RSAConference2020
Asia Pacific & Japan

A Virtual Learning Experience | 15–17 July

HUMAN ELEMENT

SESSION ID: SAO-W02V

Privilege Escalation and Whitelisting Bypass with Proxy DLLs

Jake Williams

Founder and Principal Consultant
Rendition Infosec
@MalwareJake

Agenda

- What is a proxy DLL?
- Why proxy DLLs?
- Building a proxy DLL
- Apply It!
- Closing Thoughts

What is a proxy DLL?

- A proxy DLL is a malicious DLL that exports the same function names as the legitimate DLL it is impersonating
- The proxy DLL forwards (proxies) API calls from the application to the legitimate DLL
- The proxy DLL also executes some malicious code, perhaps executing a callback or a bind shell

Why proxy DLLs?

- A proxy DLL allows attackers to persist malicious code on a system without changing autoruns
  - Adding a program to autoruns is likely to get you caught
  - It's a deviation from the baseline…
- Defenders are getting better at monitoring process lists
  - Running 64Z8HY.exe isn't working anymore…
- Application whitelisting is changing the game on which programs can actually start
  - For **LOTS** of reasons, most orgs don't whitelist DLLs

Proxy DLL Uses

- There are two primary uses of proxy DLLs:
  - Maintaining persistence
  - Privilege escalation

- Persistence is possible whenever we can write a DLL to a directory where a process loads from
- Privilege escalation is possible when we can add a DLL to a directory where a ***privileged*** process loads from

Obligatory DLL Search Order Slide…

- The Window loader finds DLLs by name, but the path is not specified, leaving the loader to follow a search path:
  - The directory from which the application loaded
  - The system directory
  - The 16-bit system directory
  - The Windows directory
  - The current directory
  - The directories in the PATH environment var

