VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

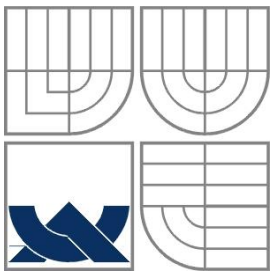
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROGRAM PRO BEZPEČNOSTNÍ KONTROLU IKON OPERAČNÍHO SYSTÉMU

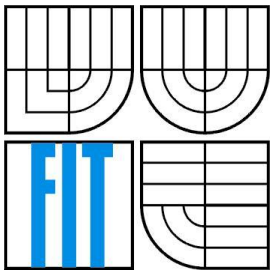
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE VLADIMÍR RUŽIČKA
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROGRAM PRO BEZPEČNOSTNÍ KONTROLU IKON OPERAČNÍHO SYSTÉMU

SECURITY ICON CHECKING APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLADIMÍR RUŽIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LADISLAV RUTTKAY

BRNO 2010

Abstrakt

Tato práce se zaobírá ikonami v systémech Windows a jejich formátem. Ukazuje jakým způsobem s nimi systém pracuje a jak můžou být zneužity pomocí malware. Dále skoumá spůsoby, jakými je možné ikony vzájemně porovnávat. Práce obsahuje také aplikaci, která ikony testuje.

Abstract

This work is about icons in Windows systems and about their format. The work shows the way, how the system use them and how malware can profit on icons. It explore how icons can be compared one to another. The work contains application, which tests icons.

Klíčová slova

ikona, obraz, bitmapa, PNG, DIB, malware, RGB, RGBA, CMY, CMYA, pixel, detekce, antivirus, průhlednost

Keywords

icon, image, bitmap, PNG, DIB, malware, RGB, RGBA, CMY, CMYA, pixel, detection, antivirus, transparency

Citace

Ružička, Vladimír: Program pro bezpečnostní kontrolu ikon operačního systému, bakalářská práce, Brno, FIT VUT v Brně, 2010

Program pro bezpečnostní kontrolu ikon operačního systému

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ladislava Ruttkaye. Další informace mi poskytli pan Pavel Krčma a Ing. Michal Španěl
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Vladimír Ružička

17.5.2010

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Ruttkayovi a také panu Krčmovi a Ing. Španělovi za poskytnuté informace nutné k vytvoření této práce

Obsah

1 Úvod

Tvorcovia škodlivého kódu hľadajú stále nové možnosti ako sa dostať do cudzích počítačov a zneužiť ich. Ja sa budem venovať prípadom, keď sa snažia zvýšiť dôveryhodnosť svojich programov tým, že ich vydávajú za iný program alebo dokument pomocou ich ikony a takto sa snažia užívateľa oklamať. Mojím primárnym cieľom bude vytvoriť aplikáciu, ktorá dokáže takéto nebezpečné programy odhaliť práve pomocou porovnávania jej ikony s ikonami, s ktorými sa užívateľ dostáva bežne do styku.

Chcel by som vysvetliť akým spôsobom prichádza k oklamaniu užívateľa. Pokúsim sa o to v druhej kapitole a takisto uvediem konkrétnu situáciu.

Detekcia škodlivého kódu pomocou ikony nepatrí medzi typické spôsoby, ktoré sú používané antivírusovými spoločnosťami. Naviac svoje konkrétne algoritmy si väčšinou strážia a nezverejňujú ich. Je teda možné, že niektoré z nich ikony testovaných súborov kontrolujú. V tretej kapitole by som chcel čitateľovi priblížiť základné rozdelenie odhaľovania malware podľa úspešnosti a pokúsiť sa o zaradenie a porovnanie mnou vytváranej aplikácie. Ďalej by som chcel čitateľa zoznámiť s existujúcimi bezpečnostnými systémami.

Štvrtá kapitola je zameraná priamo na samotné ikony, na ich formát, spôsob ich používania systémom. Takisto uvádzam aj vysvetľujúce obrázky ako to všetko do seba zapadá.

V piatej kapitole rozoberám možné spôsoby, akými môžu byť ikony medzi sebou porovnávané. Veľkú pozornosť venujem existujúcim farebným modelom a hlavne spôsobu ako vnímať priehľadnosť, ktorá sa u ikon nachádza, pretože tá celé porovnávanie komplikuje.

Ďalšie kapitoly sa už zaoberajú samotnou aplikáciou a to jej návrhom, implementáciou a takisto zhodnotením úspešnosti. Obsiahnuté sú aj príklady konkrétnych porovnávaní.

Ešte než začnem zo samotnou prácou, chcel by som pridať vysvetlenie slova malware, ktoré som už vyššie použil a spomínať ho budem pravdepodobne ešte viackrát. Podľa [1] sa jedná o softvér, ktorý bol vytvorený za účelom páchať škodu užívateľovi v akomkoľvek smere, alebo vykonávať činnosť, o ktorú užívateľ nestojí. Veľmi často je namiesto malware používané nepresné označenie vírus, ktorý je však iba určitou časťou malware spolu s mnohými ďalšími. Dokonca aj súčasné antivírusové programy by mali byť správne označované ako antimalware, keďže sa nešpecifikujú iba na samotné víry. Ich pomenovanie je už však zaužívané z historických dôvodov.

2 Problematika podvrhnutých ikon

Väčšina ľudí sa nevyzná v problematike ochrany počítačov, preto oklamať ich nie je často v tomto smere problém. Bežný užívateľ je naučený určité stereotypy práce s počítačom. Teda nerozmýšľa nad tým, čo sa deje po každom jeho kliknutí, ale automaticky opakuje už mnohokrát pred tým urobené postupy. To prináša tvorcom malware mnohé možnosti ako napadnúť počítač. Stačí dokonca aj skúseného užívateľa vystaviť situácii, ktorá je dostatočne podobná situáciám, na ktoré je zvyknutý a absolútne bez akéhokoľvek podozrenia vykoná presne to, o čo tvorcovi malware išlo. Dokonca aj keď spozoruje, že niečo nie je tak ako byť má, s ohľadom na svoje znalosti a schopnosti sa často krát predsa len rozhodne postupovať podľa svojich naučených postupov. Následky môžu byť pre neho samozrejme fatálne.

Jednou z možností ako oklamať užívateľa je použitie dobre známej ikony. Nebezpečný súbor sa mu potom javí ako iný, o ktorom vie, že je bezpečný a ani nezačne rozmýšľať, či súbor otvorí alebo nie. Ako ukážkový prípad môžem uviesť užívateľa, ktorý si stiahne z internetu súbor napr. pesničku vo formáte MP3. Namiesto toho však práve uložil do svojho počítača vírus, teda spustiteľný súbor, ktorý má ale rovnakú ikonu, ako užívateľom žiadaný MP3 súbor. V takomto prípade nemá takmer ani šancu spozorovať niečo podozrivé, keďže prípony súborov bývajú väčšinou skryté, nehovoriac o tom, že bežnému užívateľovi by to aj tak často krát nepomohlo. Podobná situácia nastáva aj v prípade, že nebezpečný súbor ma napr. ikonu archívu alebo priečinku. V tomto prípade už užívateľ môže spozorovať určité nezrovnalosti, ale aj tak pokračuje ďalej, pretože adresár alebo archív už otváral mnohokrát predtým.

Našťastie aspoň moderné operačné systémy upozornia užívateľa, keď sa pokúša spustiť neznámy program a opýtajú sa ho, či to mal naozaj v úmysle. V tejto chvíli už užívateľ spozornenie, ale s ohľadom na svoje znalosti v počítačovej oblasti, sa stále môže rozhodnúť špatne.

Tieto ukážkové prípady majú spoločnú jednu vec. Užívateľ sťahuje dáta z neznámeho prostredia čo je určite nebezpečné samé o sebe. Avšak spomínané súbory sa samozrejme môžu dostať do počítača aj iným spôsobom a nemusí to byť priamo užívateľova vina, pretože prostredie odkiaľ súbor získal mu môže byť dostatočne známe. Môže sa samozrejme nachádzať na požičanom CD, DVD alebo inom externom pamäťovom zariadení. Alebo dokonca priamo na inom počítači v lokálnej sieti. Takisto ho môže prijať ako prílohu v emaile od priateľa, ktorý sa nechal nachytať tiež. Prípadne sa už dokonca nebezpečný súbor na počítači môže nachádzať po pôsobení iného malware, ktorý stiahol tento súbor bez vedomia užívateľa.

Z toho vyplýva, že podvrhovanie ikon samé o sebe tvorcovi škodlivého kódu úspech neprinesie. Minimálne rovnako dôležitý je spôsob, akým sa tento nebezpečný súbor dostane do počítača. Týmto sa však už ja zaoberať nebudem. Pre mňa je teda podstatný konkrétny súbor, ktorého zdroj nie je známy.

3 Porovnanie bezpečnostných systémov

Drvivá väčšina riešení majú spoločnú jednu zásadnú vec. V odhaľovaní malware a chránení konkrétneho systému sú oveľa úspešnejšie u známeho škodlivého kódu, ktorého vzorky už majú vo svojej databáze. Za predpokladu, že sú vzorky dostatočne dlhé a správne vybrané, sú pri tejto metóde bezpečnostné programy veľmi presné a v prípade rezidentnej ochrany takýto škodlivý kód nemá prakticky žiadnu šancu im uniknúť. Takisto miera falošných pozitívnych detekcií je veľmi nízka. V tejto kapitole som voľne čerpal z [2].

Už menej úspešné sú bezpečnostné riešenia pri odhaľovaní nového škodlivého kódu. Nezostáva im nič iné ako sledovať správanie jednotlivých programov a s určitou pravdepodobnosťou tak detekovať škodlivý kód pokúšajúci sa napr. o zápis do registrov alebo iných rôznych miest v systéme. Možností, podľa ktorých je možné nájsť program s nekorektným chovaním je viacero, avšak majú spoločnú jednu vec. Antivírusový softvér musí analyzovať priamo zdrojový kód programu a hľadať podozrivé operácie, prípadne dokonca nutnosť spustiť program v nejakom virtuálnom prostredí. Táto činnosť je nevýhodná z hľadiska výkonu počítača. Nielenže sa musí vykonať kód skúmaného programu, ale naviac to celé prebieha v emulovanom prostredí, ktoré samo o sebe je náročné na prostriedky. Samozrejme netreba ešte zabudnúť na samotnú kontrolu vykonávaných inštrukcií a všetko spolu môže teda klásť naozaj vysoké nároky na počítač. Ďalší problém z hľadiska výkonu, ktorý so sebou prináša tento spôsob odhaľovania škodlivého kódu, prichádza v prípade príliš dlho bežiaceho skúmaného programu. Antivírusový softvér sa môže uskromniť iba na určitý čas sledovania, ktorý by mal vo väčšine prípadov stačiť na odhalenie malware. Avšak práve toto na druhej strane môžu zneužiť jeho tvorcovia. Najskôr nechajú po dostatočne dlhú dobu svoj program vykonávať korektný kód a až potom príde k samotnému infikovaniu počítača.

