VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

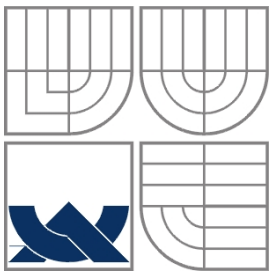
GRAFICKÁ PREZENTACE VÝSLEDKŮ STATISTICKÉ ANALÝZY PE EXE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

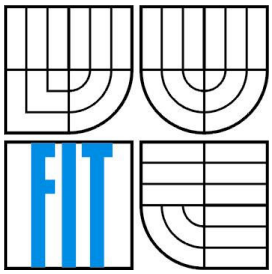
AUTOR PRÁCE
AUTHOR

JAKUB RUSNÁK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GRAFICKÁ PREZENTACE VÝSLEDKŮ STATISTICKÉ ANALÝZY PE EXE

GRAPHICAL VISUALIZATION OF PE EXE STATISTICAL ANALYSIS RESULTS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB RUSNÁK

VEDOUCÍ PRÁCE
SUPERVISOR

DR. ING. PETR PERINGER

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Rusnák Jakub**
Obor: Informační technologie
Téma: **Grafická prezentace výsledků statistické analýzy PE EXE**
Kategorie: Počítačová grafika

Pokyny:

1. Seznamte se s existujícími moduly statistické analýzy spustitelných souborů ve formátu PE a s metodami vizualizace dat.
2. Navrhněte vhodné metody pro grafickou prezentaci výstupů statistické analýzy a program, který umožní různé způsoby zobrazení výsledků analýzy.
3. Navržený program implementujte v prostředí MS Windows.
4. Výsledný program řádně otestujte a zhodnoťte jeho přínos.

Literatura:

- Dle zadání vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních dvou bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Peringer Petr, Dr. Ing.**, UITS FIT VUT
Konzultant: Obluk Karel, Ing., Ph.D., Grisoft
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jakub Rusnák**
Id studenta: 78768
Bytem: Dostojevského 37, 058 01 Poprad
Narozen: 26. 02. 1986, Košice
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Grafická prezentace výsledků statistické analýzy PE EXE
Vedoucí/školitel VŠKP: Peringer Petr, Dr. Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	..... Jakub Kroupa
Nabyvatel	Autor

Abstrakt

Táto bakalárska práca popisuje štatistickú analýzu spustiteľných súborov vo formáte Portable Executable (PE EXE) pomocou mapy entropie, ktorá sa používa na odhalenie počítačových vírusov napadajúcich tieto súbory. Zaoberá sa tiež popisom formátu PE EXE a metódami infekcie počítačových vírusov. Práca sa zameriava na návrh a implementáciu programu na grafické zobrazenie výsledkov tejto analýzy. Program je implementovaný v jazyku c++ s využitím objektového návrhu na platforme Microsoft Windows a využíva 2D grafickú knižnicu Cairo.

Kľúčové slová

Počítačový vírus, entropia, PE EXE, 2D počítačové grafika, vektorová grafika

Abstract

This bachelor's thesis describes statistical analysis of executable files in Portable Executable file format (PE EXE) with the entropy map, which is used to detect computer viruses which attack those files. It describes the PE EXE file format and methods of computer virus infection. The main objective is design and implementation of computer program for visualization of results of this statistical analysis. The application is implemented in c++ programming language with object oriented design on Microsoft Windows platform. It uses the 2D graphic library Cairo.

Keywords

Computer virus, entropy, PE EXE, 2D computer graphics, vector graphics

Citace

Rusnák Jakub: Grafická prezentace výsledků statistické analýzy PE EXE. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Grafická prezentace výsledků statistické analýzy PE EXE

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Dr. Ing. Petra Peringra. Ďalšie informácie mi poskytol Ing. Zdeněk Breitenbacher. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jakub Rusnák
2.5.2008

Poďakovanie

Ďakujem vedúcemu bakalárskej práce Dr. Ing. Petrovi Peringrovi za odbornú pomoc a konštruktívnu kritiku pri spracovaní tejto práce. Tiež ďakujem môjmu konzultantovi z firmy Grisoft, pánovi Ing. Zdenkovi Breitenbacherovi za odborné rady pri riešení praktickej časti.

Obsah

Úvod

Pri odhaľovaní a analýze počítačových vírusov skúmame spustiteľné súbory rôznymi štatistickým metódami. Jedna z nich je počítanie entropie (miery neusporiadanosti) súboru. Pomocou nej môžeme v súbore ľahko nájsť napadnuté miesto alebo porovnať viaceré vzorky konkrétneho vírusu a nájsť tak metódu na jeho odhalenie.

Táto bakalárska práca sa zaoberá návrhom a implementáciou programu na grafické zobrazenie výsledkov tejto štatistickej analýzy spustiteľných súborov vo formáte Portable Executable (PE EXE) na operačnom systéme Microsoft Windows. Práca bola riešená v spolupráci s firmou Grisoft, kde pri odhaľovaní počítačových vírusov využívajú výpočet mapy entropie. Na začiatku bolo potrebné sa dôkladne zoznámiť so štruktúrou PE EXE súborov a s použitím knižnice na výpočet entropie. Cieľom bolo navrhnúť niekoľko grafov, ktorými by sa dala entropia názorne reprezentovať a implementovať program, ktorý tieto grafy zobrazí.

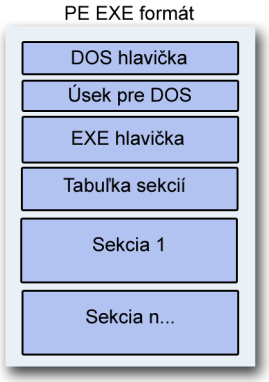
Prvá kapitola sa zaoberá popisom formátu PE EXE, metódami infekcie počítačových vírusov a výpočtom mapy entropie. Druhá kapitola popisuje návrh jej grafickej reprezentácie. Tretia kapitola sa zaoberá objektovým návrhom programu na zobrazenie týchto grafov a štvrtá popisuje jeho implementáciu a testovanie.

1 Prehľad problematiky

1.1 Formát Portable executable

Aby sme mohli štatistiky skúmať spustiteľné súbory a pochopili spôsoby infekcie počítačových vírusov na platforme Microsoft Windows, je treba podrobne spoznať formát spustiteľných súborov na tejto platforme, a to je Portable Executable (ďalej len PE EXE, podrobnejšie viď [11]). Formát je pôvodne odvodený z unixového formátu COFF. Zmysel jeho názvu tkvie v tom, že má byť „prenositeľný“ na akúkoľvek platformu s operačným systémom Windows (aj keby využívala inú CPU platformu, ako Intel). Tento formát používajú dnes všetky spustiteľné súbory a DLL knižnice na platforme Win32.

Najprv si musíme vysvetliť adresovanie v EXE súbore. V jednotlivých štruktúrach sa často nachádza RAW adresa, čo je obyčajný offset od začiatku binárneho súboru na disku. RVA adresa je relatívna virtuálna adresa. PE loader často nahrá binárny súbor do pamäte na iné miesto, ako je preferovaná adresa, pretože tú často obsadzuje už iný proces. Preto sa v PE EXE súbore používa RVA adresa, ktorá predstavuje offset od adresy na ktorú sa spustiteľný súbor nahrá od pamäte.



Obrázok 1.1: časti PE EXE súboru

PE EXE súbor sa skladá z hlavičky a jednotlivých sekcií. V hlavičke sa nachádza popis súboru a jeho sekcií. Prvou časťou hlavičky je DOS hlavička, za ktorou sa nachádza krátky úsek programu pre DOS vypisujúci chybovú hlášku pod DOSom, že tento program nie je pod ním možné spustiť. Nachádza sa tu iba kvôli spätnej kompatibilite. PE loader pod Windows v tejto hlavičke nájde odkaz na PE hlavičku a rovno na ňu preskočí.

PE hlavička (*IMAGE_NT_HEADERS*) obsahuje dôležité informácie a celom PE EXE súbore. Nachádza sa v nej odkaz na štruktúru *IMAGE_FILE_HEADER*. V nej sa nachádzajú napríklad informácie o počte sekcií (*NumberOfSections*). Keď vírus pri infekcii pridáva novú sekciu, toto číslo

zvýši [3]. V premennej *Characteristics* je uložené, či sa jedná a DLL knižnicu alebo spustiteľný EXE súbor.

PE hlavička odkazuje ďalej odkazuje na štruktúru *IMAGE_OPTIONAL_HEADER*, v ktorej sa nachádzajú ďalšie informácie potrebné pre analýzu EXE súboru. Spomenieme niekoľko zaujímavých:

- *AddressOfEntryPoint*, čo RVA adresa prvej inštrukcie binárneho kódu.
- *ImageBase* je nahrávacia adresa vo virtuálnom adresovom priestore, väčšinou to býva adresa 400000h. PE loader však na túto adresu nahrá súbor do pamäte iba vtedy, ak tento priestor už nepatrí inému procesu. Vírusy využívajú tento údaj na vypočítanie presnej pozície určitého prvku [3].
- *SectionAlignment* určuje zarovnanie sekcií v pamäti. Zarovnanie je rovné násobku tohto čísla. Veľa vírusov toto využíva na nájdenie vhodného miesta na uloženie svojho tela [3].
- *FileAlignment* je podobný údaj, ako v predošlom prípade, ale určuje zarovnanie vo fyzickom súbore na disku.
- *SizeOfImage* je veľkosť celého súboru keď je nahraný do pamäte.
- *SizeOfHeaders* určuje veľkosť všetkých spomínaných hlavičiek a tým pádom a začiatok prvej sekcie.

Hneď za hlavičkou sa nachádza pole štruktúr *IMAGE_SECTION_HEADER*, ktoré ukladá informácie o každej sekcii. Počet týchto štruktúr je rovný počtu sekcií. Upravením týchto tabuliek vírus vytvorí novú sekciu, alebo zaistí možnosť vloženia do niektorej existujúcej sekcie [3]. Spomenieme zopár dôležitých údajov potrebných pre lokalizáciu a načítanie jednotlivých sekcií:

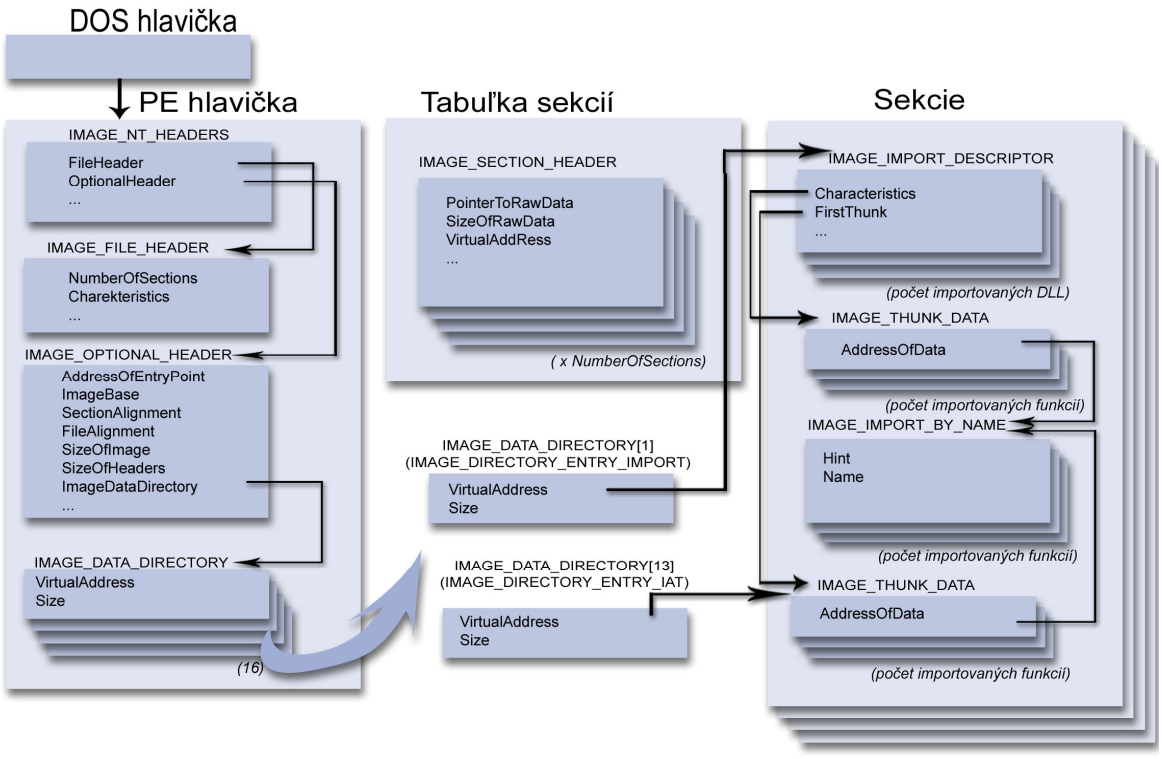
- *PointerToRawData* ukazuje na začiatok sekcie v EXE súbore na disku.
- *SizeOfRawData* určuje veľkosť sekcie vrátane zarovnania
- *VirtualAddress* obsahuje RVA adresu začiatku sekcie. Pomocou tohto údaju a RAW adresy v pointerToRawData môžeme teda lokalizovať ľubovoľnú inú štruktúru odkazujúcu sa pomocou RVA adresy do niektorej sekcie (napríklad štruktúry v *IMAGE_DATA_DIRECTORY* spomínané neskôr).

