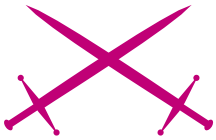


Sieci Zombie

Dawid Gołuński, Krzysztof Kułaj

stopień trudności

Sieci botnet są coraz bardziej rozpowszechnione. Z danych statystycznych wynika, że co 4 komputer domowy w sieci to Zombie. W jaki sposób się one rozprzestrzeniają, co potrafią i dlaczego nie można sobie z nimi poradzić - na te pytania Czytelnik uzyska odpowiedź po przeczytaniu tego artykułu.

Boty zaczęły rozprzestrzeniać się na masową skalę pod koniec 2003 roku - wykorzystując do tego celu lukę w interfejsie RPC, wykrytą parę miesięcy wcześniej. Luka ta stanowiła doskonałą okazję do zainfekowania ogromnej liczby komputerów przy małym nakładzie środków. Na powodzenie zmasowanych ataków złożyło się tu wiele czynników, m.in.: niska świadomość użytkowników, dostępność kodu *P-O-C* z optymalnym shellcode'm oraz dostępność kodów źródłowych botów (m.in. *Sd-bot, Spybot*). Od tego czasu oprogramowanie takie przeżywa rozkwit, doskonalone są techniki jego rozprzestrzeniania oraz ukrywania się w systemie.

Metody rozprzestrzeniania się

Jedną z podstawowych cech botów jest możliwość rozprzestrzeniania się. Techniki te ewoluowały od momentu powstania botów do dnia dzisiejszego. Początkowo boty rozprzestrzeniały się w sposób podobny do wirusów – były wysyłane jako załączniki e-mail. Treść listu była tworzona z użyciem technik inżynierii społecznej (*social engineering)* – tak, aby nakłonić użytkownika do otwarcia załącznika. Dzisiaj programy antywirusowe automatycznie skanują jednak załączniki i potrafią na podstawie sygnatur lub zachowania aplikacji wykryć znane boty. Innym sposobem ich rozprzestrzeniania się jest wykorzystywanie luk w aplikacjach sieciowych, najczęściej Microsoft Internet Explorer oraz Microsoft Outlook. Specjalnie utworzony exploit, instalujący bota, jest umieszczany na serwerze Web. Link do spreparowanej strony rozpowszechniany jest na wszelkie możliwe sposoby - poprzez komunikatory sieciowe, IRC, mail, grupy dyskusyjne czy też. fora internetowe. Gdy ofiara ak-

Z artykułu dowiesz się

- W jaki sposób działają boty (metody infekcji, ukrywanie się w systemie),
- Nauczysz się analizować skompilowany plik bota,
- Poznasz zagrożenia płynące z sieci typu botnet.

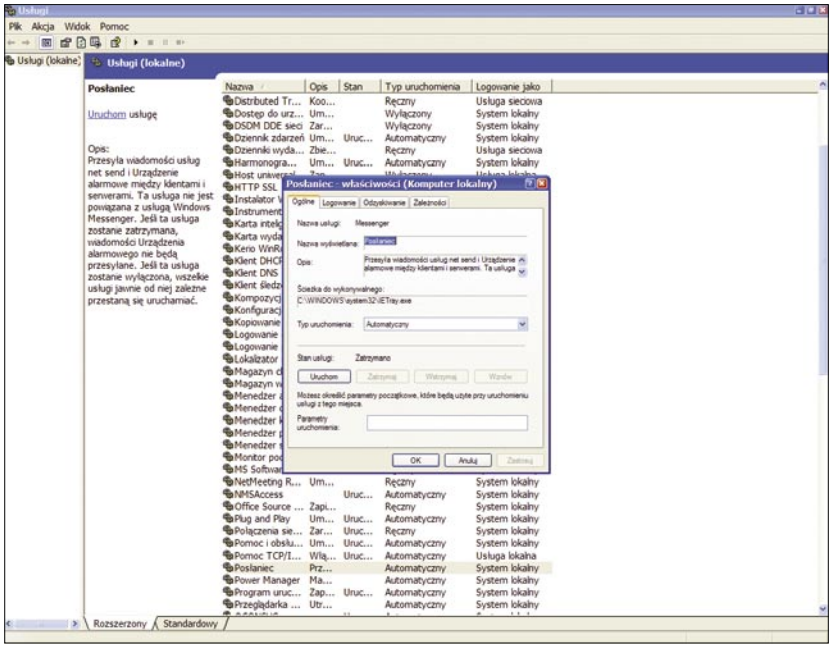
Co powinieneś wiedzieć

- znać podstawy C,
- znać podstawy assemblera,
- znać podstawy sieci TCP/IP.

