VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

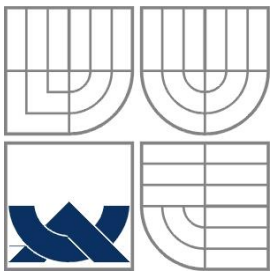
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ CITLIVÝCH INFORMACÍ Z FAT32

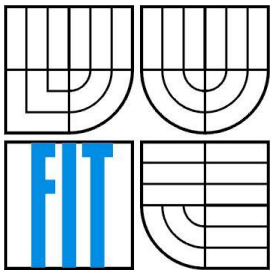
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUKÁŠ KRIŽAN
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ CITLIVÝCH INFORMACÍ Z FAT32

ANALYSIS OF PRIVATE INFORMATION FROM FAT32

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ KRIŽAN

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL OČENÁŠEK, PH.D.

BRNO 2015

Abstrakt

Cílem této práce bylo vytvořit aplikaci pracující se souborovým systémem FAT32, ve kterém vyhledá požadované soubory a obnoví je. Následně z těchto souborů získá citlivé informace. Nejprve jsou popsány principy FAT32 a způsob ukládání informací ve zkoumaných aplikacích. V posledních kapitolách je návrh, implementace a testování aplikace.

Abstract

The aim of this thesis was to create an application which can search for and recover specific files in the file system FAT32, and subsequently extract sensitive information from these files. The first part of this thesis describes the principles underlying FAT32, and focuses on how the analysed applications store information. The second part outlines the application and describes its implementation and simulations.

Klíčová slova

FAT, souborový systém, obnova dat, citlivé informace

Keywords

FAT, file system, data recovery, private information

Citace

Križan Lukáš: Získávání citlivých informací z FAT32, bakalářská práce, Brno, FIT VUT v Brně, 2015

Získávání citlivých informací z FAT32

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Križan
20. 5. 2015

Poděkování

Děkuji vedoucímu práce Ing. Pavlu Očenáškovi, Ph.D za odborné vedení a cenné rady poskytnuté při zpracování bakalářské práce.

Obsah

1 Úvod

Počítačová forenzná analýza získava dôkazy z informačných technológií, ktoré boli použité pri trestnej činnosti. Počet takýchto trestných činov je v súčasnosti na vzostupe a preto význam počítačovej forenznej analýzy narastá. Jednou z jej úloh je aj získavanie citlivých informácií, ktoré slúžia ako dôkaz alebo môžu byť použité v ďalšom procese získavania dôkazov.

Existuje mnoho aplikácií, ktoré riešia túto úlohu. Zameriavajú sa na jeden typ aplikácií, z ktorých získavajú informácie. Konkrétne sa jedná hlavne o prihlasovacie údaje a heslá, logovacie záznamy rôznych aplikácií a iné. Väčšina existujúcich riešení získava informácie z aplikácií nainštalovaných v operačnom systéme, čo je aj cieľom tejto práce. Od existujúcich riešení sa bude líšiť v získavaní informácií z prenositeľných verzií aplikácií uložených v súborovom systéme FAT32. Okrem alokovaných súborov bude aplikácia vyhľadávať aj zmazané súbory.

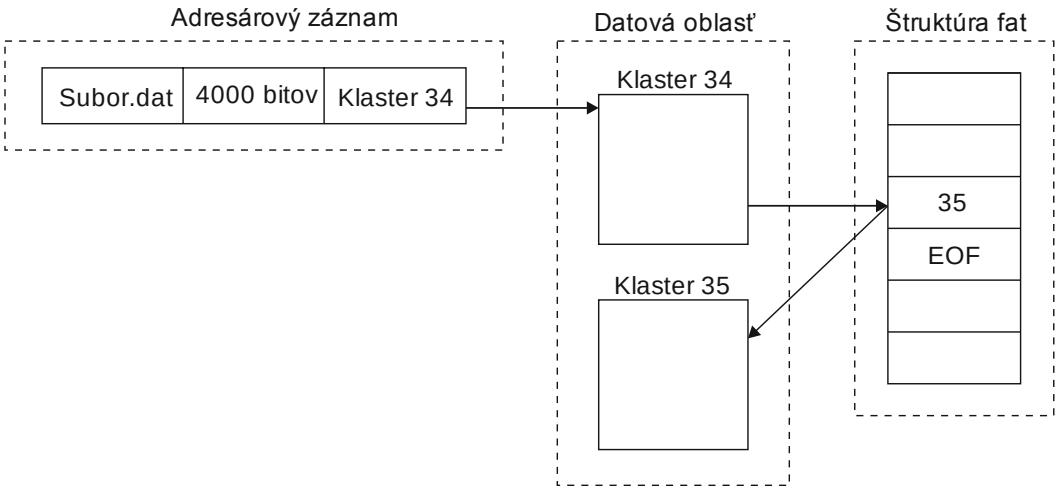
Nasledujúci text je členený do niekoľkých kapitol, ktorých obsah je nasledovný: Prvá kapitola oboznámi čitateľa s princípom fungovania súborového systému FAT, spôsobom ukladania a mazania dát. Druhá kapitola pojednáva o aplikáciách, z ktorých budú získavané prihlasovacie údaje a ich spôsob a miesto uloženia. V tretej kapitole je popísaná funkčnosť existujúcich riešení. Štvrtá kapitola obsahuje návrh aplikácie. Piata kapitola popisuje jej implementáciu. Predposledná kapitola obsahuje postupy a výsledky testovania a posledná kapitola obsahuje zhodnotenie dosiahnutých výsledkov a možnosti ďalšieho vývoja aplikácie.

2 Súborový systém FAT

Súborový systém FAT (File Allocation Table) vznikol na začiatku osemdesiatych rokov dvadsiateho storočia. Keďže bol vyvinutý pre osobný počítač IBM, sú jeho štruktúry typu little endian[1]. Je primárnym súborovým systémom operačných systémov Microsoft DOS a Windows 9x. V novších verziách Windows bol nahradený komplexnejším súborovým systémom NTFS (New Technology File System). V súčasnosti je FAT využívaný hlavne pre pamäťové karty a USB flash pamäte. [3]

Skladá sa z troch hlavných štruktúr. Prvou je zavádzací sektor, obsahujúci informácie o samotnom súborovom systéme. Ďalej štruktúra fat (pre jednoduchosť ďalej iba: fat pre označenie štruktúry a FAT pre súborový systém) a adresárové záznamy, slúžiace na uchovanie informácií o užívateľských dátach. Podľa veľkosti záznamov v štruktúre fat rozlišujeme tri typy súborového systému: FAT12, FAT16 a FAT32. Číslo za skratkou FAT určuje veľkosť záznamu v bitoch. FAT32 navyše obsahuje štruktúru FSINFO, ktorá uchováva adresu prvého nealokovaného klastra[2] a celkový počet nealokovaných klastrov. [1]

Pre každý súbor a adresár uložený vo FAT existuje adresárový záznam so svojím názvom, veľkosťou, adresou a metadátami. Ak je veľkosť súboru väčšia ako jeden klaster, adresa ďalšieho klastra sa nachádza v štruktúre fat. Vzájomné vzťahy medzi štruktúrami a dátami zobrazuje Obrázok 2.1. [2]



Obrázok 2.1 Vzťah medzi adresárovým záznamom, dátami a štruktúrou fat [2](vlastné prevedenie)

Na základe štruktúr je oddiel so súborovým systémom FAT rozdelený do troch oblastí. Veľkosť jednotlivých oblastí záleží na viacerých faktoroch, ale poradie ostáva nemenné.

- Vyhradená oblasť – obsahuje zavádzací sektor
- Oblasť FAT – obsahuje štruktúru fat
- Dátová oblasť (tiež oblasť dát) – obsahuje samotné užívateľské dáta a štruktúry a adresárový záznam. Základnou alokačnou jednotkou pre tuto oblasť je klaster.

[1] Little endian definuje poradie ukladania bitov číselného dátového typu. V tomto prípade je najprv uložený najmenej významový bit a za ním sú ukladané ďalšie bity až po najvýznamnejší bit.
[2] Klaster (tiež alokačná jednotka) je logická jednotka tvorená niekoľkými po sebe idúcimi sektormi.

2.1 Vyhradená oblasť

Vyhradená oblasť začína na nultom sektore súborového systému, pričom jej veľkosť závisí od typu FAT. Pre FAT12/16 je to typicky 1 sektor a FAT32 má zvyčajné vyhradených viacero sektorov. Obsahuje štruktúru zavádzacieho sektoru, ktorá obsahuje presnú veľkosť vyhradenej oblasti. FAT32 tu ukladá aj svoju štruktúru FSINFO. [2]

2.1.1 Zavádzací sektor

Zavádzací sektor, tiež označovaný ako BPB (BIOS Parameter Block) alebo nultý sektor, sa nachádza na prvom sektore vyhradenej oblasti FAT32. V prípade poškodenia hlavného sektora je možné použiť záložnú kópiu, ktorej adresa je uložená v hlavnom zavádzacom sektore. V dokumentácii Microsoft je táto adresa pevne určená, a to na šiesty sektor. Tým je umožnené jej automatické nájdenie v prípade poškodenia adresy v hlavnom zavádzacom sektore. [3]

FAT12/16 na rozdiel od FAT32 používajú odlišnú verziu zavádzacieho sektoru. Prvých 36 bajtov ma rovnaký význam vo všetkých verziách. Táto časť obsahuje základné informácie o oddiely. Význam všetkých bajtov je vysvetlený v Tabuľka 2.1. [3]

Bajty	Popis	Povinný
0-2	Inštrukcia skoku na zavádzací kód.	Nie
3-10	OEM názov v ASCII.	Nie
11-12	Počet bajtov na sektor, povolené hodnoty sú 512,1024, 2048 a 4096. Pre väčšiu kompatibilitu je vhodné požívať 512 bajtov.	Áno
13	Počet sektorov na klaster, povolené hodnoty sú mocniny 2, a však maximálne do 32KB.	Áno
14-15	Veľkosť vyhradenej oblasti v sektoroch, táto hodnota nemôže byť 0.	Áno
16	Počet štruktúr fat. Typicky dve pre redundanciu.	Áno
17-18	Maximálny počet záznamov v koreňovom adresári pre FAT16/32. S hodnotou 0 pre FAT32 a 512 typicky pre FAT16.	Áno
19-20	Počet sektorov v oddiely. Ak sú 2 bajty nedostačujúce na vyjadrenie hodnoty, nastavia sa na 0 a použijú sa bajty 32-35. Pre FAT32 vždy 0.	Áno
21	Typ média. Microsoft dokumentácie určuje hodnotu 0xF8 pre pevné a 0xF0 pre vymeniteľné média. Povolené hodnoty pre toto pole sú 0xF0 až 0xFF.	Nie
22-23	Vo FAT12/16 určuje veľkosť štruktúry fat v sektoroch. FAT32 nastavuje túto hodnotu na 0.	Áno
24-25	Počet sektorov na stopu.	Nie
26-27	Počet hláv v zariadení.	Nie
28-31	Počet sektorov do začiatku oddielu.	Nie
32-35	Počet sektorov v oddiely. Táto hodnota je nenulová len v prípade, že sú bajty 19-20 nastavene na 0. Pre FAT32 nesmie byť 0.	Áno

Tabuľka 2.1 Význam prvých 36 bajtov zavádzacieho sektora pre FAT12/16/32 [3]

