Reverzní inženýrství

5. Rozpoznávání kompilátorů

Ing. Tomáš Zahradnický, EUR ING, Ph.D.
Ing. Martin Jirkal
Ing. Josef Kokeš

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Verze 24. listopadu 2018

Obsah

Motivace

Otázka

Proč bychom se měli starat o určení, který kompilátor byl použit pro vytvoření daného spustitelného souboru?

Odpověď

Abychom zmenšili množství kódu, které je potřeba analyzovat. Známé výstupy kompilátoru mohou být z další analýzy vyloučeny.

Tutéž otázku můžeme položit i pro všechny knihovny třetích stran, které jsou do programu přilinkovány. Pokud je dokážeme identifikovat, můžeme je z analýzy vyloučit.
Navíc nám znalost funkcí v těchto knihovnách může pomoci pro rozšíření našeho porozumění aplikaci tím, že nám určí datové typy a významy proměnných, které jsou do nich předávány.

Úvod

- Každý kompilátor má svůj vlastní styl, který se promítá do vyprodukovaného kódu — jde např. o rozložení dat a kódu uvnitř binárního souboru.
- Každý spustitelný soubor se linkuje na specifické runtime knihovny, ať už staticky nebo dynamicky. Identifikace těchto runtime knihoven nám prozradí velmi mnoho o použitém překladači.
- Abychom mohli překladač identifikovat, potřebujeme vědět, kde uvnitř zkompilované binárky nalezneme příslušnou informaci.
- Protože binární podoba aplikací je obvykle značně rozsáhlá, znalost toho, co můžeme přeskočit, nám obvykle analýzu velmi urychlí.

Rozpoznávání kompilátorů — Microsoft Visual C++

Marketingová verze	Interní verze	CL.EXE verze	Obvyklé importy DLL	Datum vydání
6	6.0	12.00	msvcrt.dll, msvcp60.dll	Červen 1998
.NET	7.0	13.00	msvcr70.dll, msvcp70.dll	13. 2. 2002
.NET 2003	7.1	13.10	msvcr71.dll, msvcp71.dll	24. 4. 2003
2005	8.0	14.00	msvcr80.dll, msvcp80.dll	7. 11. 2005
2008	9.0	15.00	msvcr90.dll, msvcp90.dll	19. 11. 2007
2010	10.0	16.00	msvcr100.dll, msvcp100.dll	12. 4. 2010
2012	11.0	17.00	msvcr110.dll, msvcp110.dll	12. 9. 2012
2013	12.0	18.00	msvcr120.dll, msvcp120.dll	17. 10. 2013
2015	14.0	19.00	vcruntime140.dll	20. 7. 2015
2017	14.1+	19.10	vcruntime140.dll	7. 3. 2017

- Kompilátor MSVC můžeme rozpoznat prozkoumáním importního adresáře a hledáním knihoven vypsaných v tabulce.
- Symboly dekorované pomocí MSVC obvykle začínají otazníkem.

```
?doSomething@CMFCApplicationView@@QAEXXZ
```

Rozpoznávání kompilátorů — Borland

- Může importovat knihovnu `BORLNDMM.DLL`.
- Dekorovaná jména začínají symbolem `@`.

```
@TModule@ValidWindow$qp14TWindowsObject
```

- Delphi ukládají na začátek kódového segmentu názvy datových typů. Řetězce jako `Boolean`, `Integer`, `TObject`, `Char` atd., pokud je nalezneme na začátku souboru, jsou téměř zárukou, že jsme narazili na program vytvořený v Delphi.

Rozpoznávání kompilátorů — příklad Delphi

```
00400 0410 4000 0307 426F 6F6C 6561 6E01 0000  ..@...Boolean...
00410 0000 0100 0000 0010 4000 0546 616C 7365  ........@..False
00420 0454 7275 658D 4000 2C10 4000 0107 496E  .TrueŤ@.,.@...In
00430 7465 6765 7204 0000 0080 FFFF FF7F 8BC0  teger....€˙˙˙‹Ŕ
00440 4410 4000 0104 4279 7465 0100 0000 00FF  D.@...Byte.....˙
00450 0000 0090 5810 4000 0104 576F 7264 0300  ... X.@...Word..
00460 0000 00FF FF00 0090 6C10 4000 0108 4361  ...˙˙.. l.@...Ca
00470 7264 696E 616C 0500 0000 00FF FFFF FF90  rdinal.....˙˙˙˙
00480 8410 4000 0A06 5374 7269 6E67 9010 4000  „.@...String .@.
00490 0C07 5661 7269 616E 748D 4000 E810 4000  ..VariantŤ@.č.@.
004A0 0000 0000 0000 0000 0000 0000 0000 0000  ................
004B0 0000 0000 0000 0000 0000 0000 E810 4000  ..............č.@.
004C0 0400 0000 0000 0000 AC3A 4000 B83A 4000  ........¬:@.¸:@.
004D0 BC3A 4000 C03A 4000 B43A 4000 1438 4000  Ľ:@.Ŕ:@.´:@..8@.
004E0 3038 4000 6C38 4000 0754 4F62 6A65 6374  08@.l8@..TObject
004F0 F410 4000 0707 544F 626A 6563 74E8 1040  ô.@...TObjectč.@
00500 0000 0000 0000 0006 5379 7374 656D 0000  ........System..
00510 1411 4000 0F0A 4949 6E74 6572 6661 6365  ..@...IInterface
00520 0000 0000 0100 0000 0000 0000 00C0 0000  ..............Ŕ..
00530 0000 0000 4606 5379 7374 656D 0300 FFFF  ....F.System..˙˙
00540 CC83 4424 04F8 E931 5200 0083 4424 04F8  Ě D$.řé1R.. D$.ř
00550 E94F 5200 0083 4424 04F8 E959 5200 00CC  éOR.. D$.řéYR..Ě
00560 4111 4000 4B11 4000 5511 4000 0100 0000  A.@.K.@.U.@.....
```

