УДК 621.373

ОЦЕНКА СЛОЖНОСТИ ФУНКЦИЙ БАЗОВЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

М.В. ГРИГОРЬЕВ, К.П. КУРЕЙЧИК

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 2 февраля 2006

Данная работа посвящена проблеме автоматизации программно-аппаратной защиты программного обеспечения. Описывается принцип работы динамических библиотек в операционной системе в обычном режиме и режиме программно-аппаратной защиты. В работе предлагается метод оценки сложности функций базовых языков программирования на предмет эмуляции. Предложенный метод позволяет автоматизировать выбор наиболее подходящей функции для защиты, а также выявить недостатки архитектуры разрабатываемого программного обеспечения.

Ключевые слова: программно-аппаратная защита ПО, защита ПО, оценка сложности программной функции.

Введение

В настоящее время программно-аппаратная защита программного обеспечения (ПО) базируется на двух основных принципах: использование шифрования в программной и аппаратной частях защищаемого ПО и физического размещения защищаемых модулей в аппаратной части [1]. При использовании второго метода защиты вопрос о том, какую именно часть программного кода следует поместить в аппаратный модуль, обычно не рассматривают [2–4]. Решение данной проблемы оставляют разработчикам ПО, которые и должны определить важнейшие модули защищаемого ПО.

Данная задача имеет возможное решение: найти модуль программы, размер которого не превышает размера памяти аппаратного модуля. Если эти условия выполняются, то задачу можно считать решенной. В противном случае необходимо расширить объем памяти программной части, либо поместить только часть программного кода. Цель данной работы заключается в том, чтобы унифицировать процесс выбора части программного кода, который является наиболее сложным для эмуляции — повторение его взломщиками.

Принцип функционирования

Рассмотрим возможное решение поставленной задачи на примере использования динамических библиотек. Допустим, имеется один исполняемый файл и три динамические библиотеки (рисунок).

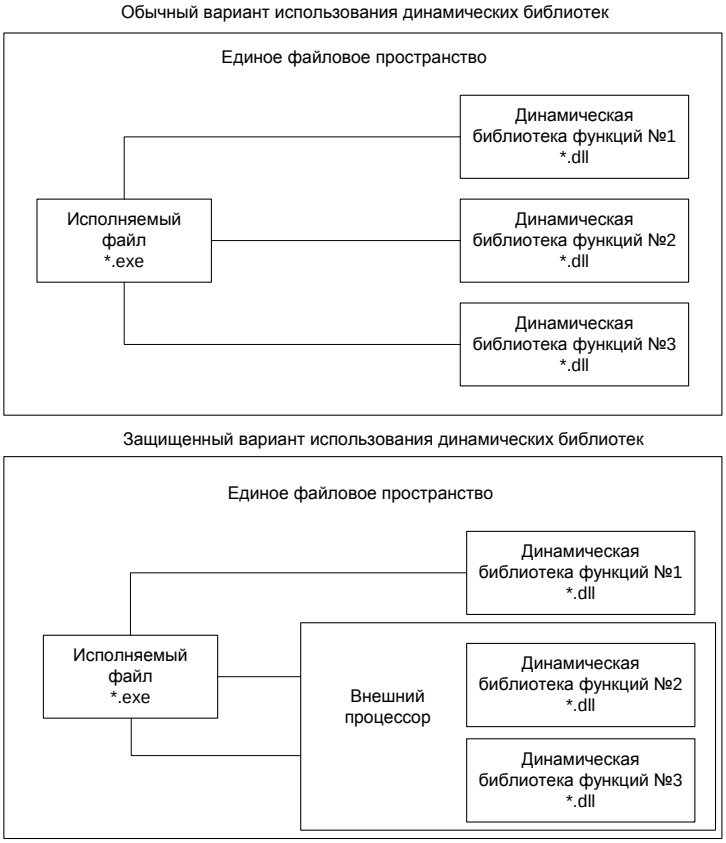

Рис. Схема использования динамических библиотек в обычном и защищенном вариантах

Пусть динамические библиотеки отображаются на память в любой момент времени функционирования ПО. Тогда момент выполнения функции из динамической библиотеки, отображенной на память, самый уязвимый, поскольку код функции может быть доступен для анализа [5].

Поместим теперь функции динамической библиотеки во внешнее защищенное устройство. В момент вызова функции библиотека не помещается в память, а выполняется внутри защищенного устройства, что и является основным фактором защиты.

Главный вопрос остается в поиске критериев отбора необходимых библиотек (функций, находящихся в этих библиотеках) для защиты так, чтобы максимально затруднить их повторение.

Исходные данные

Для того чтобы оценить сложность той или иной части программного кода, необходимо найти общие части всех языков программирования и выявить закономерности усложнения кода программы в зависимости от комбинаций этих частей.