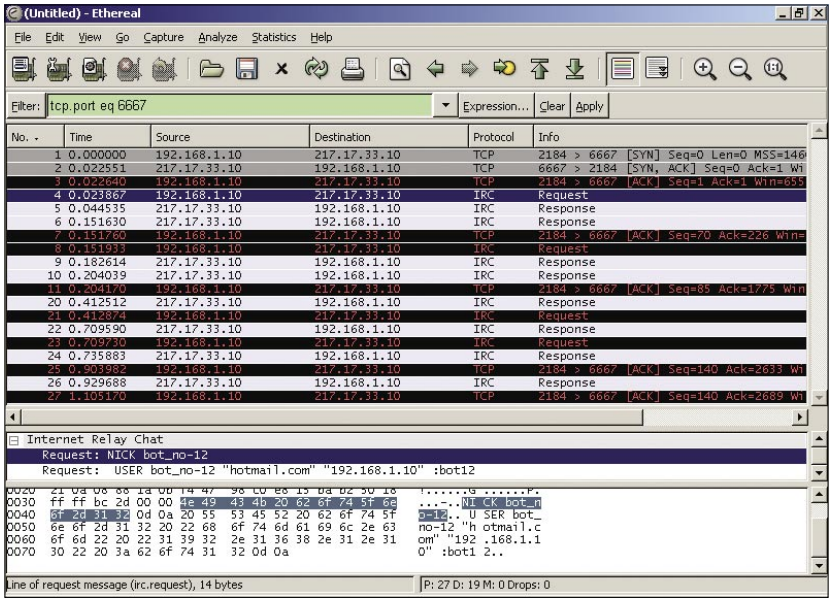
tywuje link, a używana przeglądarka jest podatna na błąd, bot zainstaluje się w systemie. Luki wykorzystywane do tego ataku są wybierane ze względu na takie czynniki, jak: ilość podatnego oprogramowania, popularność aplikacji, uniwersalny shellcode (najlepiej niezależny od wersji językowej Windows lub programu). Innym sposobem na infekcję użytkowników jest skanowanie sieci w poszukiwaniu niezaktualizowanych systemów. Najczęściej wykorzystywane luki to : SRVSVC (MS06-040), PNP (MS05-039) , ASN.1 (MS04-007), LSASS (MS04-011), DCOM (MS03-026) – (MS03-043), (MS03-049). (bardziej dociekliwi czytelnicy mogą zapoznać się ze szczegółami tych luk na stronach Microsoft, podając numery indeksowe zawarte w nawiasach). Publicznie dostępne boty, m.in. *RxBot*, *GT-Bot* posiadają zaimplementowane przeszukiwanie sieci oraz infekcję za pomocą wyżej wymienionych dziur. Ciekawym sposobem infekowania systemów jest wyszukiwanie komputerów i infekcja za pomocą *Netbios*. Schemat ataku wygląda następująco: agresor wyszukuje komputer z otwartym portem 139, następnie próbuje się z nim połączyć. Jeśli połączenie dojdzie do skutku z pustym hasłem, kopiuje się do wszystkich możliwych lokacji. Jeśli istnieje hasło, bot będzie próbował odgadnąć hasło, podstawiając najczęściej występujące słowa. Często wykorzystywane są dziury w standardowych lokalnych aplikacjach, np. Microsoft Word czy Microsoft PowerPoint. Atakujący wysyła odpowiednio spreparowany plik .doc lub .pps w załączniku do e-maila. Tym sposobem można wykorzystać niemalże każdą lukę występującą w lokalnym systemie. Jedynym ograniczeniem jest treść e-maila – musi być wystarczająco przekonywująca – co z reguły nie jest najtrudniejszym zadaniem, jeśli rozszerzenie załącznika nie jest plikiem wykonywalnym. Ten atak ma jedną wadę – nie może być on kierowany do wielu użytkowników, gdyż jeśli trafi do firm antywirusowych, jego sygnatury zostaną dodane do bazy wirusów i będzie wykrywalny, co znacząco zmniejszy możliwości rozprzestrzeniania się. Stosunkowo prostym w implementacji sposobem rozprzestrzeniania jest kopiowanie bota na komputerze ofiary, do folderów, z których korzystają programy P2P (np. *Emule, Kazaa*). Nazwy są najczęściej dobierane wg. schematów (crack-nazwa_gry.exe, key-gen-nazwa_gry.exe etc). Ta metoda jest skuteczna jedynie w przypadku dużych botnetów, ponieważ im więcej kopii tego samego pliku w sieci *P2P,* tym wyżej znajduje się on na liście wyszukiwania. Ciekawą techniką zdobywania botów, której nie można zaliczyć do infekcji są kradzieże botnetów. Ten temat zostanie poruszony w punkcie *Zdobywanie informacji o bocie*.

