



УКРАЇНА

(19) UA (11) 19187 (13) U
(51) МПК (2006)
G06F 9/445
G06K 13/00

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 ВІД НЕСАНКЦІОНОВАНОГО ВИКОРИСТАННЯ

1

(21) u200604340

(22) 18.04.2006

(24) 15.12.2006

(46) 15.12.2006, Бюл. № 12, 2006 р.

(72) Мочалов Олександр Олександрович, Гайша Олександр Олександрович

(73) НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕ-
БУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

(57) 1. Спосіб захисту програмного забезпечення від несанкціонованого використання, при якому розповсюджують програмне забезпечення, яке записане на носії з попередньо введеним у нього неповторюваним кодом, при цьому проводять виключення принаймні однієї функціональної компоненти, вводять у програмне забезпечення за допомогою елементів захисту коди апаратури і код копії програмного забезпечення, які передають постачальнику програмного забезпечення чи ліцензійному органу, від яких отримують і вводять у програмне забезпечення, що записане на носії, активаційний код, який відрізняється тим, що отриманий активаційний код являє собою цифровий підпис кодів апаратури та коду копії програмного забезпечення, а також персональних даних користувача, і програмне забезпечення перевіряє справжність цифрового підпису, що являє собою логічний захист програмного забезпечення, а також виконують дії з фізичного захисту програмного забезпечення, розшифровуючи деякі зашифровані двоключовим алгоритмом ділянки коду програмного забезпечення та вираховуючи контрольні суми ділянок коду, що захищається, і порівнюючи значення цих сум з еталонними значеннями, які зберігають у окремих двійкових файлах у зашифрованому двоключовим алгоритмом вигляді, а перед порівнянням розшифровують відкритим ключем, і після успішних перевірок цілісності та цифрового підпису дозволяють використання програмного забезпечення і проводять надання повної функціональності програмному забезпеченню, при наступних запусках програмного забезпечення перевіряють справжність і відповідність цифрового підпису виробника наявним кодам апаратури та коду копії програмного забезпечення і, при невдалій перевірці, передають нові коди апаратури постачальнику програмного забезпечення чи ліцен-

2

зійному органу, повторюючи дії по отриманню і перевірці активаційного коду, що виконувалися при першому запуску, а також виконують перевірку фізичної цілісності програмного забезпечення і, при неправильному цифровому підписі виробника або його відсутності та/або наявності модифікацій коду програмного забезпечення, завершують роботу програмного забезпечення.

2. Спосіб за п. 1, який відрізняється тим, що для забезпечення фізичного захисту контрольну суму вираховують як цифровий підпис послідовності байтів, що відповідає ділянкам коду, що захищають.

3. Спосіб за пп. 1, 2, який відрізняється тим, що операції з фізичного захисту програмного забезпечення проводять за допомогою блока коду перевірки цілісності, в який вбудовують критичну до виконання іншої функціональної частини програми інформацію і який зберігають у зашифрованому двоключовим алгоритмом вигляді, а перед виконанням розшифровують відомим відкритим ключем.

4. Спосіб за пп. 1, 2, який відрізняється тим, що для забезпечення фізичного захисту програмного забезпечення додатково розшифровують відкритим ключем необхідні ділянки коду програмного забезпечення, частину чи весь змістовий двійковий код якого зберігають в зашифрованому вигляді.

5. Спосіб за пп. 1, 2, 3 або 4, який відрізняється тим, що програмну систему захисту розміщують у відділеному від прикладної програми об'єктному файлі, який додатково тестують на цілісність шляхом вирахування контрольних сум його коду і порівняння їх з еталонними значеннями, які зберігають у окремих двійкових файлах, що поставляються разом із програмним забезпеченням, у зашифрованому вигляді, а перед порівнянням розшифровують відомим відкритим ключем.

6. Спосіб за пп. 1, 2, 3, 4, 5, який відрізняється тим, що проводять перевірку фізичного та/або логічного захистів не при завантаженні програмного забезпечення, а при виникненні деяких випадкових для користувача подій, які генерують за внутрішнім секретним алгоритмом програмного забезпечення.

(19) UA (11) 19187 (13) U

Корисна модель відноситься до галузі електроніки і може бути застосована для забезпечення захисту програмного забезпечення (ПЗ) персональних ЕОМ від несанкціонованого використання.