Ďalšou dôležitou štruktúrou, na ktorú obsahuje hlavička je pole 16 štruktúr *IMAGE_DATA_DIRECTORY*. V ňom sa nachádzajú informácie o dôležitých častiach súboru, nezávisle na tom, v akej sekcii sa nachádzajú. Jedna položka obsahuje RVA adresu štruktúry na ktorú odkazuje a jej veľkosť. Skutočné umiestnenie v EXE súbore je možné vypočítať na základe RVA adries jednotlivých sekcií. Pomocou RVA adresy sekcie a veľkosti sekcie môžeme zistiť, v ktorej sekcii sa daná štruktúra nachádza. Potom pomocou RAW adresy sekcie vypočítame RAW adresu danej štruktúry (RAW sekcie + (RVA štruktúry – RVA sekcie)). V *IMAGE_DATA_DIRECTORY* sa nachádzajú napríklad odkazy na tabuľku importov a exportov funkcií, ladiace informácie, resources a pod.

Jednotlivé sekcie, ktoré sa nachádzajú za hlavičkami a tabuľkou sekcií, obsahujú samotný binárny kód programu, resources, tabuľky importov a exportov funkcií a tak ďalej. Každá sekcia má názov (určený atribútom *Name*). Súbor nie je do sekcií delený kvôli logickému usporiadaniu (aj keď býva často pravidlom, že napríklad v sekcii s názvom „.code“ je binárny kód a pod.) ale kvôli vlastnostiam. Dáta v niektorej sekcii sú určené napríklad iba na čítanie, v inej aj na zápis.

Tabuľka importov je štruktúra, ktorá odkazuje na iné DLL knižnice, ktorých funkcie sa využívajú v aktuálnom EXE súbore. Na táto tabuľku odkazuje druhá položka z poľa *IMAGE_DATA_DIRECTORY* štruktúry. Samotná tabuľka sa skladá z poľa štruktúr *IMAGE_IMPORT_DESCRIPTOR*. Počet prvkov je zhodný s počtom DLL knižníc, z ktorých sa funkcie importujú a pole je zakončené jednou prázdnou štruktúrou. Každá položka tabuľky odkazuje na dva rôzne polia štruktúr *IMAGE_THUNK_DATA*, ktorých počet prvkov sa rovná počtu importovaných funkcií z konkrétnej DLL knižnice. Tieto zase odkazujú na jednu štruktúru *IMAGE_IMPORT_BY_NAME*, kde sú uložené názvy jednotlivých importovaných funkcií v položke *Name*. Polia *IMAGE_THUNK_DATA* sú pre každú DLL knižnicu v EXE súbore dva krát, pretože pri nahrávaní do pamäte je jedna z týchto tabuliek, nazývaná aj Import address table, je prepísaná PE loaderom na skutočné adresy importovaných funkcií. Na túto tabuľku odkazuje aj trinásta položka poľa štruktúr *IMAGE_DATA_DIRECTORY*.

Pre lepší prehľad v jednotlivých štruktúrach je priložený obrázok 1.2. Sú v ňom znázornené len popisované prvky (kvôli prehľadnosti nie presne všetky).



Obrázok 1.2: štruktúra PE EXE súboru (zjednodušene)

1.2 Počítačové vírusy

Počítačový vírus je program, ktorý dokáže infikovať ďalšie programy a ich modifikáciou je schopný zaistiť, aby obsahovali potenciálne sa vyvíjajúcu sa kópiu jeho samotného [9]. Počítačový vírus má teda blízko k svojmu biologickému originálu. Existujú aj iné formy škodlivého kódu, ako napríklad trójske kone (tvária sa ako užitočný program a pritom obsahujú iba škodlivý kód a nie sú schopne sa sami replikovať), spyware (program, ktorý bez vedomia užívateľa napríklad odosiela dôverné dáta do internetu). My sa však budeme zaoberať iba škodlivým kódom, ktorý je schopný infikovať už nejaký existujúci súbor a samostatne sa šíriť.

1.2.1 Metódy infekcie

Existuje mnoho typov vírusov, ktoré sú schopne infikovať rôzne typy súborov. Napádajú napríklad spustiteľné EXE súbory, dokumenty Microsoft Office (tzv. makro vírusy), boot sektory diskov. Nás z pohľadu analýzy súborov vo formáte PE EXE budú zaujímať tie vírusy, ktoré infikujú tieto súbory. Najjednoduchším spôsobom ako infikovať súbor je prepísať časť súboru, čím sa znehodnotí pôvodný obsah.

Ďalšou možnosťou je infekcia hlavičky. Vírus pridá svoje telo hneď za tabuľku sekcií pred začiatkom prvej sekcie a zmení adresu *AddressOfEntryPoint* v hlavičke. Telo vírusu musí byť však veľmi malé, pretože v priestore, kde sa nachádza zarovnanie veľa miesta nie je.

Iná možnosť je pripojiť sa na koniec súboru, ale nevytvárať novú sekciu. Vírus zmení položku *VirtualSize* a *SizeOfRawData* poslednej sekcie, *AddressOfEntryPoint* nastaví na vlastné telo a upraví informáciu o veľkosti súboru *SizeOfImage* [9].

Vírus môže tiež pri infekcii vytvoriť úplne novú sekciu. Upraví tabuľku sekcií, zmení hodnoty *NumberOfSections* v hlavičke a nastaví *AddressOfEntryPoint* na svoj začiatok.

Ďalšou možnosťou je pripojenie sa k existujúcej sekcii. Vírus upraví tabuľku sekcií a hodnoty *VirtualSize*, *SizeOfRawData* a *SizeOfImage* [3]. Čo sa týka zmeny adresy *AddressOfEntryPoint*, tak niektoré vírusy ju rovno upravia, iné na miesto prvej inštrukcie pôvodného programu dajú skok na svoje telo.

Vírusy využívajú pri infekcii tiež medzery v kvôli zarovnaniu medzi jednotlivými sekciami. Keďže však býva jedna medzera na uloženie tela vírusu málo, vírus rozdelí svoje telo do viacerých medzier medzi sekciami. Táto metóda má pre vírus výhodu v tom, že veľkosť súboru sa nezväčší, čo sťaží jeho spozorovanie [9].

1.2.2 Obrana vírusov pred odhalením

Jedna z najzákladnejších ochrán je zneprehľadnenie samotného kódu. Autor môže vložiť rôzne zavádzajúce inštrukcie, ktoré neovplyvňujú funkčnosť, alebo a zakódovať niektoré dáta. Iná možnosť

je komprimovať samotný binárny kód vírusu. Analýza a odhalenie takého kódu je náročnejšia a prináša to výhodu v tom, že telo vírusu sa zmenší. Vírus potom ešte okrem samotného skomprimovaného kódu obsahuje pakovač, ktorý pri vykonávaní kód rozbalí.

Ďalšia možnosť je zakódovanie tela vírusu. Podobne ako pri komprimácii obsahuje vírus na začiatku dešifrovaciu smyčku. Aby zakódovaný vírus nemal úplne konštantnú podobu, pridáva do kódovanie niekoľko premenných hodnôt. Novšie vírusy sa snažia využívať kódovanie s premennou hodnotou k tomu, aby bol každý exemplár z veľkej časti odlišný [3]. To síce komplikuje antivírusovým programom vyhľadávanie vírusov podľa konkrétneho vzorku, ale dekódovacia smyčka ostáva často podobná a je ju možno použiť k identifikácii vírusu.

Keďže bolo jednoduché zakódované vírusy nájsť podľa dekódovacej smyčky, začali sa vyskytovať vírusy, ktoré vedeli vytvárať rôzne mutované dekryptory [9]. Jedna z metód, ako dosiahnuť je napríklad použiť viacero dešifrovacích smyčiek namiesto jednej. Takéto vírusy sa nazývajú Oligomorfné.

Ďalší vývojový stupeň sú polymorfné vírusy, ktoré používajú komplikované algoritmy na to, aby bola dekódovacia smyčka v každom exempláre iná. To dosahujú napríklad vkladaním rôznych inštrukcií do kódu neovplyvňujúcich samotný algoritmus, vkladaním náhodných bajtov dát medzi bloky binárneho kódu alebo proste obmenou kódu využívaním rôznych inštrukcií. Samotné zakódované alebo k tomu ešte komprimované telo vírusu však ostáva stále rovnaké.

Existujú aj vírusy, ktoré dokážu vytvárať svoje rozdielne kópie tela bez použitia šifrovania alebo komprimovania. Takéto vírusy sa nazývajú metamorfné. Jedna z metód je napríklad rozdelenie kódu vírusu do viacerých podprogramov, ktoré sú vždy inak usporiadané.

1.3 Štatistická analýza PE EXE súboru pomocou mapy entropie

Jednotlivé sekcie a štruktúry v PE EXE súbore majú rôznu mieru usporiadanosti, teda entropiu. Napríklad programový kód má entropiu pomerne vysokú, pravidelne usporiadané štruktúry ako napríklad tabuľka importov funkcií alebo nejaký bitmapový obrázok v resources má entropiu veľmi nízku. Táto vlastnosť sa dá využiť vzhľadom na detekciu vírusov.

1.3.1 Definícia pojmu entropie

Z pohľadu matematiky je entropia v štatistike stredná hodnota informácie. Dá sa tiež povedať, že je to miera neurčitosti výskytov javov s pravdepodobnosťou p. Ak máme n rôznych javov, kde sa každý vyskytuje s pravdepodobnosťou p_i, potom sa entropia H náhodného procesu vypočíta pomocou vzorca:

$$H = -\sum_{i=1}^{n} p_i * \log_2(p_i)$$

Entropia je spojitá funkcia na intervale $0 \leq P \leq 1$.Ak sa entropia javu rovná nule, jav je jednoznačne určený (podrobnejšie viď [6]). Keď je entropia blízko nule, tak sa pravdepodobnosť javu blíži k hodnote jedna alebo nula. Entropia je maximálna vtedy, keď sú pravdepodobnosti jednotlivých javov rovnomerne rozložené. Vo fyzike sa entropia skúma napríklad v termodynamike, kde je to veličina, ktorá skúma náhodnosť a neusporiadanosť systému.

1.3.2 Využitie entropie EXE súboru

Skúmanie entropie PE EXE súboru má viaceré možnosti využitia. Napríklad ju môžeme využiť na detekciu polymorfných vírusov. Tieto vírusy sa síce v binárnej podobe od seba kus od kusu líšia, ale keď sa pozrieme na rozloženie entropie ich binárnych dát, tak často nájdeme veľmi podobné až zhodné vzorky.

Ďalej môžeme využiť rozloženie entropie na rýchlejšiu analýzu súboru. Keďže binárny kód má väčšiu entropiu ako časti súboru, ktoré nás z hľadiska vyhľadávania vírusov nezaujímajú, môžeme sa sústrediť len na miesta s vysokou entropiou.

Pozorovanie rozloženia entropie v súbore môže byť užitočné tiež pri odhaľovaní ešte neznámych komprimovaných alebo kryptovaných vírusov, keďže komprimované alebo kryptované dáta majú veľmi vysokú entropiu aj oproti normálnemu programovému binárnemu kódu.

1.3.3 Vytváranie mapy entropie

Na štatistickú analýzu pomocou entropie sa vytvára takzvaná mapa entropie. Pri tom rozdelíme súbor na šestnásťbajtové úseky a každému priradíme hodnotu entropie od 0 po 15. Čím nižšia hodnota, tým je aj nižšia entropia. Tieto hodnoty sú zoradené za sebou. Entropiu zobrazujeme v hexadecimálnej reprezentácii (teda od 0 po F), pričom nulovú hodnotu značíme kvôli prehľadnosti ako pomlčku (-). Mapa entropie sa teda skladá z poľa číslic v rozsahu od 0 po 15. Súbor s veľkosťou 1600 kilobajtov je napríklad popísaný reťazcom o dĺžke 100 kilobajtov.

