EXTENSION OF TOOL FOR EXECUTABLE-FILE ANALYSIS

Milan Zavoral
Bachelor Degree Programme (3), FIT BUT
E-mail: xzavor02@stud.fit.vutbr.cz

Supervised by: Jakub Křoustek
E-mail: ikroustek@fit.vutbr.cz

Abstract: Retargetable machine-code decompilation is used for a platform-independent transformation of input executable files into a high-level source code (e.g., C language, Python language). Accurate analysis of input executable files and gathering information about them is a crucial prerequisite in order to achieve the best decompilation results. This paper presents work focused on extension and optimization of tool for executable-file analysis, which is developed within the Lissom project at Faculty of Information Technology, Brno University of Technology.

Keywords: reverse engineering, retargetable decompilation, executable file analysis, Lissom

1 ÚVOD

Rekonfigurovatelný zpětný překladač se skládá ze dvou základních částí. Jednou z nich je vlastní zpětný překladač (dekompilátor), který zajišťuje zpětný překlad a je nezávislý na formátu vstupního souboru a cílové architektuře. Před spuštěním zpětného překladu je však nejprve nutné spustit druhou základní část – fázi předzpracování dat. Jedním z nástrojů spouštěných ve fázi předzpracování dat je i nástroj pro analýzu spustitelných souborů `fileinfo`, o němž pojednává tento článek.

2 MOTIVACE

Pro dosažení kvalitních výstupů zpětného překladače, srovnatelných co do čitelnosti s originálními zdrojovými kódy, je třeba o vstupním souboru mít řadu informací. Zpětný překladač je nezávislý na formátu vstupního souboru, proto je třeba tyto informace shromažďovat při předzpracování. Jednou z nejdůležitějších je informace o nástroji, kterým byl soubor vytvořen. Takovým nástrojem může být buď překladač, nebo tzv. *packer* (nástroj pro kompresi a ochranu spustitelných souborů). Informace o překladači je důležitá např. pro rekonstrukci instrukčních idiomů, která probíhá při zpětném překladu [1]. Zcela klíčová je informace o použitém packeru, na základě které probíhá dekomprese a odstranění ochran spustitelného souboru. Pokud není packer korektně detekován, zpětný překlad skončí neúspěšně. Mezi další informace, s jejichž znalostí lze vylepšit výstupy zpětného překladu patří např. informace o sekcích, symbolech, relokacích či záznamech pro dynamické linkování.

3 EXISTUJÍCÍ NÁSTROJE PRO ANALÝZU SPUSTITELNÝCH SOUBORŮ

Existuje několik nástrojů určených pro analýzu spustitelných souborů (např. `objdump`, `readelf`, `PE.Explorer` a `efd`). Žádný z těchto nástrojů však nelze využít pro účely zpětného překladu (u `objdump` je problémem použitá licence, `readelf` a `PE.Explorer` podporují pouze jeden souborový formát, `efd` je proprietární a dostupný pouze pro architekturu Intel x86). Žádný program dále nedokáže detekovat nástroj, kterým byl soubor vytvořen. To platí i opačně – programy pro detekci použitého nástroje neumí získávat ostatní informace. Jediným přijatelným řešením pro rekonfigurovatelný zpětný překladač je tak vytvoření vlastního nástroje pro analýzu spustitelných souborů. Tímto nástrojem je právě aplikace `fileinfo`.

4 NÁSTROJ FILEINFO

Tato kapitola popisuje stav aplikace před začátkem řešení mojí bakalářské práce a následně vlastní přínos této práce. V závěru kapitoly jsou pak diskutovány experimentální výsledky.

4.1 VÝCHOZÍ STAV

Před začátkem řešení bakalářské práce byla aplikace `fileinfo` o vstupním souboru schopna zjistit tyto informace: (1) souborový formát, (2) cílovou architekturu, (3) obsah vstupního bodu (první vykonané instrukce po načtení spustitelného souboru do paměti), (4) nástroj, kterým byl soubor vytvořen (překladač nebo packer). Podporované formáty vstupních spustitelných souborů byly PE a ELF.

4.2 RYCHLOST DETEKCE POUŽITÉHO PŘEKLADAČE ČI PACKERU

Závažným problémem byla rychlost detekce použitého překladače či packeru. Detekce je založena na vyhledávání v signaturách. Signatura je „podpis“ nástroje, kterým byl soubor vytvořen. Je reprezentována hodnotami jednotlivých půlbajtů (podrobněji viz [2]) a offsetem, který určuje oblast jejího vyhledávání v souboru. Protože offset byl reprezentován v řetězcové podobě (tz. číslo reprezentující offset bylo uloženo jako pole znaků), bylo jeho zpracování pomalé. První optimalizací byl proto převod offsetu do celočíselné reprezentace. Tato a další optimalizace (optimalizace algoritmů pro vyhledávání v signaturách, odstranění iterativně vytvářených proměnných, profilování, kompletní refaktoring) přinesly téměř osminásobné zrychlení oproti původní verzi.

