Dependency Walker

Laura Dabelić, 003645771

Mentor: prof. Marin Golub

Akademska godina 2014/2015

Sadržaj

1. Uvod

Dependency Walker je besplatan program koji skenira sve 32-bitne ili 64-bitne module (engl. *modules*) i gradi prikaz svih međusobno ovisnih modula u obliku hijerarhijskog stabla. Za svaki pronađeni modul ispisuje listu funkcija koje je taj modul koristio i koje od tih funkcija su pozivali i drugi moduli.

Koristi se pri praćenju i otkrivanju grešaka na operacijskom sustavu uzrokovanih progrešnim učitavanjem i izvođenjem modula. Neki od problema koje Dependency Walker može otkriti su:

- moduli koji nedostaju
- neispravni moduli
- pogreške u uvozu i izvozu (engl. *import* i *export*)
- problem cirkularnih ovisnosti (engl. circular dependency errors)
- pogreške pri inicijalizaciji modula

Može obrađivati bilo koji 32- bitni ili 64- bitni modul i moguće ga je pokrenuti na sljedećim Windows operacijskim sustavima:

- Windows 95
- Windows 98
- Windows Me
- Windows NT
- Windows 2000
- Windows XP
- Windows 2003
- Windows 7
- Windows 8

Moguće ga je preuzeti na stranici http://www.dependencywalker.com/. Instalacija nije potrebna, treba samo pokrenutni datoteku *depends*.

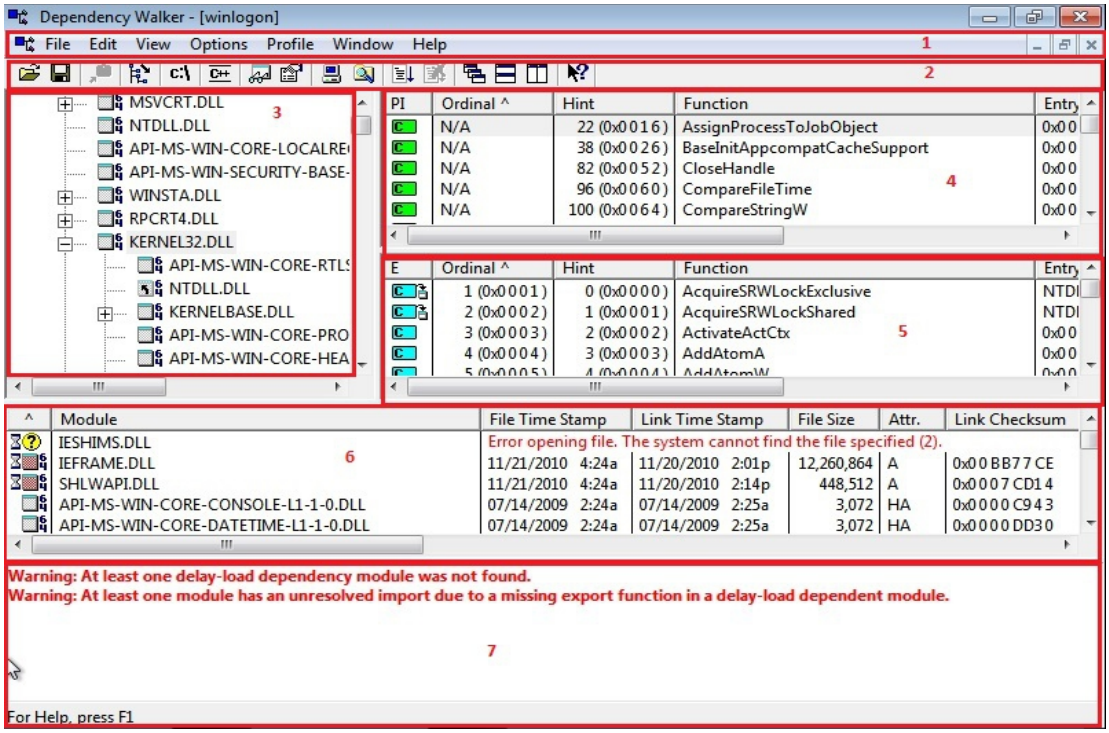
Dependency Walker je uključen kao dio nekoliko Microsoftovih proizvoda. Moguće ga je besplatno preuzeti i sa Microsoftovih stranica. Neki od proizvoda s kojima dolazi su:

- Visual Studio
- Visual C++
- Visual Basic
- Windows 2000/XP/2003 support tools
- Platform SDK
- Windows DDK
- Windows SDK
- MSDN