Поняття несанкціонованого використання включає в себе:

- нелегальне використання ПЗ без потрібних прав (ліцензії) за його призначенням;
- використання ПЗ не за його прямим призначенням, а для отримання якогось не передбаченого розробником результату (можливо, відомостей про внутрішню структуру ПЗ);
- викривлення коду ПЗ.

Реалізацію будь-якої з наведених загроз далі називатимемо „зломом” ПЗ.

Відомо про спосіб, за яким засіб підтвердження справжності в реальному масштабі часу визначає еталонний цифровий підпис для виконуваної програми, використовуючи вміст цифрового підпису, виключаючи ті ділянки виконуваної програми, для яких виконано адресну прив'язку завантажувачем програми. Засіб підтвердження справжності в реальному масштабі часу після завантаження виконуваного образу визначає цілісність цифрового підпису для перевірки того, що виконуваний образ не був модифікований неналежним чином, а також гарантує, що кожен із указників у виконуваному файлі не був переадресований неналежним чином. [Патент RU 2258252 С2, 10.08.2005 „Система и способ проверки подлинности динамически подключаемых исполняемых образов”]. Вказаний спосіб захисту не дозволяє проводити перевірку санкціонованості використання ПЗ за його ліцензією. Даний тип захисту програмного забезпечення підпадає під визначення фізичного.

Відомо також про спосіб, за яким проводять обмеження функціональності програмного забезпечення, і передають унікальний код ПЗ та код апаратури комп'ютера постачальнику, від якого отримують активаційний код і формують код на основі активаційного коду, коду апаратури та коду копії програмного забезпечення, потім порівнюють його зі схованим в даній копії ПЗ кодом, що є унікальним, і, у випадку їх збігу, дозволяють використання та/або копіювання програмного забезпечення, надаючи при цьому йому повну функціональність [Патент RU 2159953 С1, 27.12.2000 „Способ защиты программного обеспечения”]. Такий спосіб захисту не дозволяє проводити контроль модифікацій коду ПЗ. При реалізації такої модифікації можна уникнути процедури перевірки збігу еталонного та сформованого кодів, отже такий спосіб захисту є недостатнім, іншою вадою є присутність у ПЗ, що знаходиться на стороні зловмисника, еталонного коду, або процедури вирахування таких кодів. Отже, еталонний код є потенційно відомим користувачу ПЗ. Даний тип захисту програмного забезпечення підпадає під визначення логічного. Зважаючи на ідентичність методик отримання активаційного коду, даний спосіб обирається за прототип.

В основу корисної моделі поставлено задачу удосконалення способу захисту програмного забезпечення від несанкціонованого використання, що вирішується шляхом застосування двохланкового захисту з обов'язковим сумісним виконанням обох ланок, із досягненням технічного результату у вигляді підвищення стійкості способу захисту, а за рахунок цього - значним збільшенням часу, необхідного для злому програмного забезпечення, що використовуватиме даний спосіб. Стійкість захисту визначається затратами часу, необхідного на злом ПЗ.

Для зручності та однозначного розуміння доцільно навести розшифровки та визначення понять, термінів та позначень, що використовуються далі.

- Злом програмного забезпечення - виконання дій, що мають на меті створення можливості використання програмного забезпечення без санкції його виробника, або некоректним способом, або внесення змін у код програмного забезпечення.

- Фізичний злом програмного забезпечення - внесення змін у двійковий об'єктний код ПЗ, що сприяють виникненню можливості несанкціонованого його використання.

- Логічний злом програмного забезпечення - виконання дій, що в необхідній мірі імітують алгоритм способу захисту ПЗ, і спрямовані на виникнення можливості несанкціонованого його використання.

- Тиражний злом програмного забезпечення - ситуація, при якій система захисту ПЗ спроектована таким чином, що одна єдина копія ПЗ може бути нелегально розтиражована необмеженою кількістю адресатів із збереженням її належного функціонування.

- Система захисту - програмна компонента програмного забезпечення, яка виконує послідовність дій, що попереджують злом ПЗ (реалізує спосіб захисту у програмних кодах).

- Фізичний захист програмного забезпечення - дії, спрямовані на унеможливлення використання ПЗ, у двійковий код якого було внесено зміни.

- Превентивний фізичний захист - унеможливлення внесення змін до об'єктних файлів програмного забезпечення.

