VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ROZŠÍŘENÍ SYSTÉMU PRO SHLUKOVOU ANALÝZU SOUBORŮ

IMPROVING AN EXISTING SYSTEM FOR CLUSTERING OF FILES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATÚŠ JASNICKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Jasnický Matúš**
Program: Informační technologie
Název: **Rozšíření systému pro shlukovou analýzu souborů**
Improving an Existing System for Clustering of Files
Kategorie: Překladače

Zadání:

1. Seznamte se se službou Clusty, která ve společnosti Avast slouží ke shlukové analýze souborů podle jejich podobnosti.
2. Po domluvě s vedoucím vyberte několik typů souborů, které Clusty nepodporuje (např. PDF, LNK). Tento typ souborů si detailně nastudujte, především z pohledu identifikace, strojové analýzy a extrakce unikátních vlastností.
3. Navrhněte rozšíření služby Clusty o rozpoznání těchto typů souborů, jejich analýzu, extrakci vlastností, shlukové analýzy a výpočet společných vlastností souborů ve shluku.
4. Rozšíření navržená v předchozím bodě implementujte.
5. Vytvořené řešení důkladně otestujte sadou jednotkových a integračních testů.
6. Zhodnoťte svou práci a diskutujte možný budoucí vývoj.

Literatura:

- B. Everitt, S. Landau, M. Leese, D. Stahl: Cluster Analysis, Wiley (2011, 5th edition), ISBN 978-0470749913
- J. Whitington: PDF Explained: The ISO Standard for Document Exchange, O'Reilly Media (2011), ISBN 978-1449310028
- Interní dokumentace společnosti Avast.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání a rozpracování čtvrtého bodu.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Křivka Zbyněk, Ing., Ph.D.**
Konzultant: Zemek Petr, Ing., Avast
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 16. října 2018

Abstrakt

Cieľom tejto práce je rozšírenie existujúceho nástroja Clusty, vyvinutého spoločnosťou Avast Software pre zhlukovanie rôznych typov súborov, o nové typy súborov - PDF a LNK (MS Windows odkaz). Dáta potrebné ku zhlukovaniu sa získavajú statickou analýzou súborov nástrojmi tretích strán. Práca popisuje aj výber vhodných atribútov a metód na zhlukovanie. Všetky časti práce boli otestované a nasadené do produkčnej verzie.

Abstract

The aim of this work is to extend the existing tool named Clusty — developed by Avast Software to cluster various file types — with new file types, namely PDF and LNK (MS Windows shortcut). Data needed for clustering are obtained by static analysis of files by third-party tools. The work also describes the selection of suitable attributes and methods for clustering. All parts of the work have been tested and deployed into production.

Kľúčové slová

zhluková analýza, statická analýza, PDF, LNK, Portable Document Format, Shell Link, Windows Shortcut

Keywords

cluster analysis, static analysis, PDF, LNK, Portable Document Format, Shell Link, Windows Shortcut

Citácia

JASNICKÝ, Matúš. *Rozšíření systému pro shlukovou analýzu souborů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

Rozšíření systému pro shlukovou analýzu souborů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Zbyňka Křivky Ph.D. Ďalšie informácie mi poskytol Ing. Petr Zemek. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matúš Jasnický
14. mája 2019

Poďakovanie

Chcel by som sa poďakovať svojmu vedúcemu Zbyňku Křivkovi ako aj Petrovi Zemkovi za vedenie a cenné pripomienky k mojej práci.

Obsah

Kapitola 1

Úvod

Škodlivý softvér (angl. *malicious software*, alebo skrátene *malware*) je akýkoľvek počítačový program, ktorý je navrhnutý so zámerom kompromitovať systém a vykonať rôzne typy podvodov. Prvý počítačový vírus zvaný *Brain* sa objavil už v roku 1986. Odvtedy sa motivácia vývoja zmenila zo zábavy na finančnú. Malware sa používa napríklad na odosielanie nevyžiadaných e-mailov, na vykonávanie webových podvodov a na krádež osobných údajov, ako sú napríklad informácie o kreditných kartách. Škodlivý softvér je kategorizovaný do niekoľkých rodín podľa jeho funkcií a spôsobu šírenia, napríklad trójsky kôň, vírus, červ, a iné. Zhlukovanie škodlivého softvéru je dôležitou výzvou pre anti-vírusových spoločnosti. Ak by vzorky malware neboli zhlukované, analytik by musel analyzovať každú jednu vzorku manuálne, čo kvôli ich obrovskému množstvu nie je možné. Našťastie, mnoho škodlivých vzoriek sa opakuje s malými variáciami ich kódu. Deje sa to z dôvodu, že mnoho z nich je automaticky generovaných pomocou rôznych nástrojov. Na zoskupenie podobných vzoriek do rodín sú potrebné zhlukovacie metódy a následne postačí analyzovať len niekoľko vzoriek z každého zhluku [2, 20].

Jednou z anti-vírových spoločností je aj spoločnosť Avast Software. Denne prijme státisíce nových vzoriek, ktoré musí preskúmať. Ako riešenie vyššie spomínaného problému spoločnosť vyvinula nástroj *Clusty*, ktorého úlohou je prvotná analýza prijatých vzoriek a ich zatriedenie do skupín (zhlukov) podľa ich vzájomnej podobnosti.

Prijaté vzorky sú zhlukované na náklade vlastností (atribútov) získaných statickou, prípadne i dynamickou analýzou vzoriek. Vzorky môžu byť rozličných typov súborov, od binárnych spustiteľných súborov, cez archívy až po textové dokumenty. Niektoré vlastnosti sú spoločné pre všetky prijaté vzorky, napríklad veľkosť. Niektoré sú spoločné iba pre určitú kategóriu, napríklad vstupný bod pre binárne súbory. Väčšina vlastností je však unikátna pre daný typ. Z tohoto dôvodu je potrebné pre jednotlivé typy súborov používať špecifické analyzátory a špecifické metódy na zhlukovanie. Podpora pre rôzne typy je preto pridávaná postupne. Nástroj Clusty aktuálne podporuje formáty PE, ELF, Mach-O, APK, DEX, archívy a dokumenty Office. Bližšie je popísaný v kapitole 2.

Cieľom tejto práce je rozšíriť spomínaný nástroj Clusty o doposiaľ nepodporované typy súborov. Konkrétne boli vybrané dva formáty - PDF a LNK, ktoré sú popísané v kapitole 3 a 4. Pre každý nový formát je potrebné vybrať vhodné vlastnosti a analyzátor, ktorý ich vie extrahovať. Nástroj Clusty musí denne spracovať veľké množstvo vzoriek, ktoré môžu byť často poškodené (či už úmyselne alebo neúmyselne), a preto je potrebné, aby bol analyzátor rýchly a spoľahlivý. Vybrané zhlukovacie metódy, odvodené z extrahovaných vlastnosti, by mali byť schopné vytvárať čo najkvalitnejšie zhluky podobných vzoriek a rovnakých rodín.

Bol vytvorený návrh rozšírenia nástroja Clusty o oba spomínané formáty. Bližšie je popísaný v kapitole 5. Návrh zahŕňa popis existujúcich i vybraných nástrojov tretích strán, určených na statickú analýzu pridávaných formátov, a ich testovanie. Ďalej popisuje návrh implementácie nových typov súborov a štruktúru nástroja Clusty. Taktiež je popísaný výber vhodných atribútov.

Výsledná implementácia je popísaná v kapitole 6. Okrem iného popisuje atribúty, ktoré vznikajú kombináciou extrahovaných vlastností a okolnosti ich vzniku. Počas vývoja bolo potrebné nové časti podrobne testovať. Taktiež boli testované vybrané atribúty na zhlukovanie, prípadne ich kombinácie, a výsledné zhluky. Testovanie je popísané v kapitole 7. Metódy použité na zhlukovanie a zhodnotenie sú popísané v kapitole 8.

Rozšírenie nástroja Clusty, ktoré bolo v tejto práci navrhnuté, bolo nasadené do reálnej prevádzky a denne spracúva desaťtisíce prijatých vzoriek.

Kapitola 2

Nástroj Clusty

Kapitola vychádza z internej dokumentácie spoločnosti Avast Software [8] (ďalej ako Avast).

Spoločnosť Avast denne prijme 300 000 – 1 000 000 nových vzoriek (súborov), ktoré musí analyzovať. Niektoré vzorky môžu obsahovať škodlivý software a niektoré zasa nie. Spracovať každú vzorku zvlášť by zaberalo príliš veľa času a bolo by to veľmi neefektívne najmä v prípade opakovania sa vzoriek.

Škodlivé vzorky sú obecne rozdelené do rodín, podľa funkcií a spôsobu šírenia, a tak sa v jednej rodine sa nachádzajú vzorky s podobným správaním. Často sú generované rôznymi nástrojmi, a tak sa líšia iba drobnými úpravami kódu. Na zoskupenie podobných vzoriek rovnakých rodín je možné použiť zhlukovanie a následne analyzovať jednotlivé zhluky [2].

Z tohoto dôvodu firma Avast vyvinula a používa nástroj *Clusty*. Jeho úlohou je prvotná analýza prijatých vzoriek a rozdelenie týchto vzoriek do skupín, tzv. zhlukov. Pre zhlukovanie sú použité najčastejšie informácie získané statickou analýzou pomocou nástrojov vyvíjaných spoločnosťou Avast, alebo z webovej služby VirusTotal[1]. Pri niektorých formátoch sa používajú aj informácie získané dynamickou analýzou.

Clusty je implementovaný v jazyku Python s využitím MongoDB – multiplatformná NoSQL databáza, ktorá na rozdiel od klasických relačných databáz s tabuľkami ukladá štruktúrované dáta ako JSON dokumenty s dynamickými schémami. Časť nástroja týkajúca sa priamo zhlukovej analýzy je z dôvodu veľkej pamäťovej a časovej náročnosti implementovaná v C++.

2.1 Čo je zhluková analýza

Zhlukovanie je proces rozdeľovania objektov do tried (zhlukov) na základe podobnosti objektov. Triedy sú potom tvorené objektami, ktoré sú si navzájom podobné a zároveň nie sú podobné objektom v iných triedach. Podobnosť je určená na základe hodnôt atribútov týchto objektov. Zhlukovanie môže byť využité ako predspracovanie dát pre ďalšie algoritmy, predovšetkým pre algoritmy pre klasifikáciu a charakterizáciu, ktoré následne pracujú nad vytvorenými triedami.

Z hľadiska strojového učenia ide o učenie bez učiteľa. Zhlukovanie nevyžaduje žiadne preddefinované triedy ani žiadnu trénovaciu množinu príkladov [22].

[1] https://www.virustotal.com/

2.2 Postup zhlukovej analýzy

Analýza prebieha v dvoch fázach.

Fáza 1

V prvej fáze prebieha identifikácia formátu súboru vzoriek určených na zaradenie do zhlukov. Každý z formátov je možné identifikovať určitou sekvenciou úvodných bajtov alebo jednou z jeho vlastností. Identifikácia v Clustym prebieha postupným testovaním, či vzorka spĺňa kritéria identifikácie niektorého zo známych formátov. Ak vzorka napríklad obsahuje v úvodných dvoch bajtoch znaky `MZ`, je identifikovaná ako formát PE.

Vzorky pochádzajú z rozličných zdrojov (používatelia anti-vírusu Avast, VirusTotal, atď.) a sú identifikovateľné, v zmysle identifikácie konkrétnej vzorky, nie jej typu, pomocou hash reťazca SHA-256. Podporované formáty súborov sa následne analyzujú špecifickými nástrojmi. Medzi aktuálne podporované formáty patria formáty PE, ELF, Mach-O, APK, DEX, archívy a dokumenty Office. Nezávisle na rozpoznaní formátu sa pre každý súbor získajú všeobecne dostupné informácie ako napr. veľkosť, hash, fuzzy hash (implementácia v ssdeep[2]) a detekcie z VirusTotal. Získané informácie sa bez ďalších úprav uložia do databázy. Vzorky sú rozdelené do kategórii podľa rozpoznaného formátu súboru. Tie, ktoré sa nepodarilo rozpoznať sú zaradené do všeobecnej kategórie *others*.

Fáza 2

V druhej fáze prebieha samotné zhlukovanie pre každú kategóriu zvlášť. Extrahované informácie sú modifikované (typicky pomocou hash metódy) s cieľom znížiť pamäťovú náročnosť. To je výhodné najmä pri dlhých reťazcoch a zložených atribútoch, pretože je možné zistiť odlišnosť pri začiatku porovnávania atribútu.

Charakteristiky (atribúty) každej kategórie sú zoradené podľa vhodnosti na zhlukovanie určenej experimentami. Vzorka sa zaraďuje do zhluku vždy podľa najprioritnejšej z jej charakteristík. Ak vzorku nebolo možné zaradiť na základe žiadnej z charakteristík, vzorka sa označí ako nezaradená.

Po zhlukovaní všetkých vzoriek sa pre každý zo zhlukov vypočíta podobnosť jednotlivých charakteristík a výsledky sa uložia do databázy. U každého zhluku je potom možné zobraziť na základe ktorej charakteristiky bol vytvorený a miery podobnosti zvyšných charakteristík.

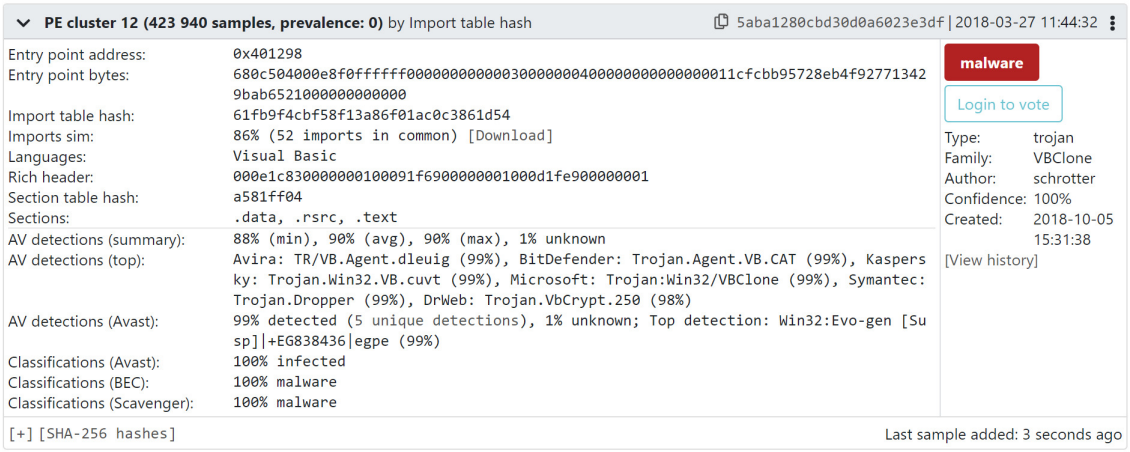
Nástroj je vytvorený tak, aby dokázal spracovať niekoľko stotisíc vzoriek každý deň. Pri procese pridávania nových formátov bol využitý ďalší jeho mód, v ktorom je možné spustiť zhlukovanie nad jedným priečinkom.

2.3 Reprezentácia zhlukov

Na obrázku 2.1 je možné vidieť príklad zhluku zobrazeného vo webovom rozhraní nástroja Clusty. Konkrétne sa jedná o zhluk PE súborov. Zhluk sa skladá (zhora nadol) z hlavičky, zoznamu spoločných charakteristík a detekčných štatistík. Na pravej strane sa nachádza klasifikácia.

Hlavička Obsahuje základné informácie o zhluku. Ako prvé je možné vidieť kategóriu (PE), počet vzoriek v zhluku (423 940) a rozšírenosť (angl. *prevalence*) ktorá označuje po-

[2] https://ssdeep-project.github.io/ssdeep/index.html



Obr. 2.1: Zobrazenie zhluku v nástroji Clusty

čet užívateľov nástroja Avast, ktorý s danou vzorkou prišli do kontaktu. V tomto prípade žiaden užívateľ z Avastu neprišiel do kontaktu so žiadnou vzorkou z tohoto zhluku, Ďalej je uvedená charakteristika, podľa ktorej vznikol tento zhluk. Tento zhluk vznikol na základe `Import table hash`, čo je hash importovacej tabuľky vzoriek PE. Na pravej strane je unikátny identifikátor zhluku (`5aba1280cbd30d0a6023e3df`) a dátum vytvorenia (2018-03-27 11:44:32).