2. Grafičko sučelje

Grafičko sučelje Dependency Walkera kad otvorimo program winlogon prikazano je na slici 2.1. Glavni prozor grafičkog sučelja možemo podijeliti na 7 logičkih cjelina:

1. Meni
2. Glavna alatna traka
3. Stablo ovisnosti modula
4. Lista uvedenih roditeljskih funkcija
5. Lista izvezenih funkcija
6. Lista modula
7. Zapisnik



Slika 2.1. - Grafičko sučelje Dependency Walkera

2.1. Meni

Meni nudi sljedeće opcije:

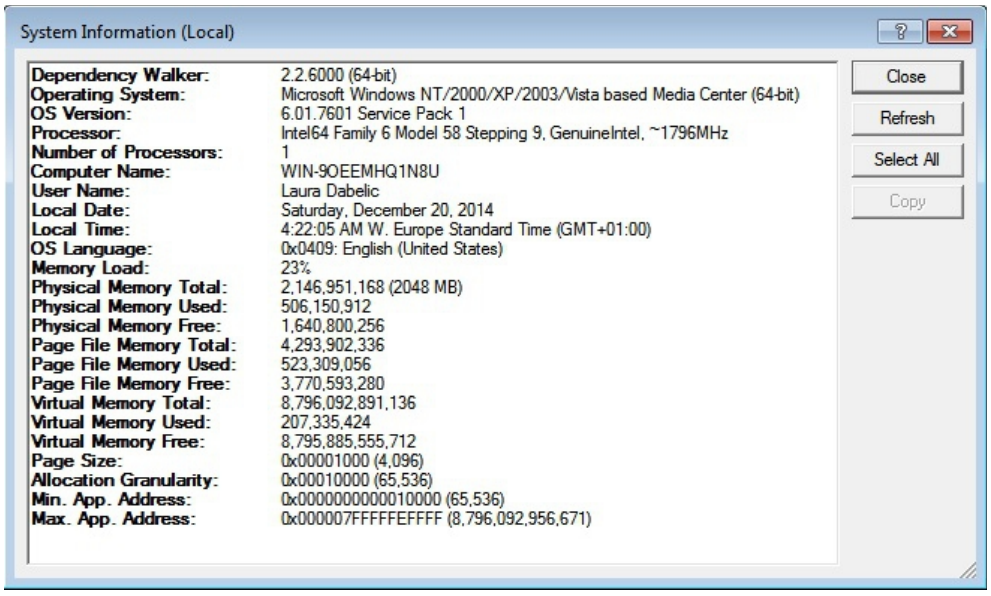
- **File** – pruža mogućnosti otvaranja modula, zatvaranja i snimanja stanja prikaza te otvaranja specifičnog modula
- **Edit** – pruža mogućnosti nalaženja specifičnog teksta u prozoru zapisnik i brisanja ispisa tog prozora
- **View** – omogućava kontrolu prikaza modula
- **Options** – pruža mogućnosti dodatne konfiguracije modula
- **Profile** – pruža mogućnosti profiliranja modula

- **Window** – pruža mogućnosti podešavanja izgleda prozora grafičkog sučelja
- **Help** – pruža pomoć pri korištenju alata

