Windows File Pseudonyms

Strange Filenames and Haiku

OR:

Pwnage and Poetry

A quick disclaimer

Most of the techniques presented here are demonstrated on web-based technologies.

HOWEVER, the techniques and principles explained should apply to any application which takes a file path or name of any sort from user input which will be used on a Windows system.

(Furthermore, I will provide a bottle of delicious beer to anyone who finds a use for these techniques which isn't in this presentation!)

Each slide of this talk ~ Shall hence be accompanied ~ By clever haiku

The Big Problem ™

- Applications tend to do string-based analysis of file paths
  - Often, to decide how to handle files
  - Or whether or not files should be served
  - Or whether the file path may be malicious in nature
- We can reference one file or directory with many different *equivalent* strings
- Applications may not expect some of these strings
  - The OS interacts with the filesystem, not the app
  - Developers *may not even be aware* that their application will accept such names!
- ***Some of these names are based on undocumented features***

One cannot expect ~ To judge books by their covers ~ And find success

8 dot 3 format

It’s self explanatory

…only to a point

8.3 ALIASES

8.3 Aliases

- In DOS, file names must conform to the 8.3 format, meaning:
  - A basename of (max) 8 characters
  - Followed by one period
  - Followed by a (max) three character extension
- Windows likes backwards compatibility
  - Each file (and directory!) created on default Windows installations has a DOS-compatible name
  - If the name isn't 8.3 compatible, a second name (an 8.3 alias or short file name [SFN]) is made
- Highly simplified, they are generated as follows:
  - Start with the first six characters
  - Add a tilde
  - And a digit to distinguish from other files starting with the same characters
  - Add a period
  - And up to the first three characters of the file extension

Windows might as well ~ Be backwards-compatible ~ With the abacus

Why do I (audience) care?

- Because “highlight.php” != “HIGHLI~1.PHP”
  - If your IDS is looking for the old PHPBB highlight command exec flaw, it may not find it
- Because “.htpasswd” != “HTPASS~1”
  - Your forbidden files may no longer be forbidden
- File type may be determined by user input
  - Whatever follows the last dot is the file extension
  - So perhaps you can upload file.phPWNED
    - And request FILE~1.PHP

And you will know me ~ By the trail of characters ~ After the last dot

But wait, there's more!

Long file names

- Maximum length: 255
- Approx. char set size: 63000
- Approx. possible names: 63000^255 = 6.78E+1223

Short file names (8.3 aliases)

- Maximum length: 12
  (-1 for dot)
- Approx. char set size: 180
- Approx. possible names: 180^11 = 6.4E+24

If every file on a Windows box has an 8.3 compatible name (and by default, this is the case) we can immensely reduce the time and resources needed to guess filenames when trying to enumerate file names by brute-force. This can be further reduced if the file extension is known, reducing the complexity to 180^8 = 1.1E+18, or 110,000,000,000,000,000 possibilities!

Huge complexities ~ Collapsing conveniently ~ To simplicity

“Let’s get rid of stuff

At the end of filenames!”

Must have once been said.

DISCARDED TRAILING CHARACTERS

Discarded trailing characters

- Certain characters at the end of file names/paths are silently discarded by Windows!
- In Windows API calls, the following items are discarded:
  - Periods
  - Spaces
- The Windows shell will also allow, in specific configurations:
  - Double quotes (as long as they are closed properly)
  - Angle brackets (only in certain configurations)
  - Extraneous current directory markers (“/.”)
  - Extraneous parent directory markers with arbitrary items (fake or real)

Trailing characters ~ Of certain varieties: ~ Removed, silently.

Equivalent file path examples

- All of the following paths are valid and equivalent when given to the Windows shell:
  - file.txt
  - file.txt.....
  - file.txt<*spaces*>
  - file.txt”””””
  - file.txt<<<>><><
  - file.txt/././././.
  - nonexistant/../file.txt
- Different technique, similar use
  - “highlight.php” != “highlight.php.”
  - “restricted.txt” != “restricted.txt<space>”

Flexibility ~ In Windows nomenclature ~ Borders on silly

Ancient dinosaurs

Used to redirect data

With some devices

DOS SPECIAL DEVICE FILES

DOS special device files

- Similar to device files on *nix
- Allows file operations to be performed on devices
- Examples include:
  - CON, the console
  - PRN, a parallel printer
  - COM1, the first serial port
  - NUL, a bit bucket (/dev/null equivalent)
- Pretty well known already, BUT…

When you speak to me ~ I redirect all of it ~ To slash-dev-slash-null.

DOS special files quirk #1

- They exist “everywhere”
- Can be accessed from any path, even:
  - Directories which you are denied access to
  - With an existing file as a “directory” which “contains” the file
- Examples of equivalent paths to CON:
  - CON
  - C:\CON
  - C:\..\..\..\..\..\CON
  - C:\restricted_dir\CON
  - C:\existing_file.txt\CON

Like apparitions ~ They exist in every place ~ And yet in no place

DOS special files quirk #2

- They can have any file extension, it's ignored
- The following examples are equivalent:
  - CON
  - CON.bat
  - CON.php
  - CON.conf
  - CON.thisisalongandarbitraryfileextension
  - CON.<1000x"A">

Mr. Shakespeare knows ~ A rose by another name ~ Still smells just as sweet

Buffer overflow

- A Windows application takes in a file name
- The file is verified as existing
- If it exists, the program does something with the file name
  - And might trust that it doesn't exceed NTFS limitations
- *What if the file name is "CON.<'A'x1000>"?*
  - Technically, it exists…
  - …but not in the filesystem, so it's not bound to NTFS limitations

Why one needs all this ~ DOS file extension stuff ~ Is just beyond me

Controlling file handling

- Don’t forget:
  - You can use ANY extension!
  - Files are often handled based on extension
- DOS special files, then, can often be handled as ANYTHING YOU CHOOSE!
- http://www.example.com/com1.php
  - What if COM1 was attached to a serial modem?
  - …Or more likely, a Bluetooth dongle?

