COVERT CHANNELS - ALTERNATE DATA STREAMS

In this article we will be covering the concept of covert channels relating to the file streams in the windows NTFS filesystem. To start with a covert channel is a way that an attacker can hide data from discovery while making it avalible to their partners in crime. Now there are many, many ways to setup a covert channel, but we will be looking at using the NTFS filesystem, the filesystem which microsoft has been using since NT4.

History
In order to understand the way this channel works you must understand how it is possible. When Microsoft implemented the NTFS filesystem they wanted to make it both more secure and also backwards compatible with the Apple MAC filesystem. NTFS allows an administrator to have a much more granular level of control over files and folders than was possible under the FAT filesystems. To accomplish both this control and the compatability each NTFS file actually has two parts. The first part can be thought of as the file data, this is where the actual data contained in the file is stored. The second part is the data about the file, this is where the controls and other file information is stored. But this functionality came at a cost, attackers are now able to hide data "behind" a file in the area reserved for data about a file, in an alternate filestream.

How is this done?
Well, certain commands are able to read and write data in the alternate filestream. Lets start with a simple example, in my test directory I have one file;

```
C:\covert>dir
 Volume in drive C has no label.
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  08:52 PM              .
2005/05/09  08:52 PM              ..
2001/08/23  02:00 PM             7,680 hostname.exe

                           1 File(s)          7,680 bytes
                           2 Dir(s)  22,563,475,456 bytes free
```

If I run that file I get a certain output;

```
C:\covert>c:\covert\hostname.exe
giles
```

Now I am going to create a text file which I want to hide from prying eyes;

```
C:\covert>dir
 Volume in drive C has no label.
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  08:57 PM              .
2005/05/09  08:57 PM              ..
2001/08/23  02:00 PM             7,680 hostname.exe
2005/05/09  08:58 PM               189 secret.txt

                           2 File(s)          7,869 bytes
                           2 Dir(s)  22,563,475,456 bytes free


C:\covert>type secret.txt
secret company data
~~~~~~~~~~~~~~~~~~
1111111111111111111
2222222222222222222
3333333333333333333
4444444444444444444
```

```
5555555555555555
6666666666666666
7777777777777777
```

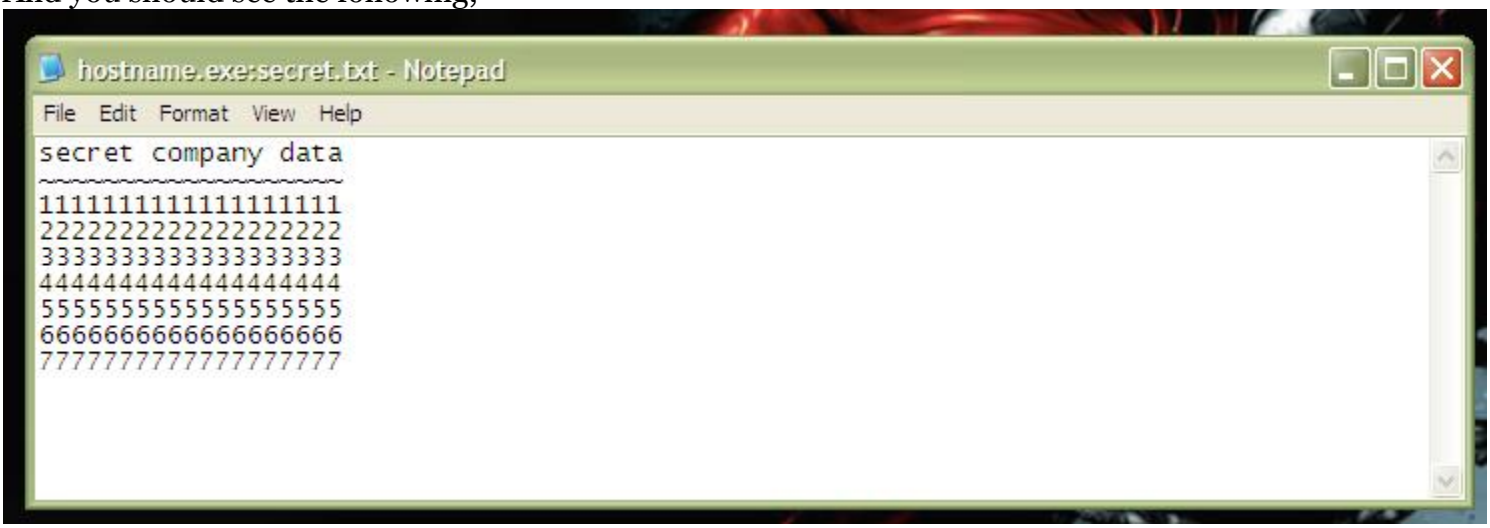
Now lets hide the text file behind the *hostname.exe* file;