V druhej časti sú uložené informácie potrebné pre daný typ FAT, čo umožňuje flexibilitu súborového systému. Na 510. až 511. bajte je signatúra 0xAA55, označujúca koniec zavádzacieho sektoru. Poloha signatúry je fixná, nemení sa ani v prípade, že je veľkosť sektoru väčšia ako 512 bajtov. Táto hodnota je rovnaká pre všetky typy FAT. Význam jednotlivých bajtov pre FAT12/16 je v Tabuľka 2.2 a pre FAT36 v Tabuľka 2.3. [3]

Bajty	Popis	Povinný
36	BIOS INT 0x13 číslo disku.	Nie
37	Nepoužívaný bajt.	Nie
38	Signatúra (0x29) potvrdzujúca platnosť nasledujúcich troch hodnôt.	Nie
39-42	Sériové číslo. Spolu s názvom oddielu slúži na identifikáciu odpojiteľných médií.	Nie
43-53	Názov oddielu v ASCII. Implicitná hodnota je "NO NAME ".	Nie
54-61	Názov súborového systému v ASCII.	Nie
62-509	Nepoužívané bajty.	Nie
510-511	Signatúra (0xAA550000).	Áno

Tabuľka 2.2 Význam posledných 476 bajtov zavádzacieho sektora pre FAT12/16 [3]

Bajty	Popis	Povinný
36-39	Veľkosť jednej fat štruktúry .	Áno
40-41	0. až 3. bit určuje aktívnu fat, ak je zakázané klonovanie. Ak je 7. bit nulový klonovanie je povolené.	Áno
42-43	Verzia FAT32, dokumentácia definuje len 0:0.	Áno
44-47	Adresa klastra s koreňovým adresárom.	Áno
48-49	Adresa sektoru s FSINFO.	Nie
50-51	Adresa sektoru so záložnou kópiou zavádzacieho sektoru.	Nie
52-63	Rezervované bajty.	Nie
64	BIOS INT 0x13 číslo disku.	Nie
65	Nepoužívané bajty.	Nie
66	Signatúra (0x29) potvrdzujúca platnosť nasledujúcich troch hodnôt.	Nie
67-70	Sériové číslo. Spolu s názvom oddielu slúži na identifikáciu odpojiteľných médií.	Nie
71-81	Názov oddielu v ASCII. Implicitná hodnota je "NO NAME ".	Nie
82-89	Názov súborového systému v ASCII.	Nie
90-509	Nepoužívané bajty.	Nie
510-511	Signatúra (0xAA550000).	Áno

Tabuľka 2.3 Význam posledných 476 bajtov zavádzacieho sektora pre FAT32 [3]

2.1.2 Zavádzací kód

Zavádzací kód je krátky úsek kódu, ktorý je po štarte zavedený BIOSom do operačnej pamäte a následne spustený. Je uložený v zavádzacom sektore a jeho úlohou je spustenie operačného systému. V prvých troch bajtoch je inštrukcia skoku s adresou na zvyšok kódu, ktorý je vo FAT12/16 na 64. až 509. bajte a vo FAT32 na 90. až 509. bajte. FAT32 môže v prípade, že je kód väčší ako 419 bajtov, použiť na jeho uloženie druhý sektor. Spolu so zavádzacím kódom sú uložené aj chybové správy. [4]

2.1.3 FSINFO

FSINFO je pomocná štruktúra využívaná pri alokovaní nového pamäťového miesta vo FAT32. Uchováva informácie o počte voľných klastrov a adresu prvého nealokovaného klastra. Použitie závisí od operačného systému, niektoré ju nemusia používať vôbec. Umiestnená je typicky v druhom alebo treťom sektore, podľa umiestnenia zavádzacieho kódu. Jej presná adresa je uložená v prvom sektore. Na disku je uložená na jeden sektor, ale využíva len prvé 4 a posledných 27 bajtov, zvyšné bajty sú nevyužité. Vysvetlenie významu jednotlivých bajtov je v Tabuľka 2.4. [2]

Bajty	Popis	Povinný
0-3	Signatúra (0x41615252).	Nie
4-483	Nepoužívané bajty.	Nie
484-487	Signatúra (0x61417272).	Nie
488-491	Počet voľných klastrov.	Nie
492-495	Adresa nasledujúceho voľného klastra.	Nie
496-507	Nepoužívané bajty.	Nie
508-511	Signatúra (0xAA550000), táto hodnota je rovnaká ako v zavádzacom sektore.	Nie

Tabuľka 2.4 Význam bajtov v štruktúre FSINFO [2]

2.2 Oblasť FAT

Oblasť fat začína hneď za vyhradenou oblasťou. Obsahuje zvyčajné dve štruktúry fat, pričom obidve sú platné iba ak je povolené zrkadlenie. Veľkosť celej fat oblasti sa vypočíta ako súčin veľkosti jednej štruktúry s ich počtom.

Fat štruktúru si možno predstaviť ako pole, kde index do poľa určuje poradie klastra, a hodnota informuje o jeho obsadenosti. Ak je klaster obsadený, jeho hodnota je nenulová a určuje číslo nasledujúceho klastra súboru alebo adresára. Ak ide o posledný klaster súboru alebo adresára, hodnota je 0xFF8 pre FAT12, 0xFFF8 pre FAT16 a 0x0FFFFFF8 pre FAT32. Pre chybné klastre, ktoré nie je možné alokovať, je to hodnota 0xFF7 pre FAT12, 0xFFF7 pre FAT16 a 0x0FFFFFF7 pre FAT32. Číslovanie klastrov v dátovej oblasti začína od čísla dva, no indexovanie štruktúry začína už od nuly, preto na indexoch 0 a 1 nie je informácia o klastroch. Typicky na indexe 0 býva uložený typ média, rovnaký ako v zavádzacom sektore, a na indexe 1 býva dirty status, ktorý indikuje chybu. Obe hodnoty sú nepovinné. [2]

Z hodnôt vidieť, že veľkosť záznamu sa líši podľa typu FAT. Preto ak chceme pracovať s fat štruktúrou, musíme najprv zistiť typ súborového sýtemu. Jedinou možnosťou, ako zistiť typ FAT, je podľa počtu klastrov v dátovej oblasti. Výpočet popisuje nasledujúci vzorec:

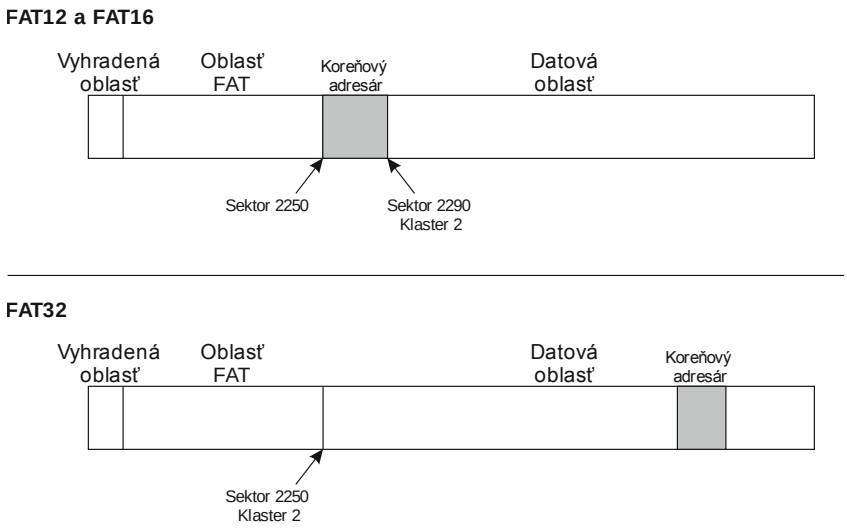
$$PocetKlastrov = \frac{PocetVsetkychSektorov - (SektorovVyhradOblasti + SektorovFATOblasti)}{PocetSektorovNaKlaster} \quad (1)$$

Ak je počet menší ako 4085 jedná sa o FAT12, pre interval 4085 až 65524 ide o FAT16. Ak je ich počet väčší ako 65524 ide o FAT32. [2]

2.3 Dátová oblasť

V dátovej oblasti sú uložené súbory s dátami a adresáre, ktoré sú uložené v alokačných jednotkách - klastroch. Klaster je skupina po sebe idúcich sektorov v dátovej oblasti. Počet týchto sektorov v skupine musí byť mocnina dvoch. Podľa špecifikácie Microsoft je maximálna veľkosť klastra stanovená na 32 kilobajtov. Každý klaster ma svoje číslo/adresu, pričom číslovanie začína od dvoch, takže prvý klaster v dátovej oblasti ma číslo dva. [2]

Pozícia prvého klastra v dátovej oblasti sa môže meniť podľa typu FAT. Vo FAT12 a FAT16 sa prvý klaster nachádza za koreňovým adresárom, ktorý nie je adresovaný za pomoci klastrov. Pri FAT32 sa prvý klaster začína na začiatku dátovej oblasti. Číslo sektoru, na ktorom začína, sa dá vypočítať ako súčet počtu sektorov vo vyhradenej oblasti a počtu sektorov v oblasti FAT. Porovnanie umiestnenia prvého klastra je zobrazené na Obrázok 2.2. [2]



Obrázok 2.2 Porovnanie umiestnenia prvého klastra a koreňového adresára [2](vlastné prevedenie)

Keďže dátová oblasť nezačína na začiatku oddielu, je adresovanie pomocou klastrov zložité. Jednoduchšie je používať na adresovanie sektory, ktoré umožňujú adresovanie vzhľadom na posunutie od začiatku oddielu. Fat štruktúra používa adresovanie podľa klastrov, a preto je dôležitý prevod medzi týmito dvoma spôsobmi adresovania. Prevod čísla klastra na číslo sektoru je zobrazený v nasledujúcej rovnici, kde premenná x je číslo klastra: [3]

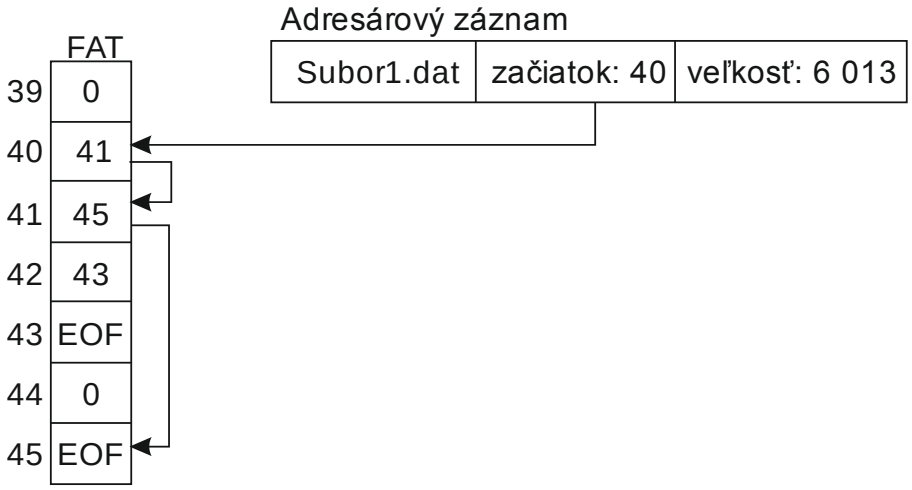
$$(x-2) * PocetSektorovNaKlaster + CisloSektoruPrvehoKlastra \quad (2)$$

