Bezpieczeństwo danych osobowych w serwisach internetowych

Bezpieczeństwo danych osobowych w serwisach internetowych

Wstęp

Czy aby na pewno nasze dane osobowe są w sieci bezpieczne? Często słyszymy o tym, że jakaś strona internetowa (portal, sklep, forum, etc) padła ofiarą ataku, w którego wyniku (zazwyczaj) zostały wykradzione dane z bazy danych, np. informacje o użytkownikach. Dlaczego tak się dzieje? Powodów jest kilka: błędy wynikające z przeoczenia programisty, nieaktualizowanie oprogramowania, z którego korzysta strona i inne. W aktualizacjach oprogramowania zwykle są załączane łatki do znanych błędów we wcześniejszych wersjach. Różni ludzie (dobrzy lub źli) szukają błędów, zazwyczaj licząc na zyski – sprzedaż wykradzionych danych osobom trzecim, lub np. nagroda za znalezienie buga[1] (w chwili pisania tego tekstu podstawowa stawka Google za zgłoszenie luki to 500 USD). Niestety, niektórzy administratorzy stron po poinformowaniu ich o błędzie wychodzą z założenia, że znalazca miał złe zamiary i grożą mu sądem. Na szczęście większość luk zostaje załatana niedługo po jej zgłoszeniu.

Nota prawna

Podstawowymi przepisami prawnymi, które definiują prawne zasady ochrony danych osobowych, są:

- Konstytucja Rzeczpospolitej Polskiej,
- Ustawa z dnia 29 sierpnia1997r. o ochronie danych osobowych (t.j. Dz.U. z 2002r. Nr 101, poz. 926 i Nr 153, poz.1271 oraz z 2004r. Nr 25, poz 219 i Nr 33, poz.285),
- Rozporządzenie MSWiA z dnia 29 kwietnia 2004r. (t.j. Dz.U. z 2004r. Nr. 100, poz 1024),
- Dyrektywa Parlamentu Europejskiego i Rady 95/46/WE z dnia 24 października 1995r. w sprawie ochrony osób fizycznych w zakresie przetwarzania danych osobowych oraz swobodnego przepływu danych,
- Konwencja 108 Rady Europy sporządzona w dniu 28 stycznia 1981r. o ochronie osób w związku z automatycznym przetwarzaniem danych osobowych.

1 **Bug** (ang. *błąd*) – błąd w działaniu aplikacji lub strony www.

Błędy na stronach internetowych

Wyróżniamy wiele rodzajów błędów, najpopularniejsze z nich to:

1. SQL[2] Injection
2. XSS (Cross Site Scripting)
3. LFI (*Local File Inclusion*) oraz RFI (*Remote File Inclusion*)
4. CSRF (*Cross Site Request Forgery*)

Informacje tu zawarte będą dotyczyć także sposobów zabezpieczenia się przed tymi błędami w aplikacjach napisanych w PHP[3].

SQL Injection

Co to jest?

SQL Injection – są to błędy polegające na możliwości zmiany wykonywanego zapytania w celu wyciągnięcia z bazy danych określonych informacji (np. informacji o użytkownikach) lub usunięciu danych. Błąd bardzo często spotykany, choć dość łatwo się przed nim zabezpieczyć. Wystarczy sprawdzać poprawność danych pochodzących od użytkownika. Rozpatrzmy taki kod wybierający dane określonego użytkownika z bazy danych (PHP):

Listing 1.

```php
<?php
  /* Dane wejściowe:
   * $_GET['id']; → id użytkownika, który ma zostać wyświetlony
   */
  $db  = mysql_connect('localhost', 'user', 'password');
  $sql = mysql_query("SELECT * FROM `users` WHERE `id` = ".$_GET['id'].";");
  /*
   * Kod wyświetlający użytkownika...
   */
  mysql_close($db);
```

Kod nie wygląda na skomplikowany, ale jest w nim wspomniany błąd typu SQL Injection – ID podawanego użytkownika nie jest sprawdzane, tylko bezpośrednio wklejone do zapytania. Ktoś się może zapytać: co z tego? W tym problem, że jeśli wywołanie strony będzie wyglądało tak:

```
http://example.com/user.php?id=1
```

Strona się wykona tak, jak miał to na myśli programista – zapytanie do bazy danych będzie wyglądało w ten sposób:

```
"SELECT * FROM `users` WHERE `id`=1"
```

Dla odmiany, jeśli strona zostanie wywołana w taki sposób:

```
http://example.com/user.php?id=1+OR+1=1
```

Zapytanie do bazy danych drastycznie się zmieni i będzie wyglądało w ten sposób:

```
"SELECT * FROM `users` WHERE `id`=1+OR+1=1"
```

W tym wypadku zamiast wybrania danych tylko jednego użytkownika, zostaną wybrane dane

2 **SQL** (ang *Structured Query Language*) – strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych.

