



JAKUB DĘBSKI

Programy antywirusowe od środka

Stopień trudności



Dziś programy antywirusowe nie są tym, czym był pierwotnie. Oprócz wirusów komputerowych wykrywają całą gamę zagrożeń, wliczając w to konie trojańskie, oprogramowanie szpiegujące, exploity, phishing, a nawet spam. Choć zagrożenia bywają całkiem różne, do ich badania wykorzystywane są podobne techniki.

Autorzy zagrożeń są twórcy. Kiedyś motywacją była dla nich sława, przecieranie nie odkrytych jeszcze metod infekcji, uznanie w wąskim gronie twórców wirusów i w oczach producentów programów antywirusowych. Sytuacja ta trwała przez ponad 15 lat i w roku 2002 powstały najbardziej skomplikowane wirusy metamorficzne, będące podsumowaniem tego okresu. Ze względu na pełną zmienność miały stanowić *Wunderwaffe*, którą autorzy wirusów pokonają ostatecznie programy antywirusowe. Tak się jednak nie stało. Twórcy silników antywirusowych podnieśli rzuconą rękawicę i wkrótce powstały nowe wersje programów, potrafiące identyfikować kod

metamorficzny. Pokonanie ostatecznej broni osłabiło morale przeciwników. Na ten okres natożyło się kilka spektakularnych medialnie aresztowań i część ludzi tworzących wirusy wycofała się ze sceny. Kolejne dwa lata można uznać za spokojne. Mimo kilku masowych infekcji (np. robaka *Blaster*) nie nastąpił wśród pojawiających się zagrożeń wielki skok technologiczny. Dopiero w roku 2004, gdy pojawił się robak *Mydoom*, a za nim *Netsky* i *Bagle*, spokój został zakończony. Nie ze względu na nowe techniki ataku, ale ze względu na inną motywację. Od tego roku czynnikiem motywującym stały się pieniądze, które przyciągnęły dziesiątki, a z czasem setki nowych twórców niebezpiecznego oprogramowania. Przed twórcami silników antywirusowych pojawiło się nowe zadanie: sprostać powodzi ataków.

Z ARTYKUŁU DOWIESZ SIĘ

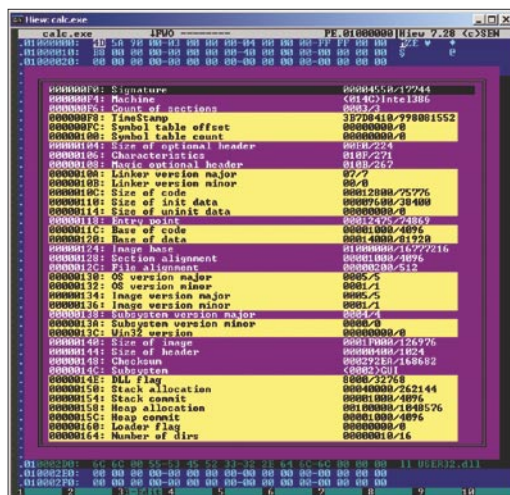
jakie techniki i algorytmy wykrywania zagrożeń stosowane są w programach antywirusowych.

CO POWINIENES WIEDZIEĆ

powinienes posiadać podstawową wiedzę na temat asemblera x86,

powinienes znać popularne algorytmy wyszukiwania,

powinienes znać budowę plików wykonywalnych.



Rysunek 1. Pola nagłówka PE, które mogą zostać zmienione bez wpływu na uruchomienie programu.

Sygnatury

Do wykrywania niebezpiecznego oprogramowania silniki antywirusowe wykorzystują różne techniki. Każda powstała jako odpowiedź na nowe zagrożenie. Choć pierwszymi niebezpiecznymi programami były konie trojańskie, rozgłos zdobyły dopiero wirusy komputerowe potrafiące się replikować. Wirusy początkowo były prostym kodem nadpisującym pliki lub sektory startowe dyskietek. Z czasem metoda ataku została ulepszona i oryginalny kod wykonywał się po zakończeniu działania wirusa, czyniąc działanie wirusa niezauważanym. Do wykrywania prostych wirusów zastosowano sygnatury. Sygnaturą niebezpiecznego programu jest ciąg bajtów, który jednoznacznie go identyfikuje. Ciąg ten powinien

być wystarczająco długi, aby nie powodował fałszywych alarmów. Prawdopodobieństwo fałszywego alarmu można ograniczyć wybierając sygnaturę z fragmentu odpowiedzialnego za niebezpieczną akcję, np. infekcję pliku. Jeżeli autor wirusa wykorzysta tę samą procedurę infekcji w innym wirusie, sygnatura taka będzie w stanie go wykryć.