2.1.1 Opis pojedinih opcija menia

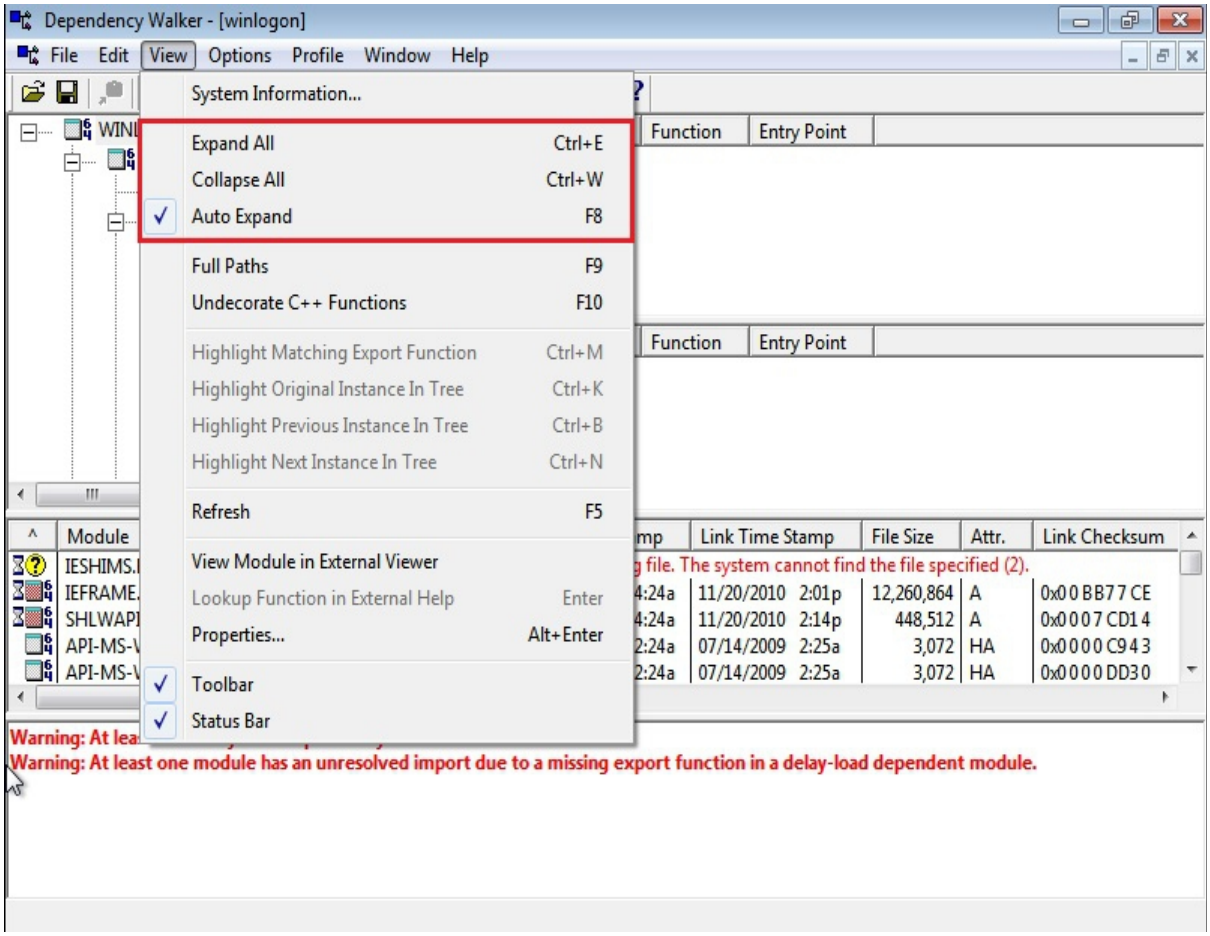
Ako u meniu odaberemo karticu *View* i opciju *System Information*, otvorit će nam se prozor sa popisom korisnih informacija o našem sustavu kako je prikazano na slici 2.2. Neke od tih informacija su:

- Inačica Dependency Walkera koju koristimo
- puno ime operacijskog sustava
- inačica operacijskog sustava
- puno ime procesrora
- broj procesora
- ime računala
- datum i vrijeme
- podaci o raspoloživoj fizičkoj memoriji
- podaci o raspoloživoj virtualnoj memoriji



Slika 2.2. - prozor System Information

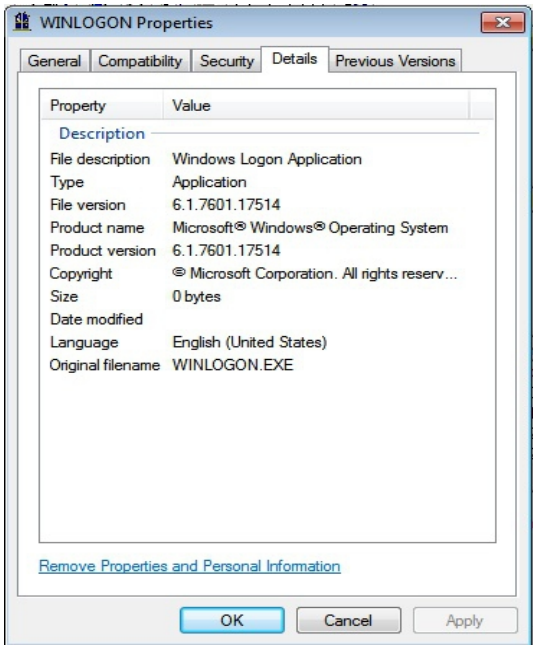
U kartici View imamo dostupne opcije za podešavanje prikaza stabla ovisnosti modula kako se vidi na slici 2.3. Ako je odabrana opcija *Auto Expand*, tada se stablo ovisnosti modula automatski proširuje kako se novi moduli dodaju. Preostale dvije opcije su *Expand All* pomoću koje možemo vidjeti apsolutno sve učitane module u stablu ovisnosti i *Collapse All* pomoću koje se sakriva prikaz svih ovisnih modula i ostaje samo korijenski modul.



