

# Wyciek informacji w aplikacjach internetowych

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

SegfaultLabs, czerwiec 2006

Wyciek informacji (*Information Leakage*) związany jest z tymi elementami aplikacji internetowych, które mogą dostarczyć atakującemu szczegółowych informacji na temat witryny. Może on je poznać przeglądając stronę, źródła HTML oraz sprawdzając jej reakcje na różne działania. Może także skorzystać z pomocy różnych narzędzi, chociażby wyszukiwarek internetowych (*Google Hacking*). Informacje te często bywają kluczowe dla skuteczności ataku. Słabymi punktami będącymi zazwyczaj przyczyną tego typu wycieku informacji są :

- Komunikaty błędów i komunikaty systemowe – te komponenty aplikacji, które mają dostęp do jawnej postaci danych, umieszczają je w zgłaszanych przez siebie komunikatach, np. komunikaty zapisywane w dziennikach zdarzeń [SzWi06, s.136]. Komunikaty błędów i komunikaty systemowe mogą być nieocenionym źródłem pożytecznych informacji na temat działania aplikacji, np. ścieżki dostępu, kolumny użyte w zapytaniach, nazwy użytkowników bazy danych, i wielu, wielu innych.
- Polecenia debugowania – niemal każda aplikacja wykorzystuje mechanizmy obserwacji działania i usuwania błędów z programu (tzw. debugowania). Są one bardzo przydatne w czasie rozwoju oprogramowania, jednak pozostawione w aplikacji mogą prowadzić do ujawnienia wielu informacji. Atakujący może znaleźć wyniki działania procesu debugowania bezpośrednio w kodzie lub wymusić wejście aplikacji w tryb debugowania poprzez modyfikację adresu URL, np. w niektórych aplikacjach wystarczy dodanie do łańcucha parametrów `debug=on` lub `Debug=Yes` [OWASP02, s.51].
- Komentarze HTML – komentarze często poprawiają czytelność i usprawniają proces dokumentowania aplikacji, jednakże umieszczone bezpośrednio w kodzie HTML mogą okazać się cennymi wskazówkami dla włamywacza [OWASP02, s.50]. Może być ich dużo i mogą nie zawierać żadnych cennych informacji lub może być ich mało ale np. z hasłami użytkowników czy też opisami tabel używanych podczas zapytań SQL [ScSh02, s.110]. Komentarze te mogą być umieszczane bezpośrednio przez programistów lub automatycznie w wygenerowanym kodzie przez różnego rodzaju programy i biblioteki pomocnicze [OWASP02, s.50]. Niemniej jednak, bez względu na sposób w jaki komentarze HTML znalazły się w kodzie, należy uznać tą lukę za bardzo kompromitującą.

Problem wycieku informacji można rozpatrywać także w innym aspekcie. Dotyczy on ujawnienia ukrytych bądź poufnych danych, plików lub dokumentów, które same w sobie mogą być przedmiotem ataku lub też mogą przyczynić się do jego przeprowadzenia. Problem nieuprawnionego dostępu do zasobów, które powinny pozostać poufne, wynika głównie z faktu, że umieszczane są one w ramach głównego katalogu dokumentów WWW serwera. W ten sposób można uzyskać do nich dostęp bezpośrednio po wpisaniu dokładnego adresu URL w oknie przeglądarki. Możliwe jest to wtedy, gdy mechanizm kontroli dostępu nie obejmuje wszystkich możliwych punktów wejścia do systemu lub gdy dopuszcza on użytkownika do zasobów, do których nie uzyskał wymaganych uprawnień podczas uwierzytelniania. Poza tym, programiści zbyt często zapominają, że nawet jeśli do danego zasobu nie prowadzą żadne bezpośrednie odnośniki ze strony, to wcale nie znaczy, że nie jest on osiągalny dla potencjalnych krakerów. Dokładny adres URL może być odnaleziony poprzez atak siłowy, sprawdzanie istnienia popularnych nazw katalogów i plików (np. /admin), a także analizę komunikatów błędów [WASC04, s.13]. Elementy systemu informatycznego, których dotyczy ten rodzaj zagrożenia to :

