

Manipulowanie parametrami wejściowymi jako skuteczny sposób ataku na aplikacje internetowe

Przemek Sobstel (<http://sobstel.org>). *SegfaultLabs*, maj 2006.

Działanie każdej aplikacji internetowej oparte jest na parametrach przesyłanych pomiędzy przeglądarką użytkownika a stroną WWW. Jeśli parametry te nie są odpowiednio weryfikowane lub przechowywane są w nich dane kluczowe dla działania aplikacji, to atakujący zyskuje możliwość przeprowadzenia skutecznego ataku poprzez modyfikację tych parametrów.

Manipulowanie parametrami może być przeprowadzone przy użyciu adresu URL, ciasteczek (*cookies*), pól formularza, a także nagłówków HTTP.

Manipulowanie łańcuchem żądania

Łańcuch żądania (adres URL) powszechnie używany jest do wskazania konkretnej strony WWW. Zagrożenie polega na tym, że atakujący może samodzielnie zmodyfikować składniki adresu URL i w ten sposób uzyskać dostęp do poufnych danych albo je nieprawidłowo zmodyfikować [SzWi06, s.64]. Włamywacz może także doprowadzić do wyświetlenia w oknie przeglądarki błędów wynikających z błędu języka skryptowego czy też nieprawidłowego zapytania kierowanego do bazy, co może mu pozwolić na zebranie informacji potrzebnych do przeprowadzania włamania [Grze04, s.61]. Poza tym, manipulowanie łańcuchem żądania może być z powodzeniem wykorzystane przy dokonywaniu wielu innych rodzajów ataku na aplikacje internetowe, od Cross Site Scripting i iniekcji kodu SQL do prób ujawnienia kodu źródłowego i innych poufnych danych.

Typowy scenariusz ataku poprzez manipulowanie adresem URL wygląda następująco [SzWi06, s.64] :

- (1) Uruchamianie witryny i zapisywanie wysyłanych do serwera kolejnych adresów URL.
- (2) Analiza budowy witryny na podstawie zebranych adresów. W szczególności brane pod uwagę są nazwy i przeznaczenie formularzy oraz poszczególnych parametrów.
- (3) Sprawdzanie reakcji aplikacji na błędne dane poprzez eksperymentowanie z najróżniejszymi, często błędnymi danymi. Wszelkie nietypowe reakcje, w szczególności komunikaty błędów, pozwalają sporo dowiedzieć się o budowie i działaniu witryny.

Dodatkowy problem stanowi fakt, że ten sam adres URL może być przedstawiony pod wieloma graficznie różnymi, a przy tym semantycznie równoważnymi, postaciami. W celu przygotowania odpowiednio spreparowanego ciągu znaków najczęściej wykorzystuje się kodowanie ASCII i kodowanie UNICODE.

Kodowanie ASCII pozwala na zapisanie dowolnego znaku w adresie URL w postaci dwucyfrowej liczby poprzedzonej znakiem procenta (%). Liczbą tą jest szesnastkowy odpowiednik kodu danego znaku w tabeli zestawu znaków ASCII. Przykładowo adresy `http://www%2Efirma%2Ecom%2Fdane` i `http://www.firma.com/dane` wskazują na ten sam folder, tyle że w pierwszym przypadku znak kropki (.) został zastąpiony kodem %2E, a znak ukośnika (/) kodem %2F [Szwi06, s.79]. Ponadto możliwe jest zapisanie jednego znaku wielokrotnie większą liczbą kodów, np. znak ukośnika (/), może być przedstawiony w następujących równoważnych postaciach : %252F, %2%66 %25%02%66 [Szwi06, s.79; Ollm02], a to tylko kilka przypadków. W praktyce ilość możliwych kombinacji jest bardzo wysoka.

Podobnie do kodowania ASCII, także kodowanie UNICODE pozwala atakującemu na zapisanie tych samych adresów na wiele różnych sposobów, np. w najbardziej rozpowszechnionym obecnie standardzie UTF-8 adres URL może wyglądać w sposób następujący : `http://www.firma.com?var=..%C0%AF../bin/l%200a1` [Szwi06, s.79].

Możliwość przedstawienia adresów URL pod wieloma różnymi postaciami powoduje, że filtry i zapory oparte na sygnaturach stają się bezradne i nie powinny być one traktowane jako skuteczny środek ochrony przed atakami na witryny internetowe. Problem ten jeszcze pogorsza fakt, że modyfikacja składników adresu URL jest czynnością bardzo łatwą. W rezultacie atak ten jest jednym z najprostszych do przeprowadzenia.

Manipulowanie polami formularza

Dane podawane w formularzach webowych zazwyczaj przesyłane są metodą POST i nie są umieszczane w adresie URL. Nie znaczy to jednak, że są odporne na próby modyfikacji pól i ich wartości. Atakujący może bowiem pobrać kod HTML formularza, a następnie zmodyfikować go i wysłać żądanie z własnego komputera albo wstrzyknąć odpowiednio przygotowany kod HTML formularza bezpośrednio na stronę. W praktyce manipulowanie polami formularza opiera się na tym samych zasadach co manipulowanie łańcuchem żądania. W obu przypadkach napastnik najpierw analizuje budowę atakowanej strony, a następnie wysyła spreparowane żądania HTTP w celu poznania lub wymuszenia pożądanego zachowania witryny.