Wybierane sygnatury mogą zawierać wieloznaczniki (ang. *wildcards*), które odpowiadają za dowolny bajt. W sygnaturach opisujących wirusy asemblerowe dowolnymi bajtami były miejsca, w których wirus przechowywał zmienne lokalne. W przypadku sygnatur z kodu wygenerowanego w językach wysokiego poziomu, wieloznaczniki wykorzystuje się do pominięcia zmiennych argumentów instrukcji procesora, zależnych od umiejscowienia kodu i danych w pamięci. Na Listingu 1. wieloznaczniki oznaczone są za pomocą '??'.
 Niektóre wirusy w ramach walki ze skanowaniem sygnaturami pomiędzy swoje instrukcje dodawały nic nierobiące instrukcje. Do wykrycia takich wirusów można zastosować wieloznaczniki odpowiadające za 0-n bajtów, ale zwykle nie są one stosowane, gdyż spowalniają algorytm wyszukiwania. Można je zastąpić większą liczbą prostych sygnatur.

Choć technika wyszukiwania sygnaturami jest prymitywna, wciąż jest szeroko stosowana przez niektóre firmy antywirusowe. W jej implementacji firmy stosują wszystkie użyteczne algorytmy z zakresu dopasowywania wzorców, wykorzystujące najczęściej deterministyczne i niedeterministyczne automaty skończone, algorytmy słownikowe i drzewa poszukiwań. Zarówno dobór sygnatur jak i same algorytmy optymalizowane są według zastosowań. Przykładowo w plikach wykonywalnych najczęściej pojawiającym się bajtem jest 0, więc nie ma sensu zaczynać sygnatur od tego bajta, gdyż byłby on zbyt często dopasowywany.

Warto obalić jeden z mitów popularnych wśród użytkowników. Zwiększanie bazy zagrożeń nie wpływa drastycznie na szybkość skanowania przez program antywirusowy. Twórcy silników antywirusowych stosują algorytmy działające w czasie logarytmicznym, dzięki czemu zwiększenie liczności bazy zagrożeń na przykład z 300 tysięcy do 400 tysięcy spowoduje kilkuprocentowy spadek szybkości

skanowania. Należy jednak pamiętać, że każda sygnatura jest związana z nazwą i niekiedy procedurą leczącą, co powoduje szybki wzrost wielkości bazy. Z tego powodu część firm wykorzystuje sygnatury generyczne, pokrywające całą rodzinę zagrożeń.

Sumy kontrolne

Sygnatury, aby nie powodowały fałszywych alarmów, muszą mieć długość zapewniającą uniknięcie kolizji. W przypadku programów pisanych w językach wysokiego poziomu sygnatury binarne muszą być bardzo długie, gdyż kod generowany przez kompilator jest podobny. Każdy kompilator stosuje określone konstrukcje, które pojawiają się w zarówno dobrych jak i niebezpiecznych programach. Ponieważ przechowywanie długich wzorców wymaga dużej ilości pamięci, autorzy programów antywirusowych sygnatury zastępują sumami kontrolnymi.

Sumy kontrolne tworzone są za pomocą powszechnie stosowanych funkcji skrótu (ang. *hash functions*), takich jak MD5 czy CRC, lub za pomocą funkcji stworzonych od podstaw dla uzyskania lepszej efektywności. W najprostszym przypadku sumy kontrolne obliczane są z całego pliku. Podejście to wydaje się naiwne, lecz jest efektywne w wykrywaniu zagrożeń spowodowanych zmianami wprowadzonymi w plikach *dobrych*. Część niebezpiecznego oprogramowania modyfikuje jeden lub kilka bajtów w pliku systemowym wyłączając ochronę lub umożliwiając wyexploitowanie go po stworzeniu procesu. Do czasu systemu Windows Vista większość oprogramowania (nawet pochodzącego od Microsoftu) nie była podpisana cyfrowo, więc tak niewielkie zmiany trudno było wykryć innymi metodami.

Obliczanie sum kontrolnych zwykle nie sprowadza się do naiwnego przetworzenia kolejnych bajtów. W przypadku sum kontrolnych uzyskiwanych z sekcji kodu zwykle wykorzystuje się dezasemblację i jak przy sygnaturach pomija bajty odpowiedzialne za zmienne argumenty instrukcji. W przypadku sum kontrolnych z początku pliku (zwykle nagłówek określonego formatu) pomija się niewykorzystywane pola, które po zmianie nie wpłyną na działanie pliku (na Rysunku 1. oznaczone kolorem *białe znaki* i komentarze.

Twórcy niebezpiecznego oprogramowania wprowadzają do plików modyfikacje, aby

oszukać skanowanie sumami kontrolnymi. Czasami proces modyfikacji przebiega w sposób automatyczny, na przykład podczas instalacji oprogramowania w systemie lub podczas pobierania pliku z serwera HTTP. Oprócz zmian nagłówka pliku często używanym trikiem jest dopisywanie bajtów na końcu pliku lub w niewykorzystanej przestrzeni w środku pliku.

Problemem, który pojawia się przy sumach kontrolnych, jest znalezienie właściwego początku dla liczenia sumy. Oprócz oczywistych miejsc takich jak początek pliku czy punkt wejścia programu występują sytuacje, na przykład podczas emulacji, w których startu nie można jednoznacznie określić. Obliczanie sum kontrolnych z obszarów zaczynających się od kolejnych bajtów może być powolne. Z pomocą przychodzi funkcje *skrótów przesuwanych* (ang. *rolling hash*), wykorzystywane np. w algorytmie Rabin-Karpar, które umożliwiają wyliczenie nowej wartości funkcji za pomocą kolejnego bajta i pierwszego bajta kontrolowanego obszaru.

