DETECTION OF USED COMPILER

Michal Kostka

Bachelor Degree Programme (4), FIT BUT

E-mail: xkostk00@stud.fit.vutbr.cz

Supervised by: Jakub Křoustek

E-mail: ikroustek@fit.vutbr.cz

Abstract: The objective of this work is a creation of compiler detector. This program can recognize compiler or packer used for application creation. PE and ELF file formats are supported. This task is solved by searching for compilers´ signature, which represents compiler-specific starting routine in the file. This work is created as a part of the project Lissom.

Keywords: compiler detection, packer identification, reverse engineering, Lissom

1. ÚVOD

Binární spustitelné soubory mohou být různého původu. Existuje celá řada překladačů, které umožňují převod zdrojového kódu na binární soubor. Vytvoření nástroje, který určí překladač použitý k vytvoření daného souboru ze souborů formátu PE/COFF (používaném zejména na Windows) a ELF (především na Linuxu), je cílem této práce.

Kromě překladačů samozřejmě existují další nástroje, které mohou vytvořit spustitelný soubor. Jako příklad lze zmínit *packery*, které vytvoří ze spustitelného souboru jiný spustitelný soubor s redukovanou velikostí, nebo *cryptory* sloužící k zašifrování sekcí spustitelného souboru (jako ochrana proti kopírování nebo proti odhalení detekci *malware*). Kvůli ucelenosti práce je vhodné, aby náš programu uměl detekovat i tyto nástroje.

Konkrétní aplikací znalosti překladače může být (a bude jí zejména v našem případě) zpětný překlad, tedy převod spustitelného souboru na zdrojový kód. Při překladu do mezikódu se totiž ztrácí značná část sémantické informace. Znalost překladače usnadní obnovu těchto informací. Znalost případného packeru nám umožní provést dekompresi souboru před vlastním zpětným překladem. Jiným využitím může být u aplikací, kde je bezpečnost kritická, určení, zda aplikace byla vytvořena bezpečným překladačem.

Tato práce vzniká v rámci projektu Lissom na FIT VUT v Brně.

2. ROZBOR

Do současné doby byla vytvořena celá řada nástrojů, jejichž úkolem byla detekce použitého překladače či jiného nástroje, využitého k vytvoření spustitelného souboru, ale bohužel jen velmi málo je jich volně dostupných na internetu. Dalším problémem je, že tyto nástroje existují většinou pouze pro Windows PE formát. Pro ostatní platformy nejsou dostupné.

Tyto nástroje v drtivé většině využívají to, že překladače umísťují na Entry Point (EP, neboli vstupní bod programu, od kterého se začíná program vykonávat), tzv. startovací rutinu. Startovací rutina je posloupnost instrukcí typických pro překladač, které se provádějí na začátku programu. Pro překladač gcc na Windows mohou tyto instrukce reprezentovat např. bajty `5589E583ECh`, což je posloupnost instrukcí `push ebp; mov ebp, esp; mov esp, imm8`. Těmto bajtům, které reprezentují startovací rutiny, říkáme signatury.

3. VLASTNÍ REALIZACE

Při vytváření jsme se inspirovali zejména nástrojem PEiD. Tento program pracuje ve třech módech – v normálním módu porovnává signatury s bajty na EP souboru, v hloubkovém módu hledá signatury v EP sekci (celé sekci obsahující EP) a v módu hardcore hledá signaturu v celém souboru [1]. Náš program mimoto umožňuje hledat signatury v místě specifikovaném uživatelem a také zejména hledat, které signatuře je obsah EP nejpodobnější, což v některých případech umožňuje detekovat překladač, pro jehož nejnovější verze neexistuje signatura v databázi.

3.1. ULOŽENÍ SIGNATUR

Signatury jsou ukládány v následujícím formátu. Každá signatura se skládá ze tří částí, z nichž první dvě jsou trojciferná hexadecimální čísla a specifikují, v jaké minimální a maximální vzdálenosti od EP má být signatura hledána. Třetí část popisuje obsah bajtů na EP, kdy každý bajt je reprezentován dvěma hexadecimálními číslicemi, tj. každý půlbajt jednou hexadecimální číslicí. Proměnlivý půlbajt je reprezentován speciálním znakem. Další speciální znak zastupuje instrukci nepodmíněného skoku (`EBxx`) a následné tímto skokem přeskočené bajty.

Signatury použité v našem programu jsme získali vlastním výzkumem, z volně dostupných databází signatur a dále od společnosti AVG Technologies.

3.2. NALEZENÍ ENTRY POINTU A ENTRY POINT SEKCE V SOUBORU

V PE souboru lze přímo v PE hlavičce položku adresa EP, která je ovšem virtuální, tedy adresa EP v paměti. Pro naše účely je ovšem nutno znát offset EP v souboru. Tuto adresu je nutno tedy nutno přepočítat podle údajů z tabulky hlaviček sekcí, kde lze nalézt virtuální adresy a offsety jednotlivých sekcí [2]. V ELF souboru je postup obdobný.

V našem programu využíváme pro tyto výpočty funkce knihoven PeLib a ELFIO.

3.3. PROHLEDÁVÁNÍ NA ENTRY POINTU NEBO NA ZADANÉM OFFSETU V SOUBORU

Při prohledávání na Entry Pointu (zadaném offsetu) je obsah signatur porovnáván s bajty na Entry Pointu (zadaném offsetu) převedenými na jejich textovou hexadecimální reprezentaci. Převedené bajty jsou ukládány pro porovnání s ostatními signaturami v databázi.

