

Bezpieczeństwo sesji w PHP – zarys problemu

Autor : Przemek Sobstel (<http://sobstel.org>)

Rodzaje ataków na sesje użytkownika

Mechanizm zarządzania sesją jest powszechnie stosowany w celu identyfikacji i śledzenia użytkownika pomiędzy kolejnymi podstronami, w rezultacie czego użytkownik nie musi podawać swoich danych uwierzytelniających - z reguły nazwy (login) i hasła - przed obejrzeniem każdej podstrony w ramach danej witryny. Użytkownikowi przydzielany jest unikalny identyfikator, dzięki któremu aplikacja jest w stanie go rozpoznać, przydzielić odpowiedni zestaw uprawnień oraz odczytać specyficzne dla niego dane (np. wybrany język). Duża grupa zagrożeń dotyczących mechanizmu autoryzacji ma związek właśnie z tym identyfikatorem sesji. Użycie przez atakującego tego samego identyfikatora co prawowity użytkownik prowadzi bowiem do nieuprawnionego dostępu do aplikacji i wykonania operacji w jego imieniu.

Wyróżnia się kilka ataków i zagrożeń związanych z sesjami. Należą do nich przechwycenie sesji, wymuszenie sesji, ujawnienie danych sesji, podmiana całej sesji oraz podmiana danych sesji.

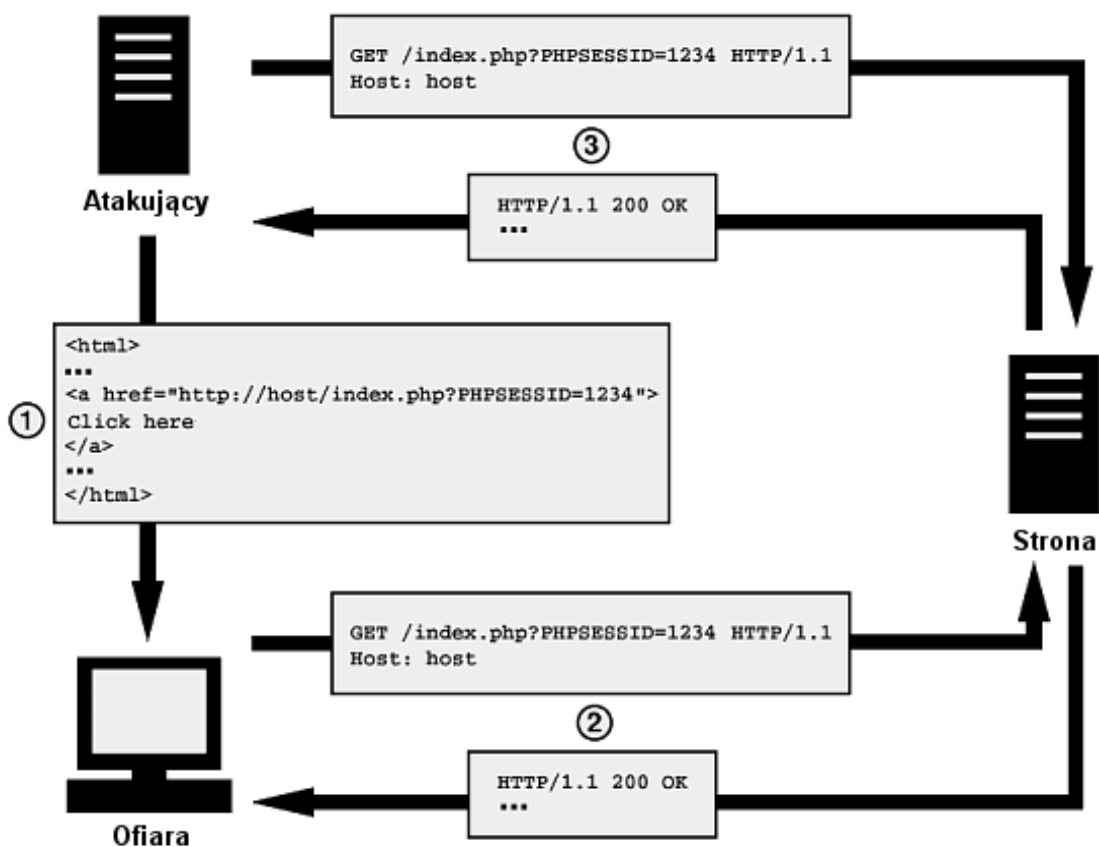
Przechwycenie sesji (*Session Hijacking*) odnosi się do tych ataków, które próbują uzyskać dostęp do istniejącej sesji użytkownika [Shif04, s.42]. Oznacza to tych użytkowników, którym identyfikator został już przydzielony. Znane są dwie podstawowe metody uzyskania identyfikatora sesji :