Spoločné charakteristiky Zhluk je vytvorený na základe jednej charakteristiky, ale môže mať aj viacero spoločných charakteristík. Čím viac charakteristík je spoločných, tým sú si vzorky podobnejšie. V príklade je možné vidieť celkovo 8 spoločných charakteristík, konkrétne sú to:

- `Entry point address` (`0x401298`)
- `Entry point bytes` (`680c504000e8f0f...000000000000`)
- `Import table hash` (`61fb9f4cbf58f13a86f01ac0c3861d54`)
- `Imports sim`
- `Languages` (`Visual Basic`)
- `Rich header`(`000e1c830000000100091f6900000001000d1fe900000001`)
- `Section table hash` (`a581ff04`)
- `Sections` (`.data, .rsrc, .text`)

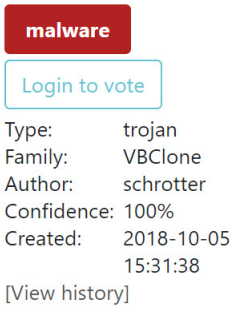
Detekčné štatistiky Detekčné štatistiky ukazujú, čo si Avast (AV) a ostatné anti-vírusy myslia o vzorkách v zhluku.

Klasifikácia Klasifikácia (viď obrázok 2.2) sa skladá z nasledovných častí:

- závažnosť (malware)
- typ (trojan) - závisí od závažnosti

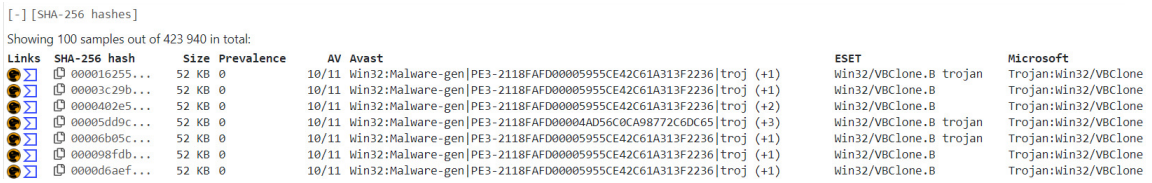
- rodina (VBClone) - názov škodlivého softvéru
- autor (schrotter) - autor ktorý vytvoril klasifikáciu
- miera dôveryhodnosti klasifikácie (100%)
- dátum vytvorenia klasifikácie (2018-10-05 15:31:38)

Pod týmito informáciami je taktiež tlačítko pre zobrazenie histórie klasifikácie.



Obr. 2.2: Zobrazenie časti klasifikácie zhluku v nástroji Clusty

Vzorky Pod zhlukom je možné vidieť 100 náhodných vzoriek, ktoré je možné stiahnuť (viď obrázok 2.3). Každá zo vzoriek obsahuje odkaz na ďalšie služby (napr. VirusTotal), hash identifikujúci vzorku, veľkosť, rozšírenosť, a vybrané výsledky detekcie z Avastu a iných anti-vírusových programov.


[-] [SHA-256 hashes]

Showing 100 samples out of 423 940 in total:

Links	SHA-256 hash	Size	Prevalence	AV	Avast	ESET	Microsoft
	000016255...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+1)	Win32/VBClone.B trojan	Trojan:Win32/VBClone
	00003c29b...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+1)	Win32/VBClone.B	Trojan:Win32/VBClone
	0000402e5...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+2)	Win32/VBClone.B	Trojan:Win32/VBClone
	00005dd9c...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00004AD56C0CA98772C6DC65|troj (+3)	Win32/VBClone.B trojan	Trojan:Win32/VBClone
	00006b05c...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+1)	Win32/VBClone.B trojan	Trojan:Win32/VBClone
	000098fdb...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+1)	Win32/VBClone.B	Trojan:Win32/VBClone
	0000d6aef...	52 KB	0	10/11	Win32:Malware-gen|PE3-2118FAFD00005955CE42C61A313F2236|troj (+1)	Win32/VBClone.B	Trojan:Win32/VBClone

Obr. 2.3: Vzorky a hashe

Kapitola 3

PDF formát

PDF (skratka z angl. *Portable Document Format*) je formát používaný k prezentácii a spoľahlivej výmene dokumentov, ktorý je nezávislý na softvéri, hardvéri aj operačnom systéme. Formát PDF, vyvinutý spoločnosťou Adobe, je od verzie 1.7 (rok 2008) otvorený štandard pod záštitou Medzinárodnej organizácie pre normalizáciu (ISO). PDF súbory môžu obsahovať odkazy a tlačítka, pole formulárov, zvuk, video i elektronický podpis [1]. Aktuálne posledná vydaná verzia je 2.0 (28.07.2017) [10].

PDF, ako aj iné jazyky pre popis stránky (angl. *page description language*), popisujú obsah (text i grafiku) stránky pomocou štruktúrovaných dát. PDF vychádza z programovacieho jazyka PostScript, oproti ktorému má však niekoľko vylepšení a zmien. Narozdiel od jazyka PostScript, PDF neobsahuje riadiace štruktúry a teda už nie je programovací jazyk. Medzi najvýznamnejšie prínosy patrí možnosť uložiť rôzne časti dokumentu do jediného súboru s použitím kompresie, linearizácia (viď sekcia 3.1), konštantný čas pre prístup k ľubovolnej stránke a zabudované fonty.

PDF formát je spätne kompatibilný, tj. program vytvorený pre čítanie súborov verzie 1.7 vie prečítať súbor verzie 1.0, rovnako ako aj program vytvorený pre čítanie súborov verzie 1.0 vie zobraziť súbor verzie 1.7. Kompatibilita je zaistená ignorovaním objektov, ktorým čítačka nerozumie [21].

3.1 Obecná štruktúra

Sekcia vychádza z knihy *PDF Explained* [21].

Typický PDF súbor pozostáva z množstva objektov, viacerých komprimačných mechanizmov, rôznych použitých fontov písma, vektorových či rastrových objektov a metadát. Ďalší obsah môžu tvoriť napr. multimédiá a interaktívne formuláre.

Metadáta Štandardné metadáta, ktoré je možné z dokumentu získať, sú napr. názov, titulok, kľúčové slová atď. Metadáta nemajú vplyv na grafický obsah dokumentu. Od verzie 1.4 je možné uložiť metadáta aj formou XML (angl. *eXtensible Markup Language*) zabudovanou do PDF pomocou Adobe XMP (angl. *eXtensible Markup Platform*).

Interaktívne formuláre Formulár umožňuje užívateľovi vypĺňať textové polia a zaškrtávať ponúkané možnosti. Formulár je následne uložený do dokumentu alebo odoslaný na špecifikovanú URL.

Bezpečnosť PDF môže byť zašifrovaný použitím RC4 alebo AES metódy. Súbor nie je zašifrovaný ako celok, ale šifrujú sa len, až na pár výnimiek, objekty typu stream a textové reťazce. Objektová štruktúra je tak dostupná bez nutnosti dešifrovania, ale obsah dokumentu zostáva chránený. Heslá na rozšifrovanie sú dve, jedno pre vlastníka umožňujúce plnú kontrolu, a druhé pre užívateľa umožňujúce vlastníkom definované úkony. Od verzie 1.3 je možné používať digitálne podpisy na overenie totožnosti používateľa alebo obsahu dokumentu.

Kompresia Kompresia sa podobne ako šifrovanie nevzťahuje na celý súbor, ale na jednotlivé objekty. Vďaka tomu je štruktúra dokumentu dostupná aj bez dekompresie a jednotlivé dekompresie sa vykonávajú len v prípade potreby.

Linearizácia Keďže pre prístup k ľubovolnej stránke je potrebné načítať ako hlavičku na začiatku súboru, tak aj uvádzací slovník na konci, musí byť, napríklad pri čítaní zo vzdialeného serveru, stiahnutý celý súbor. Riešením je linearizácia.

Linearizácia je proces usporiadania objektov v súbore tak, že všetky tie, ktoré sú pre danú stránku potrebné, sú umiestnené v priľahlých polohách a tieto skupiny tvoriace jednotlivé stránky sú taktiež usporiadané. Ďalej je na začiatku súboru, hneď za hlavičkou a prípadnými bajtami označujúcimi binárny súbor, umiestnený špeciálny slovník, ktorý jednak identifikuje súbor ako linearizovaný a jednak obsahuje údaje potrebné k lineárnemu spracovaniu. Za ním sa prípadne môže nachádzať taktiež uvádzací slovník s časťou odkazovacej tabuľky. Tento mechanizmus umožňuje zobraziť ľubovolnú stránku z dokumentu bez nutnosti načítavať celý dokument [19]. V ukážke 3.1 je možné vidieť príklad úvodu takéhoto súboru. Jednotlivé časti budú popísané v ďalšej sekcií.

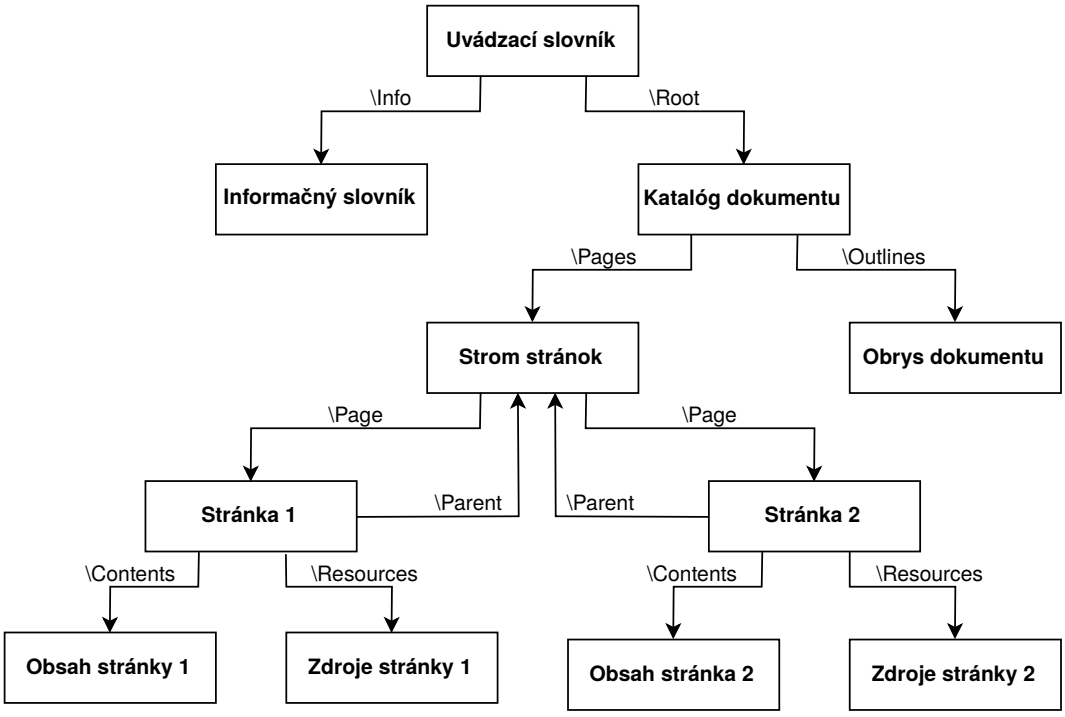
```
%PDF-1.1
%œŒß£
1 0 obj
    <</Linearized 1/L 7546/O 10/E 4079/N 1/T 7272/H [ 456 176]>>
endobj
xref
1 3
0000000015 00000 n
0000000156 00000 n
0000000367 00000 n
trailer
<<  /Root 2 0 R
    /Size 20
    /Info 3 0 R
    /Prev 8243
    /ID[<654745D09345ASQ0V7CB84053W3LFLKD>
    <8SEU8FSURJ8FOUSJRW38FUSOJ3ROF8JS>]
>>
startxref
0
%%EOF
```

Ukážka 3.1: Príklad linearizovaného PDF súboru

3.2 Štruktúra dokumentu

Dokument PDF sa skladá z objektov obsiahnutých v časti tela dokumentu (viď sekcia 3.3.2). Väčšina objektov sú slovníky. Každá stránka dokumentu je reprezentovaná objektom stránky, ktorým je slovník obsahujúci odkazy na obsah stránky. Objekty stránky sú navzájom spojené a vytvárajú strom, ktorý je odkazovaný s nepriamym odkazom v katalógu dokumentu. Príklad štruktúry PDF dokumentu je možné vidieť na obrázku 3.1.

Katalóg dokumentu Je koreňom stromu všetkých objektov v PDF. Odkaz naň sa nachádza v uvádzacom slovníku (viď sekcia 3.3.4) na konci súboru. Okrem odkazov na podstromy môže predefinovať verziu PDF alebo informovať o použití PDF rozšírenia [12].



Ukážka 3.1: Stromová štruktúra PDF dokumentu

3.3 Štruktúra súboru

Štruktúra súboru pozostáva zo štyroch častí, a to hlavičky, tela dokumentu, odkazovacej tabuľky (angl. *cross-reference table*) a uvádzacieho slovníka (angl. *trailer dictionary*). V ukážke je možné vidieť ukážku jednoduchého PDF súboru (viď príklad 3.1).

3.3.1 Hlavička

Skladá sa z jedného, prípadne dvoch riadkov. Prvý identifikuje súbor ako PDF a určuje jeho verziu:

```
%PDF-1.0
```

Druhý riadok obsahuje minimálne štyri znaky s hodnotou väčšou ako 128. Používa sa v prí-

pade ak dokument obsahuje binárne dáta. Oba riadky začínajú znakom `%`, čo značí komentár [21].

3.3.2 Telo dokumentu

Telo dokumentu pozostáva z priamych a nepriamych objektov. Pre zobrazenie dokumentu s jednou stránkou sú potrebné najmenej štyri nepriame objekty, a to `\Catalog`, `\Pages`, `\Page` a `stream`.

Priame objekty - typy Nie je možné sa na ne odkazovať

- `« »` - slovník (angl. *dictionary*) mapujúci hodnotu na kľúč
- `[ ]` - pole (angl. *array*)
- `/` - prefix lomítko označuje menný objekt (angl. *name*)
- ( ) - reťazec
- celé alebo reálne číslo
- pravdivostná hodnota (angl. *boolean*)

Nepriame objekty Je možné odkazovať sa na ne pomocou referencie. Referencia je tvaru `X Y R`, kde `X` zastupuje ID objektu, `Y` zastupuje jeho generáciu (viď sekcia 3.3.3) a `R` je znak označujúci, že ide o referenciu.

Všetky štyri nepriame objekty uvedené v príklade (viď príklad 3.1) musia byť nepriame objekty. Pre objekt `stream` je dôvodom, že je to uvedené v špecifikácii. Pre ostatné je dôvod, že sa používajú ako hodnoty v špeciálnych slovníkoch, ktoré obsahujú typový kľúč. Slovníky s definovaným typom majú pravidlá o tom, aké ďalšie páry kľúč-hodnota môžu obsahovať. Jedným z pravidiel je, že niektoré hodnoty v slovníku musia byť nepriamymi objektmi. Každý z objektov troch nepriamych slovníkov v základnom príklade sa v určitom okamihu používa ako hodnota podliehajúca tomuto pravidlu.

