Технічні науки

Ковтун Володимир Валентинович
Студент Національний технічний університет України
“Київський політехнічний інститут”
Ковтун Владимир Валентинович
Студент Национальный технический университет Украины
”Киевский политехнический институт”
Kovtun V.
Student National Technical University of Ukraine
“Kyiv Polytechnic Institute”

СИСТЕМА ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ ДО КОДУ
СИСТЕМА ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА К ИСХОДНОМУ КОДУ
SYSTEM SOFTWARE PROTECTION AGAINST UNAUTHORIZED ACCESS TO THE SOURCE CODE

Анотація: Розглянуто проблеми вже використовуваних захистів програм, а також програмних засобів, які використовують для обстеження програмного коду і висунута нова модель захисту коду програми.

Ключові слова: захист інформації , системи захисту від взлому.

Аннотация: Рассмотрены проблемы уже используемых защит программ, а также приложений, которые используют для обследования программного кода и выдвинута новая модель защиты кода программы.

Ключевые слова: защита информации, системы защиты от взлома.

Summary: The article considers problems are used for applications and also software tools used for inspection of code and launched a new security model code.

Key words: information security, protection system against hacking.

Вступ

В даний час існує величезна кількість програм, які допомагають фахівцям у всіх сферах їхньої діяльності. Наприклад, текстові редактори, які спрощують роботу з текстом, або кошти для розробки програмного забезпечення, які дозволяють програмістам створювати нові програми. Кожна з програм містить унікальні напрацювання або нововведення, яка відрізняє її від інших. Багато з розробників намагалися зберегти в секреті свою інтелектуальну власність і отримувати прибуток від розробок свого програмного забезпечення. З цією метою і були створені перші системи захисту, які вбудовувалися в програми на етапі проектування і представляли собою прості системи ідентифікації і автентифікації. Вони ґрунтувалися на знанні користувачем ключа (серійного номера), який поставлявся разом з програмним забезпеченням. Користувач при запуску програми вводив відомий йому ключ, потім система захисту звіряла його з еталонним, і якщо ключі збігалися, то користувач міг працювати з програмою, якщо ж ні, то виводилося повідомлення про те, що ключі неоднакові і програма не виконувалась. У перший час розвитку методів і систем захистів приділялося мало уваги, так як не було способів обійти існуючі. Але з розвитком інформаційних технологій стали з'являтися програмні засоби, що дозволяють зловмисникам вивчати роботу програм без знання вихідних кодів. Це були перші наладчики, за допомогою яких користувачі стали обходити існуючі методи захисту, в результаті чого багато розробників стали зазнавати збитків. З появою наладчиків стали активно розвиватися методи і системи захисту, які ускладнюють або унеможливлюють роботу наладчика. Незважаючи на безліч розроблених методів захисту програм, програмні засоби для вивчення коду також активно розвивалися, з'являлися нові засоби, такі як дизассемблер, декомпілятор і інші, які вже дозволяли краще розуміти логіку роботи програми і отримувати її вихідний текст.

На даний момент розвиток методів і технік захисту відстає від розвитку програмних засобів для вивчення коду. Розробники для захисту свого

програмного забезпечення використовують або застарілі методи і техніки захисту, або комерційні системи захисту. Деякі з цих систем надають ефективний захист і ускладнюють вивчення програм, інші ж спроектовані з використанням застарілих методів і технік, які швидко обходяться кваліфікованими програмістами. Тому необхідне створення нової системи захисту Модель системи протидії програмних засобів вивчення коду програмного забезпечення, яка повинна враховувати особливості програмних засобів для вивчення коду та існуючі способи протидії їм інших систем захистів або приклади їх обходу, так як це дозволить не допустити подібні помилки в проектуванні і розробити більш стійку систему.

Огляд основних типів програмних засобів вивчення коду

Для створення ефективної системи захисту необхідно ретельно дослідити засоби, які будуть застосовуватись зловмисниками для її вивчення. Адже в кожному програмному засобі є свої переваги і недоліки, які необхідно враховувати. Одним з програмних засобів, яке застосовується для вивчення роботи програми, є наладчик [1. С. 188-216].

Він дозволяє виконувати динамічний аналіз, в ході якого зловмисник отримує більше інформації про роботу програми. Це стає можливим, тому що наладчик створює нескінченний цикл всередині програми і чекає системної події налагодження. Коли таке повідомлення генерується, цикл переривається і викликається відповідний обробник подій. Подіями, які слугують командою для наладчика, є точки зупину, що дозволяють зупиняти процес. Таким чином, можна вивчити кожну інструкцію, виконувану програмою, перевірити змінні, аргументи стека і дізнатися дані, що знаходяться в цей момент в пам'яті, до того, як вони будуть перезаписані. Це дозволяє програмістам отримувати більше інформації про роботу програми, а також детально вивчати механізми захисту.