Rozpoznávání kompilátorů — další

- GCC
  - Při použití Cygwinu je importována knihovna `cygwin1.dll`.
  - Při použití MinGW je importována knihovna `msvcrt.dll`.[1]
  - Dekorovaná jména obvykle začínají na `_Z`.

```
_Z1hv
```

- Watcom
  - Dekorovaná jména obvykle začínají na `W`.

```
W?method$_class$n__v
```

- FORTRAN
  - Může importovat `libifcoremd.dll`, `libifportmd.dll`, `libiomp5md.dll`.

[1] Pozor, i aplikace nezkompilované v MinGW často linkují na `msvcrt.dll`.

Inicializační, běhový a knihovní kód

- Jak bylo vysvětleno v přednášce 2, běh programu začíná v jeho hlavním vstupním bodě. Toto místo je ale poměrně daleko od samotné funkce `main()`.
- Inicializační kód silně závisí na použitém kompilátoru, na zvolených kompilačních nastaveních a na použité běhové knihovně.
- Pro analýzu je **převážně** nezajímavý a jeho podrobné zkoumání je **obvykle** ztrátou času.
- Z tohoto důvodu je vhodné prolog programu označit jako "knihovní kód" a dále se jím nezabývat, protože reverznímu inženýrovi nepřináší mnoho informace.
- Totéž můžeme udělat s knihovnami, které nalezneme přilinkované do programu.

Identifikace knihoven

- Poté, co jsme identifikovali kompilátor, se můžeme pokusit nalézt co nejvíce knihovních funkcí, a ty také vyloučit z analýzy.
- To se obvykle provádí prohledáváním kódu na signatury, které byly předem pro knihovnu vygenerovány. Nalezené funkce pak:
  - přejmenujeme na jméno určené tvůrcem knihovny;
  - označíme jako knihovní kód.
- IDA Pro používá přístup, kterému říká F.L.I.R.T. (Fast Library Identification and Recognition Technology) [1], který dělá přesně to, co bylo popsáno výše.

Signatury knihoven

- Každou knihovní funkci, ke které máme zdrojový kód, můžeme popsat vzorkem kódu.
- Tímto vzorkem by mohlo být prvních X bajtů strojového kódu funkce, ale také celý kód funkce až do závěrečné instrukce `ret`.
  - Je důležité si uvědomit, že pro vytvoření vzorku je nezbytné správně zvolit přesnou verzi knihovny a také typ jejího sestavení (např. Debug nebo Release).
- Hodnotu X můžeme stanovit libovolně, dokonce můžeme vytvořit vzorek z celé funkce. Ani to ale nepomůže v případech, kdy funkce s různým jménem mají identickou implementaci (např. `htonl` a `ntohl`). V takových situacích volíme jeden z názvů a druhý ignorujeme.
- Vzorky vyhledáváme za celou knihovnu a následně ukládáme do společného souboru, kterému říkáme soubor signatur.

- IDA Pro nabízí pro tento účel FLAIR SDK.

Jak rozpoznat knihovnu? I

- Jak můžeme rozpoznat, že spustitelný program používá konkrétní knihovnu?
- Pokud je knihovna linkovaná dynamicky, je to jednoduché:
  - Windows: `dumpbin`, CFF Explorer, Dependency Walker.
  - Linux: `objdump`, `readelf`, nebo prostě `ldd`.
  - OS X: `otool` nebo `dyldinfo`.
- Pokud je knihovna staticky linkovaná, můžeme použít aplikaci `strings` k nalezení:
  - informací o copyrightu,
  - informací o verzi,
  - chybových hlášení.