3.1 Zaradenie detekcie porovnávaním ikon

Porovnávaním ikon neznámych programov s dobre známymi ikonami prichádza k hľadaniu zatiaľ ešte neodhaleného malware. Avšak vôbec neprichádza k analýze vykonávaného kódu programu a táto metóda sa skôr podobá na hľadanie už známych vzoriek škodlivého kódu. Namiesto nich sa však porovnávajú v systéme dobre známe ikony priamo s ikonou neznámeho súboru. Výkonnostné nároky sú teda bližšie k nárokom pri hľadaní známeho škodlivého kódu, ale pritom dochádza k hľadaniu malware, kde nie je podstatné, či už bol pred tým odhalený. Čo sa týka úspešnosti hľadania, je treba rozdeliť porovnávanie ikon do dvoch kategórií:

- **Totožnosť ikon** – v prípade, že nejaký spustiteľný súbor priamo používa systémovú ikonu (napr. ikonu adresára v systéme Windows), je možné tento súbor s veľmi vysokou pravdepodobnosťou označiť ako malware. Prakticky jedinou možnosťou, že by prišlo k omylu, je prípad keď vývojári užitočného programu použijú ako ikonu pre svoj program priamo ikonu používanú v systéme Windows, čo určite nie je príliš zvyčajná situácia. Táto možnosť by sa teda dala v ohľade úspešnosti priamo porovnávať s úspešnosťou pri hľadaní už známeho škodlivého kódu pomocou jeho vzorky,

- **Podobnosť ikon** – pri hľadaní podobnosti medzi ikonami je to už podstatne zložitejšie. Veľmi bude záležať na kvalite samotného porovnávania a nastavenej úrovne, kedy budú dve ikony označené za podobné. Táto možnosť by sa v úspešnosti hľadania malware mohla porovnávať s úspešnosťou pri analýze chovanie neznámeho programu, kde taktiež veľmi závisí od kvality algoritmu hľadania. Pri hľadaní už známeho malware má kvalita algoritmu vplyv skôr iba na rýchlosť a podstatná je veľkosť a aktuálnosť databázy vzoriek. Čo sa týka mieri falošných pozitívnych detekcií, táto priamo závisí od zvolenej úrovne, kedy sú ikony požadované za podobné. Obdobne to funguje aj pri analýze chovania neznámeho programu, kde je dôležité, čo bude považované za ešte korektné správanie a čo už nie.

Hľadanie škodlivého kódu podľa ikon z tohto porovnania úspešnosti a nárokov na počítač s už reálne používanými spôsobmi teda vychádza naozaj dobre. Na druhú stranu ale treba povedať, že zmienené používane spôsoby sa dajú nazývať univerzálnymi, fungujúcimi v princípe na všetky nebezpečné programy. Na rozdiel od toho, porovnávanie ikon je veľmi špecifický spôsob, ktorý je úspešný výhradne u malware, ktorý sa snaží oklamať užívateľa práve svojou ikonou. Ďalšou nevýhodou je samotná výkonová náročnosť porovnávania obrazu. Hlavne v prípade ikon používajúcich vysoké rozlíšenie (256×256) môže byť táto činnosť až príliš náročná a vyššie spomínané výhody v podobe nízkych systémových požiadavkou sú teda nereálne.

Informácie o existujúcich algoritmoch priamo súvisiacich s odhaľovaním škodlivého kódu pomocou podvrhnutých ikon, sa mi nájsť nepodarilo. Požiadal som teda o radu Pavla Krčmu[1]. Jeho odpoveď bola: „ Myslím si, že použiť nejaký obrazový porovnávací mechanizmus na ikony, je nápad nový“. Z toho vyplýva, že nemôžem návrh môjho algoritmu založiť na už existujúcich a fungujúcich bezpečnostných riešenia, kde by som mohol použiť výhody jednotlivých riešení a vyvarovať sa ich nevýhodám, ale treba začať úplne od začiatku.

3.2 Existujúce bezpečnostné systémy

Na trhu sa nachádza naozaj veľké množstvo rôznych bezpečnostných riešení. Od jednoduchých bezplatných antivírusových programov, cez špeciálne zamerané programy na určité typy malware, až po komplexné riešenia zahŕňajúce všetky spomínané aj nespomínané oblasti. Spomínať na tomto mieste všetky je prakticky nemožné. Preto sa pokúsim vybrať aspoň tie najvýznamnejšie alebo určitým spôsobom zaujímavé antivírusové systémy.

Norton (Symantec) – ponúka viacero platených verzií svojich bezpečnostých riešení. Veľmi často býva predinštalovávaný na nových počítačoch. Vďaka tomu patrí medzi najpoužívanejšie.

Microsoft Security Essentials – nové bezpečnostné riešenie priamo od tvorcu systémov Windows, ktoré využíva nové rozhranie jadra Windows určené priamo pre antivírusy. Vďaka tomu dokáže ako jediné z velkých bezpečnostných riešení odolať útoku označovanému Khobe, ktorý objavila bezpečnostná skupina Matousec [13].

[1] Pavel Krčma pracuje 10 rokov v spoločnosti AVG. Jeho súčasná pozícia je vírusový analytik.

AVG, Avast, Eset – bezpečnostné riešenia pochádzajúce z Českej a Slovenskej republiky, ktoré sa dokázali globálne presadiť.

Clam Antivirus – najvýznámnejší antivírus vydávaný pod open-source licenciou.

Zaujímavou alternatívou je aj možnosť nechať si skontrolovať počítač online, bez potreby inštalovania bezpečnostného systému. Toto umožňujú napr. Kaspersky Lab, BitDefender, Eset, Panda a ďalší. Žiadne z bezpečnostných riešení nie je 100percentné. Využívať však naraz viacero produktov je veľmi problematické, v niektorých prípadoch to dokonca ani nie je umožnené. Preto práve možnosť online kontroly v prípade potreby je veľmi dôležitá.

Zhodnotiť, ktorý zo systémov je najlepší, je až príliš komplikované. Dokonca na internete sa objavujú mnohé porovnanie, ktorých výsledky sa často veľmi líšia. Ak by som mal niektorý bezpečnostný systém vyzdvihnúť, bolo by to antivírusové riešenie od Microsoftu, ktoré ako som spomínal vyššie využíva nové moderné rozhranie jadra Windows. Za veľmi podstatné tiež považujem, aby bezpečnostný systém obsahoval rezidentnú ochranu, kedy sú v reálnom čase kontrolované všetky súbory, ku ktorým sa pristupuje. Túto vlastnosť však spĺňajú všetky významné produkty.

4 Ikony v systémoch Windows

4.1 Používanie ikon

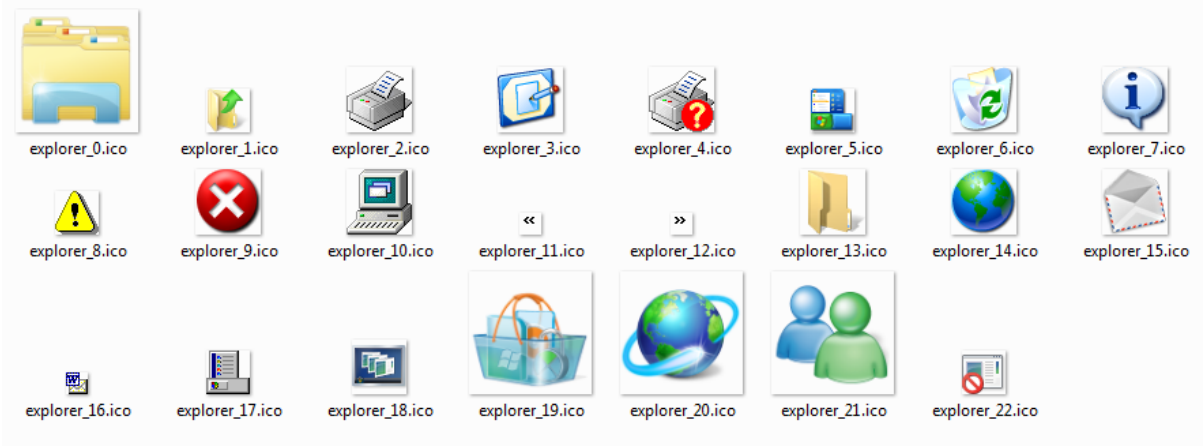
Ikona každej aplikácie sa zobrazuje na rôznych miestach v rôznych veľkostiach. Ikona môže byť zobrazená v ponuke štart, na pracovnej ploche, v paneli úloh. Takisto v adresári je možné zvoliť rôzne spôsoby zobrazenia ako dlaždice, zoznam atď. Používanie iba jedného obrázku ako ikony, ktorého veľkosť bude upravovaná podľa aktuálnej potreby, je však problematické. Pri veľkej ikone je žiaduce, aby vyzerala čo najlepšie. To znamená vysokú úroveň detailov, tieň, zobrazovanie trochu zo strany, aby vytvorila priestorový efekt [3]. Avšak pri zobrazení takto prepracovanej ikony na malom priestore, sa táto stane špatne rozpoznateľnou. Ak teda má ikona súboru vyzerať dobre a prehľadne vo všetkých prípadoch, je nevyhnutné, aby v súbore bolo viacej verzií k zobrazeniu (obr. 4.1).

Obrázok 4.1 – Najmenší obrázok nevyužíva tiene a počítač je zobrazený priamo zpredu. Obrázok prevzatý z [3]

Štandardné veľkosti obrázkov v ikonách systému Windows sú: 256 × 256, 48 × 48, 32 × 32, 16 × 16. Práve tieto veľkosti by mala obsahovať ikona, ktorá sa nachádza v aplikácii. V prípade, že aplikácia neobsahuje žiadnu ikonu, systém Windows jej pridelí implicitnú. Ikona však môže obsahovať aj obrázky iných rozmerov, takže ak systém nenájde obrázok požadovanej veľkosti, vyberie iný, ktorý najviac vyhovuje požiadavkám. Okrem samotného rozmeru však záleží pri rozhodovaní aj na farebnej hĺbke. Preto jedna ikona môže obsahovať aj obrázky rovnakých rozmerov ale s odlišnou farebnou hĺbkou.