- Pliki ukryte – wielu administratorów stron internetowych pozostawia na serwerze pliki ukryte, np. pliki z przykładami czy pliki instalacyjne, tymczasem mimo, że z reguły nie prowadzą do nich żadne odnośniki, to wciąż są one dostępne z poziomu WWW [OWASP02, s.51].
- Kopie zapasowe i pliki tymczasowe – wiele aplikacji pozostawia w katalogach pliki kopii zapasowych i pliki tymczasowe [OWASP02, s.51], w wyniku czego, podobnie do wspomnianych wyżej plików ukrytych, są one dostępne dla wszystkich. Szczególnym zagrożeniem jest uzyskanie dostępu przez atakującego do kopii zapasowych, które to są swoistą skarbnicą wiedzy o całej aplikacji. Kopie zapasowe domyślnie tworzone są automatycznie przez niektóre edytory (np. Kate, KWrite).
- Tyłne wejścia (*backdoors*) – przeważnie tworzone podczas fazy rozwijania programu dla szybszego dokonywania zmian bezpośrednio w aplikacji, poprzez umożliwienie uzyskania administratorskiego dostępu do aplikacji bez procesu autoryzacji i uwierzytelniania, jednakże bardzo często zapomina się o ich usunięciu przed umieszczeniem aplikacji w Internecie [Grze04, s.61].
- Pliki źródłowe – problem ujawnienia zawartości plików źródłowych aplikacji zazwyczaj jest wynikiem tego, że dołączane pliki często mają rozszerzenia *inc* (od angielskiego wyrazu *include*) i są przechowywane w ramach głównego katalogu aplikacji; serwer WWW nie rozpoznaje typu takich plików i traktuje je jako dokumenty zapisane zwykłym tekstem (`text/plain`); w rezultacie zawartość pliku może być wyświetlona

bezpośrednio w oknie przeglądarki [Shif05].

Ujawnienie plików źródłowych możliwe jest także, gdy plik posiada inne rozszerzenie np. *php*. Wykorzystywana jest wtedy technika zwana *Directory Traversal*. Pozwala ona na ujawnienie plików źródłowych i innych zasobów składowanych lokalnie na serwerze. Na atak ten podatne są wszelkie skrypty korzystające z plików szablonów lub w jakiś inny sposób odwołujące się do plików na serwerze [ScSh02, s.207]. Atak ten przebiega poprzez manipulację łańcuchem żądania.

Nie występuje powszechnie używany polski odpowiednik terminu *Directory Traversal*. W literaturze angielskojęzycznej używa się także określeń *Path Traversal* [WASC04a, s. 52] oraz *Path Disclosure* [OWASP02, s.40].

Podstawową formą ataku *Directory Traversal* jest przejście poza główny katalog witryny, aby uzyskać dostęp do plików systemowych, np. `../../../../etc/passwd` [ScSh02, s. 207]. Jednak atak może być także wykorzystany do ujawnienia zasobów i plików źródłowych w ramach głównego katalogu strony WWW, np. zamierzonym efektem po wpisaniu adresu `http://strona.pl/index.php?dzial=glowny.htm` może być wyświetlenie głównego szablonu znajdującego się w pliku `glowny.htm`. Tymczasem atakujący może zmodyfikować parametr `dzial` i umieścić nazwę dowolnego innego pliku w ramach serwera, w rezultacie powodując wyświetlenie go w przeglądarce. W tej sytuacji pewnym rozwiązaniem wydaje się być wymuszenie rozszerzenia pliku, jednak nie zawsze jest to skuteczne, ponieważ napastnik może umieścić w adresie znak pusty `NULL (%00)`, który oznacza koniec ciągu [ScSh02, s.208-209]. W trakcie przetwarzania takiego ciągu, w którym został umieszczony znak pusty, wszystkie znaki znajdujące się po nim zostaną obcięte, w rezultacie czyniąc powyższe zabezpieczenie bezużytecznym. Wobec tego faktu konieczność poprawnej walidacji i filtracji danych wejściowych nabiera szczególnego znaczenia.

Wystąpienie zagrożenia wycieku poufnych informacji zazwyczaj jest efektem niedopatrzienia lub braku dogłębnej analizy aplikacji. Tymczasem dzięki temu atakujący mogą dogłębnie poznać nie tylko strukturę i budowę witryny, ale i szczegółowe informacje na temat zastosowanych technik, a także dane używane do ustanowienia połączenia z bazą danych. Niewątpliwie stanowi to poważne zagrożenie.

Jak się chronić? Po pierwsze, zaimplementować dobry system obsługi błędów (np. w PHP własny *error handler*), który zwraca ogólne i jednolite komunikaty; po drugie upewnić się, że nie zostawiliśmy w HTMLu żadnych komentarzy, a w obrębie głównego katalogu WWW żadnych zbędnych plików (kopii zapasowych, plików ukrytych, instalacyjnych, itp.); po trzecie skupulatnie weryfikować i filtrować wszystkie dane wejściowe.

## Literatura

- [Grze04] Grzesiak P.: *Bezpieczeństwo stron internetowych*, Internet nr 1/2004, s.59-61.
- [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](http://www.owasp.org/index.php/Category:OWASP_Guide_Project)
- [ScSh02] Scambray J., Shema M.: *Hakerzy - Aplikacje webowe*, Translator 2002.
- [Shif05] Shiflett C.: *Essential PHP Security*, O'Reilly 2005.
- [SzWi06] Szeliga M., Wileczek R.: *PHP5. Tworzenie bezpiecznych stron WWW*, Helion 2006.
- [WASC04] *Threat Classification*, Web Application Security Consortium 2004.  
<http://www.webappsec.org/projects/threat/>