3.3.3 Odkazovacia tabuľka

Umožňuje rýchle vyhľadávanie nepriamych objektov.

```
xref
0 2
0000000000 65535 f
0000000018 00000 n
2 3
0000000077 00000 n
0000000178 00000 n
0000000457 00000 n
```

Ukážka 3.2: Príklad odkazovacej tabuľky

V ukážke 3.2 je možné vidieť príklad odkazovacej tabuľky. Prvý riadok obsahuje vždy kľúčové slovo `xref`. Po ňom nasleduje identifikátor (ďalej ako ID) nasledujúceho nepriameho

objektu a počet objektov, ktoré sú uvedené v tabuľke a ich ID je práve o jedna väčšie ako ID predchádzajúceho objektu. `0` teda znamená počiatočné ID a `2` hovorí, že budú nasledovať dva objekty. Prvý má ID `0` a druhý `1`. Tretí riadok je tvaru `nnnnnnnnnn ggggg t eol` a má viacero interpretácii v závislosti od posledného tlačiteľného znaku `t`:

- `nnnnnnnnnn` v prípade `f` - angl. *free* ukazuje na záznam nasledujúceho skrytého objektu v tabuľke (index riadku)
- `nnnnnnnnnn` v prípade `n` - angl. *in use* určuje pozíciu objektu v bajtoch od začiatku dokumentu
- `ggggg` indikuje generáciu objektu - pri každom odstránení a znovu použití objektu sa táto hodnota zvýši o jedna
- značenie konca riadku `eol` sa skladá vždy z dvoch znakov - `<space><linefeed>` v prípade Unix systémov a `<carriage return><linefeed>` v prípade systému Windows

Nultý objekt je koreňom tela a nezobrazuje sa. Tento objekt má vždy hodnotu `ggggg` nastavenú na maximálnu hodnotu 65 535.

V úplne novom PDF súbore, pre iné objekty ako nultý, bude číslo generácie 00000 a typ bude `n`. Použitie zabudovaného mechanizmu PDF na nedeštruktívne prepísanie informácií zvýši počet generácií zastaraných objektov a zmení ich typ na `f` [11].

3.3.4 Uvádzací slovník

Uvádzací slovník je východiskový bod pri zobrazovaní dokumentu. V ukážke 3.3 je možné vidieť príklad takéhoto slovníka. Slovník začína kľúčovým slovom `trailer` a musí obsahovať minimálne `\Root` a `\Size`. `\Root` vždy obsahuje referenciu (viď sekcia 3.3.2) na objekt typu `\Catalog`. `\Size` musí byť priamy objekt a obsahuje počet nepriamych objektov nachádzajúcich sa v dokumente. Po kľúčovom slove `startxref` nasleduje číslo určujúce počet bajtov od začiatku dokumentu po poslednú odkazovaciu tabuľku.

```
trailer
<< /Root 1 0 R
   /Size 5
>>
startxref
565
%%EOF
```

Ukážka 3.3: Ukážka uvádzacieho slovníka

3.4 Možnosti zneužitia

Sekcia vychádza z článku Briana Lainga [9].

PDF súbor nie je len textový dokument s možnosťou priloženia grafických prvkov, obsahuje napríklad aj rôzne metódy pre ovládanie počítača, ktoré je možné zneužiť. Toto sú niektoré z nich:

- **JavaScript** - dokáže modifikovať obsah PDF a manipulovať s funkciami programu pre zobrazenie PDF. Môže tak využívať prípadných slabostí PDF zobrazovačov.

- **systémové príkazy** - PDF súbor dokáže spúšťať systémové príkazy. Niektoré z PDF zobrazovačov preto obsahujú čierne listiny a určujú ktoré príkazy môžu byť volané.
- **vstavané súbory** (angl. *embedded files*) - do súboru PDF je možné vložiť akýkoľvek súbor. Infikovaný súbor môže byť takto ukrytý niekoľkými úrovňami potenciálne bezpečných PDF súborov.

Konkrétnym príkladom je škodlivý PDF súbor objavený v lete 2018, bližšie popísaný v blogoch Antona Cherepanova a Matta Oha [3, 18]. Škodlivý dokument obsahuje JavaScript kód, ktorý je spustený ihneď po otvorení dokumentu. Ten využije slabinu programu Adobe Reader, čo vyústi až ku spusteniu vstavaného súboru PE.

```
%PDF-1.1                                  PDF hlavička - verzia 1.1
%ëąç                                      príznak že súbor sa má čítať binárne.
1 0 obj                                   objekt 1, generácia 0
  « /Type /Catalog                        katalóg dokumentu
    /Pages 2 0 R »                        koreň stromu stránok: objekt 2,
                                                generácia 0
endobj
2 0 obj                                   objekt 2, generácia 0
  « /Type /Pages                          strom stránok
    /Kids [3 0 R]                         pole stránok
    /Count 1                              počet listov stromu stránok
    /MediaBox [0 0 300 144] »             veľkosť stránky (points), lavý dolný
                                                roh až pravý rorný roh
endobj
3 0 obj                                   objekt 3, generácia 0
  « /Type /Page                           konkrétna stránka
    /Parent 2 0 R                         rodičovský uzol
    /Resources                            zdroje stránky
      « /Font                             použitý font
        « /F1                             a pomenovanie fontu
          « /Type /Font                   slovník objektu Font
            /Subtype /Type1               typ fontu
            /BaseFont /Times-Roman        štýl fontu
      » » »
    /Contents 4 0 R »                     odkaz na obsah stránky
endobj
4 0 obj                                   objekt 4, generácia 0
  « /Length 55 »                          objekt stream dĺžky 55B
  stream
    BT                                    začiatok textu
      /F1 18 Tf                           nastavenie fontu a veľkosti písma (point)
      0 0 Td                              počiatočná pozícia 0,0
      (Hello World) Tj                    text "Hello World"
    ET                                    koniec textu
  endstream
endobj
xref                                      tabuľka odkazov
0 5                                       skupina 5 objektov, prvý má ID 0
0000000000 65535 f
0000000018 00000 n                        objekt s ID 1, odsadenie 18B
0000000077 00000 n                        objekt s ID 2, odsadenie 77B
0000000178 00000 n                        objekt s ID 3, odsadenie 178B
0000000457 00000 n                        objekt s ID 4, odsadenie 457B
trailer                                   uvádzací slovník
« /Root 1 0 R                             odkaz na katalóg dokumentu
  /Size 5 »                               dokument obsahuje 5 nepriamych objektov
startxref                                 odsadenie najnovšej tabuľky odkazov od
565                                             začiatku súboru v bajtoch
%%EOF                                     znak konca súboru
```

Ukážka 3.1: Príklad PDF súboru

Kapitola 4

LNK formát

LNK (skratka z angl. *link*) je prípona pre binárny súbor primárne slúžiaci ako odkaz (skratka) na spustiteľný súbor alebo aplikáciu v prostredí OS Microsoft Windows, Najčastejšie sa používajú na vytvorene odkazov v Štart menu a na pracovnej ploche [4]. Presnejšie je LNK dátový objekt, ktorý obsahuje informácie používané na prístup k inému objektu v mennom priestore *Windows Shell* (tj. akýkoľvek objekt viditeľný prostredníctvom Prieskumníka Windows). Medzi typy objektov, ku ktorým sa dá pristupovať, patria aj priečinky, diskové jednotky a tlačiarne. Odkaz umožňuje používateľovi alebo aplikácii pristupovať k objektu odkiaľkoľvek v rámci menného priestoru. Užívateľ alebo aplikácia nemusí poznať aktuálny názov a umiestnenie objektu [16].

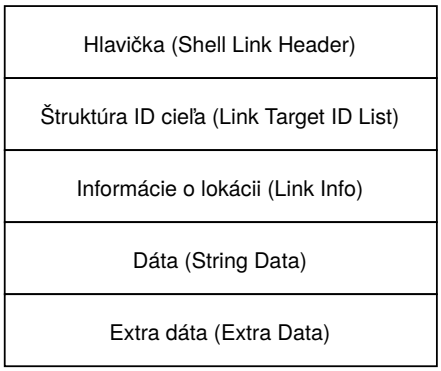
LNK súbor obsahuje množstvo užitočných informácii o počítači, na ktorom bol vytvorený, aj na ktorom sa práve nachádza. Príkladom takýchto informácii sú fyzická adresa niektorého sieťového zariadenia, sériové číslo média alebo názov zväzu disku [7].

Viac-bajtové hodnoty sú uložené v *little-endian* formáte [16].

4.1 Štruktúra

Sekcia vychádza z dokumentácie spoločnosti Microsoft [15], dokumentu Joachima Metza [14] a dokumentu Jesse Hagera [6].

Formát LNK pozostáva z postupnosti viacerých štruktúr, ako je možné vidieť na obrázku 4.1. Hlavička je povinná, zvyšné štruktúry sú voliteľné. Jednotlivé štruktúry sú popísané nižšie.



Ukážka 4.1: Štruktúra LNK súboru

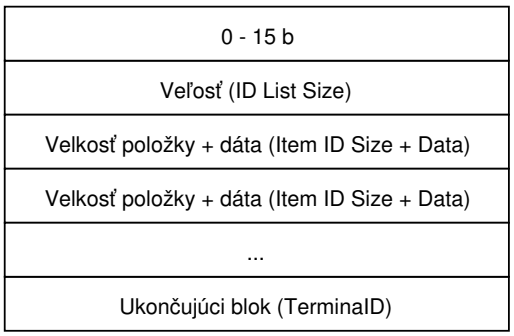
4.1.1 Hlavička

Obsahuje informácie pre identifikáciu typu súboru, časové značky práce s cieľovým súborom (čas vytvorenia, posledného prístupu, ...), príznaky označujúce prítomnosť voliteľných štruktúr, príznaky špecifikujúce informácie o cieli (napr. archív, zložka, iba na čitanie, ...), veľkosť cieľa, informáciu o spôsobe zobrazenia okna (minimalizované, maximalizované, ...), index ikony vrámci jej lokácie a klávesovú skratku.

Súbor je možné jednoznačne identifikovať ako LNK pomocou dvoch úvodných atribútov. Prvým z nich je veľkosť hlavičky, ktorá má vždy hodnotu 76 (0x0000004C), čo je ASCII hodnota pre písmeno L. Druhým je identifikátor triedy[1], ktorý musí mať hodnotu 00021401-0000-0000-C000-000000000046.

4.1.2 Štruktúra ID cieľa

Obsahuje položky špecifikujúce cieľ odkazu. Je vo forme listu štruktúr ukončenom dvoma nulovými bajtami. Štruktúru je možné vidieť na obrázku 4.2.



Ukážka 4.2: Štruktúra ID cieľa

4.1.3 Informácie o lokácii

Štruktúra obsahuje informácie potrebné na lokalizovanie cieľa v prípade, že sa cieľ nenachádza na jeho pôvodnom umiestnení. Na základe povinných úvodných atribútov tejto štruktúry je možné určiť voliteľné atribúty, ktoré môžu byť dvoje.
Atribúty lokálneho média:

- typ (CD-ROM, RAM, vzdialený, ...)
- sériové číslo
- názov zväzku

Atribúty sieťového média:

- typ sieťového poskytovateľa
- názov zariadenia (napr. `D:`)
- sieťová cesta (napr. `\\server\share`)

[1] GUID identifikujúce SW komponentu [15].

4.1.4 Dáta

Prezencia jednotlivých štruktúr nachádzajúcich sa v tejto štruktúre je určená príznakmi v hlavičke súboru. Všetky štruktúry sú voliteľné, záleží na ich poradí a sú vo formáte *dĺžka - reťazec*.

- popis - definuje text, ktorý sa zobrazí po nájdení myšou na odkaz
- relatívna cesta - definuje cestu k cieľu relatívne od svojej lokácie
- pracovný adresár - cesta ku systémovému adresáru, ktorá bude použitá pri aktivovaní odkazu
- konzolové argumenty - argumenty, ktoré budú predané cieľovej aplikácii pri aktivovaní odkazu
- lokácia ikony - definuje lokáciu ikony

4.1.5 Extra dáta

Štruktúra obsahujúca nula a viac rozličných dátových blokov, ukončená ukončujúcim blokom. Všetky neukončujúce bloky majú spoločné dva úvodné atribúty a to veľkosť a identifikátor typu bloku. Toto umožňuje identifikovať o aký typ bloku sa jedná a prípadne ho pri spracovaní preskočiť, bez nutnosti jeho spracovania.

Blok nastavení displeja špecifikuje nastavenia displeja, ktoré sa použijú v prípade, že cieľom je aplikácie, ktorá bude spustená v konzolovom okne. Špecifikuje napríklad súradnice a veľkosť okna, veľkosť a rodinu použitého fontu, vzhľad kurzoru a iné.

Blok identifikátora kódu jazyka (LCID[2]) určuje množinu znakov, ktorá sa použije pre zobrazenie textu v prípade, že cieľom je aplikácia, ktorá bude spustená v konzolovom okne.

`Darwin` (Mac OS-X) blok určuje identifikátor aplikácie, ktorý možno použiť na inštaláciu aplikácie, keď je aktivovaný odkaz.

Blok prostredia určuje cestu k premenným prostredia, ktoré sa použijú v prípade, že cesta k cieľu obsahuje premenné prostredia.

Blok lokácie ikony definuje cestu k ikone. Cesta je definovaná s použitím premenných prostredia, čo umožňuje vyhľadávanie ikon naprieč rôznymi počítačmi.

Blok známej zložky určuje cestu ku známej zložke.

Blok extra dát určuje množinu vlastností, ktoré môžu aplikácie používať na ukladať extra dát do odkazu.

[2] Viď *Windows Language Code Identifier*
https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-lcid/70feba9f-294e-491e-b6eb-56532684c37f

`Shim` blok určuje názov metódy, ktorú možno použiť pri aktivácii odkazu. Metóda pomáha aplikácii alebo funkcii fungovať podľa očakávania, aj keď ju funkcionalita operačného systému nemusí úplne podporovať alebo sa zmenila (novšia verzia OS) [5].

Blok špeciálny zložky určuje cestu ku špeciálnej zložke.

Blok sledovania dát špecifikuje dáta, ktoré môžu byť použité pri vyhľadávaní cieľa, ak sa ten nenachádza na pôvodnom mieste. Medzi dáta patrí napríklad NetBIOS meno.

Blok štruktúry ID cieľa používa sa v prípade systémov Vista a vyššie na miesto pôvodnej štruktúry (viď sekcia 4.1.2).

4.2 Možnosti zneužitia

Napriek tomu, že LNK je iba odkaz o veľkosti okolo 1 kB, jeho potenciál z pohľadu šírenia škodlivého softvéru nie je najmenší. Pri využití jeho slabostí je možné spúšťať a šíriť škodlivý softvér bez povšimnutia užívateľa.

Umožňuje napríklad stiahnuť škodlivý softvér a následne ho spustiť a popri tom spustiť aj ďalšiu aplikáciu, ktorú užívateľ očakával. Podobný príklad je uvedený v online článku od D3xt3ra [4].

LNK ďalej umožňuje používať vlastné ikony získané, okrem iného, zo súborov s príponou CPL. CPL (angl. *Control Panel Item*) sú položky ovládacieho panela, napríklad Myš, Zvuk alebo Sieť, ktoré používa operačný systém Windows. Možno ich považovať za špeciálne dynamické knižnice (DLL). Sú uložené v priečinku `Windows\System` a pri otvorení ovládacieho panela Windows sa automaticky načítajú a spustia. Útočník je tak schopný definovať, ktorý spustiteľný modul sa má načítať, a použiť súbor LNK na vykonanie ľubovoľného kódu vo vnútri prostredia Windows.

Pred niekoľkými rokmi boli takéto LNK súbory s odkazmi na ikony uložené v súboroch CPL použité v pri útoku Stuxnet [17, 13].

Kapitola 5

Návrh

Pre pridanie podpory pre nové formáty PDF a LNK je potrebné vybrať atribúty vhodné na zhlukovanie a analyzátory, ktoré tieto atribúty vedia extrahovať. Rovnako je potrebné vybrať metódy, na základe ktorých bude zhlukovanie prebiehať. V tejto kapitole je popísaný výber vhodných atribútov, nástrojov a metód, ako aj návrh následnej integrácie do nástroja Clusty.