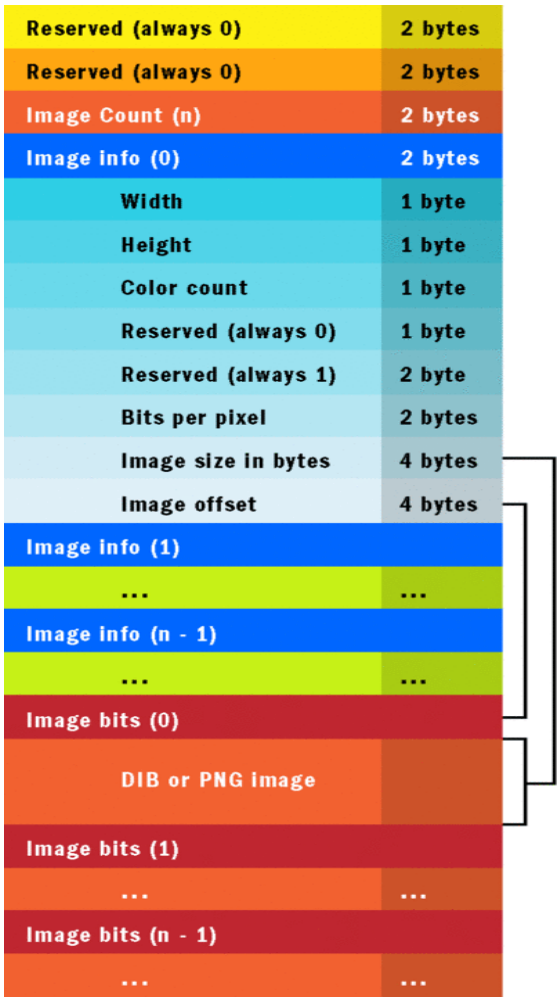
4.2 Formát ICO

Ikona obsahuje teda niekoľkokrát viacmenej taký istý obrázok v rôznych veľkostiach a farebných hĺbkach. Súbory typu EXE a DLL môžu obsahovať niekoľko vzhľadovo rozličných ikon (obr. 4.2) a každá z nich niekoľko podobných obrázkov (4.1), v rôznej veľkosti a farebnej hĺbke.



Obrázok 4.2 – Všetky ikony nachádzajúce sa v súbore Explorer.exe

V týchto spustiteľných súboroch nie sú však vložené priamo ICO súbory, ale formát, akým sú jednotlivé obrázky v ikonách uložené, je veľmi podobný [4] práve štruktúre ICO súboru. Zo štruktúry na obr. 4.3 je možné vidieť, že výška a šírka obrázku môže byť rôzna. Aj keď väčšina programov používa ikony štvorcového tvaru , nemusí to platiť vždy.



Obrázok 4.3 – Štruktúra ICO súboru. Obrázok prevzatý z [5]

4.3 Formát obrázkov

Tak ako prakticky všetko v oblasti počítačov, tak aj ikony prešli určitým vývojom, aby spĺňali nové požiadavky. Informácie o postupnom vývoji som čerpal z [5]. Prvé ikony používané v 16 bitových Windows využívali formát DDB (device-dependent bitmap). Ich problémom však bolo používanie na zariadeniach, ktoré disponovali iným spôsobom zobrazenia, než pre ktoré boli vytvorené. Preto boli nahradené formátom DIB (device-independent bitmap), ktorý je používaný dodnes. Na rozdiel od svojho predchodcu kóduje farby jednotlivých pixelov do farebnej palety RGB a nie priamo do farebnej palety konkrétneho zariadenia. Okrem formátu DIB sa nedávno začal používať ešte aj formát PNG (portable network graphics). Spôsob a parametre s akými sa tieto 2 formáty používajú u ikon najčastejšie rozoberiem bližšie:

- **DIB** – obrázok pre ikonu je reprezentovaný dvomi bitmapami. Jednou pre samotný obrázok s farebnou hĺbkou 4, 8, 24 alebo 32 bitov na pixel (zriedkavo aj jednobitový čiernobiely obrázok). Používajú sa najčastejšie bitmapy vo veľkostiach od 16 × 16 do 48 × 48 pixelov [3]. Druhá bitmapa je jednobitová a slúži ako maska. Tá určuje, ktoré z pixelov obrázku budú zobrazené a ktoré budú priehľadné,
- **PNG** – obrázok pre ikonu je reprezentovaný jednou bitmapou najčastejšie o veľkosti 256 × 256 pixelov. O farbe pixelu rozhoduje 32 bitov, 8 pre každú z farieb RGB a ďalších 8 pre alfa kanál rozhodujúci o tom, ako veľmi bude pixel priehľadný. Tento formát bol pridaný vzhľadom na narastajúce nároky na vzhľad ikon. Začal ho používať systém Windows Vista. Rovnaké parametre ikony dokáže poskytovať aj formát DIB , ale formát PNG má obrovskú výhodu v kompresii [6], vďaka ktorej je pamäťový priestor, ktorý ikona zaberá, podstatne menší.

4.4 Ikony a .NET Framework

Na prácu s ikonami poskytuje .NET Framework priamo triedu System.Drawing.Icon [7], ktorá umožňuje načítať do programu ICO súbor a potom s ním pracovať. Tak isto umožňuje previesť ikonu na bitmapu, avšak automaticky vyberie iba niektorý obrázok z ikony a neumožňuje vybrať obrázok iný. Podobné to je aj pri extrahovaní ikony so spustiteľného súboru. Na sprístupnenie všetkých obrázkov je teda nutné využiť možnosti, ktoré vytvoril Microsoft ešte pred nasadením .NET Frameworku. Avšak funkcie, ktoré priamo extrahujú ikony zasa nefungujú správne s ikonami, ktoré obsahujú obrázok vo formáte PNG, keďže Microsoft ich zavedením narušil dovtedy zaužívaný formát ICO súboru. Na internete však existujú voľne dostupné triedy (napr. [8]) postavené na .NET Frameworku, ktoré tieto problémy interne riešia a poskytujú možnosť získať všetky potrebné bitmapy naviac už prevedené do 32 bitovej farebnej hĺbky, kde 8 bitov je venovaných každej z farieb červená, zelená a modrá a posledných 8 bitov kanálu alfa, ktorý určuje priehľadnosť výslednej farby.

5 Porovnávanie ikon

5.1 Farebné modely

Prehľad základných farebných modelov podľa [9]:

- **RGB** – skladá sa z troch farebných zložiek. Základné farby sú červená, zelená a modrá. Jedná sa o aditívne miešanie farieb. To znamená, pri nulových zložkách všetkých troch základných farieb je výsledkom čierna farba a naopak pri maximálnych je výsledkom biela farba. Používa sa vo väčšine súčasných zobrazovacích zariadeniach. Sú na ňom založené obrazové formáty napr. PNG, BMP a mnohé iné. PNG formát používa rozšírený model RGBA, kde je pridaný kanál alfa pre priehľadnosť. Formát DIB používa model RGB, ale obsahuje aj informácie o priehľadnosti v podobe jednobitovej bitmapy.
- **CMY** – taktiež sa skladá sa z troch farebných zložiek, tento krát sú to farby azúrová, purpurová a žltá. Jedná sa o subtraktívne miešanie farieb, kde je základnou biela farba a postupným pridávaním jednotlivých zložiek vzniká čierna. Používa sa hlavne v tlačiarňach. Väčšinou býva ešte doplnená čiernou farbou (takýto model sa nazýva CMYK), kvôli šetreniu arieb.
- **HSV** – skladá sa z troch zložiek a to sú farebný tón, sýtosť a svetlosť. Farby ako biela, červená, modrá atď. sú dosahované pri maximálnej svetlosti. Je to užívateľsky orientovaný model, keďže umožňuje intuitívnu úpravu obrazu.
- **HLS** – podobne ako model HSV obsahuj farebný tón a sýtosť ale namiesto svetlosti obsahuje jas. Biela farba je dosiahnutá pri maximálnom jase. Farby ako červená, modrá atď. pri polovičnom. Takisto je to užívateľsky orientovaný model.

5.2 Alfa kanál

Obrázky z ikon, ktoré sa budú porovnávať využívajú priehľadnosť a nachádzajú sa vo formáte RGBA. Existujúce spôsoby na porovnávanie a klasifikáciu obrazu však s priehľadnosťou nepracujú, pretože zdroj obrazu je väčšinou fotografia. Naskytá sa teda otázka ako chápať a spracovávať alfa kanál.

5.2.1 Alfa kanál ako štvrtá farebná zložka

Je to najjednoduchšie riešenie, ale brať alfa kanál ako štvrtú farebnú zložku nezodpovedá příliš realite, pretože ostatné 3 farebné zložky sú alfa kanálom priamo ovplyvňované a dokonca nemusia byť vôbec zobrazené. V takom prípade by nemali byť do porovnávania uvažované.

5.2.2 Alfa kanál oddelený od farebných zložiek

Priehľadnosť určuje tvar obrázku. Preto každý pixel sa skladá z 3 zložiek určujúcich farbu a jednej zložky určujúcej tvar. Potom pri spracovávaní obrazu by sa spracovával oddelene tvar obrazu a samotný obraz. Takže napr. pri porovnávaní 2 obrazov by vyšli 2 výsledky podobnosi, jeden pre obrazy a jeden pre ich tvary. Opäť ale zostáva problém, že vo výsledku pre obrazy nie je nijakým spôsobom zohľadnený tvar viditeľnej oblasti.

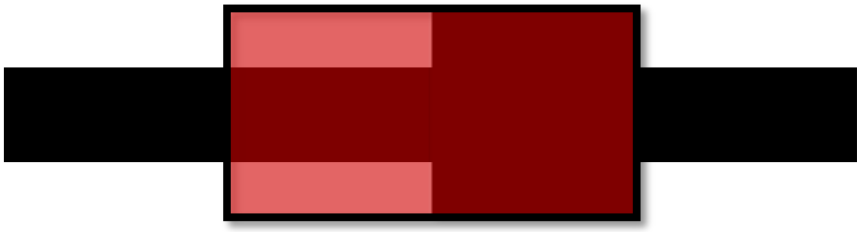
5.2.3 Alfa kanál ako váha pixela

Treťou možnosťou je brať alfa kanál ako váhu pixela. Alfa kanál určuje ako veľmi je daný pixel vidieť. Preto hodnota farebných zložiek by mala byť upravená váhou alfa kanála. Dosiahnuť toho je možné prepočítaním alfa kanála do rozmedzia od 0 do 1 a potom prenásobením každej z farebných zložiek. Takýmto spôsobom je priehľadnosť zapracovaná priamo do farebnej palety RGB. Tento prístup sa však taktiež neobíde bez problémov. Rôzne pixely v palete RGBA sa môžu stať zhodnými v palete RGB. Napr. čisto červený pixel s polovičnou priehľadnosťou nadobudne rovnakú hodnotu ako pixel, ktorý nie je priehľadný vôbec, ale hodnotu červenej zložky má presne v polovici. Pre budúce porovnávanie to však znamená jedine vyššiu mieru falošných poplachov, a detekciu obrázkov podobných to nijakým zásadným spôsobom neovplyvňuje.