Slika 2.3. - opcije za podešavanje prikaza stabla ovisnosti modula

Dodatne informacije o odabranom modulu možemo dobiti ako u kartici *View* odaberemo opciju *Properties*. Otvorit će nam se prozor prikazan na slici 2.4. Prozor *Properties* podijeljen je na nekoliko podcjelina:

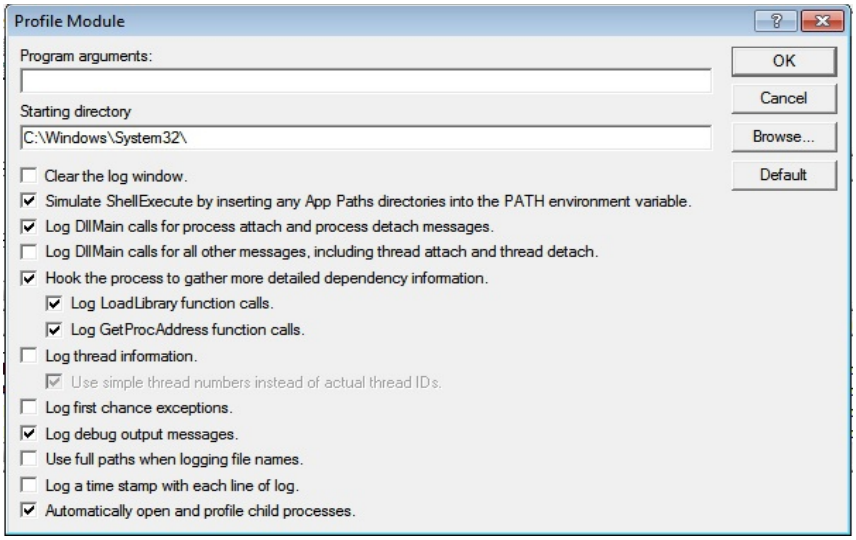
- **General** – općenite informacije o modulu
- **Compatibility** – informacije o kompatibilnosti modula sa operacijskim sustavom
- **Security** – informacije o korisnicima na sustavu i njihovim ovlastima
- **Details** – detalji o samom modulu kao što su opis, tip, inačica i korporacijski potpis
- **Previous Versions** – informacije o prethodnim inačicama modula



Slika 2.4. - Prikaz prozora *Properties* sa opcijom *Details*

Ako u kartici Profile odabaremo opciju Start Profiling, otvorit će nam se prozor prikazan na slici 2.5. Ovdje možemo detaljno podesiti (profilirati) što točno želimo da određeni modul radi kako bismo mogli promatrati njegovo ponašanje.

Za analizu zloćudnih programa posebno je zanimljivo što ovdje možemo podesiti da se bilježe aktivnosti o pozivanjima funkcija LoadLibrary i GetProcAddress jer njih često koriste zloćudni programi. Te funkcije se inače ne nalaze zapisane u zagljavlju programa (PE zaglavlje). Te dvije funkcije omogućavaju programu da pristupi bilo kojoj funkciji u bilo kojoj knjižnici operacijskog sustava. Kad su te funkcije korištene prilikom statičke analize nemoguće je odrediti koje funkcije su bile povezane sa zloćudnim programom.



Slika 2.5. - prozor za profiliranje

2.2. Glavna alatna traka

Ako nekom od alata želimo brzo pristupiti, za to nam može poslužiti alatna traka. Funkcionalnosti koje alatna traka pruža vidljive su u tablici 2.1.

Ikona	Ime ikone i prečac	Opis
	Open (Ctrl + O)	Otvara modul
	Save (Ctrl + S)	Snima trenutni prikaz modula u datoteku
	Copy (Ctrl + C)	Kopira odabrano područje kao tekst
	Auto Expand (F8)	Stablo ovisnosti modula se automatski proširuje kako se novi moduli dodaju
	View Full Paths (F9)	Prikazuje ili sakriva prikaz pune putanje na disku modula
	Undecorate C++ Functions (F10)	Uključuje ili isključuje opciju dekoriranja C++ funkcija
	View Module In External Viewer (Enter)	Za odabrane module pokreće pregled vanjskih modula
	Properties (Alt + Enter)	Otvara prozor *Properties* za odabrane module
	View System Information	Prikazuje informacije o sustavu
	Configure Module Search Order	Podešava redosljed pretrage ovisnih modula
	Start Profiling (F7)	Započinje profiliranje trenutnog prikaza modula
	Stop Profiling (Shift + F7)	Prekida profiliranje