Zwykle do identyfikacji zagrożenia stosowane jest kilka sum kontrolnych lub dopasowanie sumy kontrolnej po tym, gdy zostanie dopasowana sygnatura.

Metody algorytmiczne

Powyższe metody sprawdzają się w przypadku najpopularniejszego niebezpiecznego oprogramowania, czyli prostego oprogramowania tworzonego przez laików. Osoba posiadająca odpowiednie umiejętności może obejść wykrywanie sygnaturami za pomocą polimorfizmu lub

Listing 1. Przykładowa sygnatura z wieloznacznikami

```

53          push     ebx
8BD8       mov     ebx, eax
33C0       xor     eax, eax
A3??????? mov     [00404084], eax
6A00       push     000
E8??????? call    GetModuleHandleA
                                   ;kernel32
A3??????? mov     [00405650], eax
A1??????? mov     eax, [00405650]
A3??????? mov     [0040408C], eax
33C0       xor     eax, eax
Sygnatura:
538BD833C0A3???????6A00E8???????A3
???????A1?????
????A3???????
33C0
    
```

metamorfizmu, może utrudnić znalezienie początku wirusa za pomocą EPO (patrz Ramka), może również próbować zintegrować kod wirusa z nosicielem.

Do wykrycia tak stworzonego niebezpiecznego oprogramowania wykorzystuje się wyspecjalizowane algorytmy. Tworzenie ich przypomina pracę detektywa: analizujemy miejsce zbrodni (plik) i patrzymy, gdzie jej autor popełnił błąd. Algorytmy identyfikujące zagrożenie muszą być niezawodne, ale też szybkie. Ten drugi warunek może być spełniony właśnie dzięki przeoczeniu popełnionym przez zbrodniarza. Błędem najczęstszym jest zmiana charakterystyki pliku, będąca specyficznym *odciskiem palca* pozostawionym podczas ataku. Dzięki niemu możemy stwierdzić, czy analizowany plik posiada cechy wspólne z zagrożeniem. Choć sam *odcisk* nie jest wystarczający do identyfikacji, jest podstawą przejścia do fazy dokładnej analizy. Jeżeli nie występuje, kończymy w tym momencie sprawdzanie. Przykładowymi *odciskami palca* są określone rozmiary i atrybuty sekcji, entropia lub rozkład statystyczny w wybranych obszarach pliku, nazwy sekcji niewystępujące w normalnych plikach czy też punkt wejścia programu ustalony na sekcję danych.

Autorzy wirusów komputerowych zawsze popełniali błędy. W ciągu dwudziestu lat nie pojawił się wirus, którego nie dałoby się szybko wykryć, choć należy liczyć się z tym, że w przyszłości może się pojawić. Autorzy niebezpiecznego oprogramowania wciąż próbują i niektóre słabsze technologicznie silniki antywirusowe nie są w stanie wykryć skomplikowanych wirusów. Właśnie z powodu niewykrycia będącego na wolności wirusa Virut wiele programów nie uzyskało w grudniu certyfikatu Virus Bulletin 100.

Metody kryptoanalityczne

Zarówno wirusy infekujące pliki jak i protektory plików wykonywalnych (np. ASProtect, Yoda czy Armadillo) do ukrycia kodu wykorzystują szyfrowanie. Jedną z metod stosowanych przez programy antywirusowe do uzyskania odszyfrowanego kodu są metody kryptoanalityczne. Sprawdzają się wtedy, gdy algorytm szyfrujący jest najsłabszym punktem obrony w niebezpiecznym oprogramowaniu.

Najczęściej używaną metodą ukrywania kodu jest wykorzystanie prostych operacji typu *xor*, *add*, *sub* na kolejnych bajtach, słowach lub podwójnych słowach. Ich złamanie jest banalne i program antywirusowy jest w stanie zrobić to w ułamku sekundy. Przykładowo dla funkcji *xor*, gdy $c \text{ xor } k = p$, to $p \text{ xor } c = k$. Jeżeli *p* jest naszym tekstem jawnym (tu wirusem odszyfrowanym, który znamy) zaś *c* jest postacią zaszyfrowaną, to w łatwo możemy znaleźć klucz *k*. Jeżeli klucz jest stały lub modyfikowany prostym algorytmem, to nic nie stoi na przeszkodzie, aby odszyfrować nim resztę zaszyfrowanego obszaru.

W przypadku wirusów polimorficznych tekstem jawnym jest część odszyfrowana. W przypadku protektora pliku wykonywalnego tekstem jawnym mogą być wybrane bajty z nagłówka pliku, znane funkcje biblioteczne kompilatora, czy charakterystyczne cechy sekcji zasobów. Wykorzystywana jest każda informacja, która może zostać wybrana jako tekst jawny.

