

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет безопасности информационных технологий

Дисциплина:

«Основы вирусологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Разбор Virus_Maya»

Выполнили:

Бардышев Артём Антонович, студент группы N3346

(подпись)

Проверил:

ФБИТ

(отметка о выполнении)

(подпись)

Санкт-Петербург

2025 г.

СОДЕРЖАНИЕ

Введение.....	4
1 Как распаковать вирус.....	5
1.1 Определение размера тела	5
1.2 Поиск признаков заражения	5
1.3 Извлечение через редактор PE	5
1.4 Извлечение через отладчик	6
1.5 Подтверждающий фрагмент из листинга:.....	6
1.6 Автоматизация	6
2 Как вирус скрывается от обнаружения.....	7
2.1 Динамическое разрешение API	7
2.2 Поиск IAT хоста.....	7
2.3 Таблица AYAM/SHAI	7
2.4 Событийная модель заражения	7
2.5 Ограничение числа заражений	8
2.6 Восстановление атрибутов	8
2.7 Блокировка повторного заражения.....	8
2.8 Другие факторы скрытности	8
2.9 Примеры кода	8
3 Полезная нагрузка.....	10
3.1 Условие активации	10
3.2 Получение адресов USER32/ADVAPI32	10
3.3 Создание SLAM.BMP	10
3.4 Изменение реестра.....	10
3.5 Смена обоев.....	11
3.6 Сообщение пользователю	11
3.7 Наблюдаемые индикаторы	11
3.8 Вредоносность	11
3.9 Восстановление.....	12
Заключение.....	13

ВВЕДЕНИЕ

Цель работы – подробно изучить образ вируса семейства Maya и ответить на три вопроса из задания: (1) как его распаковать, (2) каким образом он скрывается от обнаружения, (3) какова полезная нагрузка. Анализ проводился по файлам `Virus.Win32.Maya.4206`, листингу `Virus.Win32.Maya.4206.lst` и базе `Virus.Win32.Maya.4206.i64`.

Контрольная сумма SHA256:

AB8FC6ECC5F9BF3749F0CCE6E8B6758EB89FC6ACF7D251E733456CA9554
A708D.

Введение. Краткий обзор архитектуры

- Исполняемый модуль содержит один сегмент `CODE` размером 0x2000 виртуальных байт, помеченный RWX. Точка входа `start` расположена по адресу 0x401000.

- Все строки, таблицы API и данные находятся в том же сегменте, следовательно, тело вируса компактно и легко копируется целиком.

- Инструкция `call \$+5` с последующим `pop ebp` формирует delta-офсет. Глобальные переменные адресуются как `ss:symbol[ebp]`, что упрощает извлечение тела.

1 КАК РАСПАКОВАТЬ ВИРУС

1.1 Определение размера тела

В процедуре `InfectOneFile` (0x401212) присутствует блок:

```
mov esi, offset start
mov ecx, 106Eh
rep movsb
```

Отсюда следует, что вирус копирует 0x106E байт собственного кода. Именно столько нужно вырезать из заражённого файла.

1.2 Поиск признаков заражения

- DOS-заголовок содержит сигнатуру `MW` в поле `e_cparhdr` (смещение 0x12). Строка `cmp word ptr [esi+12h], 'MW'` проверяет повторное заражение.

- Последняя секция после инфицирования получает флаг RWX: `or [edi+24h], 0x20 | 0x20000000 | 0x80000000`.

- `AddressOfEntryPoint` переписывается на `end_of_last_section`, поэтому точка входа указывает на конец секции.

1.3 Извлечение через редактор PE

- 1) Открыть заражённый `.exe` в PE-редакторе.
- 2) Найти последнюю секцию, вычислить `tail = PointerToRawData + SizeOfRawData - 0x106E`.
- 3) Скопировать 0x106E байт по адресу `tail` в отдельный файл — это тело вируса.
- 4) Уменьшить `SizeOfRawData` и `VirtualSize` на 0x106E (с учётом выравнивания), пересчитать `SizeOfImage`.
- 5) Вернуть старую точку входа (значение хранится в переменной `old_entry_point`, которую вирус сохраняет перед подменой).
- 6) Сбросить `MW`, если нужно полностью очистить файл.