	Cascade Windows	Kaskadno poredava prozore grafičkog sučelja
	Tile Horizontally	Poredava prozore horizontalno
	Tile Vertically	Poredava prozore vertikalno
	Context Help	Pruža pomoć pri pregledu određenog modula

Tablica 2.1. - Funkcionalnosti alatne trake

2.3. Stablo ovisnosti modula

Prikazuje hijerarhijski međuovisnosti između modula. Početni modul je korijenski (engl. r*oot)* koji smo odabrali otvoriti, a niže razine stablaste strukture prikazuju module koji su ovisni o njemu. Da bi spriječio kružno ispisivanje međusobno ovisnih modula, Dependency Walker prestaje sa ispisom određene grane kad naiđe na modul koji je već negdje obradio u ispisu. Dvostruki moduli označeni su malom strelicom kako je vidljivo u tablici 2.2.

Dependency Walker također provjerava koje funkcije modula su bile proslijeđene drugim modulima. Također se provjerava je li svaki od modula ispravni 32-bitni ili 64-bitni modul. Ovdje se ujedno provjeravaju i cirkularne ovisnosti.

Moduli se mogu prikazivati samo imenom ili punom putanjom na disku. Možemo određivati koliko toga želimo vidjeti opcijom *Full Paths*.

Značenje pojedinih ikona u stablu dano je u tablici 2.2. Neke ikone predstavljaju normalne module, a neke upozorenja i greške

Ikona	**Opis**
	Normalni modul.
	Dvostruki modul. Ovaj modul je već bio obrađen negdje u stablu ovisnosti.
	Proslijeđeni modul. Ovaj modul je ovisan zato što mu je modul-roditelj proslijedio neku od svojih funkcija.

	Dinamički modul. Ovaj modul bio je dinamički učitan od strane modula-roditelja.
	Modul koji je bio dinamički učitan pozivom funkcije LoadLibraryEx.
	64-bitni modul.
	Modul koji nedostaje.
	Neispravni modul.
	Modul sa upozorenjem. Ovom modulu nedostaje jedna ili više funkcija za izvoz modula-roditelja ili se nije uspio inicijalizirati tijekom učitavanja.
	Modul sa upozorenjem da mu se učitavanje odužilo. Također mu nedostaje jedna ili više funkcija za izvor modula-roditelja ili se nije uspio inicijalizirati tijekom učitavanja.
	Dinamički modul sa upozorenjem. Ovome modulu nedostaje jedna ili više funkcija kojima je modul-roditelj pokušao pristupiti pomoću GetProcAddress ili ima iste probleme opisane u prethodna dva odjeljka tablice.

Tablica 2.2. - Ikone u stablu ovisnosti modula

2.4. Lista uvedenih roditeljskih funkcija

Ispisuje listu uvedenih roditeljskih funkcija za odabrani modul. Za implicitne i proslijeđene ovisnosti odabrani modul mora izvoziti svaku funkciju koju modul-roditelj uključuje iz njega. Ako se ne dogodi izvoz funkcije koju modul-roditelj očekuje pozvati, dolazi do pogreške (engl. *unresolved external error*). Dependency Walker provjerava da li su sve funkcije ispravno izvezene.

C++ funkcije se mogu prikazivati u čitljivom formatu (engl. *human readable format*) ili u svom izvornom obliku (engl. *decorated form*).

Podprozor sa listom uvedenih roditeljskih funkcija podijeljen je na 5 stupaca:

- **PI** – *Parent Imports*. Ikone prikazane u ovom stupcu mogu biti zelene i crvene. Zelene prikazuju riješene uvoze (engl. *resolved import*) dok crvene prikazuju neriješene uvoze (engl. *unresolved import*).
- **Ordinal** – ordinalna vrijednost funkcije u obliku broja. Vrijednost ovog stupca može biti N/A što znači da je funkcija uvezena pod svojim imenom.
- **Hint** – *hint* vrijednost za uvezenu funkciju. Ovu vrijednost interno koristi operacijski sustav

da uvezene vrijednosti pridruži izvezenima.

- **Function** – ime uvezene funkcije ako je funkcija uvezena pod svojim imenom. Vrijednost ovog stupca može biti N/A što znači da je funkcija uvezena pomoću ordinalne vrijednosti.
- **Entry Point** – ulazna memorijska adresa za funkciju

2.5. Lista izvezenih funkcija

Prikazuje listu izvezenih funkcija za trenutno odabrani modul. Izvedene funkcije su one funkcije koje modul izlaže drugim modulima. Mogu se smatrati sučeljem modula. Ova lista koristi se za provjeru neriješenih vanjskih grešaka (engl. *unresolved external errors*) u odabranom modulu.

