ОПРЕДЕЛЕНИЕ ЗАПУСКА ИЛИ РАБОТЫ ПРИЛОЖЕНИЯ В ВИРТУАЛЬНОЙ МАШИНЕ НА ОСНОВЕ АНАЛИЗА КОНФИГУРАЦИОННОГО ПРОСТРАНСТВА PCI УСТРОЙСТВ

Черемнов А.Г.

Национальный исследовательский Томский политехнический университет
8xandr@gmail.com

Объём программного обеспечения в мире постоянно возрастает, в связи с чем увеличивается и количество взломов по всему миру [1].

По данным Business Software Alliance и International Data Corporation уровень пиратства в 2008 году в России составил 68%, а в ряде стран бывшего СССР 90-95% [2].

К негативным эффектам пиратства относят замедление экономического роста в развитых странах, потерю финансовой прибыли и вскрытие алгоритмического и математического обеспечения, с последующим его бесплатным неконтролируемым распространением в глобальной сети интернет [1].

При анализе скомпилированного кода коммерческого приложения довольно часто используют виртуальные машины, поскольку они позволяют достаточно легко менять параметры среды окружения, операционной системы и оборудования.

Анализ электронных ресурсов распространения пиратского программного обеспечения в Российской Федерации, на подобие http://thepiratebay.cr свидетельствует о том, что применяемые подходы не лишены недостатков, в связи с чем существует огромная потребность в разработке новых механизмов детектирования виртуальных машин, основных на статичных и независимых от используемой операционной системы методах.

В процессе своей работы виртуальные машины используют специализированное виртуальное оборудование, которое обладает уникальным идентификатором по ряду причин:

1. диапазон ограничен и защищается производителями аппаратного оборудования.
2. подмена уникального идентификатора виртуальной машиной способно нарушить работу операционной системы, может привести к непредсказуемому поведению и даже полной неработоспособности.

В начале, используя базовую систему портов ввода/вывода, производится опрос шины PCI на наличие устройств.

Конфигурационное пространство устройства (рисунок 1), предназначенное для его идентификации и настройки, составляет 256 байт. Для доступа к конфигурационному адресному пространству в PC-совместимых архитектурах выделено два основных порта, согласно спецификации шины PCI [3].

Адрес	31 ... 24	23 ... 16	15 ... 8	7 ... 0
0x00	DeviceID		VendorID	
0x04	Status		Command	
0x08	Class Code			Revision ID
0x0C	BIST	Header Type	Latency Timer	Cache Line Size
0x10	Base Address Register 0			
0x14	Base Address Register 1			
0x18	Base Address Register 2			
0x1C	Base Address Register 3			
0x20	Base Address Register 4			
0x24	Base Address Register 5			
0x28	Cardbus CIS pointer			
0x2C	SubsystemID		Subsystem Vendor ID	
0x30	Expansion ROM Base Address			
0x34	Reserved			Capabilities pointer
0x38	Reserved			
0x3C	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

Рис. 1 – Структура конфигурационного пространства PCI устройств

0CF8h – W порт адреса Address (таблица 1);

0CFCh – RW порт данных Data.

Отметим, что порты является 32-битными.

По спецификации Intel также имеются и дополнительные порты для задания адреса (0CF9h-0CFBh) и данных (0CFDh-0CFFh) [4].

Таблица 1 – Структура порта Address

Номер бит	Содержимое
0	0
1	0
2-7	Индекс регистра
8-10	Функция
11-15	Устройство
16-23	Шина
24-30	Резерв
31	C

C – флаг доступа к устройству, когда он выставлен (принимает активное значение) начинается выполняться цикл.

Порт адреса задаёт шину, устройство и адрес регистра в конфигурационном пространстве устройства [3].

Число шин вычисляется подсчётом мостов PCI-to-PCI.

