Helion

Analiza śledcza i powłamaniowa

Zaawansowane techniki prowadzenia analizy w systemie Windows 7 | Wydanie III

Harlan Carvey

Tytuł oryginału: Windows Forensic Analysis Toolkit, Third Edition:
Advanced Analysis Techniques for Windows 7

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-246-6652-2

Syngress is an imprint of Elsevier. 225 Wyman Street, Waltham, MA 02451, USA.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/anasl3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ

Analiza systemu plików

4

SPIS TREŚCI

W TYM ROZDZIALE

- Tablica MFT.
- Dzienniki zdarzeń systemowych (ang. *Event Logs*).
- Folder *Recycle Bin*.
- Pliki *prefetch*.
- Zaplanowane zadania.
- Listy szybkiego dostępu (ang. *jump lists*).
- Pliki hibernacji.
- Pliki aplikacji.

WPROWADZENIE

Podobnie jak inne systemy operacyjne, system Windows składa się z ogromnej liczby plików, z których zdecydowana większość jest zapisana w innych formatach niż prosty tekstowy ASCII. Bardzo wiele z tych plików nie wnosi niczego nowego pod względem analizy śledczej i powłamaniowej, niemniej jednak istnieje szereg plików, które są lub mogą być nieocenionym źródłem informacji dla analityka. W systemie można również znaleźć pewną liczbę zupełnie nieznanych analitykowi plików, zapisanych w formacie, który nie poddaje się przeszukiwaniu pod kątem występowania słów kluczowych. Oczywiście takie pliki mogą wnosić wiele istotnych informacji do przeprowadzanej ekspertyzy, pod warunkiem jednak, że analityk będzie zdawał sobie sprawę z ich istnienia i będzie wiedział, jak przeanalizować i jak interpretować ich zawartość.

Celem tego rozdziału nie jest przedstawianie po raz kolejny metod analizy systemu plików, ponieważ takie zagadnienia były już wielokrotnie szczegółowo omawiane w wielu powszechnie dostępnych źródłach. Zamiast tego chciałbym tutaj poruszyć sprawy związane z istnieniem pewnych plików, pokazać, co wynika z faktu ich istnienia, oraz omówić techniki analizy ich zawartości, które mogą być przydatne podczas prowadzenia dochodzenia, a które nie zawsze są znane szerokiemu gronu analityków.

Jak już wspominałem wcześniej, system Windows składa się z ogromnej liczby plików zapisanych w przeróżnych formatach. Takie pliki mogą zawierać interesujące informacje zapisane lub osadzone w określonych strukturach danych. Niektóre z takich struktur zostały szczegółowo udokumentowane przez producentów oprogramowania, a poznanie budowy innych struktur zawdzięczamy po prostu żmudnej pracy analityków. Powszechną praktyką stosowaną przez wielu analityków jest rozpoczynanie ekspertyzy od przeszukiwania zawartości dysków fizycznych, binarnych obrazów dysków twardych czy wybranych plików pod kątem występowania określonych słów kluczowych, co pozwala na wstępne zidentyfikowanie potencjalnych źródeł informacji. Taka metoda sprawdza się w przypadku prostych formatów plików, ale powinieneś pamiętać, że istnieje cały szereg plików, których zawartość może mieć nieocenione znaczenie dla prowadzonego dochodzenia, nawet jeżeli przeszukanie ich pod kątem występowania określonych słów kluczowych nie przyniosło żadnych rezultatów.

W bardzo wielu przypadkach samo istnienie określonej struktury danych wewnątrz pliku może stanowić cenną wskazówkę lub rozszerzyć kontekst analizowanych danych. Przykładowo podczas jednej z analiz powłamaniowych związanych z kradzieżą danych przeszukiwanie materiału dowodowego pod kątem potencjalnych numerów kart kredytowych przyniosło kilka trafień w jednym z plików gałęzi rejestru (analizą zawartości rejestru będziemy się szczegółowo zajmować w rozdziale 5.). Dokładniejsza analiza trafień wykazała, że wskazane numery nie były ani nazwami kluczy rejestru, ani ich wartościami, ani nawet danymi — okazało się, że odnalezione ciągi znaków znajdują się w niealokowanej przestrzeni pliku gałęzi rejestru. Dalsza analiza wykazała, że bajty danych reprezentujące odnalezione ciągi znaków znajdują się w sektorach dysku, które poprzednio były zajmowane przez jeden z usuniętych już dokumentów.

Po prostu po usunięciu tego pliku zajmowane przez niego sektory dysku zostały oznaczone jako dostępne i po pewnym czasie zostały zajęte przez rozrastający się plik gałęzi rejestru.

Przedstawiony przykład doskonale pokazuje, że dobry analityk powinien wychodzić daleko poza proste wyszukiwanie słów kluczowych i ostrożnie formułować płynące stąd wnioski, ponieważ może to mieć krytyczne znaczenie dla prowadzonego dochodzenia i może przynieść odpowiedzi na wiele ważnych pytań związanych ze sprawą. Na przykład zastanów się nad odkryciem, które omawialiśmy przed chwilą. Czy zdanie „Znalazłem poszukiwane numery kart kredytowych w rejestrze" naprawdę będzie miało jakąkolwiek wartość dla klienta zlecającego ekspertyzę? A może znacznie ważniejsze byłoby dokładne opisanie miejsca, w którym takie numery zostały znalezione, i sposobu, w jaki się tam mogły znaleźć? Celem tego rozdziału będzie zatem omówienie wewnętrznej struktury wybranych plików i pokazanie, jaką wartość może mieć dla analityka dobra znajomość struktury plików.

TABLICA MFT

W systemie plików NTFS tablica MFT spełnia rolę głównej listy plików i zawiera szereg metadanych szczegółowo opisujących wszystkie obiekty zapisane w systemie plików, takie jak pliki, katalogi czy metapliki[1]. Metadane z systemu plików mogą mieć krytyczne znaczenie dla rezultatów przeprowadzanej analizy, zwłaszcza jeżeli istnieje podejrzenie, że metadane wybranych plików zostały celowo zmodyfikowane z zamiarem ukrycia lub zamaskowania wykonanych operacji (takie działania są często określane w języku angielskim jako *anti-forensics*, czyli działania mające na celu utrudnienie analizy śledczej i powłamaniowej).

Oczywiście w tym rozdziale nie znajdziesz rozbudowanego traktatu, szczegółowo opisującego kompletną wewnętrzną strukturę tablicy MFT — ale też nie taki jest cel tej książki. Zamiast tego skoncentrujemy się tutaj na określonych strukturach danych (lub inaczej, na określonych *atrybutach* plików), które możesz wyodrębnić z poszczególnych rekordów w tablicy MFT, a dyskusję na temat samej tablicy MFT ograniczymy do krótkiego opisu rekordów MFT i dwóch wybranych atrybutów. W dalszej części rozdziału omówimy kilka narzędzi, których możesz użyć do parsowania zawartości tablicy MFT. Nie będziemy jednak zbytnio zagłębiać się w wewnętrzną budowę tablicy MFT, stąd nie powinieneś raczej oczekiwać, że po przeczytaniu tego rozdziału będziesz w stanie tworzyć od podstaw własne narzędzia służące do tego celu. Prawdopodobnie najlepszym, bardzo obszernym i szczegółowym źródłem informacji na temat wewnętrznych tajemnic tablicy MFT i systemu plików NTFS i ich wykorzystania w analizie śledczej i powłamaniowej jest wydana w 2005 roku wspaniała książka Briana Carriera, zatytułowana *File System Forensic Analysis* (wydawnictwo Addison-Wesley Professional)[2].

[1] W systemie plików NTFS każdy zapisany obiekt jest plikiem, stąd mówimy o plikach danych, plikach reprezentujących katalogi i tak dalej — *przyp. tłum.*

[2] Patrz strona *http://www.pearsonhighered.com/educator/product/File-System-Forensic-Analysis/9780321268174.page — przyp. tłum.*

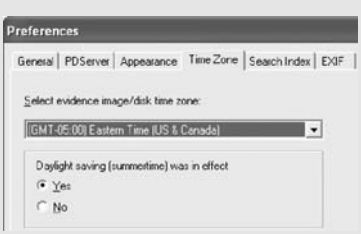
Każdy rekord w tablicy MFT składa się z 1024 bajtów. System plików NTFS traktuje każdy obiekt w nim zapisany jako plik. Poszczególne rekordy w tablicy MFT reprezentujące kolejne pliki zawierają szereg atrybutów przechowujących metadane opisujące plik, a czasami, w przypadku bardzo małych plików, przechowujących również całą zawartość pliku (tzw. atrybut danych). Pierwsze 42 bajty każdego rekordu zajmuje nagłówek (ang. *File Record Header*), który zawiera informacje takie jak liczba dowiązań (czyli liczba katalogów mających dowiązania do tego pliku, co pozwala na określenie liczby twardych dowiązań do pliku), informacje o tym, czy dany rekord opisuje plik, czy katalog, czy dany plik lub katalog jest używany lub usunięty, oraz dane o rozmiarze przestrzeni alokowanej i wykorzystywanej przez konkretny plik (inaczej rozmiar pliku).

Wszystkie rekordy opisujące pliki i katalogi mają atrybut o nazwie `$STANDARD_INFORMATION` (`$SIA`), który zajmuje 72 bajty (dla systemów Windows 2000 i nowszych) i zawiera — oprócz innych elementów — zestaw znaczników czasowych pliku, zapisanych w postaci standardowego, 64-bitowego obiektu `FILETIME`. Obiekt `FILETIME` zgodnie z definicją firmy Microsoft reprezentuje liczbę 100-nanosekundowych interwałów czasowych, jakie upłynęły od 1 stycznia 1601 roku do momentu zapisania danego znacznika. To właśnie z tymi znacznikami masz najczęściej do czynienia, kiedy pracujesz na co dzień z systemem operacyjnym (na przykład są wyświetlane na konsoli za pomocą polecenia `dir` lub w oknie menedżera plików Windows Explorer). W systemie NTFS znaczniki czasowe są zapisywane w rekordach MFT w formacie UTC (ang. *Universal Time Coordinated*). Oznacza to, że kiedy widzimy takie znaczniki czasu na przykład wyświetlone w oknie programu Windows Explorer (lub w oknie wiersza poleceń za pomocą polecenia `dir`), to system przed wyświetleniem automatycznie dokonuje ich konwersji na czas obowiązujący w strefie czasowej, której ustawienia są zapisane w rejestrze systemowym na danym komputerze.

UWAGA

Wyświetlanie znaczników czasu w aplikacjach informatyki śledczej

Większość komercyjnych aplikacji śledczych pozwala na wyświetlanie znaczników czasu ostatniej modyfikacji pliku, ostatniego dostępu do pliku i czasu utworzenia pliku (ang. *MAC timestamps — Modified, Accessed and Created*) zgodnie z ustawieniami strefy czasowej wybranej przez analityka. W programie ProDiscover Incident Response Edition taką operację możesz wykonać, wybierając z menu głównego polecenie *File/Preferences*, co spowoduje wyświetlenie na ekranie okna dialogowego *Preferences*, tak jak to zostało przedstawione na rysunku 4.1.



RYSUNEK 4.1.

Ustawienia strefy czasowej w programie ProDiscover

W atrybucie `$SIA` są przechowywane cztery znaczniki czasu: czas ostatniej modyfikacji pliku (ang. *Modified*), czas ostatniego dostępu do pliku (ang. *Accessed*), czas ostatniej zmiany rekordu MFT reprezentującego plik (ang. *Changed*) oraz czas utworzenia pliku (ang. *Born*, czyli czas „narodzin" pliku). Komplet tych czterech znaczników jest często określany mianem **znaczników MACB**, gdzie każda z liter reprezentuje odpowiedni znacznik. Czasami spotyka się nieco inny sposób definiowania tych znaczników, znany jako **znaczniki MACE** — czas modyfikacji pliku (ang. *Modified*), czas ostatniego dostępu do pliku (ang. *Accessed*), czas utworzenia pliku (ang. *Created*) oraz czas ostatniej modyfikacji rekordu MFT tego pliku (ang. *Entry modified*). Dla zachowania spójności i w celu uniknięcia nieporozumień w dalszej części tego rozdziału mówiąc o znacznikach czasu, będziemy się trzymali notacji `MACB`. Znaczniki czasu są modyfikowane i aktualizowane podczas normalnego działania systemu — na przykład kiedy jest tworzony nowy plik, jego wszystkie znaczniki czasu są ustawiane na bieżącą datę i czas. Za każdym razem, kiedy do pliku są wprowadzane jakiekolwiek zmiany (na przykład dane są dodawane, modyfikowane czy usuwane przez użytkownika lub usługę), aktualizowany jest znacznik czasu ostatniej modyfikacji pliku. Jak za chwilę zobaczysz w ramce „Znacznik czasu ostatniego dostępu do pliku", sposób aktualizacji tego znacznika podlega kilku ściśle określonym regułom.

OSTRZEŻENIE

Znacznik czasu ostatniego dostępu do pliku

Większość analityków sądzi, że kiedy dany plik jest otwierany lub w jakikolwiek inny sposób „dotykany" przez system, znacznik czasu ostatniego dostępu do pliku (przechowywany w atrybucie `$SIA` rekordu MFT) jest modyfikowany tak, aby odzwierciedlać czas wykonania takiej operacji. W praktyce jednak okazuje się, że znacznik czasu ostatniego dostępu do pliku zapisanego w systemie NTFS na dysku twardym nie zawsze jest aktualny. Ze względu na wydajność systemu plików NTFS opóźnia zapis na dysku zaktualizowanego znacznika czasu ostatniego dostępu do pliku, przechowując jednak w pamięci operacyjnej systemu poprawne wartości znaczników. System Windows aktualizuje znaczniki na dysku dopiero wtedy, kiedy wartość znacznika czasu zapisanego na dysku różni się od wartości znacznika czasu przechowywanego w pamięci operacyjnej o co najmniej godzinę lub więcej (zgodnie z informacjami podanymi na stronie firmy Microsoft, patrz *http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/fsutil_behavior.mspx?mfr=true*).

Oprócz tego w rejestrze systemu znajduje się wpis `NtfsDisableLastAccessUpdate` (patrz klucz `HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\`), którego ustawienie na wartość 1 powoduje zablokowanie aktualizacji znacznika czasu ostatniego dostępu do pliku. W systemach Windows XP oraz Windows 2003 wpis `NtfsDisableLastAccessUpdate` nie jest tworzony podczas instalacji domyślnej. W razie potrzeby jednak możesz taki wpis utworzyć ręcznie i nadać mu wartość 1. Wykonanie takiej operacji jest zalecane w celu zwiększenia wydajności systemu plików w systemach spełniających rolę dużych serwerów plików. W systemie Windows Vista (oraz jego następcy, Windows 7) wpis `NtfsDisableLastAccessUpdate` jest tworzony podczas instalacji systemu i domyślnie ustawiany na wartość 1. Nie oznacza to jednak, że znacznik czasu ostatniego dostępu do pliku nigdy nie jest aktualizowany — w praktyce niektóre operacje, takie jak utworzenie pliku, przeniesienie pliku do innego katalogu czy skopiowanie pliku, mogą powodować zmianę wartości tego znacznika. Zablokowanie aktualizacji znacznika odnosi się zatem tylko do takich operacji jak otwarcie pliku i przeglądanie jego zawartości.

Oprócz wspomnianego atrybutu `$SIA` każdy plik zapisany w systemie plików NTFS ma w swoim rekordzie MFT co najmniej jeden atrybut `$FILE_NAME` (`$FNA`). Atrybut ten składa się z 66 bajtów metadanych oraz nazwy pliku. Każdy rekord MFT może mieć więcej niż jeden atrybut `$FNA`, ponieważ oprócz standardowych długich nazw każdy plik może mieć również krótką nazwę, zapisaną w starym formacie *DOS 8.3* (aczkolwiek w razie potrzeby da się to zablokować, zmieniając wartość odpowiedniego wpisu w rejestrze). W praktyce oznacza to, że jeżeli masz plik o nazwie *To jest plik o bardzo długiej nazwie.doc*, to będzie istniał również atrybut `$FNA` zawierający nazwę *to_jes~1.doc*, która składa się z ośmiu znaków, separatora w postaci kropki i trzech znaków rozszerzenia. Jak widać, w takiej sytuacji rekord MFT reprezentujący nasz plik będzie miał jeden atrybut `$SIA` oraz dwa atrybuty `$FNA`.

Oprócz nazwy pliku w atrybutach `$FNA` są przechowywane odwołania do katalogu nadrzędnego (co pozwala różnym narzędziom na całkowite zrekonstruowanie ścieżki do pliku) oraz cztery znaczniki czasu zapisane w tym samym formacie co znaczniki w atrybucie `$SIA`. Podstawowa różnica pomiędzy tymi znacznikami polega jednak na tym, że system Windows nie aktualizuje znaczników atrybutu `$FNA` w taki sposób jak w przypadku znaczników przechowywanych w atrybucie `$SIA`. Znaczniki czasu przechowywane w atrybucie `$FNA` reprezentują oryginalną datę i czas, kiedy plik został utworzony, przeniesiony lub kiedy została zmieniona jego nazwa. Z tego względu parsowanie zawartości rekordów MFT wybranych plików i porównywanie znaczników czasu przechowywanych w atrybutach `$SIA` i `$FNA` jest jedną z typowych technik, wykorzystywanych przez analityków do sprawdzenia, czy znaczniki w atrybucie `$SIA` nie zostały celowo zmodyfikowane przez użytkownika lub złośliwe oprogramowanie w celu ukrycia pliku albo zamaskowania wykonanej operacji.

W internecie jest dostępnych co najmniej kilka bezpłatnych narzędzi typu *open source*, które pozwalają na wyodrębnianie atrybutów `$SIA` i `$FNA` z rekordów MFT. Jednym z takich narzędzi jest skrypt napisany w języku Python przez Davida Kovara, noszący nazwę *analyzeMFT.py* (patrz strona *https://github.com/dkovar/analyzeMFT*). Skrypt zawiera kilka elementów graficznych, stąd przed pierwszym użyciem powinieneś dokładnie przeczytać jego dokumentację i zainstalować odpowiednie biblioteki na swoim komputerze. Innym rozwiązaniem może być otwarcie kodu źródłowego skryptu do edycji i zmiana wiersza `noGUI = False` na wiersz `noGUI = True`. Jeżeli jesteś zapalonym użytkownikiem systemu Windows, na stronie autora znajdziesz również instalator samodzielnej, wykonywalnej wersji Windows tego narzędzia. Skrypt *analyzeMFT.py* oferuje kilka bardzo użytecznych opcji, z których jedna, `-a`, pozwala na włączenie wykrywania „anomalii” (ang. *turn on anomaly detection*). Użycie tej opcji powoduje sprawdzenie, czy 32 mniej znaczące bity znaczników czasu w atrybucie `$SIA` mają wartość `0` oraz czy data utworzenia pliku zapisana w atrybucie `$FNA` jest późniejsza niż data zapisana w atrybucie `$SIA`. Pozytywny wynik dowolnego z tych testów może wskazywać na próbę ręcznej modyfikacji znaczników czasu w celu ukrycia pliku lub zamaskowania złośliwej aktywności. Przy okazji warto zaznaczyć, że kiedy używasz opcji wykrywania anomalii, wyniki działania skryptu, zapisywane w formacie CSV, mają aż 53 kolumny, czyli na przykład w arkuszu Excela będą zajmowały kolumny od *A* do *BA*.

Jakiś czas temu napisałem w języku Perl swój własny skrypt, *mft.pl*, przeznaczony do analizy znaczników czasu plików i katalogów (oraz innych informacji) przechowywanych w MFT. Skrypt możesz znaleźć w materiałach dodatkowych do niniejszej książki[3]; szczerze mówiąc, skrypt ten ustanowił dobrą bazę do napisania innych skryptów w języku Perl, poszerzonych o dodatkowe funkcjonalności. Jak się przekonasz w nieco dalszej części tego rozdziału, skrypt *mft.pl* w czytelny i przyjazny dla użytkownika sposób wyświetla informacje zawarte w nagłówkach rekordów MFT (ang. *MFT File Entry Header*) oraz atrybutach `$SIA` i `$FNA`.

