Przegląd narzędzi do reverse engineeringu

Autor Bartosz Wójcik | Opublikowane Listopad 2014 | Magazyn Programista 11/2014 (30)

Przegląd narzędzi wykorzystywany w reverse engineeringu (inżynieria wsteczna oprogramowania). Wady, zalety, alternatywne rozwiązania.

Reverse engineering – czyli inżynieria wsteczna, to zbiór technik wykorzystywanych do analizy zamkniętego oprogramowania w celu wydobycia niedostępnych z pozoru informacji jak np. użyte algorytmy, ukryte hasła dostępu (np. do baz danych), informacje o tym jak szyfrowane są jakieś pliki aplikacji itp.

Reverse engineering stosuje się w takich dziedzinach jak analiza oprogramowania w celu odnalezienia potencjalnych luk bezpieczeństwa (exploitacja), analiza złośliwego oprogramowania (firmy antywirusowe) czy np. lokalizacja oprogramowania i gier.

Zaawansowana analiza oprogramowania wymaga znajomości struktury badanych plików, czyli najczęściej wchodzi w grę znajomość formatów plików wykonywalnych *Portable Executable* dla systemu Windows czy formatu *ELF* dla

systemów z rodziny Linux. Wymagana jest również znajomość niskopoziomowego assemblera dla platform 32 i 64 bitowych, aby poprawnie zrozumieć skompilowany kod w zamkniętym oprogramowaniu, jego strukturę i powszechnie wykorzystywane koncepcje i konstrukcje programistyczne przełożone na binarne dane.

Nawet posiadając odpowiednią wiedzę nie będziemy w stanie jej wykorzystać bez odpowiednich narzędzi. W tym artykule chciałbym przedstawić dedykowane narzędzia z podziałem na kategorie, wykorzystywane w reverse engineeringu. Większość z przedstawionych narzędzi kwalifikuje się na materiał na całkiem oddzielny artykuł, jednak moim zamysłem było przedstawienie jak największej ilości oprogramowania, aby pokazać ich różnorodne zastosowanie.

Skomplikowany charakter oprogramowania do reverse engineeringu i sam proces jego tworzenia wiąże się często z tym, że oprogramowanie takie jest również drogie, lecz starałem się także pokazać alternatywne i darmowe odpowiedniki do zaprezentowanych przykładów.

Identyfikatory

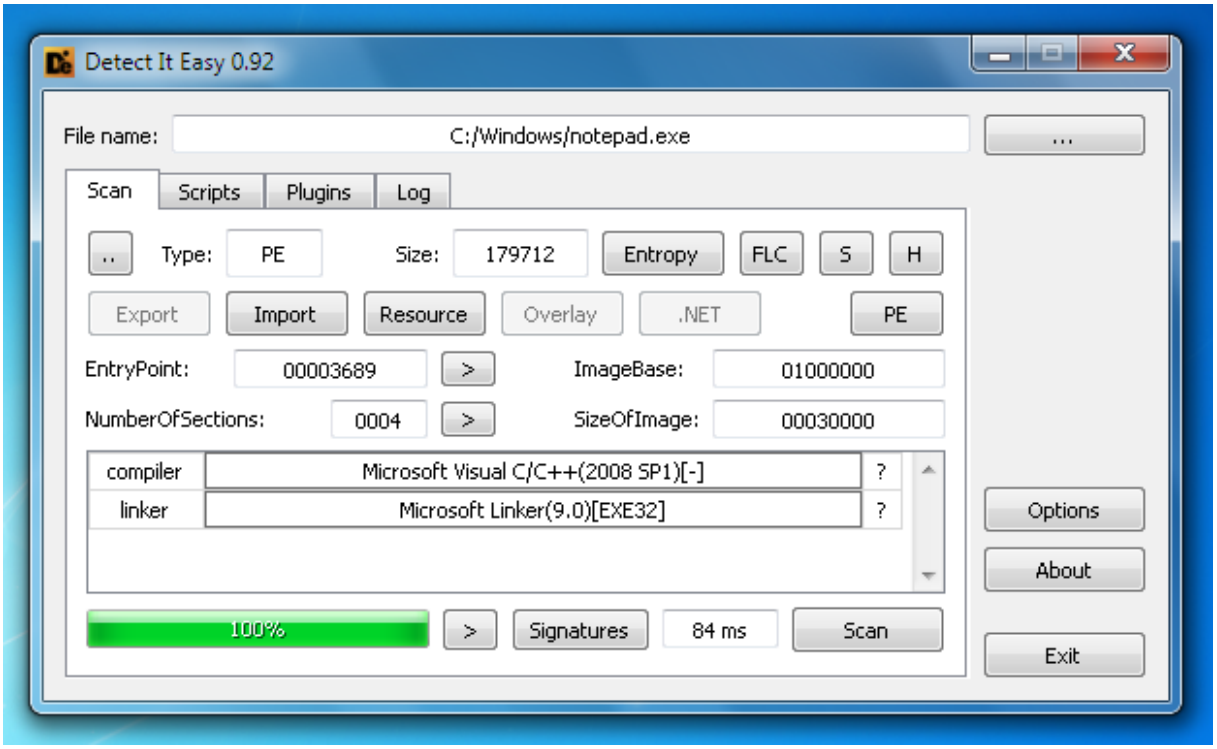
Języków programowania jak i ich kompilatorów jest cały ogrom. Poza aplikacjami stworzonymi w językach skryptowych, można wyróżnić aplikacje skompilowane do natywnego kodu procesora oraz aplikacje skompilowane do kodu przejściowego. Oprócz tego istnieje szereg metod zabezpieczających aplikacje i ich zasoby – wszystko to wpływa na końcowy efekt w postaci binarnego obrazu pliku na dysku.

Jeśli nie jesteśmy pewni, w czym utworzone zostało oprogramowanie, któremu się przyglądamy, bo nie mamy wprawy w rozpoznawaniu charakterystycznych cech w skompilowanych plikach (nazwy sekcji, importowane biblioteki etc.) – warto skorzystać z identyfikatorów (inaczej detektorów), czyli narzędzi, które posiadają bazę sygnatur popularnych kompilatorów, bibliotek programistycznych, kryptograficznych czy systemów zabezpieczeń aplikacji. Szybka analiza pozwoli nam zdecydować, jaki kolejny krok zostanie podjęty (np. rozpakowanie aplikacji).

Detect It Easy

Detektor *DIE* posiada bazę danych najpopularniejszych systemów zabezpieczających, w tym *exe-packerów*, *exe-protectorów* oraz sygnatury popularnych kompilatorów i linkerów. Do tego posiada wbudowany prosty język skryptowy, który pozwala szybko dodawać nowe definicje sygnatur. Dostępna jest także przeglądarka elementów pliku wykonywalnego *PE*.

Rysunek 1. Detect It Easy

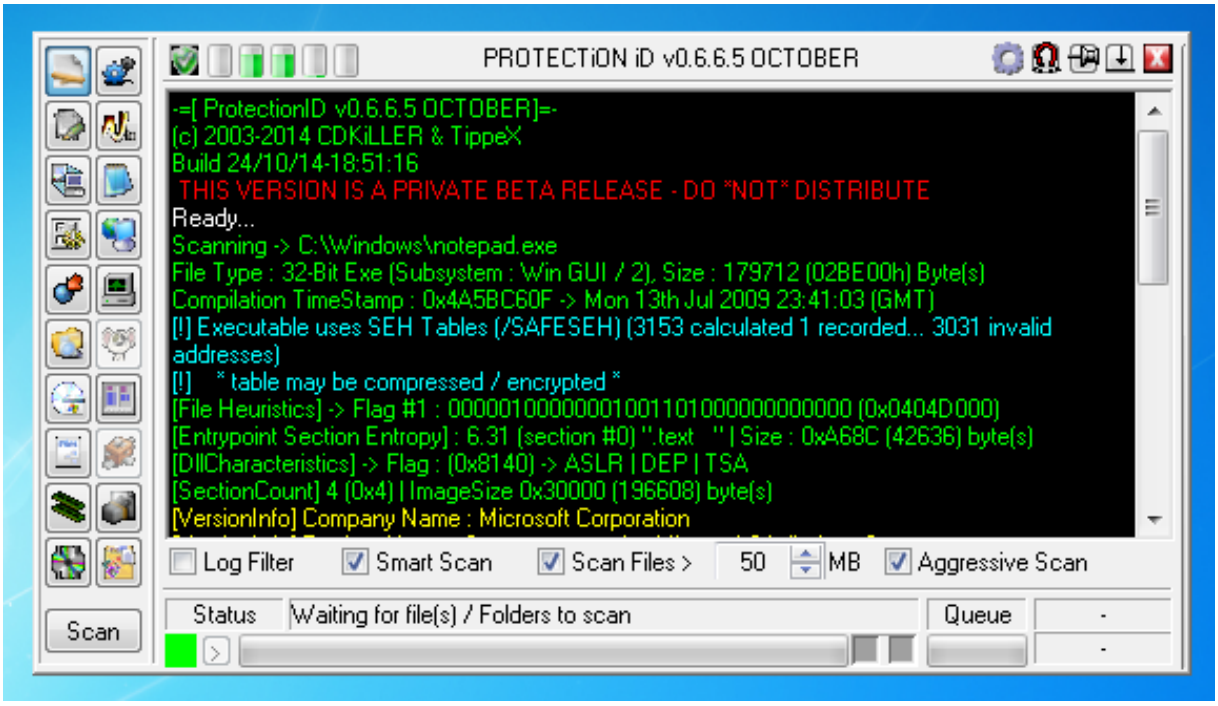


Strona	http://ntinfo.biz
Licencja	Freeware
Zalety	• Wbudowany język skryptowy • Przeglądarka struktur plików *PE* • System wtyczek • Jest na bieżąco aktualizowany • Wtyczki dla edytora *HIEW* i *CFF Explorer* • Wersje dla Windows, Mac OS X i Linux
Wady	• Niewielka baza sygnatur
Alternatywy	• *Exeinfo PE* – podobny detektor polskiego autora - http://www.exeinfo.xn.pl

ProtectionID

Detektor *ProtectionID* powstał na potrzeby wykrywania systemów zabezpieczeń gier, posiada ogromną bazę sygnatur wszelkich możliwych systemów zabezpieczeń oraz kompilatorów i linkerów. Mimo, że interfejs użytkownika nie ujmuje elegancją, to spełnia swoją funkcję znakomicie oraz jest bardzo często aktualizowany.

Rysunek 2. ProtectionID



Strona	http://pid.gamecopyworld.com/
Licencja	Freeware
Zalety	• Ogromna baza sygnatur • Bardzo często aktualizowany
Wady	• Mało intuicyjny interfejs

Deasemblery i dekompilatory

Posiadając wiedzę, z czym mamy do czynienia lub ściślej – w jakim języku programowania i kompilatorze została stworzona aplikacja, rozpoczynamy analizę w deasemblerze lub dekompilatorze. Ich zadaniem jest analiza skompilowanego, binarnego pliku i przedstawienie jego kodu i struktury w czytelnej dla człowieka postaci.

Dzięki procesowi deasemblacji lub dekompilacji, dowiemy się jak wyglądają wszystkie funkcje aplikacji, jakie ciągi tekstowe znajdują się wewnątrz i które fragmenty kodu się do nich odwołują, z jakich zewnętrznych funkcji systemu operacyjnego korzysta aplikacja lub jakie funkcje są eksportowane (np. w przypadku bibliotek dynamicznych *DLL*).

Rolą deasemblerów jest przedstawienie kodu aplikacji w postaci kodu niskopoziomowego assemblera, czyli jeśli analizowane oprogramowanie było pisane czy to w C++ , Delphi, Visual Basic czy jakimkolwiek innym języku wysokiego poziomu skompilowanego do natywnego kodu – deasembler pokaże nam jego kod wynikowy w postaci kodu assemblera x86 lub x64.

Dekompilatory potrafią lub bardzo się starają odtworzyć oryginalny kod wysokiego poziomu z kodu skompilowanych aplikacji. Jak się domyślacie odtworzenie kodu np. *C*++ z rozpoznaniem struktur danych, typów, konstrukcji języka programowania ze skompilowanego kodu assemblera jest bardzo skomplikowanym procesem, dlatego ilość narzędzi pozwalających na takie działanie jest bardzo mała, a jeśli już są w tym dobre to równocześnie są bardzo drogie.

Dekompilatory można podzielić ze względu na kategorię oprogramowania, które potrafią analizować. Kompilatory języków takich jak np. C# (cała rodzina .NET Framework), Visual Basic, Java generują kod wynikowy w postaci przejściowej, tzn. nie jest to kod bezpośrednio wykonywany przez procesor jak kod x86, ale to pseudokod (tzw. *P-Code*), który wykonywany jest przez wirtualną maszynę tych systemów programowania (czyli przykładowo do uruchomienia potrzebujemy zainstalowany np. .NET Framework lub JVM).

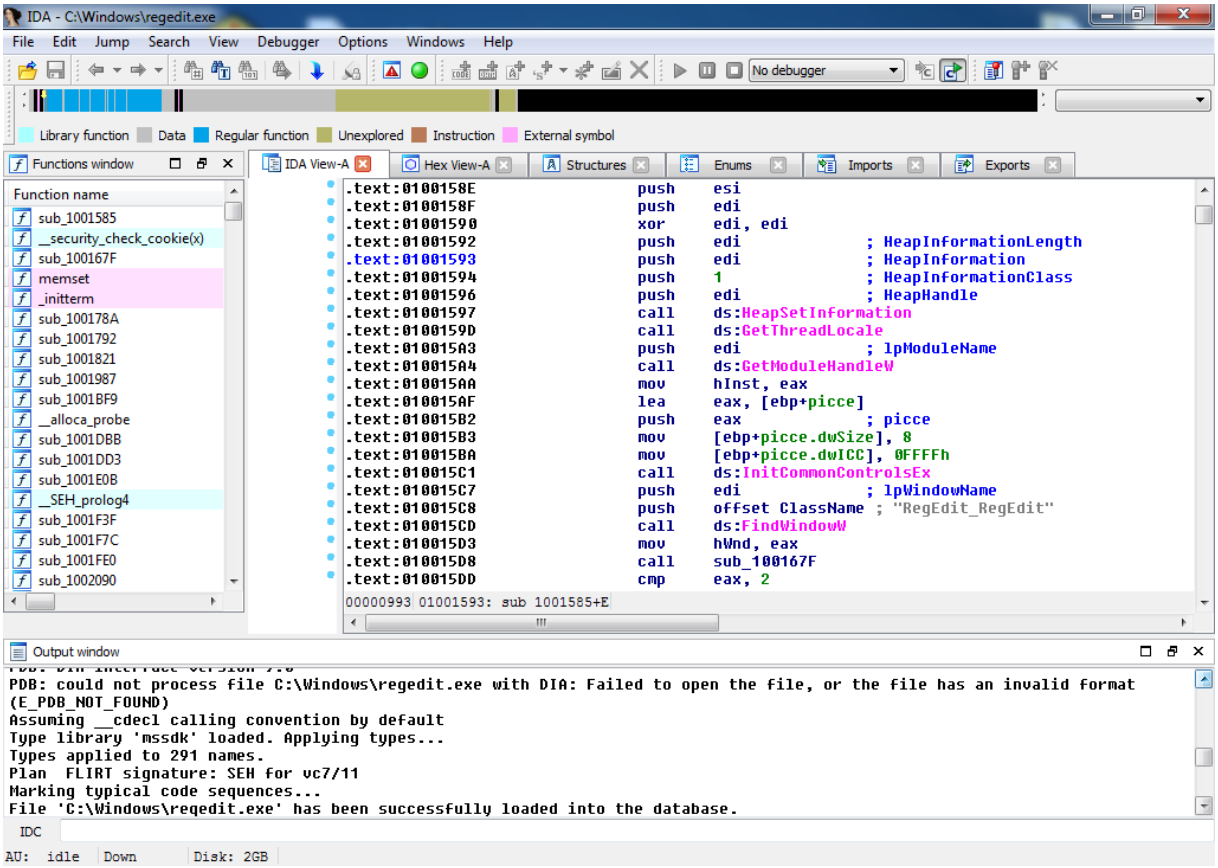
Taki kod wynikowy ze względu na prostotę oraz większą ilość informacji zapisaną w postaci pseudoinstrukcji i metadanych, przyczynił się do tego, że dekompilacja w takich przypadkach jest o wiele prostsza niż dekompilacja kodu x86 lub x64. Pociągnęło to za sobą powstanie wielu dedykowanych dekompilatorów, które stały się prawdziwą zmorą programistów tworzących w tych językach programowania, gdyż bardzo prosto każdy może podejrzeć niezabezpieczone oprogramowanie praktycznie w wersji z pięknie odtworzonym kodem źródłowym.