5.1 Výber atribútov na zhlukovanie

Pre zhlukovanie je vhodné zvoliť atribúty, ktorých hodnoty nie sú bežne zhodné pre nesúvisiace súbory. Pre lepšie určenie podobnosti súborov po zhlukovaní je možné použiť aj atribúty, ktorých hodnoty nie sú už také unikátne, ale môžu mať v rámci zhluku výpovednú hodnotu.

5.1.1 Zhlukovanie PDF súborov

Na PDF je možné sa pozerať dvoma spôsobmi, a to PDF ako výsledný dokument, ktorý vidí užívateľ po jeho spustení, a PDF ako zdrojový súbor, ktorý má na vstupe čítačka PDF dokumentov. Pri zhlukovaní by sa mali v jednom zhluku nachádzať iba podobné vzorky. Ich podobnosť je teda možné určiť z dvoch pohľadov. Vďaka linearizácii, komprimácii, šifrovaniu a neusporiadanosti objektov môže z dvoch veľmi odlišných súborov vzniknúť totožný dokument.

Atribúty, ktoré je možné získať, sa dajú rozdeliť na dáta vyššej úrovne, tj. dáta, ktoré sú použité vo výslednom dokumente (napr. text stránky), a dáta nižšej úrovne, tj. údaje, ktoré sú potrebné pri spracovávaní dokumentu (napr. referenčná tabuľka). Ako vhodné dáta vyššej úrovne boli vybrané tieto:

- text prvej stránky - je pravdepodobné, že PDF bude mať minimálne jednu stránku. Problém nastáva pri šifrovaných súboroch, prípadne pri poškodených súboroch, kde text stránky nemusí byť možné získať.
- text poslednej stránky - v prípade PDF s jedinou stránkou je prvá stránka zhodná s poslednou

Ako vhodné dáta nižšej úrovne boli vybrané tieto:

- verzia PDF - mala by sa nachádzať v každom súbore PDF, ale môžu sa stať aj prípady kedy to neplatí (viď sekcia 5.3.1). Keďže nadobúda iba určité hodnoty, sama o sebe nie je vhodná na zhlukovanie.
- metadáta (titulok, autor, atď.) - napriek tomu že nie sú povinné, nachádzajú sa vo väčšine súborov (nie nutne všetky). Medzi najzaujímavejšie patria:

  Autor - meno autora dokumentu (napr. `John Whitington`)

  Producent - názov programu, ktorý konvertoval tento súbor na PDF, ak bol pôvodne iného formátu (napr. `Acrobat Distiller 6.0 for Macintosh`)

  Tvorca - názov programu, ktorý pôvodne vytvoril tento dokument (napr. `QuarkXPress: pictwpstops filter 1.0`)

  Definície vychádzajú z knihy Johna Whitingtona [21].
- referenčná tabuľka - mala by byť súčasťou každého dokumentu. Pravdepodobnosť, že by ju mali dva rozdielne súbory rovnakú, klesá vyšším počtom objektov v súbore.
- príznak linearizácie - značí, že súbor bol lienarizovaný (viď sekcia 3.1)
- príznak šifrovania - značí, že súbor je zašifrovaný. Neznamená to nutne, že nie je možné získať text stránok. V prípade nastaveného, ale prázdneho užívateľského hesla je možné dokument otvoriť aj bez zadania hesla.
- príznak prítomnosti `\OpenAction` - značí, že pri otvorení súboru sa vykoná prednastavená akcia
- príznak prítomnosti `\AcroForm` - značí, že súbor obsahuje formulár
- URI - URI adresy obsiahnuté v dokumente
- JavaScript kód - tento kód sa vykoná pri prevedení určitej udalosti, ktorá môže byť napríklad otvorenie dokumentu, kliknutie na určité miesto, a iné.
- údaje z katalógu dokumentu - povinne by mal obsahovať odkaz na koreňový objekt a taktiež aj počet objektov, ktorý do istej miere naznačuje veľkosť súboru. Ďalej môže obsahovať aj odkaz na metadáta.
- počítadlá stránok, objektov, a iných

Okrem dát získaných statickou analýzou je možné taktiež použiť aj informáciu o zachytení jednej z výnimiek pri spracovávaní súboru:

- príznak poškodeného súboru

5.1.2 Zhlukovanie LNK súborov

Z LNK formátu je možné získať všetky informácie spomínané v sekcii o LNK formáte (viď sekcia 4.1). Keďže týchto informácii je veľké množstvo, boli vybrané iba určité z nich.

Hlavička - povinná časť formátu, ktorá okrem iného určuje aj interpretáciu bajtov nasledujúcich za hlavičkou.

- identifikátor triedy - mal by mať zakaždým rovnakú hodnotu
- časové značky - určuje čas vytvorenia, poslednej úpravy a posledného prístupu ku cieľa
- veľkosť cieľa - označuje veľkosť cieľového objektu
- index ikony - index ikony vrámci jej lokácie
- klávesová skratka - klávesová skratka pre spustenie odkazu
- voliteľné štruktúry - zoznam všetkých voliteľných štruktúr, ktoré by sa mali v súbore nachádzať, na základe príznakov v hlavičke
- atribúty súboru - zoznam všetkých nastavených atribútov (napr. skrytý alebo archív)

Dáta - prítomnosť týchto dát je veľmi častá

- popis - obsahuje text, ktorý sa zobrazí užívateľovi po nájdení na ikonu
- relatívna cesta - cesta k cieľu
- pracovný adresár - cesta k systémovému adresáru
- lokácia ikony - cesta k ikone. Ikona nemusí byť zrovna najvhodnejšia, ako atribút podľa ktorého sa bude zhlukovať, a to z dôvodu že môže slúžiť na oklamanie užívateľa tak, že sa použije jedna zo známych, ako napr. ikona IE.
- konzolové argumenty - argumenty, ktoré budú predané aplikácii, ktorá je cieľom odkazu.
- cieľová aplikácia - je získaná z posledného segmentu relatívnej cesty. Určuje cieľovú aplikáciu, prípadne zložku. Bude tak možné zjednotiť rovnaké aplikácie s rozdielnou cestou.

Informácie o lokácii - umožňujú identifikovať lokáciu cieľa

- základ lokálnej cesty - používa sa pri konštrukcií výslednej cesty k cieľu
- spoločná prípona cesty - používa sa pri konštrukcií výslednej cesty k cieľu
- lokácia - nadobúda jednu z dvoch možných hodnôt, sieť alebo lokál
- sériové číslo média - identifikuje médiu, kde sa nachádza cieľ
- typ média - typ média, kde sa nachádza cieľ
- názov zväzku (aj Unicode)
- typ sieťového poskytovateľa
- názov zariadenia (aj Unicode)
- sieťová cesta (aj Unicode)

Extra dáta

- lokácia ikony (ANSI aj Unicode)
- `Shim` dáta
- `Darwin` dáta (ANSI aj Unicode)
- cesta k premenným prostredia (ANSI aj Unicode)
- ID špeciálnej zložky
- ID známej zložky
- NetBIOS meno
- identifikátor zväzku (aktuálny aj pri vzniku)
- identifikátor súboru (aktuálny aj pri vzniku)
- MAC adresa (aktuálna aj pri vzniku)

5.2 Výber nástroja na analýzu

V tejto sekcii je popísaný výber vhodného nástroja pre analýzu oboch formátov. Podľa nástroja Clusty je najprijateľnejšia platforma hľadaného nástroja jazyk Python 3, ale v prípade kvality nástrojov postavených na inom jazyku je možné použiť, menej priamym spôsobom, aj tie.

5.2.1 PDF analyzátory

Existuje viacero programov na získavanie informácii a prácu s PDF súbormi. Keďže zhlukovanie bude vykonávané denne nad tisíckami súborov, pričom tieto súbory môžu využívať rôzne obfuskovacie a komprimačné techniky alebo využívať slabosti čítačiek a programov na prácu s nimi, je potrebné, aby vedeli získať čo najviac z vyššie spomínaných informácii (viď sekcia 5.1.1) v čo najkratšom čase a aby sa vedeli vysporiadať aj so súbormi, ktoré síce nedodržujú normu, ale sú zobraziteľné pomocou bežne dostupných a používaných PDF čítačiek.

pdfrw [1]

Modul a konzolová aplikácia v jazyku Python 2 a 3 na čítanie a úpravu PDF súborov. V porovnaní s pdfminer a PyPDF2 má veľmi rýchly parser. Taktiež je oproti nim jednoduchšia z pohľadu množstva funkcionalít a dokument reprezentuje na nižšej úrovni.

Umožňuje získať ID dokumentu, metadáta a vyhľadávať objekty. Vie sa vysporiadať s nekorektným objektom.

Nevýhody:

- očakáva správne číslo odsadenia pri `startxref`

[1] https://github.com/pmaupin/pdfrw

PDFMiner [2]

Modul a konzolová aplikácia v jazyku Python 2 a 3 na extrakciu informácii z PDF dokumentov zameraný predovšetkým na textové dáta.

Umožňuje získať obsah, metadáta, katalóg dokumentu a Python objekty stránok.

Nevýhody:

- problémy pri spracovaní súborov, ktoré nie sú úplne podľa normy
- nevie spracovať zašifrované súbory

PyPDF2 [3]

Modul a konzolová aplikácia v jazyku Python 2 a 3 schopná rozdelenia, zlúčenia, orezania a transformácie stránok súborov PDF.

Umožňuje získať text každej stránky, metadáta a to aj v XML forme, informácie z uvádzacieho slovníka a obsah.

Nevýhody:

- problémy pri spracovaní súborov, ktoré nie sú úplne podľa normy

PeePDF [4]

Konzolová aplikácia v jazyku Python 2 určená na skúmanie PDF súborov za účelom zistenia ich potencionálnej škodlivosti. Dokáže vyhľadávať podozrivé prvky dokumentu a podporuje všetky najpoužívanejšie filtre a kódovania. Taktiež umožňuje extrakciu starých verzií dokumentu.

Umožňuje vyhľadávať reťazce, metadáta a JavaScript.

Nevýhody:

- žiadne API

PyMuPDF [5]

PyMuPDF umožňuje prístup ku dôležitým funkciám MuPDF, implementovaného v jazyku C, v prostredí jazyka Python 2 a 3. MuPDF je nástroj na úpravu a konverziu PDF, XPS, EPUB a iných typov súborov.

Umožňuje extrakciu textu, obrázkov a odkazov zo stránok, získanie metadát, obsahu, vstavaných súborov, anotácii a vyhľadávanie v texte. Vie sa vysporiadať aj so súbormi, ktoré nie sú podľa normy. Vďaka implementácii v jazyku C je veľmi rýchly.

pdfparser [6]

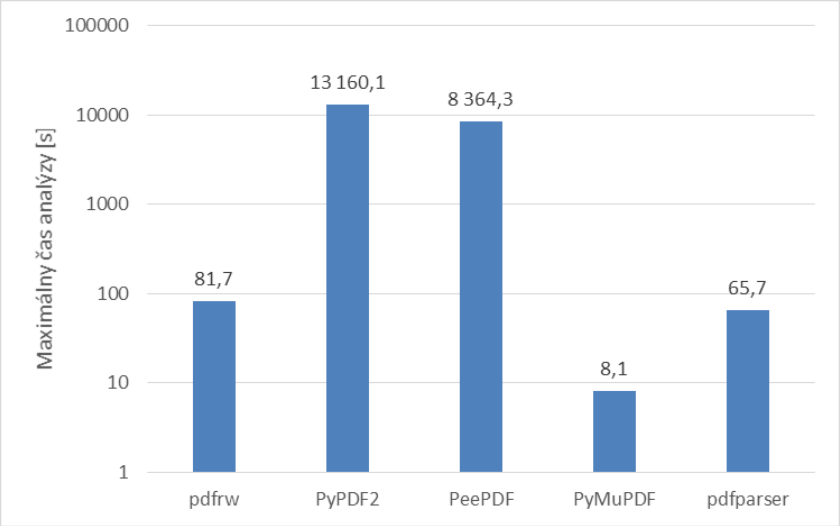
Konzolová aplikácia v jazyku Python 2 a 3. Nástroj parsuje PDF súbory a identifikuje základné elementy.

Umožňuje získať ID, metadáta a referenčnú tabuľku.

Nevýhody:

- žiadne API

[2] https://github.com/euske/pdfminer
[3] https://pypi.org/project/PyPDF2/
[4] https://github.com/jesparza/peepdf
[5] https://github.com/pymupdf/PyMuPDF
[6] https://github.com/DidierStevens/DidierStevensSuite/blob/master/pdf-parser.py



Obr. 5.1: Maximálna rýchlosť analýzy PDF súboru

Na nástrojoch boli vykonané rýchlostné testy. Tieto testy testovali priemerný a maximálny čas trvania analýzy jedného súboru na vzorke 2 380 náhodných PDF súborov, poskytnutých spoločnosťou Avast. Každý z nástrojov mal za úlohu spracovať súbor a vypísať základné informácie o ňom. Výsledky rýchlostných testov je možné vidieť v grafe 5.1 a 5.2. Niektoré súbory neboli tieto nástroje schopné úspešne analyzovať a tak schončili s chybou. Úspešnosť je možno vidieť v grafe 5.3.

Z vyššie uvedených nástrojov bola na analýzu PDF súborov vybraná dvojica `PyMuPDF` a `pdf-parser`. Oba nástroje je možné použiť v prostredí Python erzia 3, ktorú používa aj Clusty.

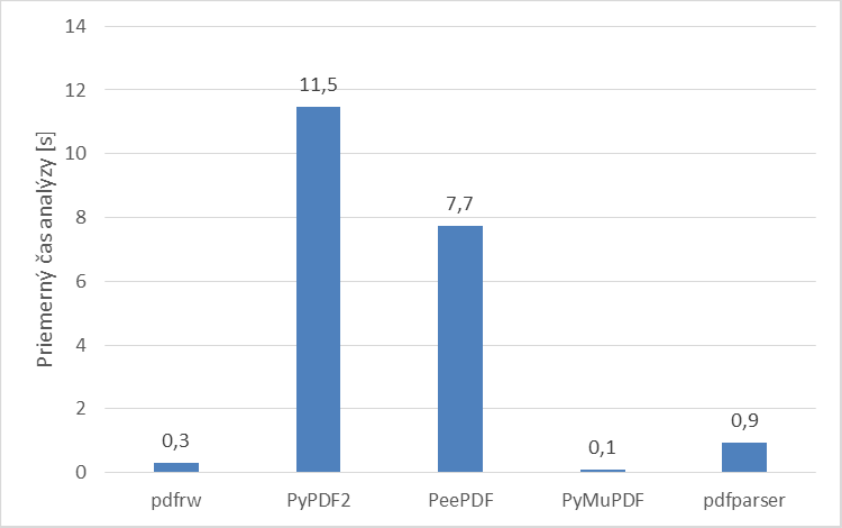
`PyMuPDF` je vďaka implementácii v jazyku C veľmi rýchly a je možné ho jednoducho nainštalovať pomocou Python rozhrania `pip`. Vie sa vysporiadať aj s poškodenými súbormi, a tak vrátiť kvalitné informácie vyššej, čiastočne aj nižšej, úrovne vo veľkej miere prípadov.

`pdf-parser` patrí taktiež k rýchlym nástrojom. Jeho účelom je doplnenie získaných dát z vyššie spomínaného nástroja o dáta nižšej úrovne. Jeho problémom je, že nie je vytvorený ako použiteľná Python modul, a tak bolo potrebné vytvoriť vhodné rozhranie. Popri tom boli zapracované aj rozšírenia, ako napríklad získavanie referenčnej tabuľky a následne pomocou nej aj získavanie kódu v jazyku JavaScript.