Dôležitým faktorom je tiež počet pixelov v obrázkoch ikon, ktorých sa tento problém týka. Sú to ako pixeli, ktoré majú iba čiastočnú viditeľnosť, tak aj tie, ktoré sú buď úplne viditeľné alebo úplne neviditeľné.

U prvej skupiny čiastočne priehľadných pixelov nie je problém zásadný, pretože sa tieto pixely nachádzajú na obrázkoch väčšinou iba v malom množstve a slúžia ako vytieňovanie pre lepší vzhľad. Naviac si treba uvedomiť, že môžu pochádzať iba z obrázkov, ktoré boli pôvodne v 32bitovej farebnej hĺbke a tých sa v štandardnej ikone nachádza výrazne menej ako tých vo formáte DIB s menšou farebnou hĺbkou. A keďže v tomto formáte je priehľadnosť riešená pomocou jednobitovej bitmapy, pixely môžu byť buď iba úplne priehľadné alebo nepriehľadné.

U druhej skupiny môže byť problém podstatne zásadnejší, pretože priehľadné pixely určujú tvar ikony a nachádza sa ich v typickom obrázku veľké množstvo, takže za určitých okolností môžu výrazne ovplyvniť porovnávanie a určiť tak za podobné aj obrázky, ktoré podobnými nie sú. U aditívneho miešania farieb modelu RGB rozšíreného o alfa kanál, sú po prevedení do RGB modelu bez priehľadnosti a upravením farebných zložiek podľa váhy alfa kanála, považované za rovnaké pixely akýkoľvek priehľadný a nepriehľadný čierny. Nie nepriehľadný biely. Dôvodom je, že u aditívneho miešania farieb je základná farba čierna (všetky 3 farebné zložky majú hodnotu 0). Dobre to vidno na obrázku 5.1. Obdĺžnik je v skutočnosti rozdelený na 2 rovnaké štvorce. Pravý je nepriehľadný a má červenú zložku nastavenú na polovicu. Ľavý je z polovice priehľadný a červená zložka je nastavená na maximum. Avšak obdĺžnik by po položení na pás základnej farby, t.j. čiernej, vyzeral ako keby bol zložený z 2 rovnakých štvorcov. Po prepočítaní na model RGB bez alfa kanála, budú mať oba štvorce rovnakú hodnotu (polovica × celá = celá × polovica) aj keď pôvodne boli rozličné. To však zodpovedá pôvodnému obdĺžniku iba za predpokladu, že je umiestnený na čiernom pozadí.

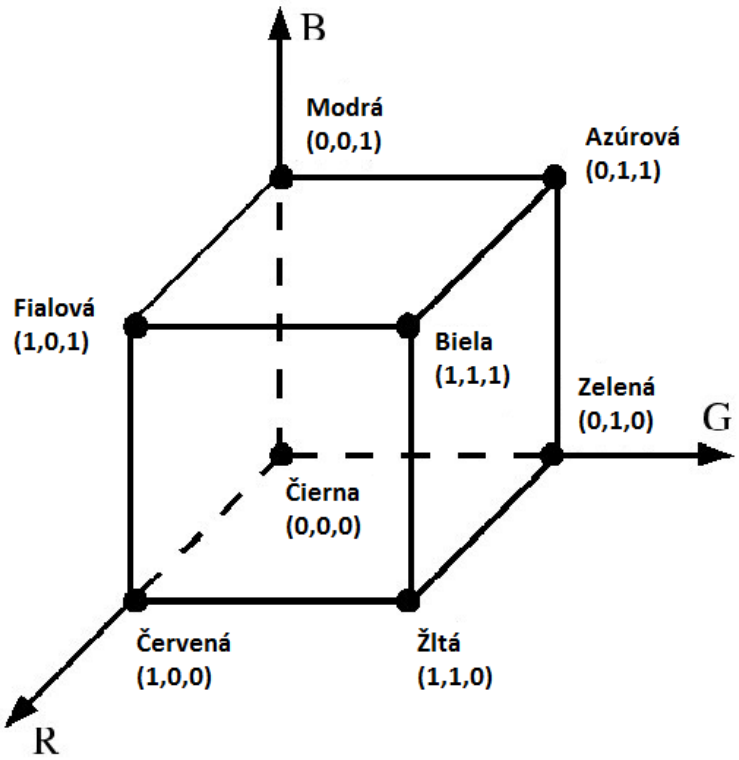


Obrázok 5.1 – Ukázka použita čiernej ako základnej farby u modelu RGB

Z toho vyplýva, že ak je obrázok umiestnený na čiernom pozadí, prevod z RGBA do RGB podľa váhy alfa kanála je absolútne korektný.

Na akom pozadí sa ikona nachádza ale samozrejme záleží na užívateľovi. V systémoch Windows sa ikona reálne objavuje na čiernom pozadí iba na pracovnej ploche, pokiaľ je na nej tmavá tapeta. Takže vo väčšine prípadov by prevod z RGBA do RGB nebol úplne korektný. Ikony v systémoch Windows sa reálne nachádzajú veľmi často na bielom pozadí. Môžu za to súborové manažéry (napr. Explorer, Total Commander). Takže ak by mal farebný model základnú farbu bielu, tak by bol prevod z RGBA do RGB zväčša korektný.

Na porovnávanie by teda bolo vhodné previesť obrázky z farebného modelu RGBA do modelu, ktorý by mal základnú farbu bielu. Avšak prevod musí byť taktiež dostatočne jednoduchý, aby výkonové nároky stúpli čo najmenej, pretože už samotné porovnávanie obrazu je výkonovo dosť náročné. Najlepšie tieto kritériá spĺňa farebný model CMY obohatený o alfa kanál. CMY je ako vidieť na obrázku 5.2 doplnkovým modelom k modelu RGB.



Obrázok 5.2 – RGB a CMY sú navzájom doplnkové modely [9]

Prevod je teda relatívne jednoduchý. Rovnice prevodu vyzerajú následovne:

$$azúrová = 1 - červená$$

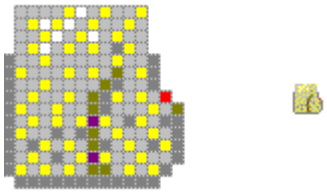
$$fialová = 1 - zelená$$

$$žltá = 1 - modrá$$

Hodnota alfa kanála zostane zachovaná v pôvodnej podobe. Tento bude zakomponovaný do CMY modelu takisto ako u modelu RGB. Po týchto prevodoch má pôvodne priehľadný pixel rovnakú hodnotu ako pôvodne nepriehľadný pixel bielej farby.

5.3 Porovnávanie obrazov

Porovnávanie obrázkov nachádzajúcich sa v ikonách sa vymyká klasickému porovnávaniu obrazu, pretože to sa zaoberá hlavne obrazmi reálneho života, ktoré nie sú vytvorené pomocou počítača ale pomocou nejakého zariadenia snímajúceho obraz. Dôvody sú zrejmé. Rozpoznávanie reálneho obrazu umožňuje napr. ovládanie elektronických zariadení pomocou rôznych gest, triedenie fotografií podľa objektov a ľudí čo sa na nich nachádzajú atď. Snímky reálneho sveta sú však veľmi problematické, pretože ich kvalita je veľmi rôzna a priamo záleží od konkrétno snímacieho zariadenia, jeho pozície pri snímaní a takisto aktuálnych svetelných podmienok. Tieto atribúty spôsobujú problémy, s ktorými sa musia porovnávacie metódy vyrovnať a to napr. rôzna svetlosť snímok, šum, rôzna veľkosť tých istých objektov, ich horizontálne a vertikálne posunutie a uhol, z ktorého sú zobrazené. Z tohto pohľadu je porovnávanie obrázkov z ikon podstatne zjednodušené. Zo spomínaných problémov ich zasahuje iba relatívne malé vzájomné horizontálne a vertikálne posunutie. Na druhej strane ikony obsahujú aj obrázky extrémne malých rozmerov a malej farebnej hĺbky, s ktorými sa treba taktiež vysporiadať.



Obrázok 5.3 – Vpravo je ikona pre adresár rozmerov 16 × 16 a 4bitovej farebnej hĺbky. Vľavo je tá istá zväčšená

Tieto rozdiely medzi reálnymi snímkami a obrázkami v ikonách je treba zohľadniť pri výbere správnej porovnávacej metódy. Tie by som rozdelil do 3 skupín a rozobral ich každé zvlášť v podkapitolách.

5.3.1 Metódy porovnávajúce pixely

Tieto metódy sú založené na porovnávaní priamo pixelov medzi sebou. Porovnávaný môže byť buď celý obraz alebo iba jeho určité časti [10]. V prípade porovnávania celých obrazov je dôležité, aby mali rovnaké rozmery. Hodnota ich vzájomnej korelácie sa vypočítava na základe rozdielov medzi všetkými zodpovedajúcimi si pixelmi. Dostatočne presné výsledky pri nižšej výpočtovej náročnosti je

však možné dosiahnuť aj keď nebudú porovnávané všetky pixely, ale iba určitá časť z ich celkového počtu. Toto platí za predpokladu, že porovnávané pixely sú rozmiestnené rovnomerne po obraze. Takýmto spôsobom prichádza vlastne k rozdeleniu obrazu na malé oblasti, ktoré sú zastupované jedným pixelom nachádzajúcim sa na dopredu určenej pozícii v oblasti. To môže vytvárať nepresnosť v korelácii, keďže pixel reprezentujúci oblasť môže byť výrazne odlišný ako ostatné, ktoré sa v danej oblasti nachádzajú tiež. Zabrániť by sa tomu dalo, ak by pixel reprezentujúci oblasť nebol dopredu určený ale bol by vyberaný tak, aby čo najviac zodpovedal celkovému zloženiu oblasti. Treba však počítať so zvýšenými výkonovými nárokmi, oproti výberu pixelu dopredu určeného. Konkrétne pri porovnávaní obrázkov z ikon by mohlo mať zmysel porovnávať iba časť obrazu. Dôvod je jednoduchý. Väčšina ikon nevyužíva celý svoj rozmer a obrázky sú na okrajoch veľmi často priehľadné. Takýmto spôsobom môže opäť prísť k zníženiu výkonových nárokov, treba ale počítať so zvýšením chybovosti metódy porovnávania.

