Платформенно-независимый и масштабируемый инструмент поиска клонов кода в бинарных файлах программ

Институт системного программирования РАН

А.К. Асланян : hayk@ispras.ru
Ш.Ф. Курмангалеев : kursh@ispras.ru>
В.Г. Варданян : vaag@ispras.ru
М.С. Арутюнян : arutunian@ispras.ru
С.С. Саргсян : sevaksargsyan@ispras.ru

Типы клонов для бинарного кода

1. Фрагменты кода, которые полностью совпадают
2. Фрагменты кода, которые могут отличаться типами, значениями данных, именами регистров
3. Фрагменты кода, которые могут отличаться типами, значениями данных именами регистров, а также могут отличаться некоторыми инструкциями (в конкретном фрагменте могут присутствовать или отсутствовать некоторые инструкции)

Примеры клонов бинарного кода (соответствующий ассемблер)

Фрагмент кода	Клон типа 1	Клон типа 2	Клон типа 3

public main
main proc near
 	
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
 	
push ebp
mov ebp, esp
mov [ebp+var_4], 5
mov eax,[ebp+var_4]
imul eax,[ebp+var_4]
leave
retn
main endp
	public main
	main proc near
	
	var_4= dword ptr -4
	argc= dword ptr 8
	argv= dword ptr 0Ch
	envp= dword ptr 10h
	
	push ebp
	mov ebp, esp
	mov [ebp+var_4], 5
	mov eax,[ebp+var_4]
	imul eax,[ebp+var_4]
	leave
	retn
	main endp
		public main
		main proc near
		
		var_4= dword ptr -4
		argc= dword ptr 8
		argv= dword ptr 0Ch
		envp= dword ptr 10h
		
		push ebp
		mov ebp, esp
		mov [ebp+var_4], 10
		mov ecx, [ebp+var_4]
		imul ecx, [ebp+var_4]
		leave
		retn
		main endp
			public main
			main proc near
			
			var_1= dword ptr -4
			argc= dword ptr 8
			argv= dword ptr 0Ch
			envp= dword ptr 10h
			
			push ebp
			mov ebp, esp
			mov [ebp+var_1], 15
			mov ecx, [ebp+var_1]
			
			leave
			retn
			main endp

Возможное применение технологии

Поиск:

1. Вредоносного кода
2. Семантических ошибок в коде
3. Нарушений условий лицензионных соглашений на использование кода библиотек

Существующие методы

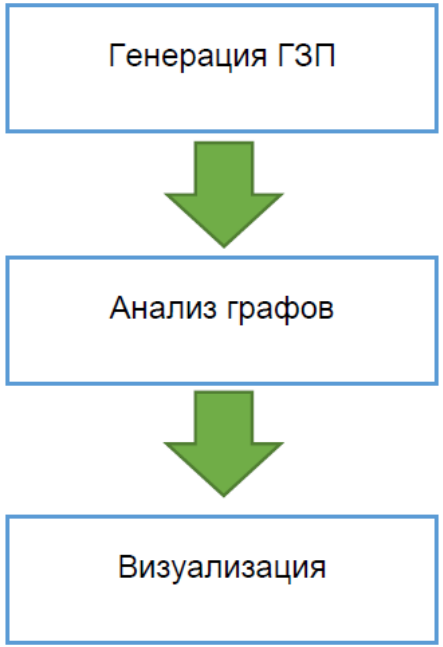
1. J. Jang and D. Brumley. Bitshred: Fast, scalable code reuse detection in binary code (cmu-cylab-10-006). CyLab, page 28, 2009; (1 тип)
2. A. Schulman. Finding binary clones with opstrings function digests: Part III. Dr. Dobb's Journal, 30(9):64, 2005; (1 тип)
3. D. Bruschi, L. Martignoni, and M. Monga. Code normalization for self-mutating malware. IEEE Security & Privacy, 5(2):46–54, 2007. (1,2 типы)
4. Thomas Dullien, Ero Carrera, Soeren-Meyer Eppler, Sebastian Porst «Automated attacker correlation for malicious code» // Technical report, DTIC Document, 2010 (1, 2 типы)
5. A. Sæbjørnsen, J. Willcock, T. Panas, D. Quinlan, and Z. Su. Detecting code clones in binary executables. In Proceedings of the 18th International Symposium on Software Testing and Analysis, pages 117–128. ACM, 2009. (1, 2, 3 типы)