Analizując informacje zawarte w tablicy MFT, powinieneś zawsze pamiętać, że różne operacje wykonywane przez użytkownika (poza prostym odczytywaniem i zapisywaniem zawartości pliku) mogą mieć wpływ na wartości znaczników czasu przechowywanych w atrybucie `$SIA`. Więcej szczegółowych informacji na ten temat znajdziesz w artykule KB299648 bazy wiedzy Microsoft, dostępnym na stronie *http://support.microsoft.com/kb/299648*. Na przykład kopiowanie lub przenoszenie plików pomiędzy katalogami w obrębie jednej partycji NTFS powoduje zachowanie znacznika czasu ostatniej modyfikacji pliku, ale kopiowanie spowoduje zmianę znacznika czasu utworzenia pliku na bieżącą datę i czas, podczas gdy przenoszenie pliku spowoduje zachowanie oryginalnej daty i czasu utworzenia pliku. Więcej ciekawych informacji znajdziesz w wymienionym wyżej artykule bazy wiedzy Microsoft, ale na chwilę obecną powinieneś po prostu zapamiętać, że to, czy i które znaczniki czasu zostaną zmodyfikowane, zależy od wielu różnych czynników, takich jak to, czy plik jest kopiowany, czy przenoszony, czy jest przenoszony do innego katalogu na tej samej partycji, czy też jest przenoszony z jednej partycji na drugą itp.

UWAGA

Testowanie

Nie udało mi się jeszcze znaleźć (ale ciągle szukam) pełnej dokumentacji szczegółowo opisującej wpływ wszystkich możliwych scenariuszy operacji plikowych na zachowanie poszczególnych znaczników czasu. Z tego względu dobry analityk powinien zawsze przetestować w praktyce przyjęte założenia, kiedy podczas analizy napotyka niestandardowe warunki czy sytuacje. Na przykład jeżeli podejrzewasz, że plik został skopiowany z przenośnej pamięci USB sformatowanej w systemie FAT32 na partycję NTFS, jak zachowały się poszczególne znaczniki czasu (zarówno dla pliku oryginalnego, jak i utworzonej kopii)? A co w przypadku kopiowania plików pomiędzy udziałami sieciowymi NTFS? W praktyce w takich sytuacjach zawsze warto zasymulować warunki danej hipotezy i na własnej skórze zweryfikować, czy przyjęte założenia były poprawne.

[3] Patrz strona *http://code.google.com/p/winforensicaanalysis/downloads/detail?name=wfa3e.zip&can=2&q=* — *przyp. tłum.*

Znaczniki czasu przechowywane w atrybutach `$SIA` mogą być modyfikowane nie tylko poprzez wykonanie opisanych wcześniej operacji, ale również w wyniku celowej działalności użytkownika. Jeżeli dany użytkownik ma prawa zapisu konkretnego pliku, może również ustawić znaczniki czasu dla tego pliku na arbitralnie wybraną datę i czas — takie modyfikacje zostaną automatycznie zapisane w znacznikach czasu przechowywanych w atrybucie `$SIA` pliku (zgodnie z artykułem *http://msdn.microsoft.com/en-us/library/cc781134* takie zmiany są wpisywane również w rekordach MFT reprezentujących katalogi). Dokonywanie takich celowych zmian jest jedną z technik mających na celu utrudnienie lub zmylenie analizy śledczej i powłamaniowej, często określaną nazwą *timestomping*, która wzięła się od nazwy jednego z pierwszych narzędzi służących do tego celu, *timestomp.exe* (aczkolwiek w czasie kiedy pisałem tę książkę, praktycznie nie byłem w stanie znaleźć już kopii tego narzędzia w internecie). To narzędzie pozwala na ustawienie wybranych znaczników czasu pliku na arbitralnie wybraną datę i czas. Spróbuj sobie wyobrazić potencjalne skutki jego użycia, kiedy system będący przedmiotem dochodzenia został skonfiskowany pracownikowi w roku 2011, wstępna analiza wykazała, że podejrzane pliki zostały utworzone w roku 2014, a znaczniki czasu ostatniego dostępu pokazują, że użytkownik ostatnio korzystał z tych plików w roku 1984. Nawet tylko jeden czy dwa pliki z tak zmodyfikowanymi znacznikami czasu byłyby wystarczającym powodem do zakwestionowania również innych plików. Jedną z wad używania narzędzia *timestomp.exe* jest jednak to, że aplikacja ta wydaje się mieć tylko 32-bitową rozdzielczość czasu, co powoduje, że jej użycie ustawia 32 mniej znaczące bity obiektu `FILETIME` na wartość 0. Dzięki temu użycie tego narzędzia jest relatywnie proste do wykrycia — przykładowo, jak już wspominałem wcześniej, wykonanie takiego testu jest częścią procesu „wykrywania anomalii" w skrypcie *analyzeMFT.py*, napisanym w języku Python przez Davida Kovara.

Inną techniką modyfikacji znaczników czasu plików jest kopiowanie wartości znaczników czasu z innego pliku, zwłaszcza takiego, który jest częścią oryginalnej instalacji systemu Windows, jak na przykład *kernel32.dll*, zlokalizowany zazwyczaj w katalogu *C:\Windows\system32*. Ta technika pozwala na uniknięcie problemów z rozdzielczością czasu, jakie trapią program *timestomp.exe*, znacznie bardziej utrudnia wykrycie pliku podczas analizy śledczej (patrz rozdział 7.), a co więcej, taką operację można łatwo wykonać za pomocą standardowych funkcji API (ang. *Application Programming Interface*) systemu Windows.

Analiza i porównywanie znaczników `MACB` z atrybutów `$SIA` i `$FNA` jest tylko jednym z przykładów doskonale ilustrujących fakt, że znajomość struktury i mechanizmów działania MFT może być źródłem bezcennych informacji dla analityka prowadzącego dochodzenie. Dobre zrozumienie budowy rekordów tablicy MFT i zasad, jakie rządzą ich tworzeniem, może dostarczyć analitykowi wielu dodatkowych informacji na temat statusu i historii badanych plików.

UWAGA

Timestomping

Jak już wspominałem wcześniej, podczas pracy nad tą książką nie udało mi się znaleźć kopii programu *timestomp.exe* w internecie. Nie stanowi to jednak żadnego problemu, ponieważ przy użyciu innych narzędzi, takich jak na przykład prosty skrypt w języku Perl, można bez trudu zmodyfikować znaczniki czasu danego pliku, kopiując wartości znaczników z innego, arbitralnie wybranego pliku. Aby to zrobić, zainstalowałem interpreter ActivePerl firmy ActiveState[4], a następnie przy użyciu polecenia przedstawionego poniżej zainstalowałem bibliotekę *Win32API::File::Time*:

```
C:\Perl>ppm install win32api-file-time
```

Po zainstalowaniu tego modułu mogłem go użyć do wywołania dwóch funkcji Windows API, pozwalających na odczytanie i zapisanie znaczników czasu wybranego pliku. Aby to zrobić, posłużyłem się następującymi poleceniami (fragment skryptu w języku Perl):

```
my ($atime, $mtime, $ctime) = GetFileTime ($file);
SetFileTime ($file2, $atime, $mtime, $ctime);
```

Żeby zilustrować działanie tego mechanizmu, dodałem parę wierszy wykorzystujących funkcję `stat()` do sprawdzania znaczników czasu wybranego pliku, a następnie uruchomiłem skrypt, który kopiował znaczniki czasu pliku *kernel32.dll* do pliku *C:|temp|test.txt*. Wyniki działania skryptu były następujące:

```
C:\Windows\system32\kernel32.dll
Creation Time: Tue Feb 28 08:00:00 2006
Last Access : Mon May 30 21:14:22 2011
Last Write : Sat Mar 21 10:06:58 2009

C:\Temp\test.txt
Creation Time: Mon May 30 17:36:12 2011
Last Access : Mon May 30 17:36:12 2011
Last Write : Mon May 30 17:36:12 2011

C:\Temp\test.txt
Creation Time: Tue Feb 28 08:00:00 2006
Last Access : Mon May 30 21:14:22 2011
Last Write : Sat Mar 21 10:06:58 2009
```

Skrypt najpierw wyświetla wyniki działania funkcji `stat()` dla pliku *kernel32.dll* (wszystkie znaczniki czasu są wyświetlane zgodnie z ustawieniami lokalnej strefy czasowej komputera; w moim przypadku był to standardowy czas letni wschodniego wybrzeża, czyli EDT[5]) oraz dla pliku *C:|temp|test.txt*. Następnie skrypt kopiuje wartości znaczników czasu z pliku *kernel32.dll* do pliku *C:|temp|test.txt* i ponownie wyświetla wyniki działania funkcji *stat()* dla ostatniego pliku. Znacznik czasu ostatniego dostępu do pliku *kernel32.dll* został zmodyfikowany po uruchomieniu skryptu i następnie skopiowany do pliku docelowego (w tym przykładzie skrypt został uruchomiony na komputerze działającym pod kontrolą systemu Windows XP SP3).

[4] Patrz strona *http://www.activestate.com/activeperl* — *przyp. tłum.*

[5] EDT — *Eastern Standard Time with Daylight Savings* (UTC–4 h) — *przyp. tłum.*

Aby zweryfikować działanie skryptu, przy użyciu programu FTK Imager skopiowałem tablicę MFT z systemu i pobrałem z niej odpowiednie dane przy użyciu omawianego już wcześniej skryptu *mft.pl*. Informacje dla pliku *test.txt* wyglądały następująco (wszystkie czasy są wyświetlone w formacie UTC, czyli *Zulu*[6]):

```
70319 FILE Seq: 15 Link: 1 0x38 3 Flags: 1
0x0010 96 0 0x0000 0x0000
M: Sat Mar 21 14:06:57 2009 Z
A: Tue May 31 01:14:22 2011 Z
C: Tue May 31 01:14:23 2011 Z
B: Tue Feb 28 11:59:59 2006 Z
0x0030 112 0 0x0000 0x0000
FN: test.txt Parent Ref: 67947 Parent Seq: 49
M: Mon May 30 21:36:12 2011 Z
A: Mon May 30 21:36:12 2011 Z
C: Mon May 30 21:36:12 2011 Z
B: Mon May 30 21:36:12 2011 Z
0x0080 48 0 0x0000 0x0018
```

Pierwszy zestaw znaczników `MACB` został pobrany z atrybutu `$SIA`, a drugi zestaw z atrybutu `$FNA`. Jak widać, znaczniki czasu pobrane z atrybutu `$SIA` odpowiadają rezultatom zwróconym przez funkcję `stat()` (oczywiście po odpowiednim przeliczeniu różnic czasu wynikających z różnych stref czasowych), podczas gdy znaczniki z atrybutu `$FNA` reprezentują oryginalne wartości znaczników `MACB` tego pliku.

Mechanizm tunelowania w systemie plików

Kolejnym zagadnieniem związanym z systemem plików komputerów działających pod kontrolą systemu Windows, które może mieć znaczący wpływ na wartości znaczników `MACB` plików obserwowane podczas analizy, jest **tunelowanie systemu plików** (ang. *file system tunneling*). Mechanizm ten jest spotykany zarówno w systemie FAT (ang. *File Allocation Table*), jak i NTFS (ang. *New Technology File System*) i został szczegółowo opisany w artykule KB172190 bazy wiedzy firmy Microsoft (patrz strona *http://support.microsoft.com/kb/172190*). Pojęcie tunelowania systemu plików odnosi się do faktu, że jeżeli w ciągu określonego czasu od momentu usunięcia danego pliku (domyślnie jest to 15 sekund) zostanie utworzony nowy plik o takiej samej nazwie, to rekord w tabeli plików (FAT lub MFT) należący do starego, usuniętego pliku zostanie ponownie użyty i przypisany do nowo utworzonego pliku. Inaczej mówiąc, jeżeli na przykład usuniesz plik o nazwie *mójplik.txt* (lub zmienisz jego nazwę) i zaraz po tym utworzysz nowy plik o takiej samej nazwie (albo zmienisz nazwę innego pliku na nazwę, jaką miał usunięty plik), to rekord w tablicy plików reprezentujący usunięty plik zostanie ponownie wykorzystany dla nowego pliku, dzięki czemu dla nowego pliku zostanie zachowana data utworzenia oryginalnego, usuniętego pliku. Zgodnie z artykułem KB172190 bazy

[6] Czas UTC (ang. *Universal Time Coordinated*) jest używany w nawigacji lotniczej i morskiej, gdzie jest znany pod swoją wojskową nazwą *Zulu Time* (*Zulu* w alfabecie fonetycznym odpowiada literze *z*, oznaczającej południk zerowy przebiegający przez londyńską dzielnicę Greenwich) — *przyp. tłum.*

wiedzy firmy Microsoft taki mechanizm *tunelowania* został zaimplementowany w celu zachowania kompatybilności wstecznej ze starszymi, 16-bitowymi aplikacjami systemu Windows, które korzystały z mechanizmu bezpiecznego zapisu plików na dysku (ang. *safe save*).

Aby zilustrować działanie mechanizmu tunelowania systemu plików, utworzyłem na moim komputerze, działającym pod kontrolą systemu Windows XP SP3, plik o nazwie *test3.txt* zajmujący 31 bajtów i odczekałem kilka dni. Znaczniki czasu z atrybutu `$SIA` tego pliku, odczytane za pomocą funkcji `stat()` języka Perl, wyglądały następująco (w formacie UTC):

```
c:\temp\test3.txt 31 bytes
Creation Time: Mon May 30 21:41:48 2011 UTC
Last Access : Mon May 30 21:41:48 2011 UTC
Last Write : Mon May 30 21:41:48 2011 UTC
```

Następnie usunąłem plik *test3.txt* i natychmiast (w ciągu maksymalnie 15 sekund) ponownie utworzyłem plik o takiej nazwie, wykonując z poziomu konsoli polecenie `echo` (na przykład `echo "A tunnel test" > test3.txt`). Nowa wersja pliku *test3.txt* zajmuje 18 bajtów, a znaczniki czasu nowego pliku wyglądają następująco (jak poprzednio w formacie UTC):

```
c:\temp\test3.txt 18 bytes
Creation Time: Mon May 30 21:41:48 2011 UTC
Last Access : Fri Jun 3 20:39:18 2011 UTC
Last Write : Fri Jun 3 20:39:18 2011 UTC
```

Jak łatwo zauważyć, data utworzenia nowego pliku jest identyczna z datą utworzenia oryginalnego pliku *test3.txt*, mimo że jego nowa wersja została „utworzona" 3 czerwca 2011 roku. Aby zweryfikować to zachowanie, za pomocą programu FTK Imager wyeksportowałem tablicę MFT tego dysku i „przepuściłem" ją przez skrypt *mft.pl*. Wartości znaczników `MACB` z atrybutów `$SIA` i `$FNA` były następujące:

```
39630 FILE Seq: 60 Link: 1 0x38 3 Flags: 1
0x0010 96 0 0x0000 0x0000
M: Fri Jun 3 20:39:18 2011 Z
A: Fri Jun 3 20:39:18 2011 Z
C: Fri Jun 3 20:39:18 2011 Z
B: Mon May 30 21:41:48 2011 Z
0x0030 112 0 0x0000 0x0000
FN: test3.txt Parent Ref: 67947 Parent Seq: 49
M: Fri Jun 3 20:39:18 2011 Z
A: Fri Jun 3 20:39:18 2011 Z
C: Fri Jun 3 20:39:18 2011 Z
B: Mon May 30 21:41:48 2011 Z
```

Jak widać, czas utworzenia pliku (znacznik `B`) w obu atrybutach, `$SIA` i `$FNA`, odpowiada czasowi utworzenia oryginalnego pliku, podczas gdy pozostałe znaczniki (`MAC`) zostały zmodyfikowane i odpowiadają czasowi utworzenia nowego pliku. Pamiętaj, w tym przykładzie utworzyłem z poziomu wiersza poleceń nowy plik — po utworzeniu pliku nie próbowałem go otwierać (co mogłoby zmienić znacznik czasu ostatniego dostępu do pliku) ani w żaden sposób modyfikować.

W praktyce wielokrotnie przekonywałem się, że zagadnienia, które omawialiśmy do tej pory w tym rozdziale, mają ogromne znaczenie podczas ustalania czasu utworzenia plików w systemach będących przedmiotem analizy śledczej czy powłamaniowej. Poprzez porównywanie znaczników czasu utworzenia, zapisanych w atrybutach `$SIA` i `$FNA` podejrzanych plików, bardzo często udawało mi się znaleźć wyraźne ślady wskazujące na próby ukrycia takich plików wśród innych plików systemowych i utrudnienia ich analizy. Całe zagadnienie stanie się łatwiejsze do zrozumienia, kiedy przeczytasz rozdział 7., w którym będziemy się zajmować analizą zdarzeń w osi czasu (ang. *timeline analysis*).

UWAGA

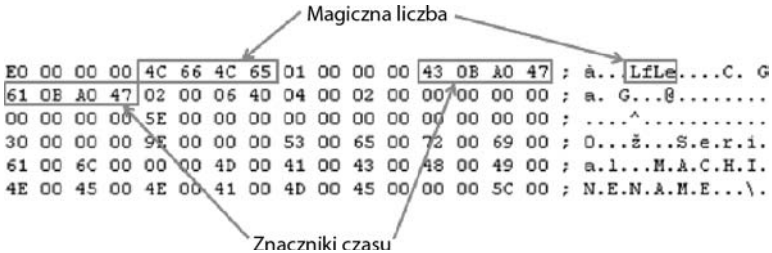
Indeksy $I30 w systemie plików NTFS

26 września 2011 roku Chad Tilbury, instruktor SANS, zamieścił na swoim blogu wpis zatytułowany *NTFS $I30 Index Attributes* (patrz strona *http://forensicmethods.com/ntfs-index-attribute*; artykuł ten został zamieszczony kilka dni wcześniej na blogu *SANS Forensic*[7]). Chad wykonał kawał naprawdę dobrej roboty, opisując atrybuty indeksu oraz sposoby ich analizy i wykorzystania do identyfikacji nazw usuniętych plików. W praktyce bardzo często zdarza się, że po zainfekowaniu komputera pliki złośliwego oprogramowania są usuwane, czy to przez napastnika, czy nawet nieumyślnie przez administratora lub użytkownika próbującego ratować system. Atrybuty indeksu mogą wskazać usunięte pliki i dostarczyć informacji o ich znacznikach `MACB` — jak opisuje Chad w swoim artykule, znaczniki czasu w atrybutach indeksu są bardzo podobne do tych, jakie możemy znaleźć w atrybucie `$FNA` rekordu MFT. Chad pokazuje również zastosowanie skryptu *indexparse.py*, napisanego w języku Python przez Williego Ballenthina, spełniającego rolę parsera atrybutów indeksu. Więcej szczegółowych informacji na temat skryptu znajdziesz na stronie *http://www.williballenthin.com/forensics/indx/index.html*.

DZIENNIKI ZDARZEŃ SYSTEMOWYCH

System Windows może rejestrować wiele różnych zdarzeń w dziennikach zdarzeń systemowych (ang. *Event Logs*). Liczba i rodzaj rejestrowanych zdarzeń zależy od konfiguracji zasad inspekcji komputera (ang. *audit configuration*) — sposoby określania bieżącej konfiguracji zasad inspekcji analizowanego systemu będziemy bardziej szczegółowo omawiać w rozdziale 5. Pliki dzienników zdarzeń w systemach Windows 2000, Windows XP oraz Windows 2003 zawierają rekordy zdarzeń zapisane w dobrze udokumentowanym formacie binarnym (patrz strona *http://msdn.microsoft.com/en-us/library/aa363646(v=VS.85).aspx*). Zgodnie z definicją formatu w nagłówku każdego rekordu znajduje się specyficzny ciąg bajtów zwany „magiczną liczbą" (ang. *magic number*), który w jednoznaczny sposób wyróżnia kolejne rekordy pliku (włącznie z rekordem nagłówka pliku, zawierającym informacje o samym dzienniku zdarzeń), tak jak to zostało przedstawione na rysunku 4.2.

[7] Patrz strona *http://computer-forensics.sans.org/blog/2011/09/20/ntfs-i30-index-attributes-evidence-of--deleted-and-overwritten-files — przyp. tłum.*



RYSUNEK 4.2.