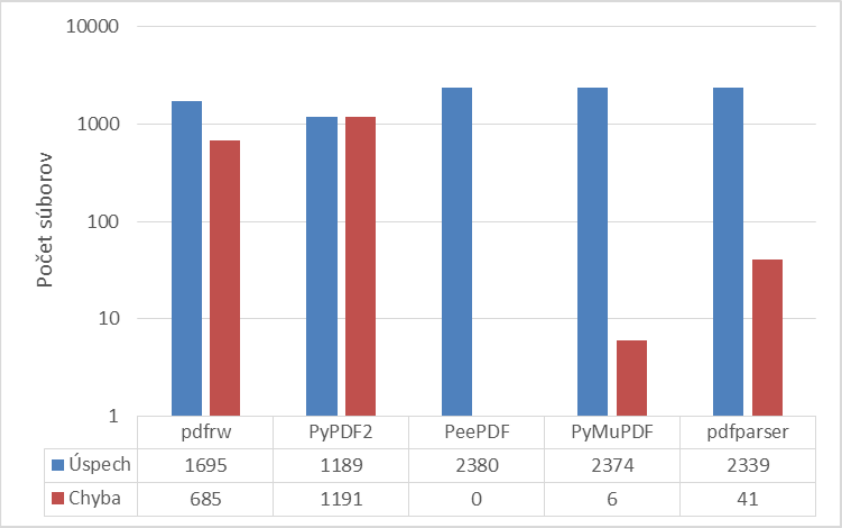
5.2.2 LNK analyzátory

V porovnaní s multiplatformovým formátom PDF, LNK je formát používaný výlučne v prostredí MS Windows. Jedná sa o súbor s binárnym obsahom, ktorého štruktúra je presne definovaná, a malé poškodenie môže viesť k nesprávnej interpretácii všetkých nasledujúcich dát. Veľkosť súborov je typicky 1-2 kB a preto je ich analýza veľmi rýchla.

Pre základnú prácu s LNK súbormi postačuje natívne grafické rozhranie poskytované bežnému užívateľovi MS Windows. Pre pokročilú prácu s nimi existuje viacero nástrojov, väčšina z nich je však viazaná na prostredie MS Windows alebo používa grafické prostredie. V prostredí Linux, v ktorom pracuje nástroj Clusty, je z nich možné použiť nasledovné:



Obr. 5.2: Priemerná rýchlosť analýzy PDF súboru



Obr. 5.3: Úspešnosť analýzy PDF súborov

pylnk [7]

Modul a konzolová aplikácia v jazyku Python 2 na čítanie, úpravu a vytváranie LNK súborov. Je možné ju nainštalovať s pomocou inštalátor Python balíčkov - *pip*.

Umožňuje prístup ku všetkým štruktúram LNK okrem exta dát. Dáta sprístupňuje formou objektu.

liblnk [8]

Robustná knižnica v jazyku C s rozhraním v jazyku Python 2 a v3 na extrakciu informácii z LNK súborov.

Umožňuje extrakciu takmer všetkých dátových štruktúr sprístupnených formou objektu.

pylnker [9]

Modul a konzolová aplikácia v jazyku Python 2, určená na čítanie LNK súborov. Je vytvorená na základe nástroja lnk-parse v jazyku Perl. Dáta sprístupňuje pomocou objektu.

Rovnako ako pylnk umožňuje prístup ku všetkým štruktúram LNK okrem exta dát.

LnkParse [10]

Modul a konzolová aplikácia v jazyku Python 2 na extrakciu dát z LNK súborov. Je možné ju nainštalovať s pomocou inštalátor Python balíčkov - *pip*. Dáta sprístupňuje pomocou objektu, ale aj vo formáte JSON.

Je možné získať všetky dátové štruktúry okrem extra dát a štruktúry ID cieľa.

lnk-parse [11]

Konzolová aplikácia v jazyku Perl pre extrahovanie informácii z LNK súborov.

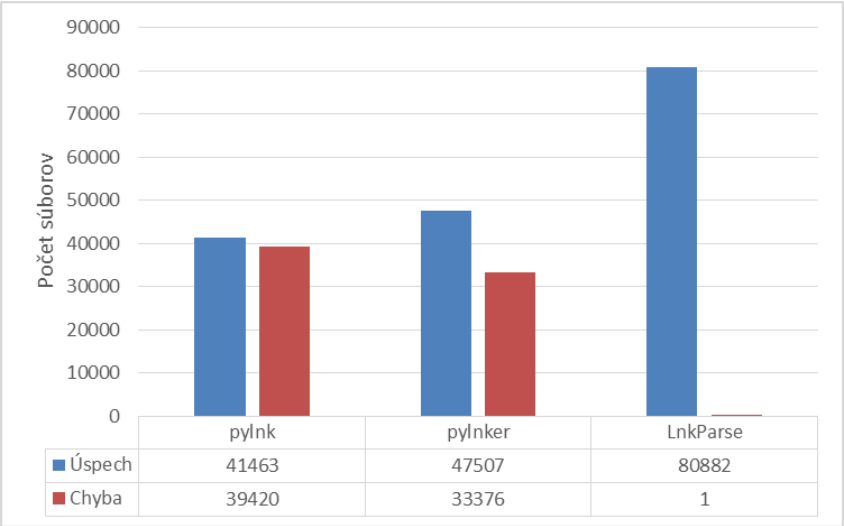
Umožňuje získať všetky informácie o súbore okrem extra dát a štruktúry ID cieľa.

Spomínané nástroje boli otestované na 80 000 náhodných vzorkách poskytnutých spoločnosťou Avast. Všetky nástroje s výnimkou `LnkParse` mali problémy pri spracovaní týchto súborov, a vyhadzovali výnimky. Úspešnosť je možmé vidieť v grafe 5.4. Druhý najúspešnejší analyzátor bol `pylnker`, ktorý dokázal úspešne spracovať dve tretiny zo súborov. Tieto dva nástroje si boli rýchlostne podobné, obom trvalo spracovanie týchto súborov okolo 20 sekúnd. Keďže rýchlosť spracovania veľkého množstva súborov bolo podobná, pre jeden súbor zanedbateľná, druhá dôležitá podmienka bola práve schopnosť spracovať každý súbor.

I keď sa podľa popisu môže zdať, že `liblnk` je vďaka implementácii v jazyku C, rozhraním pre Python 3 a množstvom získaných dát najvhodnejší, vyskytli problémy pri jeho sprevádzkovaní. Veľká rýchlosť spracovania súborov a jednoduchosť ostatných nástrojov zapríčinili, že tento nástroj nebol ďalej dopodrobna preskúmaný, opravený a následne použitý.

`LnkParse` je jednoduchý analyzátor písaný tak, že je odolný voči prípadným chybám, a vie tak získať čo najviac dát aj z niektorých neštandardných súborov. Vďaka jeho jednoduchosti nebol problém do-implementovať chýbajúce štruktúry, dokonca i migrovať na Python 3 používanom v nástroji Clusty.

[7] https://pypi.org/project/pylnk/
[8] https://github.com/libyal/liblnk
[9] https://github.com/HarmJ0y/pylnker
[10] https://github.com/silascutler/LnkParse
[11] https://github.com/lcorbasson/lnk-parse

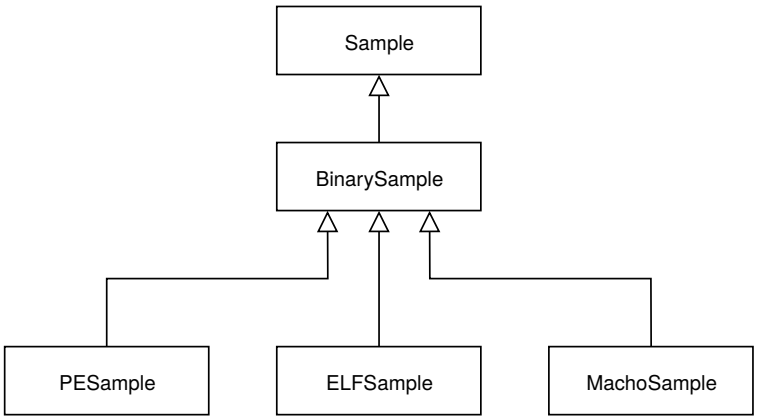


Obr. 5.4: Úspešnosť analýzy LNK súborov

5.3 Rozšírenie nástroja Clusty

Ako bolo už spomínané na strane 6, zhluková analýza prebieha v dvoch fázach. Prvá fáza je implementovaná v jazyku Python, druhá fáza je v jazyku C++. Pre pridanie podpory pre nový súborový formát je potrebné rozšíriť obe tieto časti. V Python časti prebieha identifikácia a analýza vzoriek, a získané informácie sa vhodne reprezentujú a uložia do databázy. V C++ časti prebieha samotné zhlukovanie podľa určených charakteristík.

Clusty využíva na analýzu väčšiny formátov externé nástroje. Získané údaje sú následne reprezentované buď vo forme objektu triedy `Sample`, alebo častejšie jednou z jej podtried. Trieda `SampleAnalyzer` sa stará o výber správnej podtriedy a uloženie dát do databázy. K dispozícii má jednotlivé analyzátory. Jedným z nich je aj trieda `FileAnalyzer`, ktorá sa zaoberá identifikáciou typu a podtypu analyzovanej vzorky, napr. typ - *binárny súbor* a podtyp - *PE*. Na základe určeného typu a podtypu sa na analýzu vzorky použije príslušný analyzátor a analyzované dáta sú reprezentované triedov `Sample`, pri tomto prípade to bude konkrétne `PESample`.



Obr. 5.5: Diagram hierarchie tried reprezentujúcich binárne vzorky (C++ a Python)

Diagram 5.5 zobrazuje hierarchiu tried reprezentujúcich binárne vzorky. Všetky formáty spadajúce do kategórie binárnych súborov majú spoločnú nadtriedu `BinarySample`, kde sa ukladajú informácie spoločné pre všetky typy binárnych súborov. V nadradenej triede `Sample` sú uložené informácie nezávislé na formáte, ako napríklad veľkosť súboru. Informácie sa ukladajú tak ako sú získané z analyzátorov, prípadne z menšími úpravami ako napr. odfiltrovanie nepotrebných informácii alebo zahashovanie príliš dlhých reťazcov.

Po analýze vzoriek a ich uložení do databázy, teda pri samotnom zhlukovaní, sa vzorky vezmú z databázy a reprezentujú sa triedami podobnými triedam v Python časti. Ich hierarchia je rovnaká. Keďže pri zhlukovaní prebieha porovnávanie množstva atribútov, nie len tých podľa ktorých sa zhlukuje, reťazce sú často reprezentované hashmi. Výnimkou sú prípady, keď je vhodné tieto reťazce zobrazovať vo výsledných zhlukoch.

5.3.1 PDF formát

Takáto štruktúra umožňuje jednoduché rozšírenie o ďalšie formáty súborov. Najprv je potrebné detekovať PDF súbory. Typ a podtyp sa určuje v triede `FileAnalyzer`. Podľa normy by každý súbor PDF mal bezprostredne začínať hlavičkou, ktorú by bolo možné popísať regulárnym výrazom `%PDF-[0-9]\\.[0-9](\n|\r| )`. Niektoré PDF čítačky však vedia čítať aj súbory, ktoré nezačínajú bezprostredne hlavičkou a snažia sa ju hľadať do určitej hĺbky, prípadne vedia spracovať aj súbory z nekompletnou hlavičkou.

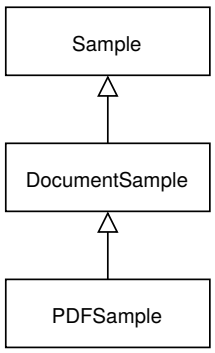
Napríklad *Adobe Acroat Reader DC (2019.010.20098)* vyhľadáva reťazec popísaný regulárnym výrazom `%PDF-[0-9]\\.[0-9]+` v úvodných 1026 bajtoch a *Chrome (72.0.3626.119)* vyhľadáva reťazec `%PDF` v úvodných 1028 znakoch.

Keďže `FileAnalyzer` už má k dispozícii aj informácie poskytnuté Linuxovým nástrojom `file`, použijú sa pri identifikácii prioritne tie a až v prípade neidentifikovania súboru ako PDF sa súbor identifikuje pomocou regulárneho výrazu `%PDF-[0-9]\\.[0-9](\n|\r| )`. Ak je hlavička úspešne nájdená, naznačuje to, že sa jedná o PDF súbor. Výsledný typ a podtyp pre PDF súbor bude `dokument - pdf`.

Pre analýzu PDF je potrebný vlastný analyzátor. Jeho výber je popísaný na strane 23. Konkrétne bola vybraná kombinácia dvoch nástrojov. Dáta z oboch nástrojov bude potrebné spojiť do jedných výsledných dát, pričom dáta z `PyMuPDf` majú prednosť.

Pre uloženie získaných dát z analýzy je potrebné pridať ďalšiu podtriedu `PDFSample` triedy `Sample`. Presnejšie podtriedu jej podtriedy `DocumentSample` ako je možné vidieť v diagrame 5.6.

Rozšírenie o PDF formát v C++ časti je rovnaké tomu v Python časti. Informácie budú reprezentované pomocou triedy `PDFSample`.



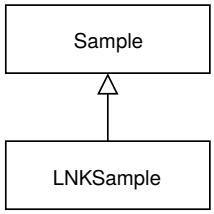
Obr. 5.6: Diagram hierarchie tried reprezentujúcich PDF vzorky (C++ a Python)

5.3.2 LNK formát

Detekcia formátu LNK je jednoduchšia ako PDF. LNK súbor je možné identifikovať pomocou dvoch úvodných atribútov. Prvým je veľkosť hlavičky, ktorá má vždy hodnotu 76 (0x0000004C) a druhým je identifikátor triedy, ktorý musí mať hodnotu 00021401-0000-0000-C000-000000000046. Pre identifikáciu však postačí aj identifikácia na základe prvého atribútu. LNK narozdiel od PDF bude dediť priamo od triedy `Sample`. Typ a podtyp určený triedou `FileAnalyzer` preto bude `lnk - None`.

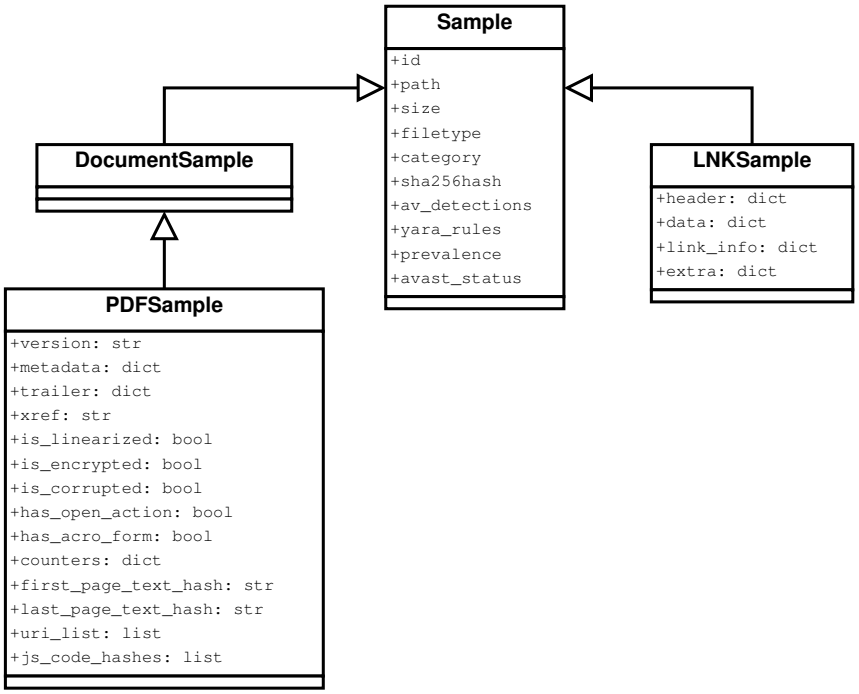
Pre analýzu LNK je taktiež potrebný vlastný analyzátor. Jeho výber je popísaný na strane 25.

Získané dáta budú v Python aj v C++ časti reprezentované novou podtriedov `LNKSample` triedy `Sample`, ako je možné vidieť v diagrame 5.7.