Общими составляющими любого из языков программирования являются:

1) библиотеки языка (набор базовых функций, константы, переменные);
2) условия ветвления;
3) циклы;
4) переменные;
5) математические и логические операции;
6) функции;
7) выход из функции.

Для современных языков программирования, являющихся объектно-ориентированными, данный базис будет являться неполным. Этот базис будет полным для не объектно-ориентированного языка, такого как Си, Паскаль, являющихся прародителями современных языков программирования.

Оценка сложности должна заключаться в выделении из ряда функций самой сложной для помещения ее в аппаратную часть.

Алгоритм вычисления сложности

Алгоритм заключается в оценке каждой составляющей базового языка программирования:

1) вызов функции библиотеки языка отнимает 1 балл, поскольку данная функция потенциально не защищена от несанкционированного доступа;

2) условное ветвление оценивается как логическая операция, при этом, являясь усложнением функции, добавляет 1 балл к каждой составляющей, входящей в ветвление, если составляющая добавляет балл или отнимает его, если составляющая отнимает балл;

3) в цикле, как и при условном ветвлении, добавляется или вычитается балл от каждого содержимого цикла с учетом математических и логических операций;

4) наличие переменной, как и ее объявление не изменяет сложность функции, если с переменной не происходит операции;

5) математические и логические операции добавляют 1 балл;

6) вызов не защищенной функции (функции, не помещенной в аппаратную часть) приравнивается к вызову функции библиотеки и отнимает 1 балл, вызов защищенной функции добавляет один балл;

7) выход из функции помещенный не в конце функции отнимает один балл, помещенный в конце функции, ничего не добавляет и не отнимает.

Просуммировав полученные значения и разделив их на количество составляющих, мы получим коэффициент сложности функции. Коэффициент сложности имеет значение только в отношении набора функций, сложность которых вычисляют.

Примеры

Пример 1.

```
long func1( long p1, long *p2 )
{
	long k = p1;// 0
	*p2 = k*7// 1
	long ret = 9; //0
	for( long i = 0; i < p1; i++ )//+1+1
	{
	ret = *p2 + i; // 1+1
	k++;//1+1
	}

	if( k*k > *p2 )//+1
	{
	ret = ret*-1;//1+1
	(*p2)++;//1+1
	}
	else//+1
	{
	ret = stdfunc( k ) * ret; // +1-1-1+1 ( stdfunc – функция базовой библиотеки )
	}
```

```
	return ret;//0
}
```

Сумма 13. Количество суммирований 20. Результат: 13/20=0,65 — коэффициент сложности.

Пример 2.
Рассмотрим функцию, приведенную в примере 1 вместе с еще одной функцией:

```
long func2( long count, long p2 )
{
	for( long i = 0; i < count; i++ )//+1+1
	{
		ret = func1( i, &k );//+1+1 если func1 защищена и -1-1 если не защищена.
	}

	return p2;//0
}
```

Сумма 4 или 0. Количество суммирований равно 4. Коэффициент сложности 1 или 0 в зависимости от того, защищена ли функция func1. Это говорит о том, что защищать только функцию func2 бессмысленно, но необходимо защитить функцию func2, если защищена функция func1.

Заключение

Приведенную оценку функции сложно применить к современным языкам программирования, но она является первым шагом на пути автоматизации защиты ПО программно-аппаратным способом. Данный метод позволяет не только определить наиболее сложные программные функции для защиты, но и выявить недоработки архитектуры разрабатываемого ПО, что повысит производительность и сбалансированность на любом этапе разработки ПО.

ESTIMATION OF COMPLEXITY OF FUNCTIONS OF BASE PROGRAMMING LANGUAGES

M.V. GRIGOR'EV, K.P KUREJCHIK

Abstract

Present work is devoted to a problem of automation of hardware-software protection of the software. The principle of work of dynamic libraries in an operational system in a usual mode and a mode of hardware-software protection is described. The article represents the method of estimation of complexity of functions of base programming languages for emulation. The offered method allows to automate a choice of the most suitable function for protection and also to reveal the drawbacks of architecture of developed program

Литература

1. *Григорьев М.В., Курейчик К.П.* // Инженерный вестник. 2005. № 1 (20). С. 26–31.
2. *Панасенко С.П.* // Вопросы защиты информации. 2003. № 4. С. 29–34.
3. *Черней Г.А., Охрименко С.А., Ляху Ф.С.* Безопасность автоматизированных информационных систем. Кишинев, 1996
4. *Скляров Д.* Искусство защиты и взлома информации. СПб., 2004.
5. *Каплан А., Нильсен М.Ш.* Windows 2000 изнутри. М., 2000.