Samotný algoritmus výpočtu entropie pre jednu šestnásťbajtovú časť súboru spočíva v tom, že ju zaradíme do skúmaného úseku vrátane jej okolia (do určitej vzdialenosti), pretože pri skúmaní usporiadanosti nás nezaujíma len samostatná časť ale aj jej susedia. Nastavíme hodnotu entropie na maximum (0xF) a prechádzame tento úsek najprv po dvoch bajtoch, potom po štyroch, a tak ďalej. Ak nájdeme nejakú pravidelnosť (napríklad hodnota sa po každé dva bajty zvyšuje o dva), znížime hodnotu entropie o jedna. Pri určovaní entropie úseku sa teda jej hodnota znižuje tým viac, čím viac sa v ňom vyskytuje pravidelností. Ak nájdeme veľmi málo pravidelných hodnôt, entropia sa skoro vôbec nezníži a ostane blízko maximálnej hodnoty (0xF). Keď sa vyskytuje v úseku veľa pravidelných hodnôt, entropia sa hocikedy zníži až na nulu, čo je minimum. Výsledkom je hodnota entropie prislúchajúca tejto šesnásťbajtovej časti závislá na okolí do určitej vzdialenosti.

Špeciálne prípady nastávajú, keď napríklad sa v súbore nachádza text. Z hľadiska analýzy vírusov nie je pre nás tento úsek zaujímavý a jeho entropia je z tohto pohľadu nízka. Preto ak nájdeme v súbore úsek čitateľných znakov dlhší ako určitý limit, znížime entropiu rovno na nulu.

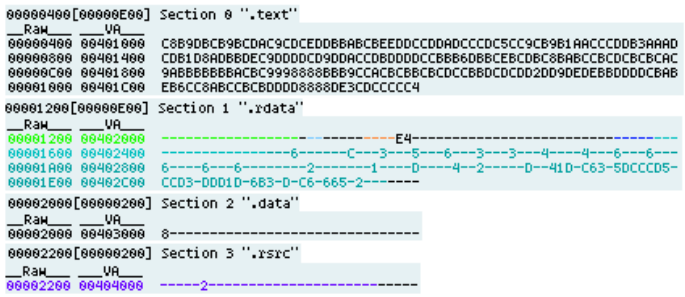
Treba zdôrazniť, že táto entropia sa odlišuje od tej matematickej. Kvôli jednoduchšej a efektívnejšej práci s jej hodnotami na počítačových systémoch reprezentujeme entropiu celými prirodzenými číslami v rozsahu od nuly po pätnásť. Tak isto to nie je striktne entropia, ktorá reprezentuje presne usporiadanosť alebo neusporiadanosť binárnych hodnôt v súbore (viď napríklad spomínaný prípad, keď sa v súbore nachádza veľa textu).

Mapy entropie väčšinou vytvárame samostatne pre každú sekciu PE EXE súboru, keďže ich obsah je z hľadiska vyhľadania vírusov touto metódou zaujímavý a tiež kvôli prehľadnosti.

1.3.4 Analýza mapy entropie

Keď máme vytvorené mapy entropie pre každú sekciu v PE EXE súbore, dĺžka každej mapy sa rovná veľkosti sekcie vydelenej 16. Prvá sekcia má často názov *.text* alebo *.CODE* a zvyčajne obsahuje samotný binárny kód programu. Hodnota entropie binárneho kódu sa pohybuje zhruba od 0x9 po 0xD. Ďalšie bežné sekcie sú napríklad *.data*, kde sa ukladajú inicializované dáta, *.bss*, ktorá obsahuje statické neinicializované globálne premenné a sekcia *.rsc*, ktorá obsahuje zdroje (resources) súboru, kde sa nachádzajú napríklad bitmapové ikony programu a pod. Názvy sekcií nie sú pevne dané, programátor si ich môže ľubovoľne zmeniť. Keďže sa v týchto sekciách nachádzajú pravidelné

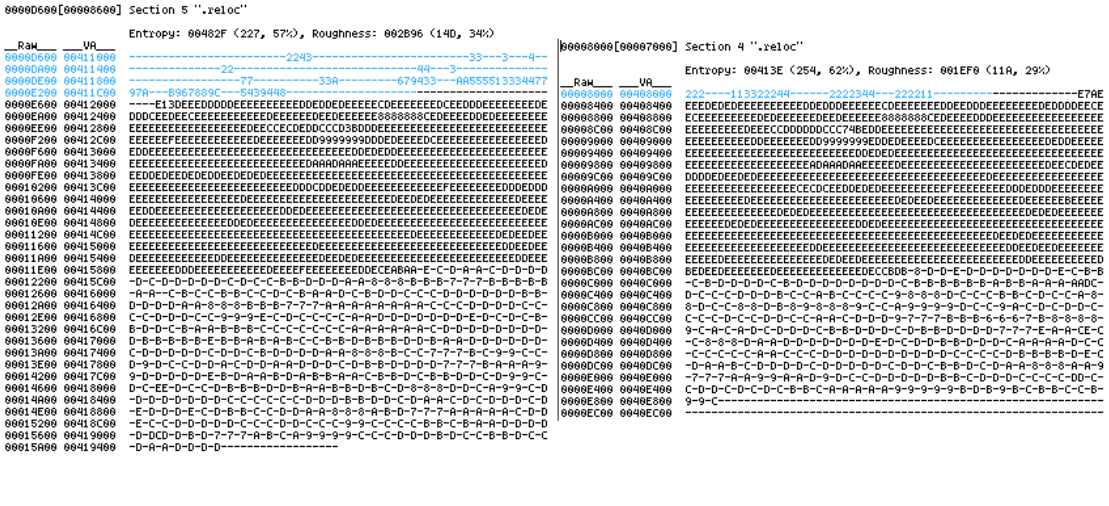
dátové štruktúry, majú tieto sekcie v porovnaní s programovým kódom veľmi nízku entropiu, často je to nula. Na obrázku 1.3 je znázornené typické rozloženie entropie súboru. Pre názornosť je zobrazená mapa entropie malého 9kB testovacieho PE EXE súboru konzolovej aplikácie.



Obrázok 1.3: mapa entropie

Keď je súbor napadnutý napríklad polymorfným vírusom so zašifrovaným alebo skomprimovaným telom, je jeho entropia napadnutej časti veľmi vysoká. Keď si tento vírus zvolí takú metódu infekcie, že vytvorí novú sekciu alebo sa pripojí do existujúcej, je v mape entropie veľmi pekne viditeľný. Do mapy napríklad pribudne nová sekcia s veľmi vysokou entropiou alebo v niektorej sekcii na mieste, kde bývajú normálne dáta s nízkou entropiou vidíme časť s vysokou hodnotou entropie.

Ako príklad je pripojený obrázok 1.4 s dvoma vzorkami rôznych PE EXE súborov napadnutých polymorfným vírusom W32/Stepan. Vírus, ktorý má vysokú entropiu sa nachádza v sekcii *.reloc*, ktorá normálne obsahuje tabuľku bázových relokácií, čo je pomerne usporiadaná štruktúra s nízkou entropiou. Tiež vidno podobnosť entropie jednotlivých vzoriek vírusu.



Obrázok 1.4: mapy entropie dvoch rôznych súborov napadnutých tým istým vírusom

Takéto informácie z mapy entropie sú veľmi vhodné na plošnú grafickú reprezentáciu, kde by spomínané vlastnosti boli pekne viditeľné.

1.4 Metódy vizualizácie dát

2D počítačová grafika sa z pohľadu programátora delí na vektorovú a rastrovú. V rastrovej grafike sa obrazová informácia popisuje maticou bodov (pixelov) [4]. Vo vektorovej grafike je obraz popísaný analyticky pomocou jednoduchých vektorových entít, ako sú úsečky, kružnice, elipsy a pod. Rastrová grafika je vhodná napríklad na uloženie fotografií, ktoré aj získavame ako maticu bodov. Pre našu vizualizáciu entropie bude však najvhodnejšia vektorová grafika, keďže budeme obraz skladať podľa dát mapy entropie z jednoduchých geometrických útvarov.

Obraz vo vektorovej grafike je popísaný v dvojrozmernom priestore týmito základnými entitami: úsečka, kružnica, krivka a polygón. Jednotlivé prvky v obraze a ich poloha sú teda presne matematicky popísané. Napríklad pri úsečke máme definované súradnice začiatočného a koncového bodu, pri kružnici máme definovaný stred a polomer. Pri zobrazovaní sa vektorový popis prevádza na rastrové zobrazenie. V tom tkvejú viaceré výhody: pri zmene veľkosti (hlavne zväčšení) neprichádzame o kvalitu a obraz môžeme ľubovoľne transformovať tiež bez straty kvality. Keď ukladáme obraz do vektorového formátu, ukladá sa len matematický popis entít, čo znamená, že aj pri veľkom rozlíšení je tento formát úsporný v porovnaní s rastrovými formátmi.

2 Návrh grafickej reprezenácie mapy entropie

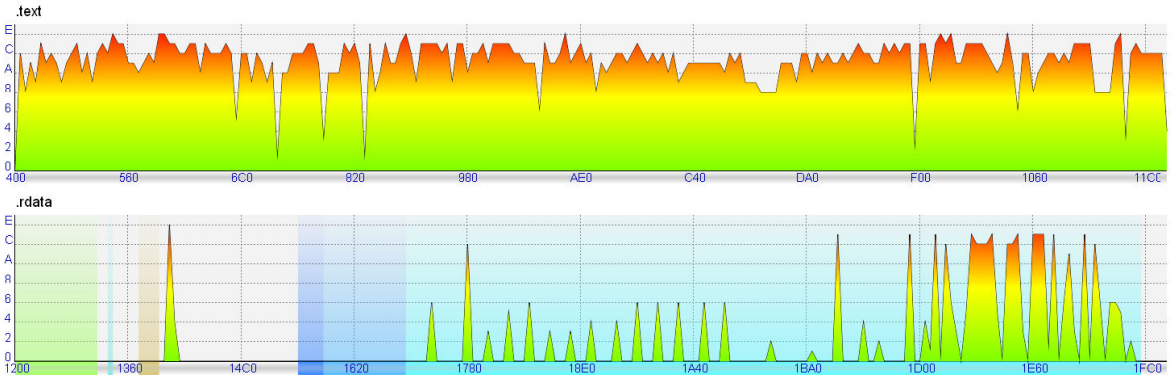
Keďže mapa entropie súboru je informácia, ktorá obsahuje dva rozmery – dĺžku súboru a samotnú hodnotu entropie v každom bode, dá sa vhodne previesť do plošnej obrazovej reprezentácie. Preto budeme entropiu zobrazovať v dvojrozmernom priestore.

2.1 Navrhnuté typy grafov

Vytvorili sme tri návrhy grafickej reprezentácie mapy entropie: X–Y graf, hviezdicový graf a kruhový graf. Každý sa hodí na iné použitie, niektoré sú vďaka jednoduchosti a prehľadnosti vhodné na skúmanie mapy entropie, iné sú vytvorené dizajnovo skôr na mediálnu prezentáciu tejto metódy.

2.1.1 X–Y graf

Najjednoduchší a pri tom prehľadný spôsob grafickej reprezentácie je jednoducho zobraziť mapu entropie ako funkciu, kde na os x reprezentuje poradie jednotlivých skupín po 16 bajtov a os y nanesieme im prislúchajúcu entropiu. Vhodné je doplniť tento typ grafu o číslovanie a mriežku, kde na y osi zvýraznime jednotlivé hodnoty entropie a na x osi nanesieme v určitých intervaloch prislúchajúcu RAW adresu súboru. Pre lepši prehľad usporiadame mapy entropie jednotlivých sekcií pod seba a nad každú doplníme názov sekcie. Navyše v jednotlivých sekciách farebne zvýrazňujeme štruktúry odkazované z *IMAGE_DATA_DIRECTORY*, ako napríklad tabuľku importov. Na obrázku 2.1 je ukážka prvých dvoch sekcií PE EXE súboru, prvá obsahuje binárny kód, druhá statické dáta a rôzne štruktúry z *IMAGE_DATA_DIRECTORY* vrátane napríklad tabuľky importov a exportov.