Fragment rekordu dziennika zdarzeń systemu Windows XP

Jak to zostało przedstawione na rysunku 4.2, ciąg znaków `LfLe`, czyli magiczna liczba rekordu dziennika zdarzeń, może być wykorzystany do identyfikacji poszczególnych rekordów w dzienniku zdarzeń. 4 bajty danych poprzedzające magiczną liczbę rekordu (na rysunku 4.2 wspomniane cztery bajty reprezentują wartość `0xE0`) określają rozmiar rekordu wyrażony w bajtach. Taka informacja jest bardzo istotna nie tylko podczas parsowania pliku dziennika zdarzeń na poziomie binarnym i wyodrębniania z pliku poszczególnych rekordów przy użyciu różnych narzędzi (w tym własnych skryptów), ale również może być wykorzystana do wyszukiwania i wyodrębniania rekordów dziennika zdarzeń ze zbiorów danych niemających wyraźnie zaznaczonej struktury, takich jak niealokowana przestrzeń dysku twardego czy plik wymiany, o których będziemy mówić nieco dalej w tym rozdziale.

Bardzo często zdarza się, że po wyodrębnieniu plików dzienników zdarzeń systemowych z utworzonego wcześniej binarnego obrazu dysku próba otwarcia takich plików w przeglądarce dziennika zdarzeń (ang. *Event Viewer*; w polskiej wersji systemu Windows XP program ten nosi nazwę *Podgląd zdarzeń*) kończy się wyświetleniem komunikatu o błędzie, informującego, że plik dziennika zdarzeń jest „uszkodzony". Najczęściej jednak w takiej sytuacji plik dziennika nie jest uszkodzony, a prawdziwa przyczyna pojawienia się takiego komunikatu jest związana z brakiem niektórych bibliotek DLL na komputerze, na którym przeprowadzamy analizę pliku. Aby sobie poradzić z tym problemem, napisałem kilka własnych narzędzi, które pozwalają na wydobycie z pliku dziennika zdarzeń informacji niezbędnych do przeprowadzenia jego analizy. Pierwszym z tych narzędzi jest napisany w języku Perl skrypt *evtrpt.pl*, który zbiera informacje o samych rekordach zdarzeń, takie jak częstość występowania poszczególnych rodzajów zdarzeń generowana według źródeł oraz identyfikatorów zdarzeń (ang. *event ID*). Poniżej zamieszczam fragment wyników działania tego skryptu dla dziennika zdarzeń aplikacji (plik *AppEvent.evt*):

```
Source              Event   ID Count
-------             -------  ------
SecurityCenter      1800    2
SecurityCenter      1807    192
Symantec AntiVirus  12      17
Symantec AntiVirus  14      17
Symantec AntiVirus  16      12
Symantec AntiVirus  53      3
```

Taki zestaw informacji pozwala na oszacowanie typów i liczby zdarzeń zapisanych w danym pliku dziennika. Jest to dobry sposób na szybką ocenę zawartości dziennika zdarzeń i sprawdzenie, czy w dzienniku zdarzeń znajdują się takie czy inne zdarzenia, które mogą mieć znaczenie dla przeprowadzanej analizy. Informacja o liczbie zdarzeń poszczególnych typów może być również przydatna do wielu innych celów. Na przykład jeżeli przeprowadzam analizę systemu związaną ze złośliwym oprogramowaniem i w dzienniku zdarzeń widzę szereg rekordów, których źródłem jest Symantec AntiVirus, to wiem, że taka aplikacja była zainstalowana w badanym systemie, co może mieć istotne znaczenie dla przeprowadzanej analizy. Taka informacja może być szczególnie przydatna w sytuacji, kiedy chciałbym zamontować binarny obraz dysku jako kolejny dysk twardy w moim systemie i wykonać pełny skan jednym z programów antywirusowych (w ramach jednego z etapów procesu wykrywania złośliwego oprogramowania, o którym będziemy mówili w rozdziale 6.). W takiej sytuacji, znając zainstalowany wcześniej program antywirusowy, mogę do wykonania skanu użyć innego programu. Analizę dzienników zdarzeń zazwyczaj rozpoczynam od sprawdzenia ustawień zasad inspekcji komputera (ang. *audit policy*), aby przekonać się, jakich rekordów zdarzeń mogę się spodziewać w dzienniku. Z drugiej strony zdarzają się sytuacje, gdy na przykład pomimo że rejestrowanie zdarzeń związanych z logowaniem do systemu zostało włączone w ustawieniach zasad inspekcji komputera, to system był włączony przez długi czas z otwartą konsolą i nikt nie musiał się do niego logować. W takiej sytuacji oczywiście w dzienniku *Zabezpieczenia* (ang. *Security Event Log*) nie było żadnych zdarzeń związanych z logowaniem nowych użytkowników.

Skrypt *evtrpt.pl* podaje również informacje na temat zakresu dat, jaki obejmuje badany dziennik zdarzeń, na przykład:

```
Date Range (UTC)
Thu Jan 18 12:41:04 2007 to Thu Feb 7 13:39:25 2008
```

Informacja o zakresie dat może być również bardzo użyteczna. Bardzo często zdarzało się, że byłem proszony o dostarczenie informacji o tym, którzy użytkownicy logowali się do systemu danego dnia lub w podanych ramach czasowych. Dzięki skryptowi *evtrpt.pl* mogłem szybko sprawdzić, czy w danym dzienniku zdarzeń znajdę jakieś rekordy z interesującego mnie okresu, czy też powinienem podarować sobie analizę dzienników zdarzeń i skoncentrować się na analizie artefaktów pochodzących z innych źródeł.

UWAGA

Logi programów antywirusowych

Większość powszechnie używanych programów antywirusowych generuje takie czy inne rodzaje logów. Wiele z nich jest zapisanych w prostym formacie tekstowym, dzięki czemu można je w łatwy sposób przeglądać i analizować, zwłaszcza jeżeli uda Ci się zaimportować taki plik do programu Excel. Bardzo często programy antywirusowe zapisują również swoje rekordy w dzienniku zdarzeń aplikacji (ang. *Application Event Log*), aczkolwiek czasami taka opcja musi dopiero zostać włączona przez użytkownika. Zdarza się również i tak, że program antywirusowy zapisuje wszystko w swoich logach i nie pozostawia żadnego śladu w dzienniku zdarzeń aplikacji.

Kolejnym narzędziem, którego bardzo często używam do parsowania rekordów dzienników zdarzeń systemowych, jest napisany w języku Perl skrypt *evtparse.pl*, analizujący zawartość plików dzienników zdarzeń na poziomie binarnym, lokalizujący poszczególne rekordy i wyodrębniający je bez wywoływania jakichkolwiek funkcji Windows API. Takie podejście do zagadnienia ma kilka niewątpliwych zalet, a jedną z najważniejszych jest to, że nie musisz się przejmować faktem, iż plik dziennika zdarzeń wydaje się „uszkodzony", co często zdarza się w przypadku narzędzi wykorzystujących standardowe funkcje API systemu Windows. Kolejną zaletą jest to, że skrypty w języku Perl nie są zależne od platformy, na której działają, dzięki czemu możesz bez problemu uruchamiać takie skrypty na komputerach pracujących pod kontrolą systemu Windows, Linux czy nawet Mac OS X. Skrypt *evtparse.pl* potrafi zapisywać wyniki działania w formacie CSV (jest on bardzo wygodny, jeżeli chcesz analizować wyniki działania w programie Excel) lub formacie TLN, wykorzystywanym do analizy zdarzeń w osi czasu (ang. *timeline analysis*), której zagadnienia będziemy szczegółowo omawiać w rozdziale 7.

Wyodrębnienie rekordów z dziennika zdarzeń to jednak dopiero połowa sukcesu. W internecie możesz znaleźć cały szereg doskonałych źródeł dostarczających szczegółowych informacji na temat tego, co poszczególne zdarzenia mogą znaczyć i jak należy je interpretować zarówno osobno, jak i w powiązaniu z innymi zdarzeniami. Jednym z moich ulubionych źródeł informacji o zdarzeniach jest portal EventID (patrz strona *http://www.eventid.net/*). Roczny abonament w wysokości 24 dolarów naprawdę nie jest wygórowany, a w zamian po zalogowaniu się do portalu pozwala nie tylko na wyszukiwanie i przeglądanie szczegółowych informacji o zdarzeniach związanych z systemami operacyjnymi firmy Microsoft, ale również na uczestniczenie w dyskusjach z innymi użytkownikami (zwykle są to administratorzy różnych systemów) na temat sytuacji i problemów, z jakimi zetknęli się w swoich środowiskach. Warto również wspomnieć, że informacje o poszczególnych zdarzeniach zawierają łącza prowadzące do odpowiednich artykułów z bazy wiedzy Microsoft, gdzie możesz znaleźć dodatkowe dane. Jeżeli poszukujesz informacji na temat zdarzeń generowanych przez określone aplikacje, to najlepiej sprawdzić w dokumentacji programu lub na stronie jego producenta, gdzie różnego rodzaju blogi i fora mogą stanowić prawdziwą kopalnię wiedzy w tym zakresie.

Kolejnym ciekawym źródłem informacji na temat zdarzeń, które możesz znaleźć w dzienniku *Zabezpieczenia* (ang. *Security Event Log*), jest portal Ultimate Windows Security Event Log (patrz strona *http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx*). Znajdziesz tam wyczerpującą listę zdarzeń wraz ze szczegółowymi objaśnieniami i wygodną wyszukiwarką. Na stronie są zamieszczone opisy zdarzeń, które są generowane w systemach Windows XP i Windows 2003, wraz z opisami ich odpowiedników w systemach Windows Vista oraz Windows 2008.

WSKAZÓWKA

Analiza dzienników zdarzeń

Przeprowadzając analizę systemu Windows, nie mam przygotowanej z góry listy zdarzeń, których poszukuję w każdym przypadku. Zazwyczaj szukam zdarzeń, które z takiego czy innego powodu mogą być związane z celami prowadzonego dochodzenia lub ekspertyzy. Lista poszukiwanych zdarzeń jest również mocno uzależniona od bieżących ustawień zasad inspekcji analizowanego systemu. Podczas mojej pracy bardzo często spotykałem się z systemami z domyślnie, fabrycznie ustawionymi zasadami inspekcji (lub z minimalnymi odstępstwami od ustawień domyślnych), ale zdarzały się również systemy, w których zasady inspekcji były znacząco zmodyfikowane (na przykład użytkownik zmieniał domyślny rozmiar plików dzienników zdarzeń systemowych). Kiedyś miałem nawet okazję analizować komputer działający pod kontrolą systemu Windows XP, na którym oprócz wielu innych ustawień, włącznie z rejestrowaniem zakończonych zarówno porażką, jak i sukcesem prób logowania, była włączona opcja śledzenia procesów (ang. *Process Tracking*). W tym konkretnym przypadku cele analizy wymagały odtworzenia pełnej aktywności systemu w osi czasu (o czym będziemy mówić w rozdziale 7.) i takie dodatkowe informacje z dziennika zdarzeń systemowych okazały się bezcenne.

Dzienniki zdarzeń systemowych nie zawsze są jedynym źródłem informacji na temat rekordów zdarzeń w systemie. Poszczególne rekordy dzienników zdarzeń możesz również znaleźć w innych miejscach, takich jak systemowy plik wymiany (ang. *pagefile*) czy niealokowane przestrzenie na dysku (ang. *unallocated space*). Kiedyś miałem okazję analizować system, gdzie w dziennikach zdarzeń systemowych było zapisanych bardzo niewiele rekordów, a w dzienniku *Zabezpieczenia* (ang. *Security Event Log*) znalazłem rekord zdarzenia o identyfikatorze 517, który wskazywał, że zawartość dzienników zdarzeń systemowych została wyczyszczona przez użytkownika. W takiej sytuacji jednym z etapów mojej analizy była próba odzyskania usuniętych rekordów zdarzeń. Aby to zrobić, najpierw przy użyciu narzędzia o nazwie *blkls.exe*, będącego częścią pakietu SleuthKit (patrz strona *http://www.sleuthkit.org/*) wyeksportowałem do postaci osobnego pliku całą niealokowaną przestrzeń dyskową z binarnego obrazu analizowanego systemu. Następnie załadowałem nowo utworzony plik do programu BinText (patrz strona *http://www.mcafee.com/us/downloads/free-tools/bintext.aspx*) i zapisałem listę zlokalizowanych ciągów znaków w osobnym pliku tekstowym. Kolejnym etapem było napisanie w języku Perl prostego skryptu, który przeszukiwał zawartość tego pliku tekstowego i wyszukiwał wszystkie wiersze zawierające „magiczny" ciąg znaków `LfLe`, identyfikujący rekordy dziennika zdarzeń. Program BinText oprócz znalezionego ciągu znaków podaje również jego położenie w pliku (*offset*) — podobną funkcjonalność po użyciu opcji `-o` ma program *strings.exe*, będący częścią dobrze znanego pakietu SysInternals (program możesz pobrać ze strony *http://technet.microsoft.com/en-us/sysinternals/bb897439*).

Dla każdego odnalezionego przez program BinText ciągu znaków `LfLe` skrypt pobierał jego położenie w pliku niealokowanej przestrzeni dyskowej (*offset*), „cofał się" o 4 bajty (wartość typu `DWORD`) i odczytywał rozmiar potencjalnego rekordu dziennika zdarzeń. Ponieważ każdy rekord dziennika zdarzeń rozpoczyna się i kończy tą samą sekwencją 4 bajtów reprezentujących rozmiar rekordu, skrypt odczytywał liczbę bajtów odpowiadającą rozmiarowi

rekordu i sprawdzał, czy pierwsza i ostatnia wartość typu DWORD jest taka sama. Jeżeli tak, odnaleziony ciąg bajtów był oznaczany jako poprawny rekord dziennika zdarzeń, wyodrębniany z pliku i analizowany (parsowany) pod kątem zawartości. Używając opisanej powyżej techniki, odnalazłem i wyodrębniłem ponad 330 rekordów usuniętych z dzienników zdarzeń. Innym sposobem na zrealizowanie takiego zadania mogłoby być pewne zmodyfikowanie skryptu *evtrpt.pl* lub *evtparse.pl* tak, aby przeszukiwał plik zawierający niealokowaną przestrzeń dyskową krokami co 4 bajty i w przypadku odnalezienia magicznego ciągu znaków sprawdzał jego poprawność i w razie pozytywnej weryfikacji odczytywał i analizował odnaleziony rekord. Jak widać, może to być bardzo wartościowa technika, zwłaszcza w sytuacji, kiedy musisz przygotować zestawienie aktywności systemu w osi czasu (ang. *timeline analysis*), o czym będziemy mówić w rozdziale 7. Przedstawiając ten przykład, chciałem pokazać, w jaki sposób znajomość różnych struktur danych w systemie Windows pozwala na odzyskiwanie dodatkowych informacji mogących mieć kolosalne znaczenie dla prowadzonej analizy.

WSKAZÓWKA

Ciekawe artefakty

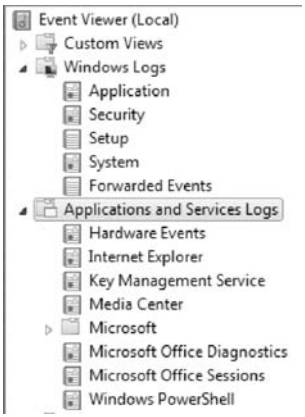
Jak już wspominałem, zazwyczaj nie korzystam z gotowej listy identyfikatorów zdarzeń, których poszukuję podczas każdej analizy systemu, ponieważ każdy przypadek jest inny i finalna lista poszukiwanych zdarzeń jest w dużej mierze determinowana przez cel prowadzonej analizy. Istnieje jednak szereg zdarzeń, które mogą być bardzo ciekawe w perspektywie analizy śledczej i powłamaniowej. Jednym z takich zdarzeń, o którym już wcześniej wspominałem, jest zdarzenie ID 517, wskazujące, że dziennik zdarzeń został wyczyszczony. Innym przykładem ciekawego zdarzenia jest ID 7035 (źródłem tego zdarzenia jest *Service Control Manager*), które jest generowane przez niektóre usługi systemowe podczas uruchamiania systemu Windows. Jeżeli znajdziesz takie zdarzenia, wskazujące na włączenie usługi systemowej przez użytkownika wiele godzin czy nawet dni po uruchomieniu systemu, może to oznaczać albo wykonywanie przez użytkownika standardowych operacji związanych z zarządzaniem systemem operacyjnym, albo może to być ślad wskazujący na przełamanie zabezpieczeń systemu i próbę zainstalowania i uruchomienia złośliwego oprogramowania. Idźmy dalej — administratorzy w wielu firmach i organizacjach często wykorzystują narzędzia takie jak *psexec.exe* (patrz strona *http://technet.microsoft.com/en-us/sysinternals/bb897553*) lub podobne (na przykład *rcmd.exe* firmy Microsoft) do uzyskiwania zdalnego dostępu do zarządzanych systemów. Użycie takiego narzędzia zazwyczaj powoduje zainstalowanie i uruchomienie związanej z nim usługi systemowej działającej w kontekście użytkownika, który uruchomił takie narzędzie. Uruchomienie usługi jest zwykle poprzedzane zdalnym zalogowaniem się do komputera (zdarzenie ID 540, typ 3 w dzienniku *Zabezpieczenia*).

Dziennik zdarzeń systemu Windows

W systemie Windows Vista firma Microsoft znacząco zmodyfikowała zarówno sposób rejestrowania zdarzeń, rodzaje rejestrowanych zdarzeń, lokalizację plików dzienników zdarzeń, jak i strukturę rekordów dzienników zdarzeń. Nowy mechanizm jest określany mianem *Dziennika systemu Windows* (w odróżnieniu od *Dziennika zdarzeń* znanego z systemów

Windows XP i 2003). W systemach Windows Vista oraz Windows 7 pliki dziennika systemu Windows są domyślnie przechowywane w folderze *C:\Windows\system32\winevt\Logs* i zapisywane w binarnym formacie XML (ang. *eXtensible Markup Language*).

Przykładowo na komputerze działającym pod kontrolą domyślnie zainstalowanego systemu Windows 7, na którym dodatkowo został zainstalowany pakiet MS Office 2007, znalazłem w katalogu *winevt\Logs* 134 różne pliki z rozszerzeniem *.evtx*. Nowe dzienniki zdarzeń są podzielone na dwie grupy: *Dzienniki systemu Windows* oraz *Dzienniki aplikacji i usług*. Na rysunku 4.3 został przedstawiony fragment okna aplikacji *Podgląd zdarzeń*, pokazujący nowe dzienniki.



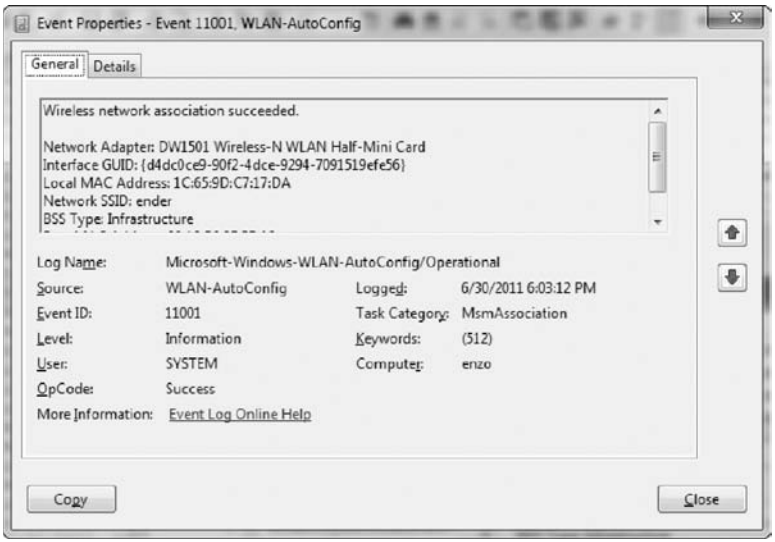
RYSUNEK 4.3.