Prilikom pretraživanja liste, Dependency Walker provjerava za svaku funkciju je li zaista proslijeđena. Proslijeđena funkcija je ona funkcija koja je izvezena iz određenog modula, ali njezin kod se zapravo nalazi u drugom modulu. Dependency Walker također učitava proslijeđeni modul ako je potrebno.

Podprozor je podijeljen na 5 stupaca od kojih 4 imaju iste nazive i isto značenje kao što je opisano u odjeljku 2.4. Jedina razlika je u prvom stupcu koji se u ovom podprozoru umjesto **PI** zove **E** što je kratica od *exports*. Ikone u ovom stupcu mogu biti plave i sive. Za detaljnije informacije o njima čitatelja se upućuje na [1].

2.6. Lista modula

Prikazuje listu svih modula koji su ovisni o korijenskom modulu (engl. *root module*) kojega je korisnik otvorio. Lista definira niz datoteka potrebnih da bi se modul učitao i pokrenuo kao proces. Moduli mogu biti prikazani samo imenom ili punom putanjom na disku. Korisnik može kontrolirati koliko toga želi vidjeti opcijom *Full Paths*.

Postoji razlika u prikazu koji daje lista modula i stablo ovisnosti modula. Lista modula prikazuje samo jedinstvene module dok stablo ovisnosti modula prikazuje sve odnose između modula. Moduli koji se u stablu ovisnosti modula prikazuju puno puta u listi modula prikazat će se samo jednom.

Značenja ikona koje se pojavljuju opisana su u tablici 2.2. Podprozor je podijeljen na sljedeće stupce:

- **Module –** ime modula ili puna putanja modula na disku
- **File Time Stamp** – datum i vrijeme kada je modul zadnji put bio snimljen
- **Link Time Stamp** – datum i vrijeme kada je modul bio izgrađen
- **File size –** veličina modula
- **Attr. –** atributi modula. Vrijednosti atributa dane su u tablici 2.3
- **Link Checksum –** suma za provjeru kada je modul bio izgrađen
- **Real Checksum** – stvarna suma modula
- **CPU** – tip procesora za koji je modul bio izgrađen
- **Subsystem –** vrsta podsistema u kojemu bi se modul trebao izvoditi
- **Symbols** – tipovi simbola koji se koriste pri ispravljanju pogrešaka (engl. *debugging*)

- **Preffered Base** – preferirana bazna adresa za učitavanje modula
- **Actual Base –** stvarna bazna adresa za učitavanje modula
- **Virtual Size** – virtualna veličina modula
- **Load Order** – redosljed učitavanja modula uzimajući u obzir druge module
- **File Ver –** inačica datoteke modula
- **Product Ver** – inačica proizvoda modula
- **Linker Ver –** inačica programa za povezivane (engl. *linker*) koja je stvorila modul
- **OS Ver –** inačica operacijskog sustava kojemu je modul namijenjen
- **Subsystem Ver** - inačica podsustava kojemu je modul namijenjen

Tablica 2.3 prikazuje atribute modula opisane u Attr.

Vrijednost atributa	Značenje
R	Samo za čitanje (engl. *read only*)
H	Sakriveni atribut (engl. *hidden*)
S	System
A	Archive
C	Komprimirani (sažeti) atribut (engl. *compressed*)
T	Privremeni atribut (engl. *temporary*)
O	Atribut koji nije spojen na mrežu (engl. *offline*)
E	Kriptirani atribut (engl. *encrypted*)

Tablica 2.3. - značenje i opis atributa modula

2.7. Zapisnik

U ovom podprozoru ispisuju se informacije o pogreškama, upozorenjima i svim aktivnostima modula. Prilikom izvođenja, Dependency Walker prikuplja informacije o trenutno aktivnim procesima. Vrste informacija koje se mogu naći u zapisniku su:

- početak novog procesa
- završetak procesa
- stvaranje nove dretve
- zatvaranje dretve
- učitavanje modula
- zatvaranja modula
- bilo koji tekst o pogreškama generiran od strane procesa
- bilo koja iznimka koja se pojavi u nekom od procesa
- pozivanje funkcije tipa LoadLibrary (važno kod analize zloćudnih programa)
- odgovor na poziv funkcije LoadLibrary
- bilo koji poziv funkcije GetProcAddress

Posljednje tri nabrojane stavke vrlo su važne kod analize zloćudnih programa i njima se prilikom analize posvećuje posebna pozornost.