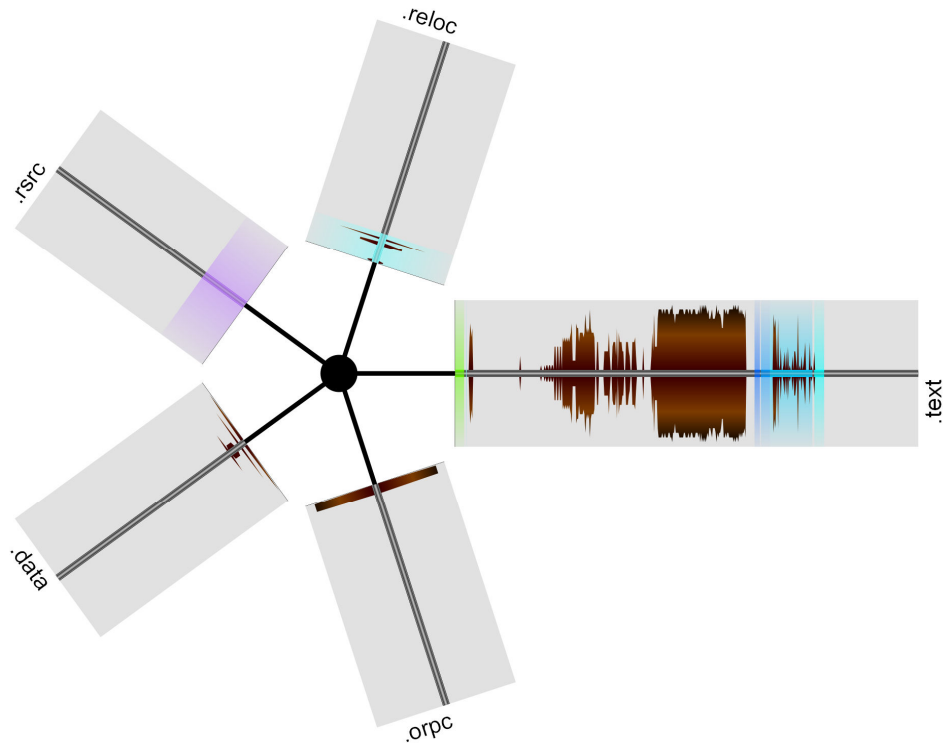


Obrázok 2.1: X–Y graf

Predošlý graf ma nevýhodu v tom, že napriek svojej prehľadnosti nie je veľmi vhodný na mediálnu prezentáciu. U veľmi veľkých súborov sa jednoducho dlhá sekcia nezmestí na monitor, preto potrebujeme graf, ktorého celkové rozmery by sa vždy blížili štvorcu.

2.1.2 Hviezdicový graf

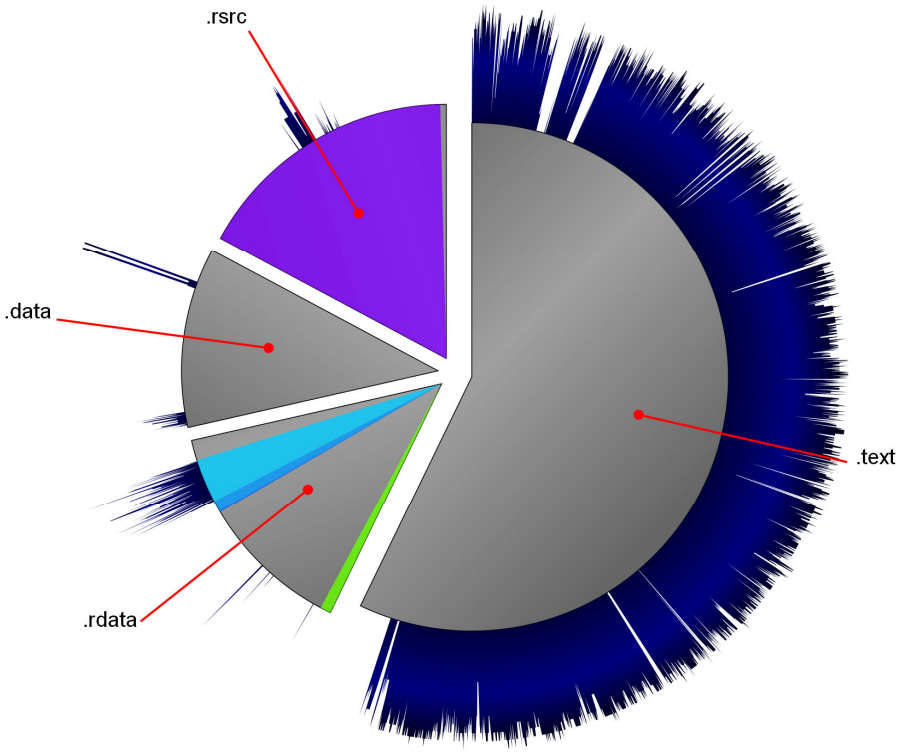
Ďalší návrh je takzvaný hviezdicový graf (viď obrázok 2.2). Je založený na predošlom návrhu, s tým, že grafy sekcií hviezdicovito usporiadame. Problém tohto grafu je, že jednotlivé sekcie nebývajú rovnako dlhé. Často prvá sekcia s kódom zaberá dĺžku skoro celého súboru a ostatné sekcie s dátami sú v porovnaní s ňou veľmi krátke, čo tento graf deformuje a z hviezdice často ostane len jadro a jedná dlhá vyčnievajúca sekcia.



Obrázok 2.2: Hviezdicový graf

2.1.3 Kruhový graf

Tretí variant je takzvaný kruhový graf (viď obrázok 2.3). Jednotlivé kružnicové výseky zobrazujú sekcie, celá kružnica zobrazuje súbor. Veľkosti kružnicových výsekov zodpovedajú dĺžkam sekcií. Obvod kruhu reprezentuje osu x z predošlých grafov, osa y znázorňujúca veľkosť entropie je vždy kolmá na dotyčnicu kružnice v danom mieste. V centrálnej časti kruhu farebne vyznačujeme jednotlivé štruktúry odkazované z IMAGE_DATA_DIRECTORY. Tento graf je na mediálnu prezentáciu najvhodnejší, keďže jeho vzhľad zostáva stále súmerný bez ohľadu na počet a dĺžku sekcií. Do grafu sú doplnené popisky zobrazujúce názvy sekcií.

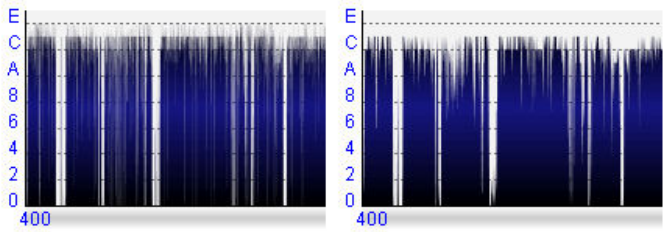


Obrázok 2.3: Kruhový graf

2.2 Optimalizácia vykresľovania

Grafy vykresľujeme pomocou rasterizácie vektorov. Každá hodnota entropie v mape má v grafe svoju vektorovú reprezentáciu vo forme úsečky. Pri súboroch s veľkosťou 1MB a viac je mapa entropie veľmi veľká. To spôsobuje pomalé vykresľovanie, keďže sa musí vykresliť osobitne každá úsečka znázorňujúca entropiu na danom mieste. Jednotlivé hodnoty sú v takom grafe tiež veľmi nahusto, keďže rozlíšenie monitora nestačí na veľký počet zobrazených hodnôt.

Preto je potrebné navrhnúť optimalizáciu vykresľovania. Pri veľkých súboroch sa mapa entropie pred vykreslením musí skrátiť a spriemerovať. Napríklad pre každé tri hodnoty entropie spravíme ich aritmetický priemer a budeme ich reprezentovať len jednou úsečkou. Ukážka tato optimalizovaného grafu je na obrázku 2.4. Keďže sa spriemerovaním kúsok skresľuje informácia, ako to vidno na obrázku, program umožňuje nastaviť, v akej miere sa má mapa entropie pri zobrazovaní spriemerovať alebo túto optimalizáciu úplne vypnúť.



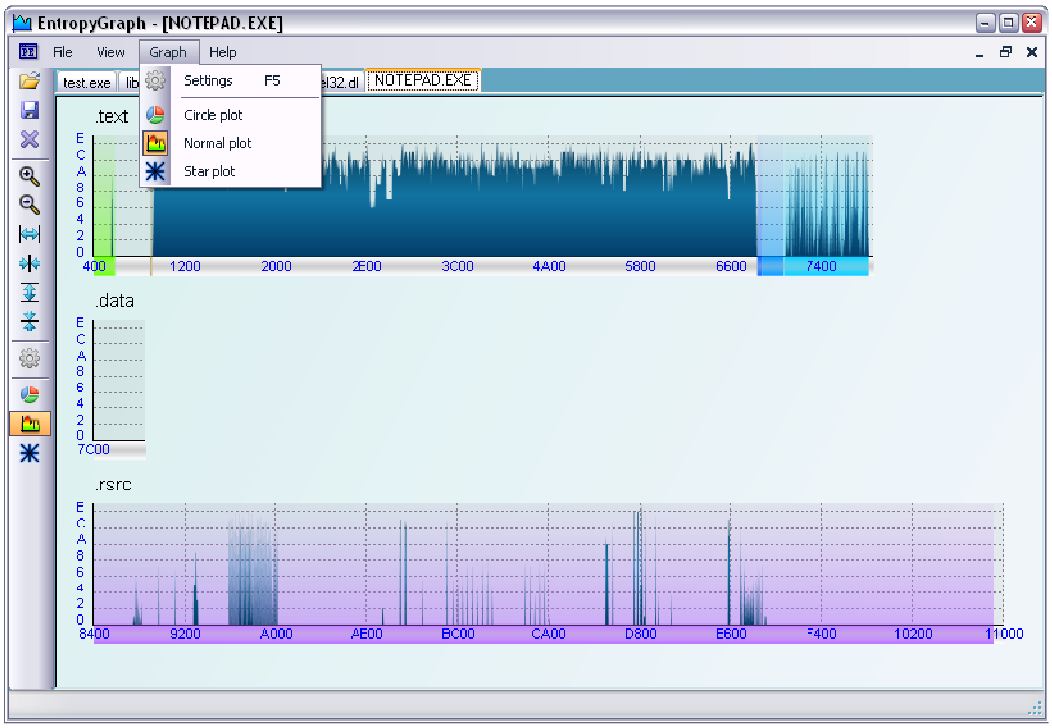
Obrázok 2.4: Vľavo pôvodná mapa entropie, vpravo spriemerované zobrazenie.

3 Návrh programu

Základná funkcionalita programu na vizualizáciu mapy entropie je načítanie PE EXE súborov a zobrazenie jednotlivých grafov. Program musí umožňovať načítať viacero súborov naraz, aby sa mohli napríklad porovnať napadnutý súbor s originálom. Farby a proporcie jednotlivých grafov by mali byť nastaviteľné. Program by mal tiež umožňovať deformáciu, priblíženie alebo oddialenie grafu podľa potreby. Keďže sa budú výstupy z programu používať na mediálnu prezentáciu, mal by umožňovať export grafov do obrazových grafických formátov. Program má byť implementovaný v jazyku C++.

3.1 Grafické užívateľské rozhranie

Keďže musí program umožňovať načítať viacero súborov naraz, riešime grafické užívateľské rozhranie (ďalej len GUI) ako MDI (multiple document interface), kde GUI pozostáva z jedného rodičovského okna v ktorom sa nachádzajú dcérske okna zobrazujúce grafy jednotlivých načítaných súborov. Jedna z typických nevýhod MDI rozhrania je, že nemáme informáciu a všetkých otvorených dcérskych oknách a užívateľ medzi nimi musí prepínať napríklad pomocou menu, čo nie je veľmi efektívne. To opravíme tak, že pridáme takzvané záložky (tab control), pomocou ktorých užívateľ môže jednoducho a rýchlo prepínať medzi dcérskymi oknami. Okrem štandardných ovládacích prvkov ako je menu, pridáme jeden panel nástrojov, na ktorom sú najpoužívanejšie funkcie rýchlo prístupné.



Obrázok 3.1: Návrh GUI

3.2 Objektový návrh programu

Pri návrhu programu sme využívali objektový prístup. Objektovo orientované programovanie (OOP) je spôsobom abstrakcie, kedy algoritmus implementujeme pomocou množiny zapuzdrených vzájomne komunikujúcich entít, z ktorých každá má plnú výpočtovú mocnosť celého počítača [5]. Základným prvkom je objekt, ktorý si pamätá svoj stav pomocou množiny atribútov a poskytuje rozhranie operácií, aby s ním bolo možné pracovať [7]. Pri práci s objektom nás zaujíma len poskytované rozhranie, a nie vnútorná implementácia, čo je princíp zapúzdrenia. Trieda je prepis, ako sa dá vytvoriť objekt daného typu.