- Контроль цілісності - послідовність дій, які дозволяють дізнатися, чи було внесено зміни до об'єктних файлів програмного забезпечення.

- Критична до виконання частини програми інформація - відомості, без яких робота даної частини програми неможлива або безглузда. Наприклад, це масив байт, що є ключем або процедурою для розшифровки якоїсь частини програми.

- Логічний захист програмного забезпечення - дії, спрямовані на унеможливлення імітації роботи системи захисту.

- Контрольна сума - криптографічне поняття, яке представляє собою функцію, на вхід якої подається довга послідовність байт, а на виході отримуємо коротку послідовність байт, що залежить від довгої. Можливими випадками контроль-

ної суми є хеш-функція та цифровий підпис.

- Цифровий підпис (електронний цифровий підпис) - криптографічне поняття, яке представляє собою масив байт, що відповідає поданій на вхід послідовності байт, та може бути сформований лише власником секретних даних (ключа). Будь-хто може пересвідчитися у справжності цифрового підпису шляхом виконання якихось операцій над цим масивом байт за допомогою відкритого ключа особи, що створила цифровий підпис, або якихось інших механізмів.

- Імітація способу захисту - виконання дій, що є секретними у способі захисту, і спрямовані на отримання якихось відомостей або дій, що дозволяють розпочати використовувати ПЗ. Якщо програмне забезпечення вираховує за внутрішнім секретним алгоритмом еталонний активаційний код, то логічним зломом буде імітація цих дій у зовнішній програмі - генераторі. Якщо ж у програмному забезпеченні зберігається еталонний серійний номер у вигляді незашифрованого набору байт, то імітацією буде просте його використання після його знаходження у двійковому образі.

- Унікальні ідентифікаційні дані - сукупність відомостей, що однозначно ідентифікують умови використання даної копії програмного забезпечення. До унікальних ідентифікаційних відомостей можуть входити персональні дані користувача програмного забезпечення, серійні номери (коди) елементів апаратного забезпечення комп'ютера, особливості його програмного середовища (такі, як перелік встановленого програмного забезпечення, кількість файлів у заданих каталогах, структура таблиці файлової системи FAT, та ін.).

Поставлена задача вирішується шляхом створення способу захисту програмного забезпечення від несанкціонованого використання, при якому розповсюджують програмне забезпечення з попередньо введеним у нього неповторюваним кодом, при цьому проводять виключення по меншій мірі однієї функціональної компоненти. Далі вводять у програмне забезпечення за допомогою елементів захисту коди апаратури і код копії програмного забезпечення, які передають постачальнику програмного забезпечення чи ліцензійному органу, від якого отримують і вводять у програмне забезпечення активаційний код. Згідно з винаходом, отриманий активаційний код являє собою цифровий підпис кодів апаратури та коду копії програмного забезпечення, а також персональних даних користувача. Потім перевіряють справжність цифрового підпису, виконуючи операції з логічного захисту програмного забезпечення. Також виконують дії з фізичного захисту програмного забезпечення. Для цього розшифровують деякі зашифровані двоключовим алгоритмом ділянки коду програмного забезпечення, та вираховують контрольні суми ділянок коду, що захищається. Потім порівнюють значення цих сум з еталонними значеннями, які зберігають у окремих двійкових файлах у зашифрованому двоключовим алгоритмом вигляді, а перед порівнянням розшифровують відкритим ключем. Після успішних перевірок цілісності та цифрового підпису дозволяють використання програмного забезпечення і проводять надання пов-

ної функціональності програмному забезпеченню. При наступних запусках програмного забезпечення перевіряють справжність і відповідність цифрового підпису виробника наявним кодам апаратури та коду копії програмного забезпечення. При невдалій перевірці передають нові коди апаратури постачальнику програмного забезпечення чи ліцензійному органу, повторюючи дії по отриманню і перевірці активаційного коду, що виконувалися при першому запуску. Також виконують перевірку фізичної цілісності програмного забезпечення, і, при неправильному цифровому підписі виробника або його відсутності, та/або наявності модифікацій коду програмного забезпечення, завершують роботу програмного забезпечення.

