Poradnik Początkującego Hackera

Przedmowa

Zapraszamy na warsztaty hackowania. Zajęcia te mają na celu przybliżyć Ci podstawowe podatności stron internetowych oraz sposoby ich wykorzystywania w atakach hakerskich. Twoim zadaniem jest znalezienie na podanych stronach tzw. flag. Możesz je rozpoznać wiedząc, że mają format: putrequest{…}. Twoim celem jest znalezienie ich wszystkich i wklejenie w zakładce **Zgłoś flagę** na stronie głównej. W dalszej części poradnika znajdziesz wskazówki, które pomogą ci w ich znalezieniu. Oczywiście musisz pamiętać, że zastosowanie podanych metod na stronach, które nie są do tego stworzone jest zabronione. Jeśli jednak spodoba ci się to co będziesz robił na warsztatach (na co oczywiście liczymy), możesz później samemu poszerzać swoją wiedzę rozwiązując CTF-y (Capture The Flag), które bez problemu znajdziesz w internecie.

Zaczynajmy!

Gdy wchodzimy na jakąś stronę widzimy, co ma nam ona do zaoferowania. Podpisane zakładki, pola do wyszukiwania, miejsce do wpisania loginu i hasła, wszystko ładnie ułożone na ekranie i współgrające ze sobą. Tak my to widzimy, jednak z perspektywy twórcy strony sprawa wygląda trochę inaczej. Musi on odpowiednio napisać **kod** w językach programowania ( m. in. HTML, CSS, PHP czy SQL), by użytkownik mógł zobaczyć dany rezultat. Jako użytkownicy możemy podejrzeć tylko część kodu napisanego przez programistę, a konkretnie te fragmenty, które są wykonywane na naszym komputerze (np. HTML i CSS). Z kolei obsługą bardziej wrażliwych danych oraz wykonywaniem instrukcji, których śledzenie przez zwykłego użytkownika jest niepożądane, zajmuje się serwer, tak więc kod, który on wykonuje (np. kod w PHP oraz SQL) nie jest jawny. Potencjalnemu atakującemu zostaje więc sprawdzenie, czy może jakiś gapiowski programista **nie zostawił czegoś interesującego w kodzie jawnym**.

Żeby zabezpieczyć wrażliwe dane, programista powinien je **zaszyfrować**. Istnieje wiele rodzajów szyfrów, jedne są bardziej skuteczne, drugie mniej. W dużym skrócie polegają one na zastosowaniu algorytmu, który za pomocą klucza tworzy zaszyfrowany tekst (szyfrogram). Dane zaszyfrowane są tajne, tak więc tylko uprawnione do tego jednostki powinny znać sposób, aby pozyskać je z szyfrogramu. Oczywiście istnieją sposoby na odszyfrowanie tekstu bez posiadania klucza, jednak w większości przypadków wymagają one dużo czasu. W informatyce istnieje również pojęcie **kodowania** danych. Polega ono na zmianie formatu danych, np. w celu łatwiejszego przesłania tekstu przez sieć komputerową. Dekodowanie, czyli proces odwrotny, jest bardzo szybkie, ponieważ celem kodowania nie jest utajanie danych. Popularnymi formatami kodowania są np. **UTF-8 i Base64**, bez problemu znajdziesz w sieci strony, które zakodują i odkodują wpisany przez ciebie tekst. Zapewne masz teraz świadomość, jak ważne jest rozróżnianie pojęć szyfrowania i kodowania. Ciekawe, czy nasz programista też się do tego stosuje…

*Przydatne linki:*[*https://cafewebmaster.com/*](https://cafewebmaster.com/) *-* w zakładce **tools** znajdziesz potrzebne narzędzia

Lubisz **ciasteczka**? Nie znam osoby która by nie lubiła. Pewnie wiele razy przeglądając różne strony internetowe natknąłeś się na informacje o wykorzystywaniu ciasteczek (**cookies**) przez tę właśnie stronę. Czy zastanawiałeś się kiedyś do czego one służą? W dużym skrócie są to dane zapisywane na komputer użytkownika przez przeglądarkę. Przykładowo, jednym z ich zastosowań jest śledzenie przez serwis internetowy, czy dany użytkownik jest zalogowany, czy nie, podczas gdy ten przechodzi z jednej strony na drugą, a innym może być na przykład sprawdzanie, czy użytkownik ma uprawnienia administratora. **Nazwy i wartości**, które mają w sobie ciasteczka, **można podejrzeć**. Możliwe jest nawet **zmienienie wartości ciasteczka** przez użytkownika w przeglądarce. Jeśli nie wiesz jak to zrobić, poszukaj pomocy w internecie. Umiejętność znajdowania potrzebnych nam informacji w sieci jest jedną z ważniejszych, by stać się dobrymi informatykami. *Aby zobaczyć skutki zmiany wartości ciasteczka, musisz odświeżyć stronę!*

Komunikacja komputerów poprzez sieci WWW odbywa się poprzez przesyłanie żądań przez przeglądarkę do serwera. Najczęściej odbywa się to automatycznie, np. po kliknięciu odpowiedniej zakładki na stronie przeglądarka generuje i wysyła odpowiednie zapytanie, na które odpowiada serwer, przesyłając zażądaną stronę. Istnieją jednak metody pozwalające użytkownikowi zmodyfikować wysyłane zapytanie. Jedną z nich jest zastosowanie narzędzia ***Burp Suite.*** Burp Suite to jeden z ważniejszych programów wykorzystywanych do testowania aplikacji internetowych pod kątem bezpieczeństwa. Przedstawię ci tylko jedno z wielu zastosowań tego programu, jakim jest możliwość modyfikowania zapytań.