Instalacja i ukrywanie w systemie

Domyślnie wszystkie boty instalują się w kluczach startowych rejestru Windows. Metoda ta jest bardzo zawodna, ponieważ wszystkie programy antywirusowe oraz antyspywarowe w pierwszej kolejności sprawdzają domyślne lokalizacje. Nawet stan-



Rysunek 1. *Wywołanie bota, ukryte w usłudze Posłańca*



Rysunek 2. *Przechwycenie adresu serwera oraz nicka bota*

Listing 1. *Przełożenie funkcji decode() na język C*

```
char *decode(
      char *text, int param, char *where)
{
int i;
   for (i = 0;text[i] != 0; i++) {
      where[i] = text[i] - param - 3*i;
         }
      Where[i]=0;
   return where;
}
```

dardowe narzędzia Windows (np. *msconfig*) są w stanie wykryć obecność bota. Jednakże nie jest to jedyna metoda automatycznego uruchamiania bota. Istnieje szereg innych, mniej lub bardziej znanych metod. Np. dopisanie się do systemowych usług lub *dll injection*.

Możliwości botów

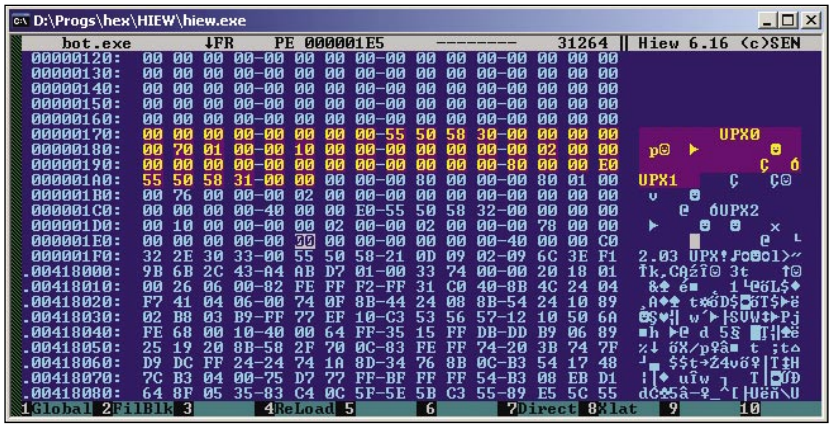
Pierwotnie boty były gromadzone w celach ataków *DOS* i *DDOS*. Najczęściej używany do tego celu jest *syn-flood* oraz *ping-flood*. Z czasem boty stały się coraz bardziej wyrafinowane. Wachlarz ich możliwości stał się dużo większy. Bardzo szybko została dodana możliwość skanowania naciskanych klawiszy. Dodatkowo, w źródle możemy określić nazwy okien, z których klawisze mają być logowane. Określając nazwę okna na:

- PuTTY– możemy uzyskać sporą ilość kont z dostępem do shella,
- nazwy serwisów aukcyjnych – hasła do kont umożliwiające zawieranie transakcji w imieniu danej osoby,
- nazwy banków – numery kart kredytowych, jak również hasła i loginy do kont bankowych.

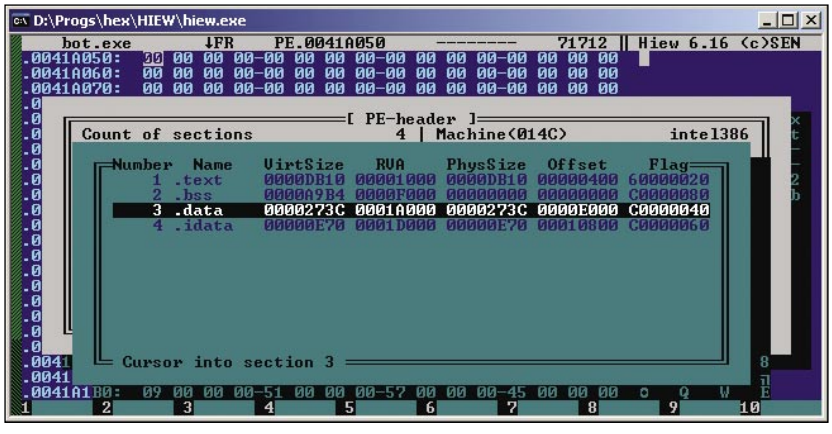
To tylko kilka przykładów, potwierdzających, że najwrażliwsze dane użytkownika są w zasięgu ręki posiadacza bonetu. Inne funkcje to np. możliwość klikania w linki internetowe. Funkcja ta jest używana do oszukiwania systemów zliczających kliknięcia w banery na stronach internetowych. Oznacza to, że osoba posiadająca bonet może w łatwy sposób wpływać na wyniki wszelkiego rodzaju sond, ankiet, głosowań itp. Jako że istnieją firmy płacące za klikanie w reklamy na stronach – takie działanie może doprowadzić do osiągnięcia korzyści finansowych. Boty często są wykorzystywane do wysyłania spamu, posiadają nawet własne silniki SMTP. Według statystyk, około 50% spamu wysyłanego w sieć pochodzi z zainfekowanych komputerów. Źródła botów potrafiących wysyłać masowo spam nie są raczej publicznie dostępne, ale nawet średniozaawansowanemu programiście rozszerzenie programu o takie funkcje nie powinno sprawić trudności.