Pri návrhu programu sme používali viacero techník OOP, ako je napríklad dedičnosť a objektová skladba. Využívame tiež návrhové vzory factory a singleton. Na obrázku 3.2 je navrhnutý diagram tried, v ktorom sú zobrazené všetky v nasledujúcom texte popisované triedy. Pre jednoduchšie a prehľadnejšie zobrazenie neobsahuje presne všetky triedy a ich metódy tak, ako to je v skutočnosti implementované, ale vyjadruje základnú štruktúru návrhu.

3.2.1 Návrh GUI tried

Keďže grafické užívateľské rozhranie pozostáva z hlavného okna a dcérskych okien zobrazujúcich grafy jednotlivých načítaných PE EXE súborov, odráža sa to aj na návrhu tried implementujúcich GUI. Hlavné okno triedy *GuiMainWindow* bude obsahovať ľubovoľný počet dcérskych okien triedy *ChildWindow*. Ich súčasťou bude vždy jedna inštancia triedy *SettingsWindow* implementujúcej okno, pomocou ktorého môže užívateľ meniť nastavenia grafov.

3.2.2 Návrh grafických komponentov

Pri návrhu tried implementujúcich jednotlivé grafy sme využili princíp dedičnosti. Všetky typy grafov majú jednu spoločnú rodičovskú abstraktnú triedu *Plot*. Od nej sú odvodené jednotlivé triedy implementujúce vykresľovanie grafov (*NormalPlot*, *CirclePlot*, atď.). Tento prístup má výhodu v tom, že pri používaní týchto tried majú všetky rovnaké rozhranie definované v rodičovskej triede, takže pri programovaní volania metód nemusíme riešiť, s akým konkrétnym typom grafu sa za behu bude pracovať.

Často majú rôzne typy grafov spoločné prvky. Napríklad hviezdicový graf môžme vytvoriť tak, že vykreslíme jednotlivé sekcie ako X-Y graf a tie transformujeme a usporiadame do hviezdice. Alebo napríklad popisky vykresľujeme u všetkých grafov rovnakým spôsobom, takže by bolo zbytočné implementovať takéto generické prvky pre každý graf samostatne. Preto využívame na to princíp objektovej skladby. Všetky typy grafov sú poskladané z generických komponent (*EPNormalGrid*, *EPLinePlot*, atď.), ktoré sa môžu ľubovoľne používať. Tieto komponenty majú

podobne ako grafy spoločnú rodičovskú abstraktnú triedu *PlotComponent*, pomocou ktorej aj zdieľajú niektoré metódy, ktorých implementácia je pre všetky odvodené triedy rovnaká.

3.2.3 Využitie návrhových vzorov

Návrhový vzor systematicky nazýva, vysvetľuje a vyhodnocuje dôležitý a v objektovo orientovaných systémoch opakujúci sa návrh [2]. Sú to popisy komunikujúcich objektov a tried, ktoré sú upravené na riešenie obecného návrhového problému.

Použitie návrhového vzoru továreň

Pri našom návrhu tried jednotlivých grafov narážame na jeden problém. Polymorfizmus nám síce umožňuje pracovať rovnako so všetkými typmi grafov, keďže sú odvodené od spoločnej rodičovskej abstraktnej triedy *Plot*, ale keď vytvárame konkrétnu inštanciu týchto tried, musíme predsa len špecifikovať v kóde presne odvodenú triedu, ktorú vytvárame (napríklad či sa jedná presne o *CirclePlot*, alebo *StarPlot*). My by sme však potrebovali aj tu nejaký virtuálny konštruktor, ktorý by nám vytvoril konkrétnu inštanciu dcérskej triedy zdedenej z *Plot* až za behu podľa toho, aký typ grafu sa má zobraziť. Samozrejme mohli by sme do kódu vložiť napríklad podmienku (napríklad príkaz *switch*), od ktorej by sme dali všetky typy grafov a na základe zvoleného typu by sa vytvoril konkrétny graf. Táto konštrukcia si ale vyžaduje úpravu kódu vždy, keď by sme napríklad pridali nový graf. Preto potrebujeme nejaký objekt, do ktorého by sme toto vytváranie presunuli a náš kód využívajúci jednotlivé grafy by nepotreboval úpravu v prípade pridávania alebo odoberania typu grafu.

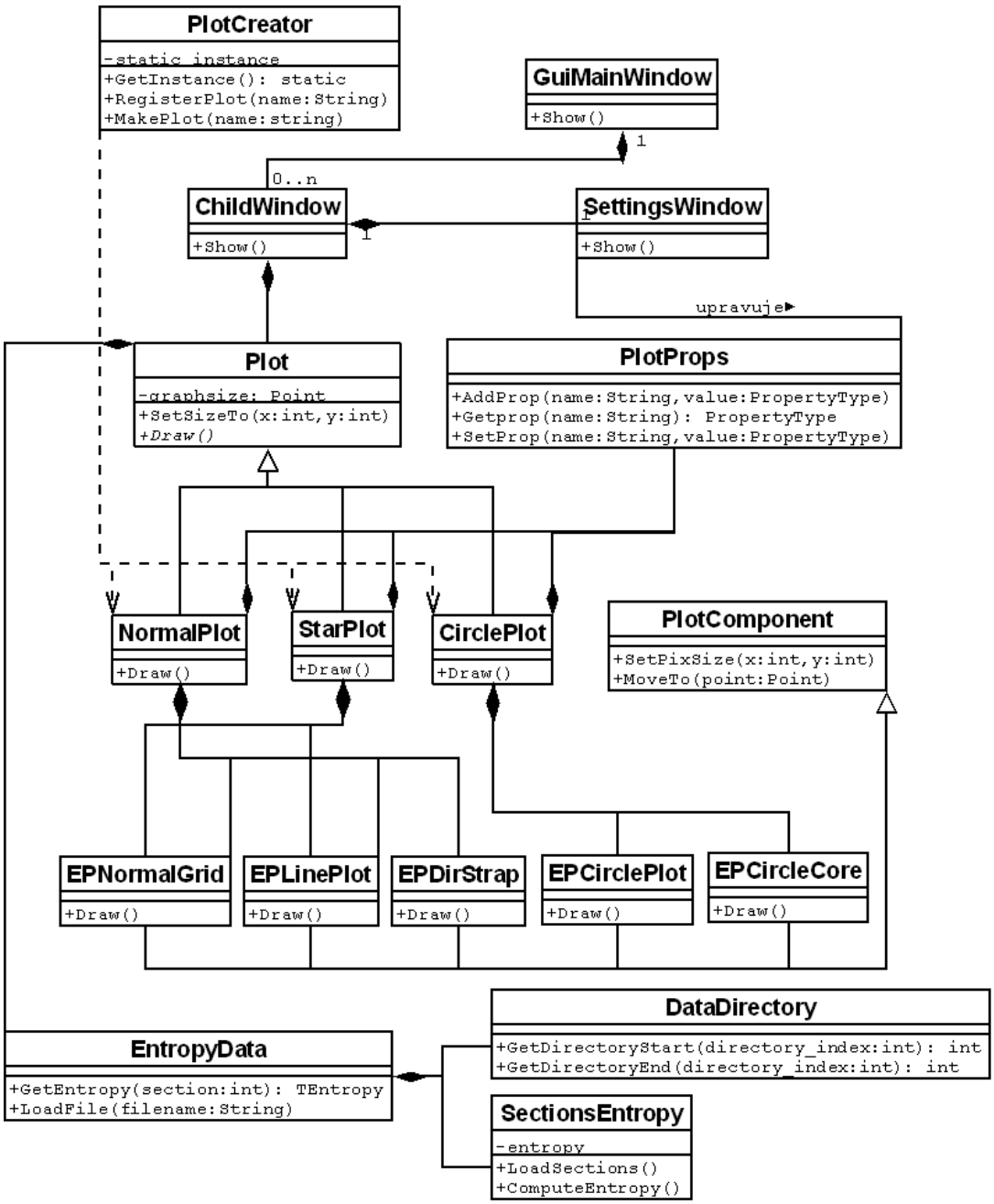
Náš problém rieši návrhový vzor továreň (factory pattern), kde jedna trieda *PlotCreator* definuje rozhranie pre vytváranie objektu [2]. My využijeme takzvanú parametrizovanú továreň, kde metóda *MakePlot(name: String)* triedy *PlotCreator* implementujúcej tento návrhový vzor bude mať jeden parameter identifikujúci konkrétny graf (viď [1]). V našom prípade je každý graf identifikovaný jedinečným reťazcom. Pri vytváraní inštancie dcérskej triedy konkrétneho grafu predáme továrni len tento reťazec a ona už bude vedieť, aký konštruktor má použiť.

Použitie návrhového vzoru singleton

Naša továreň na grafy by mal byť globálne prístupný objekt. Napríklad k jej metóde *MakePlot(name: String)* budeme pristupovať v implementácii triedy *ChildWindow*, ktorá bude v GUI vytvárať a zobrazovať jednotlivé grafy. Na inom mieste (ešte pred zobrazením prvého grafu) bude zas potrebné jednotlivé typy grafov u továrne registrovať, aby továreň vedela, ktorý identifikačný reťazec patrí ktorému grafu. Preto potrebujeme zaistiť, že inštancia triedy továrne bude v programe len jedna.

To zaistíme, keď triedu továrne *PlotCreator* implementujeme pomocou návrhového vzoru singleton (jedináčik). Ten zaistí, že trieda bude mať len jednu inštanciu a poskytne k nej globálny prístupový bod [2]. Trieda singletonu je riešená tak, že k nej pristupujeme pomocou statickej metódy

GetInstance(), ktorá nám vráti konkrétnu inštanciu triedy *PlotCreator*. Pri prvom prístupe k nej sa inštancia automatický vytvorí.



Obrázok 3.2: zjednodušený diagram tried

3.2.4 Návrh správy nastavení vlastností grafov

Jedna z požiadaviek na funkcie programu bola, že u grafov by sa mali dať meniť rôzne parametre zobrazenia, ako napríklad farba jednotlivých komponentov, šírka čiar a pod. Pôvodný návrh počítal s tým, že nastavenia grafov budú realizované pomocou metód tried implementujúcich grafiku. Tu sme narazili na jeden problém. Keďže každý graf má iné vlastnosti a nastavenia, bola by potreba vytvoriť

samostatné metódy pre každú dcérsku triedu triedy *Pot*. To by nám narušovalo návrh vzhľadom na to, že chceme, aby kód obsluhujúci vytváranie a narábanie s grafmi používal len metódy spoločnej rodičovskej triedy. Tým by sa porušila možnosť jednoducho pridávať a odoberať nové grafy.

Preto je potrebné navrhnúť samostatnú triedu *PlotProps*, ktorá bude spravovať nastavenia grafov. V tejto triede sa nachádzajú všetky nastavenia pre konkrétny typ grafu. Každá inštancia triedy grafu si pri svojej inicializácii vytvorí objekt nastavení a pridá do neho všetky pre graf špecifické nastavenia a ich implicitné hodnoty. Implementácia grafu bude potom k týmto nastaveniam pristupovať a na základe nich vykresľovať graf. Objekt GUI triedy *SettingsWindow* bude môcť k týmto nastaveniam pristupovať cez spoločné rozhranie, dynamicky zistiť ich počet a vlastnosti a umožniť užívateľovi napríklad niektoré nastavenia zmeniť.

3.2.5 Návrh dátovej reprezentácie mapy entropie

Mapu entropie bude ukladať inštancia triedy *EntropyData*, ku ktorej budú mať prístup všetky grafické objekty a na základe dát poskytovaných touto triedou budú zobrazovať grafiku. Trieda *EntropyData* sa bude skladať z objektu triedy *SectionsEntropy* implementujúcej výpočet entropie pre jednotlivé sekcie a z objektu triedy *DataDirecory*, ktorý zistí z EXE súboru polohy štruktúr odkazovaných z *IMAGE_DATA_DERECTORY* kvôli ich zvýrazneniu v grafoch.

4 Implementácia a testovanie

Program je implementovaný v jazyku C++ na platforme Microsoft Windows pomocou vývojového nástroja Visual Studio 2005. Pri implementácii dátových štruktúr a algoritmov využívame často štandardnú C++ knižnicu STL.

