DLL Hijacking Like a Boss!

Jake Williams
@MalwareJake
Rendition Infosec

$whoami

- Proud A+ certification holder
- Aspiring CEH
  - Hope to pass on my 3rd try next week
- Took computer classes in high school
  - 20 years ago - expert in QBASIC and Windows 3.1
- 11 yrs experience as a sandwich artist
  - Gives me unique perspective on dealing with business units who want “made to order” security solutions

Agenda

- Motivations
- What are DLL Sideloading and Hijacking
- Auditing Methods
- Hands on fun
- Practical defenses
- 100% fewer shirtless photos than John Strand's presentation

Why bother?

- Stealthy persistence technique for malware since most antivirus sucks at detecting DLLs
  - AV looks at the application, which is most often whitelisted and signed
- Attackers can make your multi-million dollar security investment obsolete
  - DLL Hijacking technique has been around for years, but AV still sucks
  - Executable whitelisting bypass

Is anybody doing this?

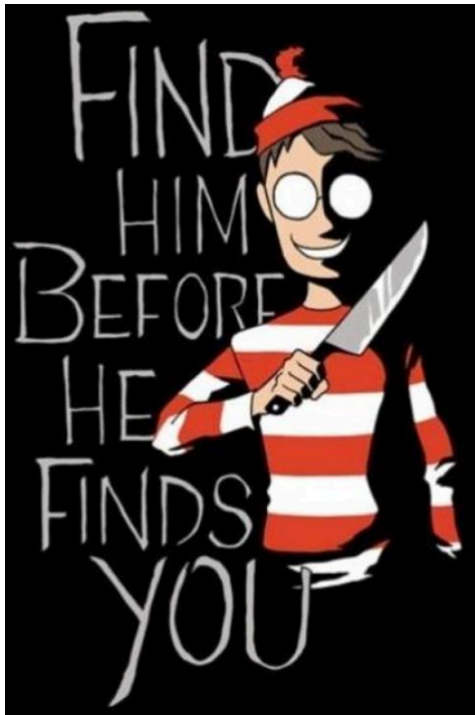
- Rendition sees this technique used in most Chinese APT compromises
  - As a pentester, if you aren't at least as good as Chinese APT then pack up and go home
- The best way to emulate a nation state threat is to find attackers who hack like a nation state

DLL Hijacking

- Import address tables don't specify a full path
- Hijacking abuses DLL search paths
- May offer privilege escalation when combined with weak directory permissions
  - Use icacls.exe to determine if you can write to the directory of a system process (e.g. a service)
  - Important note: I can't see "icacls" without thinking about the work "cankles"
    - Now you'll think of it too ☺

Where's Waldo?

- Sometimes a program may try to load a DLL that doesn't even exist on the system
  - Legacy code, checking for an available plugin, who knows why
- When the DLL fails to load, the code takes another path
  - Rob Lee calls this "Ghost DLL Injection"
- If you supply your own DLL, the app blindly loads it for you



Rules of Fight Club

- Safe DLL Search mode (on by default) moves the current directory to the end of the search order
  - Loading DLLs from working directory is dangerous
- Any DLL explicitly in the KnownDLLs registry key is not searched for since it is “known” to be located in the system directory

Registry Keys

- **KnownDLLs -** HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
- **Safe DLL Search -** HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode
- **CWDIllegalInDllSearch** - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\CWDIllegalInDllSearch

DLL Search Path (Unsafe)

1. The Current Directory.
2. The directory from which the application loaded.
3. System directory (C:\Windows\System32).
4. The 16-bit system directory (C:\Windows \System).
5. The Windows directory (C:\Windows).
6. Directories that are listed in the PATH variables.