5.3.2 Metódy získavajúce informácie o obrazoch

Z obrazov je vypočítaný ich histogram. Ten vyjadruje počet výskytov každej zo sledovaných hodnôt, prípadne môžu byť hodnoty rozdelené do intervalov. V prípade porovnávania obrazu sa naskytá možnosť vytvoriť farebný histogram, takže výsledkom bude počet jednotlivých farebných odtieňov. Takto sa dajú získať užitočné informácie o skúmanom obraze, ktoré ale môžu byť tvorené podstatne menším množstvom dát. Napr. obrázok štvorcového tvaru so šírkou 64 pixelov, kde každý pixel má jednu z 256 možných farieb a teda každý pixel zaberá 1 bajt. Veľkosť obrázku je potom 4096 bajtov. ($64 \times 64 \times 1$) Maximálny počet pixelov jednej farby je 4096, takže pre každú jednu z 256 farieb je treba vymedziť 2 bajty, na ktorých je uložený počet danej farby. Výsledné pamäťové miesto je teda 512 bajtov, čo je 8 krát menej ako pôvodný obrázok. Hlavnou výhodou však je, že pri následnom spracovaní histogramu stačí prejsť iba cez 256 farieb, narozdiel od 4096 pixelov pri spracovávaní celého obrázku. Nevýhodou však je, že vytvorenie histogramu samo o sebe je vlastne spracovanie obrázku, takže treba predsa len prejsť cez všetky pixely a až potom nejakým spôsobom spracovať histogram. A treba ešte spomenúť dôležitý fakt, že farebný histogram síce nesie presnú informáciu o počte konkrétnych farieb, ale neobsahuje informáciu o ich rozmiestnení v pôvodnom obraze. Preto obrázky, obsahujúce rovnaké farby budú označené za zhodné aj keď predstavujú úplne niečo iné. Po vytvorení histogramov z obidvoch porovnávaných obrázkov je zistenie ich korelácie jednoduché. Stačí zistiť rozdiely v počte jednotlivých farieb.

Za zmienku ešte stojí spomenúť, že histogram nemusí vypovedať iba o farbách, ktoré sa na obrázkoch nachádzajú. Dôležitou informáciou môže byť aj rýchlosť zmeny farby medzi susednými pixelmi. Táto rýchlosť zmeny sa nazýva gradient, ktorý môže byť aj orientovaný [11] a ak by nebol obraz definovaný diskrétnymi hodnotami, ale spojitou funkciou, jednalo by sa matematicky jednoducho o deriváciu v danom bode. Histogram gradientov vypovedá presne o tom, koľko ako veľkých farebných prechodov sa v obraze nachádza, ale opäť neobsahuje informáciu o ich umiestnení.

5.3.3 Metódy detekujúce významné body v obraze

Tu som čerpal z [12]. Významné body sú väčšinou na hranách objektov, ktoré sa na obraze nachádzajú. Malo by sa jednať o také body, ktoré sú stabilné voči rôznym problémom pri snímaní

obrazu spomínaným v úvode kapitoli 5.3. Dôležité je aby boli dobre identifikovateľné ich okolím, keďže z neho je vytváraný či už jednoduchý alebo zložitejší deskriptor. Po nájdení významných bodov v obidvoch porovnávaných obrazoch a určení ich deskriptorov, sú tieto medzi sebou navzájom porovnávané. Výpočtová náročnosť je zo všetkých metód najvyššia, keďže už na nájdenie významných bodov je nevyhnutné prejsť po všetkých pixeloch v obraze. Používané sú algoritmy SIFT, SURF, MSER a iné.

5.3.4 Zhodnotenie metód s ohľadom na porovnávanie ikon

- **Obrázky z ikon môžu mať veľmi malé rozmery a malú farebnú hĺbku** – toto nerobí žiadny problém metódam porovnávajúcim priamo pixely metodám získavajúcim informácie z obrazu. Metódy hľadajúce významné body by mali problém s ich hľadaním v takýchto obrázkoch (viď obr. 5.3) a takisto s vytváraním ich deskriptorov. Preto na tento účel sú nevhodné.
- **Obrázky sú rovnaké alebo vzájomne mierne posunuté** – použiť sa dajú všetky typy metód aj keď metódy hľadajúce významné body sú výrazne robustnejšie.
- **Obrázky sú rovnaké a vzájomne viacej posunuté** – v takomto prípade sú metódy porovnávajúce priamo pixely menej úspešné.
- **Obrázky nie sú rovnaké, iba sa na seba podobajú** – úspešnosť všetkých metód klesá.
- **Bude dopredu vytvorená porovnávacia množina o malom počte ikon** – metódy získavajúce informácie a metódy hľadajúce významné body môžu pri pridávaní vzorových ikon vytvoriť histogramy respektíve deskriptory jednotlivých obrázkov, ktoré budú trvalo uložené a bude treba ich vytvoriť len pre obrázky ikony testovaného súboru. Potom sa už budú vzájomne porovnávať iba histogrami a deskriptory. U metód porovnávajúcich pixely bude nutné stále vzájomne porovnávať celé obrázky. Ak by bol počet vzorových ikon veľký, malo by to výrazný dopad na výpočtový výkon.

Ak to teda celé zhrniem, metódy hľadajúce významné body sú nevhodné z dôvodu malých obrázkov s malou farebnou hĺbkou nachádzajúcich sa v ikone. Metódy porovnávajúce priamo pixely strácajú výkonovo na metódy získavajúce informácie, čo by sa výrazne prejavilo pri veľkom počte vzorových ikon, ktorý je ale nepravdepodobný, pretože nachádzať by sa v porovnávacej množine mali iba ikony, ktoré sú notoricky známe bežným užívateľom. Metódy získavajúce informácie sú na tom lepšie aj pri rovnakých obrázkoch, ktoré sú navzájom výraznejšie posunuté. Ale ich už vyššie spomínanou nevýhodou je, že vôbec neberú do úvahy pozície jednotlivých pixelov, ale iba ich štatisticky spracovávajú, čo môže viesť k vysokému počtu falošných detekcií. Ako najvhodnejšie by som teda v tomto konkrétnom prípade porovnávania známych ikon volil metódy porovnávajúce priamo pixely.

6 Návrh aplikácie

Táto aplikácia bude porovnávať ikonu konkrétneho súboru s dobre známymi ikonami používanými v systéme Windows. Jej úlohou bude odhaliť testované súbory, ktoré používajú priamo niektorú zo vzorkových ikon alebo jej upravenú verziu. Bude sa jednať o konzolovú aplikáciu, ktorá dostane ako parameter príkazového riadku názov testovaného súboru a na štandardný výstup vypíše v prípade zhody alebo podobnosti s niektorou vzorkovou ikonou jej názov a v percentách úroveň podobnosti. Ďalej by mal program umožňovať pridanie nových vzorkových ikon, vypísať zoznam všetkých vzorkových ikon a takisto možnosť odstrániť niektorú z nich.

Aplikácia bude postavená na trojvrstvovej architektúre, teda zložená z vrstvy prezentačnej starajúcej sa o spracovanie argumentov príkazovej riadky a výpis výsledkov do konzoly, vrstvy aplikačnej, ktorá bude umožňovať výber konkrétnych obrázkov a ich porovnanie a vrstvy dátovej, ktorá zaobaľuje extrakciu ikon so spustiteľných súborov a pridávanie a odoberanie vzorkových ikon z porovnávacej množiny.

Ďalej by som chcel využiť možnosti súčasných viacjadrových procesorov. Testovaný súbor bude porovnávaný s každou vzorovou ikonou vo vlastnom vlákne, čo by malo na týchto procesoroch priniesť výrazné urýchlenie behu aplikácie.

6.1 Spôsob uloženia vzorkových ikon

Po vytvorení skupiny vzorkových ikon, bude táto skupina viac menej stála a nebude veľmi rozsiahla, pretože sa bude jednať o niekoľko ikon veľmi často sa vyskytujúcich v systémoch Windows. Z tohto dôvodu som vylúčil možnosť použitia databázy na ich spravovanie. Za plne dostačujúce považujem vytvorenie jedného priečinku, v ktorom budú uložené jednotlivé ICO súbory. Ako druhá možnosť sa mi javí vytvorenie jediného súboru, ktorý by obsahoval všetky ikony, takže vo výsledku by sa manipulovalo iba s jediným súborom. Naviac pre samotné porovnávanie ikon by stačil iba jediný priechod súborom. Na druhú stranu, ale uloženie, každej ikony zvlášť do špeciálneho priečinku umožňuje prehľadnejšie zobraziť ikony, s ktorými program pracuje. Program bude umožňovať vypísanie zoznamu vzorkových ikon na štandardný výstup, ale práve vďaka tomu, že sa jedná o ikony, je obrazová informácia viac vypovedajúca. Takže ak bude každá ikona vo vlastnom ICO súbore, pri prezeraní adresára pomocou grafického prostredia Windows, bude automaticky každá priamo zobrazená. Cesta k adresáru bude nastaviteľná úpravou v konfiguračnom súbore aplikácie. Naviac bude aplikácia umožňovať predaním argumentu pri spustení zvoliť iný adresár, s ktorým sa bude pracovať pri tomto jednom behu aplikácie.

6.2 Výber konkrétneho obrázku zo sady na porovnávanie

Jedna aplikácia môže mať niekoľko ikon a tie obsahujú štandardne niekoľko obrázkov rôznych rozmerov a farebnej hĺbky. Naviac žiadny z rozmerov nie je povinný, preto nemôžem počítať s jednou konkrétnou veľkosťou, keďže nie vždy bude obsiahnutá v sade ikon. Treba zobrať do úvahy aj možnosť, že tvorca malware vloží do programu niekoľko vzhľadovo rozličných ikon a každá z nich obsahuje niekoľko obrázkov rôznych rozmerov a farebnej hĺbky, takže je nevyhnutné skontrolovať všetky tieto ikony. Dokonca sa môže stať, že obrázky u jednej konkrétnej ikony budú vzhľadovo úplne iné, takže je vhodné nevyberať ku kontrole iba niektorý z nich, ale skontrolovať úplne všetky. Najprv si teda program pripraví všetky obrázky z testovaného súboru a potom bude načítavať po jednej vzorkové ikony. Pre každý jeden obrázok extrahovaný z testovaného súboru sa potom zo vzorkovej ikony vyberie obrázok, ktorý bude mať rovnaké rozmery, alebo ak by sa taký nenašiel bude vybraný ten, ktorý má rozmery čo možno najbližšie. Ak by bolo nájdených obrázkov viac, bude z nich vybraný ten, ktorého farebná hĺbka najlepšie zodpovedá farebnej hĺbke obrázku z testovaného súboru.

6.3 Porovnávanie obrázkov

Porovnávanie obrázkov je skomplikované tromi faktormi:

- **Obrázky môžu mať iné rozmery** – budú prevedené na rozmery rovnaké. Treba zobrať do úvahy aj možnosť, že pomer šírok a výšok obrázkov nemusí byť rovnaký.
- **Obrázky môžu byť uložené každý v inom formáte** – jednobitová bitmapa, ktorá vo formáte DIB rozhoduje o priehľadnosti bude prevedená na alfa kanál a obrázok bude zložený z farebnej palety RGBA namiesto RGB a tým bude zodpovedať farebnej palete obrázku uloženého vo formáte PNG. Prevod je jednoduchý, každý pixel bude mať nastavenú priehľadnosť buď na maximálnu alebo minimálnu hodnotu.
- **Obrázky môžu mať inú farebnú hĺbku** – všetky obrázky budú prevedené na 32bitovú farebnú hĺbku.