Dzienniki systemu Windows 7 widoczne w oknie aplikacji Podgląd zdarzeń

Na rysunku 4.3 został zaprezentowany szereg nowych dzienników zdarzeń, które możesz znaleźć w systemie Windows Vista czy Windows 7. Na przykład dzienniki *Aplikacja* (ang. *Application*), *System* oraz *Zabezpieczenia* (ang. *Security*) odpowiadają znanym z systemu Windows XP/2003 plikom dzienników kolejno *AppEvent.evt*, *SysEvent.evt* oraz *SecEvent.evt*. Dziennik *Zabezpieczenia* (ang. *Security*) rejestruje bardzo wiele zdarzeń znanych wcześniej z systemu Windows XP, włącznie z logowaniem i wylogowaniem się z systemu (oczywiście w zależności od ustawień zasad inspekcji komputera). Warto jednak tutaj zwrócić uwagę na bardzo ważną zmianę — wiele zdarzeń znanych z poprzednich wersji systemu otrzymało zupełnie nowe identyfikatory. Na przykład w systemie Windows XP zdarzenie ID 528 oznaczało zalogowanie się użytkownika do systemu — w przypadku systemu Windows 7 takie samo zdarzenie ma teraz identyfikator 4624. Łatwo zauważyć, że różnica numeryczna pomiędzy tymi identyfikatorami wynosi dokładnie 4096 — ta zasada sprawdza się dla większości zdarzeń dziennika *Zabezpieczenia*. Na portalu Ultimate Windows Security znajdziesz wyczerpującą listę zdarzeń dziennika *Zabezpieczenia*, obejmującą zarówno rekordy zdarzeń systemu Windows XP, jak i Windows 7 (patrz strona *http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx*).

Na rysunku 4.3 są widoczne również takie dzienniki jak *Ustawienia* (ang. *Setup*) oraz *Zdarzenia przekazane* (ang. *Forwarded Events*). Zgodnie z dokumentacją firmy Microsoft dziennik *Ustawienia* zawiera informacje związane z konfiguracją aplikacji, aczkolwiek nawet pobieżny przegląd zdarzeń tego dziennika pokazuje, że znajdują się tam również zdarzenia związane ze statusem usługi Windows Update. Dziennik *Zdarzenia przekazane* jest dedykowany do przechowywania zdarzeń przekazywanych z innych systemów.

Pozostałe dzienniki znajdują się w grupie *Dzienniki aplikacji i usług* i przechowują zdarzenia generowane przez niektóre aplikacje i usługi, zatem nie znajdziesz tutaj raczej zdarzeń mających wpływ na funkcjonowanie całego systemu. Zdarzenia zapisywane w poszczególnych dziennikach są podzielone na cztery kategorie: *Operational*, *Admin*, *Analytic* oraz *Debug*. Po zainstalowaniu systemu Windows 7 z ustawieniami domyślnymi najczęściej będziesz się spotykał ze zdarzeniami w kategoriach *Operational* oraz *Admin*, aczkolwiek od czasu do czasu pojawi się również kategoria *Analytic*. Zdarzenia typu *Admin* są przeznaczone dla użytkowników końcowych oraz administratorów systemu i dostarczają informacji, na podstawie których administrator może rozwiązać problem, usunąć usterkę czy podjąć inną akcję. Zdarzenia typu *Operational* są najczęściej wykorzystywane do wyszukiwania i diagnozowania problemów. Na przykład w dzienniku *Microsoft-Windows-WLAN-AutoConfig/Operational* są zapisywane zdarzenia dostarczające informacji na temat kart sieciowych i sieci bezprzewodowych, z którymi łączył się dany system, tak jak to zostało przedstawione na rysunku 4.4. Informacje zapisane w rekordach takich zdarzeń mogą być bardzo przydatne nie tylko podczas diagnozowania problemów, ale mogą również dostarczać bezcennych wskazówek analitykowi prowadzącemu dochodzenie.



RYSUNEK 4.4.

Przykład zdarzenia zapisanego w dzienniku WLAN-AutoConfig/Operational

Dzienniki typu *Debug* i *Analytic* są przeznaczone dla deweloperów i wykorzystywane do diagnozowania problemów, które nie mogą być rozwiązane samodzielnie przez użytkownika.

WSKAZÓWKA

Maszyny wirtualne i wirtualne dyski twarde

Podczas pracy nad tą książką wielokrotnie miałem okazję testować wirtualne dyski twarde (ang. *VHD — Virtual Hard Disk*), montując je i odłączając od mojego systemu Windows 7 (patrz rozdział 3.). Warto zauważyć, że w dzienniku *Microsoft-Windows-VHDMP/Operational* możesz znaleźć szereg zdarzeń związanych z montowaniem plików dysków wirtualnych (zdarzenie o identyfikatorze ID 1) oraz odłączaniem takich dysków (zdarzenie o identyfikatorze ID 2). Z drugiej strony dziennik ten jest przeznaczony wyłącznie do przechowywania zdarzeń związanych z dyskami VHD. W dzienniku *Microsoft-Windows-Virtual PC/Admin* znajdziesz rekordy zdarzeń związanych z użyciem pakietu Virtual PC do tworzenia i uruchamiania maszyn wirtualnych, włącznie z użyciem „trybu XP", czyli dedykowanej wirtualnej maszyny Windows XP, pozwalającej na uruchamianie aplikacji, które mają problemy z kompatybilnością lub w ogóle nie chcą działać w systemie Windows 7. W tym dzienniku znajdziesz również informacje o aplikacjach, które zostały zainstalowane w trybie XP po uruchomieniu z poziomu systemu Windows 7. Oba dzienniki mogą dostarczyć analitykowi wielu cennych informacji, zwłaszcza jeżeli poszukuje plików, których nie ma w systemie plików Windows 7, ale które mogły znajdować się na wirtualnym dysku twardym lub w maszynie wirtualnej.

No dobrze, a czy są jakieś inne metody odczytywania danych z dzienników systemu Windows? Jednym ze sposobów, który bardzo dobrze sprawdził się w praktyce, jest zastosowanie programu LogParser firmy Microsoft (patrz strona *http://www.microsoft.com/en-us/download/details.aspx?id=24659*). Po zainstalowaniu programu można go użyć do analizy zawartości plików dzienników zdarzeń wyeksportowanych z binarnego obrazu dysku albo znajdujących się na zamontowanym wirtualnym dysku twardym. Po skopiowaniu plików dzienników zdarzeń do jednego katalogu możesz użyć następującego polecenia do wyeksportowania wszystkich rekordów zdarzeń z wybranego dziennika zdarzeń:

```
logparser -i:evt -o:csv "SELECT * FROM D:\Case\System.evtx" > output.csv
```

Korzystając z tej metody, powinieneś pamiętać, że LogParser używa wywołań funkcji API systemu Windows komputera, na którym jest zainstalowany. Z tego powodu, jeżeli pracujesz na komputerze działającym pod kontrolą systemu Windows XP, nie będziesz mógł na przykład wykorzystać go do analizy dzienników zdarzeń z systemu Windows Vista czy Windows 7, ponieważ funkcje API dzienników zdarzeń systemu Windows XP nie są kompatybilne z formatem plików dzienników zdarzeń systemów Windows Vista i Windows 7. Analogicznie, jeżeli używasz komputera pracującego pod kontrolą systemu Windows Vista lub Windows 7, nie będziesz mógł użyć programu LogParser do analizy dzienników zdarzeń z systemów Windows XP ani Windows 2003. Zapisywanie wyników działania programu w formacie CSV pozwala na otwarcie takiego pliku w programie Excel i dalszą analizę jego zawartości oraz dodawanie własnych uwag i komentarzy. Pliki danych zapisane w formacie CSV można również łatwo wykorzystywać w innych programach i skryptach, o czym przekonasz się w rozdziale 7.

WSKAZÓWKA

Konwersja formatów plików dzienników zdarzeń

Jak już wspomnieliśmy wcześniej, próba użycia programu LogParser zainstalowanego na komputerze pracującym pod kontrolą systemu Windows 7 do analizy plików dzienników zdarzeń z systemu Windows XP/2003 nie przyniesie niczego dobrego, chyba że wcześniej dokonasz konwersji dziennika zdarzeń z formatu Windows XP na format Windows 7. Aby to zrobić, powinieneś użyć narzędzia *wevtutil.exe* (wbudowane narzędzie systemu Windows 7) i wykonać polecenie przedstawione poniżej (wstawiając oczywiście odpowiednie nazwy plików i ewentualnie ścieżki):

```
D:\tools>wevtutil epl appevent.evt appevent.evtx /lf:true
```

Andreas Schuster, którego blog możesz znaleźć na stronie *http://computer.forensikblog.de/en/*, poświęcił bardzo wiele czasu i środków na badania nad rozpracowaniem formatu plików *Dziennika systemu Windows*. Efektem jego pracy jest znakomita biblioteka funkcji i procedur języka Perl oraz kolekcja narzędzi pozwalających na eksportowanie zdarzeń z dziennika. W czasie kiedy pracowałem nad tą książką, najnowsza wersja biblioteki nosiła numer 1.08[8]. W zależności od potrzeb możesz pobrać i zainstalować bibliotekę Andreasa lub skorzystać z narzędzi, które mają już tę bibliotekę zaimplementowaną, takich jak na przykład wirtualna stacja robocza SIFT (ang. *SANS Investigative Forensic Toolkit*), opracowana przez Roba Lee. W czasie powstawania tej książki ze strony *http://computer-forensics.sans.org/community/downloads* można było pobrać maszynę wirtualną ze stacją SIFT w wersji 2.1[9].

FOLDER RECYCLE BIN

Firma Microsoft doskonale zdaje sobie sprawę z tego, że użytkownikom zdarza się popełniać błędy, stąd systemy Windows mają wiele wbudowanych mechanizmów, pozwalających na odzyskiwanie na przykład przypadkowo usuniętych plików. Jednym z takich mechanizmów jest folder *Recycle Bin*, czyli inaczej mówiąc, popularny *Kosz systemowy*, który spełnia rolę repozytorium plików skasowanych w normalny sposób przez użytkownika (czyli na przykład poprzez naciśnięcie klawisza *Del* czy kliknięcie pliku prawym przyciskiem myszy i wybranie z menu podręcznego polecenia *Usuń* — warto przy tym pamiętać, że pliki usuwane ze zdalnych zasobów sieciowych czy usuwane za pomocą komend wydawanych z poziomu wiersza poleceń nie są wysyłane do tego folderu).

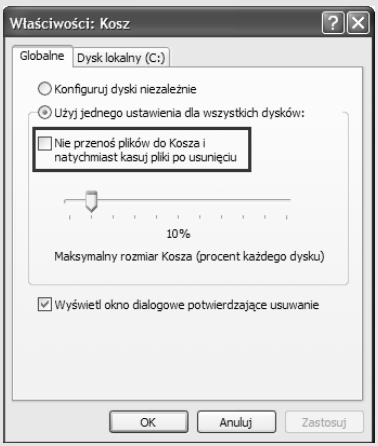
[8] W chwili powstawania polskiego tłumaczenia tej książki (grudzień 2012) dostępna była już wersja 1.1.1 — *przyp. tłum.*

[9] Obecnie najnowsza wersja to 2.14 — *przyp. tłum.*

WSKAZÓWKA

Pomijanie Kosza systemowego

Zgodnie z artykułem *http://support.microsoft.com/kb/320031* w razie potrzeby możesz wyłączyć mechanizm wysyłający skasowane pliki do Kosza systemowego. Aby to zrobić, powinieneś kliknąć ikonę Kosza prawym przyciskiem myszy i z menu podręcznego wybrać polecenie *Właściwości*. Na ekranie pojawi się okno właściwości Kosza systemowego, w którym powinieneś zaznaczyć opcję *Nie przenoś plików do Kosza i natychmiast kasuj pliki po usunięciu*, tak jak to zostało przedstawione na rysunku 4.5.

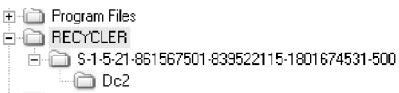


RYSUNEK 4.5.

Okno właściwości Kosza w systemie Windows XP

Zaznaczenie tej opcji powoduje utworzenie w rejestrze wpisu `NukeOnDelete` w kluczu `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\BitBucket` (o ile taki wpis wcześniej nie istniał) i ustawienie go na wartość 1. Jeżeli wspomniana wyżej opcja nie jest zaznaczona, wpis `NukeOnDelete` ma wartość 0. Mechanizm Kosza można wyłączyć globalnie dla wszystkich dysków lub selektywnie na wybranych woluminach. Jeżeli zatem podczas analizy badanego systemu pod kątem potencjalnie usuniętych plików stwierdzisz, że Kosz jest pusty, możesz sprawdzić, czy w rejestrze został utworzony wpis `NukeOnDelete` i jaką ma wartość.

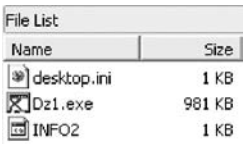
W systemie Windows XP kasowane pliki są przenoszone do folderu o nazwie *RECYCLER* i umieszczane w podkatalogu o nazwie reprezentującej identyfikator SID użytkownika (ang. *SID — Security Identifier*). Na rysunku 4.6 przedstawiono podkatalog utworzony w Koszu dla użytkownika Administrator (w systemie Windows XP).



RYSUNEK 4.6.

Folder RECYCLER systemu Windows XP wyświetlony w programie FTK Imager

Kiedy w systemie Windows XP plik zostanie usunięty i przeniesiony do Kosza, jego nazwa zostanie zmieniona zgodnie z zasadami opisanymi w bazie wiedzy firmy Microsoft, w artykule zatytułowanym *W jaki sposób Kosz przechowuje pliki* (patrz strona *http://support.microsoft.com/kb/136517*). Nazwa pliku jest zmieniana tak, że pierwszą literą nowej nazwy jest *D* (od angielskiego słowa *deleted*, czyli *usunięty*). Druga litera nazwy odpowiada literze dysku, na którym oryginalnie znajdował się usunięty plik. Następnie w nazwie pliku jest umieszczany kolejny numer usuniętego pliku, a cała nazwa kończy się oryginalnym rozszerzeniem pliku. Na rysunku 4.7 przedstawiono usunięty plik wykonywalny (z rozszerzeniem *.exe*), który przed skasowaniem znajdował się na dysku *Z:*.



RYSUNEK 4.7.

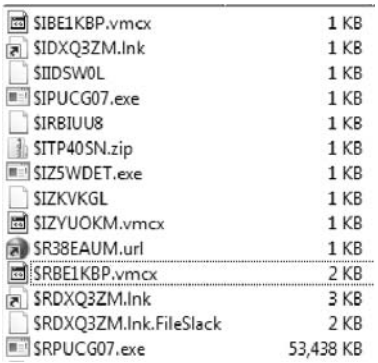
Skasowane pliki widoczne w folderze RECYCLER systemu Windows XP

Jak widać na rysunku 4.7, w folderze Kosza znajduje się również plik o nazwie *INFO2*, zawierający indeks skasowanych plików oraz informacje o ich oryginalnych nazwach, datach usunięcia i lokalizacjach, z których zostały usunięte. Do odczytywania informacji z plików *INFO2* możesz użyć napisanego w języku Perl skryptu *recbin.pl*, którego przykładowe wyniki działania przedstawiono poniżej:

```
C:\tools>recbin.pl -i d:\cases\info2
1 Mon Sep 26 23:03:27 2005 C:\Documents and Settings\jdoe\Desktop\lads.zip
2 Mon Sep 26 23:05:28 2005 C:\Documents and Settings\jdoe\LADS_ReadMe.txt
3 Mon Sep 26 23:05:28 2005 C:\Documents and Settings\jdoe\lads.exe
4 Mon Sep 26 23:23:58 2005 C:\Documents and Settings\jdoe\My Documents\Morpheus
Shared\Downloads\Toby Keith - Stays In Mexico.mp3
```

Po uruchomieniu skrypt *recbin.pl* przegląda kolejne rekordy zapisane w pliku *INFO2* i wyświetla na ekranie numery indeksów usuniętych plików, daty i czasy ich usunięcia oraz pełne ścieżki i oryginalne nazwy usuniętych plików.

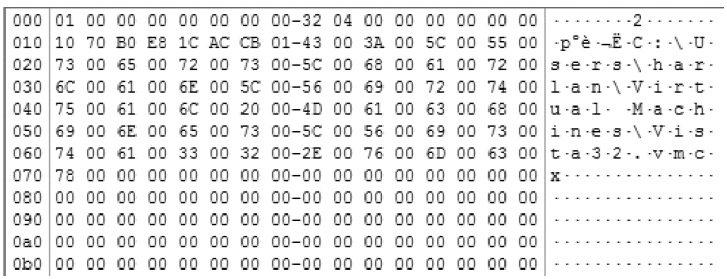
Wraz z wprowadzeniem systemu Vista firma Microsoft zmieniła format plików zapisywanych w Koszu. Kiedy pliki są usuwane z poziomu programu Windows Explorer, zostają domyślnie przeniesione do Kosza systemowego (folder *$Recycle.Bin*) i umieszczone w podkatalogu o nazwie reprezentującej identyfikator SID użytkownika. Skasowany plik otrzymuje nową nazwę, rozpoczynającą się od ciągu znaków *$R*, po którym następuje 6 kolejnych, losowo wybranych znaków. Nowa nazwa pliku kończy się oryginalnym rozszerzeniem. Dodatkowo dla usuniętego pliku jest tworzony plik indeksu, którego nazwa rozpoczyna się od ciągu znaków *$I*. Po nich następuje sześć kolejnych znaków pobranych z odpowiadającego mu pliku *$R*. Nazwa pliku indeksu kończy się takim samym rozszerzeniem jak plik *$R*. Na rysunku 4.8 zostało przedstawionych kilka usuniętych plików wraz z odpowiadającymi im plikami indeksów.



RYSUNEK 4.8.

Pliki znajdujące się w Koszu systemowym Windows 7 wyświetlone w programie FTK Imager

Na rysunku 4.9 została przedstawiona binarna zawartość przykładowego pliku indeksu. Każdy plik indeksu ma rozmiar 544 bajtów. Jak widać na rysunku 4.9, pierwsze osiem bajtów tego pliku zajmuje nagłówek pliku, a kolejnych osiem bajtów reprezentuje oryginalny rozmiar usuniętego pliku, zapisany w heksadecymalnym formacie *little-endian*[10]. Bajty od 16 do 23 zawierają standardowy, 64-bitowy obiekt `FILETIME`, reprezentujący datę i czas usunięcia pliku, a pozostałe bajty zawierają oryginalną ścieżkę i nazwę pliku, zapisane w formacie Unicode. Taka struktura pliku pozwala na łatwe odczytanie z niego potrzebnych informacji. Po odczytaniu informacji z pliku indeksu, którego nazwa rozpoczyna się od ciągu znaków *$I*, możesz odzyskać oryginalny plik z pliku, którego nazwa rozpoczyna się od ciągu znaków *$R*.



RYSUNEK 4.9.

Fragment pliku indeksu wyświetlony w widoku heksadecymalnym programu FTK Imager

[10] Format zapisu danych, w którym mniej znaczące bajty są zapisywane jako pierwsze (domyślny format zapisu danych dla procesorów rodziny Intel x86) — *przyp. tłum.*

PLIKI PREFETCH

Zdecydowana większość analityków dobrze zdaje sobie sprawę z wartości, jaką przedstawia dla prowadzącego dochodzenie mechanizm *prefetch*, a ściślej mówiąc, pliki tworzone przez ten mechanizm (dla uproszczenia będziemy je nazywać **plikami prefetch**). Podobnie jak w przypadku wielu innych artefaktów, pliki prefetch mogą dostarczyć analitykowi wielu ciekawych informacji na temat programów uruchamianych w systemie, nawet jeżeli użytkownik lub potencjalny napastnik poczyni jakieś kroki w celu ukrycia swoich działań.

Począwszy od wersji XP, systemy Windows są wyposażone w mechanizm prefetch, czyli mechanizm wstępnego ładowania programów. Wszystkie wersje systemu Windows mają zaimplementowany mechanizm wstępnego ładowania najczęściej używanych bibliotek podczas uruchamiania (ang. *Boot Prefetching*), ale tylko systemy Windows XP, Vista i Windows 7 domyślnie wykorzystują mechanizm wstępnego ładowania aplikacji (ang. *Application Prefetching*). Systemy Windows 2003 i Windows 2008 mogą używać mechanizmu wstępnego ładowania aplikacji po wprowadzeniu odpowiednich modyfikacji w rejestrze systemu.