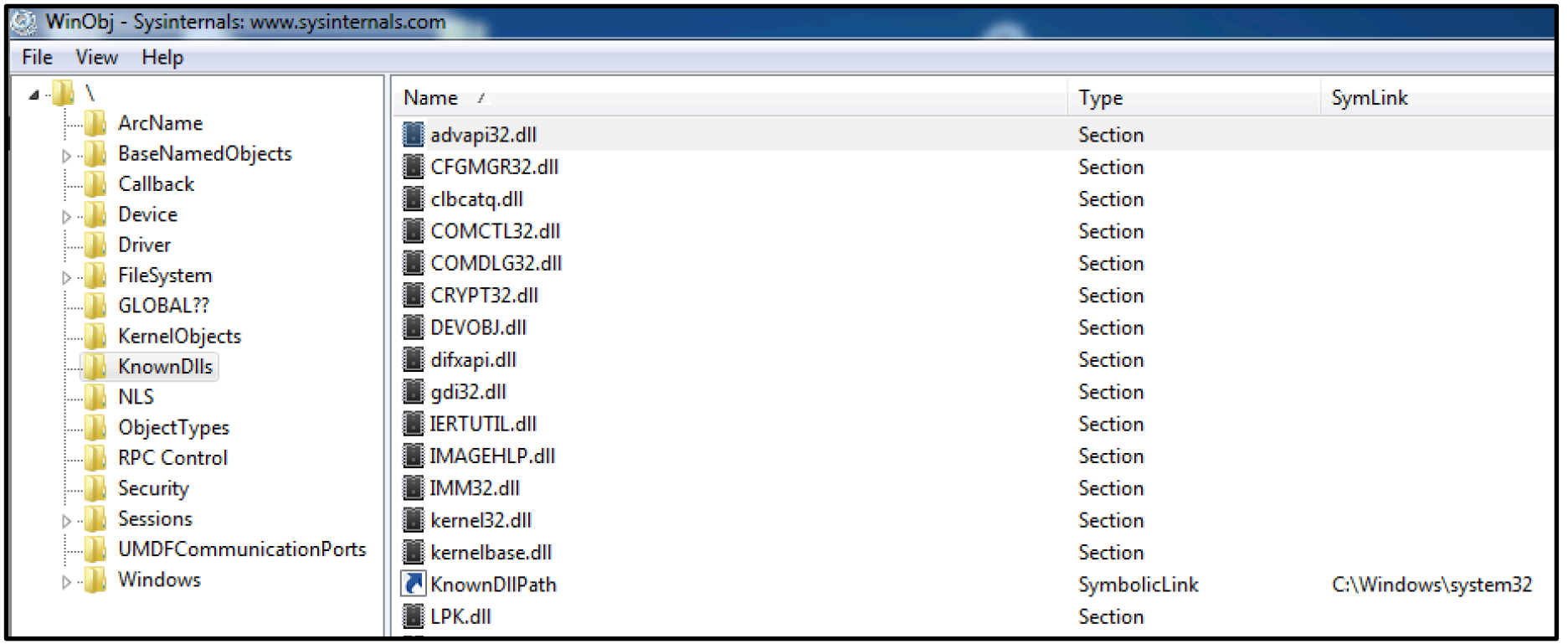
Default DLL Search Path (SafeSearch)

1. The directory from which the application loaded.
2. System directory (C:\Windows\System32).
3. The 16-bit system directory (C:\Windows \System).
4. The Windows directory (C:\Windows).
5. The Current Directory.
6. Directories that are listed in the PATH variables.

What's the common threa(t|d)?

- Either way, %PATH% is checked for the DLL if not found in another directory
- That's why we care significantly about DLL not found error messages seen in tool output
- Indicates that the application tried to load a DLL but it wasn't found
- Add a DLL and injection frequently follows

WinObj Shows KnownDLLs



Always know your KnownDLLs

- KnownDLLs don't use the typical search path
- They are always found in the system directory
- In some version of Windows, KnownDLLs are pre-loaded at system start
- KnownDLLs are not static across all versions of Windows
- The same application not vulnerable on XP may be vulnerable on Windows 8

KnownDLLs Strangeness

- Windows allows users to exclude some DLLs from KnownDLLs processing
  - Allows overrides of default behavior
- Don't rely on defaults in customer environments
  - Many poorly coded applications remove (or add) values to KnownDLLs
  - Tend to be custom apps for a customer environment