Po prevedení obrázkov na vzájomne zodpovedajúce parametre príde k porovnaniu. Porovnávať sa budú medzi sebou priamo zodpovedajúce pixely. V cykle sa bude prechádzať postupne po jednotlivých pixeloch jedného obrázku a bude sa zisťovať rozdiel medzi zodpovedajúcimi pixelmi druhého obrázku. Tento rozdiel sa spočíta a napokon vydelí počtom pixelov. Po ďalšom vydelení počtom možných farebných odtieňov sa hodnota dostáva do rozmedzia od 0 do 1 kde 0 znamená maximálnu podobnosť a 1 minimálnu. Potom ešte treba previesť túto hodnotu na percentá tak, aby maximálna podobnosť bola 100%.

$$podobnosť\ v\ percentách = \left(1 - \frac{\sum_{i=0}^{šírka}\sum_{j=0}^{výška} abs\left(pixel1_{i,j} - pixel2_{i,j}\right)}{šírka * výška * počet\ farebných\ odtieňov}\right) * 100 \qquad [6.1]$$

Pri iterácii budú pixely pred ich vzájomným odčítaním prevádzané z farebnej palety RGBA do palety CMYA, ktorá sa javí ako mierne vhodnejšia v danom prípade. Na alfa kanál sa budem pozerať ako na váhu jednotlivých pixelov a paleta CMYA bude teda prevedená na paletu CMY . Rozdiel medzi pixelmi budem počítať pre každú farebnú zložku zvlášť a potom z nich bude vytvorený priemerný rozdiel, ktorý bude dosadený do rovnice [6.1].

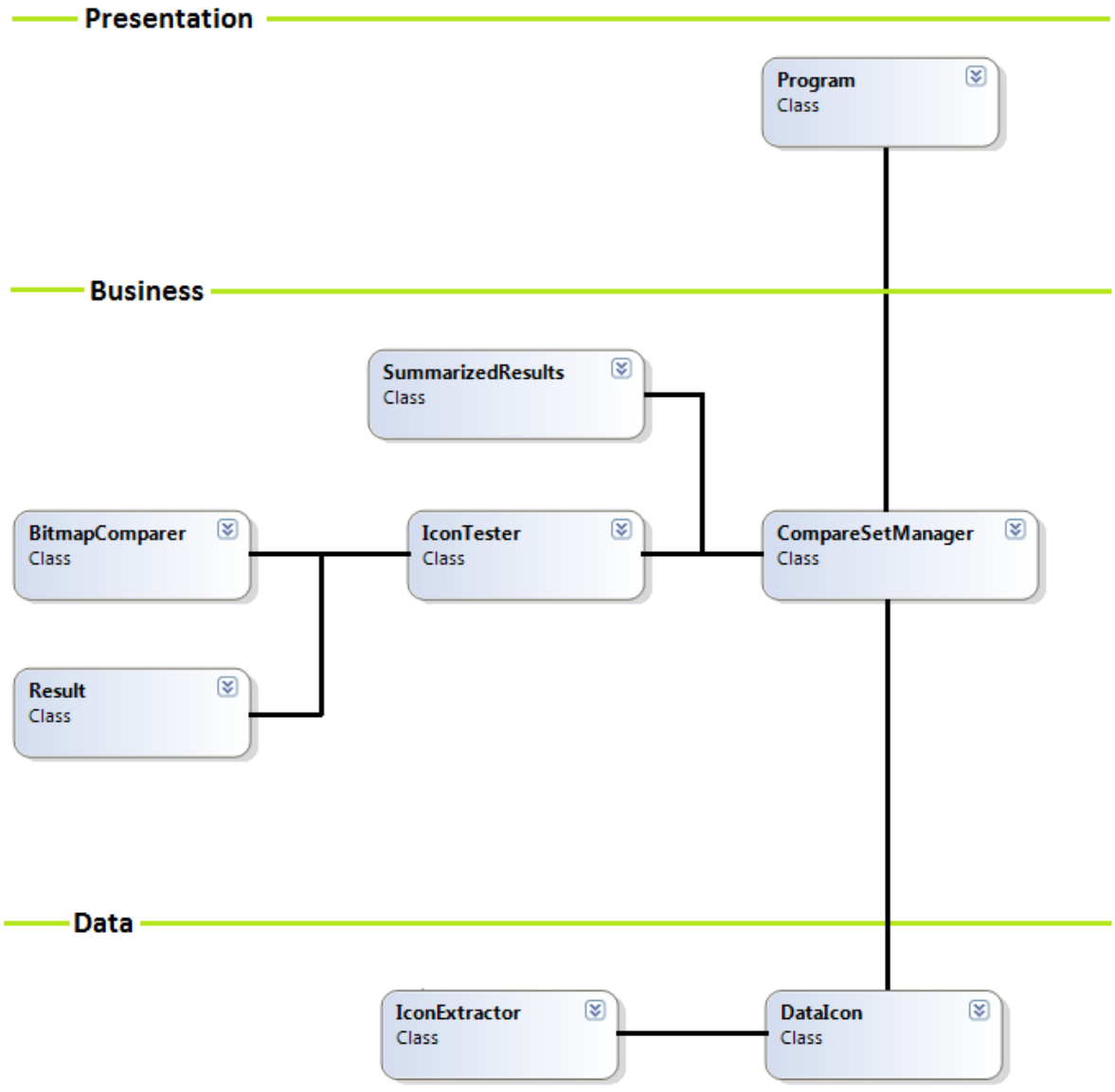
6.4 Implementačné technológie

Pri voľbe implementačných technológií som nemusel brať ohľad na platformovú nezávislosť. Väčšina tvorcov škodlivého kódu sa sústreďuje na operačné systémy Windows ako jednoznačného lídra medzi používateľmi počítačov a to isté samozrejme platí aj pre antivírusové spoločnosti, ktoré proti nim bojujú. Množstvo malware vytváraných pre ostatné systémy oproti systémom Windows je zanedbateľný. O tom svedčí aj samotný fakt, že antivírusové programy sa v týchto systémoch používajú iba minimálne.

Prakticky celý vývoj aplikácie sa točí okolo ikon. Namiesto vytvárania vlastných metód na prácu s ikonami, ako je napr. extrahovanie ikony so súboru, budem používať už existujúce voľne dostupné metódy a triedy, založené na .NET Frameworku, ktoré dokážu extrahovať ikony zo spustiteľných súborov a pracovať s nimi. Aplikáciu by som chcel postaviť na čisto objektovo orientovaných základoch a preto som sa rozhodol pre programovací jazyk C#.

7 Implementácia

Najskôr som vytvoril aplikáciu bez možnosti samotného porovnávania ikon. To znamená, že pomocou aplikácie som dokázal vytvoriť porovnávaciu množinu ikon so spustiteľných súborov a odoberať ikony. Užívateľské rozhranie ďalej umožňovalo výpis ikon v množine a výpis nápovedy. Až keď bola aplikácia v tomto smere kompletne vytvorená a otestovaná začal som implementovať samotný výber vhodných obrázkov k vzájomnému porovnávaniu a potom až porovnávanie. Najlepší pohľad na to ako aplikácia funguje umožňuje diagram tried a ich krátky popis.



Obrázok 7.1 – Diagram tried mojej aplikácie

Program – Trieda spracováva argumenty príkazového riadku a určuje aká akcia sa má vykonať. Po vykonaní informuje užívateľa o jej úspešnosti a výsledku.

CompareSetManager – Poskytuje rozhranie biznis vrstvy vrstve prezentačnej.

Icontester – Postupne prechádza všetkými ikonami z porovnávacej množiny a všetkými obrázkami nachádzajúcimi sa v ikone testovaného súboru. Pre každý jeden obrázok z ikony testovaného súboru vyberá čo najviac zodpovedajúci obrázok z hľadiska rozmerov a farebnej hĺbky z každej ikony z porovnávacej množiny.

BitmapComparer – Táto trieda sa stará o samotné porovnávanie 2 bitmáp.

Result – Pre každé jedno porovnanie 2 obrázkov sa vytvorí inštancia tejto triedy a zapíšu sa informácie o prebehnutom porovnaní: výsledok v percentách, rozmery a farebné hĺbky porovnávaných obrázkov a názov ikony z porovnávacej množiny, z ktorej pochádza jeden porovnávaný obrázok (druhý pochádza vždy z ikony testovaného súboru). Taktiež bude uvedené poradové číslo obrázku z testovaného súboru.

SummarizedResults – Vyberá, ktorý z prednastavených prahov bude použitý s ohľadom na výsledky porovnávania. Takisto spracováva všetky výsledky porovnávaní a vyberá z nich iba tie, ktoré sú nad nastaveným prahom a tie budú nakoniec zobrazené.

DataIcon – Poskytuje rozhranie dátovej vrstvy biznis vrstve. Ďalej umožňuje pridávanie ikon do porovnávacej množiny, a tak isto aj ich odstránenie a vytvorenie ich zoznamu.

IconExtractor – Trieda extrahuje ikony zo spustiteľných súborov a naviac dokáže rozdeliť ikonu obsahujúcu napr. 10 obrázkov na 10 ikon, ktoré obsahujú zhodne iba po jednom obrázku.

7.1 Porovnávanie 2 obrazov

Počas implementácie porovnávania som sa rozhodol neprevádzať obrázky na rovnaké rozmery a tým ušetriť výrazne výkonové nároky. Namiesto toho aplikácia zisťuje pomer ich strán. Pri iterácii po jednotlivých pixeloch menšieho obrázku, dochádza iba k prepočítavaniu aktuálnej pozície na pozíciu vo väčšom z nich s ohľadom práve na vzájomný pomer ich strán.

