NTFS Alternate Data Streams

H. Carvey, CISSP

Introduction

Windows NT (WNT) and *Windows 2000* (W2K) have powerful graphical user interfaces that make the job of assessing the security condition of in addition to securing these operating systems considerably easier. Changing the bad logon limit is, for example, relatively easy to both understand and do in both of these Windows operating systems. Providing adequate security does not, however, always involve working with mainstream features of applications, operating systems, and networks. *Alternate data streams* (ADSs) are an example. This little-known feature available with the *NT File System* (NTFS) in WNT 4.0 and Win2K (RICH98) has been available since the advent of NTFS in the first WNT release, WNT 3.1. Although this feature is relatively unknown by the vast majority of WNT users and administrators, it provides a potentially very powerful attack mechanism for malicious individuals intent on compromising and exploiting WNT and W2K systems.

What is an ADS? How can ADSs be created and how can executables be run in them? How can they be misused (e.g., by having malicious executables run in them)? How can they be found? This paper addresses these and other related issues concerning ADSs and security considerations.

What Is an Alternate Data Stream?

In the most fundamental sense, an ADS means embedding files in files (RUSS00). When a file is created, the data for that file are written to an unnamed data stream associated with a filename, such as “myfile.txt.” Other named streams in which additional information can also be stored can be created behind the unnamed data stream. In this case the syntax “myfile.txt:hidden.txt” (notice the colon) is used. A log of changes to the file can be maintained in ADSs, for example, so that all changes can be rolled back, if necessary.

A fixed length record within the *Master File Table* (MFT) is used to represent each file in the NTFS file system. Each record within the MFT contains 14 attributes for the file, such as the filename, standard information, and data, to name a few. These attributes can either be resident (i.e., they reside within the MFT record) or nonresident (i.e., there are pointers to the attributes that reside outside the MFT). The data attribute points to the contents of the file, which is contained in 512-byte clusters on the hard drive if the file is too large to reside within the MFT itself. NTFS MFT records can support multiple data attributes; each of these separate data attributes can point to a different set of clusters on the hard drive.

ADSs were originally implemented in WNT systems to allow Macintosh clients to save files with resource forks on NT servers along with the resource fork information (which, if not specifically preserved, will be lost). ADSs are still found only in the NTFS file system. As such, ADSs are thus not preserved for files that are transferred between disparate file systems, as with email attachments, FTP transfers, or when the file is copied to a FAT-formatted diskette. ADSs are preserved, however, if the source and destination file systems are NTFS.

Creating Alternate Data Streams

Creating ADSs is very straightforward. Assuming that the C:\ drive is NTFS-formatted, type:

```
C:\>notepad myfile.txt:hidden.txt
```

Once NotePad starts, click on the “Yes” button, type some text, then save the file. If the file “myfile.txt” does not already exist, it will be created. It will be 0 KB in size. You can see this using the dir command, or via the detailed view in Windows Explorer. However, you will not see the ADS named “hidden.txt” using either dir or Explorer.

You can also create an ADS by typing:

```
C:\>notepad :hidden.txt
```

Note that there is no other file name prior to the colon. This will produce a data stream that is visible neither through the dir command nor the Windows Explorer. There is also no telltale 0 KB file[1] to indicate the possible presence of a data stream.

You can create an ADS and move information into it using the type command. Assume, for example, that there is a file called “myfile.txt” that contains some text information. We can put this information into an ADS by entering the following command:

```
C:\>type myfile.txt >
myfile2.txt:hidden.txt
```

[1] The 0kB file does not provide a means to test for the existence of an ADS, however. Rather, it is simply used here as part of the example.

We can verify that the information was placed in the ADS by typing:

```
C:\>notepad myfile2.txt:hidden.txt
```

The echo and more commands will also permit you to create and read ADSs. For example:

```
C:\>echo This is my alternate data
stream > myfile.txt:hidden.txt
```

will create the ADS, while:

```
C:\>more < myfile.txt:hidden.txt
```

will allow the contents of the ADS to be read.