У якості коротких відомостей, що розкривають суть винаходу, слід відмітити, що технічний результат досягається за допомогою використання не лише кодів апаратури і унікального коду даної копії ПЗ, а й персональних відомостей користувача, які у сукупності формують унікальні ідентифікаційні дані даної копії ПЗ. Це дозволяє при можливому незаконному тиражуванні ПЗ дізнатися про особу нелегального розповсюджувача. Крім того, за пропонованим способом захисту, програмне забезпечення не порівнює отриманий на основі активаційного коду, коду апаратури та коду копії програмного забезпечення код з еталонним кодом, що схований в копії програмного забезпечення, а впевнюється у справжності цифрового підпису унікальних ідентифікаційних даних, яким являється активаційний код. Таким чином, на відміну від прототипу, даний спосіб не дозволяє знайти у файлах програмного забезпечення якийсь еталонний код і підставити його замість того, що вираховується на основі даних, отриманих від виробника ПЗ. Крім того, спосіб захисту, що пропонується, передбачає додаткові операції, спрямовані на унеможливлення фізичного злому ПЗ. Можливі різні види як фізичного, так і логічного захистів, але ці дві компоненти обов'язково мають бути присутніми у комплексній системі захисту. Превентивний фізичний захист має перешкоджати внесенню змін до об'єктних файлів, як недозволеній в цілому дії. Такий тип захисту має впроваджуватися на рівні операційної системи і не може бути застосований у звичайних прикладних програмних продуктах. Фізичний захист на рівні прикладної програми реалізується у вигляді підсистеми контролю цілісності коду програмного забезпечення. В загальному випадку така підсистема вираховує контрольні суми коду програмного забезпечення та порівнює їх з еталонними контрольними сумами, що поставляються разом з програмним забезпеченням в окремих файлах у зашифрованому двоключовим алгоритмом вигляді і розшифровуються відомими відкритими ключами перед порівнянням. Такий метод зберігання еталонної інформації дозволяє всім читати її завдяки відкритості ключа для розшифровки, та не дозволяє нікому, крім постачальника ліцензій, у якого є секретний ключ, створити зашифровану необхідну еталонну інформацію. Обов'язковість одночасної наявності двох типів захисту у способі захисту програмного забезпечення від несанкціонованого використання

дозволяє створити замкнений контур захисту і перекрити усі загрози, що були визначені у процесі формалізації моделі захисту. Отже, використання запропонованого двохланкового способу захисту дозволяє замкнути захисний контур навколо об'єкту захисту, та відповідно виключає можливість його обходу, які використовуються зломщиками для існуючих (одноланкових) способів захисту. Таким чином, завдяки принциповій наявності можливості збору коду з оперативної пам'яті комп'ютера, злом програмного забезпечення залишається можливим лише після довготривалої роботи висококваліфікованого зломщика на протязі не менше півроку (при середніх розмірах виконуваного файлу та усіх використовуваних функціональних бібліотек - порядку 10 мегабайт). Як приклад унікальних ідентифікаційних даних комп'ютера, можна навести серійний номер процесору комп'ютера, що неможливо підробити. Як приклад цифрового підпису унікальних ідентифікаційних даних можна навести обчислення цифрового підпису на основі системи RSA, пошук колізії якого оцінюється тисячами років роботи найшвидших ЕОМ.

Згідно з запропонованим способом захисту контрольну суму вираховують, як цифровий підпис послідовності байт коду, що захищається.

В загальному випадку математична функція, яка застосовується для вирахування контрольної суми, що необхідна для перевірки фізичної цілісності виконуваного коду, може бути небієктною, і тому можливі її колізії. Для забезпечення необхідної криптографічної стійкості вказана контрольна сума має являти собою цифровий підпис послідовності байт коду, що захищається. В такому випадку виключається можливість злому системи захисту шляхом злому функції вирахування контрольної суми, бо, як було вказано, наприклад при використанні системи RSA, час, необхідний для математичних операцій по злому системи на найшвидших сучасних ЕОМ, наближається до тисяч років.

Згідно з запропонованим способом захисту, операції з фізичного захисту проводять за допомогою блоку коду перевірки цілісності, в який вбудовують критичну до виконання іншої функціональної частини програми інформацію, і який зберігають у зашифрованому двохключовим алгоритмом вигляді, а перед виконанням розшифровують відомим відкритим ключем.