Sprawa komplikuje się w przypadku algorytmów szyfrujących generowanych w sposób losowy. Najczęściej wykorzystywanym zestawem operacji szyfrujących są prosto odwracalne *xor*, *add*, *sub*, *rol*, *ror*, którymi modyfikowany jest też klucz. Przykładowa procedura szyfrująca jednego z wirusów polimorficznych wygląda

następująco (*k* jest kluczem, *v* losową wartością, *op* losowo wybraną operacją):

```
c[i] = p[i] op1 v1
k = k op2 v2
c[i] = c[i] op3 k
```

Odszyfrowanie danych w tym przypadku jest bardziej skomplikowane obliczeniowo, gdyż wymaga sprawdzenia permutacji operacji, ale wciąż jest możliwe do zrobienia w akceptowalnym czasie.

Metody kryptoanalizy przestają być skuteczne w przypadku kryptografii z silnym algorytmem szyfrującym. Ponieważ źródła takich algorytmów są ogólnie dostępne silne algorytmy są coraz częściej używane w niebezpiecznym oprogramowaniu. Na szczęście dla użytkowników w wielu przypadkach są użyte w sposób niepoprawny, wynikający z niewiedzy lub nie doczytania dokumentacji, dzięki czemu możliwe jest ich złamanie.

Odpakowanie statyczne

Kiedy niebezpieczne oprogramowanie zaczęło tworzyć w językach wysokiego poziomu większość autorów przestała skupiać się na tworzeniu zabezpieczeń przed jego analizą. Nie miało to sensu, ponieważ na rynku dostępnych było wiele narzędzi spełniających to zadanie. Niektóre z tych narzędzi były na tyle silne, że nie potrafiły poradzić sobie z nimi programy antywirusowe. Ich zaletą było też to, że w tak zabezpieczonym pliku nie można było wykrywać samego zabezpieczenia, ponieważ zabezpieczenie używane było też w dobrym oprogramowaniu. Autorzy antywirusów musieli znaleźć rozwiązanie problemu w sytuacji, w której nie zawsze pomagała kryptoanaliza.

Jednym ze sposobów poradzenia sobie z protektorami plików wykonywalnych jest odpakowanie statyczne, czyli przeanalizowanie algorytmów protektora i przygotowanie specjalnych procedur odpakowujących w silniku antywirusowym.

Na początku należy protektor zidentyfikować. Do identyfikacji można wykorzystać skanowanie sygnaturami lub, w przypadku protektorów polimorficznych, mechanizmy służące do identyfikacji wirusów polimorficznych. Protektory w większości przypadków działają na takiej samej zasadzie jak wirusy polimorficzne – dodają warstwę zabezpieczającą

Terminologia

- *Polimorfizm* – technika ukrywania algorytmu poprzez zastąpienie oryginalnych instrukcji wykonujących algorytm równoważnymi instrukcjami. Jedna instrukcja może być zastąpiona dowolną liczbą instrukcji dających identyczny efekt. Polimorfizm jest używany w celu ukrycia algorytmu odszyfrowującego oryginalny kod, który zostanie następnie uruchomiony.
- *Metamorfizm* – technika podobna do polimorfizmu. Każda instrukcja oryginalnego kodu jest zastępowana dowolną liczbą instrukcji dających równoważny efekt. Ponieważ ukrywany jest algorytm oryginalnego kodu, nie jest potrzebne dodatkowe szyfrowanie go.
- *EPO* – z ang. *EntryPoint Obscuring* – ukrywanie punktu wejścia. Wirus infekując plik musi w pewnym momencie zostać wykonany. Proste wirusy uruchamiają się wraz ze startem nościela. Technika EPO polega na przekazaniu sterowania do wirusa w innym momencie działania programu-nościela, na przykład zamiast wywołania funkcji API. W ten sposób program antywirusowy nie może stwierdzić, w którym miejscu zaczyna się wirus.

oryginalny kod, zaś kod oryginalny pozostaje niezmienny.

Kiedy rozpoznamy konkretny protektor możemy uruchomić procedurę zdejmującą zabezpieczenie. Procedura taka musi być napisana na podstawie analizy konkretnego protektora, co ma liczne wady. Po pierwsze, jeżeli nie znamy silnika zabezpieczającego, nie możemy zabezpieczenia zdjąć. Do czasu, aż otrzymamy przynajmniej kilka próbek, nie możemy być pewni zasady działania protektora. Po drugie, odpakowanie statyczne jest wrażliwe na modyfikacje wprowadzone w zabezpieczonym pliku. Drobna modyfikacja sygnatury może spowodować, że kod nie zostanie rozpakowany. Drobna modyfikacja wygenerowanego kodu może spowodować błędne odpakowanie. Po trzecie, wiele z dostępnych protektorów posiada otwarte źródła, więc nie jest problemem zmienić sam algorytm zabezpieczenia. Po zmianie algorytmu statyczna procedura odbarczająca przestanie działać, gdyż próbuje odpakować nieobsługiwany protektor. Jeżeli nie możemy zdjąć zabezpieczenia, to atakujący może w prosty sposób wygenerować niewykrywalny wariant swojego programu poprzez użycie tego protektora.