```
C:\covert>type c:\covert\secret.txt > c:\covert\hostname.exe:secret.txt

C:\covert>del c:\covert\secret.txt

C:\covert>dir
 Volume in drive C has no label.
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  09:04 PM    <DIR>          .
2005/05/09  09:04 PM    <DIR>          ..
2005/05/09  09:04 PM             7,680 hostname.exe
               1 File(s)          7,680 bytes
               2 Dir(s)  22,563,475,456 bytes free
```

As you can see the *hostname.exe* has not changed in size according to the *dir* command. But try this;

```
C:\covert>notepad c:\covert\hostname.exe:secret.txt
```

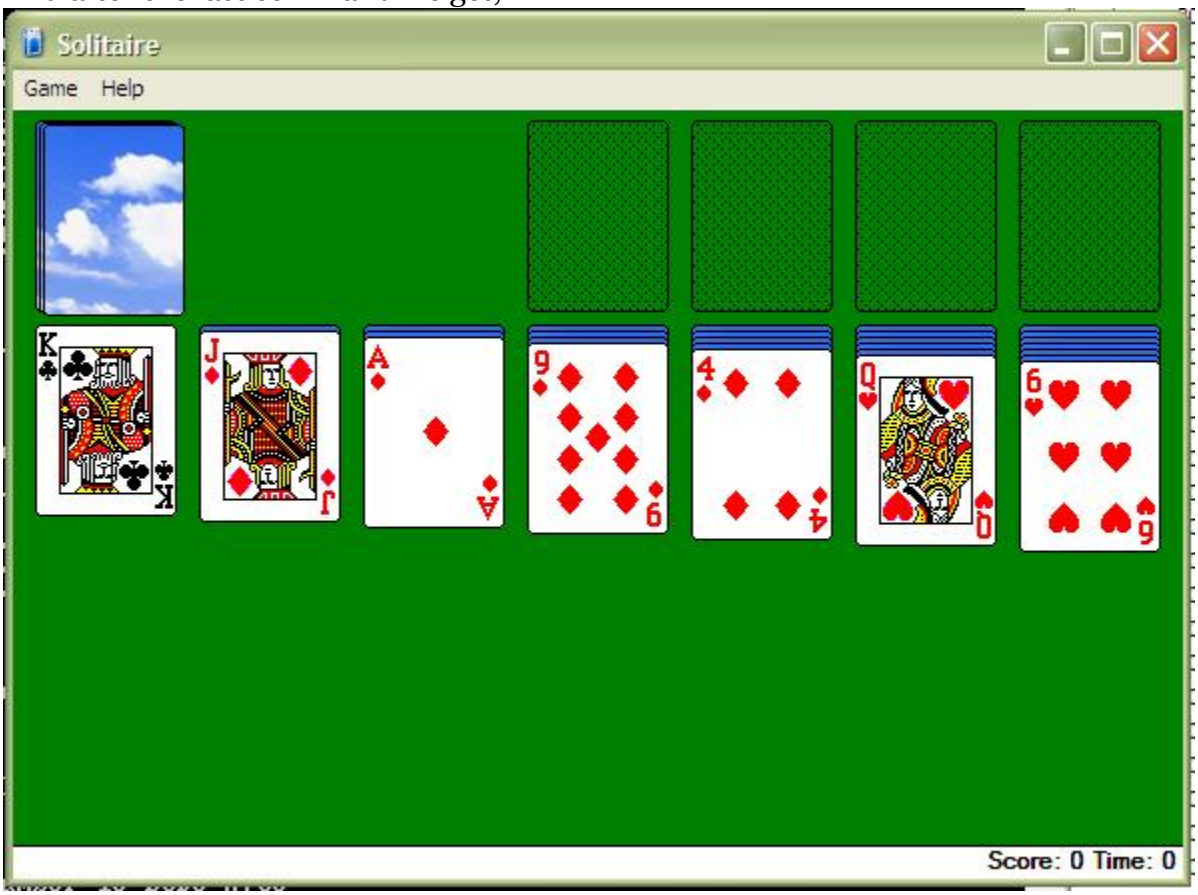
And you should see the following;



Now that is cool. But even better is that you can do this with an executable as well, I copied the *sol.exe* file into my test directory and followed all the steps to hide the file. I then deleted the *sol.exe* file from the test directory. If you do a *dir* after this you will see that *hostname.exe* has still not reported a change in size. Now we want to run the hidden exe file;