- (1) Domniemanie sesji – można zaliczyć tu następujące sposoby : wykorzystanie podatności aplikacji na atak *Cross Site Scripting* [Kols02, s.15], odczytanie identyfikatora sesji z nagłówka HTTP `Referer` wysyłanego do innego serwera [Ollm03] czy też odgadnięcie identyfikatora. Ostatni z wymienionych sposobów staje się możliwy, gdy identyfikator nie jest wystarczająco losową wartością. Istnieje wiele technik matematycznych, np. prognozowanie matematyczne, które mogą być zastosowane do odgadywania przewidywalnych, czasem sekwencyjnych, identyfikatorów sesji [ScSh02, s.155].
- (2) Podśluchiwanie ruchu sieciowego – napastnik, który ma możliwość monitorowania ruchu sieciowego pomiędzy atakowanym użytkownikiem a serwerem, może wykraść identyfikator sesji, ponieważ jest on przesyłany tekstem jawnym [Olse04, s.11], ale tylko wtedy, gdy nie zastosowano szyfrowanego połączenia SSL.

Kolejny rodzaj ataku, określanym terminem “wymuszenie sesji” (*Session Fixation*), różni się od opisanych wyżej technik tym, że napastnik nie skupia swojej uwagi na zdobyciu identyfikatora sesji ofiary, a raczej na skłonieniu ofiary do użycia identyfikatora określonego przez niego samego [Olse04, s.12]. Innymi słowy, użytkownik zostaje nieświadomie skłoniony do użycia sesji

zainicjowanej przez atakującego. Atak ten przebiega w trzech krokach [Kols02; Esse05] :

- (1) Ustanowienie sesji – w pierwszym kroku atakujący tworzy losowy identyfikator sesji lub rozpoczyna nową sesję na docelowym serwerze i pozyskuje z niej identyfikator; zazwyczaj musi on utrzymywać sesję przez wielokrotne wysyłanie żądań, aby uniknąć jej wygaśnięcia,
- (2) Wymuszenie sesji – atakujący przekazuje ofierze ustalony identyfikator (zob. rysunek 1, krok 1); przekazanie to może się odbyć na wiele sposobów m.in. może on wykorzystać techniki polegające na iniekcji i wykonaniu wrogiego kodu, np. *Cross Site Scripting*; może także po prostu skłonić ofiarę do uruchomienia odpowiednio spreparowanego odnośnika lub formularza HTML,
- (3) Uzyskanie dostępu – w ostatnim kroku atakującemu pozostaje tylko czekać, aż ofiara zaloguje się do aplikacji używając ustalonego identyfikatora (zob. rysunek 1, krok 2), po czym wykorzystując ten sam identyfikator atakujący może podszyc się pod ofiarę (zob. rysunek 1, krok 3).



Rysunek 1: Przebieg ataku wymuszenia sesji

Źródło : [Shif04]

Nieco innym rodzajem zagrożeń są : ujawnienie danych sesyjnych oraz podmiana całej sesji. Oba wykorzystują dość kontrowersyjną cechę języka PHP. Otóż w standardowej konfiguracji, PHP zapisuje sesje w pliku, w tym samym katalogu (w przypadku systemu operacyjnego Linux jest to katalog tymczasowy /tmp), w wyniku czego w środowiskach współdzielonych dane sesyjne zapisywane są w tym samym miejscu i z tymi samymi uprawnieniami. Atakujący może to wykorzystać do podmiany pliku sesji lub danych w nim zawartych. Testy dowodzą, że w ten sposób skonfigurowanych jest wiele kont u wielu dostawców usług hostingowych, a także serwerów uczelnianych [Mrug06, s. 75]. Jednakże problem nie dotyczy tylko środowisk współdzielonych. Ujawnienie plików sesji jest możliwe bowiem także np. przy użyciu *Directory Traversal*, natomiast do modyfikacji tych plików można wykorzystać chociażby podatność aplikacji na iniekcję poleceń systemowych.