Odpakowanie dynamiczne

W programach antywirusowych emulatory zostały stworzone jako odpowiedź na wirusy polimorficzne. Do dzisiaj są najskuteczniejszym sposobem radzenia sobie z nimi, ale używane są także do odpakowania protektorów. Odpakowanie z wykorzystaniem protektora nazywane jest odpakowaniem dynamicznym.

Program emulowany uruchamiany jest bezpiecznym, symulowanym środowisku. Proces emulacji zwykle zaczyna się od punktu wejścia i przebiega do czasu znalezienia sygnatury niebezpiecznego programu, do czasu, aż emulator stwierdzi, że wykonywany program jest bezpieczny, lub do czasu wyczerpania zasobów.

Emulator może zostać wykorzystany do odszyfrowania zaszyfrowanego kodu niezależnie od złożoności szyfru. Zwykle nie jest problemem dostarczenie zasobów w postaci pamięci, ponieważ taką ilość pamięci potrzebuje zabezpieczony program do uruchomienia. Problemem jest jednak szybkość emulacji. Na emulowanym systemie jest ona od kilkudziesięciu do kilkuset razy mniejsza niż w rzeczywistym działaniu.

Jedną ze skuteczniejszych metod przyspieszenia emulacji jest tłumaczenie dynamiczne (ang. *dynamic translation*), spotykane w kompilatorach JIT. Każda instrukcja, zamiast być interpretowaną, jest tłumaczona na bezpieczną wersję wykonywaną na aktualnym procesorze.

Budowa emulatora

Emulator programu antywirusowego składa się z następujących elementów:

- *Emulator procesora* – Do typowych zadań emulatora procesora należy rozkodowanie instrukcji wskazywanej przez wskaźnik EIP, wykonanie kodu zależnego od instrukcji i zapamiętanie wyników. Następnie EIP jest zwiększany tak, aby wskazywał na kolejną instrukcję. Ponieważ protektory coraz częściej wykorzystują sterowniki pracujące w trybie jądra, konieczne staje się emulowanie instrukcji działających też w tym trybie. Emulowany procesor musi posiadać pełną informację na temat stanu rejestrów ogólnego przeznaczenia, segmentowych, debugowania, flag i wskaźnika instrukcji. Wiele protektorów używa instrukcji koprocesora, MMX, SSE, więc konieczne jest przechowywanie stanów wykorzystywanych też przez nie.
- *Menadżer pamięci* – Menadżer pamięci musi odpowiadać istniejącemu w systemie. Każdy obszar pamięci składa się z ciągłej listy stron. Każda strona posiada opis zabezpieczeń i alokacji. W przypadku naruszenia zabezpieczeń lub próby uzyskania dostępu do nieprzydzielonej pamięci menadżer pamięci musi wygenerować wyjątek. Menadżer pamięci dba o przydzielanie pamięci na stercie i zarządza stosem.
- *Ładowanie procesu* – Na podstawie nagłówka pliku wykonywalnego konieczne jest stworzenie procesu w pamięci. Każda wersja systemu tworzy proces w inny sposób i przed autorem silnika pozostaje wybranie, którą wersję emulować. Do standardowych zadań przy ładowaniu procesu zaliczają się: przydział pamięci i skopiowanie sekcji, ustalenie atrybutów pamięci, załadowanie emulowanych bibliotek, wypełnienie tablicy importów (IAT), zaaplikowanie relokacji. Przy ładowaniu procesu konieczne jest ustalenie środowiska procesu (*Process*

Environment Block) na zgodny z systemem. Jest to konieczne, ponieważ wiele protektorów korzysta z informacji w nim zawartych np. do sprawdzenia, czy proces nie jest debugowany.

Ponieważ system Windows posiada mało restrykcyjny loader, możliwe jest takie zmodyfikowanie pliku PE, że wciąż będzie uruchamiany w systemie Windows, zaś wiele emulatorów stwierdzi, że plik nie posiada poprawnego formatu. Przykładowo wiele pól nagłówka nie jest wykorzystywana lub wykorzystywana jest w sposób odmienny od specyfikacji (Rysunek 1).

· *Emulacja pamięci bibliotek* – Przy tworzeniu procesu lub wywołaniu funkcji `LoadLibrary` konieczne jest wczytanie do pamięci procesu zewnętrznej biblioteki. Biblioteka taka musi być emulowana w sposób jak najpełniejszy (wraz z jej strukturą w pamięci), ponieważ niektóre protektory szukają adresów funkcji eksportowanych bezpośrednio w bibliotece, za pomocą mniej lub bardziej wyrafinowanych metod.

· *Emulacja wątków* – Ponieważ wiele zabezpieczeń wykorzystuje kilka działających jednocześnie wątków, konieczne stało się emulowanie przełączania kontekstu. Ponieważ wiele wątków w protektorach nie uczestniczy bezpośrednio w procesie rozpakowania, konieczne jest stworzenie dobrego mechanizmu ich szeregowania, aby wykonywały się głównie wątki mające znaczenie.

