CONTROL FLOW GRAPH

Ota Jirák
Master Degree Programme (2), FIT BUT
E-mail: xjirak03@stud.fit.vutbr.cz

Supervised by: Dušan Kolář
E-mail: kolar@fit.vutbr.cz

ABSTRACT

Control flow graph is one way how to capture a software structure. This structure can be used for some purposes e.g. optimization of generating code, analyzing executable programs. In this article, it is described known algorithm of getting control flow graph generally from any kind of a code. We concentrate on getting control flow graph from executable files, which is a central topic of the paper.

1 ÚVOD

Základní informace o grafech toku řízení jsem získal z přednášek prof. Meduny [1].
Graf toku řízení je diagram, který znázorňuje možné posloupnosti předávání řízení jednotlivým blokům kódu.
Využití grafů toků řízení je především v oblasti překladačů a analýzy programů. Mohou vést k optimalizaci generovaného kódu odstraněním nedostupných bloků, detekci potenciálně nebezpečných konstrukcí, ...

2 ROZBOR

Stavebními kameny takového grafu jsou *základní bloky.* Jedná se o posloupnosti příkazů. Jediným vstupním bodem tohoto bloku je jeho začátek. Jediným výstupním bodem je konec základního bloku. Díky těmto vlastnostem není možný skok doprostřed bloku ani vnořování dalších bloků. Na bloky lze tudíž pohlížet jako na atomické operace.

2.1 ZÍSKÁNÍ GRAFU TOKU ŘÍZENÍ

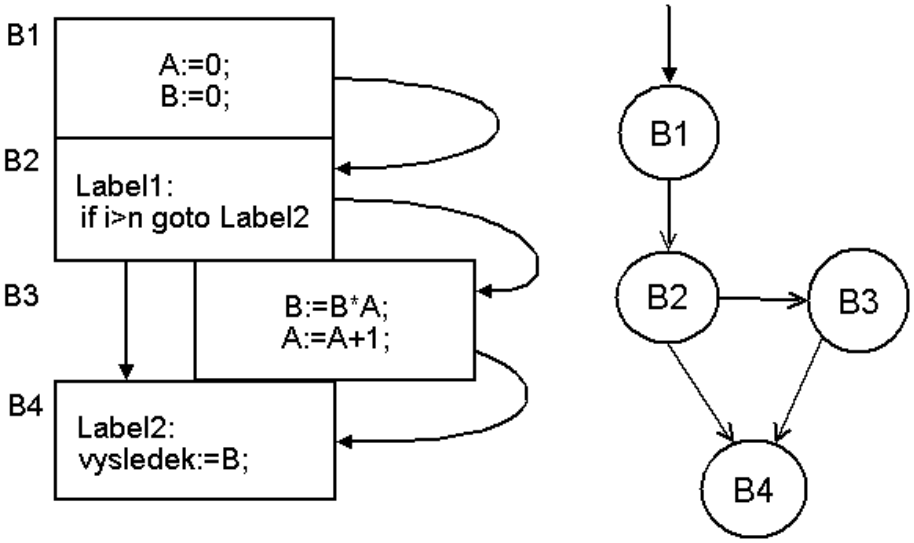
Nejprve si definujeme vedoucí příkazy:

- první příkaz programu je vedoucí,
- každý příkaz, který je návěštím pro příkazy skoku je vedoucím příkazem,
- každý příkaz, který následuje za podmíněným příkazem skoku je vedoucím.

Pro získání grafu si nejprve zajistíme seznam vedoucích příkazů. S jejich pomocí rozdělíme kód na základní bloky. Začátkem každého základního bloku je vedoucí příkaz. Poslední instrukce tohoto bloku je taková, která těsně předchází vedoucímu příkazu.
Graf vznikne popropojováním získaných základních bloků. Toho docílíme přidáním orientovaných hran. Pokud je blok ukončen instrukcí skoku, směřuje hrana na blok, jehož vedoucí příkaz je cílem skoku. Některé bloky nejsou ukončeny instrukcí skoku, protože těsně následující blok vznikl jako cíl nějakého skoku. Orientovaná hrana se tedy přidá i mezi tyto bloky.

2.2 VIZUÁLNÍ VYJÁDŘENÍ GRAFU TOKU ŘÍZENÍ

Zobrazení se do jisté míry podobá konečným automatům. Jednotlivé bloky jsou označeny jménem umístěným v kolečku. Orientovaná hrana je znázorněna šipkou. Šipkou je též označen počátek grafu. Obrázek 1 ilustruje převod kódu programu na graf toku řízení.