3. Korištenje Dependency Walkera za analizu zlućudnih programa

Format datoteke može nam puno otkriti o njezinoj funkcionalnosti. Format koji koriste Windows izvodive (engl. *executable*) datoteke jest PE (engl. *Portable Executable*). To je struktura podataka koja sadrži informacije potrebne operacijskom sustavu da izvede kod određenog programa.

PE datoteke počinju zaglavljem u kojemu su sadržane neke od sljedećih informacija:

- informacije o kodu
- vrsta aplikacije
- potrebne funkcije knjižnica (engl. *library functions*)
- prostor koji treba biti dostupan na disku
- informacije o svakoj knjižnici koja će biti učitana
- informacije o svakoj funkciji koju će program koristiti

Najčešći dijelovi PE datoteke dani su u tablici 4.1.

.text	Sadrži izvodivi kod programa
.rdata	Sadrži podatke samo za čitanje (engl. *read-only*). Ovdje se nalaze informacije o uvozu i izvozu (engl. *import* i *export*) koje može dohvatiti Depndency Walker.
.data	Sadrži globalne podatke kojima se može pristupiti iz bilo kojeg dijela programa
.idata	Sadrži informacije o uvezenim funkcijama
.edata	Sadrži informacije o izvezenim funkcijama
.pdata	Postoji samo kod 64-bitnih datoteka i sadrži informacije za upravljanje iznimkama

Tablica 4.1 – dijelovi PE datoteke

Jedna od najkorisnijih informacija koje možemo prikupiti o izvodivoj datoteci je lista funkcija koje ta datoteka uključuje (engl. *imports*). Uključene funkcije povezane su sa knjižnicama funkcija, a knjižnice funkcija povezuju se sa glavnim programom postupkom koji se zove povezivanje (engl. *linking*). Povezivanje može biti izvedeno na dva načina:

1. Statičko povezivanje
- sav programski kod iz te knjižnice kopiran je u izvorni, izvodivi kod programa
- zbog toga program postaje veći
- kod analize koda teško je odrediti razliku između koda koji je kopiran povezivanjem i koda samog glavnog programa
- razlog je u tome što u PE zaglavlju ne postoji nikakva naznaka o tome da je ikakav kod

bio povezan s kodom glavnog programa

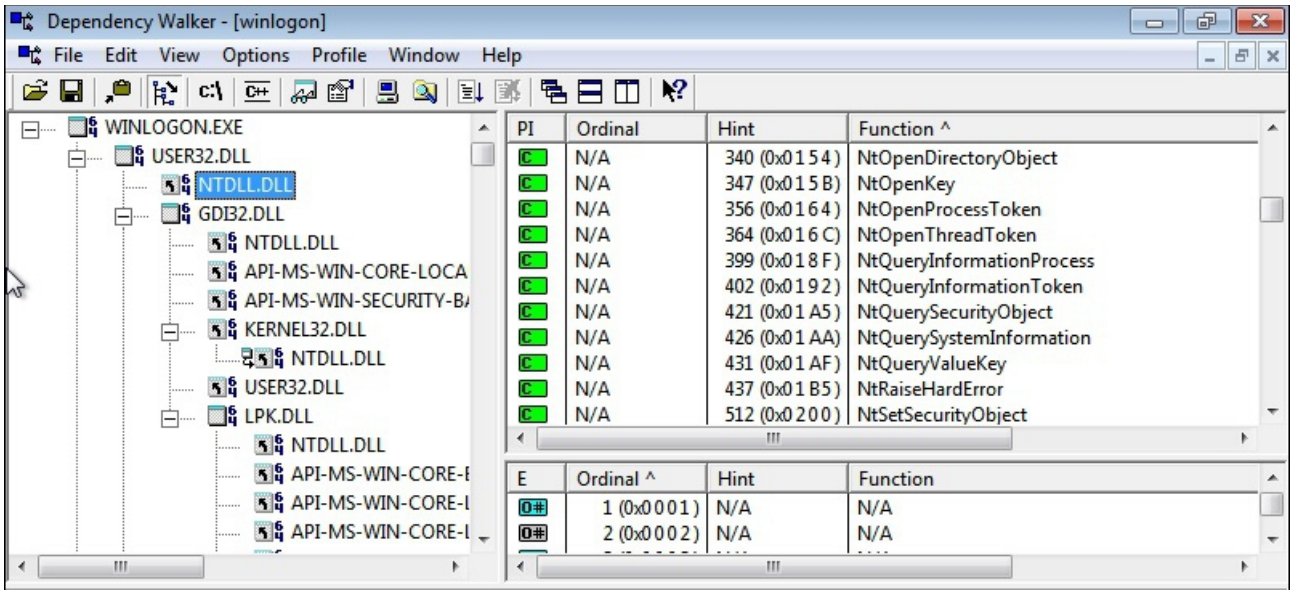
- ovaj način povezivanja veoma je popularan kod malicioznih programa, pogotovo kod onih koji su pakirani (engl. *packed*)