Ostatnie z zagrożeń opisywanych w niniejszym punkcie, czyli atak polegający na podmianie danych sesji (ang. *Session Injection*) nie do końca związany jest z samym mechanizmem zarządzania sesją. O wiele bardziej dotyczy zagrożeń iniekcji złośliwego kodu oraz manipulacji parametrami wejściowymi. Podmiana danych sesji polega na nieautoryzowanym rejestrowaniu zmiennych w sesji [Jaku03]. Atak ten możliwy jest wtedy, gdy zmienne sesyjne inicjowane lub modyfikowane są na podstawie danych pochodzących od użytkownika, a weryfikacja tych danych jest niewystarczająca.

Jak pokazują wyżej omówione zagrożenia związane z poprawnym uwierzytelnianiem i identyfikacją użytkownika, kwestia ochrony tych mechanizmów nie powinna być lekceważona, tym bardziej że standardowy mechanizm sesji zaimplementowany w PHP nie daje wystarczającego poziomu bezpieczeństwa.

Tworzenie bezpiecznego mechanizmu zarządzania sesją

Jak wspomniano wcześniej mechanizm zarządzania stanem sesji umożliwia identyfikację konkretnego użytkownika pomiędzy poszczególnymi żądaniami. Z racji, że protokół HTTP jest protokołem bezstanowym, mechanizm ten musi być obsługiwany bezpośrednio przez aplikację.

Martin Fowler wyróżnia trzy wzorce stanu sesji [Fowl05, s.81] :

- Stan sesji klienta (ang. *Client Session State*) – dane sesyjne przechowywane są po stronie klienta i przesyłane są przy każdym żądaniu,
- Stan sesji serwera (ang. *Server Session State*) – dane mogą być przechowywane w pamięci serwera lub w sposób bardziej trwały. np. w lokalnym systemie plików serwera; może to być też tabela bazy danych, gdzie kluczem jest identyfikator sesji, a jedną z wartości serializowane dane sesyjne,

- Stan sesji bazy danych (ang. *Database Session State*) – dane sesyjne również przechowywane są po stronie serwera, ale najpierw dzielone na tabele i pola, a następnie zapisywane w bazie danych w podobny sposób jak dane trwałe.

Przechowywanie danych sesyjnych po stronie klienta, jeśli chodzi o bezpieczeństwo, jest zdecydowanie najgorszym wyborem. Zagrożenia związane ze stosowaniem tej metody to ryzyko nieautoryzowanego dostępu oraz sfalszowania lub utraty danych, dlatego konieczne jest szyfrowanie danych wysyłanych do przeglądarki [Schl04, s.350]. Przechowywanie danych po stronie serwera jest więc lepszą alternatywą. Wzorzec stanu sesji bazy danych przeznaczony jest raczej do rozwiązań kompleksowych o dużej złożoności i przeważnie nie znajduje zastosowania w aplikacjach webowych. Zdaniem autora niniejszej pracy wzorzec stanu sesji serwera jest najbardziej odpowiednią metodą obsługi sesji i sprawdzi się w większości sytuacji. Jest to sposób bardzo przejrzysty w implementacji.

Istotne jest, że zarówno stan sesji serwera, jak i stan sesji bazy danych, nie odbywają się bez udziału klienta. Po stronie klienta przechowywany jest bowiem identyfikator sesji, dzięki któremu mechanizm stanu sesji wie, które dane sesyjne odczytać z magazynu danych. Jak zasygnalizowano w punkcie wcześniej głównym źródłem zagrożeń związanych z sesją jest właśnie ten identyfikator, ponieważ poznanie jego wartości przez osoby niepowołane może dać im dostęp do danych przechowywanych przez mechanizm sesji.