3 **PHP** – skryptowy język programowania zaprojektowany do generowania stron internetowych w czasie rzeczywistym.

wszystkich użytkowników (warunek **1=1** będzie zawsze prawdziwy). Dzięki temu złośliwy użytkownik może, teoretycznie, wyciągnąć z bazy danych informacje o wszystkich użytkownikach. Dlaczego piszę „teoretycznie”? Ponieważ nie zawsze serwer, z którego korzysta strona, jest w stanie wytrzymać pobranie takiej ilości danych i ta próba może się zakończyć atakiem typu DoS[4].

Jak temu zapobiec?

Zabezpieczenie się przed tego typu atakami jest stosunkowo proste – wystarczy sprawdzać poprawność danych wprowadzanych przez użytkowników i/lub usuwać wrażliwe dane. Można to zrobić na kilka sposobów:

- W naszym wypadku wystarczyłoby rzutowanie zmiennej na typ int – w takim wypadku wszystkie dane po wystąpieniu znaku niebędącego cyfrą zostaną usunięte
- Stosowanie odpowiednich funkcji, np.: **mysql_real_escape_string()** lub **addslashes()**
- Stosowanie bezpieczniejszych narzędzi podczas korzystania z baz danych, które mają wbudowane funkcje zwiększające bezpieczeństwo, np.: PDO, mysqli.

XSS

Co to jest?

Są to błędy polegające na dołączeniu do kodu strony (np. w artykule, komentarzu itp.), kodu JavaScript, dzięki któremu osoba atakująca może przejąć konto niczego nieświadomej ofiary. Dzięki odpowiedniej treści wiadomości można np. przechwycić ciasteczko użytkownika, w którym jest zapisany identyfikator sesji. Po przechwyceniu tych danych atakujący, z dość dużym prawdopodobieństwem, może mieć możliwość nieautoryzowanego dostępu do konta ofiary.

Jak się przed tym zabezpieczyć?

Na szczęście użytkownik samemu może podjąć kroki mające celu zminimalizowanie skutków tego typu ataków. Powinno się **zawsze** wylogowywać po skończeniu korzystania z danej strony oraz nie korzystać z opcji zapamiętania logowania. Oczywiście, jest ona bardzo wygodna – nie trzeba się logować przy każdej wizycie, ale atakujący ma znacznie więcej czasu na korzystanie z konta (o ile nie uda mu się poznać i/lub zmienić wcześniej hasła dostępu). Dobrym pomysłem byłoby także wyłączenie obsługi JavaScript w przeglądarce, ale jest to niestety dość uciążliwe – większość dzisiejszych stron internetowych korzysta ze skryptów w tym języku. Natomiast administratorzy stron powinni tak jak w wypadku błędów typu SQL Injection sprawdzać wprowadzane przez użytkowników dane. Najprościej jest użyć odpowiednich funkcji usuwających lub zmieniających na bezpieczne dane wejściowe, np. **htmlspecialchars()**, która zamienia znaki specjalne (które mogą spowodować błąd) na odpowiadające im odwołania znakowe[5].

4 **DoS** (ang. *Denial of Service – odmowa usługi*) – atak na system komputerowy lub usługę sieciową w celu uniemożliwienia działania. Polega na przeciążeniu systemu/łącza, przez co przestaje on odpowiadać.

5 **Odwołania znakowe** – zestaw kodów postaci &kod;. Służą do jednoznacznego zapisu różnych znaków, gdy niemożliwe jest ich bezpośrednie wpisanie. Odwołania znakowe mają dwie role. Jedną z nich, dziś już mniej istotną dzięki możliwości używania UTF-8, jest kodowanie znaków spoza zestawu kodowego używanego przez stronę. Drugą jest używanie znaków, które mają specjalne znaczenie w składni SGML/XML. Są to: & (&) oraz < (<), a także > (>), " (").