Interestingly, the copy command on WNT will not create ADSs, and will therefore not copy data to them. However, if copy is used to copy a file that has associated ADSs, the ADSs will be copied to the new file as well. This is also true for files that are moved across partitions using "drop-and-drag." Furthermore (as stated earlier), ADSs do not persist when the file is moved across disparate file systems, as when transferred via FTP, or as an email attachment, or when included in an archive of files, as with WinZip. As the authors of *Hacking Exposed* point out, the POSIX utility, cp.exe[2], can also be used to copy information to ADSs (MCCL99), as in the following example:

```
C:\>cp myfile.txt
c:\winnt\lanmannt.bmp:hidden.txt
```

ADSs can be created using other techniques. Using the Win32API::File module, ActiveState's ActivePerl can be used to create and write to an ADS. Perl can also be used to open an existing ADS using the open() command. The Perl script listed in Figure 1 demonstrates the use of Perl to create (and verify) an ADS, not only on the local system, but on remote systems, as well.

ADSs can be extremely useful to system administrators. If maintaining DSNs for databases on remote systems becomes unmanageable, records of commands or events can be stored in ADSs. Records of file changes, MD5 hashes of key system files (to verify file integrity) or key system information (serial numbers, configuration information, etc) can be stored in ADSs. When the existence of the data in the ADS is an important factor, the hidden data may be encrypted or digitally signed to ensure confidentiality and integrity. For example, system administrators may wish to generate and digitally sign a corporate identification tag. This tag can be "hidden" on the hard drive of the system to aid in tracking and managing corporate computer systems.

Running Executables in Alternate Data Streams

Text data are not the only information that can be stored in ADSs. In fact, an executable file can be placed in and run from an ADS. The authors of Hacking Exposed point out that an executable hidden in an ADS can be run using the "start" command (the command interpreter, cmd.exe, will not run executables hidden in ADSs). To demonstrate this, type the following commands:

```
C:\>type c:\winnt\notepad.exe >
myfile.txt:hidden.exe

C>:\start myfile.txt:hidden.exe
```

The second command will open a copy of NotePad.

Perl can be used to not only create an ADS (as demonstrated above), but also to run Perl scripts that are hidden in an ADS. For example, a Perl script, myscript.pl, that performs some task, can be moved to an ADS and run with the following commands:

```
C:\>Perltype myscript.pl >
myfile.txt:hidden.pl

C:\>Perlperl myfile.txt:hidden.pl
```

By using either the "start" command or Perl, WNT system administrators have a powerful tool at their

[2] This tool is from te NT Resource Kit.

```
#! c:\perl\bin\perl.exe
# Script created and tested with ActiveState
# Perl build 522

use strict;
use Win32API::File 0.02 qw( :ALL );

my $server = shift || Win32::NodeName;
my $path;

# Write alternate data stream to local or remote system

($server eq Win32::NodeName)?($path = “c:\\winnt\\”):
   ($path = “\\\\$server\\c\$\\winnt\\”);

my $targetfile = “lanmannt.bmp:test.txt”;
print “\n——————————————————-\n”;
print “Create and write to the alternate data stream:
”.$path.$targetfile."\n\n";

# This will fail if the file already exists

my $astream = Win32API::File::CreateFile( $path.$targetfile,
                                          GENERIC_ALL, 0, [ ],
                                          OPEN_ALWAYS,
                                          FILE_FLAG_SEQUENTIAL_SCAN, [ ]);

my $str =“This is an alternate data stream test created ”.localtime(time);
if ($astream){
Win32API::File::WriteFile($astream,$str,length($str),[],[]);
Win32API::File::CloseHandle($astream);
print “Alternate file stream created!\n”;
}
else {
print “Alternate file stream creation failed: $^E\n”;
}
print “\n——————————————————-\n”;
print “Open and read the created alternate data stream...\n\n”;
open(FL,$path.$targetfile) ||  die “Could not open file: $^E\n”;

while (<FL>) {
chomp;
print “$_\n”;
}
close(FL);
```

Figure 1 - **Using PERL to create and verify an ADS**

disposal. Commands can be placed in ADSs and executed at specific intervals without the user even knowing that the files exist. The security descriptor of the files that they hide behind can protect the code (or commands) in the ADS. However, any tool that is powerful and useful to the administrator will be equally powerful and useful to a malicious individual who has gained unauthorized access to WNT systems.

The Dark Side

Just as ADSs can be extremely useful to the system administrator, they can be just as powerful and useful in the hands of someone who has gained unauthorized access to an WNT or W2K system. Consider an e-commerce web site at which visitors must enter a username and password before being transferred to a secure, SSL-enabled portion of the site. This hypothetical site is hosted on an IIS 4.0 web server. A malicious individual may not have the ability or patience to properly handle the SSL portion of the web server, but may still be able to compromise the web server or operating system itself. This individual could then modify the ASP page such that usernames and passwords are saved in an ADS prior to being authenticated. The usernames and passwords can be harvested at a later date without the administrator ever being aware that the information was being collected and stored.