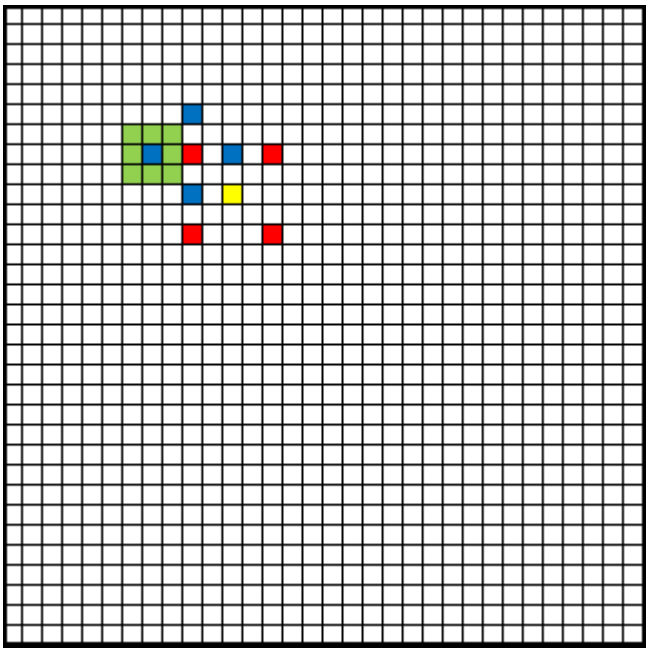
Pri následnom testovaní sa potvrdil môj predpoklad o príliš vysokých nárokoch pri porovnávaní obrázkov o veľkosti 256×256. Preto som sa rozhodol určiť maximálny počet porovnávaných pixelov na 4096 (64×64). Väčšie obrazy sú rozdelené na oblasti, z ktorých sa porovnáva iba jediný pixel. Tento pixel je napevno určený ako ľavý horný v danej oblasti a nemusí teda zodpovedať jej celkovému zloženiu. Výsledky porovnávaní sa však prakticky nezmenili. Zrýchlenie aplikácie ale bolo výrazné.

Potom som sa zameral na samotnú relevantnosť výsledkov. Ako dobrá testovacia množina sa ukázali ikony pre adresár, ktorých sa v systémoch Windows nachádza hneď niekoľko. Vzhľad nie je vždy rovnaký, obrázky sú medzi sebou často menej či viac posunuté. Výsledky testovania ukázali, že podobné obrázky sú ohodnotené od 75% až po 100% pri obrázkoch totožných. Po analýze výsledkov sa ako zásadný dôvod, prečo veľmi podobné obrázky získavajú iba 75%, ukázalo ich vzájomné posunutie.

Vzájomné posunutie totožných obrázkov o 1/16 jeho rozmeru prinášalo výsledky okolo 83%. Po pridaní väčšieho počtu testovaných ikon sa ukázalo, že nad túto hodnotu sa dostávajú aj porovnávania obrázkov, ktoré spolu priamo nesúvisia, iba majú podobný tvar a majú podobnú prevažujúcu farbu. Preto som sa rozhodol upresňovať vzájomnú polohu medzi porovnávanými obrázkami, ktorých základná korelácia sa dostala nad určitú úroveň. Postupným testovaním som tento prah stanovil na 70%. Keďže je uložený v konfiguračnom súbore (takisto aj ostatné prahy) je možné ho zmeniť aj bez prekompilovania kódu.

Na upresňovanie polohy som vytvoril niekoľko porovnávacích metód, ktoré majú spoločné, že zisťujú smer posunutia, pre ktorý sú dosahované najlepšie výsledky. Samotné porovnávanie funguje stále na rovnakom princípe ale prichádza k porovnávaniu nie priamo zodpovedajúcich si pixelov, ale mierne posunutých. Ako základný krok posunutia som zvolil 1/16 rozmeru obrázkov. Táto hodnota je vybraná s ohľadom na najmenšie používané obrázky v ikonách, ktoré majú rozmery 16 × 16 pixelov a teda krok posunutia zodpovedá presne jednému pixelu. Naviac všetky štandardné ikony sú násobkom čísla 16 a krok teda vychádza vo väčšine prípadov na celé číslo.

Prvá metóda skúma obrázky posunuté medzi sebou o krok diagonálne. Jedná sa o vektory posunutia (krok, krok), (krok, -krok), (-krok, krok), (-krok, -krok) a zrealizované teda budú 4 porovnávania tých istých obrazov, vždy rôznym diagonálnym smerom. Najlepšie pochopiteľné je to z obrázku. Potom je najlepší dosiahnutý výsledok uložený a takisto je nastavený aj najlepší vektor posunutia. Na obrázku je možné vidieť presný postup. Žltý pixel je referenčný. Pri tejto metódy bude porovnávaný s červenými pixelmi nachádzajúcimi sa na obrázku 7.2.



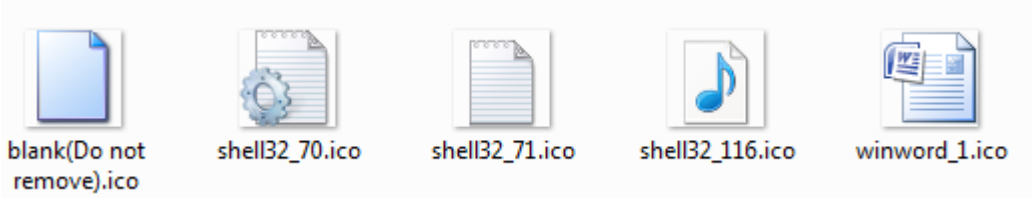
Obrázok 7.2 – Postup upresňovania na bitmape rozmerov 32 × 32

Ďalšia metóda vychádza z nového najlepšieho posunutia a pokračuje v posúvaní o krok horizontálne a vertikálne. Na obrázku 7.2 modré pixely.

Nakoniec prichádza k preskúmaniu najbližšieho okolia. Na obrázku zelené pixely. Krok sa zmenšuje na polovicu, až dokiaľ sa nerovná jednému pixelu. Stále platí, že sa vždy pokračuje s obrázkami posunutými o najlepší vektor posunutia. Maximálne celkové posunutie, ktoré je tento algoritmus schopný dosiahnuť je až 3/16 rozmeru obrázku jedným smerom (ďalšie 3/16 opačným smerom). Záleží však aj na konkrétnom rozmere obrázku, keďže pri počítaní kroku môže dochádzať k zaokrúhľovaniu na celé číslo. Toto posunutie už je dosť výrazné a v takomto prípade záleží skôr na úplne prvom porovnaní takto veľmi posunutých obrázkov, či sa výsledok dostane vôbec cez nastavený prah k podrobnejšiemu skúmaniu. Treba si taktiež uvedomiť, že pri posúvaní pozícií obrázkov, nemôžu byť porovnávané celé, ale iba ich spoločné výrezy. To má za následok ignorovanie okrajových častí obrázkov, ktoré by teoreticky mohli byť rozdielne. Ak sa k tomu pripočíta ešte posunutie, je však pravdepodobné, že obrázky nedosiahnu pri prvom porovnaní hodnotu prahu a tak nepríde k bližšiemu skúmaniu. V každom prípade výsledky pri postupnom posúvaní obrázkov zodpovedajú iba ich porovnávaným výrezom.

7.2 Problematické ikony

Počas ďalšieho testovania som narazil na ikony, ktoré vykazovali vysokú podobnosť aj keď užívateľ by ich za podobné neoznačil. Jedná sa o ikony, ktoré reálne využívajú iba malú časť z plochy ikony. V prvom rade sú to dokumenty rôznych aplikácií. Ako vidieť na obrázku 7.3, všetky tieto ikony majú rovnaký základ.



Obrázok 7.3 – niektoré z ikon využívajúcich rovnaký základ

Druhou skupinou sú ikony, ktoré sú takmer celé priehľadné. Sú to väčšinou iba pomocné ikony, ktoré nikdy ako hlavná ikona aplikácie zobrazené nebudú a používané bývajú až priamo samotnou aplikáciou. Sú to napr. ikony explorer_11.ico a explorer_12.ico nachádzajúce sa na obrázku 4.2. Ako riešenie tohto problémov som sa rozhodol pridať do porovnávacej množiny 2 ikony. Jednou je ikona neznámeho súboru (blank.ico na obr. 7.3) , ktorá koreluje s ikonami rôznych dokumentov a druhá je čisto priehľadná a tá koreluje s druhým typom problémových ikon. Ak budú obsiahnuté v porovnávacej množine umožnia pri spracovaní výsledkov nastaviť pre problematické ikony vyšší prah pri spracovaní výsledkov.

Problémovými sa ukázali celkovo obrázky, ktoré majú farebný základ biely. Keďže používam farebným model CMY so zakódovanou priehľadnosťou, pixel s pôvodne bielou farbou, má rovnakú hodnotu ako pixel celkom priehľadný, u týchto ikon sa stráca vlastne informácia o tvare. Preto som sa rozhodol neprevádzať obrázky do modelu CMY ale nechať ich v RGB modeli. Problém sa tým samozrejme nevyriešil, ale presunul sa na obrázky, ktoré majú farebný základ čierny a tých sa objavuje v systéme Windows výrazne menej.

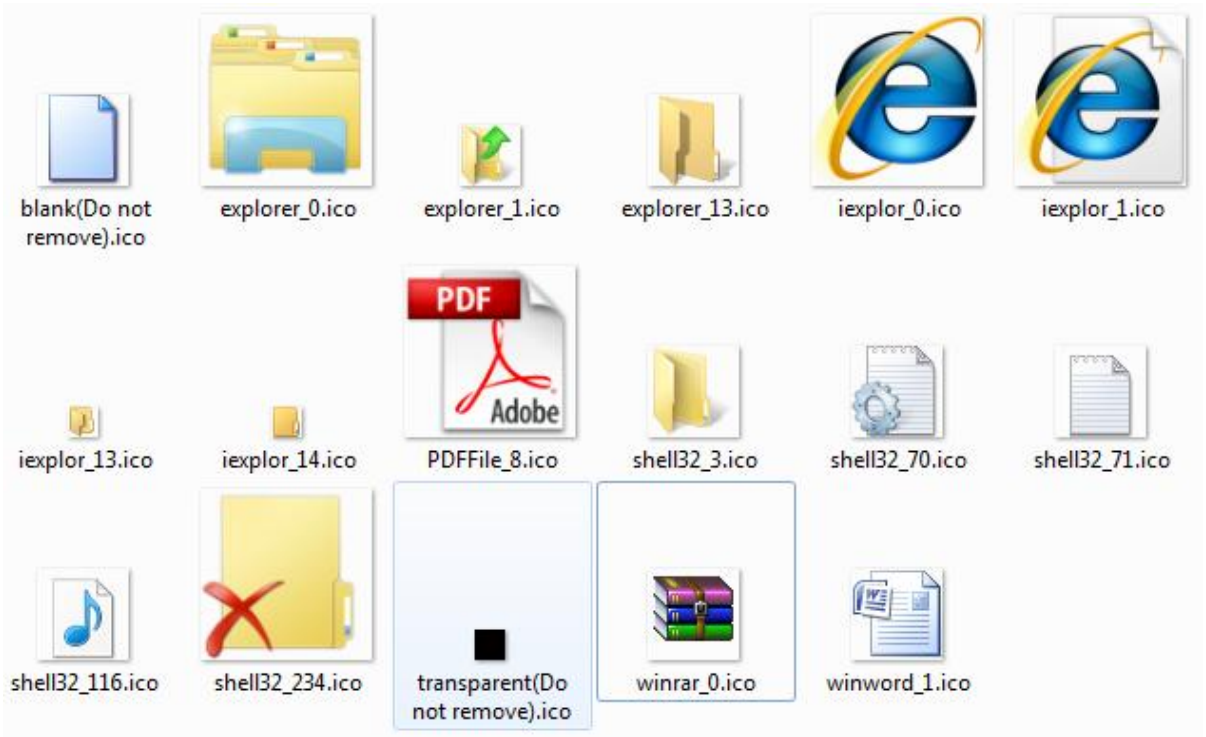
7.3 Spracovanie Výsledkov

Vypísanie výsledkov všetkých navzájom porovnávaných obrázkov, ktoré sa dostali nad nastavený prah znamená, že pre jednu vzorovú ikonu by bolo vypísaných niekoľko výsledkov. Aplikácia však má vypísať iba výslednú hodnotu a názov vzorovej ikony, s ktorou je testovaný súbor podobný. Preto je nevyhnutné, aby z výsledkov všetkých obrázkov, ktoré do danej ikony patria, bol vybraný iba ten najväčší dosiahnutý.

