



УКРАЇНА

(19) UA (11) 26708 (13) U
(51) МПК (2006)
G06F 21/00

МІНІСТЕРСТВО ОСВІТИ
І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ДЕПАРТАМЕНТ
ІНТЕЛЕКТУАЛЬНОЇ
ВЛАСНОСТІ

ОПИС ДО ПАТЕНТУ НА КОРИСНУ МОДЕЛЬ

видається під
відповідальність
власника
патенту

(54) СПОСІБ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1

(21) u200701813

(22) 21.02.2007

(24) 10.10.2007

(72) ВЕРВЕЙКО ВІТАЛІЙ МИКОЛАЙОВИЧ, UA

(73) ВЕРВЕЙКО ВІТАЛІЙ МИКОЛАЙОВИЧ, UA

(56)

(57)

Спосіб захисту програмного забезпечення, який включає використання зовнішнього апаратного засобу, який **відрізняється** тим, що попередньо у зовнішній апаратний засіб вбудовують захищену віртуальну машину, а код програми, яку необхідно захистити, за допомогою ЕОМ перетворюють шляхом криптографічного зашифрування важливих ділянок коду програми та, за необхідності контролю цілісності, обчислювання криптографічних контрольних сум, при цьому ключ розшифрування записують до захищеного блока пам'яті зовнішнього апаратного засобу, а під час виконання програми на базовій машині, перед початком виконання захищеної ділянки коду програми стан блоків пам'яті даних захищеної віртуальної машини встановлюють відповідно до стану блоків пам'яті даних базової машини на

2

момент ініціалізації, у блок захищеної пам'яті захищеної віртуальної машини завантажують повністю або, у разі великого розміру, частками захищену ділянку коду програми, у блоці розшифрування захищеної віртуальної машини розшифровують завантажену захищену ділянку коду програми з використанням ключа, що зберігається у захищеному блоці пам'яті зовнішнього апаратного засобу, у разі необхідності контролю цілісності перевіряють контрольні суми за допомогою блока контролю цілісності захищеної віртуальної машини, потім у захищеному блоці пам'яті захищеної віртуальної машини виконують інструкції захищеної ділянки коду програми, при цьому стан блоків пам'яті даних захищеної віртуальної машини змінюють відповідно до інструкцій захищеної ділянки коду програми, після завершення виконання захищеної ділянки коду стан блоків пам'яті даних базової машини встановлюють відповідно до стану блоків пам'яті даних захищеної віртуальної машини, а подальше виконання незахищених ділянок коду продовжують у базовій машині.

Корисна модель відноситься до обчислювальної техніки, зокрема до засобів захисту комп'ютерних систем від несанкціонованого використання і може бути застосована для захисту програмного забезпечення (комп'ютерних програм або комплексів програм) від:

- нелегального розповсюдження та використання;

- дослідження важливих алгоритмів роботи програмного забезпечення методами зворотного проектування (відладки, дизасемблювання, декомпіляції тощо);

- несанкціонованої модифікації критичних алгоритмів (ділянок коду програми, що їх реалізують) роботи програмного забезпечення.

Відомо, що базова машина - це електронно - обчислювальна машина або віртуальна машина, яка виконує незахищену програму, а захищена віртуальна машина - віртуальна машина, яка

здатна виконувати інструкції базової машини, захищена від зовнішнього втручання та має власну захищену пам'ять, недоступну для прямих операцій ззовні.

Відомі також способи захисту програмного забезпечення за допомогою зовнішніх апаратних засобів, що використовують жорстко задані функції, які зберігаються в постійній пам'яті засобу у вигляді мікропрограм або задані жорсткою внутрішньою логікою мікросхеми апаратного засобу. У переважній більшості випадків такі функції не можуть бути змінені користувачем. Навіть якщо зовнішній апаратний засіб дозволяє перезаписати мікропрограму у постійній пам'яті, необхідна передача нової мікропрограми у захищеному вигляді та перепрограмування пам'яті засобу захисту. В якості прикладу можна навести USB - ключ захисту програмного забезпечення HASP HL (виробник - Aladdin Software Security R.D.) Кількість перепрограмувань обмежена

(13) U

(11) 26708

(19) UA

максимально допустимою кількістю циклів перезапису постійної пам'яті. Крім того, об'єм постійної пам'яті таких засобів захисту досить обмежений, тому занести в постійну пам'ять можливо тільки невелику за обсягом (кількістю інструкцій) мікропрограму. Збільшення об'єму постійної пам'яті суттєво підвищує ціну засобу захисту.

Найближчим за технічною суттю аналогом, який вибрано у якості прототипу, є спосіб захисту програмного забезпечення із застосуванням апаратних засобів (В.Вервейко, С.Полчанинов, "Защита программного обеспечения с применением аппаратных средств", Материалы IX Международной научно - практической конференции "Безопасность информационных технологий", Киев, 2006, с.86-87), сутність якого полягає у винесенні деяких функцій алгоритму програми, що захищається, у зовнішній апаратний засіб, захищений від втручання ззовні, та обчислення результатів таких функцій безпосередньо апаратним засобом.