Постановка задачи

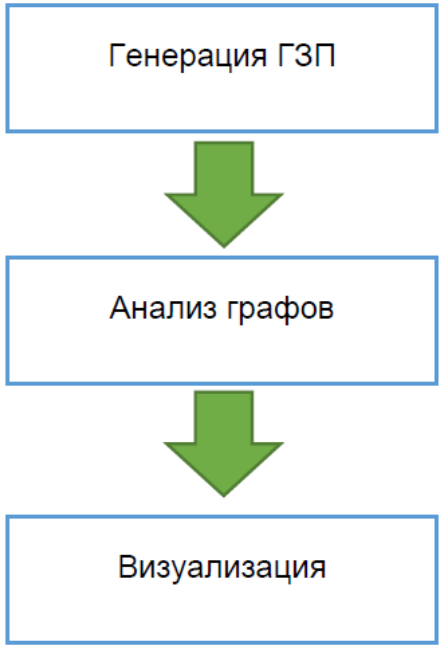
Разработать и реализовать инструмент поиска клонов в бинарном коде программ удовлетворяющий следующим требованиям:

- Поиск 1, 2, 3 типов клонов
- Процент истинных срабатываний (>90%)
- Независимость от целевой архитектуры (x86, x86-64, ARM, MIPD, PPC)
- Масштабируемый (размер бинарных файлов может достигать сотен МБ).