Opačný prevod zo sektoru na klaster, kde x je číslo sektora:

$$\frac{(x-CisloSektoruPrvehoKlastra)}{PocetSektorovNaKlaster} + 2 \quad (3)$$

2.3.1 Reťazce klastrov

V adresárovom zázname je uložená iba adresa prvého klastra súboru a veľkosť. Ak chceme získať obsah celého súboru, musíme použiť štruktúru fat. Po načítaní obsahu klastra sa z fat získa hodnota uložená na indexe, ktorý je rovnaký ako číslo aktuálneho klastra. Získaná hodnota určuje číslo nasledujúceho klastra s dátami súboru alebo obsahuje hodnotu End of File (EOF), ktorá označuje posledný klaster súboru. Všetky hodnoty štruktúry fat sú popísané v kapitole 2.2. Takáto sekvencia klastrov sa tiež nazýva reťazec klastrov a jeho princíp je znázornený na Obrázok 2.3. Veľkosť súboru vo FAT teda určuje veľkosť fat záznamu. V prípade FAT32 je veľkosť záznamu 32 bitov, pričom vrchné dva bity sú nepoužité, čo umožňuje adresovať 268435456 klastrov. Táto hodnota je ešte znížená o hodnoty rezervované pre EOF a chybné klastre. [2]



Obrázok 2.3 Princíp reťazca klastrov [2](vlastné prevedenie)

2.3.2 Adresáre

Adresár vo FAT je špeciálny typ súboru obsahujúci lineárny zoznam štruktúr s názvom adresárový záznam. Pri vytvorení nového adresára sa alokuje daný klaster a celý jeho obsah sa vynuluje. Položka s veľkosťou súboru v adresárovom zázname uloženom v nadriadenom adresáry nie je použitá a zvyčajne sa nastavuje na nulovú hodnotu. Jedinou cestou, ako získať skutočnú veľkosť adresára, je začať na prvom klastry adresára a postupovať po reťazci klastrov, až kým nenarazíme na hodnotu EOF. [3]

Pri formátovaní oddielu je vždy vytvorený koreňový adresár, ktorý je predkom každého adresára v oddiely. Inými slovami je v adresárovej štruktúre najvyššie. Ako jediný obsahuje adresárový záznam s informáciami o oddiely. Jeho pozícia v dátovej oblasti sa líši od použitého FAT. Vo FAT12 a FAT16 je vždy uložený na začiatku dátovej oblasti s pevnou veľkosťou a nevzťahuje sa naň adresovanie pomocou klastrov. Naproti tomu vo FAT32 nie je jeho pozícia ani veľkosť presne stanovená a môže sa meniť. Adresa prvého klastra je uložená v zavádzacom sektore. Rozdiel umiestnenia koreňového adresára vo FAT je zobrazený na Obrázok 2.2. [2]

Prvé dva adresárové záznamy v každom adresári sú "." a "..". Výnimkou je len koreňový adresár, ktorý tieto dva záznamy neobsahuje. Záznam "." obsahuje informácie o aktuálnom adresári a záznam ".." je použitý pre priamy rodičovský adresár. Obsahujú všetky atribúty ako ostatné adresárové záznamy, ale operačné systémy Windows neaktualizujú hodnoty v položkách času, posledných uprav a prístupu. Tieto hodnoty zostávajú nastavené na čas vytvorenia adresára. [3]

2.3.3 Adresárový záznam

Adresárový záznam je dátová štruktúra, ktorá existuje pre všetky alokované súbory a adresáre. Jej veľkosť je 32 bajtov a je uložená v jednotlivých adresároch. Skladá sa zo siedmych položiek, ktoré môžu byť nastavené pre každý súbor alebo adresár. Niektoré položky nie sú nepovinné a je na operačných systémoch či ich budú používať. [2]

Jedna z položiek obsahuje atribúty súboru alebo ich kombinácie, ako napríklad atribút adresára, skrytý súbor, iba na čítanie, archív alebo dlhý názov súboru, čo je špeciálny typ adresárového záznamu, ktorého podrobnejší popis je v kapitole 2.3.5. Všetky položky štruktúry a hodnoty, ktoré môžu nadobúdať, sú vysvetlené v Tabuľka 2.5. [3]

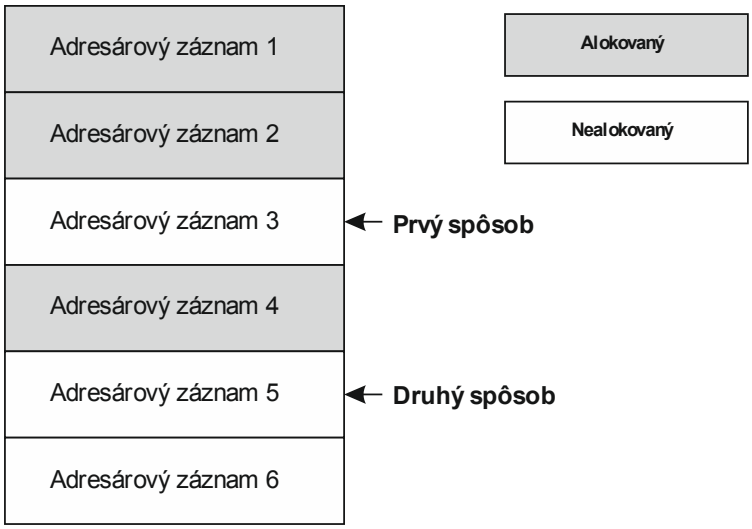
Bajty	Popis
0-10	Krátky názov súboru.

11	Atribúty súboru
	`ATTR_RAED_ONLY 0x01`
	`ATTR_HIDDEN 0x02`
	`ATTR_SYSTEM 0x04`
	`ATTR_VOLUME_ID 0x08`
	`ATTR_DIRECTORY 0x10`
	`ATTR_ARCHIVE 0x20`
	`ATTR_LONG_NAME ATTR_RAED_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID`
	
	Horné dva bity sú rezervované a pri vytvorení záznamu sú nastavené na hodnotu 0, ich hodnota sa nikdy nemení.

12	Rezervovaný bajt pre Windows NT, pri vytvorení nastavený na hodnotu 0, hodnota je nemenná.
13	Časová pečiatka vytvorenia súboru v milisekundách.
14-15	Čas vytvorenia súboru.
16-17	Dátum vytvorenia súboru.
18-19	Dátum posledného prístupu k súboru, čítanie alebo zápis. Ak bol súbor modifikovaný, položka bude obsahovať rovnakú hodnotu ako položka na bajtoch 24 až 25.
20-21	Najvyššie dva bajty adresy prvého klastra súboru.
22-23	Čas poslednej modifikácie súboru.
24-25	Dátum poslednej modifikácie súboru.
26-27	Najnižšie dva bajty adresy prvého klastra súboru.
28-31	Veľkosť súboru v bajtoch.

Tabuľka 2.5 Význam bajtov v štruktúre adresárový záznam [2]

Spôsob alokácie nových záznamov v adresári je na implementácii konkrétneho operačného systému. Existujú dva rôzne prístupy ako riešiť tento problém. Prvým prístupom je obsadenie prvého voľného miesta s dostatočnou veľkosťou. V druhom prístupe sa nový záznam uloží až za posledný alokovaný záznam. Oba prístupy sú znázornené na Obrázok 2.4. Preto adresárový záznam nemá unikátnu adresu, ako napríklad klaster, a ak chceme vyhľadať daný záznam musíme prejsť celý obsah adresára a spracovať záznam po zázname. [2]



Obrázok 2.4 Znázornenie dvoch spôsobov alokácie [2](vlastné prevedenie)

2.3.4 Krátky názov súboru

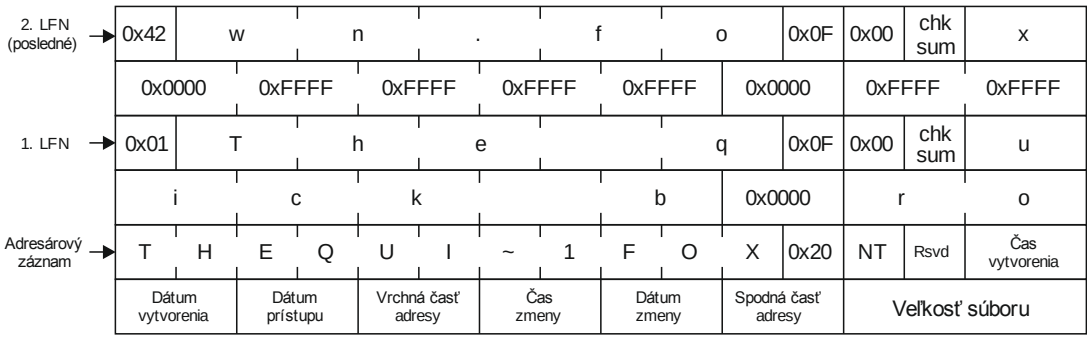
Na prvých jedenástich bajtoch adresárového záznamu je uložený krátky názov súboru. Skladá sa z dvoch častí: prvá časť s veľkosťou osem bajtov, ktorá je vytvorená zo skutočného názvu súboru. Zo skutočného názvu sa zoberie prvých osem znakov, ktoré sa skonvertujú na verzálky (ak je názov krátky, chýbajúce znaky sa doplnia medzerami). Druhá časť je vytvorená z prípony súboru. Postup je rovnaký ako pri prvej časti s rozdielom, že druhá časť obsahuje len tri znaky. [2]

V prípade, že pre súbor existuje aj záznam dlhého názvu súboru, sú posledné dva bajty prvej časti nahradené znakmi *"~"* a *"1"*. [3]

Prvý bajt ma špeciálny význam. Ak obsahuje hodnotu 0xE5 je záznam nealokovaný a súbor, o ktorom uchovával informácie je vymazaný. Hodnota 0x00 informuje o konci zoznamu záznamov. Ostatné hodnoty sú rovné prvému písmenu názvu súboru. [3]

2.3.5 Dlhy názov súboru (LFN[3])

Záznam dlhý názov súboru je špeciálny typ adresárového záznamu. Slúži na uloženie názvov súborov, ktorých dĺžka presahuje osem znakov alebo názvov obsahujúcich špeciálne znaky. Jeden znak ukladá na dvoch bajtoch a dokáže uchovať až trinásť znakov. Pre jeden súbor môže existovať viac záznamov, vďaka čomu je možné ukladať súbory s názvom dlhším ako 13 znakov. Všetky záznamy dlhého názvu súboru sú uložené hneď pred adresárovým záznamom daného súboru. Na Obrázok 2.5 je znázornený spôsob uloženia súboru s názvom *"The quick brown.fox"*. Význam jednotlivých položiek vysvetľuje Tabuľka 2.6. [2]



Obrázok 2.5 Uloženie adresárových záznamov pre súbor "The quick brown.fox" [2](vlastné prevedenie)

Bajty	Popis
0	Určuje poradie štruktúry v rámci daného súboru. Ak sa jedná o posledný záznam, k bajtu je pripočítaná hodnota 0x40.
1-10	Prvý až piaty znak názvu.
11	Atribút, musí byť nastavený na `ATTR_LONG_NAME`.
12	Musí byť nula.
13	Kontrolný súčet.
14-25	Šiesty až jedenásty znak názvu.
26-27	Musí byť nula.
28-31	Dvanásty až trinásty znak názvu.