4.1 Načítanie súboru a výpočet entropie

Pre výpočet entropie používame špeciálnu knižnicu implementovanú v C++ vytvorenú vo firme Grisoft. Tej predáme ako parameter pole bajtov reprezentujúce binárne dáta súboru a ako návratovú hodnotu dostaneme pole bajtov obsahujúce mapu entropie.

Informácie z PE EXE súboru získavame načítaním kompletného súboru do pamäte, kde je reprezentovaný ako pole bajtov a prechádzaním jednotlivých štruktúr obsahujúcich pre nás dôležité informácie. Tie sú nadefinované v hlavičkových súboroch windows.h a winnt.h, ktoré sú štandardnou súčasťou vývojových nástrojov na platforme Windows.

Informácie o sekciách získame z hlavičky PE EXE súboru, kde nás zaujíma hlavne položka *NumberOfSections*. Z tabuľky sekcií získame podrobnejšie informácie o každej sekcii ako je jej začiatok, veľkosť a pod. Pomocou týchto informácií vyselektujeme jednotlivé úseky, ktoré patria sekciám.

Keďže mapu entropie počítame osobitne pre každú sekciu, implementujeme ukladanie informácií o sekciách ako pole (STL *Vector*) štruktúr *tExeSection*. Tá obsahuje informácie o každej sekcii, ako jej názov, veľkosť, index začiatku a pod. Obsahuje tiež samotnú mapu entropie prislúchajúcu danej sekcii, ktorá je implementovaná pomocou triedy *Vector* knižnice STL ako pole typu *unsigned char*. Tieto informácie sa potom využívajú pri zobrazovaní grafov. Informácie o sekciách a výpočet entropie zapuzdruje objekt triedy *SectionsEntropy*.

Ukladanie informácií o štruktúrach odkazovaných z *IMAGE_DATA_DIRECTORY* implementujeme podobne ako u sekcií poľom (STL *Vector*) štruktúr obsahujúcich informácie, ako začiatok štruktúry, sekciu, v ktorej sa nachádza, relatívny začiatok v rámci sekcie (potrebný pre zobrazovanie v grafoch), veľkosť, atď. Mnohé odkazy sa nachádzajú často vo forme RVA adresy, nás však zaujíma RAW adresa, aby sme vedeli, kde v súbore hľadať. Túto adresu prepočítavame pomocou postupu popísaného v kapitole 1.1. Informácie o *IMAGE_DATA_DIRECTORY* štruktúrach zapuzdruje objekt triedy *DataDirecrory*.

Všetky tieto objekty sú súčasťou objektu triedy *EntropyData*. Tá zapuzdruje celé načítanie súboru a získavanie jednotlivých informácií, ktoré pomocou metód sprístupňuje objektom tried implementujúcich grafiku.

4.2 Implementácia grafiky

4.2.1 Knižnica Cairo

Na samotnú vektorovú grafiku sme použili 2D knižnicu Cairo. Táto knižnica je implementovaná v jazyku C. Je prenositeľná na viaceré platformy, medzi ne patrí Windows aj Linux. Triedy implementujúce grafiku využívajúce túto knižnicu vrátane už popisovaných tried na načítanie súboru sú teda programované tak, že by sa dali použiť aj ne inej platforme, ako Windows. Knižnica Cairo vie vykresľovať priamo na Device Context (pomocou ktorého sa vykresľuje na okno vo Windows Forms), do pamäte, alebo priamo do vektorových grafických formátov, ako je SVG alebo PDF. Tiež podporuje export vykreslenej a rasterizovanej plochy do obrázkov vo formáte PNG.

Princíp kreslenia knižnicou Cairo pozostáva z piatich základných prvkov: cieľ, zdroj, maska, cesta a kontext (podrobnejšie viď [10]). Cieľ je plocha, na ktorú kreslíme. Združuje jednotlivé vykresľované entity. Zdroj je takzvaný „štetec“ ktorým vyplňujeme alebo vytiahneme vektorové entity. Môže to byť napríklad gradient, jednoliata farba, alebo bitmapový vzor. Maska kontroluje, aká časť zdroja sa aplikuje pri vykresľovaní na cieľ. Cesta je skupina vektorových entít, ktoré môžeme transformovať, vykresľovať a pod. Kontext udržuje stav spomínaných prvkov (na aký cieľ sa kreslí, akým zdrojom kreslíme a akú masku aplikujeme).

Grafy pozostávajú z vektorových entít, ktoré sú vykresľované knižnicou Cairo. Sú to napríklad úsečka, krivka, kružnica. Ich poloha je definovaná užívateľskými súradnicami, kde bod so súradnicami 0, 0, je ľavý horný roh obdĺžnikovej plochy, na ktorú kreslíme. Tieto entity potom spojíme do cesty , ktorú vyplníme alebo vytiahneme nastaveným štetcom (zdrojom).

Knižnica Cairo nám umožňuje obraz ľubovoľne transformovať pomocou lineárnych geometrických transformácií pomocou transformačnej matice. Obraz môžeme napríklad rotovať, skosiť, posunúť alebo rôzne meniť pomer strán a veľkosť. Pri práci s 2D grafikou máme vlastne dva priestory súradníc. Jeden sú už spomínané užívateľské súradnice (user space coordinates) a druhý súradnice cieľového zariadenia, na ktoré vykresľujeme (device space coordinates). Transformácia určuje prevod medzi týmito priestormi. To znamená, že užívateľské súradnice sa transformáciou nezmenia a keď chceme nejaký obraz vykreslený na cieľové zariadenie transformovať, treba najprv spraviť transformáciu a až potom ukladať a rasterizovať vektorové entity pomocou užívateľských súradníc.

4.2.2 Vytváranie grafických komponentov

GUI triedy ovládajúce zobrazenie grafov vytvárajú konkrétne inštancie grafov pomocou metódy *MakePlot(string name)* triedy továrne *PlotCreator*, ktorej dajú ako parameter jedinečný názov grafu. Aby továreň vedela, aké grafy môže vytvárať, musí sa na začiatku pri spustení programu každý typ

grafu u objektu triedy *PlotCreator* zaregistrovať pomocou metódy *RegisterPlot(string name, tCreatorFPtr fptr)*. Prvý parameter je jedinečný názov grafu a druhý je ukazateľ na funkciu, ktorá vytvára konkrétnu inštanciu grafu a vracia naň ukazateľ. Túto funkciu si objekt triedy *PlotCreator* uloží do asociatívneho poľa (*std::map*) ako hodnotu, kľúčom je názov grafu. Pri volaní metódy *MakePlot(string name)* si vyhľadá podľa kľúča *name* funkciu na vytvorenie grafu, ktorú zavolá a vytvorí objekt konkrétneho grafu. Metóda ako vracia ukazateľ na tento objekt.

4.2.3 Implementácia grafických komponentov

Trieda Plot

Trieda *Plot* zastrešuje všetky prvky týkajúce sa knižnice Cairo, (cieľ, kontext a podobne), ktoré obsahuje ako privátne premenné. Po inicializácii objektu tejto triedy je potrebné mu priradiť pomocou metódy *LoadData* inštanciu triedy *EntropyData*, ktorá obsahuje mapu entropie a ďalšie informácie potrebné na zobrazovanie grafov. Zoomovanie je implementované pomocou metódy *SetSizeTo,* ktorá aplikuje transformačnú maticu tak, aby výsledný graf mal určené rozmery. Priblíženie grafov je obmedzené na určité maximum aj minimum. Až po nastavení rozmerov výsledného grafu je možné graf vykresliť pomocou metódy *Draw()*. Samozrejme po každej zmene rozmerov je treba graf znova prekresliť.

Vykresľovanie je implementované tak, že najprv sa vytvorí cieľová plocha (vykresľujeme do pamäte) a Cairo kontext. V priebehu vykresľovania nastavujeme rôzne stavové premenné, ako hrúbka čiary, farba výplne a podobne. V užívateľských súradniciach umiestňujeme jednotlivé vektorové entity do ciest a tie vytiahneme alebo vyplníme nastaveným zdrojom (napríklad štetcom s určenou farbou, alebo gradientom).

Často potrebujeme vykresliť čiaru, ktorá má presne danú hrúbku na výstupnom zariadení (napríklad 1 pixel na monitore). Alebo napríklad chceme, aby písmo nejakej popisky malo stále rovnakú veľkosť bez ohľadu na to, ako graf transformujeme alebo priblížime. Keďže však vykresľujeme v užívateľských súradniciach už na transformovaný obraz, zdeformuje to aj hrúbky čiar, keďže ich veľkosti definujeme pomocou rozmerov v týchto súradniciach. Preto je potrebné pred vyplnením čiary najprv aktuálnu transformačnú maticu uložiť a obraz transformovať pomocou jednotkovej matice na pôvodné rozmery zhodné s užívateľskými súradnicami. Po vyplnení treba obraz transformovať späť, aby sme mohli pokračovať v normálnom kreslení transformovaného obrazu. Na tento účel obsahuje trieda *Plot* chránené metódy *SaveTransform()* a *RestoreTransform()*.

Trieda PlotComponent

Táto trieda slúži ako rodičovská trieda pre všetky komponenty grafov. Jednotlivé komponenty sú určené na priame vykresľovanie už na existujúci Cairo kontext v existujúcej inštancii konkrétneho grafu odvodeného od triedy *Plot*. Každý komponent má kotvu určujúcu jeho polohu v užívateľských

súradniciach nastaviteľnú metódou *MoveTo*. Veľkosť každého komponentu v užívateľských súradniciach sa dá nastaviť pomocou metódy *SetPixSize*. Táto trieda obsahuje aj chránenú metódu *OptimizeEntropy*, ktorá spriemeruje entropiu na zadanú veľkosť a vráti novú skrátenú mapu entropie. Táto metóda sa využíva v dcérskych komponentoch, ktoré implementujú zobrazovanie samotnej grafickej časti mapy entropie.

4.2.4 Implementácia X-Y grafu

X-Y graf je implementovaný v triede *NormalPlot* a skladá sa z komponentov *EPNormalGrid*, *EPNormalPlot*, *EPDirStrap* a *EPSecDescription*. V cykle vykresľujeme graf mapy entropie pre každú sekciu pod seba.

Objekt triedy *EPNormalGrid* vykresľuje na zadanú pozíciu mriežku vyznačujúcu konkrétne hodnoty entropie a adresy. Tiež vykresľuje k nim prislúchajúce popisky. Jednotlivé čiary mriežky si zachovávajú svoju hrúbku bez ohľadu na to, ako je graf deformovaný alebo priblížený.

Samotný graf je implementovaný komponentom *EPNormalPLot*. Ten pre každú hodnotu entropie vykreslí samostatnú úsečku, ktorá začína v predchádzajúcej hodnote a končí v aktuálnej, kde index mapy entropie je súradnica x koncového bodu a hodnota entropie je súradnica y. Graf je vyplnený gradientom podľa nastavených farieb.

Objekt triedy *EPDirStrap* vykreslí na dané miesto pás zvýrazňujúci jednotlivé štruktúry odkazované z *IMAGE_DATA_DIRECTORY*. Inštancia triedy *EPSecDescription* vypisuje názvy sekcií nad ich grafy entropie. Výška pásu aj veľkosť textu sú nezávislé od priblíženia a deformácie grafu a ostávajú na monitore v rovnakej veľkosti. To umožňuje čitateľnosť textu bez ohľadu na zvolené priblíženie.

4.2.5 Implementácia hviezdicového grafu

Tento graf je implementovaný triedou *StarPlot* a skladá sa z tých istých komponentov, ako predošlý. Samotný hviezdicu a centrálnu časť nevykresľujeme pomocou komponentov. Pri vykresľovaní jednotlivých sekcií využívame transformácie. Najprv posunieme súradnicový systém do stredu hviezdice a potom vykresľujeme v cykle jednotlivé sekcie. Po vykreslení každej sekcie rotujeme súradnicový systém o potrebný uhol veľký 360 stupňov vydelených počtom sekcií.

V tomto grafe sa z estetických dôvodov veľkosť textu popisiek sekcií mení v závislosti na priblížení a je na užívateľovi, aby si napríklad pri malom priblížení nastavil korektnú veľkosť. Implicitne je táto veľkosť nastavená tak, že aj pri najmenšej veľkosti bude text čitateľný.

4.2.6 Implementácia kruhového grafu

Tento graf je implementovaný triedou *CirclePlot* a skladá sa z komponentov *EPCircleCore*, *EPCirclePlot*, *EPSecDescription* a *EPPLine*. V cykle vykresľujeme jednotlivé kruhové výseky predstavujúce sekcie. Po vykreslení každého rotujeme súradnicový systém o potrebný uhol.