Identyfikator sesji może być przekazywany pomiędzy kolejnymi żądaniami w postaci ciasteczka lub bezpośrednio w hiperłączach i formularzach. Każda z tych metod ma swoje wady i zalety (zob. tabela 1). Nie ma sposobu doskonałego, jednak sesje oparte na ciasteczkach gwarantują największy poziom bezpieczeństwa. Jednym z głównych zagrożeń tego rozwiązania jest to, że nagłówek `set-cookie` wysyłany jest każdorazowo w ramach danej domeny, w wyniku czego możliwym jest jego podsłuchanie przez napastnika. Aby temu zapobiec, za każdym razem kiedy aplikacja wysyła identyfikator sesji, komunikacja pomiędzy klientem i serwerem powinna odbywać się poprzez bezpieczny protokół HTTPS, zamiast standardowego HTTP. Ciasteczka nie są przesyłane wtedy tekstem jawnym, więc podsłuchiwanie połączenia nie przyniesie atakującemu żadnych korzyści. Wdrożenie SSL nie zawsze jest jednak możliwe, ale na szczęście istnieje także szereg innych sposobów, które w znacznym stopniu utrudniają przejęcie sesji użytkownika. Innym problemem ciasteczek jest możliwość nie przyjęcia ich przez odwiedzających witrynę. Jest to koszt, który trzeba ponieść, ale lepszym wyjściem jest wyjaśnienie użytkownikowi dlaczego ciasteczka są potrzebne, niż opieranie obsługi sesji na rozwiązaniu mało bezpiecznym. Poza tym, według różnych rankingów [WWW1; WWW2], kwestia ta dotyczy zaledwie około jednego procenta internautów.

Metoda	Opis i przykład	Zalety	Wady
Adres URL	<p>Identyfikator sesji osadzony jest bezpośrednio w adresie URL. Aplikacja otrzymuje go metodą HTTP GET po kliknięciu przez użytkownika na hiperłącze.</p> <p>Przykład : <code>http://s.pl/?SID=1234</code></p>	<ol style="list-style-type: none"> 1. Adres zawierający identyfikator może być wykorzystany do upewnienia się, że użytkownik porusza się po aplikacji według założonej ścieżki. 2. Dostęp do danego zasobu może być umożliwiony przez użytkownika zwyczajnie przez przekazanie adresu zawierającego identyfikator. 	<ol style="list-style-type: none"> 1. Każda osoba używająca tego samego komputera będzie mogła poznać identyfikator sesji z historii odwiedzonych stron przeglądarki. 2. Adresy URL często zapisywane są przez serwery proxy, dzienniki zdarzeń, itp. W wyniku ich analizy można przeczytać identyfikatory sesji. 3. Bardzo łatwo zmodyfikować identyfikator sesji poprzez manipulację łańcuchem żądania. 4. Kiedy użytkownik przechodzi na inną stronę, adres URL wraz z identyfikatorem sesji dostępny jest w nagłówku HTTP <code>Referer</code>. 5. Użytkownik, przekazując adres URL innym, może nieumyślnie umożliwić dostęp do własnych danych sesyjnych.
Ukryte pola formularzy	<p>Identyfikator sesji jest umieszczony jako wartość ukrytego pola. Przekazywany jest metodą HTTP POST po zatwierdzeniu formularza przez użytkownika.</p> <p>Przykład : <code><input type="hidden" name="SID" value="1234"></code></p>	<ol style="list-style-type: none"> 1. Identyfikator sesji jest trudniejszy do modyfikacji niż w przypadku adresu URL. 2. Pozwala użytkownikowi na przekazywanie adresu URL bez umożliwiania dostępu do własnej sesji. 	<ol style="list-style-type: none"> 1. Bardzo komplikuje budowę strony. W praktyce niemożliwe jest oparcie przekazywania identyfikatora sesji na całej witrynie tylko i wyłącznie używając ukrytych pól formularzy.
Ciasteczka (<i>Cookies</i>)	<p>Identyfikator sesji jest przechowywany w ciasteczkach i przekazywany w nagłówkach odpowiedzi HTTP.</p> <p>Przykład : <code>Set-Cookie: SID="1234"; path="/"; domain="s.pl"; expires="2006-06-01 00:00:00GMT";</code></p>	<ol style="list-style-type: none"> 1. Uniwersalność – doskonały mechanizm przekazywania dowolnych danych pomiędzy żądaniami. 2. Pozwalają na lepszą kontrolę czasu trwania sesji. 3. Mogą być tak skonfigurowane, aby były trwałe pomiędzy sesjami. 4. Identyfikator sesji nie może być poznany ani przez analizę źródła strony, ani adresu URL. 	<ol style="list-style-type: none"> 1. Nagłówek <code>Set-Cookie</code> wysyłany jest za każdym razem w ramach danej domeny. 2. Ciasteczka przechowywane są na komputerze użytkownika w plikach tekstowych i mogą być skopiowane.