Архитектура инструмента



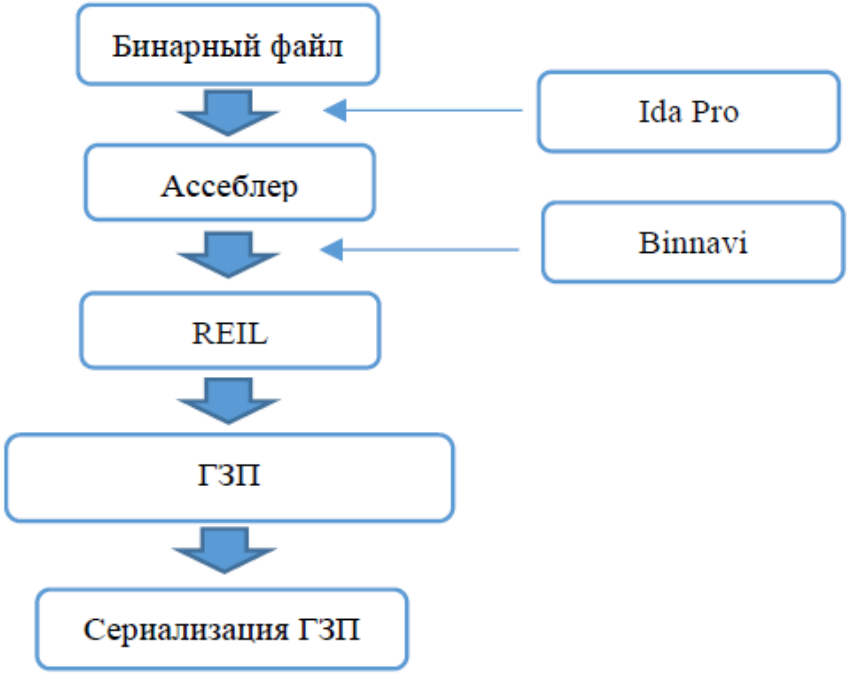
Архитектура инструмента



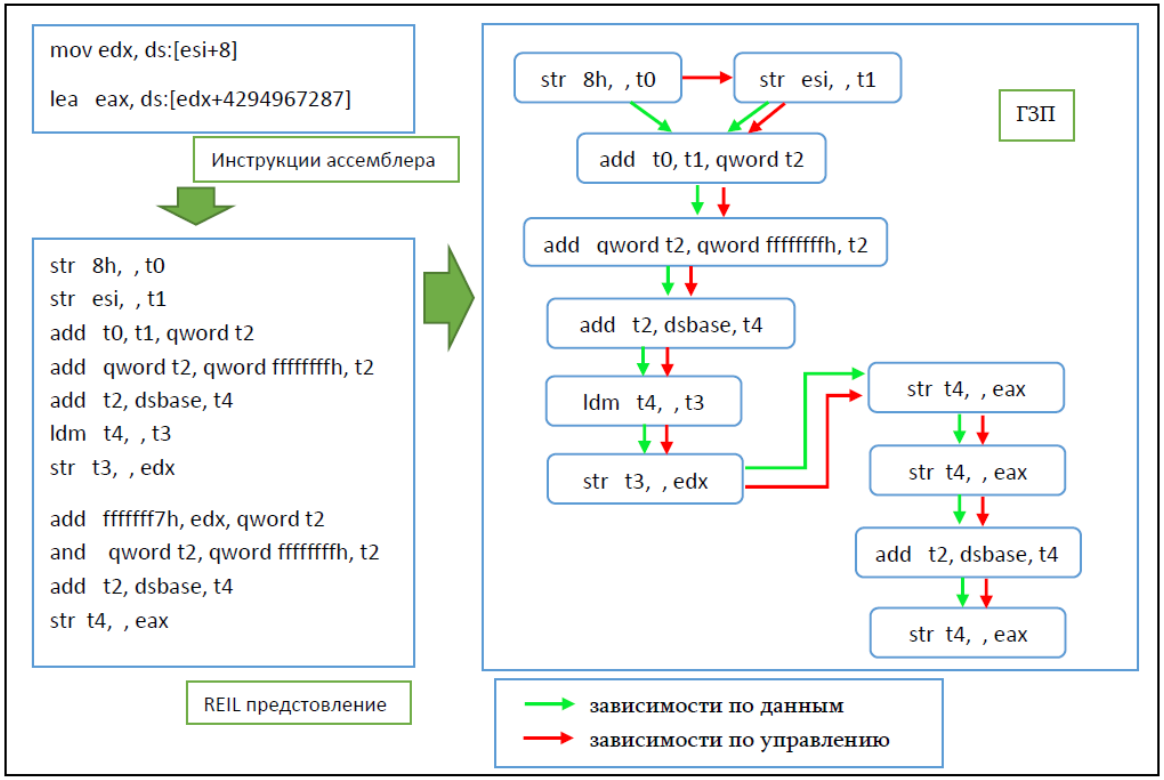
Технология обобщается для поиска клонов в исходном коде языков C/C++, JavaScript, Java.

1. С.С.Саргсян, диссертация, «Методы поиска клонов кода и семантических ошибок на основе семантического анализа программы»
2. Sargsyan S., Kurmangaleev S., Baloian A., Aslanyan H. Scalable and Accurate Clones Detection Based on Metrics for Dependence Graph // Mathematical Problems of Computer Science. 2014. Т. 42. С. 54-62
3. Саргсян С., Курмангалеев Ш., Белеванцев А., Асланян А., Балоян А. Масштабируемый инструмент поиска клонов кода на основе семантического анализа программ // Труды Института системного программирования РАН. 2015 Т. 27. № 1. С. 39-50.
4. Sargsyan S., Kurmangaleev S., Vardanyan V., Zakaryan V. Code Clones Detection Based on Semantic Analysis for JavaScript Language // 10th International Conference on Computer Science and Information Technologies. 2015. С. 182-185.

Генерация ГЗП (1-ый этап)



Пример ГЗП


mov edx, ds:[esi+8]
lea eax, ds:[edx+4294967287]
Инструкции ассемблера
str 8h, , t0
str esi, , t1
add t0, t1, qword t2
add qword t2, qword ffffffffh, t2
add t2, dsbase, t4
ldm t4, , t3
str t3, , edx
add ffffffff7h, edx, qword t2
and qword t2, qword ffffffffh, t2
add t2, dsbase, t4
str t4, , eax
REIL предстовление
ГЗП
зависимости по данным
зависимости по управлению