Testing for “Ghost DLL” injection

- Ghost DLL injection is hard to audit for completely since sometimes the DLL load request is conditional
- Audit methods
  - Run strings looking for DLLs not in IAT
  - Run code with a debugger attached and set a breakpoint on LoadLibrary
  - Use gflags with loader hints
  - Reverse engineer with IDA Pro

Tools for Auditing DLL Injection

- HDMoore's DLL Hijack Audit Kit
- Dependency Walker
- SecurityXPloded DLL Hijack Auditor
- Procmon ***__This is what you want__
- Sxstrace.exe
- Gflags + Windbg

HDMoore DIY for file extensions

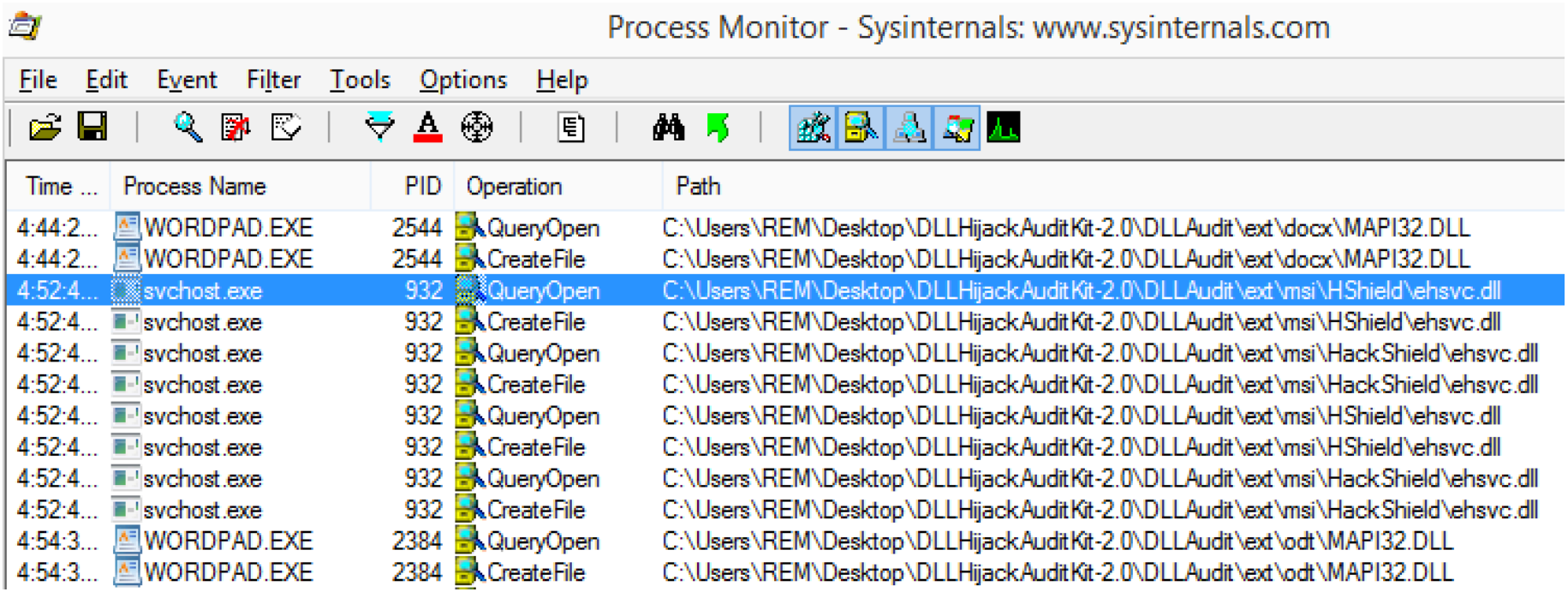
- HDMoore wrote an auditing kit for DLL hijacking that generates test cases and uses procmon to find exploitable test cases
- Runs two batch scripts, one to generate cases and another to examine procmon output
- Note that this does not find all exploitable conditions, only those for default file extension handlers

HDMoore DIY for file extensions (2)