LFI oraz RFI

O błędach LFI oraz RFI

LFI to skrót od angielskiego ***Local*** *File Inclusion (dołączenie lokalnego pliku)*, natomiast RFI, to skrót od ***Remote*** *File Inclusion (dołączenie zdalnego pliku)*. Oba błędy różnią się tylko jedną rzeczą – błąd LFI pozwala na dołączenie do aktualnej strony dowolnego pliku znajdującego się na serwerze, z którego korzysta strona, natomiast błąd RFI pozwala na dołączenie dowolnego pliku, niezależnie od tego, czy znajduje się on na aktualnie używanym serwerze, czy nie. Różnica może się wydawać niewielka, ale taka nie jest. Po pierwsze, błędy RFI występują znacznie rzadziej, głównie dlatego, że znakomita większość serwerów stron uniemożliwia dołączenie pliku, który nie znajduje się na serwerze. Dodatkowo błędy RFI pozwalają na dołączenie tzw. phpshell'a, który może umożliwić dostęp do plików i bazy danych na serwerze w znacznie wygodniejszy sposób. Dzięki temu można zdobyć np. dane użytkowników.

Jak się przed tym zabezpieczyć?

Przede wszystkim należy wymusić katalog, z którego ma być dołączany plik, oraz określone rozszerzenie dołączanych plików, np.: każdy pik **musi** mieć rozszerzenie **.php*:

Listing 2.

```
<?php
   /* Dane wejściowe:
    * $_GET['page']; → plik w katalogu ./include/
    */
   include './include/'.$_GET['id'].'.php';
```

Ale trzeba uważać: są sposoby na ominięcie tego zabezpieczenia, jak choćby wstawienie tzw. *null byte'u* (o kodzie ASCII *%00*), który „kończy” tekst – rozszerzenie jest „ignorowane”. Spójrzmy na na następny listing:

Listing 3.

```
<?php
   /* Dane wejściowe:
    * $_GET['page']; → plik w katalogu ./include/
    * $_GET['page'] = "../../../../../etc/passwd"; → plik z danymi użytkowników
   serwera (ścieżka poprawna dla Fedory, w systemach z rodziny Linux lub Windows może
   się różnić)
    */
   include './include/'.$_GET['id'].'.php';
   // Serwer będzie to widział jako:
   include './include/../../../../../etc/passwd%00.php';
   // Czyli:
   include '/etc/passwd';
```

Przy wykonaniu tego ataku powinniśmy zobaczyć zawartość pliku */etc/passwd*. W tym wypadku, podobnie jak w przypadku błędów SQL Injection, czy XSS także należy sprawdzać poprawność wprowadzanych danych. Tu nam na pomoc znowu przyjdzie funkcja **htmlspecialchars()**.

CSRF

Co to jest?

CSRF oznacza skrót od angielskiego *Cross Site Request Forgery*; jest to atak na serwis internetowy, mający na celu wysłanie nieautoryzowanego zapytania do serwera, tzn.

autoryzowanego nie przez atakującego, ale przez niczego nieświadomą ofiarę. Ten typ ataku nie jest wymierzony bezpośrednio w stronę www, raczej chodzi o to, żeby wykorzystać uprawnienia ofiary do wykonania akcji, która w innym wypadku wymagałaby jej zgody.

Atak często mylony z *Cross Site Scripting (XSS)*, z którym często jest równocześnie wykorzystywany.

Jak temu zapobiec?

Przede wszystkim należy (tak jak w wypadku ataków XSS) wylogowywać się po skończeniu korzystania z serwisu oraz unikać opcji zapamiętania logowania – znacznie to zmniejszy szansę wykonania na nas tego ataku. Poza tym będąc gdzieś zalogowanym, należy unikać korzystania z innych stron – atakujący może ukryć żądanie do serwera nie tylko w odsyłaczu, na który trzeba kliknąć, ale także w obrazku lub ramce, którą wystarczy wyświetlić. W tym wypadku użytkownik może być całkowicie nieświadomy tego, że jest ofiarą, gdyż co najwyżej zobaczy znaczek błędnego obrazka lub (w wypadku ramki) nic nie zobaczy.

Natomiast administratorzy mogą podjąć następujące środki:

- wprowadzenie kodów jednorazowych wysyłanych do użytkownika, uniemożliwiają spreparowanie zapytania. Używane zazwyczaj w bankowości internetowej, gdyż są dość trudne do zrealizowania,
- im większe konsekwencje niesie ze sobą korzystanie ze strony, tym krótszy powinien być czas możliwej nieaktywności; niestety, może to być dość uciążliwe dla użytkowników przez konieczność logowania za każdym razem,
- dodanie do każdego formularza ukrytego pola z liczbą pseudolosową i sprawdzanie, czy przesłana wartość zgadza się z tą zapisaną na serwerze, co znacznie utrudnia spreparowanie ataku.