Tabuľka 2.6 Vysvetlenie významu bajtov záznamu dlhého názvu súboru [3]

Kontrolný súčet záznamu sa počíta z krátkeho názvu súboru a je rovnaký pre všetky záznamy daného súboru. Postupne sa sčítajú všetky znaky z krátkeho názvu a medzivýsledky sa rotujú doľava o jeden bit. Cely princíp vysvetľuje nasledujúci kód: [3]

```
checksum = 0;
for (int i = 0; i < 11; i++) {
      checksum = ROR(checksum, 1);
      checksum += nasledujuci_znak;
}
```

Algoritmus 2.1 Výpočet kontrolného súčtu LFN [3]

[3] Long file name

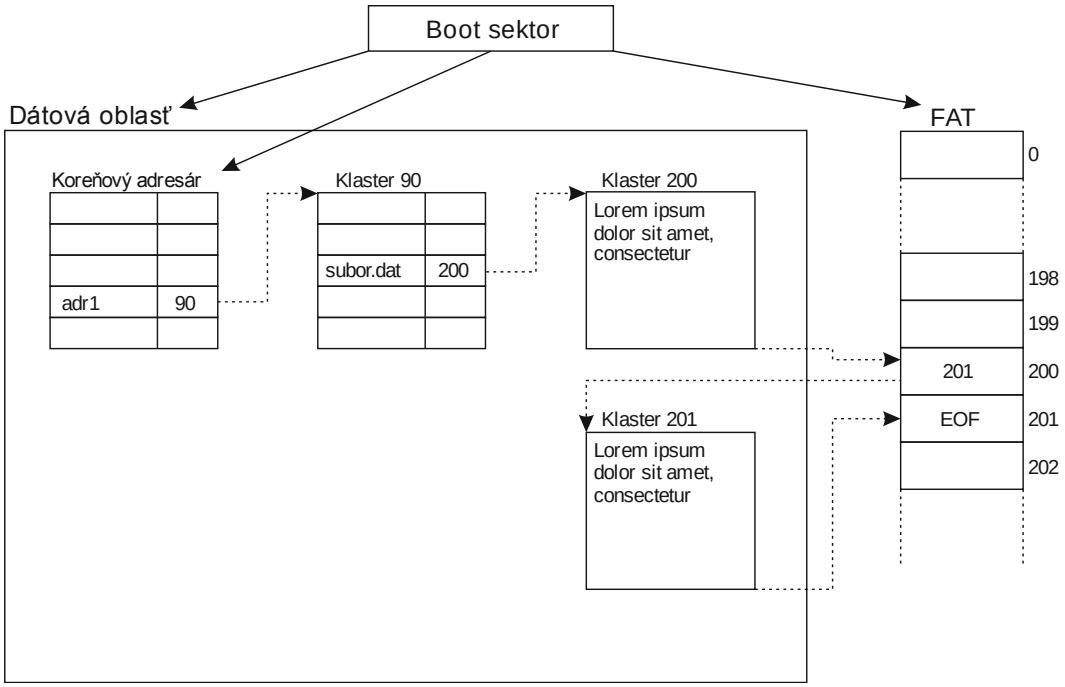
2.4 Postup ukladania a mazania súborov

V nasledujúcej kapitole je popísaný postup ukladania a mazania súborov v súborovom systéme FAT. V príklade budeme používať adresár *"dir1"*, ktorý je uložený v koreňovom adresári, a súbor *"file1.dat"* s veľkosťou 6000 bajtov, ktorý najprv uložíme do adresára *"dir1"* a následne ho zmažeme. Veľkosť klastra je 4096 bajtov. [2]

2.4.1 Ukladanie súboru

1. Zo zavádzacieho sektoru získame umiestnenie aktívnej fat štruktúry, umiestnenie dátovej oblasti a koreňového adresára v nej.
2. Prechádzame adresárovú štruktúru od koreňového adresára, až kým nenájdeme adresár *"dir1"*. Z nájdeného záznamu získame číslo prvého klastra adresára.
3. Prechádzame obsah adresára *"dir1"*, kým nenájdeme voľné miesto pre záznam.
4. Na danom mieste vytvoríme adresárový záznam a vyplníme známe hodnoty.
5. V štruktúre fat nájdeme nealokovaný klaster a nastavíme hodnotu na EOF.
6. Zapíšeme adresu klastra do adresárového záznamu zo 4. kroku a do klastra nahráme 4096 obsahu súboru *"file1.dat"*.
7. Vo fat nájdeme ďalší voľný klaster pre zvyšok dát a nastavíme hodnotu na EOF.
8. Vo fat na indexe rovnom číslu klastra z kroku päť zmeníme hodnotu na číslo klastra zo siedmeho kroku. Do klastra nahráme zvyšok dát. [2]

Výsledný stav po uložení súboru zobrazuje Obrázok 2.1.

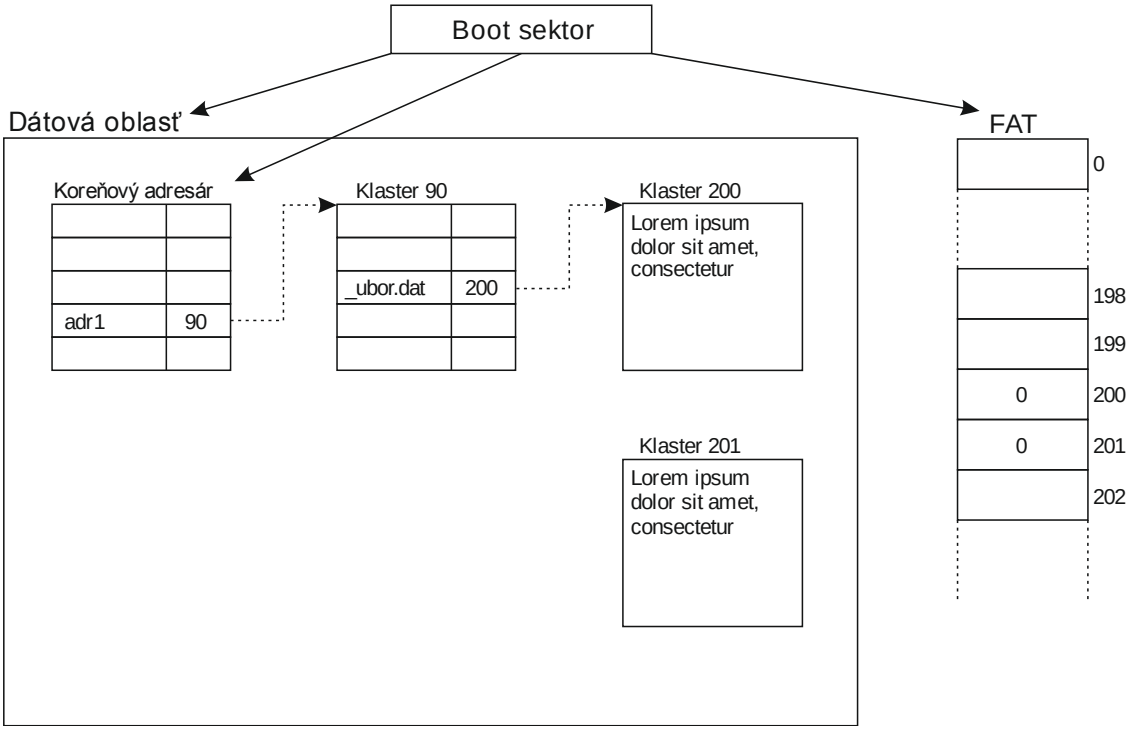


Obrázok 2.6 Výsledný stav po uložení súboru [2](vlastné prevedenie)

2.4.2 Mazanie súboru

1. Zo zavádzacieho sektoru získame umiestnenie aktívnej fat štruktúry, umiestnenie dátovej oblasti a koreňového adresára v nej.
2. Prechádzame adresárovú štruktúru od koreňového adresára až kým nenájdeme adresár *"dir1"*. Z nájdeného záznamu získame číslo prvého klastra adresára.
3. Prechádzame obsah adresára *"dir1"* kým nenájdeme záznam o súbore *"file1.dat"*, z ktorého získame číslo prvého klastra súboru.
4. Pomocou fat štruktúry získame reťazec klastrov súboru. Každému klastru z reťazca nastavíme vo fat hodnotu na 0. Tým sme označili klastre ako nealokované.
5. Zmeníme hodnotu prvého bajtu v adresárovom zázname súboru na hodnotu 0xE5, čím toto miesto označíme ako nealokované. [2]

V systémoch Windows Vista a novších som zistil odchýlku pri postupe mazania súboru. Okrem nastavenia prvého bajtu v adresárovom zázname v kroku päť, je vynulovaná aj položka s dvoma najvyššími bajtmi adresy prvého klastra súboru. Táto odchýlka komplikuje obnovu zmazaných súborov. Výsledný stav po zmazaní súboru na starších systémoch zobrazuje Obrázok 2.7.



Obrázok 2.7 Výsledný stav po zmazaní súboru [2](vlastné prevedenie)

3 Skúmané aplikácie

Vzhľadom na to, že FAT32 je v dnešnej dobe využívaná hlavne pre USB flash pamäte, bol pri výbere aplikácií kladený dôraz na existenciu prenositeľných verzií. Skúmané prenositeľné verzie sú priamo od vydavateľa alebo zo stránky portableapps.com. Vybrané aplikácie možno rozdeliť do dvoch kategórií, a to aplikácie určené na komunikáciu a FTP klienti. Aplikácie z týchto kategórií bývajú často využívané pri trestnej činnosti ako sťahovanie a šírenie nelegálneho obsahu, šírenie pedofilného obsahu, plánovanie nelegálnej činnosti a iné. Z týchto aplikácií sa budú získavať prihlasovacie údaje k rôznym službám. Tieto údaje môžu byť užitočné pri ďalšom vyšetrovaní vo forenznej analýze.

V podkapitolách sú informácie o vybraných aplikáciách. Obsahujú popis aplikácie, miesto a spôsob uloženia prihlasovacích údajov.

3.1 Pidgin

Domovská stránka programu: www.pidgin.im

Pidgin je multiplatformový program pre zasielanie rýchlych správ, bežiaci pod operačnými systémami Windows a Linux. Umožňuje v jednej chvíli prihlásenie do viacerých sietí pre zasielanie rýchlych správ. Podporuje funkcie ako prenášanie súborov, notifikáciu písania druhej strany a mnoho ďalších. Množina funkcií môže byť ďalej rozšírená pomocou zásuvných modulov. Vyvíjaný je pod licenciou GNU General Public License, preto sú všetky zdrojové kódy voľne dostupné a ktokoľvek si ich môže upraviť podľa svojich potrieb.

Podporované sú protokoly: AIM, ICQ, Google Talk, Jabber/XMPP, MSN Messenger, Yahoo!, Bonjour, Gadu-Gadu, IRC, Novell GroupWise Messenger, Lotus Sametime, SILC, SIMPLE, MXit, a Zephyr. Podporu ďalších protokolov je možné dosiahnuť použitím zásuvných modulov. [5]

3.1.1 Spôsob uloženia účtov

Pidgin ukladá všetky informácie o účtoch do XML súboru s názvom *"Accounts.xml"*. Súbor obsahuje pre každý účet značku <account>, v ktorej sú vnorené značky <name> a <password>, pre uchovanie prihlasovacieho mena a hesla. Obe hodnoty sú uložené ako otvorený text[4].

Umiestnenie súboru závisí od použitej verzie (inštalovaná alebo prenosná) programu. V inštalovanej verzii sa líši miesto uloženia podľa verzie operačného systému, na ktorom je program nainštalovaný.

Verzia inštalovaná na Windows XP:
C:\Documents and Settings\<užívateľské_meno>\Application Data\.purple

Verzia inštalovaná na Windows Vista a Windows 7:
C:\Users\<užívateľské_meno>\AppData\Roaming\.purple

Prenositeľná verzia z portableapps.com:
<umiestnenie_prenositeľnej_verzie>\Data\settings\.purple

[4] Otvoreným textom sa v kryptografii rozumie nezašifrovaný text, uložený v čitateľnej podobe.