Scenariusz przykładowego ataku na konto w banku internetowym

Jeśli użytkownik korzystający z usług banku posiada bota, w skrajnych przypadkach może stracić pieniądze na swoim koncie. Zaprezentowana technika może zostać dostosowana do każdego banku stosującego tego typu zabezpieczenia. Istnieją boty mające zaimplementowaną funkcję powiadamiania na kanale, jeśli na komputerze pojawią się okna z danym tytułem. W przypadku, gdy użytkownik wejdzie na stronę banku – wiadomość z zapisanymi klawiszami wciśniętymi w tym oknie (tj. login i hasło) zostanie przesłane do właściciela botnetu. W tym momencie może się on swobodnie logować na konto bankowe użytkownika. Jednak wszystkie kluczowe operacje na koncie są zabezpieczone hasłem jednorazowym, które zna jedynie właściciel konta oraz bank. Można spreparować stronę, do złudzenia przypomi-



Rysunek 3. *Bot spakowany UPXem*



Rysunek 4. *Sekcje dostępne w pliku bota*

nającą stronę banku. Posiadając bota na komputerze ofiary, jesteśmy w stanie przekierować użytkownika na podstawioną przez nas stronę. Gdy użytkownik -- będąc przekonanym, że kontaktuje się z właściwą stroną -- wpisze hasło jednorazowe, zamiast trafić do banku trafia do właściciela botów. W tym momencie, atakujący może zrobić wszystko z naszym kontem bankowym. Działanie to może wydawać się początkowo skomplikowane, jednak coraz więcej osób pada ofiarami tego typu przestępstw. Banki starają się walczyć z takimi praktykami, wprowadzając losowe wybieranie haseł jednorazowych z listy lub hasła przesyłane na telefonu komórkowe.

Zdobywanie informacji o bocie

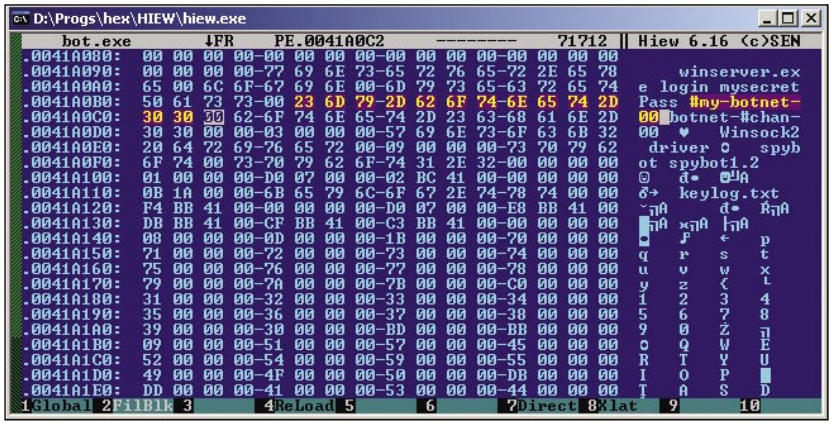
Przechwycony plik wykonywalny bota może wiele powiedzieć o docelowej sieci. Przeprowadzając odpowiednie działania, możemy poznać dokładną konfigurację bota, a w szczególności:

- Adres serwera IRC, z jakim następuje połączenie
- Nazwa kanału IRC, do którego dołącza bot
- Klucz dostępowy kanału IRC
- Dane potrzebne do uwierzytelnienia