There are several applications and techniques available that can be combined with ADSs to increase their overall, devastating effect. Elitewrap, available from the PacketStorm security archives, can be used to create an executable archive of various files that will be run in accordance with a packfile. Complicated exploits can be bundled into a single executable and hidden in an ADS.

For instance, the authors of *Hacking Exposed* use the example of hiding pwdump.exe in an ADS, then executing it via the "start" command at a later date. Pwdump.exe is a program that allows an administrative-level user to pull password representations from the SAM database, and is an integral part of the WNT password cracker, l0phtcrack. By combining this with Elitewrap, an attacker could hide the necessary files on a compromised WNT system. When executed, the Elitewrapped programs collect the password hashes and email them to a separate account.

Another attack might make use of the AppInit_DLLs Registry key (see Microsoft KnowledgeBase article Q197571 for the function of this key). This key can be used to list a DLL that is loaded whenever a GUI application is opened. A DLL designed to capture keystrokes and log them to an ADS could be loaded via this key, or possibly even hidden in this key.

Finding Alternate Data Streams

It should by now be obvious that ADSs constitute a major potential source of security-related threat in computing and networking environments in which WNT and W2K are found. Corporate information security policies should thus at a minimum require that system administrators perform regularly scheduled scans, particularly of key systems, to verify compliance with configuration standards. These scans should include a tool or process for detecting ADSs. Currently available tools available for detecting ADSs include:

- Streams.exe (written by Mark Russinovich). It is available from: http://www.sysinternals.com/misc.htm#Streams.
- "LADS" (written by Frank Heyne) It is available from http://www.heysoft.de/index.htm.

These tools use the BackupRead() and BackupSeek() API calls to locate ADSs.

Conclusion

Proper host configuration, in accordance with corporate information security policies, goes a long way in preventing and detecting the malicious use of ADSs. Employ the principle of least privilege when assigning levels of access (to systems, directories, and files) for users. Be sure to use the NTFS file system on all WNT and W2K systems, so that the file system access control list and auditing capabilities can be employed to the fullest extent possible. Enable auditing, particularly on key systems. Finally, regularly collect and analyze Event Log entries from all systems on a regular basis. Some automated tool - Perl is wonderful for this - will almost certainly be necessary. (Note: Attempts to create files or write to existing files in directories restricted from write access to users will appear as Event ID 560 in the WNT and W2K EventLog.)

Acknowledgements

Special thanks go to Joe Seitzer and Howard Kirk for verifying the Perl code for this article, and to Joe for independently verifying that Perl scripts can be run from within alternate data streams.

References

RUSS00 Russinovich, M, *Inside Win2K NTFS, Part 2.* Windows 2000 Magazine, November, 2000, pp. 45 51.

MCCL99 McClure, S., Scambray, J., and Kurtz, G., *Hacking Exposed: Network Security Secrets and Solutions*. Berkeley: Osborne, 1999.

RICH98 Richter, J. and Cabrera, L., *A File System for the 21st Century: Previewing the Windows NT 5.0 File System.* Microsoft Systems Journal, November, 1998. http://www.microsoft.com/msj/defaulttop.asp?page=/msj/1198/ntfs/ntfstop.htm

Other Resources

ActiveState Tool Corp.
http://www.activestate.com

Microsoft KnowledgeBase article Q105763, *HOWTO: Use NTFS Alternate Data Streams*

http://support.microsoft.com/support/kb/articles/Q105/7/63.asp
Note: see also Q193793 and Q188806.

Microsoft KnowledgeBase article Q197571, *INFO: Working with the AppInit_DLLs Registry Value*
http://support.microsoft.com/support/kb/articles/Q197/5/71.asp
NTFS and Alternate Data Stream FAQ, Frank Heyne
http://www.heysoft.de/nt/ntfs-alternate data stream.htm

PacketStorm Security archives
http://packetstorm.securify.com/trojans

Harlan Carvey, CISSP, is a network security engineer for Winstar Communications, Inc. He has performed vulnerability assessments, penetration tests, and security policy creation/review for the federal government, but much prefers the challenges associated with the commercial sector. He has been involved with many facets of security (to include operations, communications, physical, as well as computer and network) for the past 12 years. At one point, he even worked as a Java Programming consultant for IBM. He enjoys programming Perl on NT. He received his BSEE from the Virginia Military Institute, and his MSEE from the Naval Postgraduate School.