Tabela 1: Zalety i wady metod przechowywania identyfikatora sesji z punktu widzenia bezpieczeństwa

Źródło: Opracowanie na podstawie [Ollm03] i [Schl04, s.328].

PHP posiada, standardowo dołączane do pakietu, rozszerzenie obsługi sesji. Jednakże, z punktu widzenia bezpieczeństwa, mechanizm ten nie można uznać za w pełni wystarczający. Wymaga on od programisty implementacji dodatkowych zabezpieczeń na poziomie aplikacji.

Poniżej przedstawiono w punktach praktyki, których należy przestrzegać tworząc dobry mechanizm obsługi sesji :

- (1) Konfiguracja - mechanizm powinien wymusić używanie ciasteczek do przekazywania identyfikatora sesji poprzez odpowiednie dyrektywy konfiguracyjne : `session.use_cookies` (ustawia przekazywanie identyfikatora w ciasteczkach), `session.use_only_cookies` (ustawia pobieranie identyfikatora tylko z ciasteczek) oraz `session.trans_sid` (ustawia przekazywanie identyfikatora poprzez adres URL i formularze WWW – możliwość tą oczywiście należy wyłączyć). W konfiguracji powinny też zostać ustawione odpowiednie parametry dotyczące samego ciasteczka sesji . Szczególną uwagę należy zwrócić na ograniczenie ścieżki, dla której powinno ono obowiązywać, ponieważ domyślne ustawienie dla całego głównego katalogu „/” zwiększa liczbę możliwości ataku [Onea04, s.3]. Dyrektywy te można ustawić wewnątrz skryptu, dlatego są całkowicie obsługiwane przez komponent *spcSession* (po jego inicjalizacji, a przed wystartowaniem sesji).
- (2) Kontrola czasu trwania sesji - obsługa sesji powinna w pełni kontrolować czas, po jakim sesja traci ważność albo jest usuwana z magazynu przechowującego. Sesja powinna być przerwana, gdy [Bakl02, s.11; Alsh05, s.120]: nie stwierdzono żadnej aktywności przez pewien okres czasu (standard to 24 minuty) lub gdy sesja, mimo aktywności, trwa zbyt długo (przeważnie kilka godzin) oraz gdy nastąpi błąd bezpieczeństwa. Użytkownik powinien mieć także możliwość przerwania sesji samemu, poprzez opcje wylogowania z systemu [Ollm03; Kols02, s.13].
- (3) Usuwanie nieaktywnych sesji - mechanizm powinien automatycznie usuwać z magazynu danych te identyfikatory i dane sesyjne, których czas ważności został przekroczony. Także ciasteczka powinny mieć ograniczony czas istnienia. Oczyszczanie oferowane przez rozszerzenie *Session* nie jest doskonałe, ponieważ odbywa się ono z zadanym prawdopodobieństwem, więc możliwe jest, że sesja nie zostanie usunięta, mimo że jej czas aktywności minął [Alsh05, s. 121]. Ustawienie prawdopodobieństwa na 100% mogłoby się sprawdzić w przypadku bardzo małych serwisów o znikomej oglądalności – w innym wypadku byłoby zbyt obciążające. Alternatywnie do usuwania przestarzałych sesji, można wykorzystać demon Cron¹ lub informacje na temat wygaśnięcia przetrzymywać bezpośrednio wśród danych sesyjnych.