1. Włączamy Burp Suite. Na początku wyświetli nam się okno wyboru projektu. Jako że korzystamy z wersji darmowej, nie możemy zapisywać i otwierać istniejących projektów, dlatego klikamy przycisk **next.** W kolejnym oknie wybieramy **Start Burp**.
2. Po załadowaniu projektu wybieramy z górnej zakładki **Proxy**. I tutaj zaczyna się zabawa:

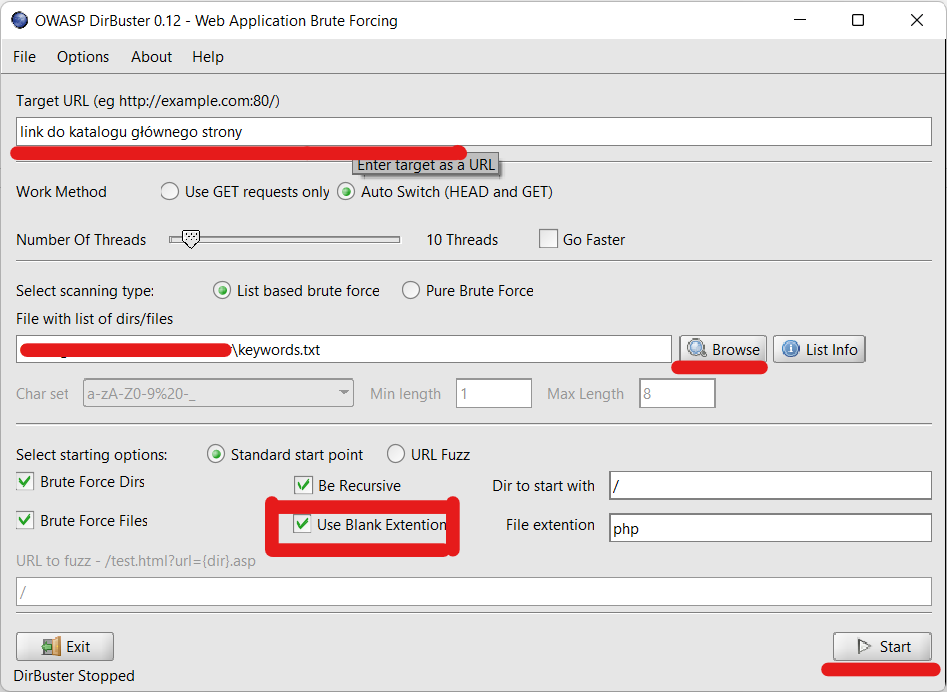
Żeby zmodyfikować przesyłana żądanie, musimy je przechwycić. Robimy to klikając w Burpie przycisk **Intercept is off**, a następnie klikając w naszej przeglądarce przycisk, który wywołuje zapytanie, które chcemy zmodyfikować. Przykładowo: mamy stronę logowania do serwisu. Żeby przechwycić żądanie, musimy włączyć opcję **intercept** zanim naciśniemy przycisk **Zaloguj** na stronie. Jeśli to zrobimy poprawnie, **przeglądarka** nie przejdzie na kolejną stronę, a w **Burpie** zobaczymy pole z naszym zapytaniem. Możemy je modyfikować wedle naszego uznania, a następnie wysłać klikając przycisk **Forward**. W tym momencie powinniśmy zobaczyć odpowiedź w naszej przeglądarce. Pamiętaj by po przesłaniu zapytania wyłączyć opcję **Intercept**, by przeglądarka wróciła do normalnego działania.

Pozostaje tylko pytanie: po co modyfikować zapytanie? Czy przeglądarka nie zrobi tego lepiej za nas? Co nam to daje? Odpowiedź jest prosta: Twórca strony może chcieć blokować nam możliwość wpisania jakiegoś typu informacji, jednak sprawdzenie tej blokady może odbywać się zarówno przed wysłaniem zapytania, jak i po. W pierwszym przypadku, chcąc tę blokadę ominąć, możemy wysłać dane, które przejdą przez zabezpieczenie, a następnie zmienić je na takie, które tych zabezpieczeń by nie przeszły, zaraz przed wysłaniem do serwera.

Dobrze zabezpieczona strona nie powinna pozwalać użytkownikowi na dostęp pewnych zasobów. Jak pewnie wiesz, zmieniając strony zmienia się również adres URL. Czy potrafisz dostrzec jakąś zależność, którą kierują te zmiany? Jeśli tak, to może zmieniając coś w adresie znajdziesz coś ukrytego.