Obr. 5.7: Diagram hierarchie tried reprezentujúcich LNK vzorky (C++ a Python)

Zjednodušený diagram hierarchie tried reprezentujúcich LNK i PDF spolu s ich atribútmi je možné vidieť na obrázku 5.8. Úplná verzia je dostupná v prílohe A.1.



Obr. 5.8: Zjednodušený diagram hierarchie tried reprezentujúcich PDF a LNK vzorky

5.4 Výber metód na zhlukovanie

Výber metód na zhlukovanie prebiehal pri oboch formátoch, PDF i LNK, experimentálnym spôsobom. Experimenty mohli byť prevádzané až po implementácii základných štruktúr a tried popísaných v kapitole 6. Na počiatku bolo spustené zhlukovanie na základe ssdeep hashu. Tato metóda už bola v minulosti implementovaná a zhlukuje iba súbory, ktoré sú si naozaj veľmi podobné.

Ako prioritná metóda na zhlukovanie všetkých existujúcich formátov sa používa metóda podľa YARA pravidiel. Ako posledná metóda sa používa metóda podobnosti ssdeep hashov.

5.4.1 PDF

Pri PDF bolo zhlukovaných 100 000 vzoriek a zhlukovanie ukázalo, že žiaden z atribútov sám o sebe nie je vhodný. Zhluky mali spoločnú len veľmi malú časť zo získaných atribútov. Najčastejšie to bola verzia (`version`), informácie z uvádzacieho slovníka (`trailer_root`, `trailer_info` a `trailer_size`) a producent (`metadat_producer`). Rovnaká hodnota každého z týchto atribútov sa nachádzala aj v zhlukoch, ktoré boli veľmi odlišné. Pri týchto atribútoch to bolo pochopiteľné, pretože napríklad `trailer_size`, čo je počet PDF objektov, môže nadobúdať ľubovolné kladné čísla, a tie sa môžu zhodovať i pri diametrálne odlišných vzorkách. Bol teda vytvorený nový atribút (`info`), ktorý sa skladá z jednej z možných kombinácii týchto atribútov:

- `trailer_root`, `trailer_size`, `trailer_info`, `metadata_producer`
- `trailer_root`, `trailer_size`, `trailer_info`
- `trailer_root`, `trailer_size`, `version`

Kombinácie boli vytvorené na základe pozorovania zhlukov vytvorených pomocou ssdeep hashu. Na prvý pohľad zhluky nevyzerali kvalitne. Do jedného zhluku spadali aj vzorky s väčším rozdielom veľkosti, dokonca ani miera dôveryhodnosti nebola príliš vysoká. No ukázalo sa, že napriek tomu sú si všetky tieto vzorky podobné z pohľadu súboru. Atribúty obsiahnuté v tejto kombinácii sú navyše viac-menej povinnou súčasťou každého PDF, takže je možné takto zhlukovať takmer všetky PDF vzorky.

Ako záložné atribúty boli vybrané hash prvej stranky `first_page_text_hash` a hash poslednej stranky `last_page_text_hash`. Tie sa pozerajú na podobnosť výsledného dokumentu (nie súboru).

Metódy a ich poradie je možné vidieť v tabuľke 5.1.

Metóda	Atribút	Podmienky
`pdf_info_100_match`	`trailer_info`	žiadne
`pdf_first_page_text_hash`	`first_page_text_hash`	žiadne
`pdf_last_page_text_hash`	`last_page_text_hash`	žiadne

Tabuľka 5.1: Vybrané metódy zhlukovania PDF vzoriek zoradené zostupne podľa priority

5.4.2 LNK

Pri LNK boli zhluky vzniknuté zhlukovaním 80 000 vzoriek pomocou ssdeep veľmi malé a mali spoločné veľké množstvo spoločných atribútov, v podstate všetky. Výber atribútov tak bol náročnejší ako pri formáte PDF.

Na počiatku boli vylúčené atribúty, ktoré mali spoločné hodnoty naprieč odlišnými zhlukmi, pripadne tie, ktoré nadobúdali iba určité hodnoty. Takto bolo vybraných sedem atribútov, podla ktorých boli vzorky postupne zhlukované. Vysedené zhluky boli preskúmané a tu sú výsledky:

aplikácia (`data_application`)
: Môže vytvárať nekvalitné zhluky. Dôvodom je, že napr aplikácia Chrome je bez dodatočných argumentov neškodná, ale s argumentami prichádza potenciálna hrozba (napr. otvorenie škodlivej stránky bez sandboxu). Riešením by bolo odfiltrovanie aplikácii z argumentami.

lokácia ikony (`data_icon_location`)
: Zhluky vyzerajú pomerne kvalitne. Občasne sa však môže vyskytnúť aj vzorka klasifikovaná ako čistá v zhluku klasifikovanom ako škodlivý, pripadne i naopak. Ďalším problémom by bola potreba odfiltrovať generickú hodnotu `shell32.dll`, a to case-insensitive. Výhodou je, že umožňuje zhlukovať aj súbory, ktoré majú rozdielne konzolové argumenty z dôvodu generovania určitej časti.

NetBIOS meno (`extra_machine_identifier`)
: Zhluky nie sú kvalitné. V jednom zhluku sa nachádzajú veľmi rozličné vzorky.

konzolové argumenty (`data_command_line_arguments`)
: Vytvára pomerne veľké a kvalitné zhluky. V zhluku sa vždy nachádzajú vzorky s rovnakou funkcionalitou.

ID známej zložky (`extra_known_folder_id`)
: Vytvára nekvalitné zhluky. V jednom zhluku sa nachádzajú veľmi rozličné vzorky.

identifikátor zväzku (`extra_birth_droid_volume_id`)
: Zhluky vyzerajú nekvalitne, pretože obsahujú rozličné vzorky. Taktiež by bolo potrebné odfiltrovať hodnotu samých núl.

sériové číslo média (`link_info_drive_serial_number`)
: Zhluky podľa tohoto atribútu taktiež vyzerajú nekvalitne.

Na základe vyššie uvedených výsledkov bolo vybraných päť metód. Atribútom prvej metódy sú konzolové argumenty a všetky nasledovné metódy predpokladajú neprítomnosť spomínaného atribútu. Za predpokladu zhlukovania podľa priority vytvárali metódy kvalitné a veľké zhluky.

Počas vývoja však bola pridaná nová funkcionalita nástroja Clusty, ktorá umožňuje prezhlukovať existujúci zhluk s vynechaním určitých metód. Táto funkcionalita umožňovala porušiť navrhnutý koncept. Pre vyriešenie problému boli pridané ďalšie tri atribúty zložené zo základných atribútov a príznaku prítomnosti konzolových argumentov:

- `extra_known_folder_id`, `data_application`
  a príznak prítomnosti `data_command_line_arguments`
- `link_info_drive_serial_number`
  a príznak prítomnosti `data_command_line_arguments`
- `extra_birth_droid_volume_id`
  a príznak prítomnosti `data_command_line_arguments`

Tieto metódy sa použijú iba v prípade prítomnosti konzolových argumentov. Pri zvolenom poradí takáto situácia môže nastať len v prípade prezhlukovania. Zvyšné metódy, založené na atribútoch získaných z LNK vzorky, boli naopak upravené tak, aby boli aplikovateľné iba pri absencii konzolových argumentov. Výsledné metódy a ich poradie:

`lnk_data_command_line_arguments`
: Metóda je založená na jedinom základnom atribúte `data_command_line_arguments`. Zachytáva väčšinu škodlivých vzoriek. I keď nezachytáva všetky škodlivé vzorky, chovanie spúšťaného cieľa zaleží už len od cieľa samotného. Umožňuje tak používať napr. metódu založenú na názve cieľovej aplikácie.

`lnk_drive_and_cli_100_match`
: Je založená na atribúte `drive_and_cli`, ktorý je zložený zo základného atribútu `lnk_link_info_drive_serial_number` a reťazca označujúceho prítomnosť konzolových argumentov. Príklad hodnoty atribútu: `"CLI arguments: True, Drive serial number: 0xd42641ca"`.

`lnk_volume_app_and_cli_100_match`
: Metóda je založená na atribúte `volume_app_and_cli` zloženom z dvoch základných atribútov `extra_birth_droid_volume_id` a `data_application` a rovnako ako pri predchádzajúcej metóde, na prítomnosti konzolových argumentov.

`lnk_folder_app_and_cli_100_match`
: Je založená na zloženom atribúte `folder_app_and_cli`, ktorý vzniká kombináciou dvoch základných atribútov `extra_known_folder_id` a `data_application`. Rovnako, ako pri predchádzajúcich dvoch metódach, je podmienkou vzniku zloženého atribútu prítomnosť konzolových argumentov, a tak aj metóda samotná je použitá iba v takom prípade.

`lnk_data_application`
: Prvá metóda, ktorá je narozdiel od všetkých predchádzajúcich metód použitá v prípade absencie konzolových argumentov. Konkrétne je založená na atribúte `data_application`.

`lnk_data_icon_location`
: Metóda je založená na atribúte `data_icon_location` a absencii konzolových argumentov.

`lnk_link_info_drive_serial_number`
: Je založená na atribúte `link_info_drive_serial_number` a absencii konzolových argumentov.

`lnk_extra_birth_droid_volume_id`
: Metóda je založená na atribúte `extra_birth_droid_volume_id` a absencii konzolových argumentov.

Kapitola 6

Implementácia

V tejto kapitole je popísaná implementácia rozšírenia nástroja Clusty navrhnutá v kapitole 5. Python časť nástroja Clusty má za úlohu analyzovať vzorky a uložiť získané informácie do databázy. Implementácia je určená pre Python erziu 3.7. Pre správu modulov tretích strán sa používa virtuálne prostredie vytvorené nástrojom virtualenv. Zhluková analýza je implementovaná v jazyku C++, konkrétne vo verzii C++17. Pre preklad sa používa nástroj CMake s prekladačom GCC a dokumentácia je generovaná nástrojom Doxygen. Na správu revízii sa používa verzovací nástroj Git.

6.1 Analýza PDF

PyMuPDF poskytuje statickú analýzu PDF súborov a sprístupňuje získané dáta, primárne vyššej úrovne, vo forme objektu dokumentu. Je možné ho nainštalovať pomocou inštalátora balíčkov `pip` pre jazyk Python. Práca s týmto nástrojom je implemetovaná v súbore `pdf_analyser.py`.

pdf-parser naopak ponúka dáta nižšej úrovne, avšak je vytvorený ako konzolová aplikácia a neponúka žiadne vhodné API. Toto API bolo potrebné dorobiť, konkrétne bola nahradená pôvodná funkcia `main()`, ktorá sa stará o parsovanie PDF súboru, za vlastnú funkciu, prispôsobenú na získavanie potrebných dát s využitím pôvodných metód. Okrem tejto funkcie boli pridané aj nové pomocné funkcie a funkcionality pre získanie referenčnej tabuľky a JavaScript kódu. Analyzátor je napísaný v jedinom súbore a nie je možné ho nainštalovať. Nástroj sa nachádza v súbore `pdf_analyser_didierstevens.py` a všetky úpravy kódu sú vyznačené.

Pre spojenie výstupov z oboch analyzátorov slúži trieda `ExtractedInfo`. Jej atribúty sú rovnaké ako dáta, ktoré je možné získať z oboch analyzátorov. Ako hlavný výstup analyzátorov je objekt tejto triedy.

Na obhospodarovanie a zjednotenie oboch analyzátorov slúži nástroj `analyze.py`. Zároveň slúži aj ako prostredník medzi externými nástrojmi a nástrojom Clusty. Jeho výstupom sú dáta získané z oboch analyzátorov, upravené do vhodnej formy (napr. filtrovanie, hashovanie), vypísané na štandardný výstup vo formáte JSON.

O analýzu PDF súborov sa v nástroji Clusty stará trieda `PDFAnalyzer` v module `clusty.pdf`. Pomocou modulu `cmd_runner` je zavolaný analyzátor `analyze.py` a jeho výstup vo forme JSON je použitý pri vytvorení objektu triedy `PDFInfo`. Tento objekt je vrátený hlavnému analyzátoru `SampleAnalyzer`, kde je použitý pri tvorbe objektu triedy `PDFSample`.

6.2 Analýza LNK

Statická analýza LNK súborov je poskytovaná nástrojom `LnkParse3`[1], čo je vylepšená verzia nástroja `LnkParse` spomínaného v sekcii 5.2.2. Na pôvodny nástroj bola vytvorená žiadosť na opravu chýb a migráciu na Python 3 (angl. *pull request*), ktorá však dlhý čas nemala žiadnu odpoveď, a tak bol vytvorený nový nástroj na základe pôvodného analyzátora. Okrem migrácie a opravy chýb boli do-implementované i nové metódy na spracovanie aktuálne nepodporovaných štruktúr formátu a taktiež bol upravený výstupný JSON formát. Nástroj je možné nainštalovať pomocou inštalátora `pip`. Práca s týmto nástrojom je implementovaná v súbore `analyze.py`, ktorý zároveň slúži aj ako prostredník medzi externým nástrojom `LnkParse3` a nástrojom Clusty. Jeho výstupom je, ako aj pri PDF, JSON extrahovaných informácii.

O analýzu LNK súborov sa v nástroji Clusty, podobne ako aj o analýzu PDF, stará trieda `LNKAnalyzer` v module `clusty.lnk`. Pomocou modulu `cmd_runner` je zavolaný analyzátor `analyze.py` a jeho výstup vo forme JSON je použitý pri vytvorení objektu triedy `LNKFInfo`. Tento objekt je vrátený hlavnému analyzátoru `SampleAnalyzer`, kde je použitý pri tvorbe objektu triedy `LNKSample`.

6.3 Zhluková analýza

Zhluková analýza je implementovaná v jazyku C++.

Všetky dáta, ktoré sa používajú pri zhlukovaní, majú definovanú vlastnú metódu pre prepočítanie daného atribútu po pridaní novej vzorky do zhluku. Ak je hodnota atribútu všetkých prepočítavaných vzoriek rovnaká, bude atribút zobrazený vo výslednom zhluku.

Atribúty, ktoré sa zobrazujú vo výsledných zhlukoch a pri pridaní každej vzorky sa znova prepočítajú, sú definované v súbore `properties_computation.cpp`. Atribúty určené priamo na zhlukovanie sú definované v Python (`clustering_method.py`) aj v C++ (`clustering_methods.cpp`) časti. Priorita vybraných atribútov, ako aj jednotlivé metódy pre zhlukovanie, sú implementované v triedach `PDFPlacer` a `LNKPlacer`.

PDF

Dáta sú pri zhlukovaní načítané z databázy a použité na vytvorenie objektu triedy `PDFSample`, pričom nie všetky dáta z databázy sú nutne použité. Zároveň sú však pri konštrukcií triedy vytvorené nové dáta, konkrétne atribút `info`, ktorý je tvorený jednou z troch možných kombinácii základných atribútov. Tento atribút je vytvorený v prípade, že vzorka obsahuje jednu z nasledujúcich kombinácii atribútov:

- `trailer_root`, `trailer_size`, `trailer_info`, `metadata_producer`
- `trailer_root`, `trailer_size`, `trailer_info`
- `trailer_root`, `trailer_size`, `version`

Kombinácie základných atribútov sú zoradené podľa priority zostupne a pri vytvorení určitej kombinácie sú príslušné základné atribúty odstránené z dôvodu duplicity pri výslednom výpise.

[1] https://github.com/Matmaus/LnkParse3

LNK

Proces je podobný ako pri PDF. Z dát získaných z databázy sa vytvorí objekt `LNKSample`. Pri konštrukcií triedy sa vytvoria aj tri nové atribúty, konkrétne `folder_app_and_cli`. `drive_and_cli` a `volume_and_cli`.