Krótki wstęp za nami, więc przed nami lista najpopularniejszych deasemblerów i dekompilatorów wraz z ich zastosowaniem.

IDA i Hex-Rays

IDA czyli ***I****nteractive* ***D****is****A****ssembler* to niekwestionowany król wśród narzędzi wykorzystywanych w reverse enigneeringu. *IDA* to deasembler i debugger z wbudowaną obsługą analizy kodu dla ponad 60 rodzajów procesorów. Posiada własny język skryptowy, bogatą bazę sygnatur najpopularniejszych bibliotek programistycznych oraz obsługę wtyczek, które dodatkowo zwiększają funkcjonalność np. obsługę poprzez skrypty w Pythonie.

Rysunek 3. Okno IDA



Najbardziej znaną i cenioną wtyczką dla *IDA* jest dekompilator *Hex-Rays*, który obsługuje dekompilację kodu x86, x64 i ARM, co jest niezastąpionym narzędziem analizy.

IDA posiada również wbudowane debuggery dla wielu platform sprzętowych, co czyni z tego narzędzia prawdziwy kombajn do analizy wszelkiej maści plików wykonywalnych.

Strona	https://www.hex-rays.com
Licencja	Komercyjna od 449 EUR oraz darmowa wersja demonstracyjna.
Zalety	• Obsługa ogromnej ilości typów procesorów • Wbudowane sygnatury popularnych bibliotek programistycznych • Spore możliwości konfiguracji • Wbudowane debuggery • System wtyczek • Język skryptowy • Wersje dla Windows, Mac OS X i Linux

Wady	• Cena • Brak dobrych, darmowych alternatywnych rozwiązań
Wtyczki	• *FindCrypt* – wyszukiwarka sygnatur algorytmów kryptograficznych - http://www.hexblog.com/?p=27 • *IDAStealth* – ukrywanie przed metodami antydebug - http://newgre.net/idastealth • *BinDiff* – porównywanie baz danych w poszukiwaniu różnic kodu - http://www.zynamics.com/bindiff.html

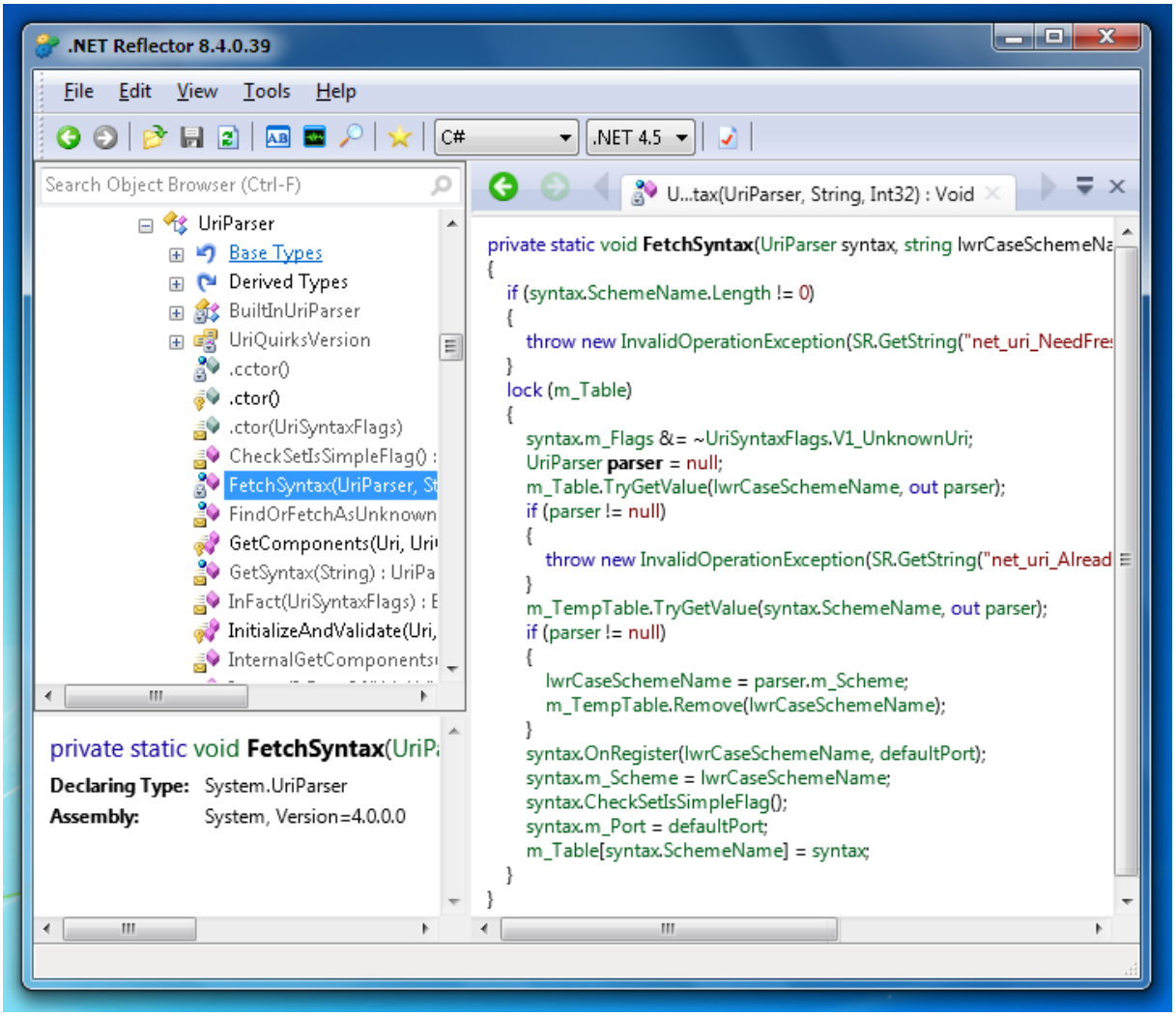
.NET Reflector

Jeśli przyjdzie nam się zmierzyć z analizą oprogramowania stworzonego w języku programowania opartym o .NET Framework czyli np. C# lub VB#, niezastąpiony okaże się dekompilator *.NET Reflector*. Dzięki niemu łatwo i sprawnie będzie można podejrzeć strukturę aplikacji i kod.

Ogromną zaletą *Reflectora* jest to, że posiada małą, lecz bardzo użyteczną bazę wtyczek, dostępna jest przykładowo wtyczka pozwalająca na odtworzenie całego projektu dla Visual Studio ze zdekompilowanej aplikacji. Dodatkowo integracja z Microsoft Visual Studio pozwala na jednoczesne debugowanie własnego kodu oraz kodu zamkniętych bibliotek.

Ze względu na łatwość dekompilacji programów stworzonych dla .NET Framework, powstało wiele narzędzi zabezpieczających, mowa oczywiście o obfuscatorach, które ze skompilowanych programów usuwają metadane, potrafią zmodyfikować kod *IL*, szyfrują ciągi tekstowe etc. Jeśli natrafimy na taki program, warto zapoznać się z deobfuscatorem *de4dot*, który automatycznie potrafi usunąć metody zabezpieczeń kilkudziesięciu typów obfuscatorów.

Rysunek 4. Okno .NET Reflector



Strona	http://www.red-gate.com/products/dotnet-development/reflector/
Licencja	Komercyjna od 99 USD oraz darmowa wersja czasowa.
Zalety	• Znakomita prezentacja i nawigacja po zdekompilowanym kodzie • Dekompilacja do wielu wyjściowych języków C#, VB#, IL • Dekompilacja i debugowanie wprost z Microsoft Visual Studio • Wiele przydatnych wtyczek jak np. patcher *Reflexil*
Wady	• Brak obsługi zabezpieczonych aplikacji (brak deobfuscatora) • Powolne uruchamianie (sprawdzanie online licencji)
Wtyczki	• *Reflexil* – edytor kodu .NET - http://reflexil.net/ • *Deblector* – debugger - http://deblector.codeplex.com/ • *ReflectionEmitLanguage* – generator kodu *IL* - http://reflectoraddins.codeplex.com/wikipage?title=ReflectionEmitLanguage

	• Baza wtyczek – https://reflectoraddins.codeplex.com/

Java Decompiler

JD-GUI czy też *Java Decompiler* to dekompilator jak nazwa wskazuje dla aplikacji Java. Pozwala na przeglądanie kodu skompilowanych unitów **.class* lub całych paczek **.jar*.

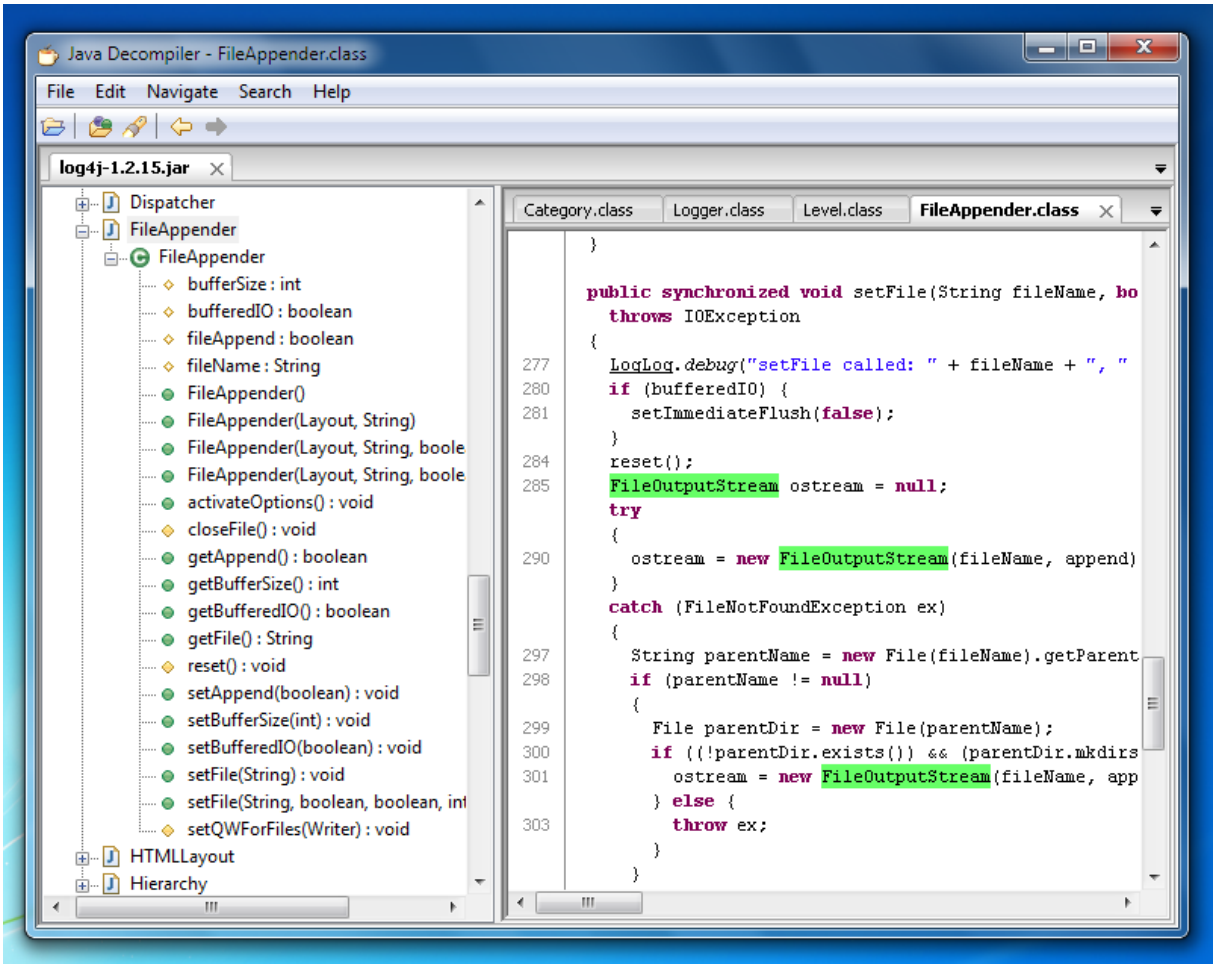
Posiada bardzo użyteczną wyszukiwarkę z filtrowaniem pozwalającą na wyszukiwanie po nazwach typów, konstruktorów, polach, metodach i stałych ciągach tekstowych.

Oprócz samodzielnej aplikacji istnieją również wtyczki dla środowiska programistycznego *Eclipse* oraz *IntelliJ IDEA*, które pozwalają na przeglądanie kodu skompilowanych modułów.

Jeśli kiedyś korzystałeś lub nadal korzystasz ze znanego dekompilatora *JAD* (który przestał być rozwijany w 2001 roku) to najwyższa pora na aktualizację, nie dość, że *JD-GUI* wspiera nowe elementy języka Java to nawigacja po zdekompilowanym projekcie jest bardzo prosta i przyjemna.

Należy tutaj wspomnieć, że tak samo jak w przypadku aplikacji .NET, które zostały zabezpieczone obfuscatorami, tak samo aplikacje w Java mogą być zabezpieczane i wtedy działanie dekompilatora jest ograniczone lub wręcz niemożliwe.