WSKAZÓWKA

Włączanie mechanizmu wstępnego ładowania aplikacji (Prefetch)

Aby włączyć mechanizm wstępnego ładowania aplikacji, powinieneś przejść do klucza `CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters` znajdującego się w gałęzi `System` rejestru i zlokalizować wpis `EnablePrefetcher`[11]. Jeżeli wpis ten jest ustawiony na wartość 1 (wstępne ładowanie plików aplikacji) lub wartość 3 (wstępne ładowanie aplikacji oraz bibliotek podczas uruchamiania systemu), oznacza to, że mechanizm *Prefetch* jest włączony.

Mechanizm wstępnego ładowania aplikacji został zaimplementowany w celu polepszenia komfortu pracy użytkownika z systemem Windows poprzez monitorowanie uruchamianych aplikacji i tworzenie w jednym, specjalnie do tego celu wyznaczonym katalogu plików zawierających informacje o lokalizacji poszczególnych programów i bibliotek niezbędnych do uruchomienia aplikacji. Dzięki takiemu rozwiązaniu każde kolejne uruchomienie danej aplikacji przebiega znacznie szybciej, ponieważ Windows nie musi już przeglądać systemu plików w poszukiwaniu określonych bibliotek DLL i innych danych niezbędnych do uruchomienia aplikacji. Dzięki informacjom zawartym w plikach prefetch Windows dokładnie „wie", gdzie należy ich szukać. Pliki prefetch są zapisywane w katalogu *C:\Windows\Prefetch* i mają rozszerzenie *.pf*. Nazwy plików prefetch składają się z nazwy aplikacji, po której następuje myślnik i ośmiobajtowy ciąg znaków heksadecymalnych będący wartością funkcji skrótu utworzonej między innymi na podstawie oryginalnej ścieżki do pliku oraz argumentów wywołania aplikacji.

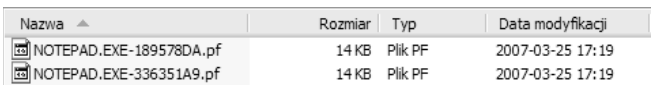
[11] Pełna ścieżka to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters\EnablePrefetcher` — *przyp. tłum.*

WSKAZÓWKA

Dyski SSD

Zgodnie z informacjami przedstawionymi na blogu *Engineering Windows 7* (patrz strona *http://blogs.msdn.com/b/e7/archive/2009/05/05/support-and-q-a-for-solid-state-drives-and.aspx*), jeżeli system Windows 7 wykryje, że został zainstalowany i działa na dysku SSD (ang. *Solid-State Drive*), niektóre mechanizmy, takie jak na przykład SuperFetch (który jest odpowiedzialny za tworzenie plików prefetch), zostają automatycznie wyłączone. W podanym artykule znajdziesz szczegółowy opis takiego zachowania systemu Windows 7, jak również listę innych funkcji i mechanizmów „siódemki", których działanie może zostać zablokowane lub zmodyfikowane po zainstalowaniu tego systemu na dysku SSD.

Aby zobaczyć przykład tworzenia pliku prefetch aplikacji, zwłaszcza jeżeli korzystasz z komputera działającego pod kontrolą systemu Windows XP, otwórz okno wiersza poleceń, przejdź do folderu *C:\Windows* i wpisz polecenie `notepad`. Na ekranie pojawi się okno dobrze Ci znanego notatnika systemowego. Zamknij je, powróć do okna wiersza poleceń, przejdź do katalogu *C:\Windows\system32*, ponownie wpisz polecenie `notepad` i ponownie zamknij okno notatnika systemowego, które pojawiło się na ekranie. Jeżeli teraz, po wykonaniu takich operacji, wejdziesz do katalogu *C:\Windows\Prefetch*, powinieneś tam znaleźć dwa różne pliki prefetch, których nazwy rozpoczynają się od *NOTEPAD.EXE* i zawierają dwie różne wartości funkcji skrótu, tak jak to zostało przedstawione na rysunku 4.10. Dokładnie z tego samego powodu w folderze *Prefetch* możesz znaleźć wiele plików dla aplikacji *RUNDLL32.EXE*.



RYSUNEK 4.10.

Dwa pliki prefetch dla programu Notepad.exe

Pliki prefetch zawierają szereg metadanych, które mogą być bardzo użyteczne dla analityka prowadzącego dochodzenie. Na przykład znajdziesz tam datę ostatniego uruchomienia aplikacji czy informację o tym, ile razy dana aplikacja została uruchomiona (ang. *run count*). W plikach prefetch znajdują się również dane woluminu, z którego aplikacja została uruchomiona, oraz lista bibliotek DLL i innych plików ładowanych przez aplikację podczas uruchamiania (zapisane w formacie Unicode). Istnieje bardzo wiele różnych narzędzi pozwalających na odczytywanie informacji zapisanych w plikach prefetch. Ze względu na ich ogromną użyteczność informacji zapisanych w tych plikach pokusiłem się nawet o napisanie w języku Perl własnego skryptu, o nazwie *pref.pl*, którego zadaniem jest parsowanie tych plików i wyświetlanie znalezionych w nich informacji (skrypt jest dostępny w materiałach dodatkowych przy-

gotowanych dla tej książki[12]). Podczas jednej z przeprowadzanych analiz znalazłem w badanym systemie nietypowy plik prefetch i postanowiłem sprawdzić jego zawartość za pomocą skryptu *pref.pl*. Poniżej zamieszczam fragment wyników działania skryptu:

```
C:\tools>pref.pl -f c:\windows\prefetch\0.8937919959151474.EXE-12EB1013.pf -p -i
c:\windows\prefetch\0.8937919959151474.EXE-12EB1013.pf Thu May 26
16:46:19 2011
(1)
EXE Name : 0.8937919959151474.EXE
Volume Path : \DEVICE\HARDDISKVOLUME1
Volume Creation Date: Fri Jan 1 22:24:09 2010 Z
Volume Serial Number: A424-CE42
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\NTDLL.DLL
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\KERNEL32.DLL
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\UNICODE.NLS
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\LOCALE.NLS
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\SORTTBLS.NLS
\DEVICE\HARDDISKVOLUME1\DOCUME~1\User\LOCALS~1\
TEMP\0.8937919959151474.EXE
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\USER32.DLL
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\GDI32.DLL
```

Jak widać na powyższym przykładzie, w plikach prefetch są przechowywane znaczne ilości metadanych. Na przykład możemy sprawdzić, ile razy aplikacja była uruchamiana (tylko jeden raz) i kiedy została uruchomiona po raz ostatni (*Thu May 26 16:46:19 2011*)[13]. Możemy również sprawdzić nazwę woluminu i ścieżkę, z której ta aplikacja została uruchomiona (w naszym przykładzie *\DEVICE\HARDDISKVOLUME1\DOCUME~1\User\LOCALS~1\TEMP\0.893791995915174.EXE*).

W wynikach działania skryptu *pref.pl* można znaleźć naprawdę bardzo interesujące informacje. Podczas pracy nad inną sprawą znalazłem jeszcze inny nietypowy plik prefetch, o nazwie *KARTCICYYIR.EXE-2CC557AD.pf*. Po „przepuszczeniu" tego pliku przez skrypt *pref.pl* w wynikach działania uzyskałem między innymi następujące informacje:

```
\DEVICE\HARDDISKVOLUME1\DOCUME~1\ABC\LOCALS~1\TEMP\KARCICYYIR.EXE
\DEVICE\HARDDISKVOLUME1\PROGRAM FILES\SOPHOS\SOPHOS ANTI-
VIRUS\SOPHOS_DETOURED.DLL
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\BDYAWUIS.DAT
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\CONFIG\SOFTWARE
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\MRYDUTAG.DAT
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\MBQTAEPO.DAT
\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\CMD.EXE
```

[12] Patrz strona *http://code.google.com/p/winforensicaanalysis/downloads/detail?name=wfa3e.zip&can=2&q=* — *przyp. tłum.*

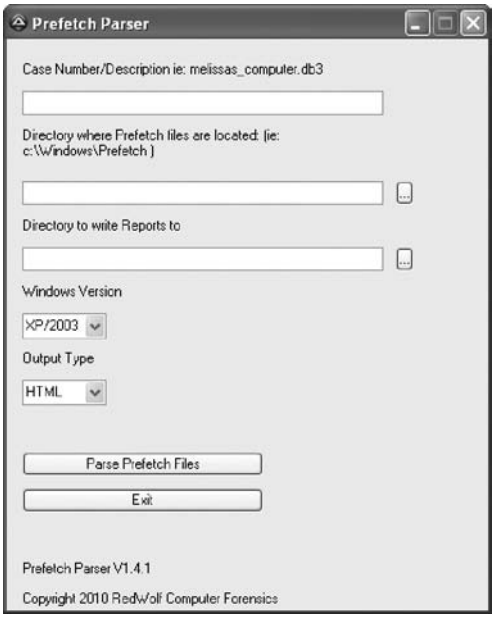
[13] Inaczej mówiąc, aplikacja została po raz ostatni uruchomiona w czwartek 26 maja 2011 r. o godzinie 16:46:19 — *przyp. tłum.*

I znów, podobnie jak w pierwszym przykładzie, jest to tylko niewielki fragment wyników działania skryptu, pokazujący interesujące artefakty, które od razu zwróciły moją uwagę. Mamy tutaj nie tylko pełną ścieżkę do pliku wykonywalnego, ale widzimy również, że aplikacja odczytywała trzy pliki z rozszerzeniami *.DAT* oraz gałąź `SOFTWARE` rejestru. Jest to doskonały przykład tego, jak bardzo pliki prefetch mogą być przydatne dla analityka prowadzącego dochodzenie, a przy tym ilustruje on ideę artefaktów pośrednich, o których wspominałem w rozdziale 1.

Plik prefetch przedstawiony w poprzednim przykładzie był artefaktem pośrednim, utworzonym po zainfekowaniu systemu złośliwym oprogramowaniem. Inaczej mówiąc, ten artefakt został utworzony przez system operacyjny w wyniku uruchomienia złośliwego oprogramowania i jego interakcji ze środowiskiem komputera. Jak się później okazało, w dzienniku zdarzeń *Aplikacja* (ang. *Application Event Log*) systemu, z którego pochodził wspomniany plik prefetch, znajdowały się rekordy zdarzeń wskazujące, że złośliwy, zainfekowany plik, *KARTCICYYIR.exe*, został wykryty i automatycznie usunięty przez oprogramowanie antywirusowe działające w tym systemie. Największą zaletą artefaktów pośrednich jest jednak to, że zazwyczaj nie są one usuwane przez użytkownika czy potencjalnego napastnika próbującego ukryć ślady swojej działalności w systemie i nie znikają, kiedy program antywirusowy automatycznie usuwa wykryte złośliwe oprogramowanie — niemal zawsze pozostaje jakiś ślad, czy to w postaci wpisów w rejestrze, plików prefetch czy wielu innych artefaktów. W omawianym przypadku metadane zapisane w pliku prefetch nie tylko pozwoliły nam na dokładne określenie, kiedy ten program został ostatnio uruchomiony, ale wskazały wolumin i pełną ścieżkę, z której program został uruchomiony. Metadane zapisane w pliku prefetch pozwoliły również na zidentyfikowanie innych plików potencjalnie powiązanych z badanym złośliwym oprogramowaniem i dostarczyły odpowiedniego kontekstu, pozwalającego analitykowi na lepsze rozpracowanie przebiegu całego zdarzenia (więcej informacji na temat zagadnienia kontekstu w analizie śledczej i powłamaniowej znajdziesz w rozdziale 7.).

Napisany w języku Perl skrypt *pref.pl* nie jest oczywiście jedynym narzędziem pozwalającym na analizę zawartości plików prefetch. Przykładem innego narzędzia jest program *PFDump.exe*, którego autorem jest Michael Spohn (patrz strona *http://malware-hunters.net/all-downloads*). Program ten jest częścią pakietu narzędzi przeznaczonych dla analityków zajmujących się wykrywaniem i analizą złośliwego oprogramowania. *PFDump.exe* potrafi odczytywać metadane z plików prefetch i pozwala na zapisywanie wyników działania w formacie *tab-delimited* (umożliwiającym łatwy eksport pliku wynikowego do Excela), formacie HTML oraz formacie XML. Jeżeli preferujesz narzędzia mające graficzny interfejs użytkownika, możesz skorzystać z programu *Prefetch Parser*[14], którego autorem jest Mark McKinnon (patrz strona *http://redwolfcomputerforensics.com/*). Mark jest również autorem kilku innych narzędzi, które możesz pobrać z jego strony internetowej. Wygląd głównego okna programu Prefetch Parser został przedstawiony na rysunku 4.11.

[14] Obecnie dostępna jest już wersja 1.5 tego programu — *przyp. tłum.*



RYSUNEK 4.11.

Okno programu Prefetch Parser, napisanego przez Marka McKinnona

Na rysunku 4.11 z pewnością zauważyłeś listę rozwijaną *Windows Version*. Dodanie tej opcji było konieczne ze względu na to, że zarówno sposób działania, jak i format plików prefetch w systemach Windows XP/2003 różni się od tego, jaki został zaimplementowany w systemach Windows Vista/7 (więcej informacji na ten temat znajdziesz na stronie *http://technet.microsoft.com/en-us/magazine/2007.03.vistakernel.aspx*). Warto jednak zauważyć, że jedyną zmianą, która ma duże znaczenie dla analizy śledczej i powłamaniowej, jest zmiana położenia (*offsetu*) w binarnej strukturze pliku prefetch elementów zawierających znacznik czasu ostatniego uruchomienia aplikacji oraz licznik uruchomień. Jeżeli używasz skryptu *pref.pl* (o którym mówiliśmy nieco wcześniej w tym rozdziale) do odczytywania metadanych z plików prefetch systemów Windows Vista lub Windows 7, powinieneś przy wywołaniu skryptu dodatkowo użyć opcji `-v`, która przełącza skrypt w tryb analizy plików prefetch tych systemów (domyślnie skrypt *pref.pl* jest skonfigurowany do analizy plików prefetch systemów Windows XP/2003). Jeżeli chcesz na własne oczy przekonać się, na czym polega różnica w offsetach położenia prefetch elementów zawierających znacznik czasu ostatniego uruchomienia aplikacji oraz licznik uruchomień w binarnej strukturze plików prefetch, możesz po prostu otworzyć skrypt *pref.pl* w dowolnym edytorze tekstu i odszukać odpowiednie miejsce w kodzie źródłowym.

WSKAZÓWKA

NTOSBOOT

Podczas analizy plików prefetch powinieneś zwrócić szczególną uwagę na plik *NTOSBOOT-BOODFAAD.pf*. Informacje zawarte w tym pliku mogą być odczytywane dokładnie w taki sam sposób jak w przypadku pozostałych plików prefetch — wspominam o tym pliku, ponieważ dosyć często zdarza się, że możesz w nim znaleźć odwołania (ścieżki) do złośliwego oprogramowania, które zainfekowało dany komputer.

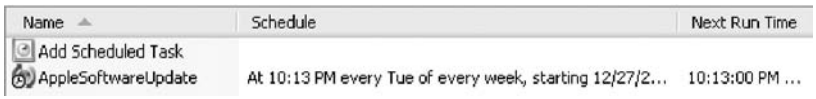
ZAPLANOWANE ZADANIA

Systemy Windows są bardzo wygodne w użytkowaniu i oferują cały szereg funkcji i mechanizmów ułatwiających życie użytkownikowi. Jednym z takich elementów jest możliwość wykonywania wcześniej zdefiniowanych zadań w ściśle określonej chwili. Mechanizm ten w systemie Windows nosi nazwę *Zaplanowane zadania* (ang. *Scheduled Tasks*), a użytkownik ma do niego dostęp na kilka sposobów, takich jak polecenie *at.exe* wykonywane z poziomu konsoli czy pracujący w trybie graficznym *Kreator zaplanowanych zadań*, przedstawiony na rysunku 4.12.

RYSUNEK 4.12.

Kreator zaplanowanych zadań w systemie Windows XP

Mechanizm zaplanowanych zadań pozwala na uruchamianie programów jednorazowo w wybranym czasie lub regularnie, w określonych z góry interwałach czasowych. Przykładem oprogramowania wykorzystującego ten mechanizm może być pakiet iTunes lub inne oprogramowanie firmy Apple, po zainstalowaniu którego w katalogu *C:\Windows\Tasks* najprawdopodobniej znajdziesz plik *AppleSoftwareUpdate.job*, tak jak to zostało przedstawione na rysunku 4.13.



RYSUNEK 4.13.

Zaplanowane zadanie o nazwie AppleSoftwareUpdate

Warto zauważyć, że sam fakt istnienia zaplanowanego zadania nie zawsze daje się bezpośrednio powiązać z aktywnością użytkownika, ponieważ takie zadania mogą być również tworzone w sposób programowy, poprzez wywołanie odpowiednich funkcji Windows API (a to z kolei może być wykonane również zdalnie, zakładając, że użytkownik zdalny ma odpowiednie uprawnienia). Jak widać, istnienie danego zaplanowanego zadania może być powiązane z instalacją jakiegoś pakietu oprogramowania, a czasami nawet z zainfekowaniem systemu złośliwym oprogramowaniem lub przełamaniem zabezpieczeń systemu. System Windows wymaga, aby użytkownik tworzący nowe, zaplanowane zadanie miał uprawnienia administratora. W momencie kiedy zaplanowane zadanie zostaje uruchomione, rozpoczyna działanie na prawach użytkownika *System*.

Takie rozwiązanie może być bardzo użyteczne dla administratora systemu, zwłaszcza w sytuacji, kiedy uprawnienia na poziomie użytkownika *System* są potrzebne tylko tymczasowo — administrator może utworzyć zadanie, które będzie wywoływało okno wiersza poleceń (na przykład *cmd.exe*), i natychmiast je uruchomić. Okno wiersza poleceń, które pojawi się na ekranie w wyniku takiej operacji, będzie działało na prawach użytkownika *System*, co pozwoli administratorowi na dostęp do obszarów systemu operacyjnego wymagających uprawnień na poziomie tego użytkownika. W artykule KB313565 bazy wiedzy firmy Microsoft (który znajdziesz na stronie *http://support.microsoft.com/kb/313565*) znajduje się szczegółowy opis tego, jak używać polecenia *at.exe* do tworzenia nowych zaplanowanych zadań. Artykuł ten został przygotowany dla systemu Windows 2000, ale polecenie `at` działa w analogiczny sposób we wszystkich nowszych wersjach systemu Windows.

W systemach Windows 2000, XP i 2003 pliki zaplanowanych zadań są przechowywane w katalogu *C:\Windows\Tasks* i mają rozszerzenie *.job*. Pliki są zapisane w formacie binarnym, którego szczegółowy opis możesz znaleźć na stronie *http://msdn.microsoft.com/en-us/library/cc248285.aspx*. Opis formatu plików *.job* znajdujący się na tej stronie jest na tyle dokładny, że z powodzeniem pozwala na tworzenie własnych narzędzi do analizy tych plików i odczytywania przechowywanych w nich informacji, które mogą mieć duże znaczenie dla analityka prowadzącego dochodzenie. Plik *.job* składa się z dwóch części, z których pierwsza ma stały rozmiar, a druga jest częścią o zmiennej długości. W stałej części takiego pliku możesz znaleźć na przykład 8-bajtowy obiekt `SYSTEMTIME`, zawierający znacznik czasu ostatniego uruchomienia zadania, a w części o zmiennej długości zapisany w formacie Unicode ciąg znaków wskazujący na właściciela (autora) zadania. Takie informacje mogą się okazać bardzo istotne dla prowadzonej ekspertyzy, ponieważ wszystko, co jest użyteczne dla administratora systemu, może być również użyteczne dla potencjalnego napastnika. Zaplanowane zadania dosyć często bywają wykorzystywane jako mechanizm umożliwiający przetrwanie złośliwego

oprogramowania w zainfekowanym systemie (ang. *persistence mechanism*; patrz rozdział 6.) czy mechanizm aktywacji trojanów, tylnych wejść do systemu (ang. *backdoors*), czy nawet niektórych standardowych usług systemowych, dających w efekcie napastnikowi zdalny dostęp do systemu.