- Used this to successfully find some vulnerable programs in a Windows 8.1 instance I use all the time
- Pros:
  - Very easy to use
- Cons:
  - Only tests default handlers and even then may be impacted by DLL loading delays
  - Does not test plugin dependencies

HDMoore DIY for file extensions (3)

- Ouch…



DLL Referenced != Always Loaded

- Wordpad.exe looks really bad, looking for mapi32.dll in the local directory
  - But it never loads the DLL from there
- Requires more investigation with a debugger to see in which circumstances it is exploitable
  - Beyond the scope of this presentation

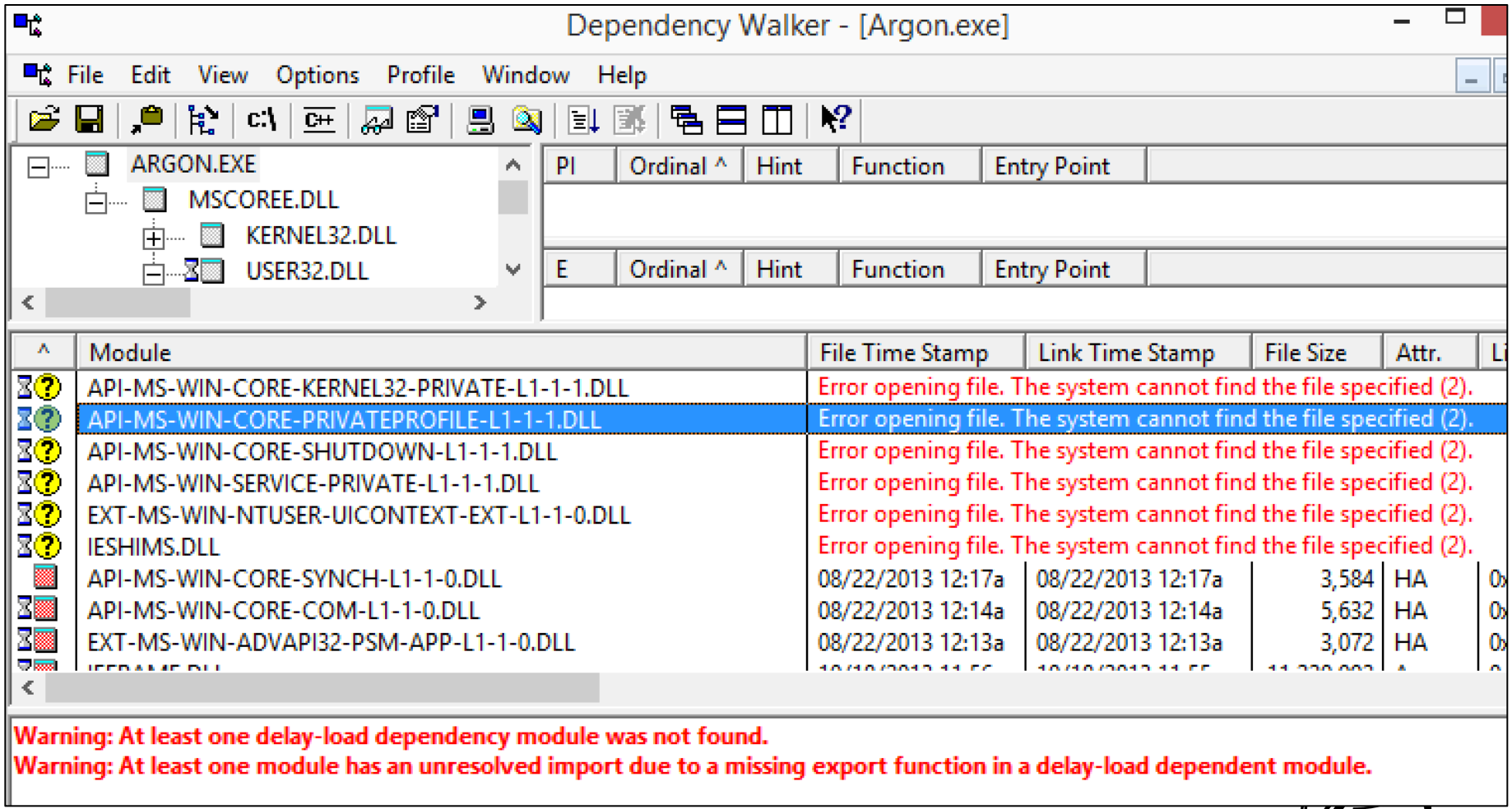
Dependency Walker

- Dependency Walker (depends.exe) recursively walks the DLLs used by an application
- This will show you the location from which DLLs are loaded into an application
- Perhaps the most interesting are those which are not found

