УДК 519.876.2:004.4

А. В. Бєгун, к.е.н., доцент,
ДВНЗ «КНЕУ імені Вадима Гетьмана»

ОСОБЛИВОСТІ ПЕРЕВІРКИ ВЛАСТИВОСТЕЙ БЕЗПЕКИ ПРОГРАМ МЕТОДОМ СТАТИЧНОГО АНАЛІЗУ

АНОТАЦІЯ. Стаття присвячена аналізу розвитку засобів безпеки програмних систем. Основну увагу приділено перевірки програмних продуктів методом статичного аналізу, який дозволяє розпізнавати властивості обчислень програм без проведення тестових експериментів.

КЛЮЧОВІ СЛОВА: властивість програми, безпека програм, методи аналізу, статичний аналіз програм, решітка, потоки даних.

АННОТАЦИЯ. Статья посвящена анализу развития средств безопасности программных систем. Основное внимание уделяется проверке программных продуктов методом статического анализа, который позволяет распознавать свойства вычислений программ без проведения тестовых экспериментов.

КЛЮЧЕВЫЕ СЛОВА: свойство программы, безопасность программ, методы анализа, статический анализ программ, решетка, потоки данных.

ABSTRACT. The article is sanctified to the market of development of facilities of informative safety of activity of the economic systems analysis. Basic attention is sanctified to safety of cloudy technologies for corporate clients. This process must exist in the conditions of the balanced action: safety is charges.

KEYWORDS. Property of the programs, program's safety, methods of analysis, static analysis of programs, grille, data streams.

Вступ. Задача перевірки властивостей безпеки програмного забезпечення відноситься до числа найактуальніших задач сучасного програмування. Така задача виникає дуже часто при виявленні не документованих сценаріїв роботи системи. І для її розв'язання часто доводиться вручну аналізувати початковий код. Пояснюється це перш за все тим, що задача виявлення багатьох властивостей програмних систем належить до складу алгоритмічно нерозв'язуємих проблем. Так, наприклад, ні єдина нетривіальна властивість обчислень програм не припускає надійної автоматичної перевірки. Але існують методи, які дозволяють отримати попередні висновки про якість і характер обчислень програм, а в деяких випадках такі висновки можуть бути точними й остаточними.

Виявлення багатьох властивостей програмних систем може здійснюватися за допомогою статичного методу аналізу потоків даних. Тут, під статичним аналізом потоків даних розуміється такий метод оцінки поведінки програми, при якому знання про властивість можуть бути отримані на основі встановлення визначених залежностях між окремими компонентами програми (операторами, змінними, об'єктами тощо) без залучення результатів тестових експериментів. Автор [1] стверджує, що в більшості випадків отримання точного рішення задачі статичного аналізу є алгоритмічно нерозв'язною проблемою. Але для практичних цілей частіше всього можна обмежитися отриманням деяких оцінок цього точного рішення і це призведе до необхідності розробки ефективних апроксимуючих алгоритмів статичного аналізу програм. Так, наприклад, не існує алгоритму, який дозволяв би отримати точну відповідь на запит про ініціалізації заданої змінної в заданій точці програми. Але при цьому можна побудувати квазіалго-

ритм, який, у випадку отримання позитивної відповіді на запит, дозволить гарантувати те, що задана змінна в заданій точці програми при будь-якому обчисленні буде мати значення, яке їй призначене.

Виклад основного матеріалу. Для отримання рішення задачі статичного аналізу потоків даних, які описують ці потоки в кожній точці програми, пропонується побудувати систему рівнянь і таке рішення сформованої системи. Безумовно, ефективність алгоритмів розв'язання створеної системи рівнянь визначається як видом самих рівнянь, так і способом їх представлення.