Na rozdiel od PDF, kde bol vytvorený jeden nový atribút, ktorý môže vzniknúť troma rôznymi kombináciami základných atribútov, pri LNK sa pre každú takúto kombináciu vytvorí zvlášť nový atribút. Je to z dôvodu, že počas procesu pridávania podpory pre nové formáty bola pridaná nová funkcionalita nástroja Clusty, ktorá umožňuje v určitom prípade prezhlukovanie už existujúcich zhlukov, s vylúčením metódy, podľa ktorej bol zhluk vytvorený, prípadne i ďalších metód. Napríklad v prípade prezhlukovania zhluku vytvoreného na základe `folder_app_and_cli` sa môže vytvoriť nový zhluk na základe atribútu `volume_and_cli`, čo pri PDF atribúte `info` nie je možné. Atribúty vznikajú len v prípade, že vzorka obsahuje ľubovolné konzolové argumenty. Spomínané atribúty vznikajú nasledovnými kombináciami:

- `extra_known_folder_id`, `data_application`
  a príznak prítomnosti `data_command_line_arguments`
- `link_info_drive_serial_number`
  a príznak prítomnosti `data_command_line_arguments`
- `extra_birth_droid_volume_id`
  a príznak prítomnosti `data_command_line_arguments`

Typy zhlukovacích metód

Pre zhlukovanie vzoriek sa aktuálne používajú dva prístupy, a to absolútna zhoda atribútov a podobnosť pomocou porovnania ssdeep hashu. Metóda porovnania ssdeep hashu sa používa ako posledná možná metóda na zhlukovanie. Používa ssdeep hash vytvorený z obsahu vzorky. Vzorky, ktoré sú si podobné, majú aj podobný ssdeep hash a skóre získané porovnaním hashov je blízke hodnote 100. Čím sú vzorky odlišnejšie, tým je získané skóre menšie. Pri porovnaní takýchto hashov je vopred určený prah podobnosti a v prípade, že podobnosť vzoriek je nad tento prah, vzorky sú označené ako podobné. Nevýhodou tejto metódy je časová zložitosť.

Okrem tohoto porovnávania na základe podobnosti sa však všetky ostatné atribúty porovnávajú na absolútnu zhodu. Príklad metódy na zhlukovanie je možné vidieť v príklade 6.1.

```
bool PDFPlacer::place_sample_by_pdf_info_100_match(const PDFSample* sample) {
   return place_sample_in_cluster(
      sample,
      "pdf_info_100_match",
      [](auto sample) { return sample->has_info(); },
      [](auto sample) { return sample->get_info(); }
   );
}
```

Kód 6.1: Jedna z metód zhlukovania PDF vzoriek

Podmienka vo vyššie spomínanom príklade 6.1 `sample->has_info();` rozhoduje o tom, či je možné vzorku zhlukovať podľa danej metódy alebo nie. Do rozhodovania je možné pri-

dať aj viacero ďalších faktorov. Napríklad v prípade, že niektoré hodnoty atribútov nie sú vhodné na zhlukovanie, je možné ich pridať na čiernu listinu. Atribúty s takýmito hodnotami nebudú použité pri zhlukovaní. Ďalším príkladom je, že niektoré metódy potrebujú určité okolnosti, aby mohli byť použité. V príklade 6.2 je možno vidieť príklad metódy pre zhlukovanie LNK súborov, ktorá okrem prezencie daného atribútu vyžaduje taktiež absenciu atribútu `data_command_line_arguments` a taktiež filtruje generické hodnoty atribútu.

```
bool LNKPlacer::place_sample_by_lnk_extra_birth_droid_volume_id(const LNKSample* sample) {
   return place_sample_in_cluster(
      sample,
      "lnk_extra_birth_droid_volume_id",
      [](auto sample) {
         return sample->has_extra_birth_droid_volume_id() &&
         !sample->has_data_command_line_arguments() &&
         !is_generic_lnk_extra_droid_volume_id(sample->get_extra_birth_droid_volume_id());
      },
      [](auto sample) { return sample->get_extra_birth_droid_volume_id(); }
   );
}
```

Kód 6.2: Jedna z metód zhlukovania LNK vzoriek

Kapitola 7

Testovanie

V priebehu vývoja bolo vytvorených a prevedených množstvo testov. Testovaná bola správna funkčnosť Python aj C++ časti, vytvorených analyzátorov (`analyze.py`) i výsledné zhluky a ich kvalita. Jednotlivé časti sú popísané nižšie.

7.1 Testovanie implementácie v Python časti

Implementácia bola testovaná jednotkovými i integračnými testami. Jednotkové testy boli implementované s použitím modulu `unittest`. Mieru pokrytia modifikovaných či novovzniknutých súborov jednotkovými testami je možné vidieť v tabuľke 7.1. Testy boli spúšťané nástrojom `nose`[1]. Výsledky testov tried pre reprezentáciu vzoriek je možné vidieť v prílohe B.1 a výsledky testov implementovaných analyzátorov je možné vidieť v prílohe B.2. Celkovo bolo pridaných 171 jednotkových testov.

Súbor	Pokrytie riadkov	Bez pokrytia	Pokrytie %
clusty/lnk.py	164	0	100 %
clusty/pdf.py	131	0	100 %
clusty/samples/lnk_sample.py	141	0	100 %
clusty/samples/pdf_sample.py	108	0	100 %
tests/real/analysis/lnk_tests.py	92	0	100 %
tests/real/analysis/pdf_tests.py	299	0	100 %
tests/unit/lnk_tests.py	40	0	100 %
tests/unit/pdf_tests.py	40	0	100 %
tests/unit/samples/lnk_tests.py	288	0	100 %
tests/unit/samples/pdf_tests.py	229	0	100 %

Tabuľka 7.1: Miera pokrytia riadkov kódu jednotkovými testami (Python)

Integračné testy boli implementované s použitím modulu `TestsBase`. Zahŕňajú jednak vopred vytypované vzorky, a jednak vzorky, ktoré boli pridané postupne na základe nových zistený neočakávaného správania pri poškodených vzorkách. Výsledky je možné vidieť v prílohe v ukážke B.3.

[1] https://nose.readthedocs.io/en/latest/

7.2 Testovanie implementácie v C++ časti

Implementácia C++ bola testovaná jednotkovými i integračnými testami za pomoci nástroja Google Test[2]. Mieru pokrytia jednotkovými i integračnými testami je možné vidieť v tabuľke 7.2. Výsledky jednotkových testov je možné vidieť v prílohe B.4 a výsledky integračných testov v prílohe B.5. Celkovo bolo pridaných 237 jednotkových 13 integračných testov.

Súbor	Pokrytie	Bez pokrytia	Pokrytie %
src/clusty/blacklists.cpp	111	1	99,1 %
src/clusty/placers/lnk_placer.cpp	62	2	96,9 %
src/clusty/placers/lnk_placer.hpp	2	0	100 %
src/clusty/placers/pdf_placer.cpp	25	2	92,6 %
src/clusty/placers/pdf_placer.hpp	2	0	100 %
src/clusty/samples/lnk_sample.cpp	153	1	99,4 %
src/clusty/samples/lnk_sample.hpp	86	0	100 %
src/clusty/samples/pdf_sample.cpp	125	4	96,9 %
src/clusty/samples/pdf_sample.hpp	66	0	100 %
tests/support/samples/lnk_sample_builder.cpp	3	0	100 %
tests/support/samples/lnk_sample_builder.hpp	32	0	100 %
tests/support/samples/pdf_sample_builder.cpp	3	0	100 %
tests/support/samples/pdf_sample_builder.hpp	62	0	100 %
tests/unit/samples/lnk_sample_tests.cpp	583	0	100 %
tests/unit/samples/pdf_sample_tests.cpp	448	0	100 %

Tabuľka 7.2: Miera pokrytia riadkov kódu jednotkovými a integračnými testami (C++)

7.3 Testovanie výsledných zhlukov a ich kvality

Vytvorené zhluky boli najskôr automatizovane analyzované pomocou už existujúcej časti nástroja Clusty. Tá na základe dostupných informácii zo služby VirusTotal ktorá klasifikovala známe vzorky. Následne je na základe prvých 1 000 vzoriek klasifikovaný i celý zhluk. Pri prevahe škodlivých vzoriek je označený ako *malware*, pri prevahe čistých vzoriek je označený ako *clean* a pri nejednoznačnosti ako *unknown*. Každý zhluk obsahuje taktiež i dôveryhodnosť (*Confidence*) tohoto úsudku.

Testovanie kvality vytvorených a analyzovaných zhlukov bolo vykonávané manuálne. Boli vyberané náhodné zhluky, prevažne tie, ktoré boli označené ako škodlivé a mali nízku dôveryhodnosť. Dôvodom bolo overenie, či sú zhluky z rovnakých rodín a či sa v zhluku nenachádzajú aj iné rodiny či dokonca vzorky označené ako čisté.

Vzorky v jednotlivých zhlukoch boli vyberané taktiež náhodne, no primárne tie, ktoré mali atypickú veľkosť či klasifikáciu. Potenciálne rôzne vyzerajúce vzorky boli stiahnuté a manuálne prehliadané a porovnávané z hľadiska podobnosti súboru, dokumentu i správania.

Pri objavení nezhody vzoriek v zhluku bola metóda považovaná za nie príliš vhodnú na zhlukovanie.

[2]https://github.com/google/googletest

Kapitola 8

Zhodnotenie

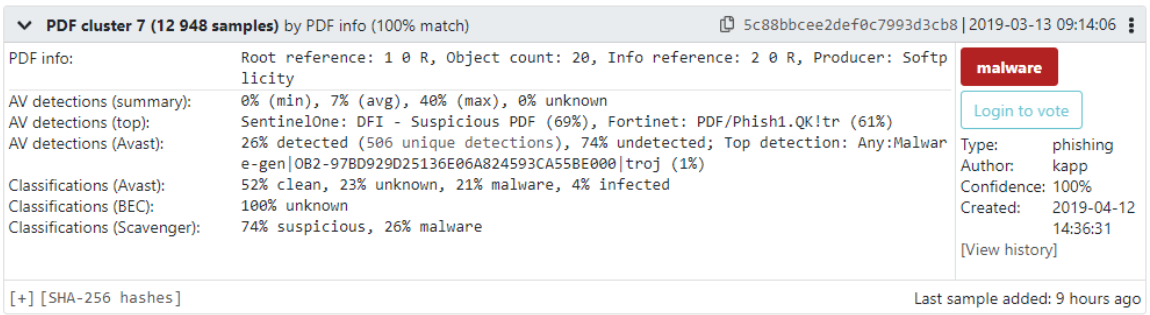
Rozšírenie nástroja Clusty o nové formáty súborov PDF a LNK, navrhnuté v kapitole 5 a implementované tak, ako je popísané v kapitole 6, bolo úspešne nasadené do produkcie. Denne je spracovaných desiatky tisíc vzoriek oboch formátov.

Výsledné zhluky

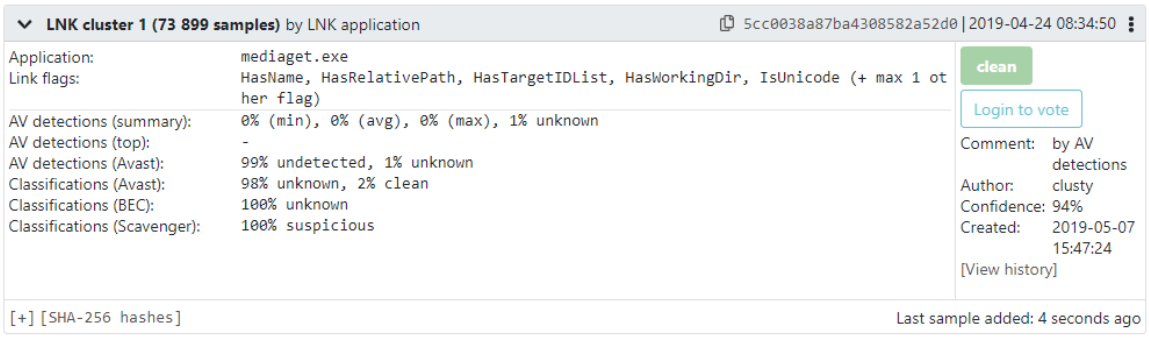
Na obrázku 8.1 je možné vidieť jeden z výsledných zhlukov z novo pridanej kategórie PDF. Tento zhluk bol vytvorený na základe atribútu `PDF info`, ktorý je zároveň jediným spoločným atribútom medzi vzorkami v danom zhluku. Zhluk obsahuje 12 948 vzoriek a bol označený ako `malware` analytikom spoločnosti Avast, preto je dôveryhodnosť úsudku 100%. Pri zobrazení histórie je možné vidieť predchádzajúce klasifikácie zhluku. Predchádzajúca klasifikácia tohoto zhluku bola taktiež `malware`, ale bola určená automatizovane a jej dôveryhodnosť bola len 8%.

Všeobecne sú zhluky vytvárané pri reálnej prevádzke tvorené priemerne z 3-4 atribútov (zložený atribút `PDF info` je počítaný ako jeden atribút). Zhluky vytvorené na základe atribútu `PDF info` mávajú menej spoločných atribútov, čo svedčí o väčšej rozmanitosti obsiahnutých vzoriek v pozitívnom zmysle. Naopak, zhluky vytvorené na základe jedného z atribútov `First page text hash` a `Last page text hash` mávajú typicky 6-7 spoločných atribútov.

Jeden zo zhlukov novopridanej kategórie LNK je možné vidieť na obrázku 8.2. Zhluk bol vytvorený na základe atribútu `Application`. Podmienkou vzniku je absencia konzolových argumentov.Z tohto dôvodu sa v hodnote atribútu `Link flags` nenachádza položka `HasArguments`. Skladá sa z 72 899 vzoriek.



Obr. 8.1: Zobrazenie zhluku v nástroji Clusty



Obr. 8.2: Zobrazenie zhluku v nástroji Clusty

Vyhodnotenie splnenia zadania

V rámci tejto práce boli splnené všetky body zadania:

- Nástroj Clusty, slúžiaci na zhlukovú analýzu súborov, je popísaný v kapitole 2.
- Na jeho rozšírenie boli vybrané dva typy súborov PDF a LNK, ktoré sú popísané v kapitole 3 a 4.
- Vytvorený návrh spomínaného rozšírenie, vrátane výberu vhodného nástroja, je popísaný v kapitole 5. Na analýzu formátu PDF bola vybraná kombinácia nástrojov `PyMuPDF` a `pdf-parser`. Na analýzu LNK bol vybraný nástroj `LnkParse3`.
- Navrhnuté riešenie bolo implementované tak, ako je popísané v kapitole 6.
- Implementácia bola otestovaná sadov jednotkových a integračných testov. Testovanie je popísané v kapitole 7.
- Spomínané rozšírenie bolo nasadené do každodennej prevádzky.

Kapitola 9

Záver

Cieľom práce bolo rozšíriť nástroj na zhlukovú analýzu súborov o podporu nových formátov PDF a LNK, konkrétne bol predstavený nástroj Clusty vyvíjaný spoločnosťou Avast. Rozoberané boli taktiež spomínané formáty z hľadiska identifikácie, strojovej analýzy a extrakcie unikátnych vlastností.

Na základe poznania daného formátu boli vytypované určité vlastnosti a preskúmané existujúce nástroje na ich extrakciu. Nástroje vybrané pre oba formáty boli mierne upravené a použité pri extrakcii vlastností použitých ako vstupné informácie o vzorke pre nástroj Clusty.

Zo všetkých získaných atribútov (vlastností) boli vybrané najzaujímavejšie, ktoré sú uložené v databáze a použité pri zhlukovaní. Z nich boli ďalej vybrané atribúty, prípadne kombinácie atribútov, ktoré sa ukázali ako vhodné na zhlukovanie.

Počas celého procesu integrácie, boli novo pridané časti nástroja niekoľko-krát testované jednotkovými i integračnými testami. Výsledná práca na integrácii oboch formátov bola nasadená do produkcie a každodenne prebieha zhlukovanie podľa vybraných metód.

V budúcnosti by bolo dobré nájsť nové atribúty, prípadne kombinácie na zhlukovanie PDF vzoriek. Atribút `info` vzniknutý jednou z troch možných kombinácii sa ukázal ako kvalitný, avšak to by sa časom mohlo zmeniť.