3.2 Miranda

Domovská stránka programu: www.miranda-im.org

Miranda je multiprotokolový klient pre zasielanie rýchlych správ určený pre operačný systém Microsoft Windows. Dostupných je vyše 350 zásuvných modulov, ktoré rozširujú funkcionalitu a pridávajú podporu ďalších protokolov. Vďaka zásuvným modulom je Miranda veľmi flexibilná a každý si ju môže prispôsobiť podľa svojich predstav. Publikovaná je pod licenciou The GNU General Public License.

Podporované protokoly: AIM, Gadu-Gadu, ICQ, IRC, Jabber, MSN, Yahoo!, Battle.Net, Bonjour, C6, Google Talk, Facebook, Fetion, IAX, iChat, Lotus, Sametime, Mail.ru Agent, MeBeam, MySpace, NetSend, Omegle, QQ, SIP RTC, Skype, Tencent QQ, Tlen, Twitter, VKontakte, Weather services, Winpopup a Xfire. [6]

3.2.1 Spôsob uloženia účtov

Miranda používa upravenú verziu databáze ICQ pre uchovanie informácií o účtoch. Uchováva informácie ako typ protokolu, meno, heslo a iné. Štruktúry obsahujúce informácie o účte začínajú vždy kľúčovým slovom *"AM_BaseProto"*. [7]

Algoritmus na zašifrovanie hesla sa líši podľa protokolu služby. Pre JABBER sa na každý znak hesla použije funkcia XOR s konštantou 0x0C. Pre zvyšné protokoly sa použije algoritmus, ktorý od každého znaku odpočíta konštantu 0x05. [7]

Umiestnenie databázy nainštalovaného programu závisí od operačného systému, na ktorom je nainštalovaný. Názov súboru s databázou sa mení podľa mena profilu, prípona súboru je *"dat"*.

Verzia nainštalovaná na Windows XP:
C:\Documents and Settings\<užívateľské_meno>\ApplicationData\Miranda\<meno_profilu>

Verzia nainštalovaná na Windows Vista a Windows 7:
C:\Users\<užívateľské_meno>\AppData\Roaming\Miranda\<meno_profilu>

Prenositeľná verzia od vydavateľa:
<umiestnenie_prenositeľnej_verzie>\ Profiles\<meno_profilu>

Prenositeľná verzia z portableapps.com:
<umiestnenie_prenositeľnej_verzie>\Data\profiles\<meno_profilu>

3.3 FileZilla

Domovská stránka programu: www.filezilla-project.org

FileZilla je obľúbené FTP riešenie, skladajúce sa z dvoch časti - servera a klienta. Dostupná je pre platformy Microsoft Windows, Linux a Mac OS X. Podporuje protokoly FTP, SFTP, FTPS. Všetky zdrojové kódy sú voľne šíriteľné pod licenciou GNU General Public License. [8]

3.3.1 Spôsob uloženia účtov

Prihlasovacie údaje k serverom budeme získavať len z FTP klienta, ktorý umožňuje ich ukladanie. Všetky informácie o serveroch spolu s menom a heslom sú ukladané do XML súboru s názvom *"recentservers.xml"*. Pre každý server je v tomto súbore značka <server>. Obsahuje vnorené značky <user> a <pass>, ktoré obsahujú prihlasovacie meno a heslo.

Hesla sú uložené vo formáte Base64[5]. Umiestnenie súboru v inštalovanej verzii sa líši podľa operačného systému.

Verzia nainštalovaná na Windows XP:
C:\Documents and Settings\<užívateľské_meno>\Application Data\FileZilla

Verzia nainštalovaná na Windows Vista a Windows 7
C:\Users\< užívateľské_meno >\AppData\Roaming\FileZilla

Prenositeľná verzia z portableapps.com:
<umiestnenie_prenositeľnej_verzie>\Data\settings

[5] www.tools.ietf.org/pdf/rfc4648.pdf

3.4 Total Commander

Domovská stránka programu: www.ghisler.com

Total Commander je súborový manažér pre systémy Microsoft Windows a Android. Umožňuje pracú s archivovanými súbormi vo formátoch ZIP, TAR, ARJ, LZH, GZ, CAB, RAR a ACE. Podpora ďalších formátov môže byť doplnená zásuvnými modulmi. Obsahuje vstavaného FTP klienta s podporou FXP. S knižnicou OpenSSL umožňuje komunikáciu protokolom FTPS. Program je publikovaný pod licenciou Shareware. [9]

3.4.1 Spôsob uloženia účtov

Vstavaný FTP klient umožňuje ukladanie prihlasovacích údajov, ktoré sa budeme snažiť získať. Všetky údaje sú ukladané do jediného súboru s názvom *"wcx_ftp.ini"*. Heslo je uložené v šifrovanej podobe, na šifrovanie sa využíva vlastný algoritmus. Ostatné údaje sú uložené ako otvorený text.

Pri dešifrovaní hesla potrebujeme generátor čísel s nasledujúcim kódom:

```
tc_random(nMax) {
     randomBase = (randomBase * 0x8088405) + 1;
     return (randomBase * nMax) >> 32;
}
```

Algoritmus 3.1 Generátor čísel používaný aplikáciou Total Commander [10]

Postup dešifrovania hesla:

1. Odstránime posledných osem znakov z hesla.
2. Zlúčime každé dva susedné znaky, čím dostaneme dvojciferné čísla v šestnástkovej sústave.
3. Obe čísla prevedieme na znaky.
4. Nastavíme premennú `randomBase` (základ pre generátor čísel), na hodnotu 849521.
5. Rotácia každého znaku doľava o hodnotu vygenerovanú `tc_random(8)`.
6. Nastavíme premennú `randomBase` na hodnotu 12345.
7. 255-krát vygenerujeme dve čísla pomocou `tc_random(dĺžkaHesla)` a znaky na ich pozíciách vymeníme.
8. Nastavíme premennú `randomBase` na hodnotu 42340.
9. Na každý znak použijeme logickú funkciu `XOR` s `tc_random(256)`.
10. Nastavíme premennú `randomBase` na hodnotu 54321.
11. Od každého znaku odpočítame `tc_random(256)`.

Umiestnenie súboru wcx_ftp.ini sa líši na základe verzie programu a operačného systému.

Verzia nainštalovaná na Windows XP:
C:\Documents and Settings\<užívateľské_meno>\Application Data\ Roaming\GHISLER

Verzia nainštalovaná na Windows Vista a Windows 7:
C:\Users\<meno_uživatela>\AppData\Roaming\GHISLER

Prenositeľná verzia od vydavateľa:
<umiestnenie_prenositeľnej_verzie>

3.5 WinSCP

Domovská stránka programu: www.winscp.net

WinSCP je FTP a SFTP klient s otvoreným zdrojovým kódom pre systém Windows. Zároveň podporuje aj starší protokol SCP. Okrem toho obsahuje konzolu s mnohými príkazmi, ktorá umožňuje vytváranie skriptov a ich spúšťanie zo súboru. Aplikácia je publikovaná pod licenciou GNU General Public License. [11]

3.5.1 Spôsob uloženia účtov

Uloženie prihlasovacích údajov sa líši podľa nastavení aplikácie. V implicitnom nastavení nainštalovanej aplikácie sa ukladajú účty do systémových registrov, pod kľúčom *[HKEY_CURRENT_USER\Software\<User>\WinSCP 2]*. V prípade prenositeľnej verzie sa ukladajú účty do súboru *"WinSCP.ini"*. Súbor je vždy umiestnený v rovnakom adresári ako spúšťací súbor programu (WinSCP.exe). Tento spôsob ukladania účtov sa dá nastaviť aj v nainštalovanej aplikácii.

Heslo je uložené v zašifrovanej podobe. Na dešifrovanie hesla potrebujeme reťazec "0123456789ABCDEF", v ktorom pozícia znaku určuje jeho hodnotu a magickú konštantu 0xA3. Algoritmus dešifrovania hesla je nasledovný: [12]

1. Zo zašifrovaného hesla zoberieme dva znaky.
2. Pomocou reťazca ich prevedieme na odpovedajúce hodnoty.
3. Spravíme súčet oboch hodnôt.
4. Na výslednú hodnotu aplikujeme logickú funkciu `XOR` s magickou konštantou.
5. Na výsledok operácie aplikujeme logickú negáciu.
6. Kroky 1 až 5 opakujeme, kým zašifrované heslo obsahuje znaky.

3.6 FashFXP

Domovská stránka programu: www.flashfxp.com
FlashFXP je FTP klient s grafickým rozhraním. Preložený je do viac ako 20 jazykov. Podporuje protokoly FTP, SFTP, FTPS. Cena programu je 29.95 USD, dostupná je aj skúšobňa verzia, ktorú je možne používať podobu tridsiatich dni zadarmo. [13]

3.6.1 Spôsob uloženia účtov

Aplikácia ukladá účty do dvoch súborov. Prvý s názvom *"Sites.dat"* je určený pre trvalé uloženie účtov. Druhý *"quick.dat"* je určený pre rýchle pripojenia. Oba súbory majú textový formát s kódovaním UTF-8. Ukladané sú informácie o adrese, porte servera a prihlasovacie meno a heslo. Heslo je uložené v šifrovanej podobe a na jeho dešifrovanie je potrebný magicky reťazec "yA36zA48dEhfrvghGRg57h5UlDv3". Algoritmus je nasledovný: [14]

1. Zašifrované heslo rozdelíme na štvorice znakov.
2. Každú štvoricu rozdelíme na dve čísla v šestnástkovej sústave.
3. Na prvé číslo aplikujeme `XOR` so znakom z reťazca na pozícii poradia štvorice.
4. Od výsledku odpočítame druhé číslo zo štvorice.
5. Ak je výsledok záporný prirátame k nemu 255.
6. Opakujeme pre každú štvoricu.

Súbory *"Sites.dat"* a *"quick.dat"* sú uložené v rovnakom adresári.

Verzia nainštalovaná na Windows XP:
C:\Documents and Settings\<užívateľské_meno>\Application Data\FlashFXP

Verzia nainštalovaná na Windows Vista a Windows 7
C:\Users\< užívateľské_meno >\AppData\FlashFXP

Prenositeľná verzia od vydavateľa:
<umiestnenie_prenositeľnej_verzie>

4 Existujúce riešenia

Existujúce riešenia možno rozdeliť na dve skupiny. Prvá získava citlivé informácie z rôznych aplikácií a predáva ich užívateľovi v zrozumiteľnej podobe. Sem patria aplikácie Advanced IM Password Recovery, MessenPass, Instant Messengers Password Recovery Master, FTP Password Recovery Master a FTP Password Decryptor. Druhou skupinou sú aplikácie, ktoré umožňujú obnovu dát z USB Flash diskov. Do tejto skupiny patria napríklad PhotoRec a Recuva.

4.1 Advanced IM Password Recovery

Domovská stránka programu: www.elcomsoft.com/aimpr.html

Advanced IM Password Recovery slúži na získavanie informácií o užívateľských účtoch z aplikácií pre zasielanie rýchlych správ. V súčasnosti umožňuje získavanie informácií z viac ako 70 rôznych aplikácií, medzi ktoré patria napríklad ICQ, Yahoo! Messenger, Pidgin, Gajim, Jabber IM, Windows Live Messenger, Google Talk a mnohé ďalšie. [15]

Okrem automatického zisťovania prihlasovacích údajov, umožňuje extrahovanie zo zadaného súboru uchovávajúceho tieto informácie. Táto funkcia nie je dostupná pre všetky podporované aplikácie. Aplikácia disponuje jednoduchým grafickým rozhraním, v ktorom sú zobrazené získané informácie formou tabuľky. Neumožňuje ich ukladanie ani export. Zaujímavou funkciou je zálohovanie a obnova súborov s užívateľskými účtami. Cena aplikácie je 39€, dostupná je aj skúšobná verzia, ktorá ma obmedzený počet podporovaných aplikácii, z ktorých získava údaje.