Система рівнянь потоків даних, що характеризує властивість поведінки програми, представляється у вигляді решітки абстрактних даних L кінцевої висоти. Елементи цієї решітки слід розглядати як знання про властивість у рамках введеної абстракції

$$P_i = f_i(g(P_{j_1}, P_{j_2}, ..., P_{j_n})), \tag{1}$$

де $P_m \in L$ значення властивості, яка аналізується, після виконання m-го оператора,

$j_1, j_2, ..., j_n$ – номери усіх операторів, які передують управлінню i-му оператору,

g: $L^n \to L$ операція обчислення точної верхньої або нижньої грані на решітці L,

$f_i : L \to L$ функція, яка має властивість дистрибутивності:

$$f_i\ (g(P_{j_1}, P_{j_2}, ..., P_{j_n})) = \mathrm{g}(f_i(P_{j_1}), f_i(P_{j_2}), \ldots, f_i(P_{j_n})), \tag{2}$$
$$j_1, j_2, ..., j_n \in \{1,2, \ldots, k\},\ i = 1,2,\ldots, k;$$

функція будується для кожного оператора програми з урахуванням його семантики,

k — кількість операторів у програмі.

Часто в кожній точці програми аналізується значення кількох властивостей одночасно. Тому P_m часто визначають як вектор властивостей $\mathrm{P_m} = (P_m^1, P_m^2, ..., P_m^t)$, де $P_m^i \in L$, $t \geq 0$ — кількість властивостей, які аналізуються. У цьому випадку функції g і f мають вигляд

$$g: L^{n \times t} \to L^t,\ f: L^t \to L^t.$$

Тоді, стосовно [2], система рівнянь (1) завжди має роз'вязок. При цьому найменший розв'язок системи можна обчислити за допомогою ітераційного алгоритму за кінцеве число кроків.

Пропонується метод [1], де система представляється у вигляді орієнтованого графа потоку управління програми, в якому вершини є оператори програми, а дуги відповідають можливості безпосередньої передачі управління від одного оператора до іншого. Стосовно використання методу будемо виходити з таких пропозицій:

— кількість властивостей, що аналізується, обмежена;

— рівняння $Pi = fi(Pj)$, де $i, j \in \{1, \ldots, k\}$, представимо у вигляді

$$P_i^s = h(f_i^1(P_j^1), f_i^2(P_j^2), \ldots, f_i^t(P_j^t)),$$

де $s = 1, 2, \ldots, t, f: L^t \to L, q \in \{1, 2, \ldots, t\}$, $h: L^t \to L$ — загальна для всього алгоритму функція;

— рівняння $P_i = g(P_1, P_2, \ldots, P_n)$ представимо у вигляді

$$P_i^s = g(P_1^s, P_2^s, \ldots, P_n^s),$$

де $s = 1, 2, \ldots, t$.

Усі три пропозиції майже завжди використовуються на практиці.

Використовуючи (2), зведемо кожне рівняння системи (1) до вигляду

$$P_i = g(f_i(P_{j_1}), f_i(P_{j_2}), \ldots, f_i(P_{j_n})).$$

Тоді система рівнянь статистичного аналізу потоків даних може бути представлена як

$$\begin{aligned} P_i^s = g^s(h(f_i^1(P_{j_1}^1), f_i^2(P_{j_1}^2), \ldots, f_i^t(P_{j_1}^t)), \ldots, \\ h(f_i^1(P_{j_n}^1), f_i^2(P_{j_n}^2), \ldots, f_i^t(P_{j_n}^t))), \end{aligned} \quad (3)$$

де $s = 1, 2, \ldots, t$.

Тут функції g^s і h не залежать від окремих операторів, які містяться в програмі, і визначаються тільки властивістю, що аналізується, та вибором абстрактних даних (решітка L).

Розглянемо теоретико-графовий спосіб представлення функцій f_i^s у вигляді повних дводольних розмічених орієнтованих графів спеціального виду. Кожна доля такого графу містить s ве-

ршин; ці вершини упорядковані, а *i*-та вершина кожної долі помічена елементом a_i, де $a_1,a_2,\ldots,a_n$ — усі властивості, які аналізуються. Кожній дузі $a_p \to a_r$ приписана формула f_i^{pr}. Тоді система (3) може бути приведена до канонічного вигляду

$$P_i^s = g^s(h(f_i^{1j_1}(P_{j_1}^1), f_i^{2j_1}(P_{j_1}^2),\ldots, f_i^{tj_1}(P_{j_1}^t)) \\ ,\ldots, h(f_i^{1j_n}(P_{j_n}^1), f_i^{2j_n}(P_{j_n}^2),\ldots, f_i^{tj_n}(P_{j_n}^t))). \quad (4)$$

Побудовані таким чином дводольні графи підставляються в граф потоку управління таким кроком: кожна дуга в графі потоку управління заміщується дводольним графом, що відповідає цій дузі.