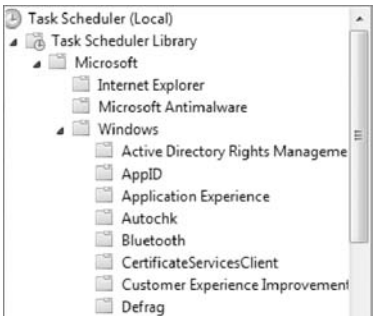
W systemie Windows 7 pliki *.job* są przechowywane w katalogu *\Windows\System32\Tasks* i jego podkatalogach i są zapisane w formacie XML, co oznacza, że możesz je otworzyć w dowolnym edytorze tekstu, takim jak choćby Notepad. Przykładowa zawartość takiego pliku, wyświetlona w programie ProDiscover, została przedstawiona na rysunku 4.14.

```
ÿþ<Task xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Source>$(@%systemroot%\system32\defragsvc.dll,-800)</Source>
    <Author>$(@%systemroot%\system32\defragsvc.dll,-801)</Author>
    <Description>$(@%systemroot%\system32\defragsvc.dll,-802)</Description>
    <URI>Microsoft\Windows\Defrag\ScheduledDefrag</URI>
  </RegistrationInfo>
```

RYSUNEK 4.14.

Fragment zawartości pliku zadania (.job) systemu Windows wyświetlonego w programie ProDiscover

W systemie Windows 7 podczas instalacji jest domyślnie tworzona całkiem spora liczba zadań. Na przykład zadanie *RegIdleBackup* jest wykonywane co 10 dni i tworzy kopie zapasowe plików rejestru, zapisując je w katalogu *\Windows\System32\config\RegBack*. Innym przykładem domyślnego zadania jest ograniczona defragmentacja dysków, która jest automatycznie wykonywana raz na tydzień. Zaplanowane zadania w systemie Windows 7 możesz przeglądać za pośrednictwem apletu *Harmonogram zadań*, znajdującego się w *Panelu sterowania* (ang. *Control Panel/Task Scheduler*), przedstawionego na rysunku 4.15.



RYSUNEK 4.15.

Fragment okna apletu Task Scheduler systemu Windows 7

Jak już wspominałem wcześniej, w systemie Windows 7 definicje poszczególnych zadań, zapisane w plikach XML *.job*), są przechowywane w strukturze podkatalogów katalogu *\Windows\System32\Tasks*.

Inną metodą tworzenia zaplanowanych zadań (oprócz polecenia *at.exe* i kreatora zaplanowanych zadań) jest użycie polecenia *schtasks.exe*. Narzędzie to zostało zaimplementowane już w systemie Windows XP i jest dostępne we wszystkich kolejnych wersjach systemu Windows (na przykład artykuł KB814596 bazy wiedzy Microsoft, dostępny pod adresem *http://support.microsoft.com/kb/814596*, opisuje, w jaki sposób można użyć tego narzędzia do tworzenia zaplanowanych zadań w systemie Windows 2003 Server). Zadania utworzone za pomocą polecenia *at.exe* (podobnie jak zadania utworzone za pomocą kreatora) są zapisywane w plikach o nazwie *AT#.job* (gdzie # to kolejny numer zadania), natomiast polecenie *schtasks.exe* pozwala na tworzenie zadań, których pliki mają pełne, opisowe nazwy.

OSTRZEŻENIE

Polecenie at.exe kontra schtasks.exe

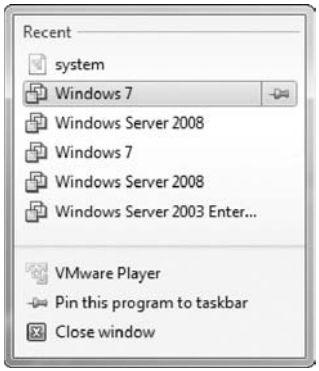
Kiedy rozpoczynasz działania związane z reakcją na incydent bezpieczeństwa i wykorzystujesz plik wsadowy (skrypt) do zbierania ulotnych informacji z działającego systemu Windows, powinieneś upewnić się, że do zbierania danych o zaplanowanych zadaniach używasz zarówno polecenia *at.exe*, jak i polecenia *schtasks.exe*. Okazuje się, że z jakiegoś powodu zadania utworzone przez pierwsze z tych narzędzi nie są „widziane" przez drugie narzędzie i odwrotnie.

Kolejnym, bardzo użytecznym źródłem informacji na temat zaplanowanych zadań jest plik dziennika (*log*) o nazwie *SchedLgU.txt*. Plik ten ma domyślnie rozmiar 32 kB i w systemach Windows 2003 i nowszych jest zlokalizowany w katalogu *\Windows\Tasks*, a w systemie Windows XP znajduje się w katalogu *\Windows*. Zazwyczaj w tym pliku są zapisywane data i czas włączania i wyłączania usługi *Harmonogram zadań* (ang. *Task Scheduler*). Pośrednio takie informacje mogą przyczynić się do ustalenia czasów włączania i wyłączania komputera, aczkolwiek ze względu na niewielkie domyślne rozmiary pliku dziennik ten obejmuje zwykle kilka czy kilkanaście ostatnich dni (najstarsze rekordy są usuwane, aby zrobić miejsce dla nowych wpisów).

W pliku *SchedLgU.txt* mogą być również zapisywane data i czas uruchomienia niektórych zadań, data i czas ich zakończenia oraz status wykonania (kod zakończenia działania, ang. *exit code*). W praktyce zdarzało mi się znajdować w tym pliku informacje o wykonaniu zadań, które były w taki czy inny sposób powiązane ze złośliwym oprogramowaniem czy próbami włamania do systemu. Zazwyczaj w takich przypadkach to zadanie było tworzone przez napastnika zdalnie, po uzyskaniu połączenia z zaatakowanym systemem za pośrednictwem skompromitowanego konta administratora systemu czy domeny. Po zakończeniu działania takie zadanie zwykle było automatycznie usuwane, niemniej jednak wpis w pliku dziennika *SchedLgU.txt* pozostawał i dzięki temu można było powiązać je z innymi zdarzeniami. Bardziej szczegółowe omówienie zagadnień związanych z analizą aktywności systemu w osi czasu (ang. *timeline creation and analysis*) znajdziesz w rozdziale 7.

LISTY SZYBKIEGO DOSTĘPU

Listy szybkiego dostępu (ang. *jump lists*) to nowy mechanizm wprowadzony w systemie Windows 7. Mówiąc w skrócie, listy szybkiego dostępu to listy plików ostatnio otwieranych przez użytkownika, pogrupowane według aplikacji używanych do otwierania tych plików (w podobny sposób pogrupowane są wpisy w rejestrze w kluczu `RecentDocs`[15]; więcej szczegółowych informacji na temat analizy rejestru znajdziesz w rozdziale 5.). Użytkownicy mogą wyświetlać listę ostatnio otwieranych dokumentów i plików poprzez kliknięcie prawym przyciskiem myszy ikony programu znajdującej się na pasku zadań. Na rysunku 4.16 przedstawiono listę szybkiego dostępu programu VMPlayer firmy VMware.



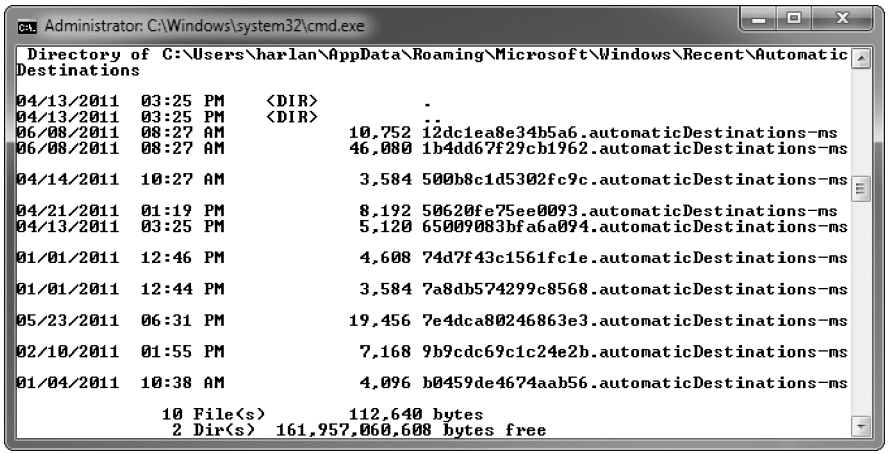
RYSUNEK 4.16.

Lista szybkiego dostępu programu VMPlayer

Zawartość listy szybkiego dostępu zależy od programu. Na przykład na liście szybkiego dostępu przeglądarki sieciowej Internet Explorer są wyświetlane adresy URL stron internetowych, a na podobnej liście programu MS Word znajdziesz zestawienie dokumentów otwieranych ostatnio przez użytkownika. Użytkownik może również na stałe „przypiąć" wybrane elementy do listy. Aby to zrobić, powinieneś kliknąć lewym przyciskiem myszy ikonę pinezki, znajdującą się po prawej stronie nazwy danego pliku czy dokumentu, tak jak to zostało zilustrowane na rysunku 4.16. Pozostałe elementy na liście będą się zmieniać w miarę upływu czasu, podczas gdy elementy „przypięte" zawsze będą dostępne. Listy szybkiego dostępu mogą być również wyświetlane dla programów widocznych w menu *Start* systemu.

Pliki zawierające listy szybkiego dostępu są przechowywane w profilu użytkownika, w katalogu *AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations*, tak jak to zostało zilustrowane na rysunku 4.17.

[15] Pełna ścieżka do tego klucza to `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs` — *przyp. tłum.*



RYSUNEK 4.17.

Zawartość folderu AutomaticDestinations znajdującego się w profilu użytkownika

Jak widać na rysunku 4.17, pliki zawierające listy szybkiego dostępu mają nazwy składające się z 16 znaków heksadecymalnych, po których następuje rozszerzenie *.automaticDestinations-ms*. Pierwsza, 16-znakowa część nazwy takiego pliku jest na stałe przypisana do danej aplikacji i jest taka sama we wszystkich instancjach systemu Windows 7. Na przykład ciąg znaków `b3f13480c2785ae` odpowiada programowi *Paint.exe*, ciąg znaków `adecfb853d77462a` jest związany z programem Microsoft Word 2007, a ciąg znaków `918e0ecb43d17e23` reprezentuje program *Notepad.exe*. Takie charakterystyczne ciągi znaków składają się na tak zwany **identyfikator aplikacji** (ang. *Application Identifier*, w skrócie *AppID*), który w unikatowy sposób reprezentuje daną aplikację, włącznie z pełną ścieżką do jej pliku wykonywalnego. Mark McKinnon z firmy RedWolf Computer Forensics, LCC, zamieścił na portalu *ForensicsWiki* obszerną listę identyfikatorów aplikacji — możesz ją znaleźć na stronie *http://www.forensicswiki.org/wiki/List_of_Jump_List_IDs*.

Środowisko analityków śledczych szybko zwróciło uwagę na fakt, że pliki zawierające listy szybkiego dostępu mają dosyć specyficzną strukturę. Jesienią 2008 roku Troy Larson, doświadczony analityk śledczy firmy Microsoft, w swoim wystąpieniu na dorocznej konferencji Microsoftu poświęconej zwalczaniu cyberprzestępczości ujawnił, że pliki list szybkiego dostępu są oparte na złożonym, binarnym formacie strukturalnym MS-CFB, który był wykorzystywany w dokumentach pakietu Microsoft Office przed wprowadzeniem wersji 2007 (szczegółowy opis tego formatu znajdziesz na stronie *http://msdn.microsoft.com/en-us/library/dd942138*). Taki złożony strukturalny format pliku jest często określany jako **system plików w pliku**, ponieważ w pliku binarnym zapisanym w tym formacie jest tworzony wewnętrzny minisystem plików, zawierający szereg wewnętrznych „plików" i „katalogów". Jednym ze sposobów przeglądania zawartości takich plików, zasugerowanym przez Roba Lee (SANS), jest otwarcie takiego pliku w programie *MiTeC Structured Storage Viewer* (patrz strona *http://mitec.cz/ssv.html*), tak jak to zostało przedstawione na rysunku 4.18.

RYSUNEK 4.18.

Plik listy szybkiego dostępu otwarty w programie MiTeC Structured Storage Viewer

Każdy z ponumerowanych strumieni danych widocznych w głównym oknie programu MiTeC Structured Storage Viewer jest zapisany w formacie znanym z plików skrótu systemu Windows (ang. *Windows shortcut files*). Jak zapewne pamiętasz, pliki skrótu, kiedy występują samodzielnie w swoim „naturalnym" środowisku, mają zazwyczaj nazwy zakończone rozszerzeniem *.lnk* (ang. *Link Files*). Firma Microsoft udostępniła szczegółowy opis formatu plików *.lnk* w dokumencie *[MS-SHLLINK]: Shell Link (.LNK) Binary File Format*, który możesz znaleźć na stronie *http://msdn.microsoft.com/en-us/library/dd871305*.

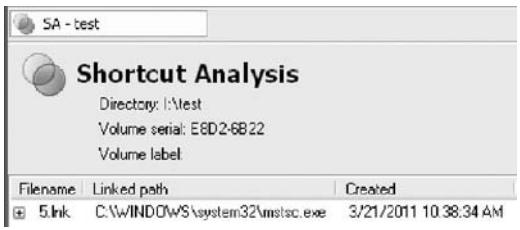
Ponieważ wspomniane strumienie danych są zapisane w formacie odpowiadającym plikom skrótu (LNK), zawierają znaczącą liczbę informacji, które mogą mieć ogromne znaczenie dla analityka prowadzącego dochodzenie. Na przykład znajdziesz tam znaczniki czasu ostatniej modyfikacji, czasu ostatniego dostępu oraz czasu utworzenia pliku docelowego (zapisane w formacie UTC). Inaczej mówiąc, kiedy jest tworzony dany strumień LNK listy szybkiego dostępu, znaczniki `MAC` pliku docelowego są wykorzystywane do wypełnienia odpowiednich znaczników czasu w strumieniu LNK listy. Jako analityk powinieneś zawsze pamiętać, że są to znaczniki czasu odnoszące się do pliku docelowego wskazywanego przez strumień danych LNK i że w żaden sposób nie są one powiązane z czasem utworzenia, ostatniego dostępu czy modyfikacji samego strumienia danych LNK.

Format strumienia danych listy szybkiego dostępu może również zawierać inne, dodatkowe informacje, takie jak na przykład argumenty podawane w wierszu wywołania pliku (o ile takie zostały użyte) czy też dodatkowy opis dokumentu. Przykładem strumienia LNK listy szybkiego dostępu zawierającego argumenty wiersza wywołania pliku oraz dodatkowy opis może być plik dostępu do zdalnego systemu dla klienta usługi terminalowej (ang. *Terminal Service*) systemu Windows 7, zaprezentowany poniżej (dane z pliku zostały odczytane za pomocą prostego skryptu napisanego w języku Perl):

```
Stream: 1
M: Tue Jul 14 00:01:53 2009
A: Tue Jul 14 01:14:27 2009
C: Tue Jul 14 00:01:53 2009
C:\Windows\System32\mstsc.exe /v:"192.168.1.24"
Connect to 192.168.1.24 with Remote Desktop Connection
```

Pozostałe strumienie LNK odczytane z pliku listy szybkiego dostępu usługi terminalowej tego systemu zawierały znaczniki czasu o dokładnie takich samych wartościach, reprezentujących czas modyfikacji, czas ostatniego dostępu oraz czas utworzenia pliku docelowego, czyli *C:\Windows\System32\mstsc.exe*. Pamiętaj również o tym, że począwszy od systemu Windows Vista, automatyczna aktualizacja czasu ostatniego dostępu do pliku została domyślnie wyłączona.

Poszczególne strumienie danych z pliku listy szybkiego dostępu mogą również być odczytywane za pomocą programów przeznaczonych do analizowania plików skrótu (ang. *shortcut/LNK file viewers*). Na przykład za pomocą programu Structured Storage Viewer możesz wyodrębnić i zapisać w pliku na dysku wybrany strumień danych. Następnie powinieneś zmienić rozszerzenie nazwy tego pliku na *.lnk*, uruchomić program *MiTeC Windows File Analyzer* (*WFA.exe*; program możesz pobrać ze strony *http://www.mitec.cz/wfa.html*) i otworzyć w nim folder, w którym zapisałeś wyodrębniony z listy szybkiego dostępu strumień danych. Program dokona analizy wszystkich plików *.lnk* znajdujących się w tym katalogu i wyświetli odczytane informacje na ekranie, tak jak to zostało przedstawione na rysunku 4.19.



RYSUNEK 4.19.

Informacje odczytane z pliku LNK, wyświetlone w programie Windows File Analyzer

Liczba informacji, które możesz odczytać ze strumienia danych pliku listy szybkiego dostępu, zależy od programu użytego do analizy plików LNK. Na przykład program *MiTeC Windows File Analyzer* nie wyświetla kolumny zawierającej dodatkowy opis pliku ani argumentów wiersza wywołania programu.

Dlaczego informacje zawarte w plikach list szybkiego dostępu mogą być przydatne dla analityka podczas prowadzonej ekspertyzy? Z prostego powodu — aby dana lista szybkiego dostępu została utworzona i wypełniona kolejnymi wpisami, użytkownik musi wykonać pewne operacje. Inaczej mówiąc, by powstał plik przedstawiony w poprzednim przykładzie, użytkownik musiał z menu *Start* systemu Windows 7 uruchomić program *Remote Desktop Connection* — stąd odnalezienie takiej informacji w listach szybkiego dostępu może stanowić dla analityka cenną wskazówkę (zwłaszcza w połączeniu z innymi informacjami) co do intencji i zachowania użytkownika. Nasz „użytkownik" może być lokalnym, prawowitym użytkownikiem komputera, ale równie dobrze może być potencjalnym napastnikiem, który w jakiś sposób uzyskał dostęp do komputera. Co najciekawsze i najważniejsze, artefakty w listach szybkiego dostępu mogą być zachowane na długo po wykonaniu danej operacji przez użytkownika, a nawet na długo po tym, jak plik docelowy zostanie usunięty.

WSKAZÓWKA

Strumień DestList

Na rysunku 4.18 widocznych jest kilka strumieni danych osadzonych w pliku listy szybkiego dostępu — dwa spośród nich są kolejno ponumerowane, a trzeci strumień nosi nazwę *DestList*. W dostępnych źródłach nie ma zbyt wielu informacji na temat struktury tego strumienia danych, aczkolwiek szczegółowa analiza ujawniła, że po zajmującym 32 bajty nagłówku kolejne rekordy strumienia *DestList* wydają się mieć uporządkowaną, spójną strukturę. Poszczególne rekordy tego strumienia są powiązane z kolejnymi, numerowanymi strumieniami LNK listy szybkiego dostępu i składają się ze 114 bajtów oraz ciągu znaków zapisanego w formacie Unicode. W tabeli 4.1 zamieszczono zestawienie poszczególnych elementów wchodzących w skład każdego z rekordów, obejmujące położenie w rekordzie (offset), rozmiar oraz krótki opis.

Tabela 4.1. Elementy składowe rekordu strumienia DestList

Offset (Dec/Hex)	Rozmiar	Opis
72 (0x48)	16 bajtów	Nazwa NetBIOS systemu, uzupełniona zerami do 16 bajtów.
88 (0x58)	8 bajtów	Numer strumienia; reprezentuje odpowiedni numerowany strumień LNK listy szybkiego dostępu.
100 (0x64)	8 bajtów	Obiekt `FILETIME`.
112 (0x70)	2 bajty	Liczba znaków w ciągu znaków Unicode następujących po tym elemencie; ze względu na fakt, że każdy znak Unicode zajmuje dwa bajty, długość tego ciągu w bajtach jest dwa razy większa niż liczba znaków ciągu.