Rysunek 5. Java Decompiler (aka JD-GUI)

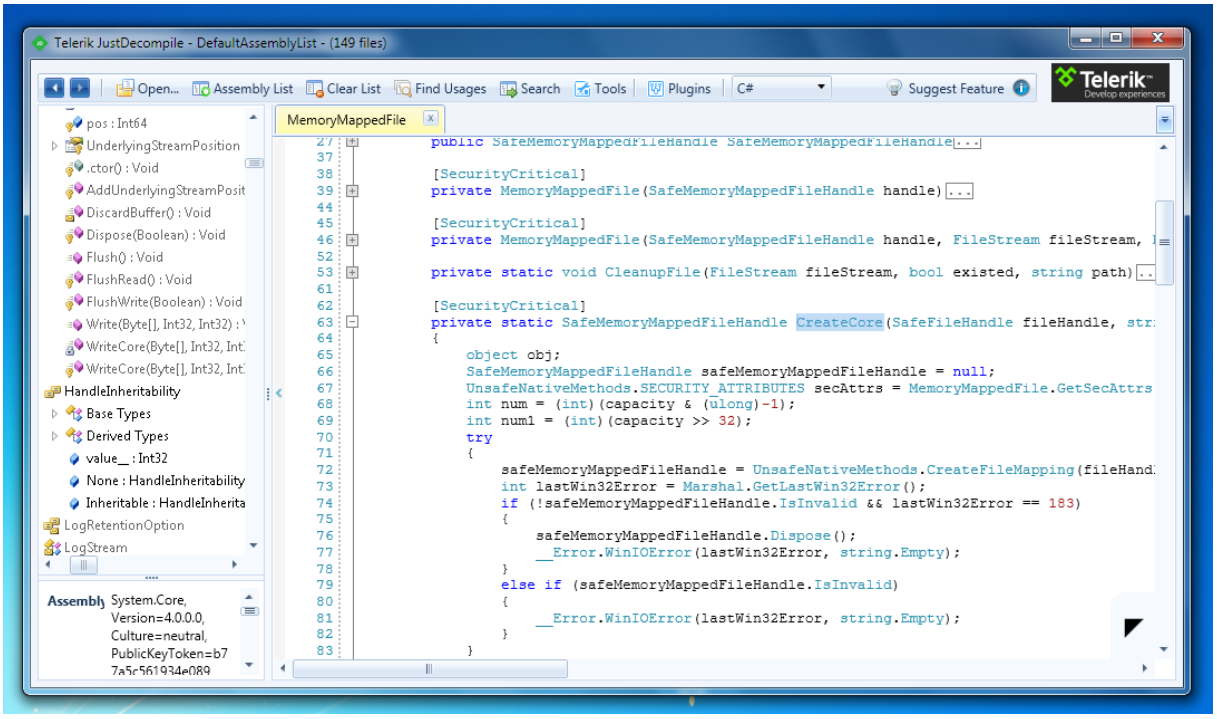


Strona	http://jd.benow.ca/
Licencja	Freeware
Zalety	• Intuicyjna nawigacja po zdekompilowanym kodzie • Wtyczki dla środowiska *Eclipse* oraz *IntelliJ IDEA*
Wady	• Brak obsługi zabezpieczonych aplikacji (brak deobfuscatora) • Brak deasemblacji do *IL* w przypadku wykrytych błędów

JustDecompile

Darmowa alternatywa dla komercyjnego *.NET Reflector* od firmy *Telerik* znanej z komponentów *UI*. Darmowa nie znaczy gorsza, posiada wbudowaną wyszukiwarkę referencji, generowanie projektu ze zdekompilowanych źródeł oraz obsługę wtyczek, w tym wtyczkę deobfuscatora *de4dot*.

Rysunek 6. Dekompilator Just Decompile

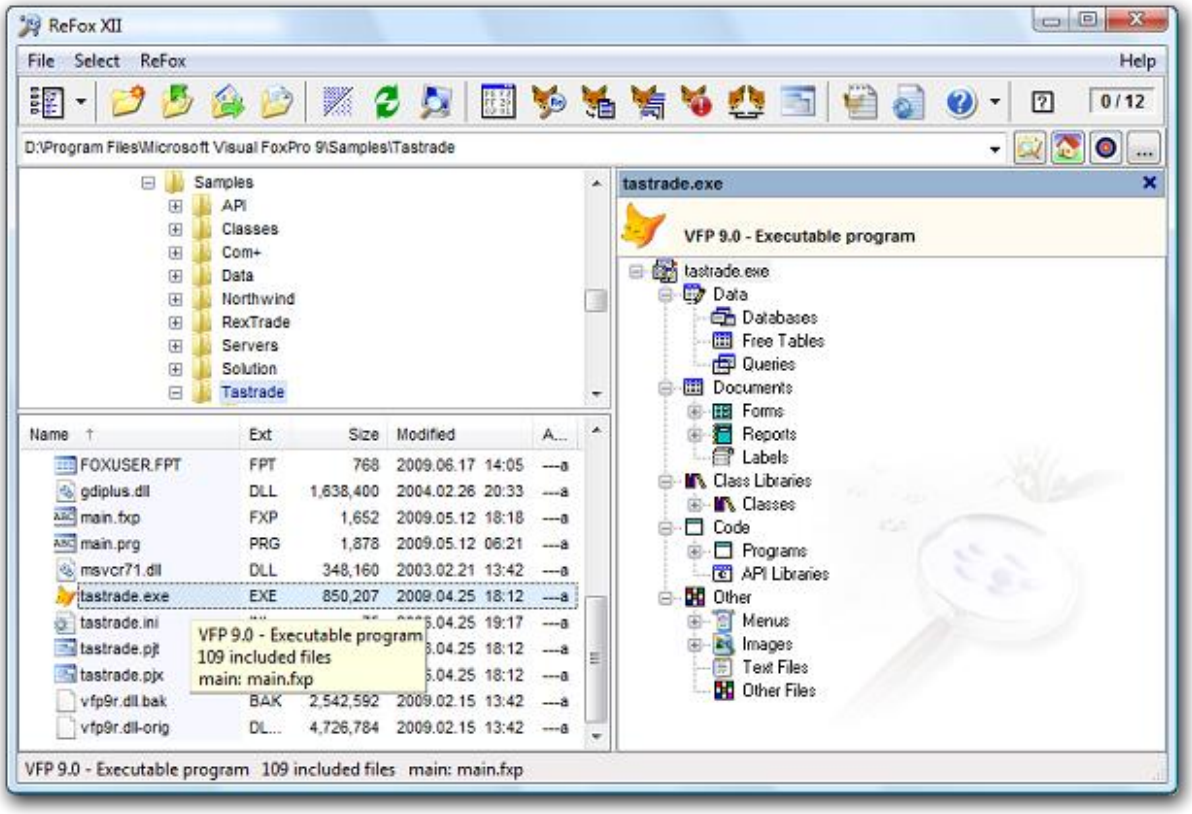


Strona	http://www.telerik.com/download/justdecompile
Licencja	Freeware
Zalety	• Obsługa własnych wtyczek • Generowanie wyjściowego kodu w C#, VB# i IL • Wtyczka dla Visual Studio
Wady	• Jest trochę toporny w obsłudze w porównaniu do *.NET Reflectora*
Alternatywy	• *dotPeek* – darmowy dekompilator od *JetBrains* - https://www.jetbrains.com/decompiler/ • *Simple Assembly Explorer* – edytor i deasembler dla .NET - https://code.google.com/p/simple-assembly-exploror/ • *DisSharp* – darmowy dekompilator - http://netdecompiler.com

ReFox

Dekompilator dla aplikacji stworzonych w bazodanowym środowisku programowania *Visual FoxPro* od firmy Microsoft. Jest to bardzo niszowe rozwiązanie do równie niszowego środowiska, jednak nie istnieją alternatywne rozwiązania pozwalające na analizę takich aplikacji, a te które są dostępne przestały być rozwijane i nie wspierają najnowszych wersji aplikacji *VFP*. *ReFox* pozwala na dekompilację klas, podgląd form oraz wbudowanych danych.

Rysunek 7. ReFox

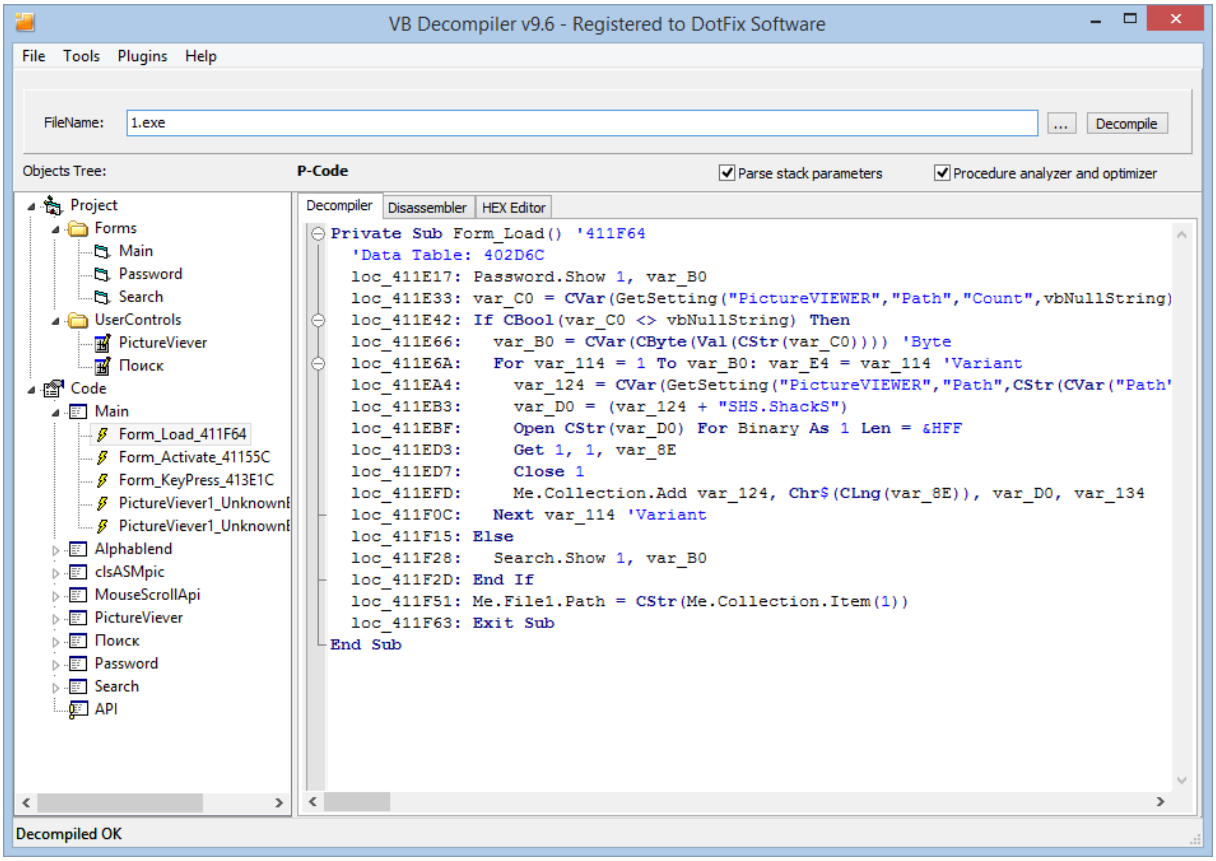


Strona	http://www.refox.net
Licencja	Komercyjna od 290 EUR oraz wersja demonstracyjna.
Zalety	• Dekompilacja klas • Podgląd form • Odtwarzanie projektów *Visual FoxPro*
Wady	• Trochę przestarzały interfejs • Czasami nie radzi sobie z dekompilacją kodu

VB Decompiler

Aplikacje stworzone w Visual Basic 5 oraz 6 należą już do przeszłości, jednak wewnętrzna struktura kodu bazująca na *P-Code* była zalążkiem technologii .NET i od samego początku sprawiała problemy przy analizie kodu, gdyż nie istniały dedykowane narzędzia do jej analizy. Można powiedzieć, że *VB Decompiler* powstał odrobinę za późno jak na potrzeby rynku, jednak jest niezastąpiony przy analizie aplikacji Visual Basic (*EXE*, *DLL* oraz kontrolek *OCX*) skompilowanych do *P-Code* (Visual Basic umożliwiał również kompilację do kodu x86).

Rysunek 8. VB Decompiler

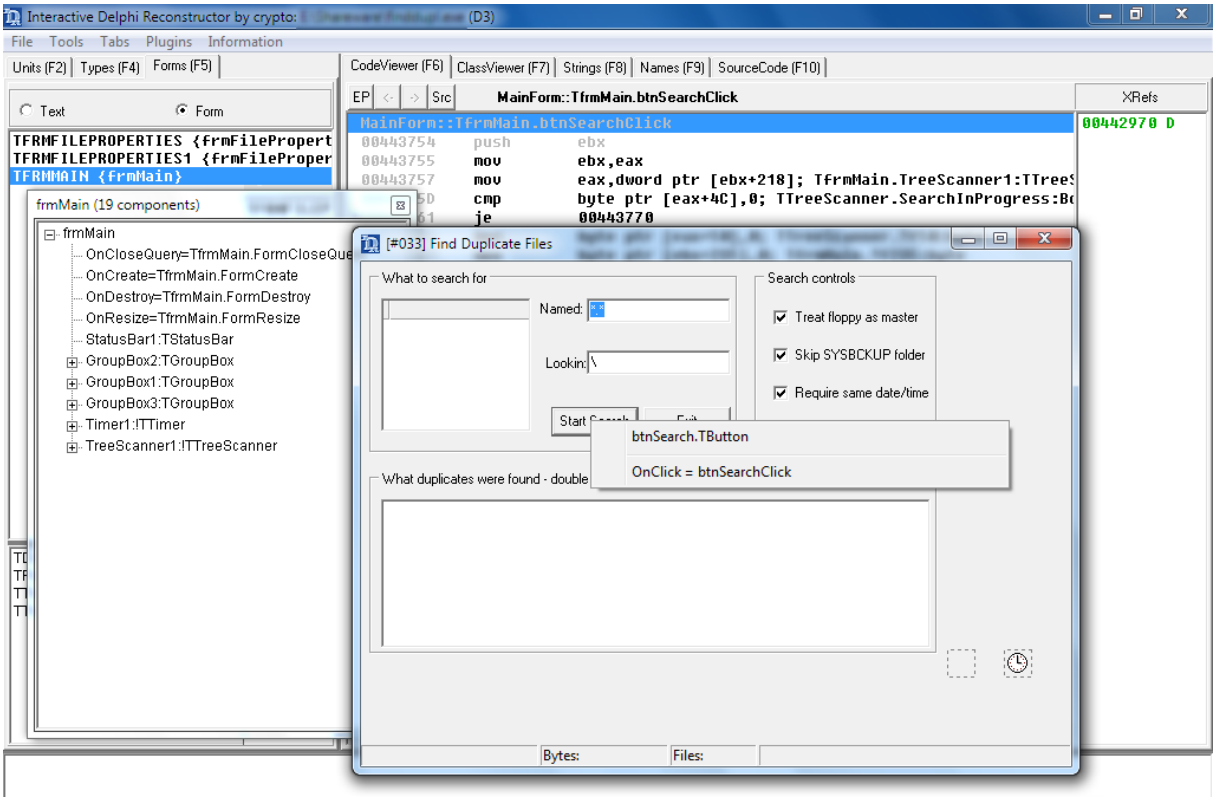


Strona	http://www.vb-decompiler.org
Licencja	Komercyjna od 99 EUR oraz wersja lite.
Zalety	• Podgląd kodu form i eventów • System wtyczek • Deasemblacja natywnego kodu *x86*
Wady	• Ograniczona nawigacja po zdekompilowanym kodzie
Alternatywy	• *VBDIS4* – dekompilator dla *VB4* - http://vbdis4.angelfire.com/

IDR

Deasembler i dekompilator *IDR* czyli *Interactive Delphi Reconstructor* służy jedynie do analizy aplikacji stworzonych w popularnym środowisku Delphi. Jest to narzędzie bardzo przydatne w porównaniu np. do *IDA* z tego względu, że potrafi przeanalizować wewnętrzne struktury aplikacji Delphi, posiada wbudowaną przeglądarkę form, która w łatwy sposób pozwala na szybkie i łatwe odnalezienie przypisanych zdarzeń do kontrolek znajdujących się na formie (np. *button1.OnClick*). *IDR* posiada bogate bazy sygnatur standardowych bibliotek środowiska Delphi we wszystkich znanych wersjach, dzięki czemu w wyjściowym deadlistingu zobaczymy przyjazne nazwy funkcji.

