NETWARE EXECUTE ONLY ATTRIBUTE - CONCIDERED HARMFUL

PETER SZOR

Data Fellows Ltd, Päivantaite 8
02210 Espoo, Finland
Peter.Szor@datafellows.com
http://www.datafellows.com/

1 INTRODUCTION

For many years the Novell NetWare ExecuteOnly attribute has been recommended as a good way to secure programs against virus infections and to disallow illegal copying. Unfortunately, this attribute is not safe and can be harmful in some cases. A small bug in old versions of NetWare Workstation Shell (for example: NET3.COM from 1989) made it possible for some viruses to write to ExecuteOnly files, even though these viruses had no NetWare-specific code at all.

The same buggy shell still works with Novell NetWare 4.10 (and this shell version is still available from Novell's web site in object format) which means that this problem exists even today. By using an old and buggy shell, fast infectors can write into ExecuteOnly files on any NetWare versions. In addition, there are other ways to exploit the same vulnerability on any version of the NetWare shell.

This happens because the ExecuteOnly attribute is not a NetWare right, but just a plain NetWare attribute. The workstation shell is supposed to handle it, but in an unsecured DOS workstation this is impossible. Since there is no write protection at the server side for ExecuteOnly files, any program capable of subverting the shell checks can not only read and copy ExecuteOnly files but also *write to them*.

Since ExecuteOnly files can not normally be read by any program or even by the administrator, an infected ExecuteOnly file can not be detected by any virus scanner (local scanners or NLMs) : this will make recovery from such an infection a major problem.

2 EXECUTE, ONLY?

Under Novell NetWare it is possible to mark program files as ExecuteOnly. Only the supervisor has the rights to do this. Because these files are not available for read by anyone (including the supervisor) this is supposed to make them unreachable for any kind of access except execution or deletion (by the supervisor only). Unfortunately, this is not perfect and the problem is fundemental.

Before the workstation can access files from the server, it has to load the NetWare client software into memory. In case of a DOS workstation the client software consists of two TSRs programs. First the client has to start IPX (low level communication protocol driver) and after that NETx, the Workstation Shell.

Early versions of NETx contained the DOS version number too. That's because the shell not only hooked DOS functions, but also patched them. It was impossible to use NET3 on DOS 5.0 or NET5 on DOS 6.0 without some strange problems. NETx adds several new sub-functions to DOS INT 21 handler (which is the most important of all of them). For example:

```
Function name in AH:

B6 Extended file attributes
B8 Print jobs
BB Set end of job status
BC Log/lock physical record
BD Release physical record
BE Clear physical record
BF Log/lock record
C0 Release record
C1 Clear record
C2 Lock physical record set
C3 Release physical record set
C4 Clear physical record set
C5 Semaphores
C6 Get or set lock mode
C7 TTS
C8 Begin logical file locking
C9 End logical file locking
```

When IPX and NETx are running the user of the workstation is able to execute LOGIN to connect to the server. If the client requests for a file, the shell (NETx) checks the location of the file. There are no special functionalities in NETx for requests for local files, since it redirects all requests which need replies from the server. Thus all EXEC (INT 21/AH=4B) calls will be redirected to the file server if the path of the executable is mapped from the server. To understand how this works, let's have a look at the NetWare client code.

The INT 21 entry of the Workstation Shell:

Shell name: NET3.COM

Size: 41462

Version: NetWare V2.12 Rev. B - Workstation Shell for PC DOS V3.x

```
1AA3:0A3F 53          PUSH    BX
1AA3:0A40 8ADC        MOV   BL,AH
1AA3:0A42 80FB69      CMP   BL,69
1AA3:0A45 720D        JB      0A54
1AA3:0A47 80EBB4      SUB     BL,B4
1AA3:0A4A 724D        JB      0A99
1AA3:0A4C 80FB40      CMP   BL,40
1AA3:0A4F 7348        JNB   0A99
1AA3:0A51 80C369      ADD   BL,69
1AA3:0A54 B700        MOV   BH,00
1AA3:0A56 D1E3        SHL     BX,1
1AA3:0A58 2E          CS:
1AA3:0A59 FFA7C208    JMP     [BX+08C2]  ; Function-table
```

The first routine at the entry point of the shell checks the INT 21 sub-function and goes to different locations by using a function table. In case of an EXEC (4B) request the following function will take control:

```
1AA3:6271 FC          CLD
1AA3:6272 3C03        CMP  AL,03 ;Sub-function handler
1AA3:6274 77F8        JA      626E
1AA3:6276 3C02        CMP   AL,02
1AA3:6278 74F4        JZ      626E
```

Every EXEC function (implicitly) has to read the program code into the workstation memory first. To do this it has to be able to open the file for reading. As ExecuteOnly files are not available for open to any user (including the supervisor) the shell has to have a backdoor. That's why the definition of ExecuteOnly is imperfect. The workstation shell whispers "I am the Shell during EXEC, please give me access to this file." And its open/read/close requests for this specific ExecuteOnly file will succeed.

I am not going to document here how the backdoor works. That's because we would not like to write a paper for virus-writers.

After it calls a simple open file function:

```
1AA3:62EF B8003D      MOV    AX,3D00  ; Open for READ
1AA3:62F2 CD21        INT    21
```

Because the INT 21 entry-point of the shell is going to handle the Open function (3D), the programmer at Novell uses a simple INT 21 interrupt here. Unfortunately, every resident program can monitor this function. Thus this interrupt will be visible to fast infectors (virus which hooks INT 21 and waits for Open sub-function)

In our test we used Burglar.1150.A. This virus is "in the wild" and it is a typical fast infector. We used the above buggy Workstation Shell version and Novell NetWare 4.10. We created several test files in one directory by using supervisor rights and flagged them as ExecuteOnly.