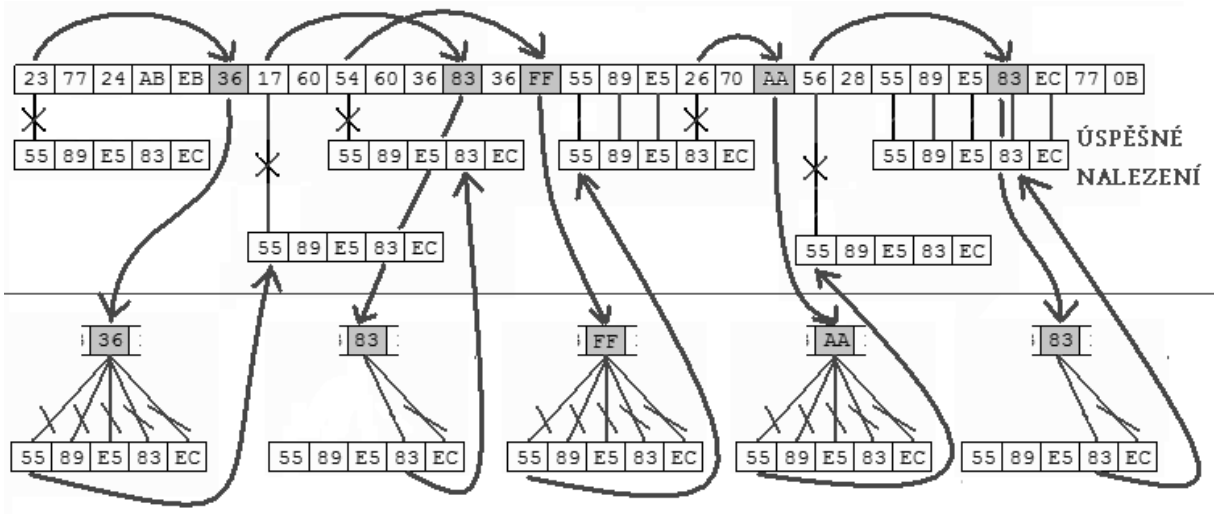
3.4. PROHLEDÁVÁNÍ OBLASTI

Při prohledávání určité oblasti souboru (EP sekce, zadané uživatelem, nebo celého souboru) využíváme pro zjištění jednotlivých signatur jednu z metod pro nalezení podřetězce v řetězci, Quicksearch, která je velmi efektivní (průměrná časová složitost je méně než $O(n)$ a blíží se k $O(n/(m+1))$, kde n je délka řetězce a m délka podřetězce) a velmi jednoduchý [3]. Kvůli proměnlivým bajtům nelze využít knihovní funkce.

Princip algoritmu Quicksearch vidíme na Obrázku 1. Na začátku přiložíme podřetězec k začátku prohledávaného řetězce a poté porovnáváme bajt po bajtu. Ovšem v případě neúspěšného porovnání se podíváme na bajt v řetězci za koncem podřetězce. Pokud se tento bajt vyskytuje v podřetězci, přiložíme podřetězec tak, aby odpovídající poslední znak podřetězce byl na dané pozici a porovnáváme postupně další znaky podřetězce. Pokud ne, přiložíme podřetězec hned za znak.

3.5. VYHLEDÁNÍ PODOBNOSTÍ

Na EP je rovněž možno vyhledávat, jakým signaturám se obsah EP nejvíc podobá. To provádíme tak, že pro každou signaturu vypočteme počet shodných významových půlbajtů (tedy mimo těch proměnlivých) v signatuře a v souboru po půlbajt, který se liší a vrátíme překladač, u kterého byl počet nejvyšší. Tento výsledek rozhodně není přesný, ale může nás nasměrovat, pokud nejsme schopni jiným způsobem překladač zjistit.



Obrázek 1: Demonstrace principu metody Quicksearch na signatuře pro překladač gcc. Spojnice mezi bajty podřetězce a řetězce značí porovnání, přeškrtnutá spojnice neúspěšné porovnání. V části nad čarou porovnáváme zleva doprava a při neúspěchu skáčeme za přiložený řetězec, v části pod čarou porovnáváme zprava doleva a při úspěchu provádíme vhodný posun vzorku.

4. ZÁVĚR

V rámci této práce vytvořený program je schopen detekovat překladač nebo jiný nástroj využitý k vytvoření spustitelného souboru ve formátu PE nebo ELF. Databáze signatur obsahuje více než 1800 signatur různých verzí překladačů, packerů a dalších nástrojů, což je oproti známým volně dostupným detektorům překladačů více než dvojnásobek. Samozřejmě schopnost detekovat tyto nástroje nikdy nebude dokonalá. Problémem je, že signatury různých překladačů mohou být velmi podobné nebo i stejné, dále existují snahy signatury podvrhnout, navíc neustále vznikají nové překladače a ne všechny jsou veřejně známé. Přesto náš nástroj zvládá rozpoznat nejběžnější nástroje. Je také jedním z prvních nástrojů, který se zaměřuje na ELF formát a vyhledává podobnosti. Práci je možno dále rozšiřovat, například zaměřením na další platformy a formáty souborů nebo přidáváním signatur dosud programu neznámých nástrojů.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory grantu TAČR TA01010667 a výzkumného záměru MSM0021630528.

REFERENCE

[1] PEiD. Webová prezentace dostupná na URL http://www.peid.info/ (leden 2011).

[2] Microsoft Portable Executable and Common Object File Format Specification. Microsoft Corporation, 2008. Dokument dostupný na URL http://kishorekumar.net/pecoff_v8.1.htm (leden 2011).

[3] Beneš, M.: Vyhledávání v textu. 2001. Dokument dostupný na URL http://htmltolatex.sourceforge.net/samples/sample3.html (leden 2012).