Podsumowanie

Wszystkie typy błędów są znajdowane na stronach internetowych, niezależnie od tego, czy należy ona do osoby prywatnej, czy do wielkiej korporacji. Najczęściej znajdowane są błędy SQL Injection oraz XSS. Często występowanie tych błędów jest wynikiem lenistwa lub niewiedzy programistów, którym nie chce się lub nie potrafią prawidłowo zabezpieczyć strony. Dość szumnym przykładem były strony internetowe, które miały rzekomo usuwać platformę mikroblogową „śledzik” z serwisu nk.pl. Tak naprawdę po kliknięciu przycisku na stronie zostawał dodany wpis na w/w mikroblogu, przez co znajomi otrzymywali komunikat o tym, że ktoś „usunął śledzika” i poleca tę stronę. Zachęceni tym znajomi robili to samo i napędzali interes. Twórcy strony zarabiali na wyświetlaniu reklam i, nawet jeśli została ona szybko zamknięta, mogli oni sporo zarobić. Jest to wręcz podręcznikowy przykład wykorzystania błędu CSRF połączonego z atakiem socjotechnicznym[6]. Poza tym od czasu do czasu słychać o tym, że w wyniku jakiegoś błędu zostają wykradzione dane użytkowników z innych dużych portali, takich jak *onet.pl, wp.pl, filmweb,.pl* itp.

6 **Socjotechnika** – ogół metod i działań służących do uzyskania pożądanego zachowania się jednostek i grup ludzkich. Dokładniej zostanie opisana w dalszej części.

Sposoby przechowywania danych przez strony internetowe

Wstęp, czyli jak strony przechowują nasze dane

Dane na serwerach www są zazwyczaj zbyt słabo zabezpieczone. Po pierwsze często administratorzy i/lub użytkownicy posiadają zbyt słabe hasła chroniące dostępu do konta. Co gorsza, często hasła dostępu do strony są takie same jak hasło do bazy danych, FTP[7] lub panelu administracyjnego serwera. Często, w wyniku wykorzystania innych błędów atakujący mogą przejąć hasła do kont użytkowników lub administratorów. Na szczęście większość witryn przechowuje na serwerze hasła w formie zaszyfrowanej, czyli tzw. haszy. Oznacza to, że w przeciwieństwie do typowego szyfrowania operacja ta nie jest odwracalna. W momencie logowania do serwisu podane przez użytkownika hasło jest haszowane i porównywane z tym zapisanym w bazie. Jeśli dane podane przez użytkownika są zgodne z tymi na serwerze, użytkownik zostaje zalogowany. Czasami, choć dość rzadko, poprzez błędy na stronie, opisane wcześniej, atakujący może uzyskać nieautoryzowany dostęp.

Niestety, zdarzają się odstępstwa od szyfrowania haseł i, co gorsza, zdarzają się nie tylko w wypadku małych stron, ale także tych dużych. Dobrym przykładem może być Allegro oraz Gadu-Gadu, które trzymają hasła w tzw. plaintekście, czyli w formie niezaszyfrowanej – po ich zdobyciu agresor może bez problemu zalogować się na konto ofiary. Nie wiadomo, dlaczego te firmy tak postępują. Może wierzą w absolutne bezpieczeństwo swych serwerów i aplikacji? Tylko dlaczego? Przecież nie od dziś wiadomo, że absolutne bezpieczeństwo w odniesieniu do systemów komputerowych to mit. To, w jaki sposób z czegoś korzystamy to kompromis między użytecznością i wygodą a bezpieczeństwem. Całkowicie bezpieczny system komputerowy to taki, który nie jest podłączony do żadnej sieci, nigdy nie będzie włączany i nikt nie będzie miał do niego dostępu. Ale czy będzie on użyteczny? Raczej nie. Pozostałe dane, oprócz haseł, nie są szyfrowane, gdyż spowolniłoby to znacznie działanie strony, a poza tym te dane są zazwyczaj ogólnie dostępne. Najlepszym sposobem na zapewnienie bezpieczeństwa naszych danych jest podawanie ich tylko tyle, ile jest niezbędne, lub tyle ile chcemy komuś pokazać. Nie należy podawać na stronach internetowych danych, w nadziei, że nikt niepowołany ich nie zobaczy. To samo tyczy się zdjęć – zdarzały się przypadki, gdzie ktoś wrzucał do portalu społecznościowego zdjęcie, na którym był widoczny adres, lub był on wpisany w profilu, a później ta sama osoba informowała na tymże serwisie o tym, że jest na wakacjach i nikogo nie ma w domu. Równie dobrze można bezpośrednio zaprosić złodzieja do domu...