Даний спосіб за прототипом, як і спосіб захисту програмного забезпечення згідно з корисною моделлю, яка заявляється, включає використання зовнішнього апаратного засобу. Однак винесення частини функцій алгоритму програми, що захищається, у зовнішній апаратний засіб викликає труднощі у випадку змінення алгоритму функціонування програми, що захищається, які пов'язані з необхідністю змінення функції, що обчислюється зовнішнім апаратним засобом, оскільки це вимагає внесення змін до мікропрограми шляхом перепрограмування постійної пам'яті пристрою, що значно ускладнює роботу розробника та користувача захищеної програми. Крім того, функцію для апаратного засобу розробляють за допомогою інших засобів розробки та мов програмування, ніж ті, що використовувалися для розробки програми, що захищається. Ці недоліки призводять до зайвих матеріальних витрат та витрат часу при використанні способу за прототипом.

В основу корисної моделі поставлено задачу у способі захисту програмного забезпечення шляхом введення зашифровування важливої ділянки коду програми за допомогою криптографічного алгоритму та використання захищеної віртуальної машини, яка вбудовується у зовнішній апаратний засіб, й виконує зашифровані ділянки коду програми, забезпечити виключення необхідності перепрограмування апаратного засобу захисту при отриманні нових версій програми; можливість виконання мікропрограм великого розміру у апаратному засобі захисту без збільшення матеріальних затрат; виключення необхідності використання додаткових засобів розробки та вивчення нових мов програмування розробниками програмного забезпечення, що захищається, та, у підсумку, зменшити матеріальні витрати та витрати часу, необхідні для захисту програмного забезпечення.

Задача, яка поставлена, вирішується за рахунок того, що у відомому способі захисту програмного забезпечення, що містить

використання зовнішнього апаратного пристрою, відповідно до корисної моделі попередньо у зовнішній апаратний пристрій вбудовують захищену віртуальну машину, а важливі ділянки коду програми зашифровують за допомогою криптографічного алгоритму та, за необхідності контролю цілісності, обчислюють криптографічні контрольні суми, при цьому ключ розшифровування записують у зовнішній апаратний засіб, а під час виконання програми, перед виконанням захищеної ділянки коду програми виконують ініціалізацію захищеної віртуальної машини у відповідності зі станом базової машини, у власну захищену пам'ять віртуальної машини завантажують повністю або, у разі великого розміру, частками захищену ділянку коду програми, засобами захищеної віртуальної машини розшифровують завантажену захищену ділянку коду програми з використанням ключа, що зберігається у зовнішньому апаратному засобі, у разі необхідності контролю цілісності перевіряють контрольні суми, потім виконують інструкції захищеного коду програми, при цьому стан захищеної віртуальної машини змінюють відповідно до інструкцій захищеного коду, після завершення виконання захищеної ділянки коду стан базової машини встановлюють відповідно до стану захищеної віртуальної машини, а виконання незахищених ділянок коду продовжують у базовій машині.

Технічний результат, якого можна досягти при використанні корисної моделі, виражений в тому, що забезпечується виключення необхідності перепрограмування апаратного засобу захисту при отриманні нових версій програми; можливість виконання мікропрограм великого розміру у апаратному засобі захисту без збільшення матеріальних затрат; виключення необхідності використання додаткових засобів розробки та вивчення нових мов програмування розробниками програмного забезпечення, що захищається, та у підсумку, зниження матеріальних витрат та витрат часу під час створення захищеної програми та роботи із захищеною програмою.

В якості прикладу реалізації описаного способу розглянемо його реалізацію для найбільш розповсюдженої базової машини з 32-х розрядною Intel - сумісною архітектурою (IA-32). В наведеному прикладі припустимо, для спрощення, що у захищену пам'ять захищеної віртуальної машини завантажуються тільки захищені ділянки коду, для збереження даних використовується оперативна пам'ять базової машини, та захищена ділянка коду повністю вміщується у захищену пам'ять.

Перед зашифрованою ділянкою коду ставлять виклик функції ініціалізації захищеної віртуальної машини. Функція ініціалізації виконує наступні операції:

1. Встановлює стан регістрів загального призначення (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP) захищеної віртуальної машини відповідно значенням цих регістрів центрального процесора на момент виклику функції ініціалізації.

2. Встановлює стан регістру флагів (EFlags) захищеної віртуальної машини відповідно

значенню цього регістру центрального процесора на момент виклику функції ініціалізації.

3. Встановлює значення регістру EIP захищеної віртуальної машини, яке дорівнює адресі першої інструкції захищеного коду програми.

4. Передає управління захищеній віртуальній машині.

5. Після завершення роботи віртуальної машини встановлює значення регістрів центрального процесору EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EFlags у відповідності значенням цих регістрів захищеної віртуальної машини.

6. Передає управління за адресою, яка дорівнює регістру EIP захищеної віртуальної машини.

При цьому захист програмного забезпечення засобами захищеної віртуальної машини, вбудованої у зовнішній апаратний пристрій, може здійснюватись таким чином:

1. Завантажити захищену ділянку коду програми в захищену пам'ять захищеної віртуальної машини.

2. Розшифрувати в захищеній пам'яті захищену ділянку коду та перевірити її цілісність. Якщо цілісність ділянки коду не підтверджено, повернути управління функції ініціалізації з кодом помилки.

3. Якщо значення регістру EIP не знаходиться в межах адресів захищеної ділянки коду, закінчити роботу захищеної віртуальної машини і повернути управління функції ініціалізації.

4. Виконати інструкцію за адресою EIP, у разі необхідності завантажити дані з оперативної пам'яті базової машини та/або зберегти дані в оперативній пам'яті базової машини. За результатом виконання інструкції, обчислити нове значення регістру EIP.

5. Перейти до пункту 3.