Rysunek 9. IDR



Strona	http://kpnc.org/idr32/en/
Licencja	Wersja darmowa z opcją zakupu bardziej aktualnej kopii (jednak na niewiadomych zasadach, gdyż próby kontaktu z autorem się nie powiodły).
Zalety	• Podgląd form Delphi i eventów kontrolek • Eksport mapy z nazwami funkcji i zmiennych (np. dla *IDA* lub debuggera) • Wbudowane sygnatury wszystkich wersji środowiska Delphi
Wady	• Nieregularne aktualizacje • Niejasne zasady dostępu do najnowszych wersji

Debuggery

Każdy programista prędzej czy później poznaje funkcjonowanie debuggera w swoim ulubionym środowisku programowania. Dzięki debuggerowi prześledzimy na żywo działanie aplikacji, zobaczymy jak instrukcje wpływają na zawartość

pamięci czy zmiennych oraz będziemy w stanie wyłapać potencjalne błędy. Jednak debugowanie własnego oprogramowania, gdy mamy dostępne informacje o kodzie źródłowym i debugujemy najczęściej kod wysokiego poziomu, wprost ze środowiska programistycznego, jest prostą i przyjemną igraszką w porównania do debugowania aplikacji bez dostępu do kodu źródłowego. W tym miejscu przychodzą z pomocą dedykowane debuggery z zaawansowaną analizą binarnych struktur aplikacji, gdzie do ich poprawnej obsługi wymagana jest już znajomość języków niskiego poziomu oraz podstaw funkcjonowania procesora, dla którego skompilowana została aplikacja.

OllyDbg

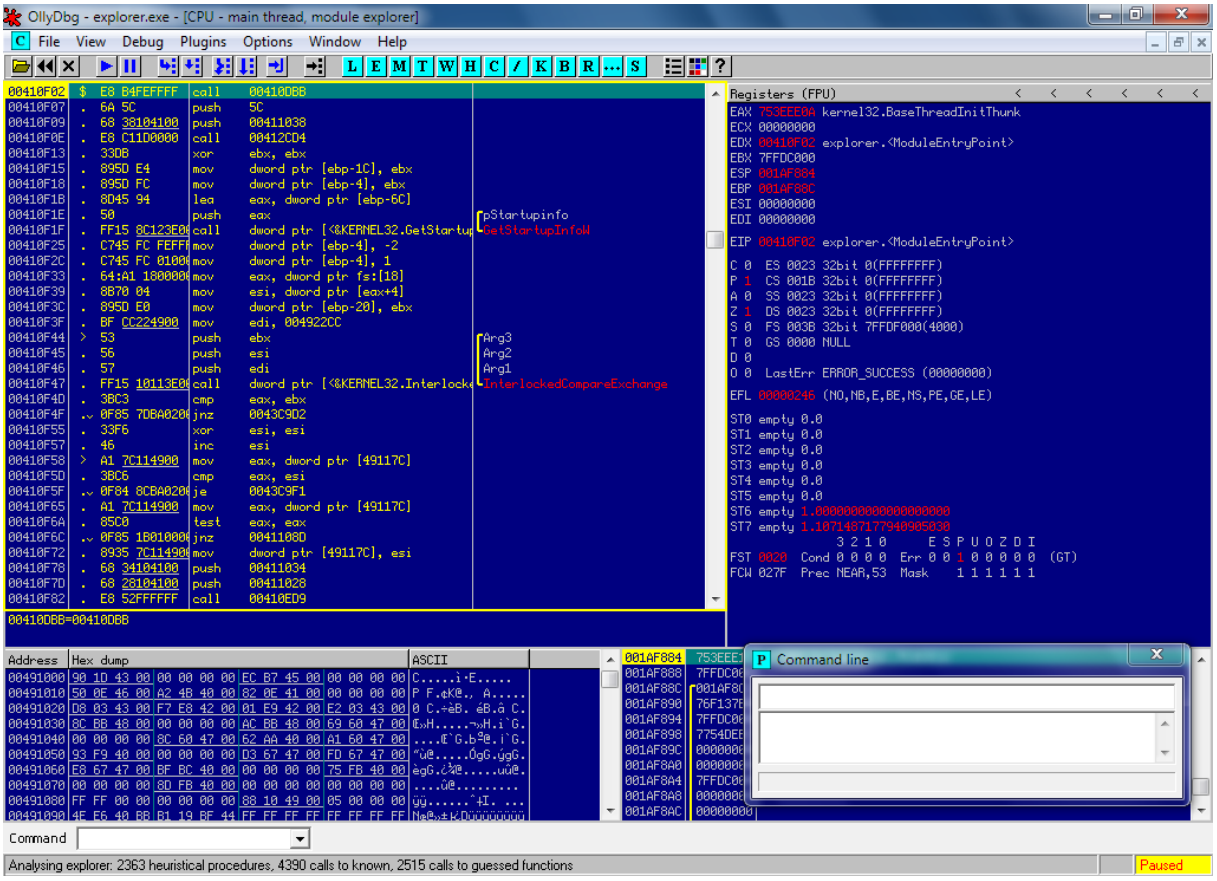
Jest to de facto standardowy debugger dla Windows w świecie reverse engineeringu (zaraz obok debuggera wbudowanego w deasembler *IDA*). Posiada zaawansowaną analizę kodu aplikacji oraz umożliwia ingerencję w niemal każdy aspekt funkcjonowania aplikacji.

Z bardziej ciekawych funkcji, *OllyDbg* umożliwia warunkowy tracing kodu, posiada ogromną bazę wtyczek np. ukrywających jego obecność przed metodami antydebug (wtyczka *Phant0m*), czy wtyczki pozwalające na sterowanie działaniem debuggera z poziomu skryptów (wtyczka *ODbgScript*), a samych skryptów, najczęściej stosowanych do rozpakowywania zabezpieczonych aplikacji znajdziemy setki.

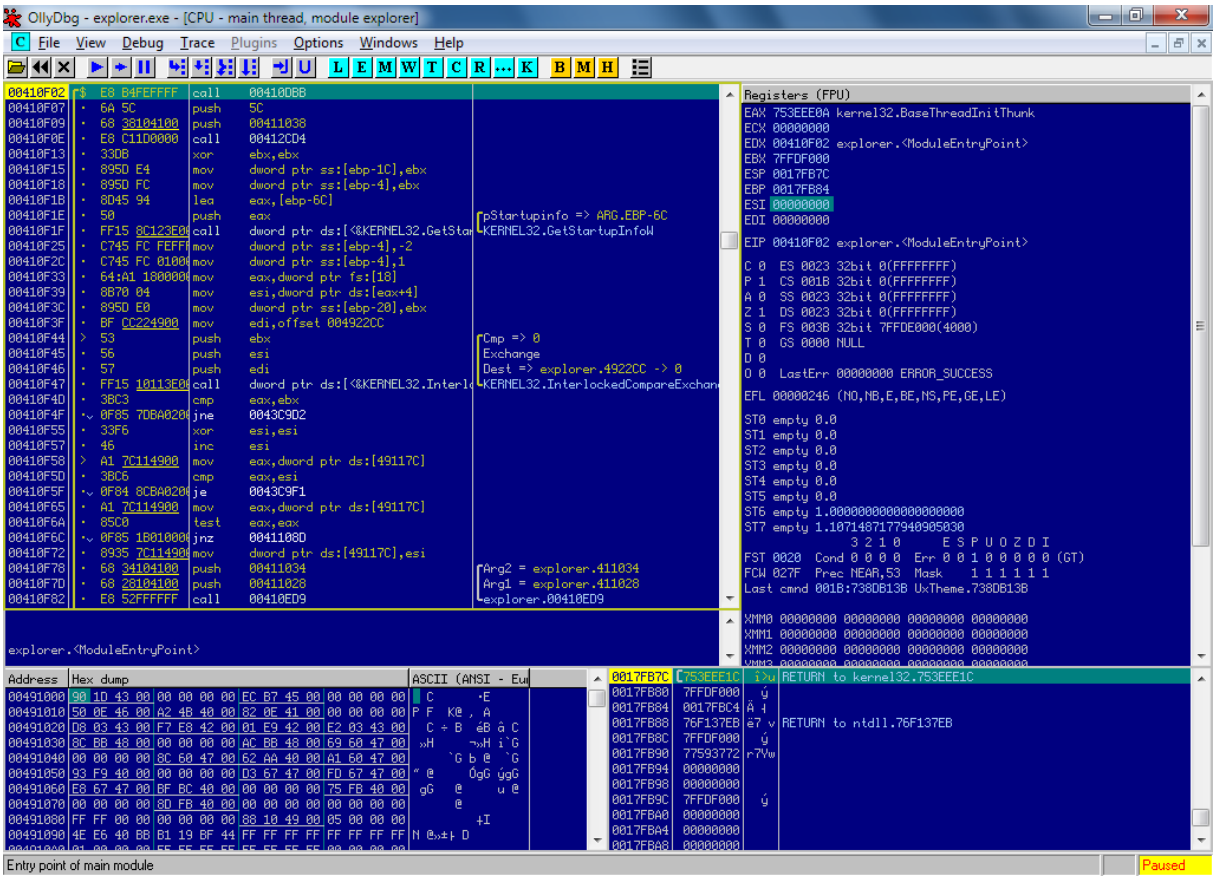
O popularności *OllyDbg* niech świadczy fakt, że żaden inny popularny debugger łącznie z legendarnym debuggerem systemowym *SoftICE* nie doczekał tylu wtyczek i zmodyfikowanych wersji jak *OllyDbg*. Jako ciekawostkę można podać fakt, że powstała specjalna wersja *OllyDbg* pod nazwą *Immunity Debugger* z wbudowaną obsługą skryptów Python-a, przeznaczona do analizy malware oraz tworzenia exploitów.

Istnieją obecnie dwie wersje OllyDbg, stara numerowana 1.10, do której powstało najwięcej rozszerzeń oraz nowa wersja 2.01, która po mału zyskuje na popularności. Dobrą wiadomością jest również to, że w produkcji znajduje się 64 bitowa wersja debuggera, której po prostu brakowało w związku z coraz większą popularnością 64 bitowych systemów operacyjnych.

Rysunek 10. OllyDbg v1.10



Rysunek 11. Ten sam kod w OllyDbg 2.01

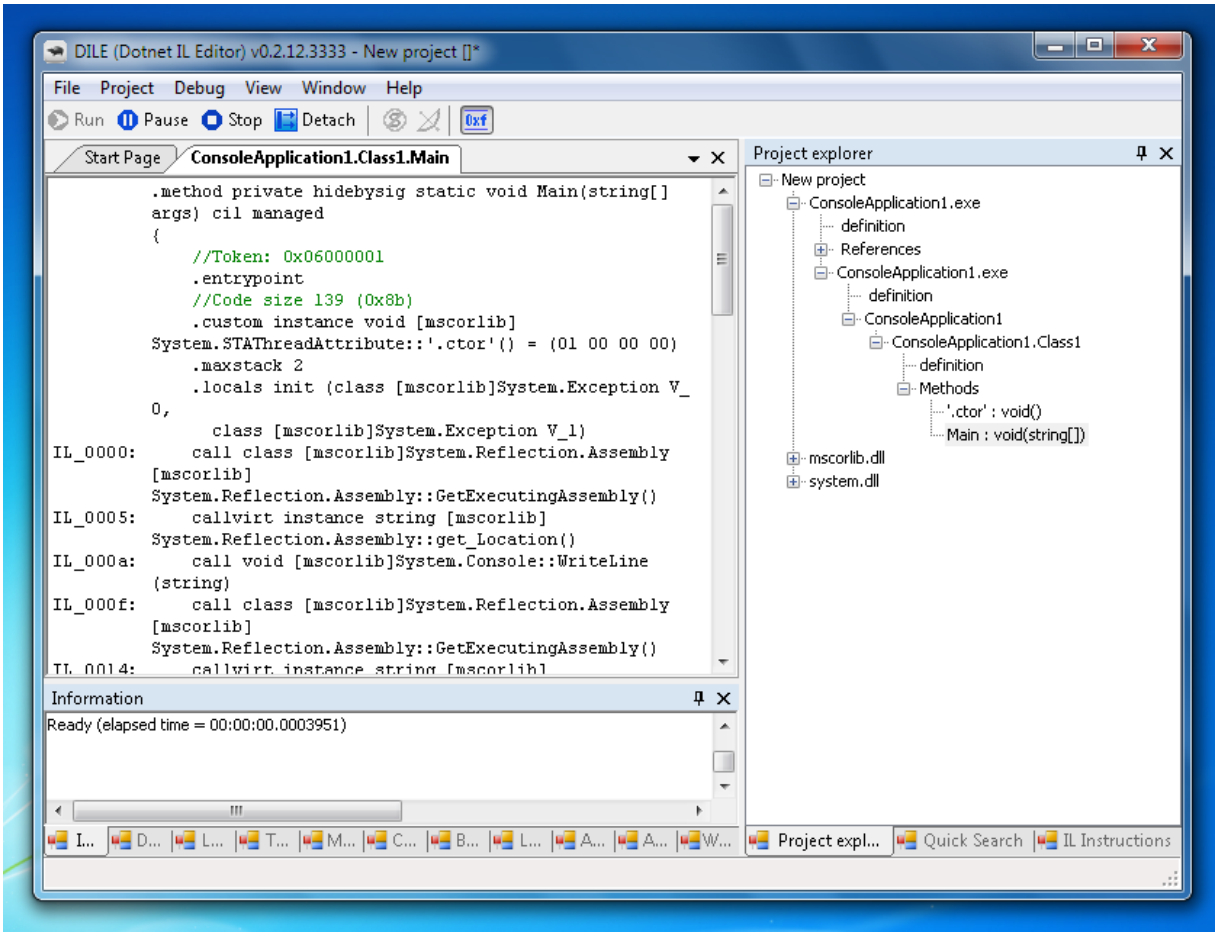


Strona	http://www.ollydbg.de
Licencja	Według strony shareware za darmo (wychodzi freeware?)
Zalety	• Znakomita analiza kodu aplikacji • Bogate opcje konfiguracyjne • Ogromna baza wtyczek i skryptów
Wady	• 64 bitowa wersja ciągle w rozwoju
Wtyczki	• *Phant0m* – ukrywanie obecności debuggera - http://www.woodmann.com/collaborative/tools/index.php/PhantOm • *ODbgScript* – język skryptowy - http://odbgscript.sourceforge.net/ • *OllyDumpEx* – dumper - http://low-priority.appspot.com/ollydumpex/
Alternatywy	• *WinDbg* – debugger systemowy od Microsoft - http://msdn.microsoft.com/en-us/windows/hardware/hh852365.aspx • *GDB* – debugger dla *Linuxa* - http://www.gnu.org/software/gdb/ • *bugdbg* – 64 bitowy debuggery polskiego autora - http://pespin.w.interia.pl