Sposoby łamania haseł

Jeśli hasła dostępu są haszowane, zdecydowanie to utrudnia, choć nie uniemożliwia, dostępu do konta. Agresor, po zdobyciu hasła w tej formie (np. poprzez atak SQL Injection), musi je najpierw złamać, aby móc zalogować się na konto. Jest na to wiele sposobów, najpopularniejsze to:

- wyszukiwanie w internetowych bazach danych,
- korzystanie z tęczowych tablic[8],
- ataki słownikowe
- ataki brute force

7 **FTP** (ang. *File Transfer Protocol – Protokół Transferu Plików*) – protokół typu klient-serwer, który umożliwia przesyłanie plików z serwera i na serwer poprzez sieć.

8 **Tęczowe tablice** – baza skrótów wykorzystywana w łamaniu haseł zakodowanych jednokierunkową funkcją skrótu (funkcję haszującą).

Internetowe wyszukiwarki

Najszybszym sposobem jest pierwszy, ale niestety daje zwykle słabe efekty, przede wszystkim, jeśli podczas haszowania zostały użyte sole[9].

Tęczowe tablice

Podobnie jest w przypadku zastosowania tęczowych tablic – w momencie, gdy zostaną zastosowane sole stają się one bezużyteczne – trzeba by było mieć osobą bazę dla każdej soli, co nie jest możliwe – jedna baza zajmuje od kilku do kilkuset gigabajtów, a przy zastosowaniu 3 znaków soli (małe, wielkie litery, cyfry) ilość możliwych soli wynosi prawie ćwierć miliona. Przy zastosowaniu 4 znaków soli składającej się z tych samych znaków ilość możliwych kombinacji wzrasta ok. 60 razy. Wygenerowanie i przechowywanie tak wielu danych byłoby zbyt czasochłonne i kosztowne.

Ataki słownikowe

W tym wypadku o wiele lepiej wypadają ataki słownikowe – polegają one na podaniu do stosowanej w serwisie funkcji haszującej „wyrazu" znajdującego się w słowniku i porównaniu z haszem z bazy danych. W zależności od wielkości słownika, szansa powodzenia może wynosić nawet 90%. Zwykle podczas łamania hasła tym sposobem istnieje możliwość dołączenia soli, która także jest zapisywana w bazie danych (musi zostać użyta podczas sprawdzania hasła). Z powodu dużej skuteczności tego typu ataków (jest to kompromis między skutecznością a szybkością) zaleca się stosowanie haseł niezawierających wyrazów ze słownika (tu nie ma różnicy, czy hasłem będzie „*konstantynopolitańczyk*" czy „*gwiazda*" – oba hasła zostaną złamane w podobnym czasie) oraz zawierających cyfry i znaki specjalne. Niestety, część programów łamiących hasła tą metodą dodaje do wyrazów ze słownika na początku i końcu cyfry, aby zwiększyć szansę na powodzenie ataku.

Atak siłowy (brute force)

Długość hasła, jeśli nie ma go w internetowych bazach danych, tęczowych tablicach, czy słownikach, ma wpływ przy łamaniu go metodą brute force – czyli przy ataku siłowym. Polega on na podaniu do funkcji haszującej wszystkich możliwych kombinacji. Wymaga on bardzo dobrego sprzętu, aby zmniejszyć czas łamania (dobrze się tu sprawują karty graficzne (GPU) – są wielokrotnie wydajniejsze od tradycyjnego procesora). Ten atak, teoretycznie, daje szansę na złamanie hasła blisko 100% – im dłuższe i bardziej skomplikowane hasło tym więcej czasu będzie potrzeba na jego odzyskanie, a więc w praktyce zmaleje szansa na to, że rzeczywiście zostanie ono złamane. W tym wypadku także dobrym pomysłem jest równoczesne stosowanie małych, dużych liter, cyfr oraz znaków specjalnych. Znacznie to zwiększy bezpieczeństwo naszego hasła, a co za tym idzie – zmniejszy prawdopodobieństwo przejęcia konta przez osoby trzecie.

Przykład

Alicja ma niesłownikowe hasło składające się tylko z małych i dużych liter oraz cyfr, np.: *jH53aSh*, a hasła w serwisie są haszowane przy użyciu algorytmu md5[10].