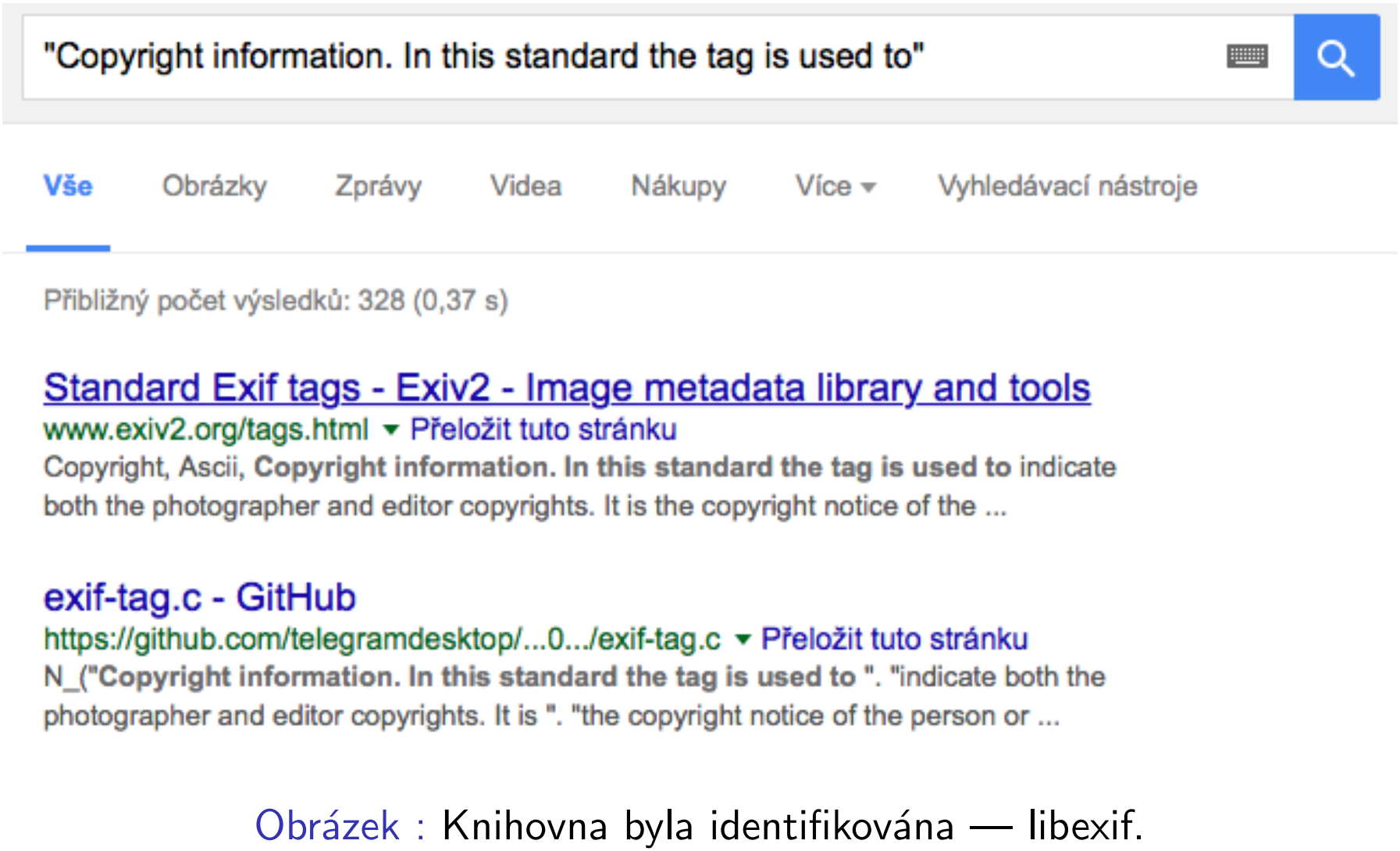
  Následně můžeme nalezené řetězce ouvozovkovat a vložit do některého vyhledávacího enginu!
  - Tímto postupem sice obvykle nedokážeme určit zcela přesnou verzi knihovny, ale obvykle získáme velmi dobrý odhad.

Jak rozpoznat knihovnu? II

```
mail:MacOS admin$ strings -a MediaManager
...
Unknown Exif Version
Exif Version %d.%d
0100
FlashPix Version 1.0
0101
FlashPix Version 1.01
...
Copyright information. In this standard the tag is used to indicate both the...
...
This tag is used to record the name of an audio file related to the image...
...
```

Obrázek : Nástroj `strings` použitý na program za účelem odhalení řetězců použitelných pro identifikaci knihovny

Jak rozpoznat knihovnu? III



Obrázek : Knihovna byla identifikována — libexif.

Literatura


Chris Eagle: *The Ida Pro Book 2nd Edition*, no starch press, 2011.


Eldad Eilam: *Reversing: Secrets of Reverse Engineering*, Wiley Publishing, Inc., 2005.


Reverend Bill Blunden: *The Rootkit Arsenal*, Wordware Publishing, Inc., 2009.