Offsety poszczególnych elementów wymienionych w tabeli 4.1 są liczone od pierwszego bajta po 32-bajtowym nagłówku strumienia, a każdy z wymienionych elementów bezpośrednio sąsiaduje z następnym, bez żadnych separatorów pomiędzy nimi. 8-bajtowy obiekt `FILETIME` znajdujący się w jednym z elementów jest najprawdopodobniej wykorzystywany do sortowania listy według ostatnio używanych plików (ang. *MRU — Most Recently Used*) lub według najczęściej używanych plików (ang. *MFU — Most Frequently Used*). Szczegółowa budowa struktury tego strumienia jest nadal przedmiotem mozolnych badań wielu analityków. Jedna z metod polega na otwieraniu kilku różnych plików w kilku różnych aplikacjach (na przykład Microsoft Word, Adobe Reader, Microsoft Paint i inne), zapisywaniu dokładnego czasu wykonania takiej operacji, a następnie analizie zawartości całego pliku listy szybkiego dostępu, włącznie ze strumieniem *DestList*. Badania nad strukturą tego strumienia zostały zapoczątkowane przez Jimmy'ego Wega, analityka śledczego pracującego w wymiarze sprawiedliwości w stanie Montana, i zostały później potwierdzone i uzupełnione przez innych analityków, z autorem tej książki włącznie.

Listy szybkiego dostępu, o których mówiliśmy do tej pory, były zlokalizowane w folderze *AutomaticDestinations*. Użytkownik może jednak dodatkowo tworzyć dla wybranych plików i aplikacji swoje własne listy szybkiego dostępu, które są zapisywane w profilu użytkownika, w folderze *AppData\Roaming\Microsoft\Windows\Recent*. Pliki takich list mają rozszerzenie *.customDestinations-ms*. Podobnie jak w poprzednim przypadku, nazwy plików rozpoczynają się od 16-znakowego identyfikatora aplikacji (wstępne testy wykazały, że oba typy list

wykorzystują takie same identyfikatory aplikacji). Zgodnie z dokumentacją przygotowaną przez Troya Larsona własne listy szybkiego dostępu składają się z jednego lub więcej elementów zapisanych w formacie LNK (ang. *Shell Link Format*), ale bez zalet wynikających z rozdzielenia poszczególnych elementów na osobne strumienie LNK, jak to miało miejsce w przypadku list typu *.automaticDestinations-ms*.

Jak można się spodziewać, istnieje cały szereg narzędzi pozwalających na analizę zawartości list szybkiego dostępu. Mark Woan jest autorem nie tylko narzędzia pozwalającego na analizę plików skrótu (program lnkanalyzer, patrz strona *http://www.woanware.co.uk/?page_id=121*), ale również narzędzia umożliwiającego przeglądanie i pełną analizę plików list szybkiego dostępu (program JumpLister, patrz strona *http://www.woanware.co.uk/?page_id=266*). Oba narzędzia wymagają wcześniejszego zainstalowania pakietu .NET v4.0. W internecie znalazłem również zapowiedź innego narzędzia, Windows Jump List Extractor, którego autorem jest Alex Barnett, ale w momencie publikacji tej książki program nie był jeszcze dostępny.

Wykorzystując opublikowaną przez firmę Microsoft dokumentację formatów MS-CFB oraz LNK, napisałem (w języku Perl, a jakże!) swoje własne narzędzie do analizy list szybkiego dostępu. Kod składa się z dwóch modułów w języku Perl, z których jeden parsuje strumienie danych w formacie Windows LNK, a drugi listy szybkiego dostępu, zlokalizowane w folderze *AutomaticDestinations* (włącznie ze strumieniami *DestList*). Takie rozwiązanie daje mi dużą elastyczność w implementacji samego mechanizmu parsowania oraz sposobu prezentacji wyników działania programu. Na przykład korzystając z tych dwóch modułów (dosłownie, poprzez zastosowanie w języku Perl słowa kluczowego `use`), napisałem skrypt, który odczytywał pojedynczy plik listy szybkiego dostępu z folderu *AutomaticDestinations*, parsował strumień *DestList*, parsował wszystkie strumienie numerowane i następnie wyświetlał wyniki posortowane w kolejności MRU, jak to zostało zilustrowane poniżej:

```
Fri Apr 15 11:41:56 2011
C:\Windows\System32\mstsc.exe /v:" 192.168.1.12"
Tue Apr 5 16:26:19 2011
C:\Windows\System32\mstsc.exe /v:"192.168.1.10"
Wed Mar 16 18:45:58 2011
C:\Windows\System32\mstsc.exe /v:"ender"
Mon Feb 7 14:09:40 2011
C:\Windows\System32\mstsc.exe /v:" 192.168.1.7"
```

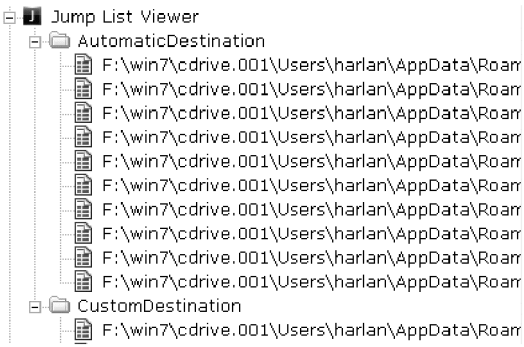
Informacje zamieszczone powyżej pochodzą z pliku listy szybkiego dostępu klienta RDP (ang. *Remote Desktop*) i przedstawiają połączenia, jakich dokonałem z mojego komputera działającego pod kontrolą systemu Windows 7 do innych systemów w moim laboratorium (kilka z nich to maszyny wirtualne). Takie informacje można w bardzo łatwy sposób przekształcić do formatu przydatnego podczas tworzenia zestawienia zdarzeń w osi czasu (patrz rozdział 7.).

OSTRZEŻENIE

Parser list szybkiego dostępu

Moduły bibliotek języka Perl oraz napisane przeze mnie skrypty przeznaczone do parsowania zawartości plików list szybkiego dostępu są dosyć surowe (w zasadzie najlepszym rozwiązaniem byłoby powiedzenie, że narzędzia te są jeszcze w wersji *alfa*) i kiedy powstawała ta książka, *nie* nadawały się jeszcze do udostępnienia szerokiej społeczności użytkowników. Z tego powodu wspomniane skrypty nie zostały załączone do materiałów dodatkowych tej książki. Co więcej, mam obawy związane z tym, że pomimo iż system Windows 7 jest już dostępny od dłuższego czasu, to jednak temat list szybkiego dostępu jest relatywnie nowy i jego znaczenie dla analizy śledczej nie jest jeszcze powszechnie znane. Z tych względów podjęcie decyzji o udostępnieniu narzędzia dostarczającego informacji pochodzących z list szybkiego dostępu wydawało mi się jeszcze zbyt pochopne — po prostu nie chciałem, aby używanie tego narzędzia bez dogłębnej znajomości natury list szybkiego dostępu prowadziło do konfuzji lub — co gorsza — wyciągania nieprawidłowych wniosków w prowadzonej sprawie. Mam jednak nadzieję, że być może już w niedalekiej przyszłości uda mi się wyczyścić i zoptymalizować kod tych programów oraz zwiększyć ich funkcjonalność, dzięki czemu będę mógł je udostępnić wszystkim zainteresowanym.

We wszystkich wersjach programu ProDiscover (niestety z wyjątkiem bezpłatnej wersji Basic Edition) jest dostępny wbudowany moduł analizy list szybkiego dostępu (ang. *Jump List Viewer*), przedstawiony na rysunku 4.20.



RYSUNEK 4.20.

Moduł Jump List Viewer programu ProDiscover

Aby uruchomić moduł analizy list szybkiego dostępu, powinieneś otworzyć wybrany projekt w programie ProDiscover, następnie kliknąć prawym przyciskiem myszy katalog profili użytkowników i z menu podręcznego, które pojawi się na ekranie, wybrać opcję *Find Jump List Files…*. ProDiscover rozpocznie skanowanie podkatalogów w poszukiwaniu plików list szybkiego dostępu zarówno typu *automatic*, jak i typu *custom* oraz analizę zawartych w nich danych (z wyjątkiem strumienia *DestList* w listach typu *automatic* — a przynajmniej, zgodnie z dokumentacją, nie było takiej możliwości w wersji ProDiscover 7.0.0.3).

PLIKI HIBERNACJI

W laptopach działających pod kontrolą systemu Windows XP lub Windows 7 bardzo często można znaleźć na dysku tzw. pliki hibernacji, czyli pliki, które zawierają skompresowany zrzut zawartości pamięci systemu Windows, tworzony w momencie, kiedy komputer (zazwyczaj właśnie laptop) przechodzi w tryb niskiego poboru mocy, inaczej mówiąc, w tryb hibernacji (uśpienia). Nietrudno sobie zatem wyobrazić, że pliki hibernacji zawierają bardzo wiele interesujących informacji, takich jak między innymi wszystkie procesy i połączenia sieciowe, które były aktywne w momencie utworzenia pliku hibernacji. Takie informacje mogą być niezwykle użyteczne na przykład podczas rozwiązywania problemów ze złośliwym oprogramowaniem, które zostało zainstalowane w badanym systemie i następnie usunięte. Oprócz tego, bazując na informacjach z pliku hibernacji, możesz na przykład udowodnić, że dany użytkownik był zalogowany w systemie, czy też pokazać, że w danym momencie była uruchomiona określona aplikacja. Podobnie jak wiele innych artefaktów, pliki hibernacji są bardzo często pomijane przez wszelkiego rodzaju aplikacje przeznaczone do „czyszczenia" systemu czy nawet do usuwania śladów aktywności użytkownika. Dane z plików hibernacji nie są również usuwane w przypadku odinstalowania aplikacji czy usunięcia zainfekowanej aplikacji przez program antywirusowy.

Do analizy pliku hibernacji możesz na przykład użyć pakietu Volatility Framework (patrz strona *http://code.google.com/p/volatility/*), który pozwala na przeglądanie zawartości tego pliku tak, jakby to był standardowy plik zawierający zrzut pamięci operacyjnej (ang. *memory dump*). Aby zainstalować pakiet Volatility Framework, powinieneś zajrzeć do sekcji Wiki tego projektu[16], gdzie znajdziesz szczegółową instrukcję postępowania (w czasie kiedy powstawała ta książka, na stronach Wiki można było znaleźć dokładną instrukcję instalowania pakietu w wersji 1.4, której autorem jest Jamie Levy, jeden z wolontariuszy pracujących nad projektem Volatility).

Szczegółowe omawianie zagadnień związanych z analizą zawartości pamięci operacyjnej systemu Windows wykracza daleko poza ramy tej książki, zwłaszcza że istnieje wiele znakomitych publikacji poruszających takie tematy, jak choćby wydana w roku 2011 książka *Malware Analyst's Cookbook and DVD* (wydawnictwo Wiley; patrz sekcja „Literatura i inne źródła" na końcu tego rozdziału). Warto jednak zapamiętać, że informacje znalezione w pliku hibernacji mogą mieć nieocenioną wartość dla analityka i prowadzonego przez niego dochodzenia — w wielu przypadkach analitycy znajdowali w tym pliku informacje, które okazywały się kluczowe dla sprawy, takie jak na przykład klucz do szyfrowanych woluminów dysku i inne.

[16] Patrz strona *http://code.google.com/p/volatility/wiki/VolatilityIntroduction?tm=6 — przyp. tłum.*

PLIKI APLIKACJI

W systemie Windows można znaleźć cały szereg plików charakterystycznych dla poszczególnych, ściśle określonych aplikacji, które mogą mieć ogromne znaczenie dla analityka prowadzącego sprawę. Informacje zawarte w niektórych „oczywistych" plikach, takich jak dzienniki aplikacji czy pliki konfiguracyjne, mogą być bardzo istotne, ale w praktyce rzeczywista wartość takich źródeł będzie mocno zależała od samej natury i celów prowadzonej analizy czy dochodzenia. W pozostałej części tego rozdziału omówimy niektóre pliki, z jakimi możesz się spotkać podczas analizy różnych systemów (oczywiście w zależności od tego, jakie aplikacje zostały wcześniej zainstalowane w badanym systemie). W każdym z przypadków pokażemy również aplikacje i narzędzia, których możesz użyć do analizy zawartości takich plików. Zanim jednak rozpoczniemy, chciałbym zastrzec, że nie będzie to w żaden sposób pełna, ostateczna i jedyna słuszna lista narzędzi, ponieważ opracowanie takiej listy jest po prostu niemożliwe. Deweloperzy i projektanci przez cały czas tworzą nowe narzędzia i mechanizmy pozwalające na przechowywanie informacji o zdarzeniach w dziennikach aplikacji czy danych konfiguracyjnych — dobrym przykładem mogą tutaj być wybrane przeglądarki sieciowe, które odeszły już od przechowywania historii przeglądanych stron czy zakładek w plikach tekstowych lub binarnych bezpośrednio na dysku i zamiast tego zapisują takie informacje w bazach danych SQLite.

WSKAZÓWKA

Dostęp do baz danych SQLite

Jednym z najlepszych narzędzi, jakie udało mi się znaleźć, przeznaczonych do odczytywania danych z baz danych SQLite, jest pakiet *SQLite Database Browser* (patrz strona *http://sqlitebrowser.sourceforge.net/*). Program jest darmowy i bardzo przyjazny dla użytkownika (zarówno podczas instalacji, jak i później). Z aplikacji można korzystać bezpośrednio z poziomu wiersza poleceń, ale użytkownik ma również do dyspozycji znacznie bardziej wygodny interfejs graficzny.

Ze względu na fakt, że niemal codziennie powstają nowe aplikacje, a dodatkowo często pojawiają się nowe wersje już istniejących pakietów, utrzymanie na bieżąco aktualizowanej listy jest zadaniem co najmniej trudnym, jeżeli nie praktycznie niemożliwym, nawet jeśli mielibyśmy się ograniczyć do spojrzenia na całe zagadnienie z perspektywy aplikacji przeznaczonych do analizy śledczej i powłamaniowej. Moim zamiarem jest po prostu zaprezentowanie alternatywnych rozwiązań i aplikacji, które mogą dostarczyć dodatkowych danych na poparcie tez, dowodów i wniosków w prowadzonych przez Ciebie analizach czy też uzupełnić brakujące fragmenty informacji. Na przykład dzienniki różnych aplikacji mogą być bardzo użyteczne, ponieważ w wielu przypadkach wpisy w takich dziennikach są tworzone wyłącznie w sytuacji, kiedy użytkownik jest zalogowany i korzysta z aplikacji. Takie informacje, w połączeniu na przykład ze zdarzeniami zapisanymi w dziennikach systemu Windows (lub brakiem takich zdarzeń), mogą nam pomóc nakreślić obraz aktywności użytkownika. Oczy-

wiście nie jestem tutaj w stanie omówić wszystkich dostępnych aplikacji (nawet pokrótce), ale zamiast tego spróbuję po prostu przedstawić wybrane typy plików, których analizę powinieneś rozważyć w swoich badaniach.

Logi programów antywirusowych

Logi tworzone przez programy antywirusowe będziemy co prawda bardziej szczegółowo omawiać w rozdziale 6., ale teraz chciałbym choćby pobieżnie poruszyć ten temat, dla zachowania spójności całego wywodu. Logi programów antywirusowych mogą mieć ogromne znaczenie dla analityka prowadzącego dochodzenie z bardzo wielu powodów. Podczas pracy nad jednym z ostatnich incydentów przeglądałem logi takiego programu i znalazłem wpis informujący o tym, że pewnego określonego dnia program antywirusowy wykrył i usunął kilka plików zidentyfikowanych jako złośliwe oprogramowanie. Kilka tygodni później pliki o takich samych nazwach ponownie pojawiły się w systemie, aczkolwiek tym razem nie zostały wykryte przez program antywirusowy (nie zostały zidentyfikowane jako złośliwe oprogramowanie). Była to dla mnie bardzo ważna informacja — wpisy w logu programu antywirusowego wskazały mi, jakich plików powinienem szukać, pozwoliły bardziej precyzyjnie wyznaczyć okno czasowe, w którym system był narażony na atak (ang. *window of compromise*), inaczej mówiąc, pozwoliły określić, jak długo złośliwe oprogramowanie działało w systemie. Takie informacje były bardzo istotne dla klienta, dla którego przeprowadzałem analizę, nie tylko dlatego, że ich określenie było wymagane przez organ nadzorczy firmy, ale również dlatego, że pozwoliło zredukować rozmiary okna czasowego infekcji (daty utworzenia drugiego zestawu plików zostały zweryfikowane poprzez dokładną analizę wpisów w tablicy MFT).

Logi programów antywirusowych mogą też (i to jest ich kolejne zastosowanie) wspomóc analityka w określeniu rodzaju złośliwego oprogramowania, które zagnieździło się w badanym systemie. Na przykład pakiet MRT firmy Microsoft (ang. *Microsoft Malicious Software Removal Tool*) jest często domyślnie instalowany na wielu komputerach i regularnie aktualizowany za pośrednictwem usługi Windows Update. Warto zauważyć, że MRT jest raczej aplikacją zaprojektowaną do usuwania określonych, specyficznych zagrożeń (szerzej będziemy o tym mówić w rozdziale 6.) niż programem antywirusowym ogólnego przeznaczenia. Analiza zawartości pliku *mrt.log* (który znajdziesz w katalogu *Windows\debug*) pozwoli Ci na zorientowanie się, kiedy aplikacja została zaktualizowana, oraz na poznanie wyników przeprowadzonych do tej pory skanów. Przykładowa zawartość logu programu MRT została przedstawiona poniżej:

```
Microsoft Windows Malicious Software Removal Tool v3.20, June 2011
Started On Wed Jun 15 21:13:25 2011
Results Summary:
----------------
No infection found.
Microsoft Windows Malicious Software Removal Tool Finished On Wed
Jun 15 21:14:4
5 2011
Return code: 0 (0x0)
```

Jak łatwo zauważyć, w pliku *mrt.log* są zapisane daty aktualizacji programu; możesz je porównać z datami aktualizacji publikowanymi w artykule KB891716 bazy wiedzy firmy Microsoft (patrz strona *http://support.microsoft.com/kb/891716*) i w ten sposób zorientować się, czy jeszcze jakieś inne aktualizacje *powinny* być zainstalowane. Zwróć uwagę, że w tym artykule znajdziesz również przykładowe zawartości logu MRT utworzone w sytuacji, kiedy zostanie wykryte złośliwe oprogramowanie.

Komunikator Skype

Skype jest bardzo użytecznym narzędziem komunikacyjnym, które funkcjonuje na rynku już od dłuższego czasu. Wiosną 2011 roku firma Microsoft zakupiła prawa do komunikatora Skype (za sumę około 8,5 miliarda dolarów). Skype działa na platformach Windows, Linux i Mac OS X, ale może być również zainstalowany i uruchamiany w produktach firmy Apple (iPhone, iTouch, iPad) oraz innych smartfonach i tabletach działających pod kontrolą systemu operacyjnego Android. Skype jest nie tylko dominującym na rynku narzędziem do połączeń wideo, ale także spełnia rolę komunikatora internetowego, pozwalającego na wymianę informacji poza „normalnymi", tradycyjnymi kanałami (takimi jak komunikatory AOL Instant Messenger, IRC i inne).

Jakiś czas temu zainstalowałem komunikator Skype na komputerze działającym pod kontrolą systemu Windows XP i przeprowadzałem szereg testów, mających na celu sprawdzenie samej aplikacji oraz narzędzi pozwalających na parsowanie logów tworzonych przez ten pakiet. W czasie pracy nad książką korzystałem z programu Skype w wersji 5.3[17], która zapisywała wszystkie zdarzenia w logu o nazwie *main.db* zlokalizowanym w katalogu *\Application Data\Skype\nazwa_użytkownika* w moim profilu. Przykładami narzędzi, które pozwalają na analizę informacji zapisanych w tym pliku, są *Skype Log View* (patrz strona *http://nirsoft.net/utils/skype_log_view.html*) oraz *Skype History Viewer* (patrz strona *http://skypehistory.sourceforge.net*). Na rysunku 4.21 przedstawiono fragment interfejsu użytkownika programu *Skype Log View* z wyświetloną zawartością pliku *main.db*.