DILE

Debugger dla aplikacji napisanych pod .NET Framework. Jest on dość topornym narzędziem, jednak czasami niezastąpiony. Przypomina trochę debugger wbudowany w Visual Studio, wspominam o nim tylko dlatego, że jest jednym z nielicznych debuggerów dla aplikacji .NET bez dostępu do kodów źródłowych, istnieją również wtyczki dla *.NET Reflector-a* służące debugowaniu (wtyczka *Deblector*).

Rysunek 12. DILE



Strona	http://sourceforge.net/projects/dile/
Licencja	GNU GPL
Zalety	• Że w ogóle istnieje
Wady	• Cała masa • Skomplikowany interfejs użytkownika

Hex edytory

Jeśli przeanalizowaliśmy naszą aplikację w deasemblerze, prześledziliśmy jej działanie w debuggerze, może zajść potrzeba ingerencji czy to w kod programu w celu wprowadzenia odpowiednich poprawek lub zmiana jakichś ciągów tekstowych, stałych wartości albo innych informacji zawartych w pliku binarnym aplikacji.

Do tego celu wykorzystywane są hex edytory. Z czasów, kiedy czytałem magazyn o grach Top Secret, hex edytory kojarzyły mi się jedynie z modyfikacją savegame'ów, gdzie czytelnicy co miesiąc posyłali *offsety* (czyli adresy w pliku)

oraz wartości jakie należało zmienić w plikach ze stanem gry, aby np. uzyskać określoną ilość gotówki w grze czy zasobów.

Na rynku dostępnych jest wiele hex edytorów z wieloma różnymi funkcjami i zastosowaniami jak np. wbudowany podgląd struktur danych (czyli taki hex edytor jest w stanie wizualnie pokazać dla przykładu elementy bitmapy albo jakiejś wewnętrznej struktury pliku wykonywalnego). Przykładem takiego specjalistycznego hex edytora jest np. znany *WinHex*, który znajduje zastosowanie w informatyce śledczej czy pracach związanych z odzyskiwaniem danych (posiada wbudowane wsparcie dla wielu systemów plików), jednak moim zdaniem nie nadaje się do prac typowo związanych z przysłowiowym grzebaniem w binarnych plikach aplikacji, mimo, że posiada odpowiednie funkcje.

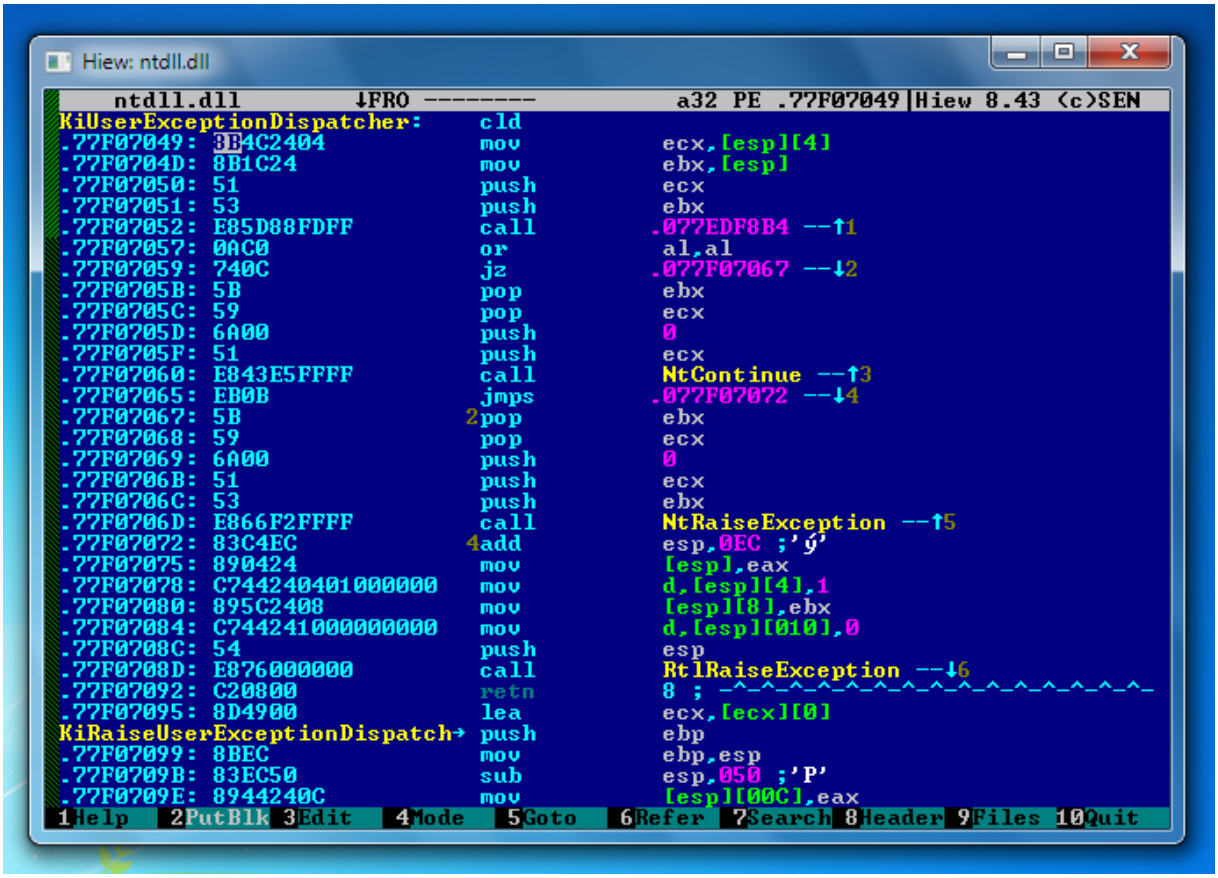
HIEW

Mój numer jeden, jeśli chodzi o hex edytory, bez którego nie wyobrażam sobie pracy. Z pozoru wygląda jak stara konsolowa aplikacja, jednak to prawdziwa bestia. *HIEW* (od *Hacker's View*) to hexedytor, deasembler obsługujący architektury procesorów x86, x64, ARM V6, posiada obsługę plików *NE*, *LE*, *PE/PE32+*, *ELF/ELF64*. Program posiada ogromną bazę użytkowników, rozwijany jest od 1991 roku i regularnie jest aktualizowany.

Dzięki *HIEW* możemy edytować nie tylko dane pliku binarnego, ale jeśli to aplikacja – także jej kod. Wbudowany deasembler pozwala nawigować po kodzie i jego funkcjach oraz w prosty sposób modyfikować istniejące instrukcje z pomocą wbudowanego assemblera, czyli nie trzeba znać na pamięć kodów hex poszczególnych instrukcji, ale można napisać np. `mov eax,edx`, a *HIEW* automatycznie skompiluje taką instrukcję i wstawi ją do pliku binarnego.

HIEW niejednokrotnie potrafi zastąpić narzędzia pokroju *IDA*, jeśli mamy do wykonania jakąś prostą czynność, jego największą zaletą jest szybkość działania oraz wbudowane opcje analizy kodu i jego bezpośrednia modyfikacja.

Rysunek 13. HIEW

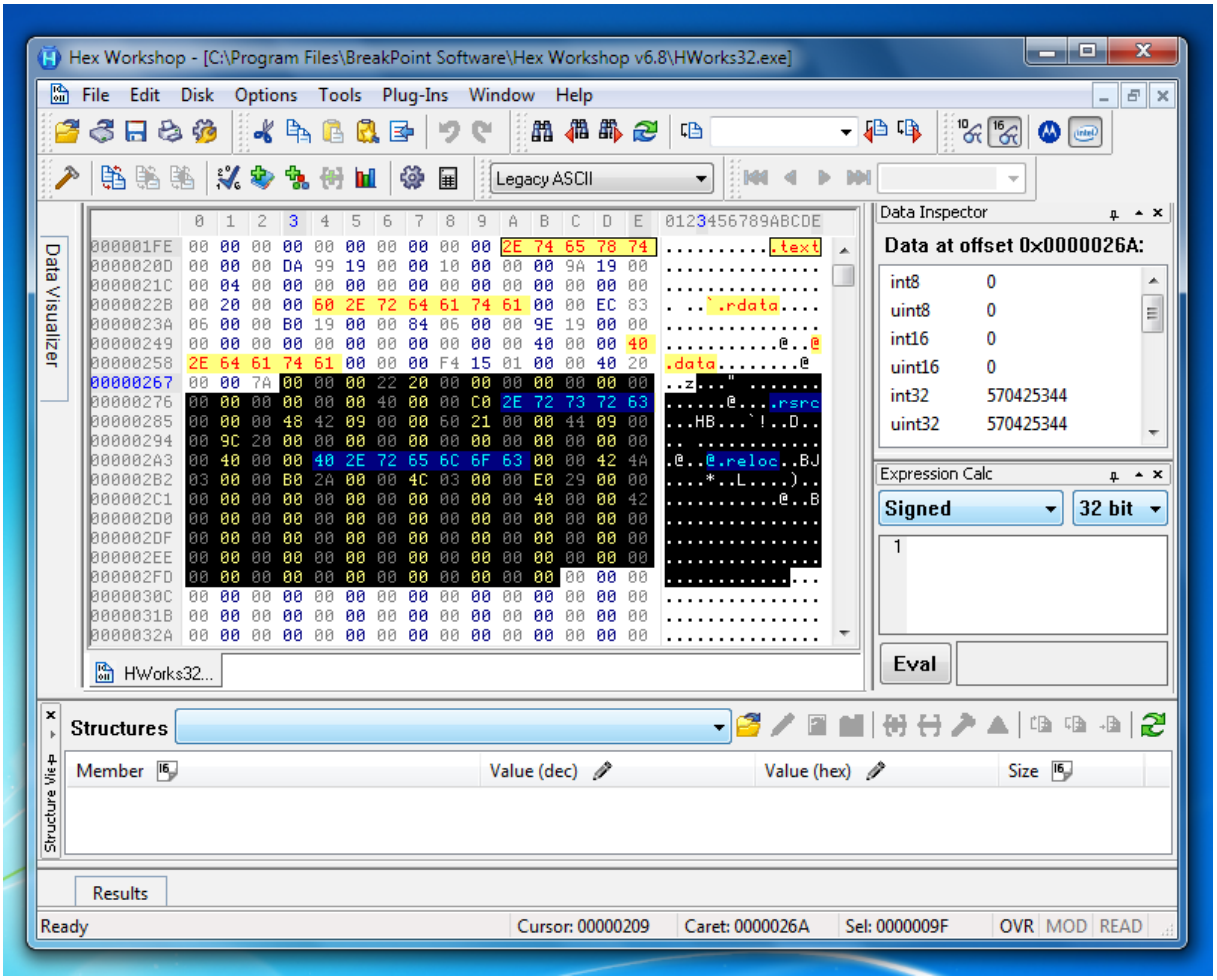


Strona	http://www.hiew.ru/
Licencja	Komercyjna od 19 USD oraz wersja demonstracyjna.
Zalety	• Wbudowany deasembler i assembler dla wielu architektur procesorów • Obsługa struktur wielu formatów plików wykonywalnych • System wtyczek
Wady	• Brak zakładek
Alternatywy	• *BEYE (Binary Eye)* – darmowy, konsolowy hex edytor i deasembler - http://sourceforge.net/p/beye/ • HT Editor – darmowy hex edytor i deasembler - http://sourceforge.net/projects/hte/

Hex Workshop

Okienkowy hex edytor z masą przydatnych opcji, porównywaniem plików, operacjach bitowych na blokach kodu, generowaniem sum kontrolnych, posiada podgląd struktur najpopularniejszych typów plików.

