

Ochrona procesu uwierzytelniania w aplikacjach internetowych

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

Uwierzytelnianie użytkowników można zdefiniować jako proces weryfikacji czy dany użytkownik jest tą osobą, za którą się podaje [OWASP02, s.16]. Najpopularniejszym sposobem uwierzytelniania użytkowników w Internecie są obecnie hasła. Niestety ciągle obserwowany jest wzrost efektywności rozmaitych programów do łamania szyfrowanych haseł [KLG03, s.393].

Wyróżnia się dwa podstawowe sposoby łamania haseł [SGT05, s.229] :

- Atak słownikowy (*dictionary attack*) - zautomatyzowany atak skierowany przeciwko systemowi uwierzytelniania, który polega na sprawdzeniu kolejnych, gotowych haseł znajdujących się w bazie danych, tzw. słowniku,
- Atak siłowy (*brute-force password attack*) - polega na omijaniu zabezpieczeń systemu przez podejmowanie prób zalogowania się przy użyciu każdego dopuszczalnego hasła; w tej metodzie analizowany jest każdy możliwy przypadek.

Bez względu na szczegółowe nazwy tych ataków, należy je traktować jako techniki siłowe, czyli takie, które w drodze zautomatyzowanego procesu metodą prób i błędów próbują odgadnąć dane uwierzytelniające.

Według innego podziału można wyróżnić następujące ataki siłowe [WASC04, s.11] :

- Zwyczajne ataki siłowe (*normal brute force attacks*) – atakujący używa nazwy użytkownika i dopasowuje do niego hasła.
- Odwrócone ataki siłowe (*reverse brute force attacks*) – atakujący używa jednego hasła i dopasowuje do nich nazwy użytkowników. W systemach z milionami kont, prawdopodobieństwo tego, że wielu użytkowników posiada to samo hasło jest wysokie.

Wiele systemów uwierzytelniających borykało się z problemem ataków siłowych, dlatego opracowano metody ochrony przed nimi [John04, s.24] :

- Wprowadzenie sztucznej przerwy między momentem wprowadzenia loginu i hasła a wyświetleniem komunikatu o powodzeniu lub niepowodzeniu uwierzytelniania.
- Blokowanie konta po osiągnięciu pewnej określonej liczby prób logowania zakończonych niepowodzeniem.

Niestety metody te nie do końca sprawdzają się w sieci WWW. Wadą pierwszego z wymienionych sposobów jest to, że atakujący może wykonać wiele prób logowania jednocześnie, w wyniku czego zabezpieczenie staje się mało skuteczne, a nawet może otworzyć drogę do

przeprowadzenia ataku odmowy usługi (*Denial Of Service*) [John04, s.24]. Podobnie wspomniane później blokowanie konta nie jest rozwiązaniem najlepszym. Do głównych wad tej metody należą [Burn04] :

- Nawet po zablokowaniu konta, atak może być kontynuowany. Atakujący może nieprzerwanie powodować blokowanie tego samego konta, nawet kilka sekund po jego odblokowaniu, w ten sposób efektywnie uniemożliwiając dostęp do konta.
- Atakujący może przeprowadzić atak *Denial Of Service* poprzez zablokowanie dużej liczby kont.
- Dzięki otrzymywanym komunikatom błędów, atakujący może stworzyć bazę nazw użytkowników systemu. Wykorzystywany jest tu fakt, że niemożliwe jest zablokowanie konta, które nie istnieje.
- Blokowanie konta jest nieskuteczne wobec tzw. odwróconych ataków siłowych.