```
C:\covert>type c:\covert\sol.exe > c:\covert\hostname.exe:sol.exe

C:\covert>del sol.exe

C:\covert>dir
 Volume in drive C has no label.
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  09:19 PM    <DIR>          .
2005/05/09  09:19 PM    <DIR>          ..
2005/05/09  09:18 PM             7,680 hostname.exe
               1 File(s)          7,680 bytes
               2 Dir(s)  22,561,697,792 bytes free

C:\covert>start c:\covert\hostname.exe:sol.exe
```

And after the last command we get;



How is that? Also what you need to remember is that if anyone looked at the file in the test directory the file size would look normal, and more importantly if they ran the *hostname.exe* file, it would run as normal. A very nice way to hide data. But what if you want to get the hidden file back out again? I mean what is the good of a covert channel if you cannot get your data back again? Well for that you are going to need a little bit of outside help.

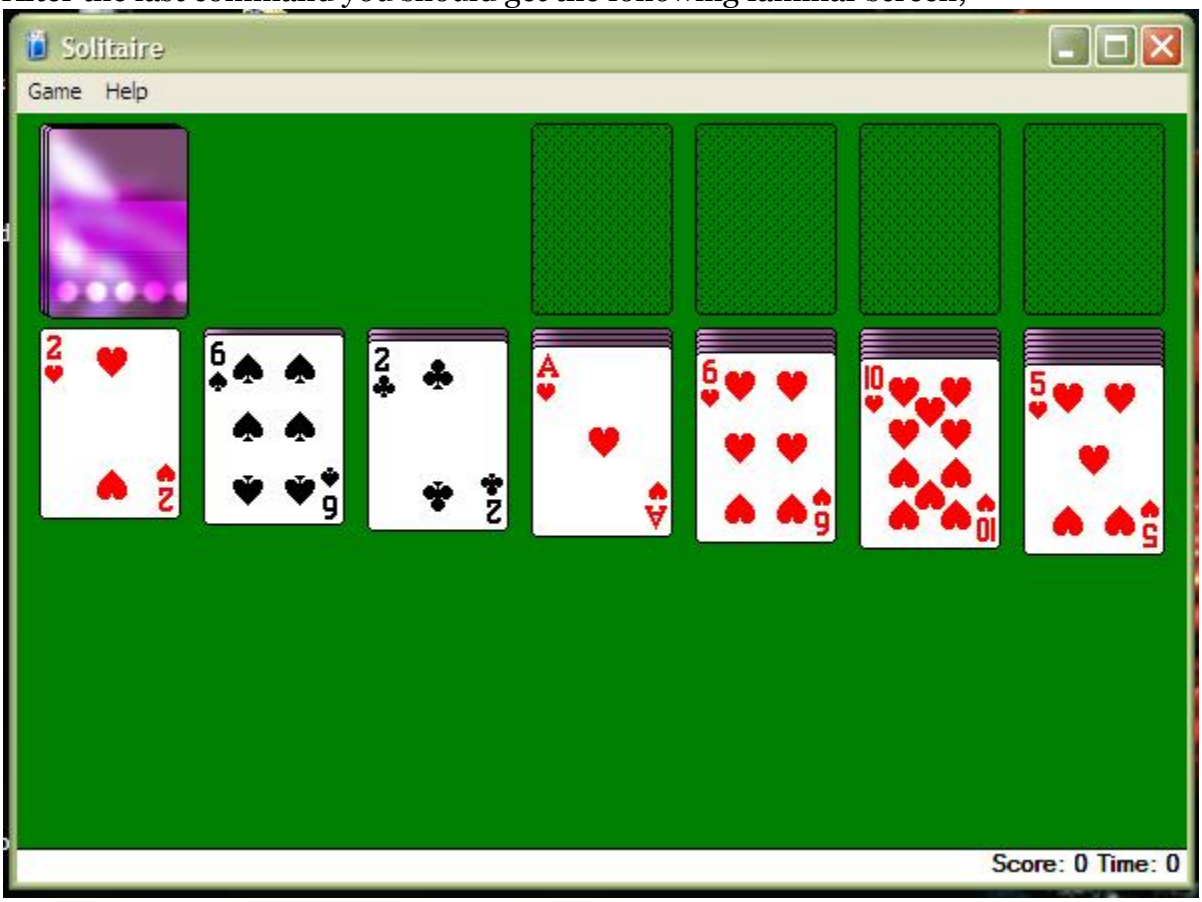
More Nefarious Deeds

First you need the *cp.exe* utility from the windows NT resource kit (for ease of use you can get it here), then you do the following;