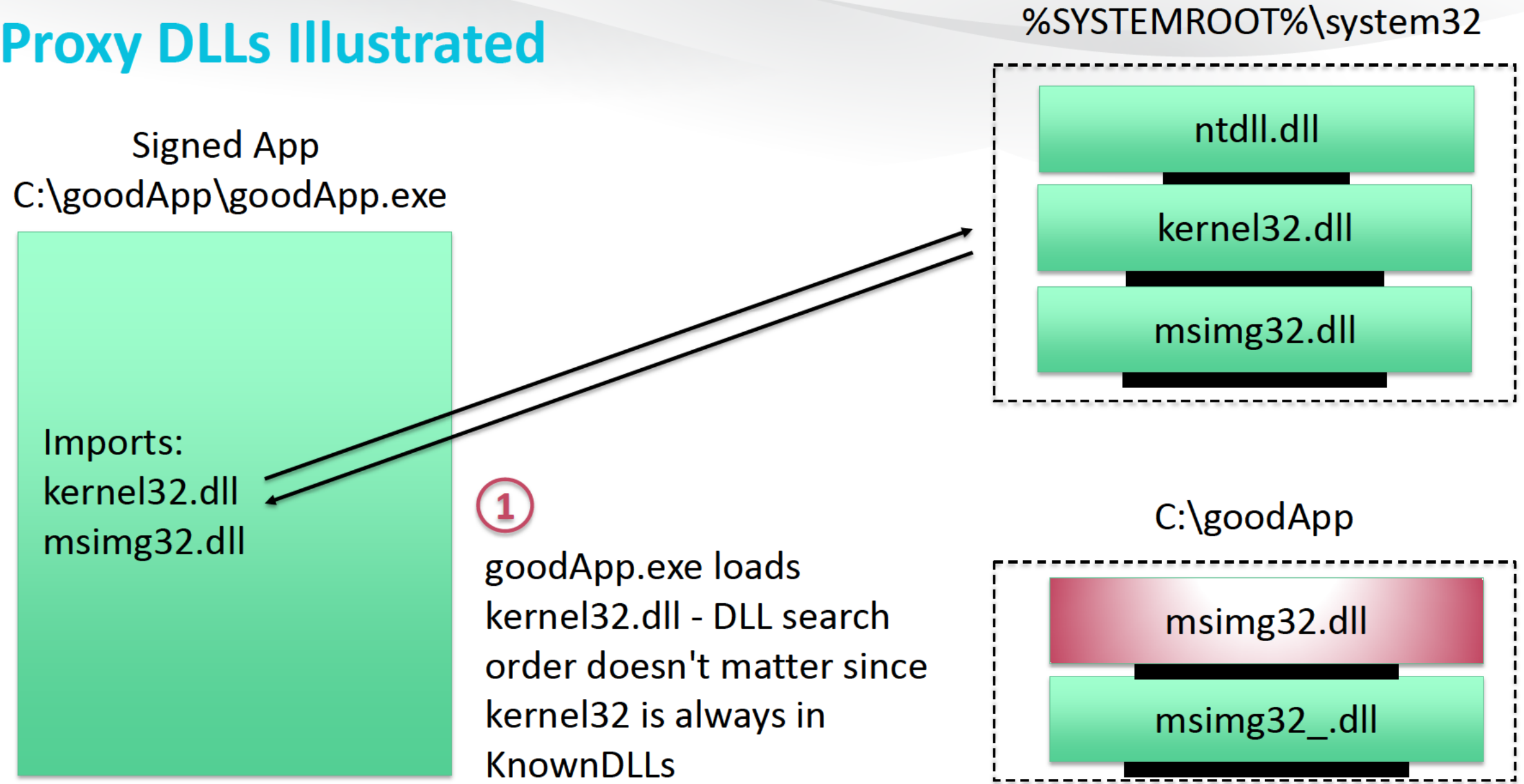
Picking The DLL To Proxy

- Find a DLL that:
  - Is loaded by our target application
  - Does not exist in the same folder as the target application
  - Is not in Known_DLLs on the target system

What Are Known DLLs?

- KnownDLLs are DLLs that are effectively preloaded
- These DLLs won't show up in the list of loaded modules
- Technically they aren't loaded, but handles to these DLLs already exist in the object manager
- Any attempt to load one of these DLLs will point to shared memory rather than load our proxy from disk
- The list of KnownDLLs changes depending on the operating system version

Proxy DLLs Illustrated



Proxy DLLs Illustrated (2)

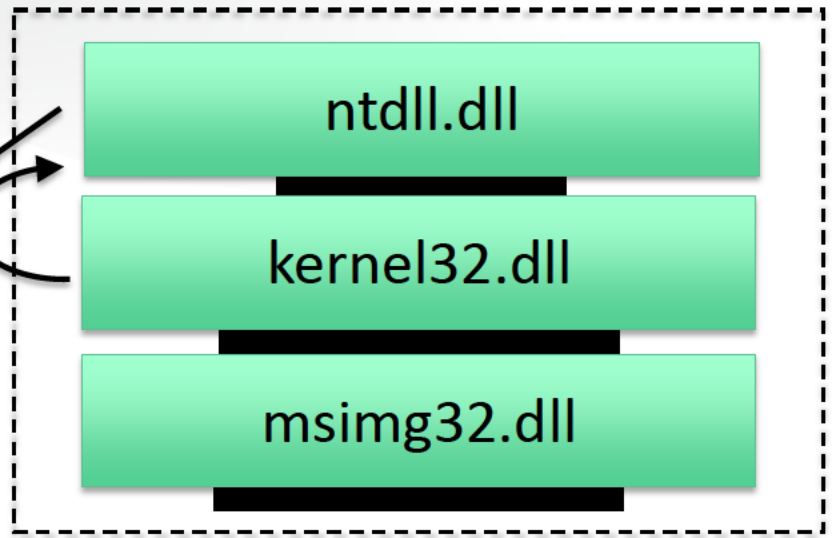
Signed App
C:\goodApp\goodApp.exe

Imports:
kernel32.dll
msimg32.dll

Implicit loads:
ntdll.dll

2

Kernel32.dll imports ntdll.dll, so it is also loaded into goodApp.exe. Because ntdll.dll is a KnownDLL, it is



Proxy DLLs Illustrated (3)

Signed App
C:\goodApp\goodApp.exe

Imports:
kernel32.dll
msimg32.dll

Implicit loads:
ntdll.dll

3

goodApp.exe loads msimg.dll - due to DLL search order, the malicious DLL of the same name is loaded

%SYSTEMROOT%\system32

ntdll.dll
kernel32.dll
msimg32.dll

C:\goodApp

msimg32.dll
msimg32_.dll

Proxy DLLs Illustrated (4)