Dependency Walker (2)

- Privilege Escalation
  - Admin tools usually run with elevated permissions
  - Check service autoruns for other auto-elevation possibilities
- Pros
  - Easily finds missing DLLs (and those not in %PATH%)
- Cons
  - Does not address all runtime DLL loading scenarios

Dependency Walker vs. Argon.exe



DLL Hijack Auditor

- The DLL Hijack Auditor from SecurityXploded offers another way to search for vulnerable programs
- Was found to not be as reliable as Dependency Walker or HDMoore Audit kit
  - But it's easier to use

DLL Hijack Auditor (2)

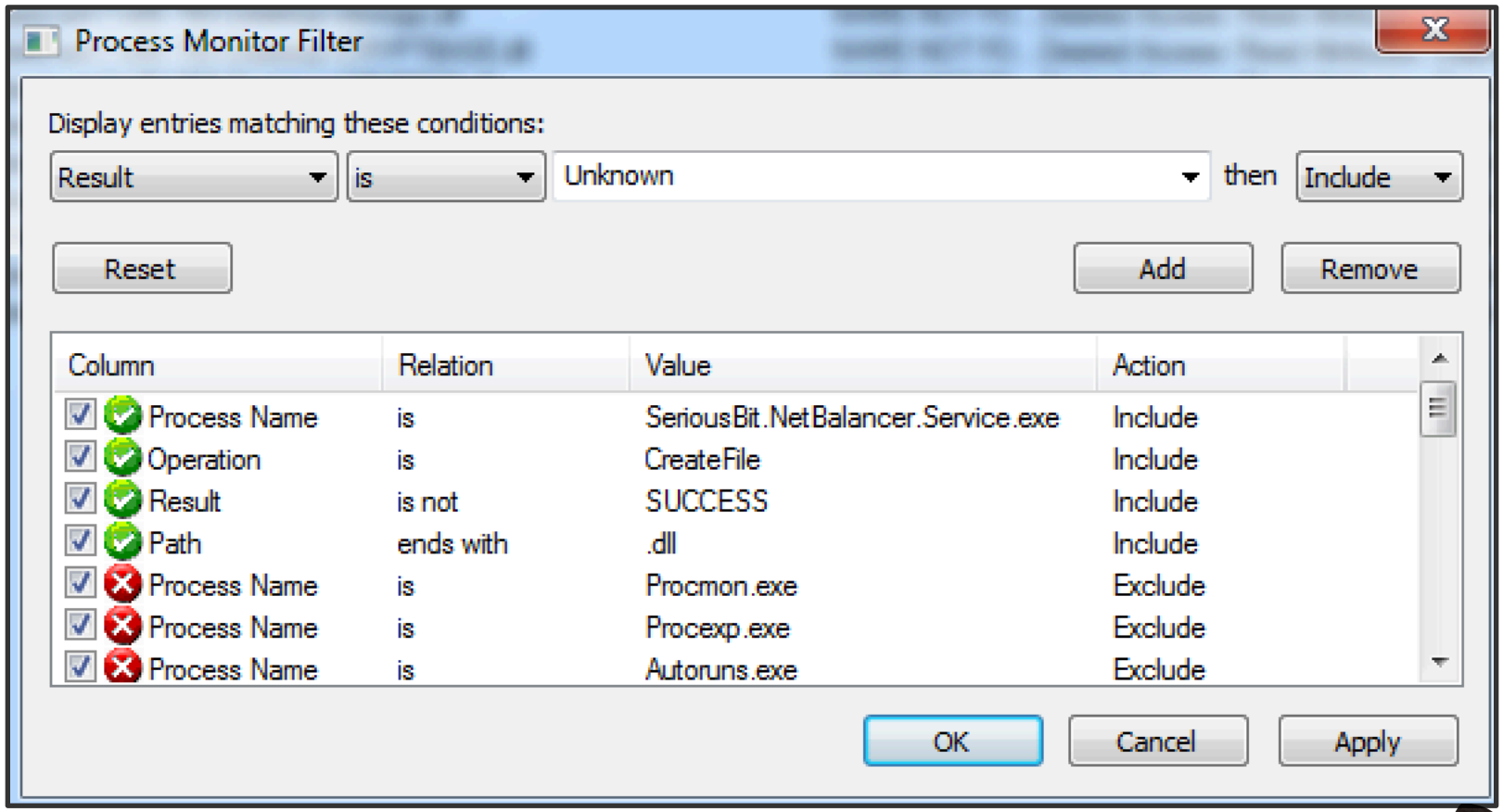