Блок коду, що перевіряє збіг еталонної та вирахуваної контрольних сум при перевірці цілісності коду програмного забезпечення в загальному випадку може бути модифікований, тому потребує відповідного фізичного захисту. Блок коду програмного забезпечення, що виконує вказану перевірку, має бути зашифрованим двохключовим алгоритмом і розшифровується відомим відкритим ключем, крім того, в нього вбудована якась важлива критична інформація, без якої програма не функціонуватиме. Секретність другого ключа забезпечує неможливість модифікації блоку перевірки (його код представляє собою інформацію, яку можна читати, але не можливо підробити). Як приклад, можна навести двохключовий алгоритм шиф-

рування RSA, час злому якого на найшвидших сучасних ЕОМ наближається до тисяч років. Прикладом критичної інформації для виконання функціональної частини програми є ключ для розшифровки якихось її зашифрованих ділянок чи зашифрованої службової інформації виконуваної програми (заголовку PE-файлу).

Згідно з запропонованим способом захисту розповсюджують програмне забезпечення, частина чи весь змістовний код якого зашифрований двохключовим алгоритмом, окрім коду, що проводить розшифровку усіх зашифрованих ділянок. Під час роботи програмного забезпечення розшифровують відкритим ключем зашифровані ділянки коду по мірі необхідності у ньому.

Якщо весь код програмного забезпечення, окрім системи захисту є відкритим, тобто незашифрованим, то існує можливість відтворення його зломщиком у новому сформованому „вручну” виконуваному файлі, але без наявної системи захисту. Для запобігання такої можливості деякі функціональні ділянки коду програмного забезпечення мають бути зашифровані, що виключить вказану можливість і збільшить часові витрати кваліфікованого зломщика на декілька місяців (при середніх розмірах виконуваного файлу та усіх використовуваних функціональних бібліотек - порядку 10 мегабайт, та приблизно 50% частці різних рознесених по виконуваному файлу зашифрованих функціональних ділянок). Наприклад можливе шифрування алгоритмом RSA декількох значних функцій, вбудованих в програму.

Згідно з запропонованим способом захисту програмну систему захисту, що реалізує спосіб захисту, вміщують в окремий об'єктний файл. При цьому спочатку проводять тестування цього окремого файлу на цілісність, вираховуючи контрольні суми його коду і проводять порівняння з еталонними контрольними сумами, що поставляють разом із програмним забезпеченням у зашифрованому двохключовим алгоритмом вигляді, а перед порівнянням розшифровують відомим відкритим ключем. Потім проводять тестування на цілісність коду прикладного програмного забезпечення по схемі, розглянутій вище.

Для зручності розробки прикладних програм та можливості промислового використання запропонованого способу захисту програмного забезпечення від несанкціонованого використання систему захисту, що реалізує вказаний спосіб, слід розмістити у окремому файлі. Тоді для забезпечення фізичного захисту програмного забезпечення необхідно окрім тестування на цілісність виконуваного файлу програмного забезпечення ще додатково тестувати на цілісність окремий файл системи захисту, вираховуючи його цифровий підпис та порівнюючи з еталоном. Таке додаткове ускладнення, окрім додаткової зручності, має збільшити затрати часу зломщика не менш ніж на місяць, порівняно з вищерозглянутими способами за рахунок необхідності аналізу двох файлів з кодом, їх протоколу взаємодії, спільних даних, і т.д. Наприклад, можливе розміщення системи захисту у динамічній бібліотеці DLL.

Згідно з запропонованим способом захисту

операції з фізичного та/або логічного захистів проводять не при завантаженні програми, а при виникненні деяких випадкових для користувача подій, які генерують за внутрішнім секретним алгоритмом програмного забезпечення.

Загальним додатковим методом захисту програмного забезпечення від несанкціонованого використання є переховування місцезнаходження процедур, що уособлюють фізичний та логічний захист програмного забезпечення, у коді програмного забезпечення. Для утруднення локалізації таких блоків коду, до них слід звертатися не за відомими зломщику подіями, які він може відсте-

жувати, а за внутрішніми подіями програмного забезпечення, що є випадковими для зовнішнього спостерігача. Таким чином, виконувати операції з логічного та/або фізичного захисту програмного забезпечення слід, наприклад, після випадкового інтервалу часу з моменту завантаження програмного забезпечення. Такий додатковий контур захисту має збільшити час злому ще на кілька місяців, порівняно з вищенаведеними способами. Як приклад випадкової для користувача події можна навести потрапляння під час роботи курсору миші в точку екрана з заданими координатами (176,322).