Rysunek 14. Hex Workshop



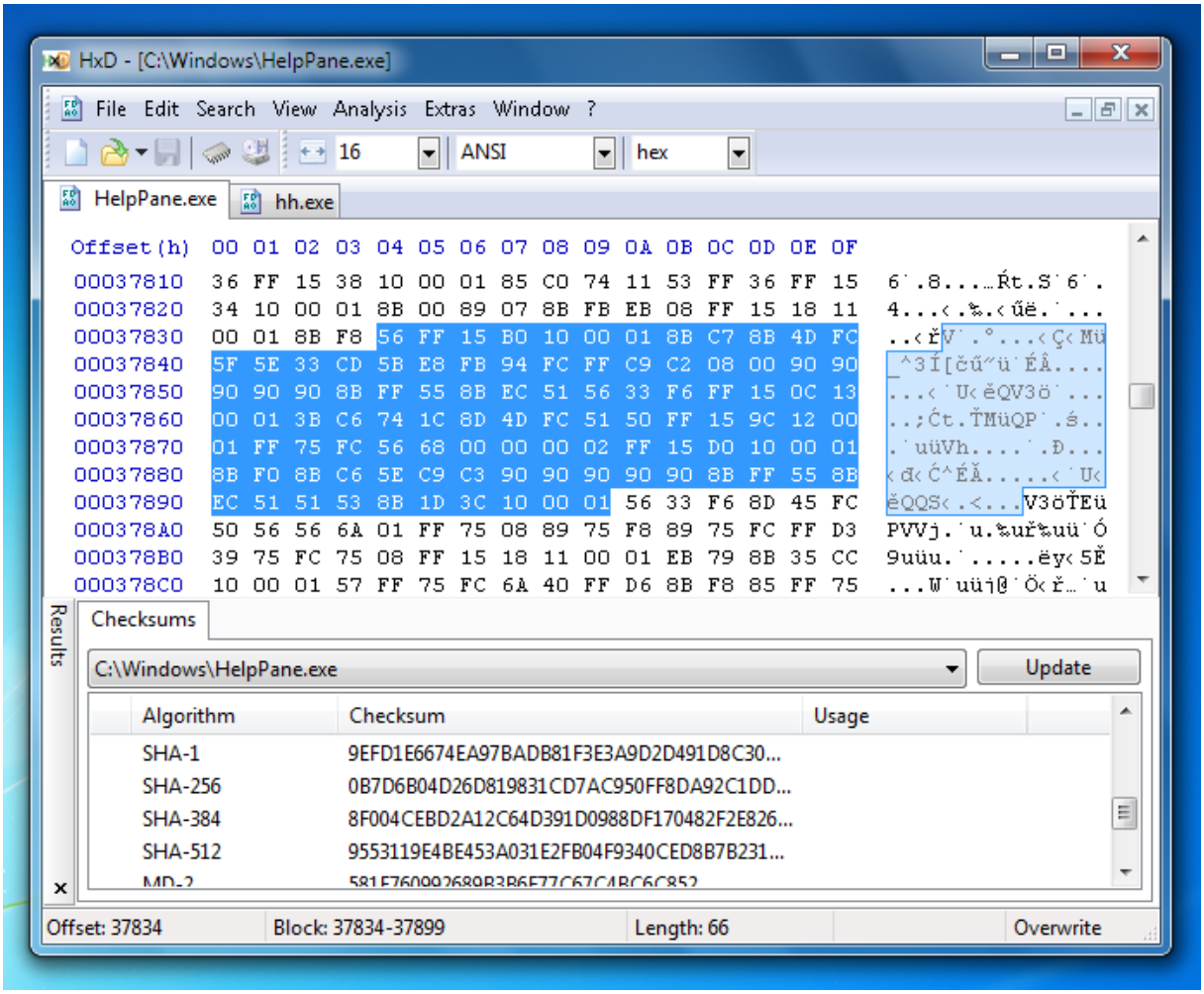
Strona	http://www.bpsoft.com
Licencja	Komercyjna od 89.95 USD oraz wersja ograniczona czasowo.
Zalety	• Zaawansowane operacje bitowe na blokach danych • Możliwość edycji dysków • Wbudowane obliczanie sum kontrolnych i skrótów kryptograficznych • Automatyczne wyszukiwanie wszystkich ciągów tekstowych
Wady	• Bałagan w interfejsie graficznym • Wygórowana cena w porównaniu do dostępnych alternatyw
Alternatywy	• 010 Editor – zaawansowany hex edytor - http://www.sweetscape.com/010editor/

HxD

Darmowy hex edytor ze wszystkimi podstawowymi funkcjami i opcjami jak edycja, wyszukiwanie, porównywanie plików. Pozwala na równoczesną pracę z

wieloma plikami, można także otworzyć z jego poziomu pamięć wybranych procesów oraz uzyskać bezpośredni dostęp do danych dysków.

Rysunek 15. HxD



Strona	http://mh-nexus.de/en/hxd/
Licencja	Freeware
Zalety	• Prosta obsługa • Edycja wielu plików na raz • Możliwość edycji pamięci procesów i danych dysków • Eksport danych do formatu danych plików programistycznych • Wbudowane obliczanie sum kontrolnych i skrótów kryptograficznych
Wady	• Brak zaawansowanych opcji modyfikacji (jak np. operacje XOR na blokach danych) • Minimalistyczny interfejs

Alternatywy	• *Hex Editor Neo* – darmowy hex editor - http://www.hhdsoftware.com/free-hex-editor

Edytory zasobów

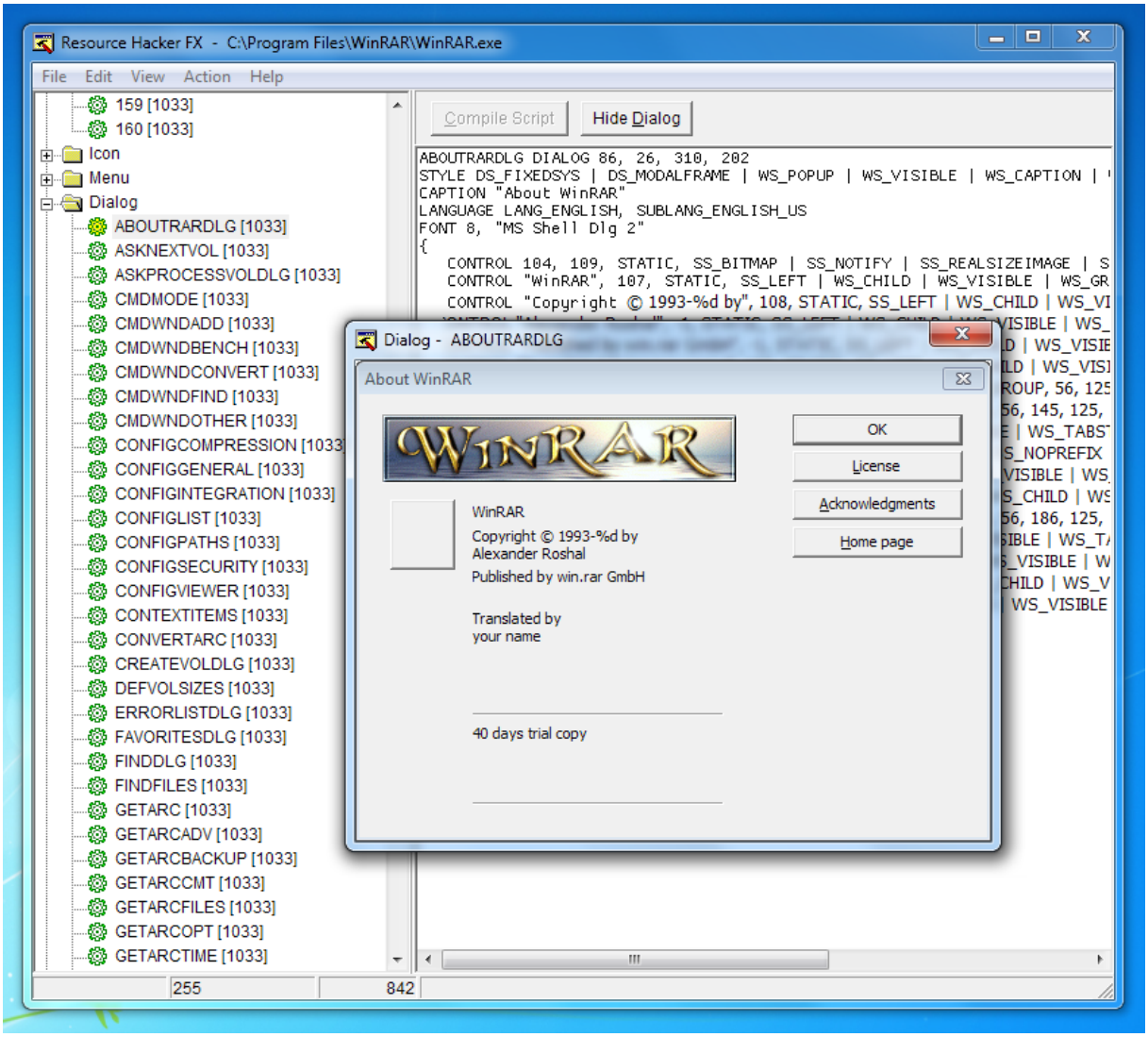
Charakterystyką aplikacji dla Windows jest to, że zasoby takie jak ikony, grafiki, formy, lokalizowane teksty i inne informacje, mogą być zapisane w strukturze pliku *PE* w specjalnym obszarze zwanym zasobami. Dane te są zapisywane na etapie linkowania. Dzięki temu wszystkie pliki aplikacji zapisane są w jednym wyjściowym pliku *EXE* czy *DLL*. Jeśli zajdzie potrzeba zmiany tych informacji to o ile ich rozmiar nie ulegnie zmianie, będzie można dokonać modyfikacji korzystając z hex edytora, jednak, jeśli chcemy dodać jakieś nowe dane lub ustawić inne, które fizycznie mają większy rozmiar (np. zmienić tekst na dłuższy, podmienić grafikę) – ze względu na strukturę tych danych należy posłużyć się odpowiednim edytorem zasobów.

Oprócz modyfikacji danych w zasobach aplikacji, edytory zasobów są wykorzystywane po prostu do podglądnięcia, jakie dodatkowe dane zapisane są w pliku aplikacji.

Resource Hacker FX

Jedynym z najpopularniejszych edytorów zasobów był swego czasu Resource Hacker, jednak od dawna przestał być rozwijany, ale jego popularność sprawiła, że powstały dedykowane łatki, które dały mu drugie życie.

Rysunek 15. Edytor zasobów Resource Hacker FX

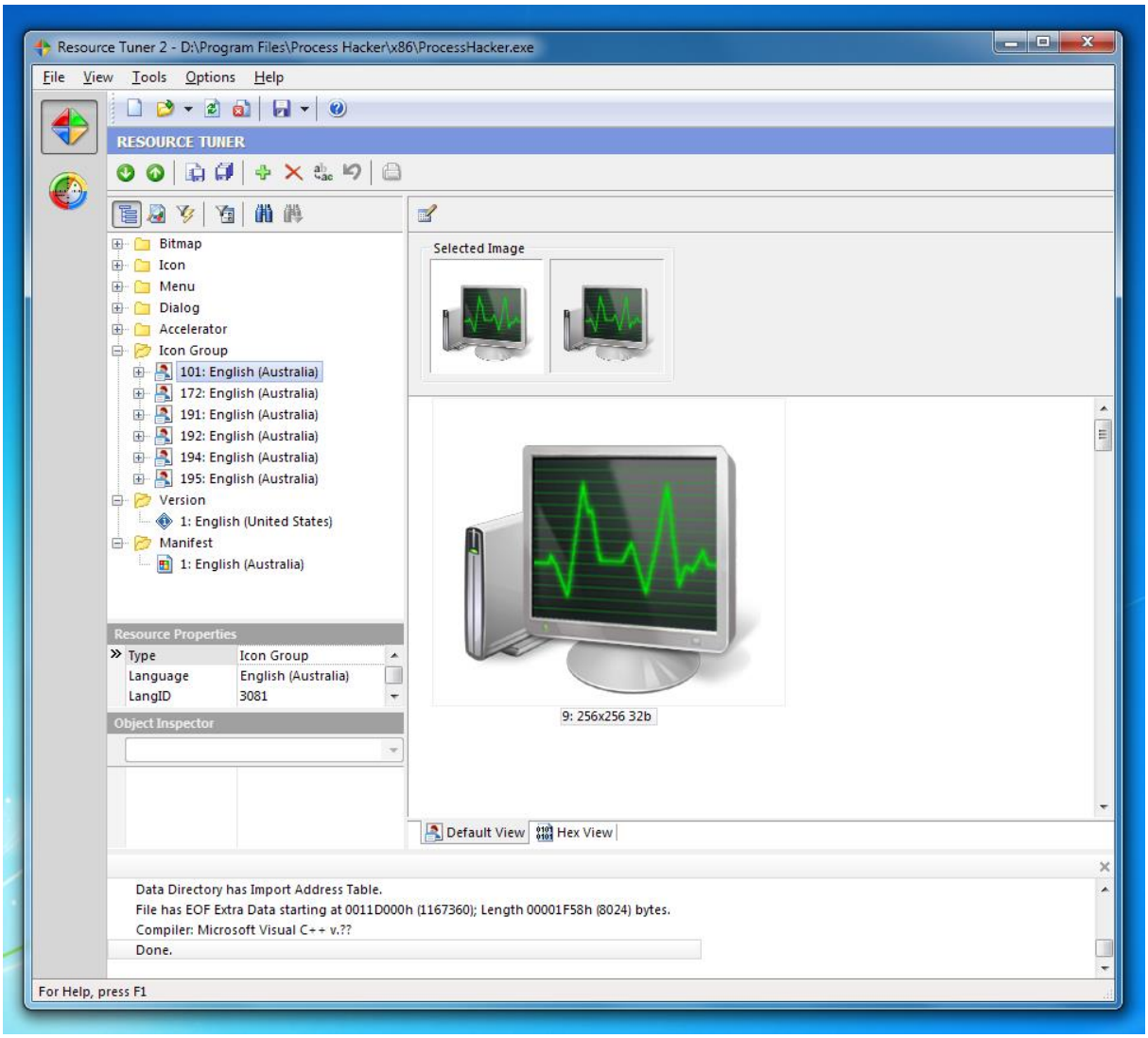


Strona	http://rammichael.com/resource-hacker-fx
Licencja	Freeware
Zalety	• Szybkość działania • Możliwość manipulacji danych na poziomie języka skryptowego zasobów
Wady	• Łatki dramatycznie nie zmieniają przestarzałych funkcji • Brak podglądu form Delphi • Brak kolorowania składni dla elementów XML (jak np. manifesty)
Alternatywy	• *XN Resource Editor* – darmowy edytor zasobów - http://www.wilsonc.demon.co.uk/d10resourceeditor.htm

Resource Tuner

Znakomity edytor zasobów od twórców *PE Explorer*. Posiada wbudowane unpackery np. dla kompresora *UPX* czy *FSG*, edycję zasobów można wykonać korzystając z przyjaznych wizardów. *Resource Tuner* posiada również wbudowany skaner, dzięki któremu można przeskanować dowolny katalog w poszukiwaniu zasobów określonego typu.

