

СПОСІБ ЗАХИСТУ АДРЕС ПЕРЕХОДІВ

Для роботи на майбутніх чіп-картках мають бути допущені прикладні програми третіх фірм, які можуть викликати функції операційної системи. При цьому може існувати небезпека, що ці прикладні програми третіх фірм містять спроби маніпулювання ~~своєю~~ шкідництва стосовно інших частин програм. Таке втручання могло б бути здійснене не шляхом використання дійсного посилання функції програми операційної системи із прикладної програми, а шляхом використання іншої адреси переходу. Внаслідок цього код операційної системи міг би бути виконаним не правильно і, наприклад, могла б бути спровокована втрата інформації шляхом помилкового переписування вмісту області пам'яті.

Згідно з рівнем техніки планують перехід на попередньо задані адреси з постійними інтервалами. При цьому спочатку слід перевірити, чи лежить попередньо задана адреса у діапазоні адрес, дозволених для модуля. В разі позитивного результату здійснюється перехід на попередньо задану адресу, а звідти виконується подальший перехід на власне функцію. Недоліком такого способу є те, що при цьому має бути здійснений вирахуваний

перехід, який веде до пропуску у потоці інструкцій процесора. Альтернативно перехід на функцію може бути здійснений із затримкою після кількох програмних інструкцій. Це створює проблеми оптимізації для компілятора, якщо на початку функції введено цикл і постійно має здійснюватися перехід туди і назад між кодом у попередньо заданій адресі і завантаженим кодом.

Тому задачею винаходу є розробка способу захисту адрес переходу вказаного вище типу, згідно з яким відпадає необхідність у здійсненні вирахованого переходу і, таким чином, не виникають пропуски у потоці інструкцій процесора.

Згідно з винаходом ця задача вирішена завдяки тому, що допустимі адреси переходів можуть бути використані безпосередньо і можуть бути розпізнані шляхом встановлення кореляції між даними, які не можуть перебувати всередині однієї і тієї ж інструкції.

При цьому компілятор або компоновщик шляхом організації програмного коду може забезпечити, що лише легальні адреси переходу виконують умови цієї кореляції. Наприклад, кореляція може бути встановлена завдяки тому, що комірка пам'яті безпосередньо до або після адреси переходу містить адресу корельованих даних.

Переважаюча можливість полягає в тому, що комірка пам'яті безпосередньо до або після адреси переходу містить посилання на відповідний запис у захищеному списку легальних адрес переходу.

Особливо доцільною є автоматична перевірка наявності кореляції між даними при виконанні виклику функції.

Наступною доцільною ознакою є додаткова автоматична перевірка розміщення корельованих даних у попередньо заданій, зарезервованій області пам'яті, при виконанні виклику функції.

Поки програмні інструкції не перевищують певного максимального числа n байт, може бути застосоване ще одне рішення згідно з винаходом, а саме використання спеціального коду пустої команди, який служить для уникнення випадкової кореляції і може бути додатково введений компілятором або компоновщиком.

При цьому особливо доцільним є встановлення кореляції між кодовими даними, віддаленими щонайменше на n байт.

Крім того, згідно з винаходом адреса переходу може бути захищена шляхом введення спеціальної послідовності байтів, яка не може виникнути всередині регулярного коду, наприклад, пустого коду.

Таким чином, згідно з винаходом можна уникнути пропуску у потоці інструкцій процесора шляхом безпосереднього переходу на адресу покажчика функції. Правда, в такому разі слід потурбуватися, щоб легальні адреси переходів відрізнялися нелокальною кореляцією кодів. Компілятор чи компоновщик мусить шляхом організації програмного коду забезпечити, щоб таку кореляцію мали лише легальні адреси переходів. При цьому "нелокальна кореляція" означає кореляцію між даними, розміщеними не в одній і тій же окремій інструкції.

Таким чином, можливі такі переважні форми виконання винаходу:

1. Кореляція з даними у зарезервованих для цього областях пам'яті: В разі простої реалізації в комірці пам'яті може міститися, наприклад, безпосередньо перед адресою переходу адреса корельованих даних, які, наприклад, відповідають легальній адресі переходу функції. При виконанні виклику функції може автоматично здійснюватися перевірка виконання умови цієї кореляції і/або перевірка розміщення корельованих даних у спеціально передбаченій, зарезервованій області пам'яті. Механізм на перший погляд дуже схожий на використовуваний досі механізм попередньо заданої адреси переходу, але його перевагою є те, що не використовується вирахований перехід і інструкції функції можуть утримуватися безпосередньо в модулі вибірки з упередженням (префетчері) потоку даних. Пропуск виникає лише у помилковому випадку нелегального переходу.
2. Кореляція з програмними даними у незарезервованих областях пам'яті. Передумовою цього рішення є те, що програмні інструкції не перевищують певну максимальну кількість n байт. Тоді області даних більшої довжини у кодовому сегменті мають бути вилучені. Крім того, передумовою цього методу є використання спеціального пустого коду, який не використовується у нормальному коді і додатково

вводиться компілятором/компоновщиком лише для уникнення випадкової кореляції.

В межах цієї форми виконання винаходу можна вирізнити ще два різні типи рішень:

- а) Встановлюють кореляцію між даними, віддаленими щонайменше на n байт.

При цьому компілятор чи компоновщик має забезпечити уникнення можливої випадкової кореляції у коді шляхом введення проміжного пустого коду.

Можлива реалізація виглядить таким чином:

Безпосередньо перед адресою переходу функції стоїть значення, що є функцією наступних $n+m$ ($m \geq 0$) байт. Якщо ця кореляція випадково виникне де-небудь у коді, то компілятор чи компоновщик має усунути цю випадкову кореляцію. Оскільки в ході $n+m$ байт згідно з передумовою закінчується щонайменше одна істинна інструкція, після їх закінчення може бути введена серія пустих кодів, доки значення функції не зміниться. При цьому функція може бути довільно вибрана в певних межах.

- б) Адреса переходу захищається шляхом введення спеціальної послідовності байт, яка не може виникнути всередині регулярного коду. Прикладом може бути серія пустих кодів.

Таким чином, згідно з винаходом адреси переходів захищаються шляхом нелокальної кореляції кодів, яка може виникнути лише в адресах переходів.

Завдяки цьому можна відмовитися від механізму із попередньо заданими адресами переходів, при якому використовують вирахований перехід, що спричинює пропуск у потоці інструкцій.

Згідно з винаходом за адресою переходу здійснюється безпосередній перехід у функцію. Наступні інструкції можуть бути завантажені в потік незалежно від позитивної чи негативної верифікації адреси переходу. Завдяки цьому, підвищується ефективність тестованих викликів функцій.

За довіреністю

Патентний повірений



Пахаренко О.В.