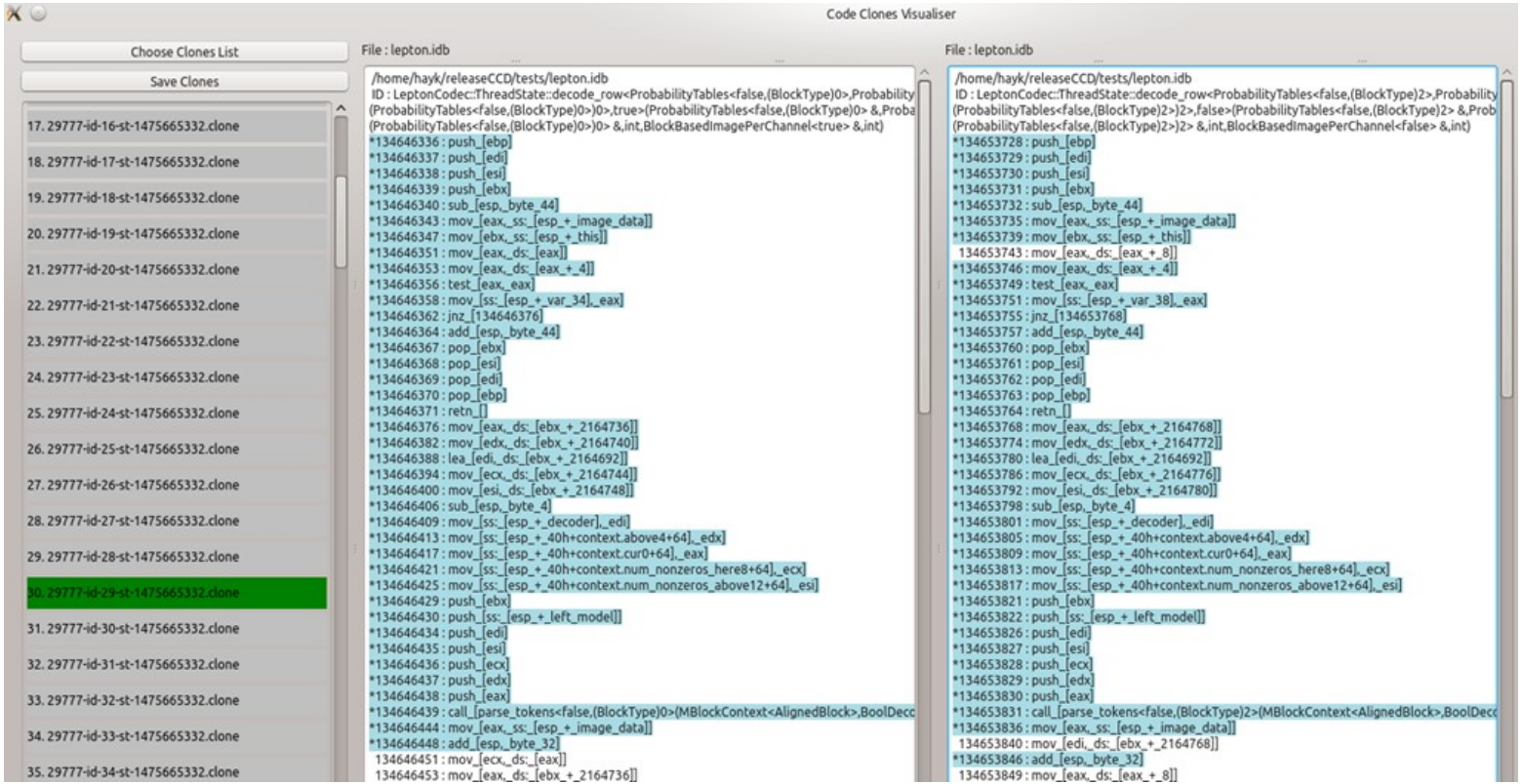
Анализ ГЗП (2-ой этап)

1. Загрузка ГЗП
2. Разделение ГЗП на подграфы (единицы сравнения)
3. Поиск наибольшего общего подграфа двух ЕС:

- Фильтрация пар по параметру МЧИ
- Сопоставление вершин и проверка удовлетворению МПС

4. Фильтрация результатов

Визуализация результатов (3-ий этап)



Результаты

Бинарный файл	Размер бинарного файла	Количество найденных клонов	Время работы инструмента
libxml2.so	1.8 МБ	2123	1м.46 с.
libfreetype.so	816 КБ	270	57 с.
openssl	764 КБ	40	35 с.
d8	33 МБ	40397	26 м.52 с.

Бинарные файлы получены компиляцией исходного кода с флагами -O0 -fno-inline, МЧИ равно 30, МПС равно 90%

Результаты

Бинарный файл	Размер бинарного файла	Количество найденных клонов	Время работы инструмента
libxml2.so	1.8 МБ	437	1м.25 с.
libfreetype.so	740 КБ	73	36 с.
openssl	672 КБ	71	26 с.
d8	20 МБ	19453	17 м.26 с.

Бинарные файлы получены компиляцией исходного кода с флагами -O3 -fno-inline, МЧИ равно 30, МПС равно 90%

Спасибо за внимание!