A riddle for you… ~ When is a CON not a CON? ~ When it’s a dot-jar!

What an awful mess!

I can’t write haiku about

Namespace prefixes...

NAMESPACE PREFIXES

Namespace prefixes

- Used when files can't be referred to with normal paths
  - Because they're really devices
  - Because they don't exist on the local filesystem
  - Because they have strange names

A distant echo ~ Of a victim, falling dead ~ The hunter shouts "PWNED!"

Minimal parsing prefix

- An invalid name or path can sometimes be used anyway
  - MAX_PATH can be exceeded
  - Some restricted characters can be used
  - Reserved basenames can be used
- Just precede it with \\?\
  - Must be an absolute path
    - No current directory indicator (./)
    - No parent directory indicator (../)

You don’t like the rules? ~ Double wack, question mark, wack. ~ You’re welcome, buddy.

UNC (Short and Long)

- Used to refer to files on SMB shares
  - Can be used to refer to files across the Internet
- \\server_name_or_ip\share\file
  - This is “Short UNC”
  - Nothing terribly special
- \\?\UNC\server_name_or_ip\share\file
  - This is “Long UNC”
  - Allows for the use of the \\?\ prefix with UNC paths

What’s the best thing ~ About SMB traffic? ~ Credential replay!

NT device namespace prefix

- Used to refer to device namespace
- These paths start with \\.\
- Examples include:
  - \\.\airpcap00\
    - An AirPcap card
  - \\.\GLOBALROOT\Device\HarddiskVolume1\
    - The first hard disk volume on the machine
    - Might be equivalent to, for instance, C:\
    - Doesn't need an assigned drive letter!
  - \\.\CdRom0\
    - The first disc drive on the computer
- WinObj from Sysinternals will allow you to browse the NT device namespace

The device namespace ~ Allows access to devices ~ Using file paths

NTLM credential capture

- When accessing SMB shares, authentication may be requested
  - If an attacker runs the SMB server, you can bet it will
- The SMB client machine will often send stored credentials automatically
  - And as you may know these credentials can be replayed or cracked
  - And we can trigger a machine to access an SMB share with a UNC path!

A replay attack ~ With SMB credentials ~ Should not still succeed!

Directory traversal

- “C:\” doesn’t match:
  - \\?\C:\
  - \\127.0.0.1\C$\
  - \\127.3.13.37\C$\
  - \\?\UNC\127.0.0.1\C$\
  - \\.\GLOBALROOT\Device\HarddiskVolume1\
- …but they’re all equivalent!

It is hard to stop ~ Directory traversal ~ Now more than ever

Buffer overflow

- Minimal parsing prefix allows for the use of paths exceeding MAX_PATH
  - Some developers don't know you can exceed MAX_PATH
  - …or assume that if the file exists that it can't exceed MAX_PATH

NOP NOP NOP NOP NOP ~ NOP NOP NOP NOP NOP Shellcode ~ Pointer to NOP sled

Making Windows rootkits deadlier

- Imagine that you're a Windows sysadmin
- Someone creates a file named "CON" with the minimal parsing prefix
- You try "type CON" at the command line
  - Your command prompt "hangs"
  - None of your programs open it properly
  - Windows Explorer can't delete it
  - You cry
  - You pretend it doesn't exist
    - or convince yourself it really should be there

My reaction to ~ "Undocumented feature" ~ Is unbridled rage.

Now I understand,

But I still don’t believe you.

SHOW ME THE MONEY!

DEMONSTRATION: NGINX AND PHP ON WINDOWS


Recycle Bin
Mozilla Firefox
PHP backdoor
nginx-0.7.63
start
php.ini - Notepad
Mozilla Firefox
C:\WINDOWS\Syste...
7:56 PM

Thank you!

There’s no dumb question…

“Is the computer plugged in?”

Is pretty bad, though.

Do I have the time

To continue presenting?

That sure would be nice…

BONUS ROUND (DELETED SLIDES)

So?

- So “file.phtml” is processed as PHP code
  - And “FILE~1.PHT” is served without processing
- So “file.phPWNED” can be uploaded
  - And “FILE~1.PHP” can be executed

Wait, what did you say? ~ Remote code execution? ~ NOW I’m listening!

How are 8.3 aliases generated?

- It's somewhat complicated, but in short:
  - Remove incompatible characters
  - Convert spaces to underscores
  - Take the first six characters of the basename
  - Add "~<digit>"
    - The digit is used to distinguish files with names starting with the same six characters
    - This convention isn't used after the first 3 collisions
  - Truncate the file extension to three characters
  - Make all the characters uppercase
- This is simplified due to time constraints, read my paper for more details!

Your name is too long ~ And uses weird characters. ~ Here's another one!

Denial-of-Service

- A theoretical application accepts file names and reads the associated files
- This application blocks any file named “CON”, “AUX”, “PRN” etc. to prevent DoS
  - Applications will generally pause to read from a file until EOF
  - EOF may never arrive from devices like AUX
- It does NOT block files named, for instance, “AUX.txt”
  - Which we know is equivalent to AUX

…And while we’re at it, ~ Since we’re speaking of Shakespeare… ~ All’s well that ends well!