Going Native with Procmon

- Muralidharan Vadivel (opensecurityresearch.com) has a great article on auditing for “ghost dlls” using ProcMon
- Search for operation CreateFile and result of “Name not found”
- The LoadImage operation is also useful to filter for detecting missing DLLs

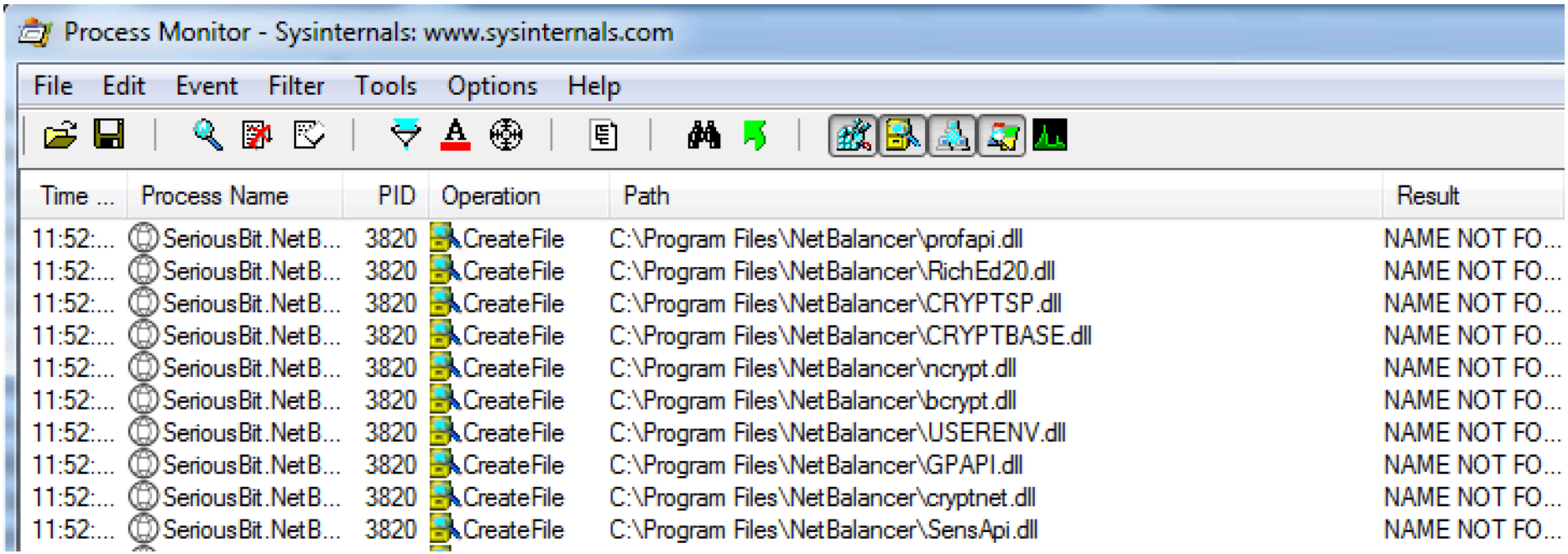
Fun With Procmon

- Let's do a quick demo here
- Fun application called NetBalancer
  - Network throttling at the granularity of individual applications
- Extra cool because it's not just an application, it runs a service

Setting Up Procmon



Oh yeah!