Na szczęście w środowisku sieciowym wykształciły się także inne, bardziej skuteczne, metody przeciwdziałania atakom siłowym. Jedną z nich to użycie techniki zwanej CAPTCHA (ang. *Completely Automated Public Turing Test to Tell Computers and Humans Apart*). Polega ona na wyświetleniu obrazka z losowym napisem, który użytkownik musi wpisać w specjalne pole. Ponieważ roboty sieciowe przeprowadzające atak nie mogą odczytać napisu, nie będą też w stanie poznać wyniku autentykacji. Jedną z podstawowych wad tej metody jest to, że uniemożliwia ona dostęp do uwierzytelniania niektórym osobom niepełnosprawnym, a także użytkownikom korzystającym z przeglądarek tekstowych. Z tychże powodów powinna być akceptowana tylko dla stron wymagających wysokiego poziomu bezpieczeństwa [John04, s.25].

Innym sposobem jest blokowanie ze względu na adres IP. Problemem może być tutaj możliwość skorzystania przez atakującego z różnych serwerów pośredniczących, a także fakt, że zablokowanie jednego adresu IP, może powodować automatyczną blokadę wielu innych użytkowników, którzy posiadają ten sam adres [Burn04]. Mimo tych problemów, rozwiązanie to i tak wydaje się o wiele lepsze niż blokowanie na podstawie nazwy użytkownika (loginu), ponieważ można wtedy uniknąć możliwości blokowania kont użytkowników przez napastnika.

Wartym odnotowania jest fakt, że atak siłowy z reguły wymaga dużych nakładów czasowych, co może być często zniechęcającym czynnikiem dla napastnika. Kluczową rolę odgrywa tutaj stopień skomplikowania haseł przechowywanych w systemie. Dlatego też jednym ze sposobów ochrony jest wymuszanie na użytkownikach wybierania haseł trudnych do odgadnięcia. Rozwiązanie takie można zaobserwować chociażby w systemie operacyjnym Linux (przy użyciu biblioteki *cracklib*). Hasła, aby uznać za bezpieczne, powinny przestrzegać następujących

wymogów [SzWi06, s.91; FSSF01, s.7]:

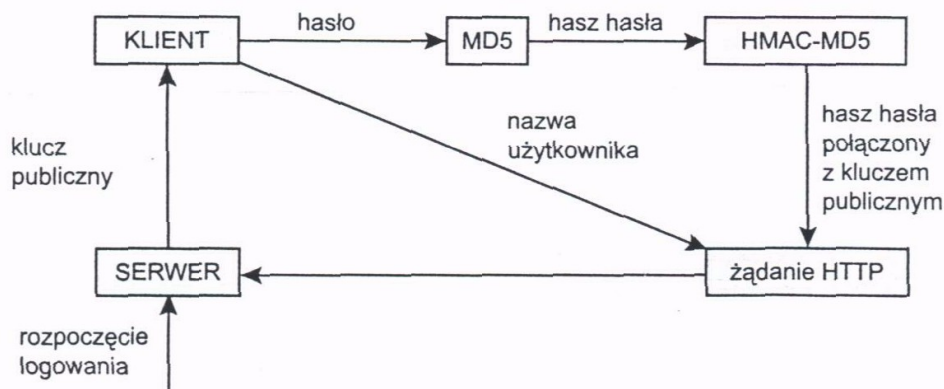
- Hasło nie powinno składać się z imienia, nazwiska, adresu, daty urodzenia, nazwy użytkownika lub jego znajomego albo członka rodziny.
- Hasło nie powinno być wyrazem jakiegokolwiek języka.
- Hasło powinno mieć odpowiednią długość oraz być przynajmniej 8-12 znakowe.
- Hasło powinno być kombinacją różnych znaków, tj. wielkich i małych liter alfabetu, cyfr oraz znaków specjalnych.