Atakujący może przerobić formularz HTML według własnego uznania, np. usunąć ograniczenie ilości wprowadzanych znaków (atrybut `maxlength`), ominąć kontrolę poprawności danych po stronie klienta, zmienić wartości ukrytych pól (typ pola `hidden`) lub zmodyfikować typy pól formularza [Shif05]. Na szczególną uwagę zasługują pola ukryte, które stanowią wygodny sposób na przekazywanie różnych danych pomiędzy kolejnymi podstronami. Zagrożenie pojawia

się wtedy, gdy są to dane kluczowe, takie jak np. cena produktu czy poziom uprawnień. Konsekwencje modyfikacji tych parametrów przez napastnika są oczywiste. Tego typu błędy znajdują się nawet w dużych, komercyjnych aplikacjach WWW [SzWi06, s.73].

Manipulowanie nagłówkami żądania HTTP

Nagłówki żądania HTTP służą do przekazywania serwerowi przez klienta informacji o sobie i swojej konfiguracji oraz preferowanych formatach dokumentów [Wong00, s.53]. Problem polega na tym, że analogicznie do wszystkich innych informacji pochodzących od klienta, nagłówki HTTP mogą być zmodyfikowane przez napastnika [BaKl02, s.8]. Co prawda przeglądarki internetowe z reguły nie umożliwiają zmiany wartości nagłówków, lecz atakujący może napisać własny skrypt wysyłający żądania HTTP lub skorzystać z jednego z darmowych i ogólnie dostępnych serwerów Proxy, które umożliwiają modyfikacje dowolnych danych wysyłanych z przeglądarki [OWASP02, s.46]. Zagrożenie jest realne, tymczasem w języku PHP wartości pól nagłówków HTTP, obok wielu innych parametrów, są umieszczone w tablicy globalnej `$_SERVER`, której nazwa może sugerować o tym, że pochodzą one z bezpiecznego źródła. Niestety poprawność zapisanych w tych polach nagłówków nie jest automatycznie sprawdzana [SzWi06, s.71], przez co powinny być traktowane jako potencjalne zagrożenie. Stanowi to problem szczególnie jeśli nagłówki te są wykorzystywane dla wzmocnienia bezpieczeństwa. Przykładowo nagłówek `Referer` może być użyty do sprawdzania czy żądanie zostało wysłane z właściwej podstrony. Celem jest przeciwdziałanie zapisywaniu witryny na dysk, a następnie modyfikacji formularza i wysłania go z własnego komputera przez atakującego [OWASP02, s.46]. Jednak fakt, że wartość nagłówka, na którym opiera się to zabezpieczenie, może być łatwo zmodyfikowane, sprawia że osiągnięto tutaj efekt odwrotny od zamierzonego – umożliwiono przeprowadzenie ataku tam, gdzie ze względu na zaimplementowane funkcje ochronne, nikt się tego nie będzie spodziewał.

Manipulowanie wartościami ciasteczek

Mechanizm ciasteczek (cookies) opiera swoje działanie o niewielkie pliki tekstowe, składowane na komputerze użytkownika [Grze04, s.61]. Poprawność zapisanych w ciasteczkach danych nie jest automatycznie sprawdzana – żadne sumy czy kody kontrolne nie są zapisywane razem z danymi, dlatego pozwala nie tylko odczytać, ale również zmodyfikować plik ciasteczka, zmieniając wartości parametrów czy dopisując nowe parametry i ich wartości [SzWi06, s.70]. Podobnie jak wspomniane w niniejszym punkcie adresy URL, pola formularzy oraz nagłówki HTTP, jako dane pochodzące z zewnątrz, także cookies nie powinny być używane w aplikacji bez wcześniejszej filtracji i walidacji.

Literatura

- [BaKl02] Bar-Gad I., Klein A.: *Developing Secure Web Applications*, Sanctum Inc. 2002.
- [Grze04] Grzesiak P.: *Bezpieczeństwo stron internetowych*, Internet nr 1/2004, s.59-61.
- [Ollm02] Ollmann G.: *URL Encoded Attacks*, *TechnicalInfo.net* 2002.
<http://www.technicalinfo.net/papers/URLEmbeddedAttacks.html>
- [OWASP02] *A Guide to Building Secure Web Applications*, Version 1.1.1, The Open Web Application Security Projects 2002.
http://www.owasp.org/index.php/Category:OWASP_Guide_Project
- [Shif05] Shiflett C.: *Essential PHP Security*, O'Reilly 2005.
- [SzWi06] Szeliga M., Wileczek R.: *PHP5. Tworzenie bezpiecznych stron WWW*, Helion 2006.
- [Wong00] Wong C.: *HTTP. Leksykon kieszonkowy*, Helion 2000.