- Pick your poison
- This digitally signed application will load any of these DLLs for us

RENDITION INFOSEC
Security by any legal means

What's the big deal?

- Malware can persist now by loading a malicious DLL in the legitimate service
- This totally screws with forensics and IR professionals who like to look for service anomalies

Gflags for the win!

- Gflags is a tool included with the Windows Debugging Tools
- Gflags is great for finding memory errors but is also great for finding DLL issues
- The sls flag (show loader snaps) shows loader data, including dll's loaded, which directory they are loaded from, SxS redirection, and functions imported

Finding craziness with gflags

- Run glags.exe –i tested.exe +sls
- Load the app in windbg
- Save the data to a text file
- To find attempts to load DLLs, search the output for FindOrMapDependency

Debug Output

- Look for
  - LdrpFindOrMapDependency
  - LdrpFindOrMapDll
  - LdrpFindKnownDll

```
0f6c:0e84 @ 03538062 - LdrpFindOrMapDependency - ENTER: DLL name: msvcrt.dll.
0f6c:0e84 @ 03538062 - LdrpFindOrMapDll - ENTER: DLL name: msvcrt.dll
0f6c:0e84 @ 03538062 - LdrpFindKnownDll - ENTER: DLL name: msvcrt.dll
0f6c:0e84 @ 03538062 - LdrpFindKnownDll - RETURN: Status: 0x00000000
0f6c:0e84 @ 03538062 - LdrpMapViewOfSection - ENTER: DLL name: C:\Windows\system32\msvcrt.dll
ModLoad: 764c0000 76583000   C:\Windows\system32\msvcrt.dll
```

Sxstrace.exe

- If you don't want all the detailed information from a debugger and gflags but still want data from SxS loads, sxstrace (builtin) can help
- Tracks specific information about the versions of DLLs loaded from the WinSxS directory
  - And other locations where it may attempt to load a DLL from

Sxstrace in Action

```
C:\jake>sxstrace trace -logfile:tracefile.etl
Tracing started. Trace will be saved to file tracefile.etl.
Press Enter to stop tracing...


C:\jake>sxstrace parse -logfile:tracefile.etl -outfile:tracefile.txt
Parsing log file tracefile.etl...
Parsing finished! Output saved to file tracefile.txt.
```

```
INFO: Begin assembly probing.
        INFO: Did not find the assembly in WinSxS.
        INFO: Attempt to probe manifest at C:\Windows\assembly\GAC_32\Microsoft.Windows.Common-Controls\6
        INFO: Attempt to probe manifest at C:\Program Files\windows nt\Accessories\en-US\Microsoft.Windows
        INFO: Attempt to probe manifest at C:\Program Files\windows nt\Accessories\en-US\Microsoft.Windows
        INFO: Attempt to probe manifest at C:\Program Files\windows nt\Accessories\en-US\Microsoft.Windows
        INFO: Attempt to probe manifest at C:\Program Files\windows nt\Accessories\en-US\Microsoft.Windows
        INFO: Did not find manifest for culture en-US.
```

Practical Defenses

- Audit the software you install on your machines
  - Especially the stuff that your in-house “developers” create
- Don’t rely on only automated tools, they don’t catch everything
- Hire a pentester who knows more than nmap and nessus (I think that’s everyone in this room)
  - And scope the test with enough time to uncover the same bugs determined attackers will find

Practical Defenses (2)

- Always check permissions on all directories in the system %PATH% variable
  - World write permissions are bad here for other obvious reasons too
- .Net developers: be specific in your manifests
- Windows developers: call LoadLibrary with explicit path, don't rely on the system
  - Or call SetDllDirectory/AddDllDirectory

Questions?

Jake Williams

@MalwareJake

Rendition Infosec