Objekt triedy *EPCircleCore* vykresľuje centrálnu časť kruhového grafu. Jedná sa o kruhový výsek, v ktorom sú zvýraznené farebne formou menších kruhových výsekov jednotlivé štruktúry odkazované z *IMAGE_DATA_DIRECTORY*.

Samotný graf vo forme kruhového výseku je implementovaný komponentom *EPCirclePlot*. Ten vykresľuje podobne ako *EPNormalPlot* úsečku pre každú hodnotu mapy entropie s tým rozdielom, že využívame rotáciu aby mal graf kruhový tvar. Po vykreslení každej úsečky rotujeme súradnicový systém o uhol rovný uhlu kruhového výseku vydeleného veľkosťou mapy entropie.

Inštancia triedy *EPSecDescription* vypisuje názvy jednotlivých sekcií a objekt triedy *EPPLine* slúži na vykreslenie čiary spájajúcej názov sekcie s jej polkruhovým zobrazením. Niekedy pozostáva súbor zo sekcií s veľmi malou alebo nulovou veľkosťou nasledujúcich hneď za sebou. Keďže sa názvy sekcií zobrazujú v určitej vzdialenosti do stredu grafu, mohli by sa v tomto prípade prekrývať. To je ošetrené tak, že na základe veľkosti kruhového výseku sa zisťuje, či by sa text mohol prekrývať a ak áno, tak sa posunie ďalej od stredu pod predošlý text.

4.3 Grafické užívateľské rozhranie

Na implementáciu GUI sme použili .NET framework verzie 2.0. Súčasťou tejto sady nástrojov je knižnica Windows Forms (skrátene WinForms), ktorá slúži na vývoj GUI pod Windows (podrobnejšie viď [8]). Obsahuje triedy na vytváranie rôznych prvkov GUI, ako napríklad okno, panel nástrojov, menu a pod. Triedy implementujúce GUI komponenty spomínané v kapitole 3.3.1 sú odvodené od triedy System::Windows::Forms::Form.

Po spustení programu sa zobrazí okno vytvorené inštanciou triedy *GuiMainWindow*. Pri otvorení súboru sa vytvorí objekt triedy *EntropyData*, ktorý súbor načíta súbor a vypočíta entropiu a objekt triedy *ChildWindow*, ktorý zobrazí okno s grafom. Okno *ChildWindow* si ako privátnu premennú uchováva objekt triedy *EntropyData*, ktorý sprístupňuje objektom grafických tried.

4.3.1 Prepojenie grafiky s užívateľským rozhraním

Po vytvorení inštancie grafu pomocou metódy *MakePlot* a priradení mapy entropie grafu metódou *LoadData* nastavíme veľkosť grafu na veľkosť aktuálneho dcérskeho okna, do ktorého graf vykresľujeme. Metódou *Draw* ale vykreslíme graf len do pamäte. Pri zložitých grafoch s množstvom entít vykresľovanie vektorov samozrejme chvíľu trvá. Preto trieda *Plot* obsahuje metódu

DrawToHdc, ktorá prekreslí už vyrenderovaný obsah pamäte do okna. Táto metóda sa volá vždy, keď sa toto okno potrebuje prekresliť.

Tento prístup má svoje výhody aj nevýhody. Výhoda spočíva v tom, že prekresľovanie je veľmi rýchle, keď napríklad pracujeme s už vykresleným grafom vo veľkom priblížení. Keď ním pohybujeme, tak sa prekresľuje len bitmapový obsah pamäte na okno a nemusí sa volať metóda *Draw*. Nevýhoda je v tom, že pri veľkom priblížení sa v pamäti nachádza kompletný bitmapový obraz grafu, čo je pamäťovo náročné. Keďže si mapu entropie si najčastejšie prezeráme ako celok, pamäťová náročnosť pri veľkom priblížení až tak neprekáža.

4.4 Nastavenia grafov

Jednotlivé grafy sú plne nastaviteľné. Užívateľ môže meniť farby prvkov, hrúbky čiar, zapínať a vypínať niektoré komponenty (napríklad mriežku, popisky). Nastavenia sú ukladané pomocou inštancie triedy *PlotProps*. Sú implementované tri typy nastavení: farba (reprezentovaná objektom triedy *Color*, ktorý obsahuje RGB hodnoty farebnej informácie), číslo a pravdivostná hodnota. Tie sú uložené v troch asociatívnych poliach (STL trieda *map*), kde kľúčom je jedinečný názov nastavenia typu *string* a hodnotou samotné nastavenie. K nastaveniam sa pristupuje metódou *GetProps*, ktorá má jeden parameter typu *string*, ktorý obsahuje reťazec identifikujúci nastavenie a vracia samotnú hodnotu nastavenia. Metóda *AddProp* pridá nastavenie do zoznamu nastavení a metóda *SetProp* nastaví už existujúce nastavenie na určitú hodnotu.

Každá inštancia grafu obsahuje objekt triedy *PlotProps*. Pri vytváraní grafu si graf zaregistruje pomocou metódy *AddProp* všetky potrebné nastavenia, na základe ktorých sa bude zobrazovať. Komponenty grafov tiež pracujú s týmito nastaveniami, takže ak chceme použiť v grafe napríklad komponentu *EPNormalGrid*, musíme tiež zaregistrovať jej nastavenia. Na to má každý komponent statickú metódu, napríklad u *EPNormalGrid* zaregistrujeme nastavenia pomocou metódy *EPNormalGrid_InitializeSettings*. Ďalšie prípadné inštancie tejto triedy budú používať tie isté nastavenia.

Inštancia *PlotProps* je prístupné z grafu metódou *GetProps*, pomocou ktorej pracuje s nastaveniami inštancia GUI triedy *SettingsWindow* umožňujúca užívateľovi nastavenia dynamicky meniť. Po zmene nastavení sa graf vždy znova prekreslí.

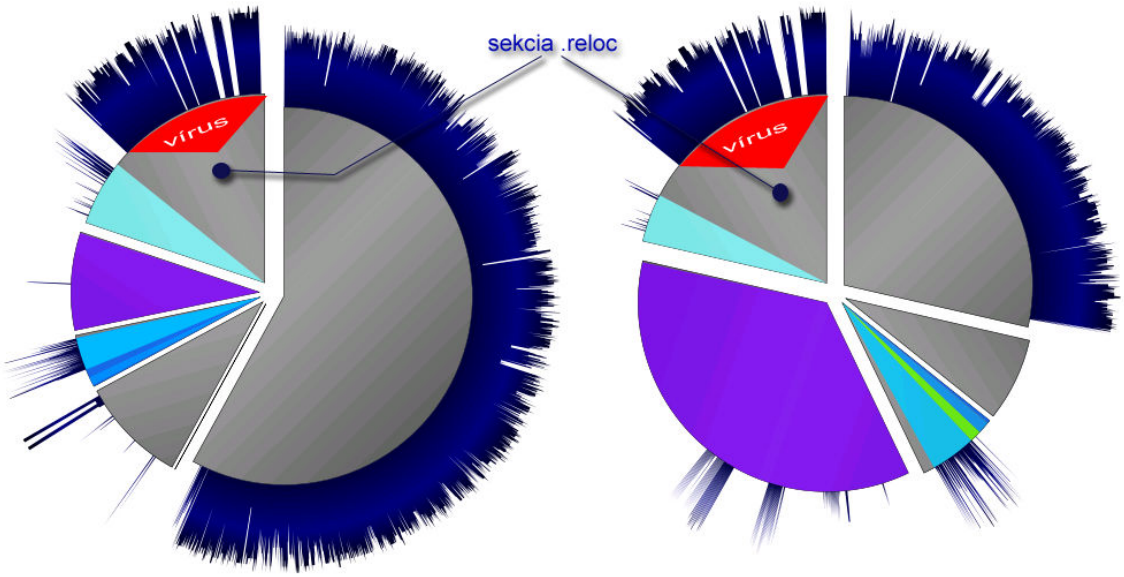
Nastavenia sa automaticky ukladajú a načítavajú pomocou súboru XML. Na to je použitá knižnica TynyXML implementovaná v jazyku C++.

4.5 Testovanie programu

Program bol testovaný pod operačnými systémami Microsoft Windows XP aj Vista. Kvôli bezpečnosti sa pri testovaní vzoriek vírusov využíval virtuálny stroj. Okrem testovania načítania

a zobrazenia grafov rôznych vzoriek PE EXE súborov sa testovala aj účinnosť grafickej reprezentácie na analýzu napadnutých súborov.

Napríklad na obrázku 5.1 vidíme mapy entropie dvoch rôznych vzoriek súborov napadnutých polymorfným vírusom Win32/Andras.7300. Jedná sa o úplne rozličné programy (prvý ja kalkulačka – calc.exe, druhý poznámkový blok – notepad.exe). Vidíme, že vírus napadá poslednú sekciu s tabuľkou relokácií (vyznačená svetlomodrou farbou v poslednej sekcii). V tejto sekcii nemajú dáta väčšinou vysokú entropiu, takže je na prvý pohľad jasné, že oblasť s vysokou entropiou na konci súboru je podozrivá. Keď sa pozrieme na tieto dva vzorky, vidíme dokonca, že sa časť mapy entropie prislúchajúca vírusu sa zhruba zhoduje u oboch vzoriek, aj keď sa jedná o binárne celkom odlišné vzorky. Červené označenie miesta napadnutého vírusom a modré popisky názvu sekcie sú doplnené kvôli názornosti, program to v grafe takto nezobrazuje.



Obrázok 4.1: Porovnanie dvoch vzoriek vírusu Andras

5 Záver

Hlavný cieľ práce bolo navrhnúť rozumnú grafickú reprezentáciu mapy entropie a navrhnúť program, ktorý tieto grafy zobrazí. Program takto prezentuje túto zaujímavú metódu štatistickej analýzy PE EXE súborov. Pomocou jednotlivých grafov pekne vidieť účinnosť tejto metódy, kde ľahko nájdeme napadnutú časť súboru alebo dokážeme spozorovať podobnosť mapy entropie rôznych vzoriek polymorfných vírusov.

Vo firme Grisoft už implementovali podobný nástroj, ten však zobrazoval mapu entropie iba ako pole hexadecimálnych hodnôt. Pre laika to bolo ťažko čitateľné a nevhodné na prezentáciu tejto metódy. Výstupy z nášho grafického programu sú naopak na mediálnu prezentáciu priamo vytvorené.

Samozrejme sa nejedná o konečnú verziu programu. V budúcnosti by sa mohla implementovať okrem ďalších variant grafov aj interaktivita, kde by užívateľ mohol presúvať jednotlivé grafické prvky pomocou myši, ako to je napríklad v grafických editoroch. Iná možnosť vylepšenia by bolo pridať aj nejaké trojrozmerné grafy (implementované napríklad v OpenGL).

Literatúra

[1] Alexandrescu, A. Moderní programování v C++. Brno, Computer press, 2004, s. 223-227.

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Návrh programů pomocí návrhových vzorů. Praha, Grada, 2003.

[3] Hák, I. Moderní počítačové viry. [bakalárska práca]. Univerzita Hradec Králové. Dostupné na URL: http://www.viry.cz/viry.cz/kniha/kniha.pdf (máj 2008)

[4] Kršek, P. Základy počítačové grafiky, Studijní opora. Vysoké učení technické v Brně, Fakulta informačných technológií, Brno, 2006, s. 10-14. Dostupné na URL: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/texts/izg_opora.pdf (máj 2008)

[5] Křivka, Z., Kolář, D. Principy programovacích jazyků a objektově orientovaného programování, IPP – II, Studijní opora. Vysoké učení technické v Brně, Fakulta informačných technológií, Brno, 2006, s. 11. Dostupné na URL: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IPP-IT/texts/IPP-II-ESF-1_1_printable.pdf (máj 2008)

[6] Macák, T. Teorie Informace. Česká zemědelská univerzita v prahe, s. 2-3. Dostupné na URL: http://pef.czu.cz/~macak/kybernetika/kb5-cv.doc (máj 2008)

[7] Prispievatelia Wikipedie. Objektově orientované programování [online]. Wikipedia: Otvorená encyklopédia. Posledná modifikácia: 1. apríla 2008. [cit. 10. 05. 2008]. Dostupné na URL: http://cs.wikipedia.org/wiki/Objektově_orientované_programování (máj 2008)

[8] Sells, C. C# a WinForms, programování formulářů Windows. Brno, Zoner Press, 2005.

[9] Szor, P. Počítačové viry. Brno, Zoner Press, 2006.