Hasła powinny być także składowane przez system w sposób bezpieczny, uniemożliwiający ich wykradnięcie i poznanie przez osoby trzecie. Mechanizmy, które przechowują hasła w postaci jawnej lub zaszyfrowanej odwracalnie, w znacznym stopniu obniżają poziom bezpieczeństwa aplikacji [SzWi06, s.93]. Warty odnotowania jest tu fakt, że wszelkie algorytmy szyfrujące powinny być tworzone tylko przez doświadczonych ekspertów z dziedziny kryptografii. Bardzo często udowodniano bowiem, że algorytmy tworzone przez amatorów były słabe i łatwe do złamania [FSSF01, s.5]. Naturalnym wyborem na bezpieczne przechowywanie haseł jest użycie jednokierunkowych funkcji skrótu. Spośród dostępnych rozwiązań najlepszą alternatywą jest funkcja haszująca SHA-1, ponieważ jest dużo szybsza niż konstrukcje stosujące szyfry blokowe, a przy tym daje dłuższy skrót niż MD5, dzięki czemu jest bardziej odporna na ataki siłowe [Schn02, s.543-559]. Jednakże dla przechowywania haseł, algorytm MD5 również daje wystarczający poziom bezpieczeństwa, gdyż odtworzenie hasła o zadanym haszu jest niewykonalne nawet w ciągu miliardów lat (wymaga 2^{128} operacji) i jak dotąd nie odkryto żadnego sposobu na znaczne skrócenie tego czasu [ScHa05, s.10]. Jako dodatkowe zabezpieczenie należy, przed haszowaniem, hasło połączyć z dowolnym, wcześniej ustalonym ciągiem znaków. W Internecie pojawiły się bowiem bazy haszów MD5 wraz z jawnymi odpowiednikami¹ i odgadnięcie „czystego” haszu jest dzięki nim o wiele prostsze.

Innym zagrożeniem związanym z hasłami, jest możliwość ich podsłuchania (ang. *sniffing*) w momencie logowania się użytkownika do systemu. Najskuteczniejszą ochroną jest tutaj zastosowanie połączenia szyfrowanego SSL, ale nie zawsze jest to możliwe. Wtedy z pomocą przychodzi weryfikacja HMAC (ang. *Keyed-Hashing for Message Authentication*). Celem HMAC jest weryfikowanie integralności danych przesyłanych w Internecie (np. poprzez formularze webowe) przy użyciu skrótów kryptograficznych i kodów uwierzytelniających komunikaty. Jak pokazał Łukasz Lach [LaSt06, s.34-46], algorytm ten można wykorzystać także do zabezpieczenia formularza uwierzytelniającego. Rozwiązanie to polega na generowaniu skrótu hasła na podstawie losowo wybranego klucza, zarówno po stronie klienta (w momencie wpisania hasła), jak i po

¹ <http://gdataonline.com/seekhash.php>, <http://md5.rednoize.com/>, <http://passcracking.com/>

stronie serwera. Po wysłaniu formularza wyniki obu funkcji skrótu są ze sobą porównywane, w wyniku czego można stwierdzić czy dane nie zostały zmodyfikowane. Przy każdej próbie logowania generowany jest całkowicie nowy klucz, więc wynikowy skrót hasła także będzie zawsze inny. W efekcie nawet jeśli napastnikowi uda się podsłuchać połączenie, to i tak nie będzie mógł wykorzystać zdobytych informacji do uwierzytelnienia. Przebieg autentykacji przy użyciu tej metody został przedstawiony na rysunku 1.



Rysunek 1: Schemat działania bezpiecznego systemu logowania opartego na HMAC

Źródło: [LaSt06, s.35]

Innym problemem dotyczącym bezpieczeństwa procesu uwierzytelniania jest funkcjonalność przeglądarek internetowych umożliwiająca buforowanie ostatnio odwiedzonych stron. Gdy napastnik ma możliwość skorzystania z tego samego komputera, co ofiara (np. w szkole, bibliotece, kawiarence internetowej, itp.), może używając przycisku „Wstecz” cofnąć się na stronę logowania i ponownie wysłać buforowane w pamięci przeglądarki dane uwierzytelniające ofiary. Istnieją następujące możliwe sposoby zabezpieczenia przed tym niepożądanym zachowaniem :