Pri formáte LNK by bolo vhodné určitým spôsobom detekovať reťazce náhodných znakov, a tie nahradiť reťazcom generickým. Dôvodom je, že atribúty sú porovnávané na absolútnu zhodu, a tak čo i len najmenšia nezhoda znamená zaradenie do odlišného zhluku. Príkladom môže byť doplnenie náhodného reťazca do konzolových argumentov (reťazec nezmení chovanie):

```
/c start \"streamerdata\\stream.txt\" PklQR14zZOXHjHn  & exit
/c start \"streamerdata\\stream.txt\" f1GlhuXPzNlF8FN  & exit
/c start \"streamerdata\\stream.txt\" 1RwFofA1FLWOqlg  & exit
/c start \"streamerdata\\stream.txt\" [RANDOM_STRING]  & exit
```

Literatúra

[1] *PDF: tři písmena, která změnila svět*. [Online; navštívené 17.12.2018].
URL https://acrobat.adobe.com/cz/cs/acrobat/about-adobe-pdf.html

[2] Arefkhani, M.; Soryani, M.: Malware clustering using image processing hashes. In *2015 9th Iranian Conference on Machine Vision and Image Processing (MVIP)*, November 2015, ISSN 2166-6784, s. 214–218, doi:10.1109/IranianMVIP.2015.7397539.

[3] Cherepanov, A.: *A tale of two zero-days*. Máj 2018.
URL https://www.welivesecurity.com/2018/05/15/tale-two-zero-days

[4] *Analyzing the Windows LNK file attack method*. Február 2019, [Online; navštívené 20.04.2019].
URL https://dexters-lab.net/2019/02/16/analyzing-the-windows-lnk-file-attack-method

[5] Fortin, M.; Goktepe, M.: *Application compatibility in the Windows ecosystem*. Január 2019, [Online; navštívené 20.04.2019].
URL https://blogs.windows.com/windowsexperience/2019/01/15/application-compatibility-in-the-windows-ecosystem/#8gm3U75goUOUyrOU.97

[6] Hager, J.: *The Windows Shortcut File Format*. Máj 2009, verzia 1.0.
URL https://ithreats.files.wordpress.com/2009/05/lnk_the_windows_shortcut_file_format.pdf

[7] Hassan, N.: *Windows Forensics Analysis: A Practical Guide Using Windows OS*, kapitola 7. Apress, Január 2019, ISBN 978-1-4842-3837-0, s. 223–224, doi:10.1007/978-1-4842-3838-7_7.

[8] Křoustek, J.; Zemek, P.: *Kategorizace malware vzorků*. Interní dokumentace společnosti Avast.

[9] Laing, B.: *Malicious Email Attachments – Protection from Infected PDF files*. August 2017, [Online; navštívené 13.01.2019].
URL https://www.lastline.com/blog/malicious-email-attachments

[10] Lewiecki, M.: *Taking Documents to the Next Level with PDF 2.0*. September 2017, [Online; navštívené 17.12.2018].
URL https://theblog.adobe.com/taking-documents-to-the-next-level-with-pdf-2-0

[11] *PDF file format*. [Online; navštívené 19.12.2018].
URL http://lotabout.me/orgwiki/pdf.html

[12] Lukan, D.: *PDF File Format: Basic Structure*. Jún 2018, [Online; navštívené 19.12.2018].
URL https://resources.infosecinstitute.com/pdf-file-format-basic-structure/#gref

[13] Matrosov, A.; Rodionov, E.; Harley, D.; aj.: *Stuxnet Under the Microscope*. 2010.
URL https://www.esetnod32.ru/company/viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf

[14] Metz, J.: *Windows Shortcut File format specification*. Marec 2013, verzia 0.0.12.
URL https://www.dfir.training/windows/lnk/116-windows-shortcut-file-lnk-format/file

[15] *Shell Link (.LNK) Binary File Format*. September 2018, [Online; navštívené 20.04.2019].
URL https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943

[16] *Shell Links*. Máj 2018, [Online; navštívené 20.04.2019].
URL https://docs.microsoft.com/en-us/windows/desktop/shell/links

[17] Mimoso, M.: *Stuxnet LNK Exploits Still Widely Circulated*. Apríl 2017, [Online; navštívené 29.04.2019].
URL https://threatpost.com/stuxnet-lnk-exploits-still-widely-circulated/125089

[18] Oh, M.: *Taking apart a double zero-day sample discovered in joint hunt with ESET*. Júl 2018.
URL https://www.microsoft.com/security/blog/2018/07/02/taking-apart-a-double-zero-day-sample-discovered-in-joint-hunt-with-eset

[19] Rosenthol, L.: *Developing with PDF*. [Online; navštívené 18.01.2019].
URL https://www.oreilly.com/library/view/developing-with-pdf/9781449327903/ch01.html

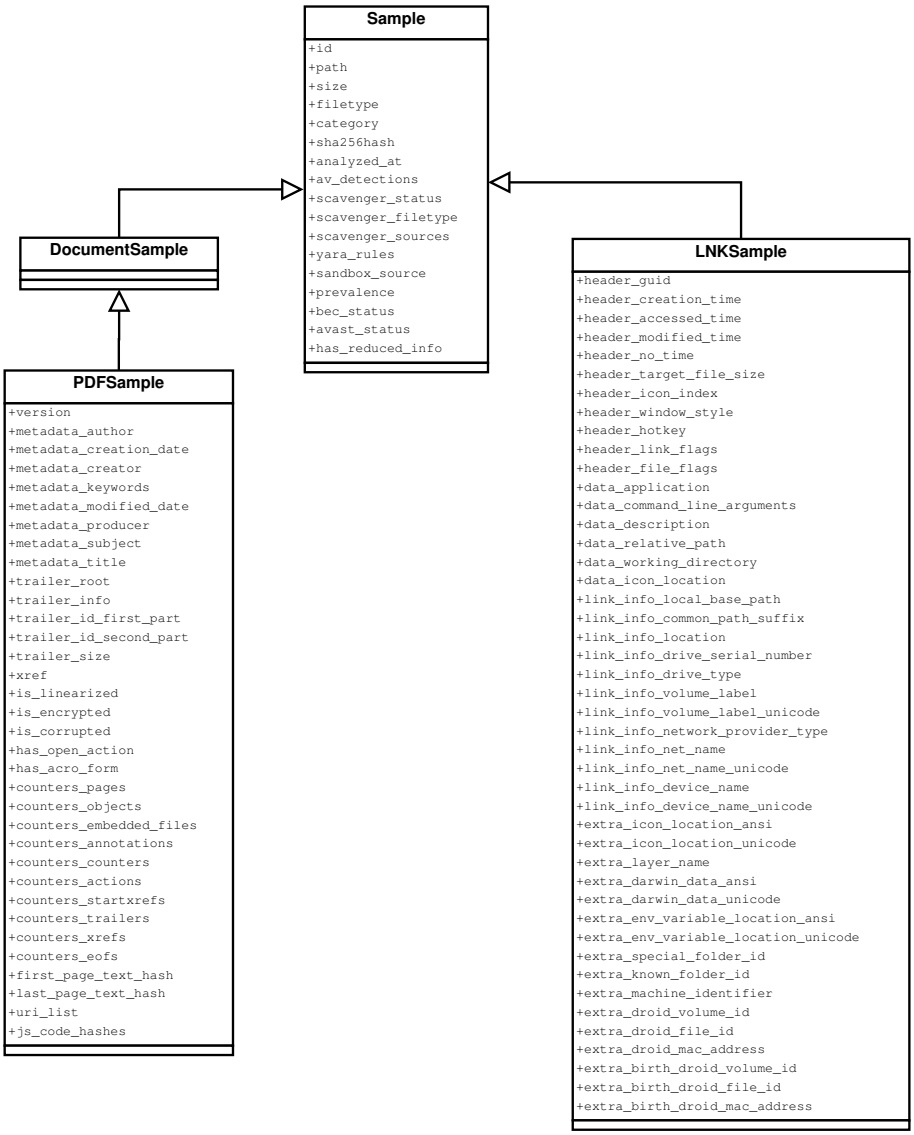
[20] Singh, N.; Khurmi, S. S.: Malware Analysis, Clustering and Classification: A Literature Review. In *International Journal of Computer Science And Technology*, ročník 6, Marec 2015, ISSN 0976-8491.

[21] Whitington, J.: *PDF Explained*. O'Reilly, 2011, ISBN 978-1-449-31002-8.

[22] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Získávání znalostí z databází - ZZN*. September 2010.

Príloha A

Úplný diagram tried



Obr. A.1: Diagram hierarchie tried reprezentujúcich PDF a LNK vzorky

Príloha B

Výstupy testov

```
[clusty]$ nosetests tests/unit/samples/lnk_sample_tests.py
.....................................................................................

----------------------------------------------------------------------
Ran 93 tests in 0.020s

OK
[clusty]$ nosetests tests/unit/samples/pdf_sample_tests.py
.....................................................................
----------------------------------------------------------------------
Ran 72 tests in 0.011s

OK
```

Kód B.1: Spustenie jednotkových testov pre triedy vzoriek (Python)

```
[clusty]$ nosetests tests/unit/lnk_tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.004s

OK
[clusty]$ nosetests tests/unit/pdf_tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.004s

OK
```

Kód B.2: Spustenie jednotkových testov pre analyzátory (Python)

```
[clusty]$ make real-tests
...
test_lnk_example_is_analyzed_correctly (tests.real.analysis.lnk_tests.LNKTests) ... ok
test_lnk_with_almost_all_attributes_is_analyzed_correctly (tests.real.analysis.lnk_tests.
    LNKTests) ... ok
test_lnk_with_network_properties_is_analyzed_correctly (tests.real.analysis.lnk_tests.
    LNKTests) ... ok
test_lnk_without_any_time_is_analyzed_correctly (tests.real.analysis.lnk_tests.LNKTests)
    ... ok
...
test_pdf_version_2_0_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests) ... ok
test_pdf_version_2_0_via_incremental_save_is_analyzed_correctly (tests.real.analysis.
    pdf_tests.PDFTests) ... ok
test_pdf_version_2_0_with_UTF_8_string_and_annotation_is_analyzed_correctly (tests.real.
    analysis.pdf_tests.PDFTests) ... ok
test_pdf_version_2_0_with_image_with_BPC_is_analyzed_correctly (tests.real.analysis.
    pdf_tests.PDFTests) ... ok
test_pdf_version_2_0_with_offset_start_is_analyzed_correctly (tests.real.analysis.pdf_tests
    .PDFTests) ... ok
test_pdf_version_2_0_with_page_level_output_intent_is_analyzed_correctly (tests.real.
    analysis.pdf_tests.PDFTests) ... ok
test_pdf_with_annotation_and_embedded_file_and_metadata_is_analyzed_correctly (tests.real.
    analysis.pdf_tests.PDFTests) ... ok
test_pdf_with_corrupted_version_is_analyzed_correctly (tests.real.analysis.pdf_tests.
    PDFTests) ... ok
test_pdf_with_corruption_and_unfinished_pdf_name_is_analyzed_correctly (tests.real.analysis
    .pdf_tests.PDFTests) ... ok
test_pdf_with_corruption_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests) ...
     ok
test_pdf_with_encryption_and_simple_password_is_analyzed_correctly (tests.real.analysis.
    pdf_tests.PDFTests) ... ok
test_pdf_with_linearisation_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests)
    ... ok
test_pdf_with_multiple_different_trailers_is_analyzed_correctly (tests.real.analysis.
    pdf_tests.PDFTests) ... ok
test_pdf_with_obfuscation_and_js_code_is_analyzed_correctly (tests.real.analysis.pdf_tests.
    PDFTests) ... ok
test_pdf_with_trash_before_header_and_after_eof_is_analyzed_correctly (tests.real.analysis.
    pdf_tests.PDFTests) ... ok
test_pdf_with_wrong_object_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests)
    ... ok
test_pdf_with_wrong_offset_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests)
    ... ok
test_pdf_without_eof_is_analyzed_correctly (tests.real.analysis.pdf_tests.PDFTests) ... ok
...
```

Kód B.3: Spustenie integračných testov (Python)

```
[build]$ ./tests/unit/clusty_unit_tests
[==========] Running 1028 tests from 105 test suites.
[----------] Global test environment set-up.
...
[----------] 43 tests from BlacklistsTests
...
[ RUN      ] BlacklistsTests.
    is_generic_lnk_extra_droid_file_id_returns_false_if_lnk_extra_droid_file_id_is_not_generic

[       OK ] BlacklistsTests.
    is_generic_lnk_extra_droid_file_id_returns_false_if_lnk_extra_droid_file_id_is_not_generic
     (0 ms)
[----------] 43 tests from BlacklistsTests (2 ms total)
...
[----------] 129 tests from LNKSampleTests
...
[ RUN      ] LNKSampleTests.get_link_info_device_name_unicode_returns_correct_value
[       OK ] LNKSampleTests.get_link_info_device_name_unicode_returns_correct_value (0 ms)
[----------] 129 tests from LNKSampleTests (3 ms total)
...
[----------] 102 tests from PDFSampleTests
...
[ RUN      ] PDFSampleTests.has_js_code_hashes_returns_correct_value
[       OK ] PDFSampleTests.has_js_code_hashes_returns_correct_value (0 ms)
[ RUN      ] PDFSampleTests.has_js_code_hashes_returns_true_when_there_is_js_code_hashes
[       OK ] PDFSampleTests.has_js_code_hashes_returns_true_when_there_is_js_code_hashes (0
     ms)
[ RUN      ] PDFSampleTests.
    has_js_code_hashes_returns_false_when_there_is_no_js_code_hashes
[       OK ] PDFSampleTests.
    has_js_code_hashes_returns_false_when_there_is_no_js_code_hashes (0 ms)
[----------] 102 tests from PDFSampleTests (2 ms total)
...
[----------] Global test environment tear-down
[==========] 1028 tests from 105 test suites ran. (102 ms total)
[  PASSED  ] 1028 tests.
```

Kód B.4: Spustenie jednotkových testov (C++)

```
[build]$ ./tests/real/clusty_real_tests
[==========] Running 321 tests from 26 test suites.
[----------] Global test environment set-up.
...
[----------] 8 tests from LNKPlacerTests
...
[ RUN      ] LNKPlacerTests.
    place_sample_places_lnk_sample_with_data_application_in_data_application_cluster
[       OK ] LNKPlacerTests.
    place_sample_places_lnk_sample_with_data_application_in_data_application_cluster (519
    ms)
[----------] 8 tests from LNKPlacerTests (3867 ms total)
...
[----------] 5 tests from PDFPlacerTests
...
[ RUN      ] PDFPlacerTests.
    place_sample_places_pdf_sample_with_last_page_text_hash_in_last_page_text_hash_cluster
[       OK ] PDFPlacerTests.
    place_sample_places_pdf_sample_with_last_page_text_hash_in_last_page_text_hash_cluster
    (581 ms)
[----------] 5 tests from PDFPlacerTests (2906 ms total)
...
[----------] Global test environment tear-down
[==========] 321 tests from 26 test suites ran. (207380 ms total)
[  PASSED  ] 321 tests.
```

Kód B.5: Spustenie integračných testov (C++)

Príloha C

Obsah priloženého média

Priložené DVD obsahuje nasledovnú adresárovú štruktúru:

README.txt
: Dokument obsahujúci základné informácie o obsahu DVD.

thesis.pdf
: PDF dokument obsahujúci text bakalárskej práce.

thesis/
: Zložka obsahujúca zdrojové súbory bakalárskej práce (LaTeX).

clusty/
: Zložka obsahujúca zdrojové súbory nástroja Clusty. Prítomné sú len súbory vytvorené alebo upravené v rámci bakalárskej práce. Stromová štruktúra priečinkov je zachovaná.

parsers_lnk/
: Zložka obsahujúca testované LNK analyzátory a testy.

parsers_pdf/
: Zložka obsahujúca testované PDF analyzátory a testy.