[10] Urman, M. Cairo Tutorial [online]. Posledná modifikácia: 3.februára 2008. [cit. 10. 05. 2008]. Dostupné na URL: http://cairographics.org/tutorial/ (máj 2008)

[11] Zemánek, J. Obecná struktura PE souboru [online]. Posledná modifikácia: 27.júna 2001. [cit. 10. 05. 2008]. Dostupné na URL: http://www.builder.cz/art/asembler/PE1.html (máj 2008)

Zoznam príloh

Príloha 1. Metriky kódu

Príloha 2. Užívateľský manuál

Príloha 3. CD so zdrojovými kódmi programu, manuálom a programovou dokumentáciou

Príloha 1. Metriky kódu

Uvedené sú štatistiky z vlastného zdrojového kódu programu (stav k dátumu 13.mája 2008). Zdrojové kódy použitých knižníc nerátame.

Počet súborov:	25
Počet tried:	24
Počet riadkov (bez komentárov):	4417
Veľkosť spustiteľného súboru (bez ladiacich informácií):	479 232 bajtov

Príloha 2. Užívateľský manuál

1 Popis programu EntropyGraph

Program je určený na zobrazenie grafov mapy entropie PE EXE súborov. Na výber sú tri typy grafov: *Normal plot* (X-Y graf), *Circle plot* (kruhový graf) a *Star plot* (hviezdicový graf). Grafy je možné pomocou nastavení upraviť a exportovať do formátov PNG, PDF a SVG.

1.1 Normal plot (X-Y graf)

Tento graf zobrazuje mapy entropie sekcií zoradené pod seba. Popisky (entropia a RAW adresy) sú zobrazené v hexadecimálnych hodnotách. Štruktúry z *IMAGE_DATA_DIRECTORY* sú vyznačené pomocou farebných pásov.

1.2 Star plot (Hviezdicový graf)

Tento graf zobrazuje mapy entropie sekcií usporiadané do hviezdice. Graf má pevný pomer strán a nedá sa naťahovať. Inak preň platí to, čo u predošlého.

1.3 Circle plot (Kruhový graf)

Graf zobrazuje mapy entropie sekcií po obvode kruhových výsekov. Veľkosť výseku zodpovedá veľkosti sekcie, celý kruh (uhol 360 stupňov) zodpovedá veľkosti celého súboru. V centrálnej časti kruhu farebne vyznačujeme jednotlivé štruktúry odkazované z *IMAGE_DATA_DIRECTORY*. Je možné zobraziť legendu zobrazujúcu názvy sekcií. Graf má pevný pomer strán a nedá sa naťahovať.

2 Použitie programu

Program môže načítať len súbory vo formáte PE EXE (spustiteľné EXE súbory, súbory knižníc DLL a pod.). Keďže algoritmus výpočtu entropie je náročný na výkon, môže trvať načítanie väčších súborov (zhruba nad 1MB) rádovo aj niekoľko desiatok sekúnd. Je možné načítať viacero súborov naraz a prepínať medzi nimi pomocou záložiek v hornej časti okna. Po načítaní súboru sa zobrazí implicitný graf *Normal plot*. Typ grafu môžeme prepínať pomocou menu alebo panela nástrojov. Grafy môžeme ľubovoľne priblížiť alebo oddialiť. Natiahnutie a roztiahnutie na výšku alebo na šírku je možné iba u grafu *Normal plot*. Graf je možné po priblížení posúvať priamo myšou. Graf je možné automaticky roztiahnuť na veľkosť okna kliknutím na ikonu aktuálneho grafu.

2.1 Položky menu, panela nástrojov a klávesové skratky.

Menu	Toolbar	Kláves. skratka	Popis
File → Open	Open	Ctrl + O	Zobrazí dialóg na otvorenie súboru
File → Export	Export	Ctrl + S	Zobrazí dialóg na export grafu do súboru.

File → Close	Close	Ctrl + Q	Zatvorí načítaný PE EXE súbor.
File → Exit		Alt + F4	Ukončí program.
View → Zoom in	Zoom in	+	Zväčší graf
View → Zoom out	Zoom out	-	Zmenší graf
View → Stretch horizontal → bigger	Stretch horizontal +		Natiahne graf do šírky. Prístupné len u grafu Normal plot.
View → Stretch horizontal → smaller	Stretch horizontal -		Zúži graf. Prístupné len u grafu Normal plot.
View → Stretch verticall → bigger	Stretch vertical +		Natiahne graf do výšky. Prístupné len u grafu Normal plot.
View → Stretch vertical → bigger	Stretch vertical -		Zníži graf. Prístupné len u grafu Normal plot.
Graph → Settings	Settings	F5	Zobrazí okno Plot Settings s nastaveniami grafu.
Graph → Normal plot	Normal plot		Zobrazí X-Y graf.
Graph → Circle plot	Circle plot		Zobrazí kruhový graf.
Graph → Star plot	Star plot		Zobrazí hviezdicový graf.
Help → EntropyGraph Help		F1	Zobrazí túto nápovedu.
Help → About			Zobrazí krátku informáciu o programe.

2.2 Nastavenia grafov

Nastavenia grafov je možné meniť pomocou okna *Plot Settings*. Na záložke *Properies* sa nachádzajú nastavenia umožňujúce zapínať a vypínať jednotlivé prvky grafov. Záložka *Colors* obsahuje nastavenia farieb. Kliknutím na položku (farebný obdĺžnik vyplnený nastavenou farbou) je možné zmeniť farbu. Záložka *Sizes* obsahuje numerické nastavenia (veľkosti, počet a pod.). Tlačidlo *OK* uloží zmeny a zatvorí okno. Tlačidlo *Apply* má tú istú funkciu s tým rozdielom, že sa okno s nastaveniami ešte nezatvorí. Tlačidlo *Cancel* zatvorí okno a zruší všetky neuložené zmeny.

2.3 Zoznam nastavení grafov

Legenda: N – Normal plot, S – Star plot, C – Circle plot

Nastavenie	Typ grafu			Popis
	N	S	C	
Properties				
Auto max x grid limit on	x	x		Zapína a vypína automatické obmedzenie hustoty mriežky (na osi x zobrazujúcej súbor).
Grid antialiasing on	x	x		Zapína vyhladené zobrazenie mriežky
Grid on	x	x		Zapína mriežku
Grid text on	x	x		Zapína popisky x a y hodnôt na mriežke
Plot line on	x	x	x	Zapína obvodovú čiaru graf. Vhodné na grafy súborov s malou veľkosťou.
Pointing line on			x	Zapína čiaru ukazujúcu na sekciu v kruhovom grafe.
Section name text on	x	x	x	Zapína popisky názvov sekcií
Section name text on	x	x	x	Zapína popisky s názvami sekcií.
Colors				
Axis color	x	x		Farba os
Background gradient color 1	x	x	x	Farba pozadia
Background gradient color 2	x	x	x	Farba pozadia
Circle color			x	Farba kruhu u kruhového grafu.

Circle line color			x	Farba obvodovej čiary kruhového grafu.
Data directory strap color	x	x		Farba pásu zvýrazňujúceho sekcie z IMAGE_DATA_DIRECTORY.
Grid background color	x	x		Farba pozadia mriežky
Grid color	x	x		Farba mriežky
Grid text color	x	x		Farba popisiek na mriežke
Plot gradient color 1	x	x	x	Prvá farba výplne grafu.
Plot gradient color 2	x	x	x	Druhá farba výplne grafu.
Plot gradient color 3	x	x	x	Tretia farba výplne grafu.
Plot line color	x	x	x	Farba obvodovej čiary grafu.
Pointing line color			x	Farba čiary ukazujúcej na sekciu u kruhového grafu.
Section name text color	x	x	x	Farba popisiek s názvami sekcií.
Sizes				
Circle line width			x	Šírka obvodovej čiary kruhového grafu.
Circle spacing			x	Odstup jednotlivých sekcií v kruhovom grafe.
Data directory strap height	x	x		Výška pásu zvýrazňujúceho sekcie z IMAGE_DATA_DIRECTORY.
Grid text size	x	x		Veľkosť popisiek. Táto veľkosť je nezávislá na priblížení.
Maximal x entropy size	x	x	x	Maximálna zobraziteľná hodnota entropie. Ak bude mať sekcia väčšiu entropiu, tak sa zaokrúhli na túto hodnotu.
Maximal x grid number	x	x		Maximálny počet popisiek x-hodnôt. (funguje, ak je zapnutý Auto max x grid limit on) Reguluje hustotu x-ovej mriežky.
Plot line width	x	x	x	Hrúbka obvodovej čiary grafu.
Pointing line width			x	Šírka čiary ukazujúcej na sekciu u kruhového grafu.
Section name text size	x	x	x	Veľkosť popisiek s názvami sekcií.
X grid per units	x	x		Hustota mriežky na osi x. (za koľko jednotiek osi x – 16 bajtov súboru – vykreslí čiara mriežky s popiskou obsahujúcou RAW adresu)
Y grid per units	x	x		Hustota mriežky na osi y. (za koľko jednotiek osi y – jednotka entropie – vykreslí čiara mriežky s popiskou obsahujúcou hodnotu entropie)

2.4 Štruktúry IMAGE_DATA_DIRECTORY

Štruktúry odkazované z *IMAGE_DATA_DIRECTORY* sú v grafoch zvýraznené farbami. Tieto farby je možné nastaviť pomocou *Plot Setings* na záložke *Colors*. Táto záložka slúži aj ako legenda pre jednotlivé farby. Nasledujúca tabuľka obsahuje zoznam 16 štruktúr odkazovaných z *IMAGE_DATA_DIRECTORY*. Sedemnásta zvýrazňovaná štruktúra je pole *IMAGE_THUNK_DATA* a osemnásta je pole *IMAGE_IMPORT_BY_NAME*. Názvy súhlasia s názvami nastavení ich farieb.

Index	Názov
1	Exports data directory color
2	Imports data directory color
3	resources data directory color
4	Exceptions data directory color
5	Security data directory color
6	Relocations data directory color
7	Debug info data directory color
8	Architecture specific data data directory color

9	Global pointer data directory color
10	Thread Local Storage data directory color
11	Load Configuration data directory color
12	Bound Imports data directory color
13	Import Address Table data directory color
14	Delay Load Imports data directory color
15	COM Runtime descriptor data directory color
16	0x0F data directory color
17	IAT: Jump Table data directory color
18	IAT: Name Table data directory color

2.5 Export grafu

Grafy je možné exportovať do formátov PNG, PDF a SVG. Graf sa vyexportuje v takej veľkosti, v akej sa zobrazuje na obrazovke. Ak je potrebná určitá veľkosť obrázka, je potrebné graf najprv adekvátne priblížiť alebo oddialiť.

2.6 Poznámky k nastaveniam grafov

Keď potrebujeme presnú mapu entropie pri veľkých súboroch, je možné zaokrúhľovanie vypnúť nastavením hodnoty *Maximal x entropy size* na veľmi vysokú hodnotu. POZOR, vykresľovanie grafu veľkej mapy entropie môže trvať veľmi dlho. Implicitná hodnota, pri ktorej to ide celkom rýchlo je 2000.

Hustotu mriežky ovplyvňujeme nastaveniami *X grid per units* a *Y grid per units*, kedy hodnota 1 znamená, že sa čiara mriežky vykresľuje za každú jednotku na danej osi. Čím vyššia hotnota, tým je nižšia hustota mriežky. Pri malom priblížení sa môže niekedy veľký text popisiek prekrývať, čo nevyzerá veľmi esteticky. Preto je potrebné zmeniť buď hodnotu *X*/*Y grid per units* (ovplyvníme tým hustotu mriežok a popisiek) alebo zmenšiť veľkosť písma popisiek (*Grid text size*). Keď nechceme riešiť hustotu mriežky na x osi pre rôzne veľké súbory, je vhodné zapnúť nastavenie *Auto max x grid limit on*, ktoré automaticky obmedzí počet čiar x-mriežky na určitú hodnotu (nastaviteľnú pomocou *Maximal x grid number*).

U kruhového grafu je možné, že sa názvy sekcií nebudú pri atypických veľkostiach sekcií zobrazovať korektne. Dá sa to opraviť úpravou priblíženia.

3 Inštalácia programu

Na použitie programu stačí skopírovať súbory *EntropyGraph.exe*, *libcairo-2.0.dll*, *libpng13.dll* , *zlib1.dll* a XML súbory s nastaveniami grafov s názvami „*PROPS názov grafu.xml*“ do niektorej zložky. Keď program v spúšťacom adresári XML súbor s nastavením nenájde, vytvorí nový pri prvom pokuse uložiť nastavenia. Je potrebné mať nainštalovaný .NET Framework 2.0 alebo novší.