Predtým je však ešte treba vysporiadať sa z problematickými ikonami. Najskôr sa iteráciou cez všetky výsledky porovnávaní obrázkov, zistia všetky obrázky, ktoré vykazujú zvýšenú koreláciu s ikonou na obr. a ikonou priehľadnou. Pre všetky tieto obrázky bude potom nastavený prísnejší prah vo všetkých ich výsledkoch.

7.4 Výber ikon do porovnávacej množiny

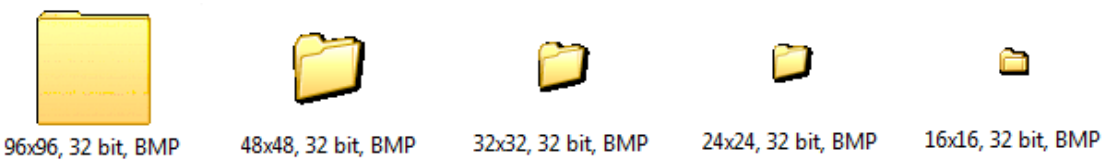
Vybrať som sa snažil ikony, ktoré poznajú prakticky všetci užívatelia systémov Windows, pretože práve tie sú pre tvorcov malware najzaujímavejšie. V prvom rade som pozbieral všetky ikony adresáru, ktoré sa v systéme Windows zobrazujú. Ďalej je to ikona Internet Explorera. Pridal som taktiež ikonu programu Winrar, keďže obrovské množstvo súborov sťahovaných z internetu sú práve archívy. Nakoniec som vložil rôzne ikony pre dokumenty často používaných typov.



Obrázok 7.4 – Ikony v testovacej množine. Ikona čisto priehľadná je systémom zobrazovaná ako čierna

7.5 Zhodnotenie výsledkov

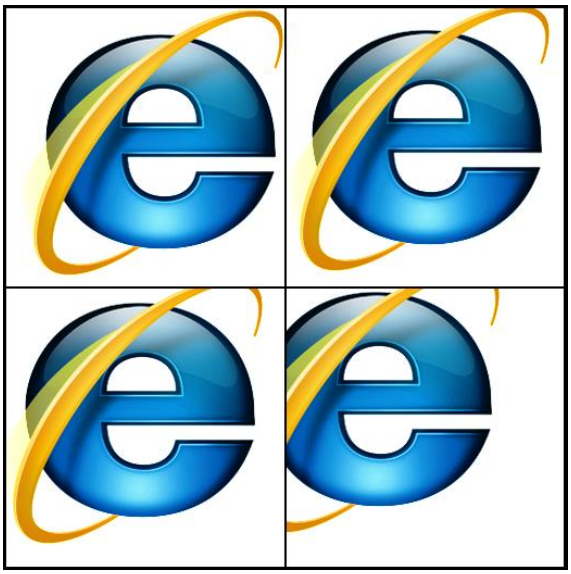
Po dokončení aplikácie, vyladení, nastavení všetkých prahov a vytvorení porovnávacej množiny som pristúpil k finálnemu testovaniu na asi 25 spustiteľných súboroch, ktoré sa nachádzali na mojom počítači. Výsledky boli veľmi dobré a presné, avšak treba vziať do úvahy, že mnoho z týchto súborov som používal na priebežné testovanie. Mal som k dispozícii aj jednu reálnu vzorku malware, ktorá používa práve ikonu adresára. Avšak konkrétne táto ikona sa v mojej porovnávacej množine nenachádzala, iba jej podobné.



Obrázok 7.5 – Obrázky extrahované zo vzorky malware

Pri otestovaní aplikácia vypísala podobnosť na 90% s ikonou iexplore_14.ico a 88% s shell32_234.ico (obr. 7.4). Takže v tomto reálnom teste na podobnosť ikon bola úspešná.

Ďalej som testoval navzájom posunuté ikony. Vytvoril som posunuté obrázky, ktoré som následne otestoval s originálnym.



Obrázok 7.5 – Vľavo hore originál, vpravo hore posunutý o vektor (12, 4), vľavo dole o (19, 10), vpravo dole o (58, 20). Všetky ikony majú rozmery 256 × 256

Najviac posunutá ikona už detekovaná nebola, pretože neprešla cez základný test a k upresňovaniu polohy už vôbec neprišlo. Ostatným 2 bola pridelená podobnosť s pôvodnou ionou 100%.

Časová náročnosť lineárne závisí na množstve vzorkových ikon v porovnávacej množine a počte obrázkov, ktoré sa nachádzajú v testovanom súbore. Pri 17 vzorkových ikonách a 12

obrázkoch z testovaného súboru doba behu programu bola 1,9 sekundy na dvojjadrovom procesore Intel Core 2 Duo 1,6 GHz. Pri využití viacerých vlákien sa tento čas skrátil na 1 sekundu. Pri využití viacjadrových procesorov by sa mal ďalej výrazne skracovať. Počas behu sa vytvorilo 1 vlákno pre každú jednu porovnávanú ikonu, teda dokopy 17 vlákien. Ak by bol program nasadený na systém s veľkým počtom jadier, bolo by možné vytvárať vlákno zvlášť pre každé jedno porovnávanie 2 obrázkov. Počet vlákien by tak bol viac 200 v tomto konkrétnom prípade.

8 Záver

V mojej práci som sa zameral na problematiku ikon v systémoch Windows a ich možné zneužitie tvorcami malware. Vysvetlil som štruktúru .ICO súboru a možné formáty obrázkov, ktoré sa v ikonách nachádzajú a takisto ich štandardné použitie. Ďalej som vysvetlil akým spôsobom môžu byť ikony zneužívané a okrem toho, akým spôsobom by mohli byť medzi sebou porovnávané, kde som bral zvlášť ohľad na ich priehľadnosť.

Významnou časťou tejto práce bolo vytvorenie aplikácie, ktorá dokáže extrahovať ikony zo spustiteľných súborov a otestovať ich na podobnosť so vzorkovými ikonami nachádzajúcimi sa v porovnávacej množine.

Do budúcnosti by bolo pravdepodobne vhodné vytvoriť ešte jednu metódu porovnávania, tento krát založenú na získavaní významných bodov z obrazu napr. SURF. Táto by potom bola aplikovaná pri porovnávaní väčších obrázkov. Výhodou by bola možnosť vytvoriť zoznam významných bodov pre obrázok už pri pridávaní ikony do porovnávacej množiny. Po spustení testovania by boli vytvorené pre obrázky z testovaného súboru a potom by už prichádzalo iba k porovnávaniu týchto príznakov, čo by mohlo priniesť urýchlenie porovnávania a určite zaujímavá by bola aj samotná úspešnosť.

Ak by sa mal výrazne zväčšiť počet vzorkových ikon, bolo by namieste urýchliť ich načítanie. Jednou možnosťou je umiestnenie všetkých ikon do jedného dátového súboru. Avšak v takom prípade by bolo nutné vytvoriť ešte grafické užívateľské rozhranie pre pridávanie ikon do porovnávacej množiny, tak aby mal užívateľ možnosť vybrať, ktoré ikony zo spustiteľného súboru chce naozaj pridať a ktoré nie.

Literatúra

[1] Kubík, Pavel: Kryptovirologie, diplomová práce, Brno, FIT VUT v Brně, 2008

[2] Hák, Igor: Moderní počítačové viry [online], aktualizované 2005-09-15 [nap. navštívené 2010-05-13]. Dostupné na URL: < http://www.viry.cz/viry.cz/kniha/kniha.pdf >

[3] Microsoft Corporation:Icons [online] [nap. navštívené 2010-05-13]. Dostupné na URL: < http://msdn.microsoft.com/en-us/library/aa511280.aspx >

[4] Microsoft Corporation:Icons [online] [nap. navštívené 2010-05-13]. Dostupné na URL: < http://msdn.microsoft.com/en-us/library/ms997538.aspx >

[5] Kerr, Kenny: Decoding Windows Vista icons with WIC, MSDN Magazine 2008-06, Dostupné na URL: < http://msdn.microsoft.com/en-us/magazine/cc546571.aspx >

[6] Roleofs, Greg: A basic introduction to png pictures [online] nap. navštívené 2010-05-13]. Dostupné na URL:
< http://msdn.microsoft.com/en-us/library/system.drawing.icon.aspx >

[7] Microsoft Corporation: Icon class [online] [nap. navštívené 2010-05-13]. Dostupné na URL: < http://msdn.microsoft.com/en-us/library/system.drawing.icon.aspx >

[8] Kageyu, Tsuda: Extract icons from EXE or DLL files [online] [nap. navštívené 2010-05-13]. Dostupné na URL: < http://www.codeproject.com/KB/cs/IconExtractor.aspx >

[9] Kršek, Přemysl: Základy počítačové grafiky [online] [nap. navštívené 2010-13-5]. Studijní opora predmetu IZG, FIT VUT v Brně. Dostupné na URL
< https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IZG-IT/texts/izg_opora.pdf >

[10] Dobeš, Michal: Zpracování obrazu a algoritmy v C#. BEN – technická literatura, Praha, 2008. ISBN 978-80-7300-233-6

[11] Dalal, Navneet, Triggs, Bill: Histograms of Oriented Gradients for Human Detection, CVPR'05 [online] [nap. navštívené 2010-13-5]. Dostupné na URL:
< http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf >

[12] Čížek, Roman: Detekce význačnách bodů v obraze, bakalářská práce, Brno, FIT VUT v Brně, 2008

[13] Kolektív autorov skupiny Matousec: KHOBE – 8.0 earthquake for Windows desktop security software [online] [nap. navštívené 2010-15-5]. Dostupné na URL:
< http://www.matousec.com/info/articles/khobe-8.0-earthquake-for-windows-desktop-security-software.php >

Zoznam príloh

Príloha 1 DVD obsahujúce prácu v elektronickej podobe, zdrojové súbory aplikácie, manuál.