We started the virus on the same workstation where we were logged in to the server with normal user rights (no supervisor rights) but we did have modify rights to the test directory. The virus could work perfectly as far as the user had modify rights on the test directory. **That means there is no write protection on ExecuteOnly files at the server side.** If a workstation is able to write to an ExecuteOnly file, the NetWare Server will allow it.

For an easy test we tried to use the DOS internal command ECHO, and redirected its output to an ExecuteOnly file. The command we used was:

```
ECHO X >> test.com
```

To our surprise, we were able to modify ExecuteOnly files with this method, even with the latest NETX.

Let's see how the INT 21 entry of a more current release of NETX.EXE looks like:

Size: 78654

Version: NetWare Workstation Shell v3.32 (931117) PTF)

```
1AB3:07B8 53           PUSH     BX
1AB3:07B9 9C           PUSHF
1AB3:07BA 8ADC         MOV      BL,AH
1AB3:07BC B700         MOV     BH,00
1AB3:07BE 80FB6D       CMP      BL,6D
1AB3:07C1 720D         JB       07D0
1AB3:07C3 80EBB3       SUB      BL,B3
1AB3:07C6 7210         JB       07D8
1AB3:07C8 80FB41       CMP      BL,41
1AB3:07CB 730B         JNB      07D8
1AB3:07CD 80C36D       ADD      BL,6D
1AB3:07D0 D1E3         SHL      BX,1
1AB3:07D2 9D           POPF
1AB3:07D3 2E           CS:
1AB3:07D4 FFA71D06     JMP      [BX+061D]
```

There are no big changes here, it is the same function-table handler.

In case of an EXEC (4B) request the following function will take control:

```
1AB3:4F8E 3C03         CMP      AL,03
1AB3:4F90 77F9         JA       4F8B
1AB3:4F92 3C02         CMP      AL,02
1AB3:4F94 74F5         JZ       4F8B
```

There are no changes here either. The way the shell requests an ExecuteOnly file is the same too, but we can not document it in this paper.

But there is a major change in case of the open call. The Shell calls its own INT 21 entry-point directly -not through interrupts. This means that the call is not visible to other TSRs (viruses) anymore.

```
1AB3:501D B8003D        MOV     AX,3D00
1AB3:5020 9C            PUSHF
1AB3:5021 0E            PUSH    CS
1AB3:5022 E893B7        CALL    07B8; Shell entry-point
```

We used INTRSPY [1] to check the OPEN and EXEC interrupts. (INTRSPY is a small TSR which can monitor all interrupts.)

In case of NET3.COM (buggy Shell) the answer was the following:

```
EXEC: INT 21h AX=4B00 BX=0D03 CX=0D56 DX=41B9  Timer: 880420
File:         F:\VIRUS!\TEST.COM

OPEN: INT 21h AX=3D00 BX=008C CX=0012 DX=41B9  Timer: 880420
File:         F:\VIRUS!\TEST.COM
```

Thus the OPEN call from the Shell is visible to INTRSPY. The timer value was the same.

In case of NETX.EXE (fixed Shell) we received that:

```
EXEC: INT 21h AX=4B00 BX=0D03 CX=0D56 DX=41B9 Timer: 893702
File:         F:\VIRUS!\TEST.COM

OPEN: INT 21h AX=3D00 BX=0001 CX=0000 DX=001F Timer: 893810
File          A:\REPORT.TXT
```

We can see an OPEN call here, but this call is already for another file.

3 CONCLUSON

The ExecuteOnly flag can not stop illegal copying or viruses alone. An attacker has too many methods to bypass this particular protection.

Attack 1: The attacker can save an image of the executed program from the memory of the workstation. Sometimes this can be difficult (for example, with relocated code), but generally this is feasible.

Attack 2: The attacker uses a buggy shell with a resident program, which hooks INT 21 and waits for file open. Then the program copies the file to some other location without the ExecuteOnly flag.

Attack 3: The attacker modifies the Shell code in memory (or in the NETx program file itself) by patching few bytes. This removes the ExecuteOnly protection. From that workstation every user/program can copy/ modify ExecuteOnly flagged programs. In this case the attacker can recreate an ExecuteOnly program at the same location (read it to memory/recreate/write it back with the same name), and the ExecuteOnly flag is going to be disappear. This makes the file available to every user.

Is this the end of Novell NetWare security? Definitely not. Real rights are handled by the NetWare server correctly. However we must not think that ExecuteOnly is a secure solution in NetWare.

In the future we have to be more careful with sentences like the text below:

“In NetWare 4.1, files may be set as "Execute Only," permitting a client to execute but not copy the file from the server. Many NetWare customers have found this attribute of great benefit as it prevents illegal copying and redistribution of corporate software. Windows NT Server 3.51 does not provide this functionality.”

“…the Execute-Only attribute cannot be cleared, nor can the executable file open for read or write. This means that even if a NetWare-aware virus were written, the virus could not infect a program flagged Execute-Only…”

REFERENCE

[1] Andrew Schulman, Ralf Brown, David Maxey, Raymond J. Michels, Jim Kyle - UNDOCUMENTED DOS; 2nd edition, 1994, Addison-Wesley Publishing Company

ACKNOWLEDGEMENTS

Special thanks to Ferenc Leitold and Vesselin Bontchev whose knowledge and experience have contributed to this paper.