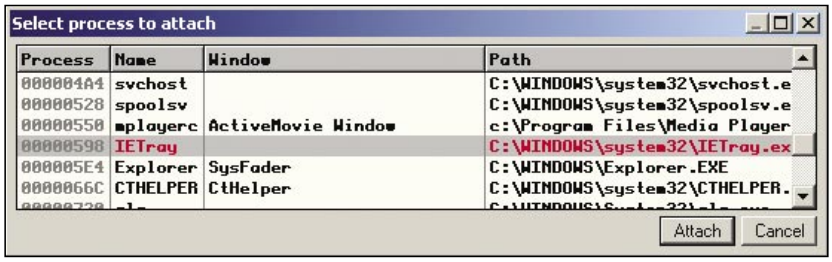
Te informacje mogą posłużyć do namierzenia atakującego, zniszczenia botnetu (znając dane potrzebne do uwierzytelnienia, jesteśmy w stanie zalogować się na każdym z zaatakowanych komputerów oraz odinstalować program bota), przejęcia botnetu (poprzez poinstruowanie komputerów, aby wgrały określoną przez nas wersję bota jednocześnie usuwając starą).

Podsłuchiwanie, czyli sniffing

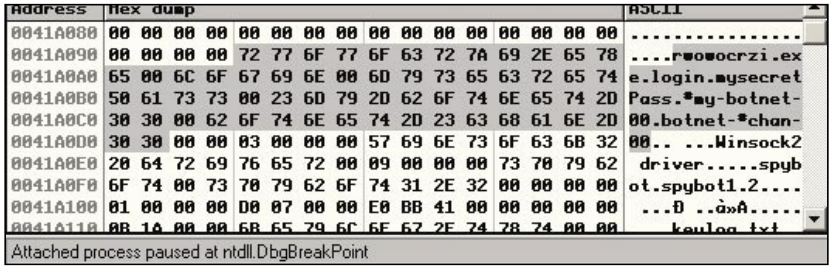
Najłatwiejszym, a zarazem najszybszym sposobem na ustalenie adresu docelowego serwera, nazwy kanału oraz klucza kanału, jest podsłuchanie sieciowych pakietów wychodzących z komputera, na którym znajduje się bot. W tym celu można posłużyć się popularnym snifferem o nazwie *Ethereal*. Po uruchomieniu sniffera, należy wybrać odpowiedni interfejs za pomocą okna `Capture->`*Interfaces*. Po kliknięciu przycisku *Capture* przy danym interfejsie, rozpocznie się nasłuchiwanie pakietów. Wystarczy teraz uruchomić plik wykonywalny bota i odczekać kilkanaście sekund. Po wciśnięciu przycisku *stop* w programie *Ethereal* nasłuchiwanie zostanie przerwane, a



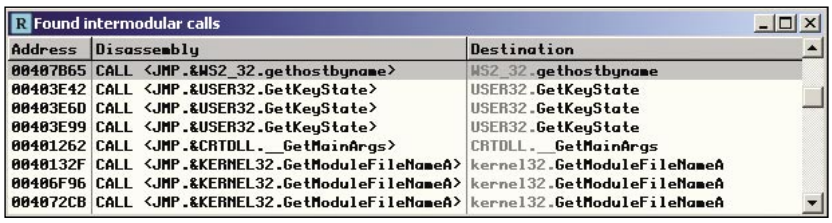
Rysunek 5. *Niezakodowane dane konfiguracyjne bota*



Rysunek 6. *Podłączanie pod proces bota*



Rysunek 7. *Dane widoczne po podłączeniu się pod proces bota*



Rysunek 8. *Lista odwołań do funkcji API*

W Sieci

- http://www.google.com/codesearch – wyszukiwarka źródeł ułatwiająca odnalezienie botów (RxBot, GT-bot Sd-Bot, Spy-Bot.)
- http://www.secretashell.com/codomain/peid/ – strona programu PEiD
- http://www.ollydbg.de/ – strona programu OllyDbg
- http://upx.sourceforge.net/ – strona programu UPX

przechwycone pakiety zostaną wyświetlone.

W jednym z pierwszych pakietów zobaczymy nawiązanie połączenia TCP pomiędzy botem a serwerem IRC. Uzyskujemy tym samym adres serwera oraz numer portu, na którym działa usługa IRC. Jeżeli podczas nasłuchiwania mieliśmy uruchomione również inne programy sieciowe – warto ustawić filtr, który usunie nieistotne dla nas pakiety. Jeżeli połączenie z serwerem zostało nawiązane na porcie 6667, można wówczas ustawić filtr, który pokaże jedynie pakiety odnoszące się do tego portu:

```
tcp.port eq 6667
```

Kolejne pakiety odzwierciedlają komunikację bota z usługą IRC. *Ethereal* oznacza polecenia wydawane przez bota jako *IRC Request.* Środkowe okno ukazuje treść danego polecenia. Jak określa specyfikacja protokołu IRC, klient zobowiązany jest przesłać nazwę pod jaką będzie widziany – *nick*. W tym przypadku *nick* bota ma postać *bot_no-12*. Następnym istotnym poleceniem jest *JOIN*, wywołane w taki sposób:

```
JOIN #my-botnet-00 botnet-#chan-00
```

Pierwszy parametr stanowi nazwę kanału, a drugi klucz.