Для розв'язання системи (4) використовуються два етапи:

1) будується дводольний граф, який відображає функції зміни властивостей для кожної процедури та оператора, що міститься в цій процедурі;

2) на основі отриманих функцій обчислюється значення властивостей для кожного оператора, які залежать від значень властивостей на вході програми.

Якщо кожна функція f_i^{pr} може бути представлена структурою даних, розмір якої обмежений деякою константою, а час виконання кожної операції над цим функціями також обмежений деякою константою, то у випадку виконання умов трьох пропозицій алгоритм статичного аналізу потоків *даних є коректним. Час виконання методу не перевищує величини* $O(ED^3)$, де E — кількість дуг у графі потоку управління, D — кількість властивостей аналізу для кожного оператора [2].

Реалізація і апробація ряду прикладів дозволило сформулювати і розв'язати такі задачі:

— аналіз властивості ініціалізованості змінної;
— аналіз властивості змінної мати постійне значення;
— аналіз властивості змінної мати постійний знак;
— перевірка використання змінних;
— аналіз глобальних змінних.

Перші три задачі являються класичними задачами статичного аналізу потоків даних, дві останніх можна розв'язати іншими шляхами. Але залучення запропонованого апарату для комплексу задач прискорить створення аналізуючої програми та залишить продуктивність на привабливому рівні.

Таблиця 1

РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ З ПРОГРАМНИМ ПРОДУКТОМ

Властивості	Програма PROG.CPP	Програма MATRIX.CPP	Програма COMPR.CPP
Кількість рядків	27	312	1280
Кількість процедур	4	19	41
Час роботи «ініціалізованість змінних» (хв:с:мс)	382	307	8:564
Час роботи «константність значень змінних» (хв:с:мс)	385	306	9:088
Час роботи «від'ємність змінних» (хв:с:мс)	413	261	5:134
Час роботи «зміна та використання глобальних змінних» (хв:с:мс)	335	1:674	8:51:317
Час роботи «траси використання змінних» (хв:с:мс)	32	323	5:382

Аналізатор властивостей програм реалізований мовою програмування С++, в якому ядро представлено у вигляді (табл. 1) набору класів з абстрактною структурою, а визначення нащадків такої структури призводить до конкретизації задачі

Висновки. Сучасний ринок засобів безпеки програмного забезпечення потребує нових, надійніших методів застосування, особливо в аналізі властивостей конкретних програм. Тут, безумовно, корисним було залучення методу аналізу потоків даних, який був реалізований у вигляді універсальної програми статичного аналізу.

При реалізації запропонованого алгоритму виникли задачі вибору компактного і ефективного способу представлення функцій канонічної системи рівнянь, розв'язання яких надасть можливість використовувати результати в якості ядра перспективної системи перевірки властивостей безпеки програмного продукту. Такий підхід, забезпечений відомими методами захисту, створює умови для повноцінної реалізації комплексної безпеки програмних систем.

Література

1. *Neilson F., Neilson H., Hankin C.* // Principles of Program Analysis. Berlin — Heidelberg: Springer-Verlag, 1999. — 450 p.

2. *Иванов К. С., Захаров В. А.* О противодействии некоторым алгоритмам статического анализа програм / Математика и безопасность информационных технологий. Материалы конференции в МГУ 23—24 октября 2003 г. — М.: МЦНМО, 2004. — 426 с.

3. *Черноножкин С. К.* Меры сложности программ (обзор) // В сб.: Системная информатика. — Вып. 5. — 1997. — С. 188—228.

Стаття надійшла до редакції 24.04.2013 р.