4.2 MessenPass

Domovská stránka programu: www.nirsoft.net/utils/mspass.html

Aplikácia MessenPass je určená na obnovu užívateľských účtov z aplikácií pre zasielanie rýchlych správ. Oproti Advanced IM Password Recovery disponuje menšou množinou podporovaných aplikácií. Zisťovanie informácií prebieha automaticky a výsledky sú zobrazované v grafickom rozhraní. Zaujímavou funkciou je výpočet sily hesla. Vybrané údaje môžu byť exportované do formátov XML, HTML a CSV. Aplikáciu možno spúšťať aj v príkazovom riadku bez grafického rozhrania. Automatické ukladanie do súboru v danom formáte sa dá nastaviť pomocou parametrov. Aplikácia funguje bez inštalácie a pre osobné použitie je zadarmo.

Podporovane aplikácie: Windows Live Messenger, Yahoo Messenger, Google Talk, ICQ, AIM, Trillian, Miranda, Pidgin, MySpace IM, PaltalkScene a Digsby. [16]

4.3 Instant Messengers Password Recovery Master

Domovská stránka programu: www.rixler.com/instant_messengers_password_recovery.htm

Instant Messengers Password Recovery Master je ľahko použiteľné riešenie na zobrazovanie prihlasovacích údajov uložených v aplikáciách pre zasielanie rýchlych správ. Umožňuje skopírovanie prihlasovacieho mena alebo hesla do schránky. Okrem toho všetky získané údaje môžu byť uložené do textového súboru. Cena aplikácie sa líši podľa použitia: pre osobné použitie je 19,95$ a pre komerčné 39,85$. Je možné stiahnuť skúšobnú verziu, ktorá je obmedzená na zobrazenie hesiel len pre niektoré z podporovaných aplikácii. [17]

Podporované aplikácie: Windows Live Messenger, ICQ, MSN Messenger, Yahoo! Messenger, Trillian, Trillian Astra, Miranda, GAIM, Pidgin, SIM, R&Q, AIM, AIM Pro, QIP, PSI, FAIM, Paltalk, Google Talk, Excite Private Messenger, MySpaceIM a IM2. [17]

4.4 FTP Password Recovery Master

Domovská stránka programu: www.rixler.com/ftp_password_recovery.htm

Aplikácia FTP Password Recovery Master je určená na získavanie prihlasovacích údajov z FTP klientov. Je od rovnakého vývojára ako Instant Messengers Password Recovery Master a majú rovnakú cenu aj funkcie.

Podporované aplikácie: Total Commander, Far Manager, WS_FTP, CuteFTP, FlashFXP, FileZilla, FTP Commander, SmartFTP, BulletProof FTP, TurboFTP, FFFTP, Core FTP, FTP Explorer a Directory Opus. [17]

4.5 FTP Password Decryptor

Domovská stránka programu: www.securityxploded.com/ftp-password-decryptor.php

FTP Password Decryptor získava prihlasovacie údaje z FTP klientov. Aplikácia môže byť po nainštalovaní spustená s jednoduchým grafickým rozhraním alebo v príkazovom riadku. Získané údaje je možne exportovať do HTML, XML, CSV alebo textového súboru. Aplikácia je dostupná zadarmo.

Podporované aplikácie: FileZilla, SmartFTP, FlashFXP, FTPCommander, Dreamweaver, WS_FTP. [18]

4.6 PhotoRec

Domovská stránka programu: www.cgsecurity.org/wiki/PhotoRec

PhotoRec je multiplatformový program na obnovu zmazaných dát z pevných diskov, USB Flash pamäti, z pamäti digitálnych kamier a CD-ROM. Obnovuje súbory na základe formátu. Vďaka tomu umožňuje obnovu aj pri poškodenom súborovom systéme. Rozpoznáva viac ako 440 formátov. Aplikácia je publikovaná pod licenciou GNU General Public License. [19]

Pri spustení neumožňuje výber súborov k obnove, ale obnoví všetky súbory, ktoré nájde. Obnoveným súborom nezachováva pôvodný názov ani adresárovú štruktúru.

4.7 Recuva

Domovská stránka programu: www.piriform.com/recuva

Recuva je program s grafickým rozhraním umožňujúci obnovu zmazaných dát. Po prehľadaní pamäte sú nájdené súbory vypísané. Na rozdiel od PhotoRec môžeme vybrať zo zoznamu súborov iba tie, ktoré majú byť obnovené.

Umožňuje dva spôsoby prehľadávania pamäte. Pri prvom spôsobe prejde adresárovú štruktúru od koreňového adresára a snaží sa vyhľadať zmazané súbory. Druhý spôsob nazývaný hlboké skenovanie, prechádza pamäť sektor po sektore a snaží sa nájsť súbory k obnove. Tento spôsob je oproti prvému účinnejší ale pomalší. [20]

Program je možné bezplatne stiahnuť z domovskej stránky, existuje aj verzia Professional ktorej cena je 25 USD. [20]

5 Analýza a návrh aplikácie

Navrhovaná aplikácia by mala vyhľadať požadované súbory zo zadaného oddielu so súborovým systémom FAT32. Následne tieto súbory spracuje a získa z nich relevantné informácie. Získané informácie poskytne používateľovi v zrozumiteľnej forme.

Ako cieľovú platformu tohto projektu som si zvolil operačný systém Windows. Môj výber vyplýva z toho, že všetky skúmané aplikácie sú pre operačné systémy Windows. Aplikácia nebude obsahovať grafické používateľské rozhranie. Spúšťaná bude z príkazového riadku a všetky možnosti aplikácie bude možné nastaviť pomocou parametrov. Výhodou tohto riešenia je umožnenie použitia aplikácie v skriptoch, vďaka čomu je možná automatizácia získavania informácií z viacerých oddielov.

Aplikácia bude pozostávať z dvoch hlavných častí. Prvou je trieda pre prácu so súborovým systémom FAT32. Pracovať bude na úrovni sektorov a klastrov, čo zabezpečí úplnú kontrolu nad súborovým systémom. Tento prístup umožní vyhľadávanie zmazaných súborov v oddiely. Trieda by mala obsahovať rozhranie, pomocou ktorého jej budú predané názvy súborov a kľúčové slová, na základe ktorých bude vyhľadávať požadované súbory v oddiely. Nájdené súbory následne obnoví do výstupného adresára, ktorý bude možné nastaviť pred spustením vyhľadávania. Ak nebude nastavená cesta výstupného adresára, použije sa cesta adresára, z ktorého bola aplikácia spustená. S obnovenými súbormi je možné ďalej pracovať a analyzovať ich ďalšími aplikáciami. Objekt bude obsahovať logovací systém, ktorý bude zaznamenávať informácie o získaných súboroch, a to konkrétne posunutie od začiatku oddielu v bajtoch a názov výstupného súboru. Log bude možné vypísať na štandardný výstup alebo uložiť do súboru.

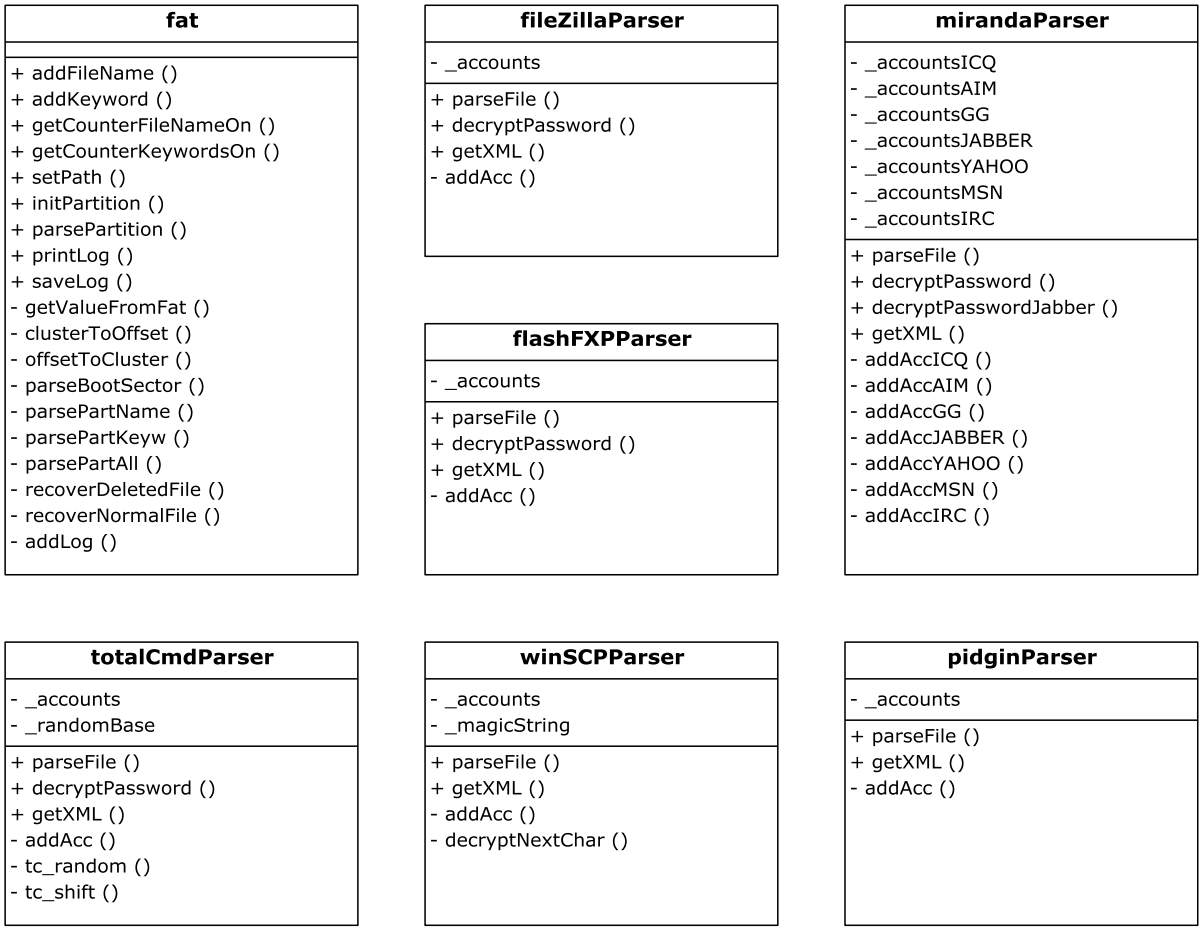
Druhú časť aplikácie tvoria triedy, ktorých úlohou je získanie citlivých informácií zo zadaného súboru. Triede bude predaná cesta k súboru, trieda tento súbor spracuje a uchová získané informácie. Pokiaľ súbor obsahuje zašifrované údaje, musí trieda obsahovať metódu na ich odšifrovanie. Pre každú skúmanú aplikáciu bude existovať jedna trieda. Triedy sa budú líšiť v implementácii metód získavajúcich citlivé informácie a v implementácií metód dešifrujúcich šifrované informácie. Každá trieda bude obsahovať metódu, ktorá umožní získanie všetkých uložených informácií. Výstupom metódy bude reťazec vo formáte XML[6]. Formát som vybral na základe čitateľnosti ľudským okom a zároveň pre existenciu množstva knižníc pre prácu s ním. Zvažoval som aj formát CSV[7], ktorý však nie je úplne vyhovujúci pre tento účel, nakoľko CSV je určený pre tabuľkové údaje, no triedy získavajú zo súborov rôzne typy údajov, ktoré je zložité uložiť vo forme tabuľky. Diagram tried je na Obrázok 5.1.