- Zaraz po dokonaniu autentykacji należy automatycznie przekierować użytkownika na inną stronę (w PHP instrukcja `header('location: url')`) [Kohl04, s.7-9]. Metoda ta ma także dodatkową zaletę. Standardowo, gdy użytkownik spróbuje odświeżyć stronę zaraz po zatwierdzeniu formularza, przeglądarka spyta o ponowne wysłanie danych. Jest to działanie niepożądane, ale dzięki zastosowaniu automatycznego przekierowania zachowanie to nie wystąpi.
- Przekazywać w formularzu specjalny token (np. hasz MD5) - ten sam token powinien być przechowany jako jedna ze zmiennych sesyjnych, a po zatwierdzeniu formularza oba tokeny powinny być porównane. Następnie token należy usunąć lub zastąpić nowym. Gdy atakujący spróbuje wysłać ponownie dane uwierzytelniające, tokena sesyjnego już nie będzie i zostanie zgłoszony błąd.

- Zaimplementować wcześniej opisaną weryfikację HMAC. Działa podobnie do sposobu polegającego na przekazywaniu tokena. Klucz oparty jest na zbliżonej zasadzie jak wspomniany token.

Każda z tych metod stanowi wystarczające zabezpieczenie, ale nic nie przeszkadza, aby mogły być one ze sobą łączone.

Specyficzną odmianą zagrożeń związanych z procesem uwierzytelniania są ataki socjotechniczne. Największy problem polega na tym, że nie są to ataki techniczne, a jedynie próby nakłaniania użytkowników do wykonania nierozsądnych działań. Najczęściej przyjmują następującą formę [Schl04, s.332] :

- Podawanie się za administratora systemu i wysyłanie wiadomości poczty elektronicznej do użytkowników z prośbą o podanie haseł.
- Utworzenie lustrzanej kopii strony logowania witryny i nakłanianie użytkowników do prób logowania (Phishing).

Wyżej wymienione formy ataku często są też ze sobą łączone. Ochrona przed tego typu zagrożeniami na poziomie aplikacji jest niemożliwa. Praktycznie jedyne, co można zrobić, to uświadomić użytkowników o istniejącym niebezpieczeństwie tak, aby starali się go unikać.

Literatura

- [Burn04] Burnett M.: *Blocking Brute-Force Attacks*, IIS Tips & Tricks Resources, 2004.
<http://www.iis-resources.com/modules/AMS/article.php?storyid=275>.
- [FSSF01] Fu K., Sit E., Smith K., Feamster N.: *Dos and Don'ts of Client Authentication on the Web*, Massachusetts Institute of Technology 2001. <http://cookies.lcs.mit.edu/pubs.html>
- [John04] Johnston P.: *Authentication and Session Management on the Web*, GSEC 2004.
http://www.giac.org/certified_professionals/practicals/gsec/4206.php
- [KLG03] Klevinsky T.J., Laliberte S., Gupt A.: *Hack I.T. Testy bezpieczeństwa danych*, Helion 2003.
- [Kohl04] Kohli K.: *Stealing passwords via browser refresh*, Paladion 2004.
http://www.paladion.net/technical_papers.php
- [LaSt06] Lach Ł., Stanowski M.: *Kryptografia w PHP*, PHP Solutions nr 4/2006, s.34-39.
- [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
- [Pipk02] Pipkin D.L.: *Bezpieczeństwo informacji. Ochrona globalnego przedsiębiorstwa*, Wydawnictwa Naukowo-Techniczne 2002.
- [ScHe05] Schwaha P., Heinzl R.: *Zagrożenia związane ze stosowaniem algorytmu MD5*, Hackin9 nr 1/2005.
- [Schl04] Schlossnagle G.: *PHP. Zaawansowane programowanie. Vademecum profesjonalisty*, Helion 2004.
- [Schn02] Schneider B.: *Kryptografia dla praktyków : protokoły, algorytmy i programy źródłowe w języku C*, WNT, Warszawa 2002.
- [SGT05] Szmít M., Gusta M., Tomaszewski M.: *101 zabezpieczeń przed atakami w sieci*

komputerowej, Helion 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/>