Istnieje bardzo małe prawdopodobieństwo (niewiele większe od zera), że agresor da radę znaleźć to hasło w internetowych wyszukiwarkach lub złamie je przy użyciu tęczowych tablic. Natomiast, korzystając z GPU ze średniej półki cenowej (np. ATI Radeon 5770) jest w stanie złamać to hasło w ciągu godziny. Przy dodaniu jeszcze jednego znaku z tego samego zakresu

9 **Sól** – tekst dodawany do haszowanego hasła, w celu utrudnienia jego odszyfrowania.

10 **MD5** – funkcja jednokierunkowa, nadal jedna z najczęściej stosowanych do haszowania haseł w serwisach, dość szybka i niezbyt bezpieczna, nie zaleca się jej stosowania.

(tzn. małej litery lub cyfry) czas łamania hasła zwiększa się ponad trzydziestokrotnie. Przy kartach graficznych z najwyższej półki (jak choćby Nvidia GTX 580) ten czas można skrócić nawet kilkukrotnie, natomiast przy zastosowaniu CPU, czyli tradycyjnego procesora (nawet tego z najwyższej półki), ten czas znacznie by się wydłużył.

Sposoby zwiększenia bezpieczeństwa haseł

Rady dla użytkowników

Jak już wspominałem wcześniej, użytkownicy powinni stosować skomplikowane hasła, składające się z małych i dużych liter, cyfr oraz znaków specjalnych. Im będzie ono dłuższe i bardziej skomplikowane, tym mniejsza szansa jego utraty (o ile jest haszowane). Dodatkowo powinno się stosować osobne hasło dla każdego serwisu. Dzięki temu utrata konta w jednym serwisie nie będzie oznaczać utraty kont gdzie indziej. Tutaj pojawia się problem, a mianowicie: jak zapamiętać te wszystkie hasła? Niestety, tu znowu nie jest tak łatwo – trzeba osiągnąć kompromis pomiędzy bezpieczeństwem a wygodą. Poza tym nie powinno się nikomu podawać haseł – prośba o nie to prawie zawsze próba wyłudzenia, administratorom nie jest ono do niczego potrzebne – mają więcej możliwości niż przysłowiowy Kowalski. Należy jeszcze wspomnieć, że powinno się okresowo zmieniać hasła.

Rady dla administratorów

Przede wszystkim, bezwzględnie, należy haszować **wszystkie** hasła użytkowników. Bez tego nie można zapewnić użytkownikom należytego bezpieczeństwa – ich starania przy używaniu skomplikowanych haseł nie mają w takim wypadku sensu. Poza tym należy stosować bezpieczne funkcje skrótu, jak choćby te z rodziny SHA-2 (SHA-224, SHA-256, SHA-384 SHA-512) – zwiększają one bezpieczeństwo haseł, gdyż trudniej w nich wywołać kolizje[11]. Dodatkowo te algorytmy działają zdecydowanie wolniej, niż np. *md5*, przez co zdecydowanie wydłużają łamanie haseł metodą słownikową lub siłową. Dodawanie soli jeszcze bardziej zwiększa bezpieczeństwo, ale tylko pod warunkiem, że atakujący ich nie pozna. Sole są zazwyczaj krótkim tekstem dodawanym do haszowanego hasła, mającym na celu utrudnienie jego odtworzenia. W tym wypadku zasada jest analogiczna do haseł – im dłuższa i bardziej skomplikowana sól, tym większe bezpieczeństwo. Najlepiej, aby sole były unikatowe dla każdego hasła oraz losowe – żeby uniemożliwić lub zdecydowanie utrudnić ich odtworzenie, a co za tym idzie – także odtworzenie oryginalnego hasła.

Poza tym, dobrze jest zastosować jakąś politykę dotyczącą bezpieczeństwa haseł, np. hasło musi mieć co najmniej 8 znaków i zawierać co najmniej jeden znak specjalny i/lub cyfrę. Należy jednak uważać – niedawno osobiście spotkałem się z niemożnością użycia pewnego hasła (ponad 10 znaków, w tym 3 znaki specjalne), gdyż... program stwierdził, że moje hasło jest za słabe, bo powinno:

- mieć minimum 6 znaków,
- zawierać minimum jedną cyfrę.

Przypominam, że moje hasło było o 5 znaków dłuższe oraz zawierało znaki specjalne (bardziej zwiększające bezpieczeństwo hasła niż cyfry[12]). Niestety, ale byłem zmuszony do zmiany hasła na słabsze...

11 **Kolizja** – tekst podany do funkcji skrótu, który tworzy taki sam hasz, jak szukany. Niekoniecznie taka sama jak oryginalne hasło.

12 Cyfr jest zdecydowanie mniej niż znaków specjalnych, więc podczas próby złamania hasła jest mniej możliwości do sprawdzenia (o tym więcej pod koniec tego tekstu).