· *Emulacja przerwania* – Emulacja przerwania była głównie wykorzystywana przy emulacji trybu rzeczywistego procesora w czasach wirusów DOSowych. System Windows w trybie użytkownika uniemożliwiawołanie większości z nich, najczęściej rzucając wyjątek.

· *Emulacja wyjątków* – Strukturalna obsługa wyjątków (ang. *Structural Exception Handling*) jest jedną z najpopularniejszych metod ukrywania ścieżki wykonania. Funkcje obsługi wyjątków są ustalone, następnie wywołany jest wyjątek, który powoduje przejście do innego miejsca w kodzie. Ponieważ wyjątek może być spowodowany na wiele sposobów (np. niedozwolony dostęp do pamięci, przerwanie, dzielenie przez zero, przepełnienia arytmetyczne), konieczne

jest emulowanie każdego z nich. W nowych wersjach systemu Windows istnieje również *Vectored Exception Handling*, który jest w protektorach rzadko używany.

Emulacja API – Protektory plików, aby utrudnić debugowanie wywołują szeroką gamę funkcji API. Poczynając od najprostszych jak *IsDebuggerPresent* po tak oryginalne jak *GetProcessAffinityMask*. Tworzenie emulatora wymaga implementowania jak największej liczby używanych funkcji, choć na szczęście większość z nich może zwracać wartości domyślne. Często wartości zwracane przez wywoływane funkcje nie są nigdzie wykorzystywane. Ponieważ protektory używają niekiedy nieudokumentowanych funkcji *Native API*, konieczna jest emulacja również tego poziomu.

Emulacja komunikatów – Jeżeli emulator ma służyć nie tylko do odpakowania pliku, ale też do analizy behawioralnej, konieczne jest dodanie emulacji i przetwarzania komunikatów. Przykładowym komunikatem, który trzeba zasymulować, jest naciśnięcie klawisza *[Enter]* lub aktywacja okna (ustalenie *focusu*).

Emulacja środowiska zewnętrznego – Do analizy behawioralnej konieczne jest zaimplementowanie zewnętrznego środowiska procesu. Wiele infektorów plików może zostać wykrytych jedynie, gdy są w stanie zainfekować plik, wiele robaków można wykryć według komunikacji sieciowej, wiele koni trojańskich dodaje się w rejestrze jako *Browser Helper Object*. Aby możliwe było wykrycie takich akcji konieczne jest stworzenie wirtualnego systemu operacyjnego, posiadającego systemy plików, rejestr i dostęp do sieci. Im więcej elementów zewnętrznych zostanie zaimplementowanych (np. wirtualny serwer HTTP lub IRC), tym skuteczniejsza staje się heurystyka.

Zastosowanie emulacji do odpakowania likwiduje wszystkie problemy, które występują w przypadku statycznego odpakowania. Wiele programów antywirusowych nie posiada jednak dobrze zaimplementowanego emulatora, przez co nie radzi sobie z protektorami plików wykonywalnych. Jest tak, ponieważ stworzenie pełnego emulatora komputera z systemem operacyjnym, jest

zadaniem skomplikowanym, czasochłonnym i żmudnym. Emulator wymaga zespołu ludzi stale pracującego nad nim, ponieważ protektory używają coraz to nowszych sztuczek i wykorzystują coraz więcej funkcji API, które trzeba symulować.

Wykrywanie protektorów

Wiele firm antywirusowych nie będąc w stanie przedostać się przez warstwy zabezpieczeń jedynie informuje o nich użytkownika. Z jednej strony jest to rozwiązanie dobre, ponieważ wykrywa wszystkie tak chronione niebezpieczne oprogramowanie. Z drugiej strony powoduje fałszywe alarmy, gdyż niektóre firmy wykorzystują protektory jako zabezpieczenie przed crackernami. Większość komercyjnych zabezpieczeń wymaga maksymalnie kilku dni pracy dobrego crackera, żeby zabezpieczenie złamać. Można być pewnym, że jeżeli aplikacja jest tego warta, to zostanie złamana.

Jeszcze niedawno stosunek zabezpieczonego malware do zabezpieczonego dobrego oprogramowania wynosił 10:1. Aktualnie wynosi on ponad 1000:1. Wykrywanie samych zabezpieczeń wydaje się więc dobrym pomysłem, ale sytuacja komplikuje się w przypadku zdobywającego popularność .NET, gdzie obfuscatory są cenione i stosowane od dawna ze względu na łatwość dekompilacji pierwotnego kodu.

Analiza behawioralna i jej przyszłość

Wiele programów antywirusowych oprócz analizy behawioralnej podczas emulacji stosuje analizę behawioralną w działającym systemie. Jest to standardowy element HIPS (*Host Based Intrusion Prevention System*). Ze względu na istnienie *Kernel Patch Protection* w systemie Windows Vista wykrywanie intruzów tym sposobem zostało mocno ograniczone. Przyszłością wydaje się być wirtualizacja stosowana w procesorach (AMD-V, ITV), którą można zastosować ją na jeden z trzech sposobów:

- Cały system jest uruchamiany w wirtualnym środowisku i nadzorowany,
- Tworzony jest wirtualny komputer (*sandbox*), w którym program przed uruchomieniem jest analizowany behawioralnie,
- Każdy proces działa w wirtualnym środowisku i jego akcje przenoszone

są do środowiska rzeczywistego w momencie, kiedy program nadzorujący stwierdzi ich bezpieczeństwo.