4.3 POČET ZÍSKÁVANÝCH INFORMACÍ

Dalším problémem byl nízký počet získávaných informací. Analýza vstupního souboru byla proto rozšířena, díky čemuž je nyní aplikace schopna získat informace např. o typu souboru (dynamicky linkovaná knihovna vs. klasický spustitelný soubor), endianitě (pořadí bajtů), sekcích, symbolech, relokacích, záznamech pro dynamické linkování, segmentech (ELF), adresářích dat (PE), příznacích (*flags*), velikosti a obsahu souborových hlaviček aj. Implementovány byly také specifické analýzy pro architekturu ARM, které ve spustitelném souboru dokáží rozpoznat kód instrukčního rozšíření THUMB. Tato informace je důležitá pro další části zpětného překladače.

Informace jsou zapisovány do konzole a do souboru ve formátu XML. Konzolový výstup je přehledný pro uživatele, naopak pomocí XML souboru lze informace snadno předávat dalším částem zpětného překladače.

Vedle počtu informací získávaných o souborech ve formátech PE a ELF byla také přidána podpora pro textový spustitelný formát LOFF (Lissom Object File Format), který je využíván v několika nástrojích projektu Lissom.

4.4 HEURISTICKÁ DETEKCE POLYMORFNÍCH PACKERŮ

Jak již bylo uvedeno výše, pokud byl spustitelný soubor vytvořen packerem, není bez korektní detekce použitého packeru zpětný překlad možný. Důvodem je skutečnost, že packer modifikuje obsah původního binárního souboru a na jeho vstupní bod umístí kód, který po nahrání programu do paměti zajistí vykonání původního, chráněného kódu. Jednotlivé packery tak od sebe lze rozpoznat na základě obsahu vstupního bodu. To však neplatí pro tzv. *polymorfní packery*, které pro každý soubor generují zcela unikátní obsah vstupního bodu. Tyto packery nelze detekovat pomocí signatur. Zástupcem této rodiny packerů je např. nástroj `Morphine encryptor`.

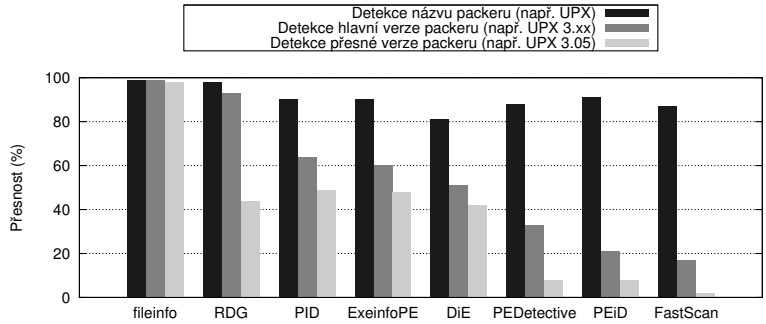
Tyto packery lze detekovat pouze pomocí speciálních heuristik, sestavených na základě ruční analýzy. Analýza sleduje společné vlastnosti všech souborů vytvořených konkrétním nástrojem. Tyto vlastnosti pak tvoří heuristiku, která se v programu implementuje jako sada podmínek. Příklad takové

heuristiky zapsané v jazyce C je na obr. 1. V rámci bakalářské práce bylo implementováno několik takových heuristik.

```
if(file_format == PE && target_architecture == INTEL_X86 &&
   entry_point_offset >= 0x400 && entry_point_offset <= 0x1400 &&
   sections[0].name == ".text" && sections[1].name == ".data" &&
   sections[2].name == ".idata" && sections[2].size == 0x200)
{
   return "Morphine 1.2";
}
```

Obrázek 1: Heuristika pro Morphine encryptor v1.2.

Po analýze a implementaci heuristik byly provedeny experimenty na sadě 5317 binárních souborů ve formátu PE. Experimenty byly zaměřeny na úspěšnou detekci packerů, kterými byly soubory vytvořeny. Do testovací sady byly zahrnuty i soubory vytvořené polymorfními packery. Nástroj `fileinfo` obsadil ve všech kategoriích (podrobnosti viz obr. 2) první pozici.



Obrázek 2: Výsledky experimentů zaměřených na úspěšnou detekci použitého packeru.

5 ZÁVĚR

V tomto článku byl představen analytický nástroj `fileinfo`, důvody pro jeho vznik a požadavky na jeho funkcionalitu. Článek shrnul metody rychlostních optimalizací, díky kterým je aplikace několikanásobně rychlejší, což ve svém důsledku urychluje i celý proces zpětného překladu. Díky rozšíření počtu získávaných informací se nástroj vyrovná ostatním analyzátorům. Heuristiky pro polymorfní packery pak umožňují zpětný překlad pro dosud nepodporované nástroje. Při experimentech zaměřených na detekci packerů dosahoval nástroj `fileinfo` výrazně lepších výsledků než ostatní existující řešení. Nástroj lze rozšiřovat přidáváním podpory pro další formáty spustitelných souborů a také implementací dalších architekturně specifických analýz, například pro architekturu MIPS.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory grantu FIT-S-14-2299 – Výzkum pokročilých metod ICT a jejich aplikace.

REFERENCE

[1] Křoustek, J. a Pokorný, F. Reconstruction of Instruction Idioms in a Retargetable Decompiler. In: *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*. Kraków: IEEE, 2013, s. 1507–1514. ISBN 978-1-4673-4471-5.

[2] Křoustek, J. a Kolář, D. Preprocessing of Binary Executable Files Towards Retargetable Decompilation. In: *8th International Multi-Conference on Computing in the Global Information Technology*. Nice: IARIA, 2013, s. 259–264. ISBN 978-1-61208-283-7.