Obrázek 1: Převod kódu na graf toku řízení

2.3 ANALÝZA EXE SOUBORU

Druhou oblastí využití grafu toku řízení je analýza programů. Tato oblast je obtížnější, protože nejprve musíme z EXE souboru získat výpis programu v jazyce symbolických instrukcí. Toho docílíme pomocí disasembleru. Mezi volně dostupné patří např. OllyDbg [2].
Pro možnosti automatické analýzy může být výhodnější vytvořit disasembler vlastní. Použitím cizích programů jsme nuceni analyzovat jejich výstupy. Ztrácíme tím také možnost využívat informace získávané při běhu vlastního disasembleru. Specifikace formátu EXE souboru se nachází na webových stránkách firmy Microsoft [3]. Informace o kódování instrukcí jsou dostupné na stránkách firmy Intel [4].
Získávání grafu z EXE souboru se odehrává již popsaným způsobem. Instrukcemi skoku jsou JMP a CALL. Instrukce RET způsobí ukončení základního bloku. Orientovaná hrana pak ukazuje na základní blok začínající za instrukcí CALL, která funkci vyvolala. Základní blok začínající vstupním bodem programu je označen jako počátek grafu.

2.4 PROBLÉMY SPOJENÉ S ANALÝZOU EXE SOUBORU

Analýza EXE souborů je spojena s řadou překážek. Mezi nejzávažnější patří samopřepisovatelné kódy. Při zkoumání takového programu statickou analýzou dosáhneme maximálně detekce rozbalující smyčky, která přepis programu zajišťuje. Řešením by mohla být dynamická analýza, která má také svoje úskalí. Simulace procesoru je velmi obtížná a ostatní techniky neumožňují automatické zpracování pro účely další analýzy.
Neméně vážnou komplikací při analýze instrukcí asembleru jsou samotné možnosti asembleru. Jako parametry instrukcí JMP a CALL lze kromě přímé hodnoty adresy skoku použít také ukazatele do paměti nebo hodnotu uloženou v registru. Obsah paměti nebo registru lze modifikovat různými instrukcemi. Zjistit cíl skoku by opět vedlo k simulaci.
Překladače velmi často optimalizují opakovaná volání jedné funkce uložením adresy cíle v registru. Např. CALL ESP zavolá funkci, jejíž adresa je v registru ESP. Nahrání adresy funkce bývá pomocí instrukce MOV. Získání cíle takového volání znamená detekovat místo zápisu do registru a zajištění, že tento registr nebude přepsán.
Jsme schopni detekovat konstrukci typu:

```
MOV ESP, 410000
CALL ESP
...
CALL ESP.
```

Detekování změny hodnoty registru znamená určit, které parametry jsou cílem jednotlivých instrukcí a určit, zda se tam neobjeví registr použitý při volání funkce či skoku.
Ani tento postup není schopen určit novou hodnotu cíle, pokud byla hodnota přepsána. Toho je schopna pouze simulace běhu programu.
Pokud je volání pomocí registru v jiném bloku, než došlo k zápisu cílové hodnoty, hrozí nebezpečí různých cílů při editaci cílové hodnoty v předcházejících blocích.

3 VÝSLEDKY A ZÁVĚR

Analýzou zdrojových kódů, případně programů, jsme do jisté míry schopni získat graf toku řízení. Linearizací tohoto grafu získáme řetězec charakterizující analyzovaný kód. Pokud tento řetězec koresponduje s řetězcem škodlivého kódu, detekujeme testovaný kód jako škodlivý.
Pomocí grafu toku řízení jsme schopni získat přesnější výpis strojových instrukcí. Začneme soubor zpracovávat od vstupního bodu. Při větvení grafu si zapamatujeme reference začátků ostatních větví. Po dokončení zpracovávané větve pokračujeme s některou uloženou referencí. Můžeme se tak vyhnout chybnému dekódování a zvýšit šance na odhalení samopřepisovatelných kódů a jiných potenciálně nebezpečných konstrukcí.

REFERENCE

[1] http://www.fit.vutbr.cz/study/courses/VYP/

[2] http://www.ollydbg.de/

[3] http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

[4] http://www.intel.com/products/processor/manuals/index.htm