1 W systemie Windows można wykorzystać w tym celu harmonogram

- (4) Unikanie trwałych sesji – pozwalają na identyfikację użytkownika pomiędzy sesjami przeglądarki bez konieczności ponownego podawania danych uwierzytelniających. Typowym przykładem takiej funkcjonalności jest, udostępniana w momencie logowania, opcja “Zapamiętaj mnie” [Endl01, s.6; Murp06; s.19].
- (5) Stosowanie silnego algorytmu tworzenia identyfikatora sesji o dużej długości – aby identyfikator był trudny do odgadnięcia musi być [Ollm03]: losowy, trudny do odgadnięcia, trudny do odtworzenia oraz nie za krótki. Wbudowany mechanizm obsługi sesji prezentuje się pod tym względem bardzo dobrze - do wyboru daje funkcje skrótu MD5 (128-bitowa) i SHA-1 (160-bitowa). Jedyne wątpliwości może budzić fakt, że algorytm tworzenia identyfikatorów jest znany – można go podejrzec w kodzie źródłowym PHP, więc odgadywanie identyfikatorów sesji jest nieco ułatwione [WiLa05, s.368]. Z pomocą przychodzą tutaj dyrektywy konfiguracyjne `session.entropy_file` oraz `session.entropy_length`, które powodują zmianę w działaniu generatora liczb losowych.
- (6) Uniemożliwienie tzw. adopcji sesji (ang. *session adoption*) – mechanizm powinien ignorować identyfikatory pochodzące od użytkownika, które nie znajdują się w magazynie danych – akceptowalne są tylko te, które system sam utworzył [Kols02, s.12; Alsh06, s.55].
- (7) Rotacja identyfikatorem – przy zmianie poziomu uprawnień użytkownika identyfikator sesji powinien ulec zmianie. Absolutnym minimum jest, aby ta zmiana nastąpiła po przejściu procesu uwierzytelniania. Ilia Alshanetsky proponuje [Alsh05, s.126] regenerację identyfikatora przy każdym nowym żądaniu. O ile zapewnia to bardzo wysoki stopień bezpieczeństwa, może mieć zbyt duże konsekwencje wydajnościowe. Wszystko zależy od wielkości i rodzaju systemu informatycznego. Pośrednim rozwiązaniem może być podmiana identyfikatora co kilka żądań.
- (8) Blokowanie buforowania zawartości po stronie klienta – strony zawierające informacje przeznaczone tylko dla użytkowników posiadających określony zestaw uprawnień, nie powinny być buforowane zarówno w pamięci podręcznej przeglądarki, jak i w pamięci serwerów pośredniczących [Murp06, s.19]. Rozszerzenie dołączane do PHP automatycznie ustawia odpowiednie nagłówki odpowiedzi HTTP.
- (9) Po stronie klienta przechowywany jest tylko identyfikator sesji – w ciasteczkach nie powinno być umieszczane żadne dane poza identyfikatorem sesji [Alsh05, s.119]. Nic nie stoi na przeszkodzie, aby wszystkie pozostałe dane były przechowywane po stronie serwera w magazynie danych.
- (10) Dane sesyjne można zabezpieczyć dodatkowo poprzez szyfrowanie bezpośrednio przed

umieszczeniem w magazynie danych [Shif05]. Odnosi się tylko do tych systemów, w których wymagany jest wysoki poziom bezpieczeństwa, a dostęp do tabeli bazy danych przechowującej dane sesyjne mogą mieć niepowołani do tego użytkownicy.

- (11) Krytyczne operacje dla działania systemu oraz konta użytkownika, jak np. zmiana hasła, powinny być poprzedzone ponowną autentykacją [Murp06, s.19]. Ogranicza to zakres ataku, gdy napastnikowi uda się wykraść identyfikator sesji.
- (12) Rozszerzenie PHP *Session* domyślnie przechowuje dane sesyjne w plikach na serwerze lokalnym w jednym katalogu. Na serwerach współdzielonych, dostęp do tego katalogu mają wszyscy użytkownicy i w praktyce mogą uzyskać dostęp do wszystkich identyfikatorów oraz danych sesji. Niektóre źródła radzą [Mrug06, s.75] ustawienie dyrektywy konfiguracyjnej `session.save_path`, która definiuje katalog przechowywania plików sesji. Nie gwarantuje ona jednak ochrony na odpowiednim poziomie, bowiem inni użytkownicy w dalszym ciągu będą mieli dostęp do danych sesji, jeśli tylko uda im się odnaleźć ich położenie [Alsh06, s.53]. W literaturze duża część autorów, w tym autor tego tekstu, skłania się więc raczej do wykorzystania bazy danych jako magazynu przechowywania danych sesyjnych (jeśli tylko wydajność aplikacji nie jest elementem krytycznym). Rozszerzenie funkcjonalności standardowej biblioteki obsługi sesji jest o tyle łatwiejsze, że umożliwia ono zdefiniowanie własnej obsługi bazy danych (`session_set_save_handler`).