Na podstawie tych informacji jesteśmy w stanie połączyć się z botnetem. Po wejściu na kanał powinniśmy ujrzeć pozostałe komputery tworzące sieć. Ze względu na brak hasła, nie jesteśmy w stanie dokonać uwierzytelnienia, dlatego jesteśmy ograniczeni do biernej obserwacji kanału.

Czytanie pliku binarnego

Główną wadą poprzedniej metody jest brak możliwości uzyskania hasła. Podgląd pliku binarnego bota, stwarza niekiedy szansę na odczytanie zmiennych przechowujących dane konfiguracyjne – w tym hasła. Dobrze do tego celu nadaje się edytor szesnastkowy o nazwie *hiew*. Przed rozpoczęciem analizy pliku należy zwrócić uwagę na to, czy plik nie jest skompresowany pakerem dla plików wykonywalnych, takim jak UPX. Jest to dosyć częstą praktyką wśród twórców botów, ze względu na to, że skompresowanie pliku exe pozwala na znaczne zmniejszenie objętości, a także utrudnia wykrycie programom antywirusowym. Jeżeli mamy wątpliwości co do pakera, jaki został użyty do skompresowania pliku, możemy posłużyć się narzędziem *PeiD* (link w ramce *W Sieci*).

W tym przypadku bot jest skompresowany programem UPX. W celu rozpakowania pliku należy wydać komendę:

```
upx -d  bot.exe
```

Jeśli nie uzyskaliśmy błędów przy dekompresji, możemy wczytać otrzymany plik do edytora *hiew*. Wciskając [*F4*] przełączamy edytor w tryb heksadecymalny.

Następnie przy pomocy sekwencji klawiszy [*F8*] , [*F6*] przechodzimy do tabeli, w której znajduje się spis sekcji pliku EXE. Aby obejrzeć zade-

```
00407B52 . 68 AC134100    PUSH spybot.004113AC
00407B57 . 6A 1E          PUSH 1E
00407B59 . FF75 08        PUSH DWORD PTR SS:[EBP+8]
00407B5C . E8 3B97FFFF    CALL spybot.0040129C        funkcja dekodujaca!
00407B61 . 83C4 0C        ADD ESP,0C
00407B64 . 50             PUSH EAX                    Name
00407B65 . E8 726D0000    CALL <JMP.&WS2_32.gethostbyname>   gethostbyname
```

Rysunek 9. *Kod przy wywołaniu funkcji gethostbyname*

```
0040129C  $ 53            PUSH EBX
0040129D  . 56            PUSH ESI
0040129E  . 8B7424 0C     MOV ESI,DWORD PTR SS:[ESP+C]
004012A2  . 8B5C24 14     MOV EBX,DWORD PTR SS:[ESP+14]
004012A6  . 31C9          XOR ECX,ECX
004012A8  . EB 11         JMP SHORT spybot.004012BB
004012AA  > 0FBE040E      MOVSX EAX,BYTE PTR DS:[ESI+ECX]
004012AE  . 2B4424 10     SUB EAX,DWORD PTR SS:[ESP+10]
004012B2  . 6BD1 03       IMUL EDX,ECX,3
004012B5  . 29D0          SUB EAX,EDX
004012B7  . 88040B        MOV BYTE PTR DS:[EBX+ECX],AL
004012BA  . 41            INC ECX
004012BB  > 803C0E 00     CMP BYTE PTR DS:[ESI+ECX],0
004012BF  . 75 E9         JNZ SHORT spybot.004012AA
004012C1  . C6040B 00     MOV BYTE PTR DS:[EBX+ECX],0
004012C5  . 89D8          MOV EAX,EBX
004012C7  . 5E            POP ESI
004012C8  . 5B            POP EBX
004012C9  . C3            RETN
```

Rysunek 10. *Ciało funkcji decode*

References in spybot:.text to 0040129C

Address	Disassembly	Comment
00401628	CALL spybot.0040129C	
004035B5	CALL spybot.0040129C	
00403967	CALL spybot.0040129C	
004039BA	CALL spybot.0040129C	
00407B5C	CALL spybot.0040129C	funkcja dekodujaca!

Rysunek 11. *Referencje do funkcji dekodującej*

Trudne pojęcia

- kod P-O-C – czyli *Proof Of Concept,* jest to nieszkodliwy kod (np. uruchamiający kalkulator) – dołączany do opisu błędu/techniki jako dowód istnienia błędu oraz pokaz możliwości jego wykorzystania.