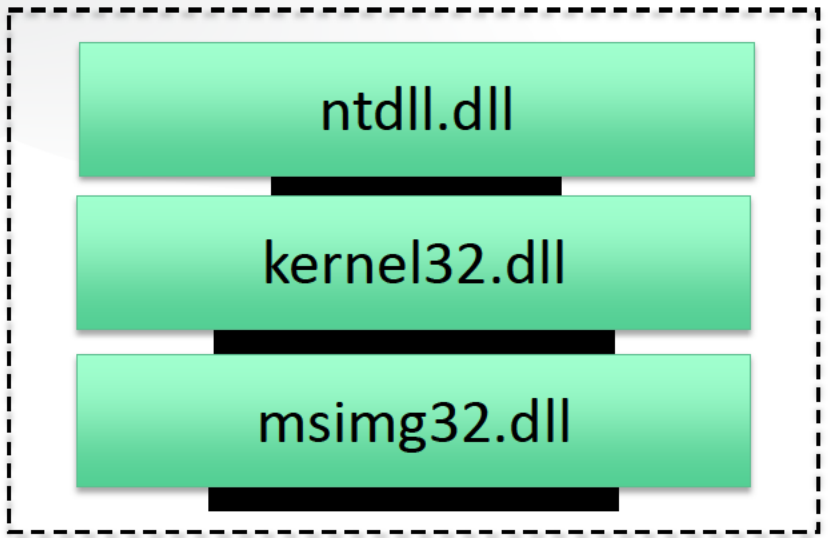
Signed App
C:\goodApp\goodApp.exe

Imports:
kernel32.dll
msimg32.dll

Implicit loads:
ntdll.dll
msimg32_.dll

4

The malicious msimg32.dll needs to load the real msimg32.dll. Windows won't allow a name collision so we need a legitimate copy of the DLL with a different name (though it need not be in our directory)

