników po rozsądnym czasie bezczynności, np. po 15 czy 20 minutach. Staraj się zapewnić odpowiednią równowagę pomiędzy bezpieczeństwem a użytecznością.
- Zawsze dawaj użytkownikowi jawną możliwość wylogowania się; najlepiej za pomocą przycisku lub opcji wylogowania.
- Podczas korzystania z plików cookie sesji upewnij się, że ustawiono wszystkie niezbędne flagi bezpieczeństwa:
  - Atrybut `Secure` służy do zapobiegania używaniu ciasteczka sesji w niezaszyfrowanej komunikacji.
  - Atrybut `HttpOnly` służy do blokowania dostępu do wartości ciasteczka za pośrednictwem języków skryptowych. Zmniejsza to podatność na ataki typu ***Cross-site scripting*** (**XSS**).
  - Używaj nietrwałych plików cookie sesji, bez atrybutów `Expires` lub `Max-Age`.
  - Ogranicz atrybut `Path` do katalogu głównego serwera (/) lub do określonego katalogu, w którym hostowana jest aplikacja.
  - Atrybut `SameSite` jest obecnie obsługiwany tylko w przeglądarkach Chrome i Opera. Zapewnia dodatkową ochronę przed wyciekami informacji i atakami typu ***Cross-site request forgery*** (**CSRF**), zapobiegając wysyłaniu plików cookie na serwer przez zewnętrzne witryny.
- Połącz identyfikator sesji z rolą i uprawnieniami użytkownika i używaj go do weryfikacji uwierzytelnienia przy każdym żądaniu.

Więcej szczegółowych informacji na ten temat zawiera dokument *Session Management Chect Sheet*, który znajdziesz na stronie projektu OWASP pod adresem: *https://www.owasp.org/index.php/Session_Management_Cheat_Sheet*.

Podsumowanie

W tym rozdziale dokonaliśmy przeglądu różnych sposobów, w jakie aplikacje internetowe przeprowadzają uwierzytelnianie użytkownika, aby ograniczyć dostęp do zasobów uprzywilejowanych lub poufnych informacji, oraz sprawdzaliśmy, jak utrzymywana jest sesja, ponieważ protokół HTTP nie ma wbudowanych żadnych mechanizmów zarządzania sesjami. We współczesnych aplikacjach najczęściej stosowanymi podejściami do tego zagadnienia są uwierzytelnianie oparte na formularzach oraz identyfikatory sesji wysyłane w plikach cookie.

Przeanalizowaliśmy również najczęściej występujące błędy i luki w implementacji mechanizmów uwierzytelniania i zarządzania sesjami oraz omawialiśmy sposoby, w jakie napastnicy mogą je wykorzystywać za pomocą wbudowanych narzędzi przeglądarki lub za pomocą innych narzędzi, dostępnych w systemie Kali Linux, takich jak Burp Suite, OWASP ZAP i THC Hydra.

W ostatniej sekcji omówiliśmy najlepsze praktyki, które mogą łagodzić błędy uwierzytelniania i zarządzania sesjami, wymagając uwierzytelniania dla wszystkich uprzywilejowanych komponentów aplikacji, wykorzystując złożone losowe identyfikatory sesji i wymuszając silną politykę haseł. To tylko niektóre z technik zapobiegania takim wadom i łagodzenia ich.

W następnym rozdziale omówimy najczęściej występujące rodzaje podatności i luk w zabezpieczeniach pozwalające na przeprowadzanie ataków ze wstrzykiwaniem kodu, sposoby wykrywania i wykorzystywania ich w testach penetracyjnych, a także przedstawimy sposoby naprawiania aplikacji i skutecznego zapobiegania takim atakom.

Skorowidz

A

J

K

L

Ł

M

N

O

P

R

S

T

U

V

W

X

Z

Ż

Kali Linux. Sprawdź, czy *na pewno* jesteś bezpieczny!

Powtarzające się przypadki masowego naruszania bezpieczeństwa informacji, nielegalnego gromadzenia danych wrażliwych o użytkownikach czy nieuprawnionego wykorzystywania systemów do dezinformacji zmuszają organizacje do uznania cyberbezpieczeństwa za swój priorytet. Z drugiej strony coraz krótszy cykl rozwojowy aplikacji i usług sieciowych, a także rosnąca liczba technologii o coraz większej złożoności sprawiają, że projektant aplikacji nie jest w stanie w pełni przetestować tworzonego oprogramowania pod kątem bezpieczeństwa. W tych warunkach rośnie rola specjalistów w dziedzinie testów bezpieczeństwa. Wśród narzędzi dla profesjonalnych pentesterów słynny system Kali Linux zasługuje na szczególną uwagę.

To drugie, uzupełnione i zaktualizowane wydanie znakomitego przewodnika dla testerów aplikacji. Wyjaśniono w nim koncepcję etycznego hakingu i testów penetracyjnych oraz omówiono narzędzia dostępne w systemie Kali Linux. Przedstawiono mechanikę klasycznych ataków, również tych z wstrzykiwaniem kodu SQL i poleceń. Bardzo dokładnie przeanalizowano kwestie kontroli danych wejściowych. Ważną część książki stanowi omówienie najnowszych problemów z warstwami kryptograficznymi w komunikacji sieciowej, a także prezentacja zautomatyzowanych fuzzerów do wykrywania potencjalnych luk i podatności w aplikacjach internetowych. Oprócz opisów poszczególnych technik ataków znalazły się tu również wskazówki dotyczące zapobiegania tym atakom i łagodzenia ich skutków.

Najciekawsze zagadnienia:

- metodyki testów penetracyjnych i podstawowe pojęcia z tym związane
- klasyczne metody ataków: wstrzykiwanie kodu, ataki XSS i CSRF
- wykorzystywanie podatności kryptograficznych
- ataki file inclusion, HTTP Parameter Pollution (HPP) oraz wycieki informacji
- prowadzenie ataku ze strony klienta
- zautomatyzowane skanery i fuzzery

Gilberto Najera-Gutierrez jest doświadczonym pentesterem. Uzyskał tak ważne certyfikaty jak OSCP, ECSA oraz GXPN. Od prawie 20 lat pasjonuje się cyberbezpieczeństwem. Przeprowadzał testy penetracyjne sieci i aplikacji dla największych korporacji, agencji rządowych i instytucji finansowych w Meksyku i Australii.

Juned Ahmed Ansari jest ekspertem w dziedzinie cyberbezpieczeństwa. Zajmuje się analizą zagrożeń i badaniami w zakresie bezpieczeństwa aplikacji. Posiada takie certyfikaty jak GXPN, CISSP, CCSK i CISA. Obecnie prowadzi zespół specjalistów zajmujących się testami penetracyjnymi i zagadnieniami offensive security w dużej korporacji.

ISBN 978-83-283-5123-3