O autorach

Dawid Gołuński jest samoukiem, pasjonatem, od wielu lat interesującym się Informatyką, a w szczególności aspektami bezpieczeństwa. Studiuje sieci komp. w ramach programu Cisco Network Academy na Politechnice Poznańskiej.

golunski@crackpl.com

Krzysztof Kułaj jest studentem kierunku Elektronika i Telekomunikacja na Uniwersytecie Zielonogórskim, na co dzień zajmuje się wdrożeniami systemów antywirusowych.

krzysztof_kulaj@o2.pl

klarowane dane, wybieramy sekcje *.data*. Jeżeli dane nie zostały dodatkowo zakodowane, poruszając się po tej sekcji, ujrzymy dane konfiguracyjne bota, takie jak komenda logowania, hasło, nazwa kanału, klucz, adres serwera, wersja bota itp.

Deasemblacja

Stosunkowo często zdarza się, że najbardziej istotne dane wewnątrz bota (hasła, adresy serwerów, nazwy kanałów) są kodowane. Ma to miejsce, aby uniemożliwić ich odczytanie z poziomu hexedytora. W takim przypadku konieczna może okazać się deasemblacja programu oraz przeprowadzenie inżynierii wstecznej w celu ustalenia sposobu kodowania ciągów znakowych. Wykorzystamy debugger o nazwie *OllyDbg*, aby przedstawić analizę działania dwóch zmodyfikowanych wersji *SpyBota*. Sprawa jest prosta, gdy łańcuchy znaków dekodowane są nieopodal początku programu. Warto wspomnieć, że domyślna wersja SpyBota postępuje właśnie w taki sposób. Wystarczy wtedy uruchomić bota, a następnie podłączyć się pod utworzony przez niego proces za pomocą opcji *File --> Attach*.

W dolnym oknie znajduje się sekcja danych. Z chwilą podłączenia pod proces bota –ujrzymy zdekodowane łańcuchy znaków.

Sprawa jest nieco trudniejsza, gdy ciągi znaków są dekodowane tuż przed wywołaniem danej funkcji. Przykładowo:

```
strcmp( dekoduj(
   zakodowany_ciag), password);
```

W takim wypadku konieczne jest odnalezienie funkcji dekodującej. Jeżeli uda się ją odnaleźć, będziemy w stanie prześledzić jej działanie oraz sprawdzić, jakie ciągi znaków są do niej przekazywane. Wyszukanie procedury może przysporzyć pewne problemy, gdyż tylko kilka najważniejszych łańcuchów podlega kodowaniu, a program w czasie swojego działania wywołuje dużą liczbę funkcji. Sprawdzanie każdej z nich zajęłoby sporo czasu. Dlatego najlepiej z góry ustalić funkcję, do której na pewno przekazywany jest

uprzednio zdekodowany string. Np. jeśli podczas przechwytywania pakietów snifferem zobaczyliśmy napis:

```
DNS Standard query A irc.efnet.pl
DNS Standard query response
      A 217.17.33.10
```

To oznacza, że bot korzysta z funkcji *gethostbyname()*, aby rozwiązać nazwę hosta (ircserwera) na adres IP. Idąc tym tropem, możemy przypuszczać, że gdzieś w kodzie znajduje się wywołanie podobne do:

```
gethostbyname(decode(
   zakodowana_nazwa_ircserwera))
```

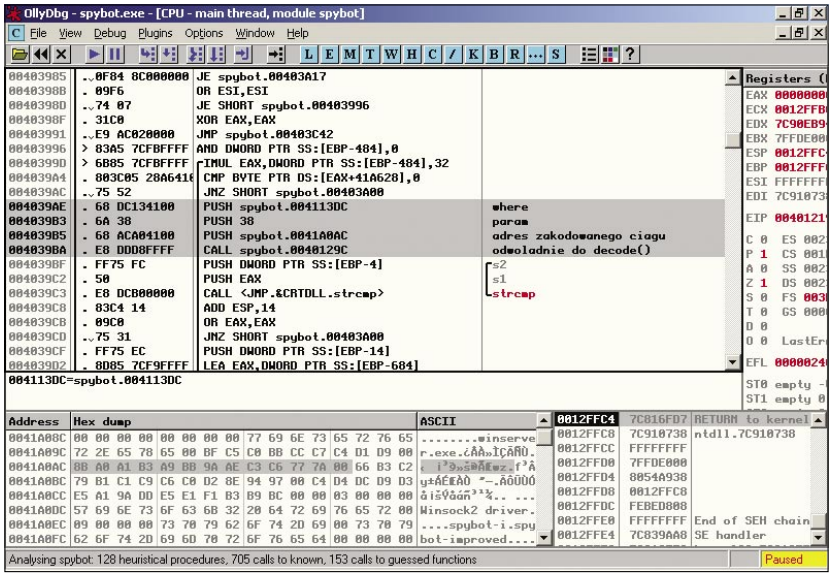
OllyDbg umożliwia wylistowanie wszystkich odwołań do funkcji API, poprzez wybranie z menu kontekstowego *Search for --> All intermodular calls*.