Rysunek 16. Edytor zasobów Resource Tuner



Strona	http://www.heaventools.com/resource-tuner.htm
Licencja	Komercyjna od 49.95 USD oraz 30 dniowa testowa wersja.
Zalety	• Przyjazny interfejs użytkownika • Wsparcie w postaci wizardów • Wbudowane unpacker

Wady	• Brak niskopoziomowej edycji struktury zasobów (skryptów)
Alternatywy	• *Resource Builder* – podobny edytor - http://www.resource-builder.com

Edytory i narzędzia pomocnicze

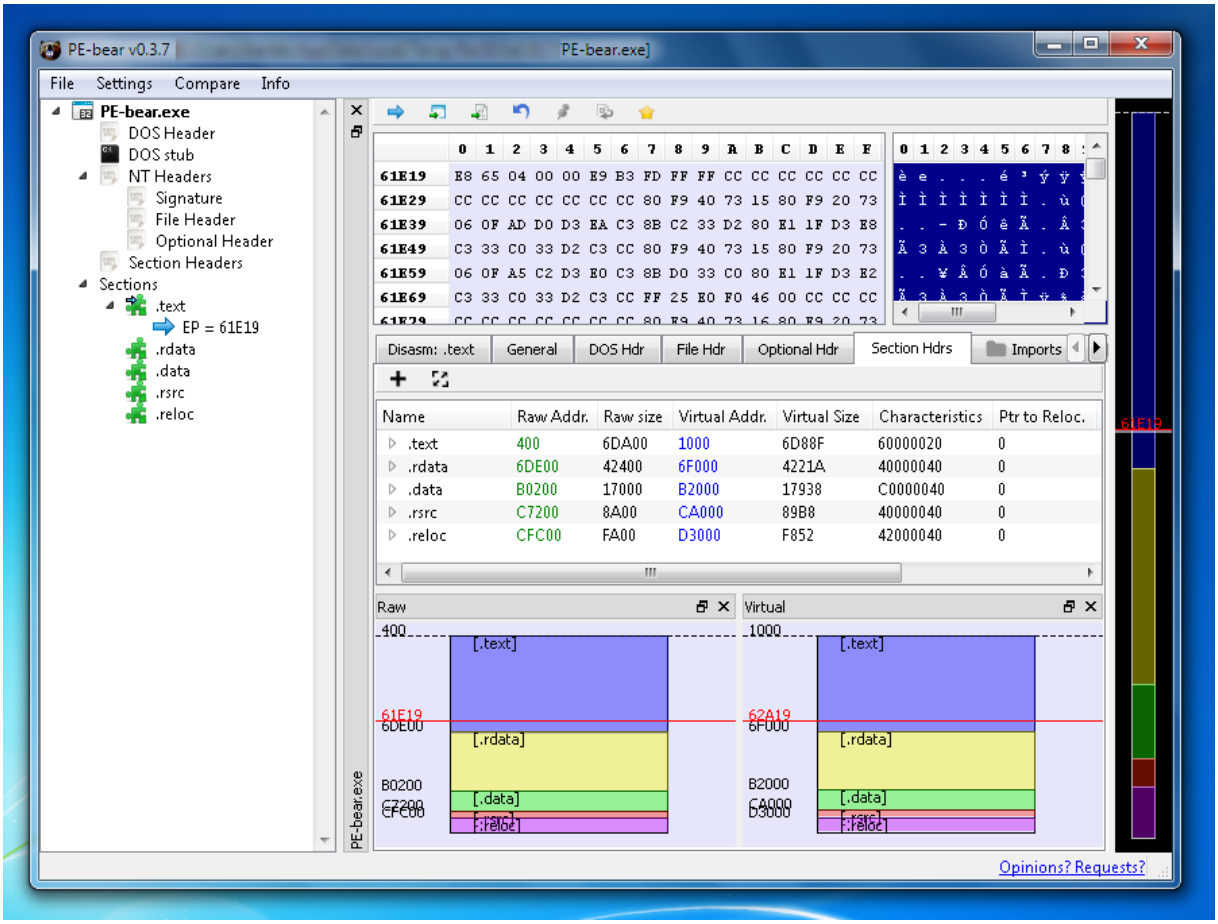
Reverse engineering wymaga wyspecjalizowanych narzędzi do konkretnych celów, oprócz standardowych narzędzi jak deasemblery, dekompilatory czy debuggery – powstało wiele dedykowanych narzędzi, które pomagają w analizie aplikacji oraz edytorów, kilka z nich znajdziecie poniżej.

PE-Bear

Znakomita przeglądarka i edytor struktur plików wykonywalnych z wbudowanym prostym deasemblerem, porównywarką plików *PE* na podstawie wartości ze wszystkich struktur (unikalne rozwiązanie nawet na skalę światową), detekcją sygnatur popularnych *exe-packerów / exe-protectorów*, hex edytor i wizualizacja graficzna struktur sekcji.

Narzędzie stworzone przez polską programistkę (tak – dobrze przeczytałeś) idealnie nadaje się do niskopoziomowej analizy plików *PE/PE32+*, docelowo przeznaczone do analizy złośliwego oprogramowania.

Rysunek 18. Edytor PE-Bear

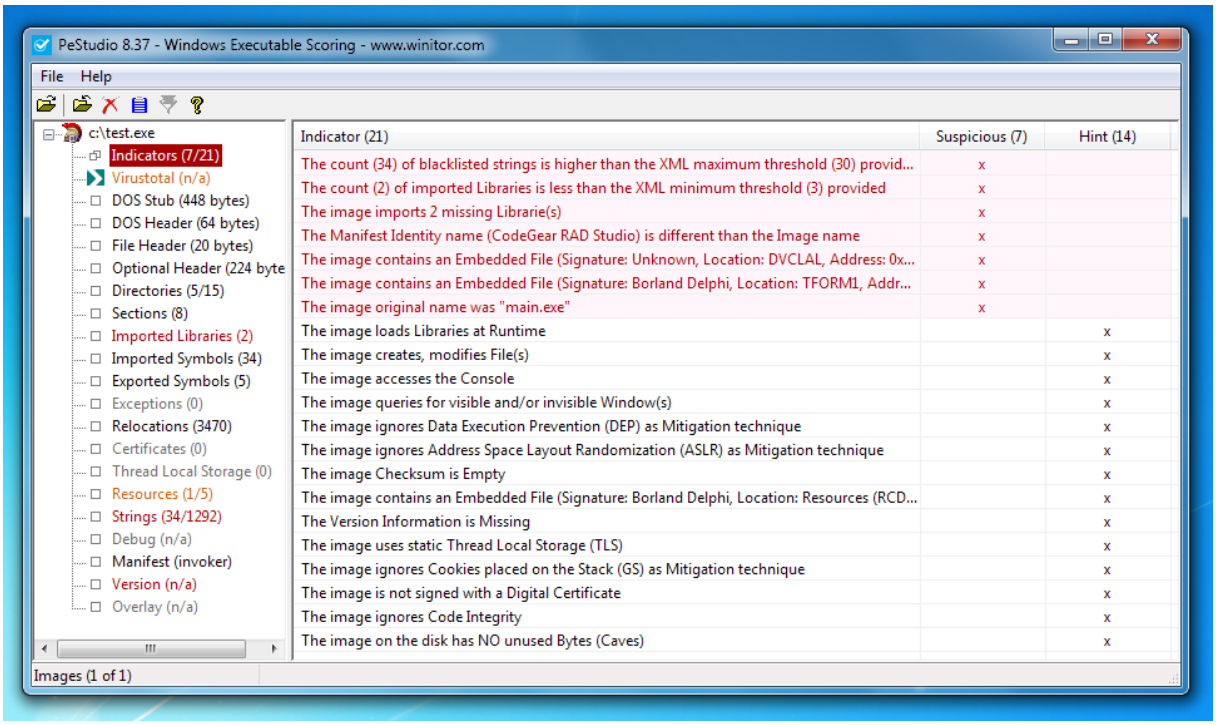


Strona	http://hshrzd.wordpress.com/pe-bear/
Licencja	Freeware
Zalety	• Unikalne funkcje • Prosta edycja struktur plików *PE/PE32+* • Detekcja popularnych *exe-packerów* i *exe-protectorów* na podstawie sygnatur • Wersja dla systemu Windows oraz Linux
Wady	• Prosty deasembler (zbyt prosty) • Brak opcji konfiguracyjnych
Alternatywy	• *CFF Explorer* – edytor PE - http://www.ntcore.com/exsuite.php • *Cerbero Profiler* – zaawansowany profiler - http://cerbero.io/profiler/ • *PE Insider* – tego samego autora co *CFF Explorer* i *Cerbero Profiler* - http://cerbero.io/peinsider/ • *PE Explorer* – edytor PE - http://www.heaventools.com/

PeStudio

Ciekawe narzędzie, które oprócz tego, że wyświetla podstawowe informacje o pliku wykonywalnym, posiada również zbiór reguł, które potrafią wykryć nieprawidłowe elementy w strukturze pliku wykonywalnego (wszelkiego rodzaju anomalie) oraz elementy, które potencjalnie mogą świadczyć o infekcji pliku. Bardzo przydatne narzędzie dla osób, które, na co dzień pracują z plikami *PE*.

Rysunek 19. PeStudio



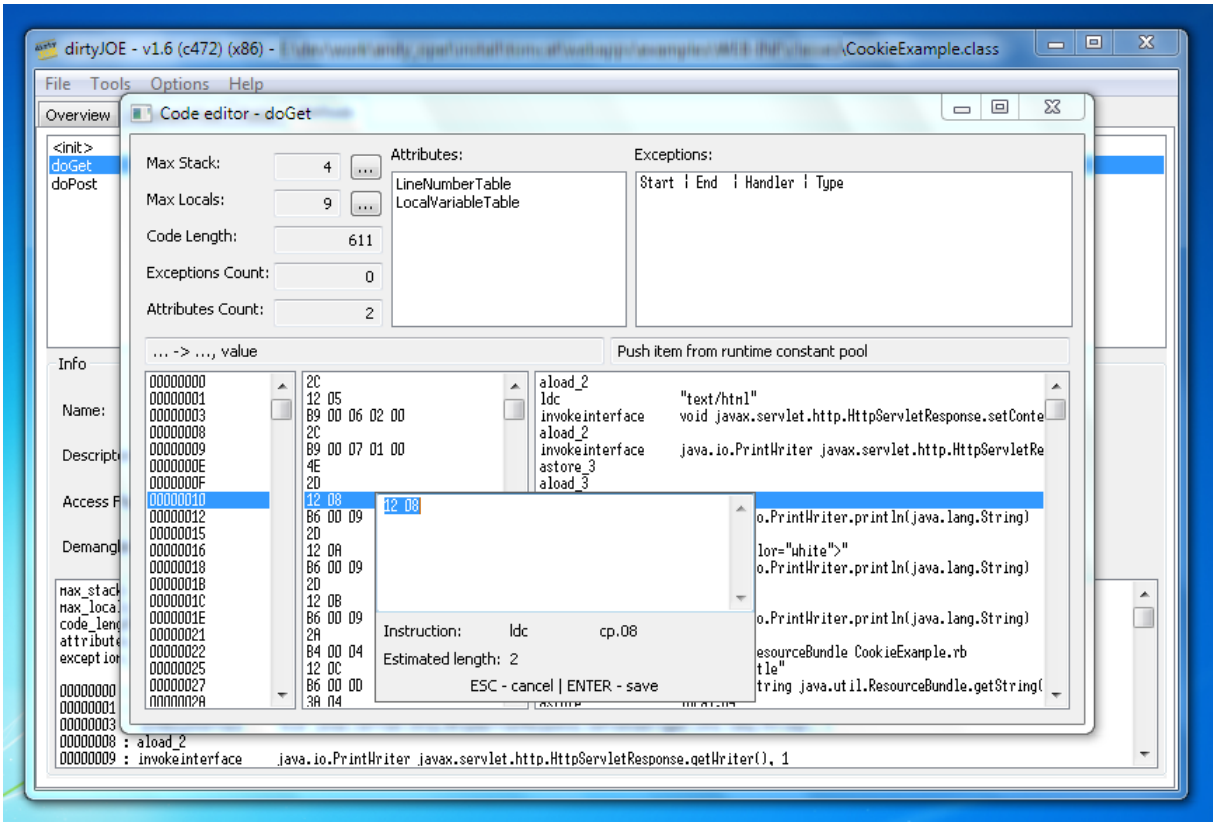
Strona	http://www.winitor.com
Licencja	Darmowa do niekomercyjnego użytku.
Zalety	• Wykrywanie anomalii w plikach wykonywalnych • Wygodna przeglądarka struktur plików *PE*
Wady	• Niektóre reguły są zbyt wyśrubowane

dirtyJOE

Zaawansowany edytor dla skompilowany plików Java. Unikalne narzędzie polskiego autora do modyfikacji kodu z wbudowanym deasemblerem oraz assemblerem, edytor pozwala również na modyfikację wszelkich struktur w skompilowanych plikach **.class*. *dirtyJOE* przydaje się w przypadku, gdy chcemy zmodyfikować zabezpieczone pliki (po użyciu obfuscatora dla Java), gdy

tradycyjne metody dekompilacji, modyfikacji i rekompilacji zawiodą, *dirtyJOE* okazuje się niezastąpiony.

Rysunek 20. Edytor dirtyJOE



Strona	http://dirty-joe.com
Licencja	Darmowa do niekomercyjnego użytku.
Zalety	• Deasembler i assembler instrukcji *JVM* • Dodawanie i edycja pól takich jak np. ciągi tekstowe • Wersja 32 i 64 bitowa • Wtyczka dla Total Commander
Wady	• Surowy interfejs • Niewygodny edytor kodu

Ekstraktory i rippery

Pliki aplikacji jak i same paczki aplikacji potrafią kryć dodatkowe informacje, jak np. ukryte ikony, obrazki, pliki dźwiękowe, biblioteki etc. Jeśli chcemy szybko sprawdzić co znajduje się wewnątrz aplikacji lub np. w całym katalogu instalacyjnym oprogramowania, musimy posłużyć się odpowiednim ekstraktorem lub ripperem.

Universal Extractor

Oprogramowanie pozwala na ekstrakcję plików z archiwów, samorozpakowujących się archiwów oraz instalatorów. Niezwykle przydatne, jeśli chcemy bez przeprowadzania instalacji dowiedzieć się, co znajduje się w paczce instalatora, gdzie często znajdują się jakieś dodatkowe skrypty instalacyjne czy pomocnicze biblioteki.