Innym sposobem na odnalezienie ukrytych zasobów na stronie jest tzw. **directory traversal** (path traversal). Możemy go wykorzystać na przykład wtedy, gdy widzimy, że strona otwiera nam plik, np. tekst albo zdjęcie (Możemy to poznać po tym, że w źródle strony nie zobaczymy żadnego kodu). Podatność ta polega na zmianie ścieżki pliku, który ma być otwarty. Zmiany możemy dokonać dopisując na końcu adresu URL ?filename=<nazwa\_pliku>. Pojawia się jednak problem – nie wiemy jakie nazwy mają ukryte pliki. Ciekawe co się stanie, gdy przejdziemy do katalogu, w którym znajduje się nasz plik. Możemy to sprawdzić, wpisując w miejsce nazwy pliku ‘..’. ‘..’ oznacza przejście o jeden katalog w górę w drzewie katalogów. Byłoby bardzo lekkomyślne ze strony programisty, gdyby zaprogramował stronę tak, by dawała nam informację o nazwach plików w katalogu, w którym jesteśmy. W ten sposób moglibyśmy łatwo znaleźć flagę oraz sprawdzić, czy nie zostały ukryte jakieś dodatkowe zadania…

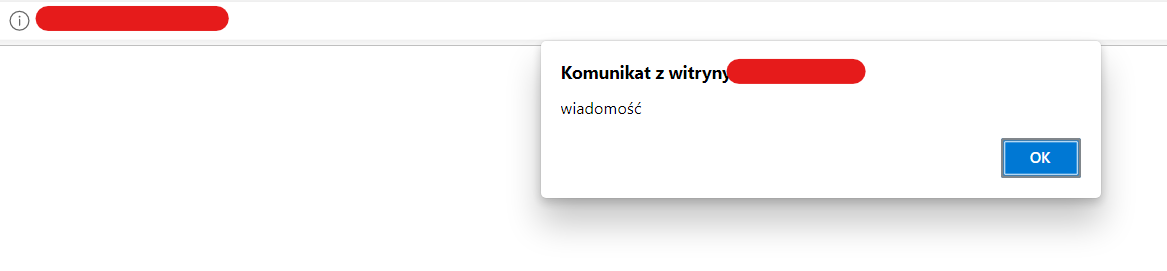
Musimy jednak mieć na uwadze, że ręczne wyszukiwanie zasobów może być bardzo czasochłonne. Na szczęście istnieją narzędzia, które automatycznie wyszukują istniejące katalogi na stronie. Jednym z takich narzędzi jest program **DirBuster.** Jego zasada działania jest prosta: podajemy link do strony, od której program ma rozpocząć poszukiwanie (najlepiej zacząć od strony głównej). Następnie musimy wybrać plik tekstowy, w którym znajdują się nazwy katalogów, które program ma sprawdzić. Przygotowaliśmy do tego gotowy plik o nazwie keywords.txt (Będzie on znajdował się w katalogu, który otworzy się po kliknięciu przycisku **Browse**). Następnie zaznaczamy opcję **Use Blank Extention** i klikamy przycisk start. Program powinien wyświetlić nam znalezione katalogi. Większość z nich już znamy, pytanie tylko, czy nie znalazło się coś wcześniej przed nami ukrytego. Poniżej screenshot z zaznaczonymi ważnymi dla nas miejscami w programie.



**Injection (Wstrzyknięcie)** to jedna z groźniejszych i najbardziej popularnych podatności aplikacji internetowych. Występuje ona, gdy użytkownik ma możliwość wpisywać dane np. do logowania albo wyszukania na stronie, jednak zamiast zwykłego tekstu wpisze kod w odpowiednim języku programowania. Niezabezpieczona strona wykona wtedy wpisany kod. Na początek przyjrzymy się metodzie wstrzykiwania zwanej XSS (Cross Site Scripting). Podatność strony na XSS można rozpoznać, gdy strona ta wyświetla podanej wcześniej przez nas dane. Przykładowo, możemy wstrzyknąć kod HTML np. <h1> napis </h1> i zobaczyć co zwróci nam strona. Jeśli tekst na wyjściu różni się od tego, który został wyświetlony bez podawania tagów HTML-a, znaleźliśmy podatność na XSS. Dobra wiadomość dla nas, jednak zmiana formatu napisu nie wydaje się zbyt groźna ani ciekawa. Co jeszcze możemy zrobić, żeby stało się coś więcej? Przecież naszym zadaniem jest niezmiennie wyciągnięcie flagi ze strony. Z pomocą przyjdą nam skrypty JavaScript i funkcja alert(). Funkcja ta wyświetla wiadomość wpisaną między nawiasami w nowym oknie wyświetlonym przez przeglądarkę (przykład poniżej). Wiadomość musimy umieścić pomiędzy pojedynczymi apostrofami. Może za pomocą odpowiedniej wiadomości możemy poprosić o coś stronę?

Sposób wstrzyknięcia: <script> alert(‘wiadomość’) <\script>

Skutek wykonania skryptu:



Pamiętaj, że tak jak w jednym z poprzednich zadań musisz ominąć pewne zabezpieczenia.