Існують три основні види точок зупину [2]:

1. Програмні точки зупину - часто використовуються, так як дозволяють зупиняти процесор при виконанні інструкцій програми, представляють собою однобітну інструкцію, яка зупиняє процес і передає управління обробнику переривань. Кількість встановлюваних програмних точок в програмі необмежено. Але при їх установці модифікується код програми, яку може відстежити механізм захисту цілісності і припинити роботу програми.

2. Апаратні точки зупину - даний тип точок встановлюється на рівні процесора в спеціальних регістрах і не модифікує програму, що вивчається. Всього можна встановити до чотирьох точок зупину.

3. Точки зупинки пам'яті - дозволяють змінювати дозвіл на сторінці пам'яті, в якій можуть міститися важливі дані. Завдяки цьому такі точки дозволяють детально вивчати механізми захисту, засновані на порівнянні введеного пароля з еталонним, так як еталонний пароль під час порівняння знаходиться в пам'яті у відкритому вигляді.

Стає очевидно, що наладчик являє собою сильний програмний засіб для вивчення програм, так як дозволяє отримувати багато цінної інформації про їхню роботу.

Наступний тип програмних засобів, так званий дизассемблер [3], дозволяє перетворювати машинний код в текст програми на мові асемблера, який дає можливість програмістам дізнатися архітектуру програми, а також визначити місце розміщення захисних механізмів. Ще один тип програмних засобів для вивчення коду називається шістнадцятирічним редактором [4]. Він дозволяє редагувати дані, такі як образ диска, вміст оперативної пам'яті і інші. Варто відзначити, що дані представлені у вигляді ланцюжків байтів в шістнадцятирічній системі числення. Завдяки цьому типу засобів стає можливим знайти в досліджуваної програмі впроваджений еталонний ключ без використання наладчика або дизассемблера. Однак у даний момент ключі генеруються в самій програмі і в відкритому вигляді не зберігаються. З цієї причини в існуючих системах захисту застосовується кілька методів і технік, які підвищують стійкість до вивчення програмного забезпечення, але разом з

тим мають свої недоліки. Наприклад, багато систем захисту спроектовані за загальним шаблоном, розкриття якого призводить до нейтралізації захисту. Під шаблоном розуміється застосування однотипних методів і технік протидії програмних засобів. З цієї причини багато хто з систем захисту є неефективними, так як алгоритм протидії їм, в більшості випадків, є схожим.

Модель системи протидії програмних засобів вивчення коду

Для усунення наведених недоліків в системах захисту була розроблена нова модель системи, яка дозволить створити унікальну систему захисту і більш ефективно протидіяти програмних засобів для вивчення коду.

Система являє собою систему впровадження модулів в програму яка захищається. Система впровадження на основі відомих алгоритмів впровадження коду вбудовує захисні модулі в програму. Захисні модулі створюються самим розробником і являють собою об'єктні файли, написані на мовах програмування C / C ++, асемблера і ін., Які містять в собі свої методи і техніки або застосовують існуючі, що дозволить усунути однотипність захисту, так як в разі вивчення зловмисником модуля захисту та розробки алгоритму протидії даний модуль може бути швидко замінений іншим, а вивчення системи впровадження із застосуванням програмних засобів також не наддасть зловмисникові ніякої корисної інформації. Ще однією перевагою такої системи є тісна інтеграція захисних механізмів з самою програмою, що дозволяє усувати критичні помилки на етапі проектування. До того ж такий підхід дозволяє створювати кілька модулів, які можна буде впроваджувати в програму поступово, у міру вивчення і обходу існуючих механізмів захисту.

Варто відзначити, що модулі можуть містити різноманітні захисні техніки, наприклад текст програми може бути зашифрований і розшифровуватися за допомогою системи впровадження або мати власний дешифратор. Тому система спроектована з урахуванням можливих модифікацій під потреби розробників, дозволить їм самостійно створювати більш ефективний захист і зберігати в секреті свою інтелектуальну власність.

Література:

1. Dang B., Gazet A., Bachaalany E. Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation. Indianapolis, 2014 – 383 p.
2. Общие сведения об отладке: точки останова. URL: http://msdn.microsoft.com/ru-ru/library/vstudio/4607yxb0%28v=vs.100%29.aspx (дата обращения: 20.12.2014).
3. Disassembler // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Disassembler (дата обращения: 18.12.2014).
4. Hex editor // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Hex_editor (дата обращения: 18.12.2014).