Inżynieria społeczna, czyli słów kilka o socjotechnice

W odniesieniu do informatyki, bo o tym tutaj mowa, socjotechnika to zestaw metod mających na celu uzyskanie niejawnych informacji od użytkownika. Agresor zwykle wykorzystuje niewiedzę lub łatwowierność użytkowników. Czasami, chociaż rzadko, wystarczy poprosić użytkownika o jakieś dane podając się za administratora lub dobrego znajomego i powiedzieć, że potrzebuje się do czegoś określonych informacji, np.: coś u nas nie działa. Potrzebujemy tych danych, żeby sprawdzić, czy to problem z naszym kontem, czy to jednak nie to. Socjotechnika jest bardzo rozległym tematem, wystarczającym do napisania wielu książek, dlatego tutaj zostanie jedynie wspomniana.

Phishing

Jednym z rodzajów ataków wykorzystujących naiwność i nierozważność użytkowników są strony phishing-owe – są to podróbki oryginalnych stron o takim samym wyglądzie i, czasami działaniu, które mają na celu uzyskanie od użytkownika np. loginu i hasła. Ataki bardzo często spotykane, z powodu prostoty ich przeprowadzenia, także w innych odmianach – zamiast strony www rozsyłane maile lub wiadomości w komunikatorach internetowych. W większości przypadków (ponad 90%) wystarczy zwracać uwagę na adres aktualnej strony[13] i jeśli strona to udostępnia – certyfikat bezpieczeństwa.

Przykład

Alicja dostała link do pewnej strony, podpisany jako jej ulubiony serwis społecznościowy, więc kliknęła nie patrząc na adres **(!)** i została przeniesiona na stronę *http://example.com/*. Zobaczyła, że rzeczywiście wygląd jest taki sam, więc wypełniła formularz i została przeniesiona już na prawdziwą stronę. Niczego nieświadoma, właśnie straciła login i hasło, a co za tym idzie – także konto.

13 Są sposoby na wyświetlenie fałszywej strony pod prawdziwym adresem – tzw. DNS Poisoning.

Testy, czyli skąd te dane?

Łamanie haseł – przykład

Parametry środowiska testowego 1

Test przeprowadzono na komputerze o parametrach:

- CPU: AMD Athlon X2 240 2,8GHz
- GPU: Powercolor ATI Radeon 5770 1024MB DDR5/128bit
- RAM: 2GB DDR2

Parametry środowiska testowego 2

Test przeprowadzono na komputerze o parametrach:

- CPU: Intel Core i7 980X Extreme Edition 3,33Ghz
- GPU: dual PALIT GeForce GTX 580 1536MB DDR5/384bit
- RAM: Corsair DDR3 24GB 1600MHz

Do odzyskiwania hasła został użyty darmowy program *IGHASHGPU* (dostępny na stronie autora). Może to nie były warunki laboratoryjne, ale zawsze dają jakieś wyniki poglądowe ;).

Wyniki odzyskiwania hasła z md5

Hasło podane w przykładzie (*jH53aSh*) zostało zakodowane przez funkcję skrótu *MD5*, co dało hasz: *072c80a51637fd4fd27f5bc2dd31fbd2*. Podczas przeprowadzania testu na komputerze nie były wykonywane żadne czynności i nie działał żaden inny program. Wyniki:

1. środowisko testowe 1
   - program sprawdzał 1 223 809 018 haseł w ciągu sekundy
   - szukanie trwało 21min 8,39s
   - w tym czasie zostały sprawdzone 1 551 942 811 648 możliwości
   - zostało sprawdzone 42,42% możliwości z zakresu 4-7 znaków (małe, duże litery, cyfry)
2. środowisko testowe 2
   - program sprawdzał 4 346 230 964 haseł w ciągu sekundy (o ok. 260% szybciej niż na pierwszym stanowisku)
   - szukanie trwało 5min 58,17s (o 15min 10,22s szybciej niż na pierwszym stanowisku)
   - w tym czasie zostały sprawdzone 1 553 064 787 968 możliwości
   - zostało sprawdzone 42,46% możliwości z zakresu 4-7 znaków (małe, duże litery, cyfry)

Wyniki odzyskiwania hasła z sha1

Dla porównania wyniki odzyskiwania tego samego hasła (*jH53aSh*) i w tych samych warunkach, które tym razem zostało zakodowane przez funkcję skrótu *sha1*, co dało hasz: *51f9dac26d640a0265b566dd5b158b500ead75b8*:

1. środowisko testowe 1
    - program sprawdzał 397 970 581 haseł w ciągu sekundy (więc o ok. 68% wolniej w porównaniu z md5 na tym samym stanowisku testowym)
    - szukanie trwało 1h 4min 53,39s (43min 45s dłużej w porównaniu z md5 na tym samym stanowisku testowym)
    - w tym czasie zostały sprawdzone 1 551 724 707 840 możliwości
    - zostało sprawdzone 42,42% możliwości z zakresu 4-7 znaków (małe, duże litery, cyfry)
2. środowisko testowe 2
    - program sprawdzał 1 433 180 824 haseł w ciągu sekundy (więc o ok. 260% szybciej w porównaniu z pierwszym środowiskiem testowym)
    - szukanie trwało 17min 3,58s (47min 1s krócej w porównaniu z pierwszym środowiskiem testowym)
    - w tym czasie zostały sprawdzone 1 551 829 565 440 możliwości (nie pytajcie, dlaczego więcej niż w poprzednim wypadku...)
    - zostało sprawdzone 42,42% możliwości z zakresu 4-7 znaków (małe, duże litery, cyfry)

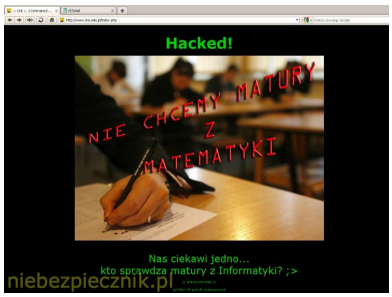
Przyczyny takiego wyniku

Hasło zostało znalezione względnie szybko (przewidywany czas: ok. 1h dla *md5* oraz prawie 2,5h dla *sha1*), ponieważ zaczynało się od litery bliskiej początkowi alfabetu, a wszystkie możliwości były testowane rosnąco alfabetycznie – najpierw cyfry, później małe i na końcu wielkie litery.

UWAGA!

NIE należy się sugerować tym, że w tym akurat programie hasła są testowane rosnąco alfabetycznie, w innym mogą być testowane np. w odwrotnej kolejności. Poza tym, jeśli hasło jest słownikowe to i tak na nic się nie zda...

Ciekawostki



1. Po tegorocznej (2010r.) maturze, absolwenci (?) postanowili to uczcić i... podmienili stronę indeksu Centralnej Komisji Egzaminacyjnej.
2. W połowie września bieżącego roku pewien polak, przestawiający się jako *porkythepig* postanowił zostawić niespodziankę na stronie amerykańskiej *Defence Logistic Agency* – mianowicie po wpisaniu na klawiaturze jego nicku na ekranie pojawiał się fragment filmu „Miś" (piosenka pt. "Jestem wesoły Romek").
3. Wspominana kilkukrotnie Alicja to nazwa fikcyjnej postaci, przyjęta jako imię jednej z osób występujących w przykładzie, w informatyce lub fizyce. Zazwyczaj prowadzi komunikację z Bobem, czyli drugą osobą. Jest to łatwiejsze do zrozumienia, głównie w bardziej skomplikowanych przykładach, gdzie występuje kilka osób, niż np. pisanie *Osoba A* i *Osoba B*.
4. domena *http://example.com/* została stworzona po to, aby można było w różnych przykładach podawać adres strony bez używania określeń typu: „*użytkownik A wchodzi na jakąś stronę*" – zdecydowanie łatwiej zrozumieć „*Alicja wchodzi na stronę http://example.com/*" – brzmi to i lepiej i bardziej profesjonalnie ;)

Bibliografia

- UODO Survival Kit
- http://webhosting.pl/Chcesz.zarobic.Szukaj.luk.w.Google.Chrome.1337.dolarow.czeka
- http://pl.wikipedia.org/wiki/SQL
- http://pl.wikipedia.org/wiki/PHP
- http://pl.wikipedia.org/wiki/DoS
- http://pl.wikipedia.org/wiki/Odwo%C5%82ania_znakowe_SGML
- http://pl.wikipedia.org/wiki/CSRF
- http://pl.wikipedia.org/wiki/In%C5%BCynieria_spo%C5%82eczna_%28informatyka%29
- http://pl.wikipedia.org/wiki/FTP
- http://www.socjotechnika.net/socjotechnika/wprowadzenie-do-socjotechniki/
- http://niebezpiecznik.pl/post/polak-zhackowal-amerykanska-strone-rzadowa/
- http://niebezpiecznik.pl/wp-content/uploads/2010/05/cke_hacked.jpg
- + wiedza własna