2. Dinamičko povezivanje

- operacijski sustav traži potrebne knjižnice kada je program učitan
- kada program pozove funkciju iz povezane knjižnice (engl. *linked library*), funkcija se izvodi unutar te knjižnice
- zloćudni programi povezuju se s knjižnicama funkcija samo kada je određena funkcija potrebna, a ne pri pokretanju programa

Dependency Walker ispisuje samo funkcije koje su dinamički povezane. Jedan od načina uočavanja malicioznog programa jest da pogledamo koje sve dinamički povezane knjižnice (engl. *dynamic linked library*) program koristi.

Maliciozni program koristit će funkcije koje pripadaju jezgrenom načinu rada operacijskog sustava i kojima korisnik nema izravan pristup. Jedna od takvih funkcija je NTDL.DLL čiji prikaz u Dependency Walkeru je vidljiv na slici 4.1.



Slika 4.1. - ispis funkcije NTDL.DLL

Ova funkcija predstavlja sučelje prema jezgri Windows operacijskog sustava. Programi ne uključuju ovu datoteku izravno, premda je uvijek neizravno koristi funkcija KERNEL32.DLL kao što je i vidljivo na slici 4.1. Sučelje kojemu pristupa funkcija NTDL.DLL ostvaruje funkcije koje uključuju manipuliranje procesima, dretvama i registrama. Neke od tih funkcija uključuju:

- NtOpenProcessToken
- NtOpenThreadToken
- NtCreateKey
- NtDeleteValueKey
- NtEnumerateKey

- NtQueryInformationProcess
- RtlCheckRegistryKey
- RtlPcToFileHeader

Dakle, ako neki program koristi jezgrene funkcije i ako one nisu neizravno uključene, možemo posumnjati da se radi o zloćudnom programu. Također, ako u ispisu imamo jako malu listu datoteka, možemo posumnjati da se radi o pakiranom kodu koji često koriste zloćudni programi kako bi prikrili svoju aktivnost.

Windows sadrži nekoliko funkcija koje omogućavaju programeru da uključi funkcije koje se ne nalaze u zaglavlju programa. Dvije funkcije koje se najčešće koriste spomenute su već prije, a to su LoadLibrary i GetProcAddress. Osim njih, također se veoma često koriste i funkcije LdrGetProcAddress i LdrLoadDll.

Ako vidimo neke od ovih funkcija u ispisu i još k tome ako je cjelokupna lista funkcija poprilično kratka, možemo gotovo sigurno reći da se radi o pakiranom zloćudnom programu.

Osim jezgrenih funkcija, potrebno je promatrati i ispis u stupcu *Ordinal* u listi uvezenih roditeljskih funkcija i listi izvezenih funkcija. Izvodive datoteke mogu uključivati funkcije pomoću parametra navedenog u stupcu *Ordinal* umjesto imena. Prilikom takvog načina uključivanja, ime funkcije se nikada ne pojavljuje u glavnom programu i zbog toga je teže odrediti koja funkcija je bila korištena. Funkciju možemo odrediti po vrijednosti parametra koji je upisan u stupac *Ordinal* umjesto imena funkcije.

4. Zaključak

Dependency Walker je veoma koristan alat u otkrivanju i uklanjanju pogrešaka na sustavu. Također se može koristiti kod analize zloćudnih programa. U tim slučajevima se najčešće kombinira sa nekim od drugih alata za analizu kao što su Pocess Explorer (koji ga može i pokrenuti preko svog korisničkog sučelja), IDA Pro, PEiD ili PEView. Grafičko sučelje je intuitivno i dobro organizirano što olakšava njegovo korištenje. Ipak, da bi korisnik iskoristio sve mogućnosti koje Dependency Walker pruža trebao bi dobro poznavati način rada operacijskih sustava i prevođenja programskih jezika. Stoga se čitatelju preporuča detaljnije izučavanje navedenih područja prije rada s Dependency Walkerom.

5. Literatura

[1] Dependency Walker : http://www.dependencywalker.com/

[2] Fakultet elektrotehnike i računarstva; Radovi iz područja računalne sigurnosti; 2014; http://os2.zemris.fer.hr/index.php?show=start

[3] Michael Sikorski; Andrew Honig. Practical Malware Analysis: The Hands-On Guide To Dissecting Malicious Software. San Francisco: no starch press, 2012