Если в ответ приходит 0FFFFh – то устройства не существует. Как было отмечено выше, устройства виртуальных машин имеют уникальные идентификаторы DeviceID, VendorID, SubsystemID и Subsystem Vendor ID в конфигурационном пространстве. Исходный код на языке Delphi для получения необходимых идентификаторов приведён ниже.

```
Function IsDevice(Bus,Dev,Reg,Fun
:Byte):DWORD;
var
Addr : DWORD;
Temp : DWORD;
Resultat : DWORD;
Begin
GetPortVal($0CF8, Temp, 4);
Addr := (1 shl 31) + (Bus shl 16) +
(Dev and $1F) shl 11 +
(Fun and $07) shl 8 + (Reg and $3F)
shl 2;
// Пишем в порт адрес
SetPortVal($0CF8, Addr, 4);
GetPortVal($0CFC, Resultat, 4);

// Возвращаем что было
SetPortVal($0CF8, Temp, 4);
Result := Resultat;
End;
```

Схема обхода представляет собой три вложенных цикла по параметрам:

— Bus = 00h...FFh;
— Dev = 00h…1Fh;
— Fun = 00h…07h.

Обход повторяется два раза, в первый раз производим подсчёт количества мостов N PCI-to-PCI. Таким образом, во втором обходе Bus=00h...Nh.

Начиная с операционной системы Windows Vista, любая возможность прямого доступа к портам устройств закрыта [5], возможно, с целью повышения безопасности и устойчивости операционной системы. Для прямого доступа к портам ввода-вывода используем драйвер, подробно описанный в [6]. Функциональная схема информационно-логического взаимодействия представлена на рисунке 2.

Модификация поведения программы при попытке её исследования осуществляется так, как показано ниже.

```
if (VendorID = $1013) and (DeviceID
= $00B8) and (SubvendorID = $1AF4)
and (SubsystemID = $1100) then begin
FlagResult := TRUE; goto EndFunc;
end;
if (VendorID = $1022) and (DeviceID
= $2020) and (SubvendorID = $1AF4)
and (SubsystemID = $1100) then begin
FlagResult := TRUE; goto EndFunc;
end;
   if (VendorID = $1033) and
(DeviceID = $0194) and (SubvendorID
= $1AF4) and (SubsystemID = $1100)
then begin FlagResult := TRUE; goto
EndFunc; end;
   if (VendorID = $1106) and
(DeviceID = $3038) and (SubvendorID
= $1AF4) and (SubsystemID = $1100)
then begin FlagResult := TRUE; goto
EndFunc; end;
```

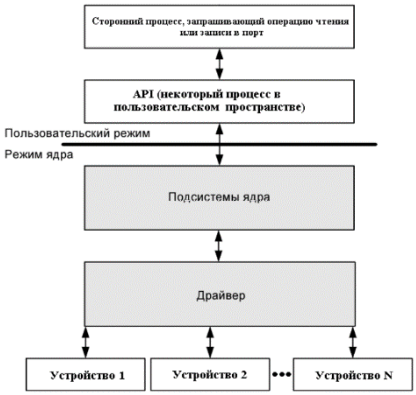

Рис. 2 – Функциональная схема информационно-логического взаимодействия между драйвером и процессом

В результате работы реализован принцип модификации поведения ПО при попытке его исследования, обеспечивающий повышенную безопасность программного кода, специализируемый модуль детектирования виртуальных машин с использованием драйвера для прямого доступа к портам, которые в настоящий момент используются для защиты программного обеспечения для восстановления информации с жестких дисков.

Литература

1. Даррелл Пейнетьер. Живучесть пиратства и его последствия для творчества, культуры и устойчивого развития. – Париж: UNESCO, 2005 – 21 с.

2. Sixth Annual BSA and IDC Global Software Piracy Study. URL: http://global.bsa.org/globalpiracy2008/studies/globalpiracy2008.pdf (Дата обращения: 20.10.2015)

3. Tanenbaum Andrew S. Modern Operating Systems. Third Edition. – University of Michigan, 2010. – p. 1120.

4. Intel® 64 and IA-32 Intel Architecture Software Developer's Manual. URL: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html (Дата обращения: 17.10.2015)

5. Microsoft Developer Network. MSDN Library. URL: http://msdn.microsoft.com/enus/library/default.aspx (Дата обращения: 17.10.2014)

6. Черемнов А. Г. , Аврамчук В. С. Разработка драйвера для прямого доступа к портам для Microsoft Windows 8.1 x86-x64 [Электронный ресурс] // Молодежь и современные информационные технологии: сборник трудов XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых: в 2 т., Томск, 12-14 Ноября 2014. - Томск: ТПУ, 2014 - Т. 1 - С. 277-278. - Режим доступа: http://www.lib.tpu.ru/fulltext/c/2014/C04/V1/C04_V1.pdf.