Celá aplikácia bude riadená z funkcie main. Na začiatku zavolá funkciu, ktorej predá parametre, s ktorými bola aplikácia spustená. Funkcia parametre spracuje a nastaví príslušné premenné. Po spracovaní parametrov je vytvorený objekt na prácu so súborovým systémom. Následne mu je predaná cesta k oddielu získaná zo zadaných parametrov. Rovnako mu je nastavený výstupný adresár a podľa skúmaných aplikácií názvy súborov a kľúčové slova. Po skončení vyhľadávania sa obnovené súbory predajú príslušným triedam na získanie citlivých informácií. Nakoniec sa z každej triedy získa XML reťazec s údajmi a uloží sa do výstupného súboru "FatPasswordParserAcc.xml". Umiestnenie výstupného súboru je rovnaké ako umiestnenie obnovených súborov.

[6] Extensible Markup Language (www.w3.org/XML)
[7] Comma Separated Value (www.tools.ietf.org/html/rfc4180#page-2)

Tento odsek obsahuje zoznam a vysvetlenie parametrov aplikácie:

- **-noftp** potlačí vyhľadávanie účtov FTP klientov
- **-noim** potlačí vyhľadávanie účtov klientov rýchlych sprav
- **-noprint** potlačí výpis na štandardný výstup
- **-h** alebo **--help** vypíše nápovedu
- **--output=<adresa>** adresa výstupného adresára
- **--input=<adresa>** jediný povinný parameter, ktorý slúži na zadanie adresy vstupného oddielu (pre logicky disk \\\\.\\<pismeno disku>:)



Obrázok 5.1 Diagram tried navrhovanej aplikácie

6 Implementácia aplikácie

Pri výbere programovacieho jazyka som kládol dôraz na existenciu funkcií, ktoré umožňujú bezproblémový prístup k diskom na úrovni sektorov v prostredí Windows. Z vyhovujúcich jazykov som sa rozhodoval medzi Visual C++ a jazykom Python vzhľadom na čiastočnú znalosť oboch jazykov. Výhodou jazyka Python je multiplatformovosť nakoľko sa jedná o interpretovaný jazyk. Oproti tomu Visual C++ poskytuje väčšiu rýchlosť, ktorú považujem pre tento projekt za výhodnejšiu, a preto som sa rozhodol aplikáciu implementovať pravé v tomto jazyku. Všetky triedy sú navrhnuté a implementované tak, aby ich bolo možné použiť samostatne aj v iných projektoch.

6.1 Implementácia triedy fat

Ako prvú som implementoval triedu fat, ktorá pracuje so súborovým systémom FAT32. Po otvorení oddielu sa trieda pokúsi prečítať zavádzaci sektor. Nakoľko USB Flash disky umožňujú čítať len bloky dát o veľkosti násobkov veľkosti sektorov, čítanie prebieha v cykle. Veľkosť bloku pri čítaní je na začiatku nastavená na 512 bajtov. Pri neúspešnom čítaní sa táto hodnota zdvojnásobuje až do hodnoty 4096 bajtov, čo je maximálna veľkosť sektoru podľa špecifikácie Microsoft. Pri presiahnutí maximálnej hodnoty aplikácia končí s chybou. Ak je čítanie úspešné, trieda skontroluje typ súborového systému a získa informácie zo zavádzacieho sektora.

Pred vyhľadávaním a obnovou súborov bolo nutné implementovať pomocné funkcie na prevod čísla klastra alebo sektora na posunutie k začiatku oddielu v bajtoch. Ďalej bola potrebná funkcia pracujúca so štruktúrou fat, ktorá vracia hodnotu na základe zadaného čísla klastra. Vzhľadom na veľkosť fat štruktúry nie je v pamäti uchovávaná celá, ale vždy sa načíta iba sektor obsahujúci potrebný záznam. Obsah väčšiny súborov je uložený za sebou, a preto je veľká pravdepodobnosť, že ďalší požadovaný záznam bude v rovnakom sektore ako predošlý. Za účelom ušetrenia počtu čítaní a tým zrýchlenia aplikácie je posledný načítaný sektor uchovaný v pamäti. Ak uložený sektor neobsahuje hľadaný záznam, načíta sa nový klaster, ktorý je uložený namiesto súčasného. Funkcia vracia celú 32 bitovú hodnotu aj s vrchnými dvoma vyhradenými bitmi.

Následne som pristúpil k implementácii metódy vyhľadávajúcej súbory na základe kľúčových slov. Metóda slúži na vyhľadávanie súborov, ktoré začínajú s pevne daným reťazcom, podľa ktorého sa dajú identifikovať. Prechádzané sú všetky klastre v dátovej oblasti. Kľúčové slová sú vyhľadávané na daných pozíciách v prvom sektore klastra, čo umožňuje väčšiu rýchlosť ako pri prehľadávaní celého klastra. Ak nastane zhoda, metóda zistí fat hodnotu aktuálneho klastra, podľa ktorej vyberie metódu na obnovu zmazaného alebo nezmazaného súboru. Jednotlivé kľúčové slova sú ukladané v štruktúre, ktorá obsahuje kľúčové slovo, pozíciu, názov a príponu výstupného súboru, predpokladanú veľkosť súboru a počet nájdených súborov. Ak je predpokladaná veľkosť súboru nastavená na nulu, obnoví sa iba aktuálny klaster.

Druhá vyhľadávacia metóda slúži na vyhľadávanie podľa názvu súboru. Existujú dva spôsoby ako tento problém riešiť. Prvý je prechádzať adresárovú štruktúru od koreňového adresára. Druhým spôsobom je prechádzanie všetkých klastrov a identifikovanie adresárových záznamov v nich. Prvý spôsob je rýchlejší, no naproti tomu druhý umožňuje vyhľadanie adresárov, na ktoré neexistuje odkaz z adresárovej štruktúry. Pri vyhľadávaní zmazaných súborov je vhodnejší druhý postup. Metóda v prípade nájdenia adresára volá metódu na spracovanie adresárov. Tá načíta celý adresár a postupne prechádza adresárové záznamy a porovnáva názvy. Ak nastane zhoda so zadaným názvom, z daného

adresárového záznamu získa potrebné informácie na obnovu súboru. Pri zmazaných súboroch kontroluje horné dva bajty adresy prvého klastra na nulu. V prípade zhody je metóda na obnovu zmazaných súborov volaná s pôvodnou adresou a následne s adresou, kde horné dva bajty sú nahradené podľa adresy aktuálneho adresára.

Princíp oboch metód som spojil do jednej, ktorá je použitá, ak sú zadané kľúčové slova aj názvy súborov. Metóda vtedy obmedzí prechod dátovou oblasťou na jeden namiesto dvoch.

Po vyhľadaní je nutné súbory obnoviť. Obnovu súborov som rozdelil do dvoch metód: na obnovu nezmazaných súborov a na obnovu zmazaných súborov. V prvom prípade sa postupne prechádza reťazcom klastrov a ich obsah sa zapisuje do výstupného súboru. V druhom prípade nie je možné získať reťazec klastrov, a preto je obnova celého súboru možná iba ak poznáme veľkosť súboru. V opačnom prípade sa obnoví iba prvý klaster. Obnovenie sa začína na prvom klastry a za ďalší klaster s obsahom sa určí nasledujúci nealokovaný klaster. Počet obnovených klastrov závisí od veľkosti súboru. Po obnovení súboru sa v oboch metódach zapíšu informácie do loggera.

Logger je implementovaný ako vektor štruktúr, ktoré obsahujú reťazec s názvom výstupného súboru a celočíselný typ uchovávajúci posunutie nájdeného súboru k začiatku oddielu v bajtoch. Implementované boli metódy pre vloženie záznamu, vypísanie záznamov a uloženie do súboru.

6.2 Implementácia tried získavajúcich informácie

Patria sem triedy fileZillaParser, flashFXPParser, mirandaParser, pidginParser totalCmdParser a winSCPParser. Všetky triedy obsahujú metódu získavajúcu informácie zo súboru. Implementácia sa líši na základe formátu súboru danej aplikácie.

Pri XML súboroch nebolo možné použiť knižnice pracujúce s týmto formátom. Dôvodom je možnosť poškodenia vstupného súboru. Implementoval som riešenie, ktoré v súboroch vyhľadáva dané značky a z nich získava informácie. Navzdory princípu XML jazyka pracujem s nemenným poradím značiek, vďaka čomu zámedzím nesprávnemu spojeniu nesúvisiacich informácií.

Ďalším typom sú textové súbory, kde je každá jedna informácia uložená na samostatnom riadku. V tomto prípade je súbor čítaný po riadkoch a každý riadok je následné spracovaný.

Posledným formátom je upravená databáza ICQ používaná aplikáciou Miranda. Informácie sú uložené v štruktúrach, ktoré sa pre jednotlivé protokoly líšia. Najprv je vyhľadané kľúčové slovo "AM_BaseProto" ktorým začína každá štruktúra. Následne je zistený protokol, na základe ktorého sa získavajú ďalšie informácie.

Dešifrovanie informácií pre každú aplikáciu bolo implementované podľa algoritmov popísaných v kapitole 3.

Získané informácie si trieda ukladá vo vektore štruktúr. Tie trieda poskytuje pomocou metódy, ktorá ich prevedie do formátu XML a vráti ako reťazec.

7 Testovanie

V tejto kapitole postupne popíšem postup testovania funkčnosti tried. Následne sa zameriam na testovanie použiteľnosti celej aplikácie. Tato časť ďalej obsahuje porovnanie aplikácie s už existujúcimi riešeniami.

Pri oboch fázach testovania som používal aplikáciu Active Disk Editor[8], ktorá umožňuje zobrazenie a editáciu oddielov na úrovni jednotlivých bajtov. Ďalej boli použité nástroje systému Linux: dd na vytvorenie súboru s oddielom, príkazom `dd if=/dev/random of=fat.img bs=1m count=3000` a nastroj mkdosfs, pomocou ktorého bol následne oddiel naformátovaný na súborový systém FAT32.

7.1 Testovanie funkčnosti tried

Testovanie funkčnosti tried prebiehalo vo dvoch fázach. Prvá fáza prebiehala počas implementácie jednotlivých metód. V tejto fáze bolo odhalených a opravených najviac chýb. V druhej fáze bola trieda otestovaná ako celok.

7.1.1 Testovanie triedy fat

Ide o najrozsiahlejšiu triedu v projekte, preto aj jej testovanie zabralo najviac času. Najprv bolo potrebné otestovať funkčnosť pomocných metód, pred ich použitím v ďalších metódach. Túto časť bolo možné automatizovať postupným predávaním hodnôt a porovnaním výstupu s očakávaným výsledkom.

Pri testovaní metódy vyhľadávajúcej súbory na základe kľúčových slov boli na oddiely vytvorené adresáre rôznej hĺbky. Do adresárov bolo nahraných niekoľko náhodných súborov a pevný počet hľadaných súborov s kľúčovými slovami. Po skončení vyhľadávania bol počet nájdených súborov porovnaný s počtom nahraných súborov s kľúčovými slovami. Následne bolo niekoľko súborov a adresárov z oddielu zmazaných a testovanie bolo spustené znovu.