```
C:\covert>dir
 Volume in drive C has no label.
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  09:29 PM    <DIR>          .
2005/05/09  09:29 PM    <DIR>          ..
2005/05/09  12:47 PM            46,352 cp.exe
2001/08/23  02:00 PM             7,680 hostname.exe
2001/08/23  02:00 PM            56,832 sol.exe
               3 File(s)        110,864 bytes
               2 Dir(s)  22,561,570,816 bytes free

C:\covert>cp c:\covert\sol.exe c:\covert\hostname.exe:sol.exe
c:\covert\sol.exe => c:\covert\hostname.exe:sol.exe [ok]

C:\covert>del sol.exe

C:\covert>c:\covert\hostname.exe
giles

C:\covert>cp c:\covert\hostname.exe:sol.exe c:\covert\out.exe
c:\covert\hostname.exe:sol.exe => c:\covert\out.exe [ok]

C:\covert>dir
 Volume in drive C has no label.
```

```
 Volume Serial Number is BCD5-A7C3

 Directory of C:\covert

2005/05/09  09:32 PM    <DIR>          .
2005/05/09  09:32 PM    <DIR>          ..
2005/05/09  12:47 PM            46,352 cp.exe
2001/08/23  02:00 PM             7,680 hostname.exe
2001/08/23  02:00 PM            56,832 out.exe
               3 File(s)        110,864 bytes
               2 Dir(s)  22,561,501,184 bytes free

C:\covert>c:\covert\out.exe
```

After the last command you should get the following familiar screen;



So now we can copy data into the alternate filestream, as well as copy it back out again. As of this time (05-2005), there is no version of windows that ships with any default tools which can be used to detect these alternate file streams. With the normal windows commands and programs there is no way for an administrator to find the hidden files. Now thats the scariest part of this whole thing!

All is Not Lost

Fear not however. There are freely avalible tools out there to help us combat this. One of them is the *streams.exe* utility from SysInternals, take a look;

```
C:\covert>cd \

C:\>streams

Streams v1.3 - Enumerate alternate NTFS data streams
Copyright (C) 1999-2001 Mark Russinovich
Sysinternals - www.sysinternals.com

usage: streams [-s] <file or directory>
-s     Recurse subdirectories

C:\>streams -s c:\covert
```

```
Streams v1.3 - Enumerate alternate NTFS data streams
Copyright (C) 1999-2001 Mark Russinovich
Sysinternals - www.sysinternals.com

c:\covert\hostname.exe:
         :sol.exe:$DATA 56832
```

As you can see I used it to recursively check the test directory I was using, and it found the alternate file stream, and gave me the details (just to show I am a nice guy you can get this file here). Also bear in mind that this little trick does have it's shortcomings. It will only work on NTFS filesystems, and can only be transferred to another remote NTFS filesystem using windows netbios protocols and removable -NTFS formatted- media. Any other transfer mechanism (FTP for example) or transferring the file to another type of filesystem (say an old windows 98 box using FAT32) and the alternate file stream would be lost.

Final Thoughts

As always, these are tricks the bad guys know already. So play around, have some fun, learn and start checking for it. You can only protect against what you know.