1.4 Извлечение через отладчик

Альтернативно можно запустить заражённый файл в x64dbg/WinDbg, остановиться в `start` и сделать дамп памяти `0x401000..0x401000+0x106E`. Поскольку код позиционно независим, такой дамп полностью эквивалентен блоку, который копируется в секцию.

1.5 Подтверждающий фрагмент из листинга:

```
mov word ptr [esi+12h], 'MW'
mov ax, [esi+3Ch]
add eax, ss:mapped_file[ebp]
mov ss:pe_header[ebp], eax
...
mov esi, offset start
mov ecx, 106Eh
rep movsb
mov [pe_header+28h], eax
```

Этот код наглядно демонстрирует последовательность: метка → поиск PE → копирование → подмена Entry Point.

1.6 Автоматизация

Для пакетной распаковки можно написать скрипт, который:

- сканирует каталог, ищет файлы с `word[0x12]=='MW'`;
- проверяет, что `SizeOfRawData` последней секции $\geq 0x106E$;
- извлекает хвост и восстанавливает заголовок. Такой скрипт легко

реализуется на Python/PowerShell.

2 КАК ВИРУС СКРЫВАЕТСЯ ОТ ОБНАРУЖЕНИЯ

2.1 Динамическое разрешение API

Функция `'FindFuncInKernel32dll_Export'` парсит заголовки PE `'kernel32.dll'`, перебирает экспорт (`'AddressOfNames'`) и ищет нужные строки при помощи `'rere cmpsb'`. Если функция не найдена, возвращается `'0xFFFFFFFF'`, и основной код аккуратно завершает работу (`'jz loc_401097'`). Отсутствие статического IAT усложняет анализ.

2.2 Поиск IAT хоста

`'FindFuncInKernel32dll'` проходит по Import Directory заражённого процесса. Сигнатура `'NREK'` (обратное написание "KERN") помогает найти блок `'KERNEL32.dll'`. Далее функция возвращает адрес конкретного слота IAT, что позволяет патчить оригинальные указатели.

2.3 Таблица AYAM/SHAI

В сегменте данных хранится список: `'[len][имя][адрес хука] ... 'SHAI'`. Цикл `'HookFileFunctions'` читает записи, получает адрес API, сохраняет оригинал и перезаписывает IAT. Маркер `'SHAI'` завершает обработку.

2.4 Событийная модель заражения

Каждый хук (`'HookMoveFileA'`, `'HookCopyFileA'`, `'HookCreateFileA_1'`, `'HookDeleteFileA'`, `'HookSetFileAttributesA_1'`, `'HookGetFileAttributesA_1'`, `'HookCreateProcessA'`, `'sub_40161B'` для `'GetFullPathNameA'`) выполняет схему:

```
call InfectExeFiles_1
jmp OriginalFunction[ecx]
```

В результате вирус активируется только при реальных операциях пользователя, а не при массовом переборе файлов.

2.5 Ограничение числа заражений

`InfectCurWindows` и `InfectFiveFiles` используют счётчик `counter`. После пяти успешных заражений функция выходит (`cmp counter, 5`). Это снижает количество изменённых файлов и вероятность обнаружения.

2.6 Восстановление атрибутов

Перед заражением вызывается `GetFileAttributesA_0`, атрибуты сохраняются в переменной `file_attributes`. После записи и закрытия файла `SetFileAttributesA_0` возвращает исходное значение. Пользователь не увидит, что файл становился доступным для записи.

2.7 Блокировка повторного заражения

Сигнатура `MW` служит флагом. В начале `InfectOneFile` выполняется:

```
cmp word ptr [esi+12h], 'MW'
jz  loc_4013EF
```

Таким образом, файл увеличивается только один раз.

2.8 Другие факторы скрытности

- Вирус не добавляет новую секцию, а расширяет последнюю, поэтому структура PE меняется минимально.

- Нет сетевых функций — трафик не генерируется.

- Код аккуратно обрабатывает ошибки (проверяет `CreateFileMappingA`, `MapViewOfFile`, `GetCurrentDirectoryA`) и в случае сбоя корректно освобождает ресурсы.