Testovanie metódy vyhľadávajúcej podľa názvu prebiehalo podobným postupom. Rozdiel bol v mazaní súborov, ktoré bolo uskutočnené vo dvoch operačných systémoch: Windows XP a Windows 7. Dôvodom je odlišný postup mazania. Bolo nutné otestovať spôsob identifikácie adresára v klastroch. Následne bola overená funkčnosť metódy spracúvajúcej adresár, najmä algoritmus načítania celého adresára, konkrétne ide o spracovanie adresárových záznamov, porovnanie názvov, získanie správnej adresy a veľkosti súboru. Primárne pri zmazaných adresároch bolo nutné zistiť či adresár pokračuje v nasledujúcom klastry. Nasledovalo súvisiace spracovanie záznamov obsahujúcich dlhý názvov súboru, výpočet kontrolného súčtu, načítanie celého názvu a prevod UTF-16 znakov na UTF-8.

Pri testovaní obnovy nezmazaných súborov bol dôraz kladený na prechádzanie reťazcom klastrov a obnovu na základe veľkosti odlišnej od reálnej veľkosti súboru. Pri zmazaných súboroch bolo testované určovanie ďalšieho klastra s obsahom súboru.

Nakoniec bola otestovaná trieda ako celok na niekoľkých vytvorených obrazoch oddielov a následne na reálne používaných USB flash kľúčoch. Pri testovaní som zistil, že ak nejaká aplikácia číta z USB kľúča, systém zablokuje prístup k oddielu na úrovni sektorov.

[8] www.disk-editor.org

7.1.2 Testovanie tried získavajúcich informácie

Pri testovaní týchto tried bolo dôležité otestovať metódy na získavanie informácií zo súboru a jednotlivé dešifrovacie algoritmy. V prvom prípade boli metóde predávané súbory danej aplikácie. Nasledovali súbory, ktorým boli zmazané alebo prepísané rôzne ich časti, čím som sa snažil napodobniť chyby, ktoré môžu vzniknúť pri obnove súborov. Následne som súbory spracoval sám a moje výsledky porovnal s výstupom metódy. V prípade XML formátu boli často spojené nesúvisiace informácie a preto som sa rozhodol implementovať vyhľadávanie s pevným poradím XML značiek, ako je popísané v kapitole 6.2.

Overenie správnosti implementácie dešifrovacích algoritmov bolo možné automatizovať. Metóde boli postupne predávané zašifrované údaje z vopred pripraveného zoznamu a výstup bol porovnaný s očakávaným výsledkom. V zozname sa nachádzali aj nekompletné alebo inak poškodené heslá. V tomto prípade úspešnosť dešifrovania takto pozmeneného údaju záležala na algoritme šifrovania.

7.2 Testovanie použiteľnosti aplikácie

Moja aplikácia (ďalej pod názvom FatPasswordParser) bola porovnávaná s existujúcimi riešeniami na obnovu zmazaných dát, PhotoRec a Recuva. Pri testoch bol použitý obraz oddielu, v ktorom boli pred každým testom ručne zmenené údaje, potrebné pre daný test.

Pri prvom teste bola v prázdnom oddiely vytvorená adresárová štruktúra, do ktorej boli nahrané hľadané súbory skúmaných aplikácii. Následne boli súbory zmazané pod systémom Windows XP. V tomto teste sa podarilo všetkým aplikáciám obnoviť súbory.

V druhom teste bol na adresy zamazaných súborov nahraný nový obsah. Recuva ako jediná umožnila obnovu chybných dát.

Tretí test sa líšil v zmazaní celých adresárov. V tomto teste sa aplikáciám PhotoRec a FatPasswordParser podarilo súbory obnoviť. Aplikácia Recuva bola schopná obnovy súborov so zapnutým hlbokým skenovaním.

Na začiatku štvrtého testu bol oddiel zaplnený na 75% svojej kapacity. Následne boli nahrané súbory skúmaných aplikácií. Všetky hľadané súbory mali vrchné dva bajty adresy prvého klastra súboru rovnaké ako adresár, v ktorom sa nachádzali. Postupne boli všetky zmazané pod systémom Windows XP. V tomto teste sa podarilo všetkým aplikáciám obnoviť súbory.

V piatom teste bol použitý upravený obraz zo štvrtého testu. Horné dva bajty adresy zmazaných súborov boli vynulované. Týmto som sa snažil simulovať mazanie súborov pod systémom Windows 7. Aplikácií PhotoRec sa podarilo obnoviť iba súbory s jej známym formátom. Recuva obnovila súbory s chybným obsahom. Aplikácií FatPasswordParser sa podarilo obnoviť všetky súbory, no zároveň obnovila aj súbory s chybným obsahom. Počet chybných súborov závisel od počtu nealokovaných klastrov na adresách s nulovými hornými dvoma bajtmi.

V poslednom, šiestom teste bol oddiel pripravený ako v teste štyri. Rozdiel bol v adresách nahraných súborov skúmaných aplikácií. Vrchné dva bajty adresy súboru boli rôzne od adresy adresára, ale stále zostali nenulové. Aplikácie PhotoRec a Recuva si viedli rovnako ako v predošlom teste. Aplikácii FatPasswordParser sa podarilo obnoviť iba súbory vyhľadávané podľa kľúčového slova. V ostatných prípadoch obnovila chybné súbory. Počet záležal od alokovanosti klastrov na daných adresách.

	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5	TEST 6
PhotoRec	Všetko	Nič	Všetko	Všetko	Čiastočne	Čiastočne
Recuva	Všetko	Chybne	Všetko	Všetko	Chybne	Chybne
FatPasswordParser	Všetko	Nič	Všetko	Všetko	Všetko + chybne	Čiastočne + chybne

Tabuľka 7.1 Prehľad výsledkov testov

Porovnanie získavania informácií z obnovených súborov bol možný iba s existujúcimi riešeniami, ktoré umožňovali zadanie vstupného súboru. V prípade nepoškodených súborov dosiahli aplikácie rovnaké výsledky. Oproti tomu pri poškodených súboroch väčšina existujúcich riešení zlyhala, nakoľko neboli navrhnuté na spracovanie poškodených súborov. Aplikácii FatPasswordParser sa podarilo získať väčšinu informácii nachádzajúcich sa v takomto súbore.

8 Záver

Výsledkom práce je funkčná aplikácia získavajúca citlivé informácie zo súborového systému FAT32. Behom implementácie som sa snažil vylepšiť aplikáciu tak, aby vyhľadala a obnovila čo najväčší počet hľadaných súborov aj za cenu zvýšenia chybných obnovení. Zároveň som sa snažil o zrýchlenie behu aplikácie, čo som dosiahol znížením počtu čítaní z oddielu. Aplikácia bola podrobená rôznym testom, ktorých výsledky sú uvedené v kapitole 7.2. Na výsledkoch je vidieť, že aplikácia si vedie rovnako a niekedy dokonca lepšie ako existujúce riešenia. Silnou stránkou aplikácie pri obnove súborov oproti konkurencii je zohľadnenie nového spôsobu mazania súborov v systémoch Windows Vista a novších, kedy sú vynulované najvrchnejšie dva bajty adresy. Pri získavaní informácií z poškodených súborov aplikácia dosahuje lepšie výsledky ako už existujúce riešenia.

Behom vypracovávania práce som si preštudoval princípy fungovania súborového systému FAT32, ktoré som popísal v kapitole 2. Rozšíril som si znalosti jazyka Visual C++ a vyskúšal prácu vo vývojovom prostredí Visual studio.

8.1 Možnosti ďalšieho rozšírenia

Navrhnutú aplikáciu je možné ďalej rozšíriť o ďalšie podporované aplikácie, z ktorých získava citlivé informácie. Potrebné je iba vytvoriť triedu získavajúcu informácie zo súboru danej aplikácie. O vyhľadanie súborov sa postará trieda fat, ktorej stačí predať názov alebo kľúčové slovo, podľa ktorého má vyhľadávať.

Vďaka zvolenému programovaciemu jazyku je možné do aplikácie implementovať jednoduché grafické rozhranie. Toto rozšírenie som chcel implementovať spolu s textovým režimom. Žiaľ jazyk neumožňuje kombináciu oboch rozhraní v jednom spustiteľnom súbore.

Ďalším možným smerom, ktorým by sa mohla aplikácia rozšíriť, je podpora súborového systému NTFS. Toto rozšírenie by znamenalo reimplementáciu najrozsiahlejšej triedy v projekte, no poskytovalo by komplexné riešenie na obnovu súborov v operačných systémoch Windows.

Literatúra

[1] CASEY, Eoghan. Digital evidence and computer crime: forensic science, computers and the Internet. 2.vyd. San Diego, Calif.: Academic Press, 2004, 690 p. ISBN 01-216-3104-4

[2] CARRIER, Brian. File system forensic analysis. Upper Saddle River: Addison-Wesley, 2005, 569 s. ISBN 03-212-6817-2

[3] Microsoft Corporation. FAT: General Overview of On-Disk Format. [online], 2000. Dostupné z: www.msdn.microsoft.com/en-us/windows/hardware/gg463080.aspx

[4] GIAMPAOLO, Dominic. *Practical file system design with the BE file system*. San Francisco: Morgan Kaufmann Publishers, c1999, x, 237 p. ISBN 15-586-0497-9

[5] PIDGIN CONTRIBUTORS. *Pidgin, the universal chat client* [online]. 2009 [cit. 2015-05-19]. Dostupné z: www.pidgin.im

[6] MIRANDA IM. *Miranda IM - Home of the Miranda IM client. Smaller, Faster, Easier* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.miranda-im.org

[7] TALEKAR, Nagareshwar. Exposing the Password Secrets of Miranda. *SecurityXploded* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.securityxploded.com/ftp-password-decryptor.php

[8] FILEZILLA. *FileZilla - The free FTP solution* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.filezilla-project.org

[9] GHISLER, Christian. *Total Commander* [online]. 2014 [cit. 2015-05-19]. Dostupné z: www.ghisler.com

[10] FITZL, Csaba. Total Commander FTP Password Recovery Tool. *Csabyblog* [online]. 2013 [cit. 2015-05-19]. Dostupné z: www.sites.google.com/site/csabyblog/home/tcpwrecovery

[11] *WinSCP* [online]. 2014 [cit. 2015-05-19]. Dostupné z: www.winscp.net

[12] LIEB, Jonas. WinSCP Session Password Decryption. *JonasLieb* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.jonaslieb.com/blog/2015/02/20/winscp-session-password-decryption-part-2

[13] *FlashFXP - Secure FTP Client Software for Windows. Upload, Download, and Synchronize your files.* [online]. 2014 [cit. 2015-05-19]. Dostupné z: www.flashfxp.com

[14] HALL, Benjamin. VB.Net FlashFXP Password Decryption. *PlanetSourceCode* [online]. 2004 [cit. 2015-05-19]. Dostupné z: www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=2152&lngWId=10

[15] ELCOMSOFT CO. LTD. *Password recovery, forensic, forensics, system and security software from ElcomSoft* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.elcomsoft.com

[16] *NirSoft - freeware utilities: password recovery, system utilities, desktop utilities* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.nirsoft.net

[17] RIXLER SOFTWARE. *Rixler password recovery tools* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.rixler.com

[18] TALEKAR, Nagareshwar. FTP Password Decryptor. *SecurityXploded* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.securityxploded.com/ftp-password-decryptor.php

[19] GRENIER, Christophe. *CGSecurity - Data recovery: TestDisk & PhotoRec* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.cgsecurity.org

[20] PIRIFORM LTD. *Piriform* [online]. 2015 [cit. 2015-05-19]. Dostupné z: www.piriform.com