Jak widzimy na Rysunku 8, odwołanie do funkcji *gethostbyname()* występuje pod adresem *00407B65*. Wciskając enter na tej linii zostaniemy przeniesieni do kodu, który widzimy na Rysunku 9.

Argument dla funkcji *gethostbyname()* znajduje się w rejestrze EAX, który jest odkładany na stos instrukcją `PUSH EAX`. Co jednak zawiera ten rejestr? Jak się domyślamy – jest to wynik poszukiwanej przez nas funkcji - dekodującej łańcuchy znaków (nazwijmy ją *decode*). Na podstawie jej wywołania możemy wywnioskować, że przyjmuje ona 3 argumenty oraz zwraca wskaźnik do zdekodowanego ciągu znaków (który, w tym przypadku przekazywany jest do *gethostbyname()*). Oczywiście możemy zajrzeć do jej wnętrza.

Po wykonaniu dokładnej analizy tej funkcji, jesteśmy w stanie opisać jej działanie za pomocą języka C.

Funkcja działa w następujący sposób: argument *text* zawiera adres dekodowanego ciągu znaków. Pętla wykonuje się tyle razy, ile znaków ma dekodowany ciąg. Przy każdym obiegu pętli od kodu ASCI znaku odejmowana jest liczba określona parametrem *param* oraz pozycja znaku (numerując od zera) pomnożona przez 3. Zdekodowany ciąg znaków trafia pod adres wskazywany przez parametr *where*. Aby poznać zakodowane łańcuchy znaków, wystarczy wyświetlić wszelkie odwołania (referencje) do funkcji *decode*. Czynimy to, zaznaczając linijkę określoną jako *funkcja dekodujaca!*, a następnie wybieramy z menu kontekstowego *Find references to --> Call destination*



Rysunek 12. *Jedno z wywołań funkcji decode*

Wybierając ostatnią z referencji, trafimy do kodu przedstawionego na Rysunku 12.

Znajdziemy tutaj wszystkie informacje potrzebne do zdekodowania stringu:

- adres, pod którym mieści się zakodowany łańcuch znaków (zaznaczony w dolnej części rysunku)
- parametr kodujący *param*

Podstawiając te dane do funkcji dekodującej, otrzymamy ciąg znaków *SecretPass!!* Zamiast rozpracowywać algorytm dekodujący, można oczywiście ustawić pułapkę (*breakpoint*) na wyjściu z funkcji *decode* - co zapewni zatrzymanie programu w momencie zakończenia procedury dekodującej – pozwalając na swobodne odczytanie zdekodowanego łańcucha (jego adres wskazuje rejestr EAX).

Obrona przed botami

Ponieważ boty często są pisane od podstaw, programy antywirusowe nie posiadają ich sygnatur. Pewnym rozwiązaniem może być używanie firewalla, który jest w stanie zablokować ruch wychodzący z niezaufanego programu. Należy tu jednak zaznaczyć, że firewalle nie zawsze są całkowicie skuteczne. Odpowiednio napisany bot niekiedy jest w stanie przedostać się poprzez zaporę, używając techniki zdalnych wątków (*remote threads*).

Najlepszą metodą ochrony przed botami jest uniemożliwienie infekcji – czyli przeciwdziałanie metodom ujętym w rozdziale dot. rozprzestrzeniania się botów. Należy posiadać aktualne poprawki bezpieczeństwa, nigdy otwierać linków z nieznanych źródeł oraz używać silnych haseł w przypadku udostępniania zasobów Windows.

Podsumowanie

Zabezpieczenie przed sieciami Zombie to jedno z najważniejszych zadań, jakie stoi przed społecznością Internetu. Według przeprowadzonych badań, 25% komputerów osobistych znajduje się w co najmniej jednej sieci Zombie. Oznacza to, że statystycznie rzecz biorąc, co czwarty z Czytelników ma zainstalowanego co najmniej jednego bota. Artykuł pokazuje, że bonety niosą za sobą liczne niebezpieczeństwa, w wyniku których łatwo utracić wrażliwe dane czy pieniądze zgromadzone na koncie bankowym. Nie pozwólmy, aby nasz komputer stał się narzędziem w rękach intruza. ●