Record Number	Action Type	Action Time
93	Outgoing Call	6/2/2010 7:56:02 PM
94	Outgoing Call	6/2/2010 7:56:02 PM
103	Outgoing Call	6/2/2010 7:56:02 PM
123	Incoming Call	6/10/2010 3:02:03 PM
149	Incoming Call	6/28/2010 9:48:46 AM
157	Outgoing Call	6/28/2010 9:50:39 AM
164	Outgoing Call	6/28/2010 9:53:34 AM
168	Chat Message	6/28/2010 12:13:03 PM
169	Chat Message	6/28/2010 12:13:49 PM
170	Chat Message	6/28/2010 12:13:51 PM
171	Chat Message	6/28/2010 12:13:54 PM

RYSUNEK 4.21.

Fragment interfejsu programu Skype Log View z wyświetloną zawartością pliku main.db

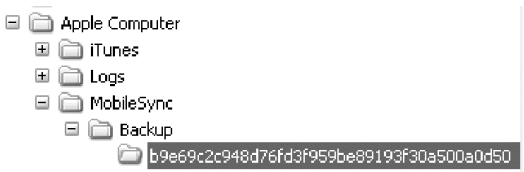
[17] Obecnie (grudzień 2012 r.) jest już dostępna wersja 6.0 — *przyp. tłum.*

Informacje zawarte w pliku *main.db* mogą być bardzo użyteczne dla analityka, który na ich podstawie może wskazać nie tylko na to, że komunikacja między dwoma użytkownikami miała miejsce, ale również może stwierdzić, kto dzwonił do kogo, o której godzinie itd. Zapisy z tego logu mogą też np. pośrednio wskazywać, kiedy dany komputer był włączony, czy też potwierdzić lub obalić oświadczenie użytkownika, że o takiej czy innej porze korzystał (lub nie) ze swojego komputera.

Produkty firmy Apple

Wielu z nas korzysta z bardzo popularnych obecnie produktów firmy Apple, takich jak iPod, iTouch[18], iPhone czy nawet tablet iPad. Wielu z nas zapewne używa również pakietu iTunes do synchronizacji danych i tworzenia kopii zapasowych danych przechowywanych na tych urządzeniach. W kwietniu 2011 roku dwóch analityków (Alasdair Allan i Pete Warden, patrz artykuł na stronie *http://radar.oreilly.com/2011/04/apple-location-tracking.html*) odkryło, że od momentu wprowadzenia systemu operacyjnego iOS 4 iPhone oraz iPad „śledzą" miejsce pobytu użytkownika, zapisując datę i czas oraz koordynaty geograficzne (długość i szerokość geograficzną) miejsca, w którym znajduje się urządzenie. Według autorów artykułu w telefonach iPhone takie informacje są zapisywane w pliku *consolidated.db*.

Kiedy użytkownik za pomocą pakietu iTunes synchronizuje swoje urządzenie Apple z komputerem działającym pod kontrolą systemu Windows XP, kopia zapasowa danych urządzenia jest zapisywana w profilu użytkownika, w katalogu *Application Data\Apple Computer\MobileSync\Backup* (w przypadku systemu Windows 7 jest to katalog *\Users\<**nazwa_konta_użytkownika**>\AppData\Roaming\Apple Computer\MobileSync\Backup*). Kiedy synchronizowałem odtwarzacz iTouch z komputerem z systemem Windows XP, dane były zapisywane w podkatalogu o nazwie składającej się z szeregu znaków heksadecymalnych, jak to zostało zilustrowane na rysunku 4.22.



RYSUNEK 4.22.

Ścieżka do podkatalogu MobileSync\Backup

Kopia zapasowa zapisywana w tym katalogu składa się z wielu plików, których nazwy są tworzone w podobny sposób (również składają się z ciągu znaków heksadecymalnych). Znajdziesz tam między innymi plik o nazwie *Info.plist*, zapisany w formacie XML (możesz otworzyć go do edycji w dowolnym edytorze tekstu), zawierający informacje o urządzeniu, którego kopia zapasowa jest tworzona, oraz pliki *Manifest.mbdb* i *Manifest.mbdx*, zawierające translację

[18] Potoczna nazwa odtwarzacza iPod Touch — *przyp. tłum.*

pomiędzy oryginalnymi nazwami plików i nazwami składającymi się z ciągów znaków heksadecymalnych.

Program iPhoneBackupBrowser (jego autorem jest niejaki reneD; program możesz pobrać ze strony *http://code.google.com/p/iphonebackupbrowser/*) jest bezpłatnym narzędziem, które pozwala na przeglądanie zawartości katalogów kopii zapasowych urządzeń Apple, tworzonych za pomocą pakietu iTunes, począwszy od wersji 9.11. Program automatycznie wykorzystuje informacje zawarte w pliku *Manifest.mbdb* i dokonuje translacji nazw plików do oryginalnej postaci. Strona wiki[19] na portalu Google Code zawiera szczegółowe informacje o strukturze formatu pliku nazw *Manifest.mbdb* oraz pliku indeksu *Manifest.mbdx* (na podstawie których możesz w razie potrzeby napisać własne narzędzie do parsowania tych plików). Program iPhoneBackupBrowser działa nieco lepiej w systemie Windows 7 niż w systemie Windows XP, co prawdopodobnie jest związane z niektórymi funkcjami API wykorzystywanymi przez ten program.

Po pobraniu pakietu i umieszczeniu go w wybranym katalogu możesz uruchomić narzędzie o nazwie *mbdbdump.exe* (działa całkiem nieźle również w systemie Windows XP) i jako parametru wywołania użyć ścieżki do katalogu zawierającego plik *Manifest.mbdb*, tak jak to zostało przedstawione poniżej:

```
D:\tools\iphone>mbdbdump [ścieżka do katalogu] > output.txt
```

Plik utworzony w wyniku działania tego polecenia będzie zawierał informacje odczytane z pliku *Manifest.mbdb*, które pozwolą Ci na wyszukiwanie określonych plików, na przykład *consolidated.db*.

Kiedy znajdziesz plik, którego nazwa składająca się z ciągu znaków heksadecymalnych odpowiada plikowi *consolidated.db*, możesz odczytać jego zawartość za pomocą narzędzi pozwalających na odczytywanie danych z baz danych SQLite, takich jak SQLite Browser (o którym już wspominałem wcześniej w tym rozdziale) czy dodatek SQLite Manager do przeglądarki Firefox (możesz go pobrać ze strony *https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/*). Po przeprowadzeniu bardziej dogłębnych badań analitycy stwierdzili, że informacje o dacie, czasie i koordynatach geograficznych miejsca, w którym znajduje się urządzenie, nie zawsze są bardzo precyzyjne, niemniej jednak patrząc z ogólnej perspektywy, z dosyć dużą dokładnością pokazują trasę, jaką urządzenie przebyło w wybranym czasie.

WSKAZÓWKA

Kopie zapasowe odtwarzacza iTouch

Choć moje testy przeprowadzałem na kopii zapasowej plików z odtwarzacza iTouch, a nie telefonu iPhone, to mimo to wyniki były bardzo interesujące. Po przeanalizowaniu wyników działania programu *mbdbdump.exe* mogłem bez problemu stwierdzić, jakie aplikacje były zainstalowane na urządzeniu, czy zlokalizować pliki konfiguracyjne np. sieci bezprzewodowych, do których się wcześniej łączyłem. Wszystkie tego typu informacje mogą mieć ogromną wartość dla analityka prowadzącego dochodzenie.

[19] Patrz strona *http://code.google.com/p/iphonebackupbrowser/wiki/MbdbMbdxFormat — przyp. tłum.*

Analityk może wykorzystać dane zapisane w kopiach zapasowych urządzeń Apple do uzupełnienia informacji o urządzeniach podłączanych do komputera, przeanalizowania aktywności użytkownika w sieci WWW, a nawet do sprawdzenia jego miejsc pobytu (a w zasadzie do sprawdzenia miejsc pobytu badanego urządzenia, co nie zawsze musi być równoznaczne z miejscem pobytu użytkownika) w określonym czasie.

UWAGA

Urządzenia pracujące pod kontrolą systemu Android

Okazuje się, że produkty firmy Apple to nie jedyne urządzenia, które mogą rejestrować informacje o swoim położeniu. W kwietniu 2011 roku Cory Altheide (z firmy Google) zwrócił moją uwagę na stronę użytkownika *packetlss/Android-locdump* (patrz *https://github.com/packetlss/android-locdump*), gdzie autor opisuje pliki, w których urządzenia pracujące pod kontrolą systemu Android (aczkolwiek bez wymieniania nazw urządzeń) najwyraźniej przechowują informacje o położeniu poszczególnych sieci Wi-Fi oraz wież przekaźnikowych telefonii komórkowej. Po przeczytaniu tego artykułu od razu sprawdziłem mojego BackFlipa (Motorola MB300, system Android z jądrem 2.6.29), ale nie znalazłem plików opisywanych przez autora (aczkolwiek w artykule znajduje się wzmianka o tym, że aby zobaczyć te pliki, musisz pracować na prawach użytkownika *root*). Na stronie znaleźć można również skrypt napisany w języku Python, który parsuje informacje zapisane w plikach *cache.wifi* oraz *cache.cell* i wyświetla je w formacie przyjaznym dla użytkownika. Podobnie jak w przypadku urządzeń firmy Apple, takie informacje mogą mieć ogromne znaczenie dla prowadzonego dochodzenia. Co ciekawe, jeżeli użytkownik tworzył kopie zapasowe swojego Androida, to istnieje spora szansa na to, że możesz znaleźć takie informacje w plikach kopii zapisanych na dysku twardym badanego komputera.

Pliki graficzne (zdjęcia, obrazy)

Jestem szczęśliwym posiadaczem zarówno służbowego, jak i prywatnego telefonu komórkowego. Oba modele telefonów mogę w skrócie określić mianem smartfonów i tak jeden, jak i drugi ma wbudowany cyfrowy aparat fotograficzny. Oprócz tego mam odtwarzacz iTouch, który również jest wyposażony w kamerę. Każde z tych urządzeń mogę podłączyć bezpośrednio do komputera i skopiować zapisane w nich zdjęcia i filmy do wybranego folderu na dysku. Zdecydowana większość produkowanych obecnie podobnych urządzeń ma wbudowany taki czy inny aparat cyfrowy, a wiele z nich ma możliwość robienia zdjęć i filmów w jakości HD. Co więcej, bardzo wiele urządzeń ma również wbudowane odbiorniki GPS (ang. *Global Positioning System*) — niedawno jeden z moich kolegów zrobił zdjęcie swoim telefonem pracującym pod kontrolą systemu Android, kilka razy przesunął palcem po ekranie dotykowym i na wyświetlaczu pokazała się mapa Google z zaznaczonym miejscem wykonania tego zdjęcia. Ten przykład dobrze pokazuje, że zwykłe, wydawałoby się, zdjęcia cyfrowe mogą zawierać znaczną liczbę dodatkowych informacji zapisanych w postaci metadanych. Powstaje zatem zasadnicze pytanie: jak można uzyskać dostęp do tych informacji?

Jednym z naprawdę znakomitych narzędzi, pozwalających na odczyt metadanych z wielu różnych typów plików, jest program EXIFTool, którego autorem jest Phil Harvey (patrz strona *http://www.sno.phy.queensu.ca/~phil/exiftool/*). Format *EXIF* (ang. *Exchangeable Image File*)

to standard definiujący sposób zapisu znaczników metadanych w plikach multimedialnych tworzonych przez aparaty cyfrowe, a także smartfony i skanery. Narzędzie napisane przez Phila jest w stanie odczytywać metadane EXIF z plików zapisanych w wielu różnych formatach. Program możesz wywoływać bezpośrednio z wiersza poleceń, tak jak to zostało zaprezentowane poniżej:

```
D:\tools>exiftool -a -u -g1 D:\pictures\img_3791_2165.jpg
---- ExifTool ----
ExifTool Version Number     : 8.60
---- System ----
File Name                   : img_3791_2165.jpg
Directory                   : D:/pictures
File Size                   : 4.0 MB
File Modification Date/Time : 2010:07:18 15:32:06-04:00
File Permissions            : rw-rw-rw-
---- File ----
File Type                   : JPEG
(...)
---- IFD0 ----
Make                        : Canon
Camera Model Name           : Canon EOS DIGITAL REBEL XTi
Orientation                 : Horizontal (normal)
```

Jak zapewne zauważyłeś, w pewnym miejscu wstawiłem ciąg znaków (`...`), który oznacza, że pewna część wyników została pominięta — nie jest to oczywiście wynik działania żadnego aparatu ani programu. Ponieważ badane zdjęcie zrobiłem osobiście, mogłem zweryfikować poprawność działania programu — rzeczywiście, zdjęcie zostało zrobione aparatem Canon EOS Digital Rebel Xti.

W dalszej części wyników działania programu znalazłem następujące informacje:

```
Canon Image Type        : Canon EOS DIGITAL REBEL XTi
Canon Firmware Version  : Firmware 1.1.1
Owner Name              : unknown
Serial Number           : 2271247134
Canon Model ID          : EOS Digital Rebel XTi / 400D / Kiss Digital X
(...)
Internal Serial Number  : H3624774
```

Jak łatwo zauważyć, mamy tutaj dwa potencjalne numery seryjne mogące w unikatowy sposób zidentyfikować konkretny egzemplarz aparatu, którym zostało wykonane zdjęcie. A zatem, jeżeli użytkownik skopiował zdjęcie z aparatu na dysk twardy, a sam aparat jest dostępny do zbadania, to analityk może wykorzystać informacje zapisane w metadanych i użyć ich (w połączeniu oczywiście z innymi danymi, takimi jak informacje zapisane w rejestrze, wskazujące, że taki aparat był podłączany do danego komputera itp.) do jednoznacznego powiązania zdjęcia z konkretnym aparatem i konkretnym użytkownikiem.

Jakie inne informacje mogą być zapisywane w plikach zdjęć? Jak już wspominałem, bardzo wiele współczesnych smartfonów i aparatów cyfrowych ma wbudowane moduły GPS,

które osadzają w zdjęciach metadane zawierające dokładne koordynaty geograficzne miejsca wykonania zdjęcia. Narzędzie EXIFTool jest w stanie odczytywać takie informacje, jak również całą masę innych danych osadzanych w zdjęciach zapisanych w różnych formatach plików.

WSKAZÓWKA

Metadane w innych plikach

Narzędzie EXIFTool jest w stanie odczytywać metadane zapisywane nie tylko w plikach graficznych, ale również w dokumentach tworzonych przez wiele różnych programów. Według informacji, które możesz znaleźć na stronie autora, EXIFTool może na przykład odczytywać metadane z dokumentów pakietu Microsoft Office 2007 (np. **.docx*, **.pptx* czy **.xlsx*), dzięki czemu staje się dla analityka bardzo wartościowym narzędziem. Innym narzędziem, które pozwala na odczytywanie metadanych z dokumentów pakietu Microsoft Office 2007, jest skrypt *read_open_xml.pl*, który możesz pobrać ze strony *http://snortdlp.googlecode.com/svn-history/r115/trunk/src/python/read_open_xml.pl*. Informacje, które możesz uzyskać za pomocą tych narzędzi, mogą być bardzo użyteczne (aczkolwiek ich przydatność w głównej mierze zależy od rodzaju prowadzonego dochodzenia).

Na koniec krótkie podsumowanie naszej dyskusji na temat metadanych. Bez żadnych wątpliwości możemy stwierdzić, że metadane mogą mieć ogromne znaczenie dla prowadzonego dochodzenia i mogą okazać się dla analityka prawdziwą kopalnią interesujących informacji. Powinieneś jednak pamiętać, że nie wszystkie pliki zawierają metadane, a co więcej, nawet jeżeli dany format pliku wspiera zapisywanie metadanych, to nie zawsze takie metadane są w pliku osadzane. Krótko mówiąc, jeżeli nie jesteś w stanie odczytać z jakiegoś pliku metadanych, to zwykle oznacza to, że ich po prostu tam nie ma. Na przykład zdjęcia cyfrowe wykonywane za pomocą telefonów komórkowych czy smartfonów nie zawsze mają osadzone koordynaty GPS miejsca wykonania zdjęcia. Przyczyn takiego stanu rzeczy może być co najmniej kilka, na przykład dany model telefonu czy aparatu nie ma wbudowanego modułu GPS, oprogramowanie użyte do wykonania zdjęcia nie zapisuje koordynatów GPS w pliku albo po prostu użytkownik wyłączył tę funkcję.

PODSUMOWANIE

W systemach Windows można znaleźć tysiące plików zapisanych w różnych formatach, zarówno otwartych, jak i zastrzeżonych. W zależności od rodzaju sprawy, nad którą pracujesz, lub od tego, czego naprawdę szukasz, takie czy inne pliki mogą mieć dla Ciebie mniejsze lub większe znaczenie. Moim zamiarem w tym rozdziale było zwrócenie Twojej uwagi na pewne wybrane rodzaje plików i na ich potencjalne znaczenie dla prowadzonego dochodzenia.

Jednym ze szczególnie ważnych zagadnień poruszanych w tym rozdziale był fakt, że pliki mogą być czymś więcej niż tylko binarnymi strumieniami danych. Na przykład jeżeli znasz strukturę danych zapisanych w pliku i wiesz, w jaki sposób takie struktury są wykorzystywane przez aplikację czy nawet przez system operacyjny, to możesz z tego wyciągnąć znacznie więcej

wniosków, niż wynikałoby z samych danych (możesz poznać kontekst danych). Kiedyś pracowałem z pewnym analitykiem, który w przenośnym pliku wykonywalnym w systemie Windows (ang. *PE — Portable Executable*; struktura takich plików została szczegółowo omówiona w drugim wydaniu mojej książki *Windows Forensic Analysis*) znalazł dosyć szczególny ciąg znaków, reprezentujący nazwę pliku. Zanim to odkrycie znalazło się w końcowym raporcie, musieliśmy przeprowadzić dalsze analizy mające na celu uzupełnienie kontekstu tego odkrycia, na przykład, czy ta nazwa pliku odnosiła się do biblioteki DLL wymienionej w tabeli importowanych funkcji API, czy może raczej do pliku wykorzystywanego jako repozytorium danych? Odpowiedzi na takie pytania, aczkolwiek niezbyt trudne do znalezienia, mogą mieć bardzo znaczący wpływ na dalszą analizę badanego systemu oraz wnioski, jakie zostaną umieszczone w końcowym raporcie.

LITERATURA I INNE ŹRÓDŁA

- Carrier B., *File System Forensic Analysis*, Pearson Education, Upper Saddle River 2005.
- Carvey H., *Windows Forensic Analysis (2nd ed.)*, Syngress Publishing, Inc., Burlington 2009.
- Ligh M. H., Adair S., Hartsteing B., Richard M., *Malware Analyst's Cookbook and DVD*, Wiley, New York 2011.

Skorowidz

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

Ś

T

U

V

W

Y

Z

Obowiązkowa lektura dla każdego inżyniera bezpieczeństwa systemów!

Kiedy spełni się najgorszy sen każdego administratora i użytkownika, kiedy zostaną przełamane wszystkie zabezpieczenia i intruz dostanie się do systemu, trzeba działać szybko i precyzyjnie. Jednym z kluczowych aspektów jest przeprowadzenie analizy powłamaniowej. Dzięki niej można ocenić skalę szkód i słabe punkty zabezpieczeń oraz podjąć działania mające na celu zapobieżenie takim incydentom w przyszłości.

W Twoje ręce oddajemy kolejne wydanie niezwykłej książki, napisanej przez jednego z najlepszych specjalistów w zakresie analizy śledczej i powłamaniowej. W trakcie lektury zdobędziesz szczegółową wiedzę na temat metod prowadzenia analizy w systemie Windows. Dowiesz się, jak analizować system plików, rejestr oraz logi systemu. Ponadto nauczysz się wykrywać złośliwe oprogramowanie oraz korzystać z usługi VSS.

Znajdziesz tu także przydatną listę kontrolną, która pozwoli Ci błyskawicznie wkroczyć do akcji po włamaniu. To obowiązkowa lektura dla każdego specjalisty od bezpieczeństwa systemów!

Poznaj najlepsze praktyki:

- reagowania na incydenty
- prowadzenia analizy śledczej i powłamaniowej
- wykrywania złośliwego oprogramowania
- kontroli rejestru systemu

helion.pl
księgarnia internetowa

Nr katalogowy: 14093

Księgarnia internetowa
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

ISBN 978-83-246-6652-2
9 788324 666522
cena: 59,00 zł