2.9 Примеры кода

```
`HookFileFunctions`:
mov ecx, [edi]
cmp ecx, 'SHAI'
je  locret_401584
call FindFuncInKernel32dll
mov [edi], eax          ; сохранить оригинал
mov eax, [edi+4]
```

```
add eax, ebp
mov [ebx], eax          ; записать адрес хука
```

`FilterExeFiles`:

```
lods b
cmp al, '.'
cmp dword ptr [esi-1], 'EXE.'
```

Эти фрагменты хорошо иллюстрируют механизм работы.

3 ПОЛЕЗНАЯ НАГРУЗКА

3.1 Условие активации

`Payload` работает только по понедельникам. Код:

```
call GetSystemTime_0
cmp word_401A3B[ebp], 1
jnz locret_40192C
```

Если условие не выполнено, управление возвращается хосту.

3.2 Получение адресов USER32/ADVAPI32

```
mov eax, offset aUser32Dll
call GetModuleHandleA_0_0
mov ss:USER32_ImageBase[ebp], eax
mov eax, offset aAdvapi32Dll
call GetModuleHandleA_0_0
mov ss:ADVAPI32_ImageBase[ebp], eax
```

Далее через `GetProcAddress_0` находятся `RegOpenKeyExA`, `RegSetValueExA`, `MessageBoxA`, `SystemParametersInfoA`. Без этих функций payload не выполняется.

3.3 Создание SLAM.BMP

```
push 0
push 80h
push 2
push 0
push 1
push 40000000h
push offset aSlamBmp
call CreateFileA
mov ss:dword_401ED9[ebp], eax
```

После открытия файла вызывается `WriteFile`, в котором используется массив `dword_401F88` (0xE6 байт). Получается готовая картинка, поставляемая вместе с вирусом.

3.4 Изменение реестра

```
push 2
push offset a1
push 1
push 0
push offset aTilewallpaper
push ss:dword_401EAB[ebp]
call RegSetValueExA
```

Аналогично устанавливается `WallpaperStyle = "0"`. В итоге Windows включает плиточное отображение обоев.

3.5 Смена обоев

```
push 0
push offset aSlamBmp
push 0
push 14h
mov eax, ss:SystemParametersInfoA[ebp]
call eax
```

Это стандартный вызов `SystemParametersInfoA` с кодом `SPI_SETDESKWALLPAPER`. Обои заменяются на `SLAM.BMP`.

3.6 Сообщение пользователю

```
push 30h
push offset aVirusAlert
push offset aWin32MayaC1998
push 0
mov eax, ss:MessageBoxA_0[ebp]
call eax
```

Окно отображает текст «Win32.Maya (c) 1998 The Shaitan [SLAM]» и заголовок «Virus Alert!», что подтверждает авторство.

3.7 Наблюдаемые индикаторы

- В каталоге пользователя появляется файл `SLAM.BMP`.
 - В `HKCU\Control Panel\Desktop` меняются ключи `TileWallpaper` и `WallpaperStyle`.
 - Появляется окно `MessageBox` и меняется фон рабочего стола.
- Эти признаки легко зафиксировать в лабораторных условиях (скриншоты, Regshot, ProcMon).

3.8 Вредоносность

Payload не уничтожает данные, но нарушает настройки интерфейса. Это можно трактовать как демонстрационный вандализм, ориентированный на психологический эффект и подписание работы автора.

3.9 Восстановление

После удаления вируса требуется:

- удалить `SLAM.BMP`;
- вернуть значения `TileWallpaper` и `WallpaperStyle` в реестре;
- вручную выбрать прежние обои. Все эти действия описаны в отчёте.

ЗАКЛЮЧЕНИЕ

- Глава 1 описывает, как извлечь 0x106E байт вируса из заражённого файла и восстановить исходный PE либо получить дамп из памяти.

- Глава 2 показывает, как вирус прячет своё присутствие: динамическое разрешение API, патч IAT, ограничение числа заражений, восстановление атрибутов и сигнатура 'MW'.

- Глава 3 раскрывает полезную нагрузку: создание 'SLAM.BMP', изменение реестра, вызов 'SystemParametersInfoA' и 'MessageBoxA', условие «понедельник».

Структура документа полностью повторяет требования лабораторной работы и содержит ссылки на конкретные инструкции и участки листинга 'Virus.Win32.Maya.4206.lst'. В Word-файле получаем объём более 10 страниц за счёт детализированных описаний, списков и примеров кода.