Heurystyka

Metody heurystyczne umożliwiają wykrycie wariantu znanego zagrożenia lub – w swojej najbardziej rozwiniętej postaci – również zidentyfikowanie zupełnie nowego zagrożenia. Heurystyka, czyli ochrona proaktywna, została dokładniej opisana w styczniowym numerze Hackin9. Zainteresowanych tematem odsyłam do tego numeru.

Ze stosowania zaawansowanych metod heurystycznych najbardziej znany jest silnik programu NOD32 Antivirus firmy ESET, czyli silnik ThreatSense®. Należy podkreślić, że metody heurystyczne, stosowane przez firmę ESET mocno różnią się od typowych, prostych heurystyk. Silnik ThreatSense® stosuje zaawansowaną heurystykę łączącą pełną emulację systemu z analizą behawioralną, statyczną analizą kodu i wzorcami generycznymi. Uzyskane dane są klasyfikowane za pomocą algorytmów sztucznej inteligencji niespotykanych w innych programach antywirusowych, ale wykorzystywanych np. w badaniach nad ludzkim genomem. Kod tych algorytmów został zoptymalizowany do pracy z niebezpiecznymi plikami, w których różnorodność danych jest znacznie większa niż ACGT łańcucha DNA. Siłę tych algorytmów potwierdzają testy heurystyk programów antywirusowych, w których ESET NOD32 Antivirus uzyskuje ponad 70% wykrywalność nowych zagrożeń nie powodując fałszywych alarmów. Testy ochrony proaktywnej przeprowadza m.in. austriacki ośrodek AV-Comparatives (www.av-comparatives.org).

Skanowanie formatów plików

Programy antywirusowe nie przetwarzają całych plików, gdyż byłoby to zbyt wolne. W większości przypadków nie operują też na bajtach, gdyż jest to zbyt niski poziom abstrakcji, aby możliwe było przeprowadzenie skutecznych analiz. Silniki programów antywirusowych muszą znać strukturę pliku, na którym operują i skanować go różnie, w zależności, czy jest to plik formatu exe, archiwum zip czy kod html.

Możemy wyróżnić główne kategorie formatów przetwarzanych obiektów:

- pliki wykonywalne (np. *com*, *dos exe*, *pe exe*, *elf*),
- pliki wykonywalne z kodem pośrednim (*Visual Basic*, *MSIL*, *Java*),
- pliki skryptowe (*vbs*, *bat*, *sh*, *html*),
- protokoły sieciowe (*http*, *ftp*, *icq*, *msn*),
- kontenery, w których są osadzone inne obiekty:
 - archiwa (*zip*, *rar*, *tgz*),
 - instalatory (*Wise*, *NSIS*),
 - obiekty OLE (*doc*, *ppt*, *xls*),
 - pliki poczty, skrzynki pocztowe (*mbox*, *dbx*, *pst*, *msg*, *tnef*),
 - obrazy płyt (*iso*, *mds*, *nrg*),
 - zasoby plików wykonywalnych,
 - pliki pomocy,
 - pliki urządzeń mobilnych (*sis*),
 - kontenery wieloplikowe (*rar*, *r01*, *r02...*),
 - inne kontenery.

Największym problemem, w przypadku parsowania formatów plików są zamknięte źródła o niedostępnej specyfikacji... ale też otwarte źródła, które każdy może zmodyfikować. O ile w przypadku zamkniętej specyfikacji praca wymaga inżynierii wstecznej, niekiedy będącej na pograniczu prawa, to możemy być pewni, że autor dla własnej wygody trzyma się opracowanego przez siebie standardu. Otwarte źródła (na przykład instalatora NSIS) powodują, że każdy może wprowadzić do niego zmiany, przez co powstałych wariantów formatów są dziesiątki. W efekcie każdy wariant jest formatem zamkniętym i firma antywirusowa musi ponownie skorzystać z inżynierii wstecznej.

W przypadku udostępnienia przez producenta specyfikacji nie zawsze można na niej polegać, gdyż implementacja w programie producenta nie zawsze jest z nią zgodna. Powoduje to występowanie luźnych formatów, które mimo licznych modyfikacji wciąż są interpretowane bez zgłaszania błędów. Luźne formaty plików lub ich całkowity brak (jak jest w przypadku języków skryptowych), powodują liczne komplikacje i spowalniają działanie programów antywirusowych. Program antywirusowy nie może z ich powodu stwierdzić, że plik na pewno nie jest danego formatu, więc musi zostać przeskanowany pod kątem zagrożeń występujących w tym formacie.