Mechanizm obsługi sesji może wprowadzić dodatkowe zabezpieczenia polegające na sprawdzaniu czy żądanie pochodzi z tego samego adresu IP, przeglądarki czy też zostało wysłane z określonego adresu URL. W praktyce jednak żadne z tych rozwiązań nie jest doskonałe. Jest to wynikiem tego, że podczas przeglądania internetu kolejne żądania w ramach jednej i tej samej sesji mogą przechodzić przez różne serwery proxy – dla serwera wygląda to tak, jakby pochodziły z różnych adresów IP [Schl04, s.333]. To samo dotyczy ciągu określającego przeglądarkę. Fakt ten znacznie zmniejsza funkcjonalność całej aplikacji, ponieważ w praktyce ogranicza dostęp użytkowników do systemu, co przeważnie jest nie do zaakceptowania.

Jeden artykuł to za mało, aby kompleksowo i szczegółowo opisać problemu bezpieczeństwa sesji użytkownika, dlatego gorąco zachęcam do zgłębiania tematu we własnym zakresie – szczególnie polecam pozycje wymienione na końcu tego artykułu.

Literatura

- [Alsh05] Alshanetsky I.: *PHP Architect's Guide to PHP Security*, Marco Tabini & Associates, Inc. 2005.
- [Alsh06] Alshanetsky I.: *Security Corner: All Your Session Are Still Belong to Us!*, PHP Architect nr 6/2006, s.50-55.

- [BaK102] Bar-Gad I., Klein A.: *Developing Secure Web Applications*, Sanctum Inc. 2002. http://www.cgisecurity.com/lib/WhitePaper_DevelopingSecureWebApps.pdf
- [Endl01] Endler D.: *Session ID Brute Force Exploitation*, iDEFENSE Labs 2001. <http://www.cgisecurity.com/lib/SessionIDs.pdf>
- [Esse05] Esser S.: *PHP and Session Fixation*, Hardened-PHP Project 2005. http://www.hardened-php.net/php_and_session_fixation.48.html
- [Fowl05] Fowler M.: *Architektura systemów zarządzania przedsiębiorstwem. Wzorce projektowe*, Helion 2005.
- [Jaku03] Jakubowicz M.: *Bezpieczeństwo skryptów PHP*, Konferencja PHPCon2003, 21. listopad 2003, Poznań.
- [Kols02] Kolsek M.: *Session Fixation Vulnerability in Web-based Applications*, ACROS Security 2002. http://www.acros.si/papers/session_fixation.pdf
- [Mrug06] Mrugalski J.: *Techniki zatruwania sesji w PHP*, PHP Solutions nr 3/2006, s.72-75.
- [Murp06] Murphy C.: *Security for websites – breaking sessions to hack into a machine*, (IN)SECURE Magazine nr 3/2006, s.18-20. <http://www.insecuremag.com/archive.html>
- [Ollm03] Ollmann G.: *Web Based Session Management*, TechnicalInfo.net 2002. <http://www.technicalinfo.net/papers/WebBasedSessionManagement.html>
- [Olse04] Olsen O.K.: *Security Aspects of a PHP/MySQL-based Login System for Web Sites*, 2004. <http://my.opera.com/community/articles/php/securitysystem/>
- [Schl04] Schlossnagle G.: *PHP. Zaawansowane programowanie. Vademecum profesjonalisty*, Helion 2004.
- [ScSh02] Scambray J., Shema M.: *Hakerzy - Aplikacje webowe*, Translator 2002.
- [Shif04] Shiflett C.: *PHP Security*, Konferencja ApacheCon US 2004, 13-17.11.2004, Las Vegas (USA). <https://security.nihilisme.ca/files/php-security.pdf>
- [Shif05] Shiflett C.: *Essential PHP Security*, O'Reilly 2005.
- [WiLa05] Williams H., Lane D.: *PHP i MySQL. Aplikacje bazodanowe*, Helion 2005.
- [WWW1] *Security Space*. http://www.securityspace.com/s_survey/data/index.html
- [WWW2] *Ranking.pl*. <http://ranking.pl/>