Rysunek 21. Universal Extractor

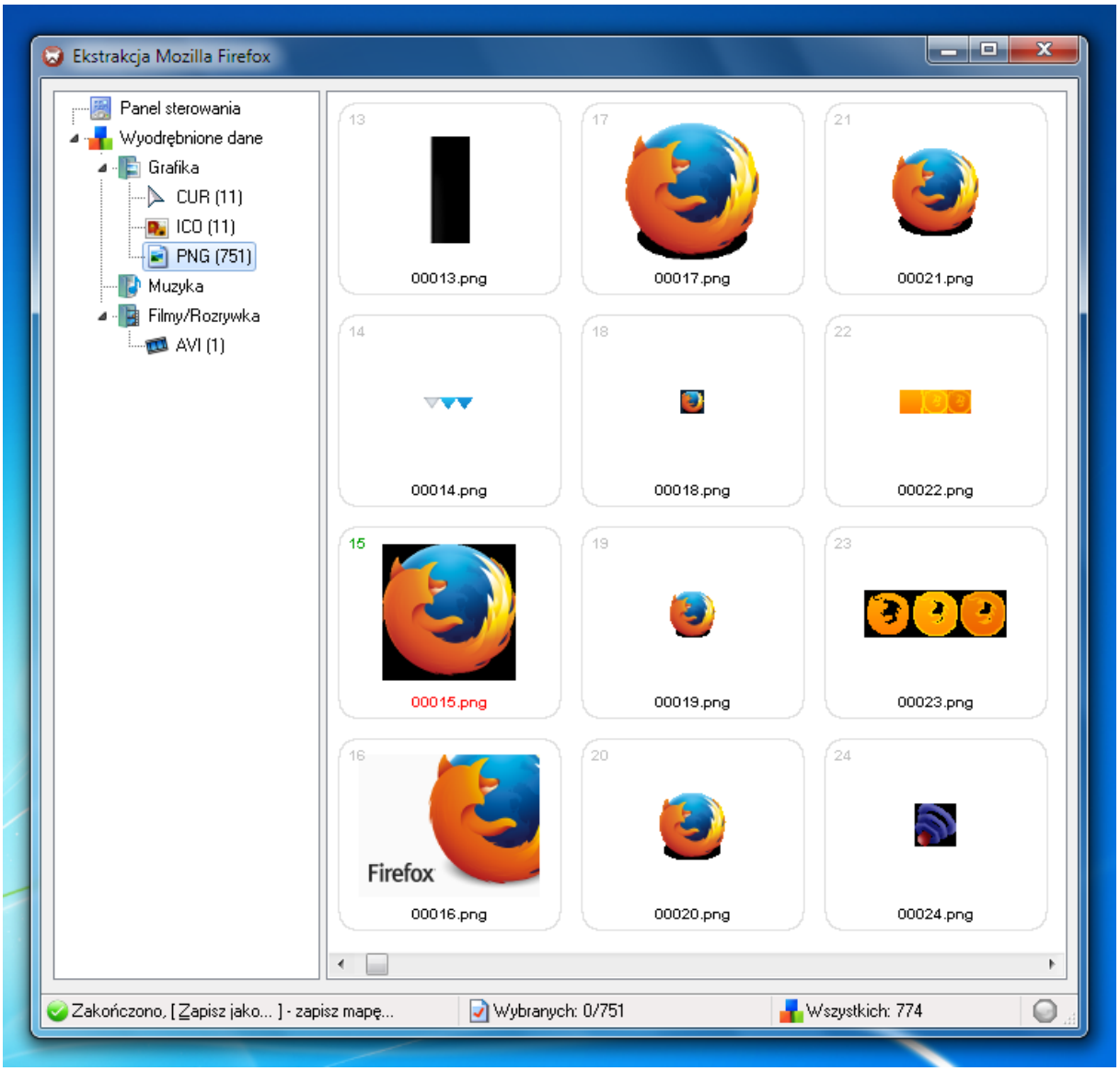
Strona	http://legroom.net/software/uniextract
Licencja	Freeware
Zalety	• Obsługa archiwów (w tym samorozpakowujących) • Ekstrakcja plików z popularnych systemów instalacyjnych
Wady	• Dawno nie było aktualizacji • Czasami trzeba skorzystać z alternatywnych rozwiązań dla nowszych wersji systemów instalacyjnych
Alternatywy	• *innounp* – ekstraktor popularnego systemu instalacyjnego *InnoSetup* - http://innounp.sourceforge.net/

MultiExtractor

Ekstraktor wszelkiego rodzaju plików multimedialnych jak pliki graficzne, ikony, pliki dźwiękowe, filmy, modele 3D, animacje Flash, fonty etc. z dowolnych plików

binarnych. Wbudowane unpakery dla prezentacji multimedialnych, dynamiczne rozpakowywanie danych z pamięci procesów, prosta przeglądarka czynią z tego oprogramowania ciekawe narzędzie, jeśli chcemy szybko podejrzeć, co skrywa się w plikach aplikacji.

Rysunek 22. MultiExtractor



Strona	http://www.multiextractor.com
Licencja	Komercyjna od 19 USD oraz wersja demonstracyjna.
Zalety	• Ekstrakcja wielu formatów plików graficznych • Ekstrakcja z pamięci procesów • Rozpoznawanie popularnych formatów kontenerów

Wady	• Od dłuższego czasu nie było dodanych nowych formatów plików • Czasami potrafi się zawiesić, zwłaszcza przy dużej ilości plików
Alternatywy	• *AllMedia Grabber* – podobny ekstraktor - http://www.fotissoftware.com/multimedia.htm

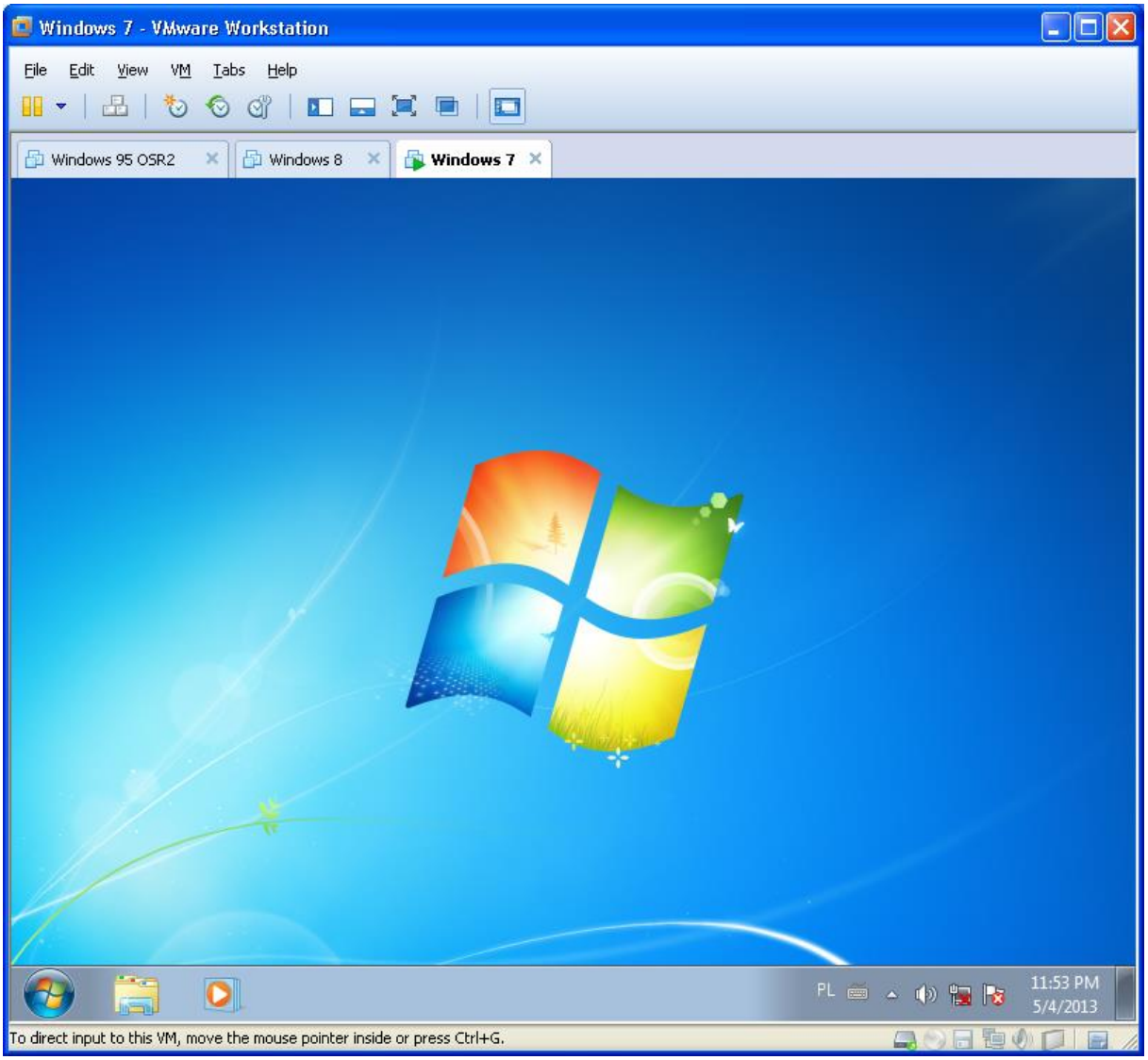
Wirtualne środowiska

Analiza nieznanego oprogramowania może być ryzykowna, zwłaszcza, jeśli musimy uruchomić oprogramowanie i nawet wykonanie tej czynności pod debuggerem może się dla nas źle skończyć, jeśli oprogramowanie gdzieś po drodze uruchomi w tle wątek, a ten zainstaluje nam jakiegoś *rootkita* lub inne złośliwe oprogramowanie. Warto zabezpieczyć się przed takimi przypadkami i uruchamiać takie podejrzane oprogramowanie pod nadzorem wirtualnej maszyny.

VMware

Najbardziej znane oprogramowanie tworzące wirtualne środowisko, gdzie możemy zainstalować dowolny system operacyjny i bez zbędnego ryzyka testować oprogramowanie.

Rysunek 23. VMware Workstation

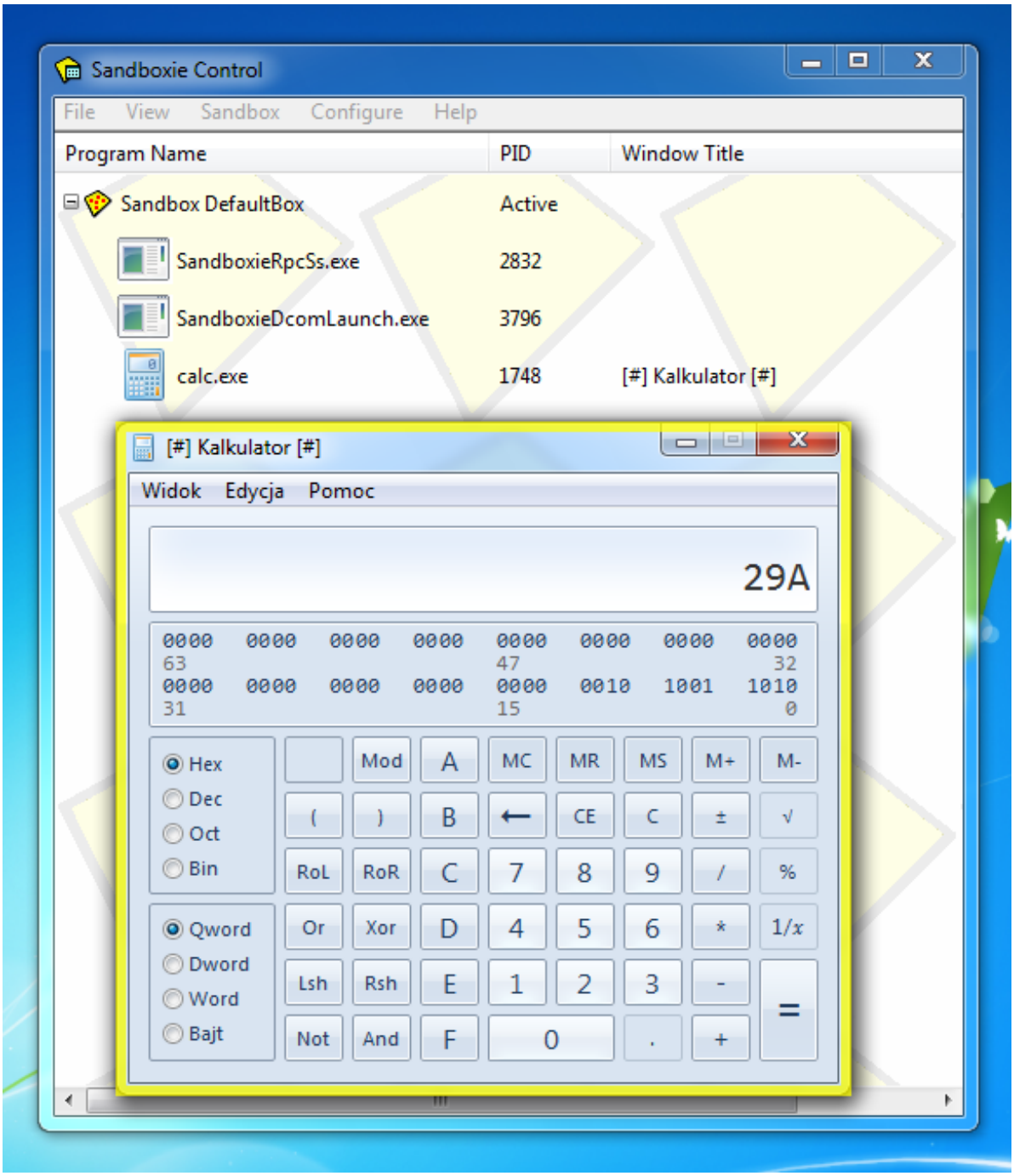


Strona	http://www.vmware.com/products/workstation/
Licencja	Komercyjna od 225 EUR oraz darmowa wersja testowa.
Zalety	• Kompatybilność z wieloma systemami operacyjnymi • Wykorzystanie wirtualizacji sprzętowej do poprawy wydajności
Wady	• Wymaga sporych zasobów sprzętowych i na wolniejszych maszynach potrafi zawiesić cały system
Alternatywy	• *VirtualBox* – darmowy odpowiednik od Oracle - https://www.virtualbox.org • *Parallels Desktop* – wirtualizacja dla Mac OS - http://www.parallels.com/eu/products/desktop/

Sandboxie

Oprogramowanie tworzące wirtualną piaskownicę dla uruchamianych aplikacji. Wszelkie operacje wykonywanie w wirtualnej piaskownicy nie mają wpływu na system, gdzie uruchamiane są tak odizolowane aplikacje. Idealne rozwiązanie do debugowania lub szybkiego sprawdzenia poprawności działania aplikacji bez konieczności obawy o skutki uboczne.

Rysunek 24. Sandboxie



Strona	http://www.sandboxie.com

Licencja	Komercyjna od 15 EUR oraz darmowa wersja testowa.
Zalety	• Doskonała izolacja uruchamianych aplikacji bez konieczności stosowania dedykowanych wirtualnych środowisk
Wady	• Troszkę przestarzały interfejs użytkownika
Alternatywy	• *WinJail* – chroot dla Windows - http://www.winquota.com/wj/index.html

To nie koniec, a dopiero początek...

Zaprezentowane narzędzia to jedynie skrawek tego, co można znaleźć na rynku, istnieje wiele innych darmowych czy eksperymentalnych projektów oraz takich, które zostały w pewnym momencie porzucone, ale są warte zainteresowania. Zachęcam Was do odkrywania tajników reverse engineeringu, a jeśli znajdziecie coś interesującego – napiszcie maila.

O Autorze

Bartosz Wójcik (support@pelock.com)

Autor interesuje się filozofią zachodu, posiada czarny pas w jodze, pomiędzy oglądaniem *Futuramy* i *South Park* spędza czas na nie wiadomo czym, poza tym jest zwolennikiem zamkniętego oprogramowania i zagorzałym aktywistą działającym na rzecz diety wysokoglutenowej.