Skanowanie formatów plików ściśle związane jest z wykrywaniem exploitów. Plik niespełniający wymagań specyfikacji w wielu przypadkach umożliwia przeprowadzenie bufora lub inny atak na aplikację. Ponieważ producenci oprogramowania dostarczają łatę z opóźnieniem, a część użytkowników chce być chroniona nawet bez aktualizacji oprogramowania, na barkach twórców antywirusów spoczywa zadanie wykrywania błędów w plikach i protokołach sieciowych.

Kolejnym problemem jest kodowanie. Dane mogą być zakodowane na wiele sposobów (UNICODE, UTF, BASE64, UUENCODE, BINHEX) i każdy z nich program antywirusowy musi sprowadzić do postaci znormalizowanej. W przypadku plików binarnych występują różnice kodowania kolejności bajtów na różnych procesorach.

Innym problemem, bardzo trudnym do obejścia, są formaty szyfrowane hasłem dostarczonym użytkownikowi drogą zewnętrzną. Hasła stosowane w archiwach wysyłanych pocztą elektroniczną trafiają do użytkownika w formie obrazka lub tekstu. Można próbować wykorzystać słowa listu do odszyfrowania archiwum, można łamać hasła metodą *brute-force*, ale w przypadku długiego hasła dostarczonego w obrazku pozostają zawodne techniki OCR.

Moduł leczący

Leczenie w programach antywirusowych pierwotnie dopasowane było do konkretnych wersji wirusa lub konia trojańskiego. Każdy jednoznacznie wykrywany program mógł mieć przypisaną procedurę leczącą. Współcześnie odchodzi się od wykrywania sygnaturami na rzecz uogólnionych wzorców i metod heurystycznych. W tym przypadku leczenie musi przebiegać w sposób uogólniony. Program antywirusowy usuwa z systemu wszystkie odnośniki do niebezpiecznego obiektu, zabija jego proces, usuwa pliki.

Niestety, nie jest możliwe przeanalizowanie każdego pliku i przygotowanie na niego szczepionki. Niebezpieczne oprogramowanie działa w środowisku dynamicznym. W każdej chwili aktualizuje się, jest zastępowane nową wersją pobraną z Internetu lub ściąga dodatkowe komponenty. Zdefiniowana stała procedura lecząca nie usunie wszystkich zagrożeń.

Program antywirusowy, oprócz usunięcia obiektów, powinien przywrócić zmiany dokonane przez niebezpieczny program,

ale znów nie jest to możliwe. Nie można stwierdzić, czy zmiany w systemie dokonał niebezpieczny program, czy też administrator systemu. Wiele robaków wyłącza domyślną zaporę ogniową, ale robią to też użytkownicy. Administratorzy kawiarenek internetowych wyłączają Menadżer Zadań, ale robią to też konie trojańskie. Niebezpieczne oprogramowanie odbiera użytkownikom uprawnienia w systemie, ale może zrobić to ojciec informatyk nadzorujący pracę swoich dzieci. Czy ustawienia domyślne powinny być przez antywirusa przywracane?

Inną kwestią jest zagadnienie sensu leczenia. W środowiskach stawiających na bezpieczeństwo w przypadku ataku cały system jest czyszczony, gdyż nie ma gwarancji, że po infiltracji zostało znalezione i usunięte z niego wszystko, co jest niebezpieczne. Współczesne systemy informatyczne, a nawet same systemy operacyjne, są zbyt skomplikowane, aby je w pełni przeanalizować. Środowiska rządowe lub korporacyjne zdają sobie z tego sprawę. Odmienne wymagania mają użytkownicy domowi, którzy są przyzwyczajeni do posiadania zainfekowanych systemów i nie widzą w przypadku infekcji problemu. Sytuacja ta powoli się zmienia wraz z rozpowszechnieniem niebezpiecznego oprogramowania powodującego szkody finansowe wśród użytkowników.

Podsumowanie

Prawo patentowe Stanów Zjednoczonych spowodowało, że większość z powszechnie używanych na świecie technik antywirusowych, nawet tych najprostszych, została opatentowana. Po kilku procesach sądowych firmy antywirusowe przestały omawiać techniki, które stosują w swoich programach, przez co trudno jest znaleźć dokładne informacje na ich temat. Artykuł ten stanowi przegląd popularnie używanych technik i opisuje problemy, z jakimi stykają się twórcy silników antywirusowych. Mam nadzieję, że uchylił rąbka tajemnicy.

Jakub Dębski

jest starszym analitykiem w firmie ESET (producent m.in. programów ESET NOD32 Antivirus oraz ESET Smart Security), specjalizującej się w ochronie proaktywnej z wykorzystaniem najbardziej zaawansowanych mechanizmów analizy heurystycznej. Wcześniej przez wiele lat pracował w polskich firmach antywirusowych. Był kierownikiem projektu silnika mks_vir; następnie ArcaVir. Studia na Wojskowej Akademii Technicznej ukończył obroną pracy dyplomowej na temat wykorzystania sieci neuronowych w detekcji zagrożeń internetowych.
Kontakt z autorem: debskijakub@wp.pl