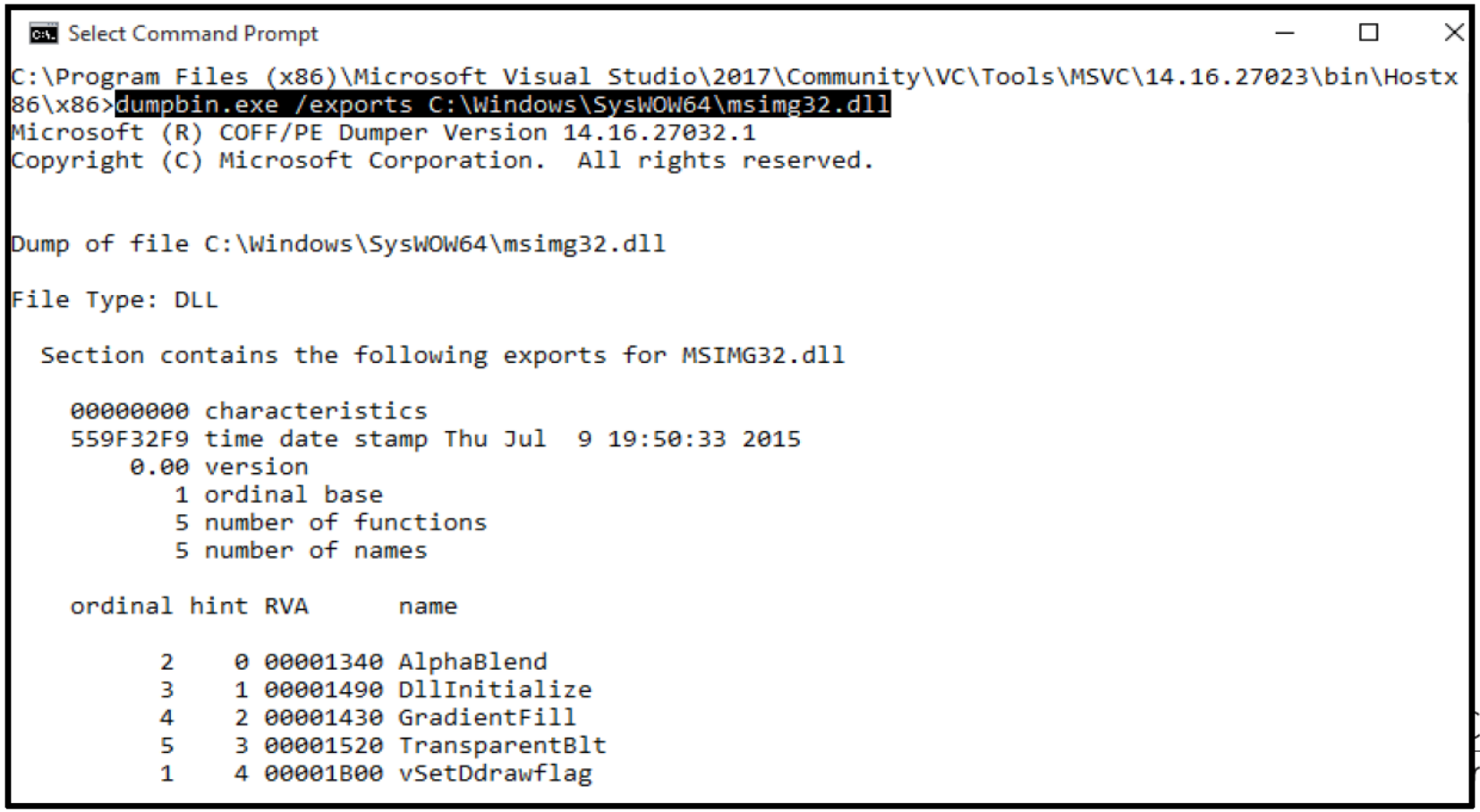


Building A Proxy DLL

- To build a proxy DLL, you need a few tools:
  - Tools to read and dump imports (dumpbin works well)
  - A compiler (I usually use Visual Studio)
  - Time and patience (or some quick Python)

Building A Proxy DLL (2)

- First, we use dumpbin to list all the exports
  - GUI tools do this too, but text output rocks…



Building A Proxy DLL (3)

- Next, we copy the export names from the output
  - These are needed to direct the linker to create proxy exports

```
    00000000 characteristics
    4A5BC263 time date stamp Mon Jul 13 16:25:23 2009
        0.00 version
           1 ordinal base
           5 number of functions
           5 number of names

    ordinal hint RVA      name

          2    0 00001210 AlphaBlend
          3    1 000010F6 DllInitialize
          4    2 00001304 GradientFill
          5    3 00001320 TransparentBlt
          1    4 0000137E vSetDdrawflag

  Summary
```

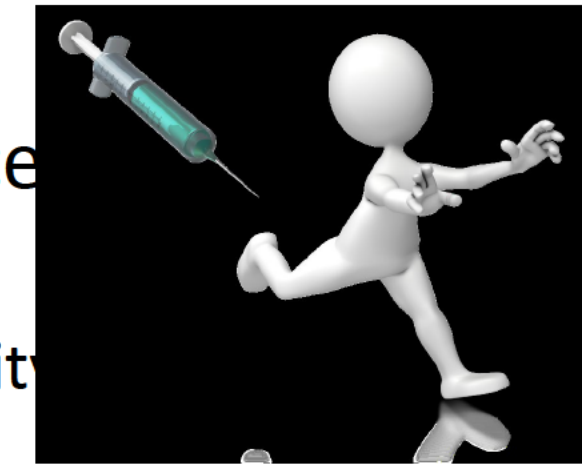
Building A Proxy DLL (4)

- Now, we create #pragma statements for all imports that are included in the DLL proxy source code
  - Note the extra 'm' in the DLL name msimmg32.dll
  - Our malicious DLL proxy DLL is named msimg32.dll, so we need to proxy to a legitimate copy of msimg32.dll with a different name
  - A #pragma statement is used to direct the linker to create an export for each required function we will proxy

```
#pragma comment(linker, "export:AlphaBlend=msimmg32.AlphaBlend")
#pragma comment(linker, "export:DllInitialize=msimmg32. DllInitialize")
```

Building A Proxy DLL (5)

- Copy the pragma statements into a DLL project in Visual Studio
- Place malicious code into a function called from DllMain
- Shellcode payloads can be created with Metasploit
  - Of course shellcode directly from Metasploit is likely to be caught by AV…
  - More advanced custom payloads can be created by hand
- Many custom payloads will inject into another proce
  - Remember - this is usually about maintaining persistence
  - Code injection into a required process enhances survivabilit



Apply What You Have Learned Today

- For red team and offensive security practitioners:
  - Examine the software loaded on your organization's golden images and look for software that auto starts with elevated privileges
    - Preferably digitally signed
  - Determine if these software packages allow for unprivileged users to add files to the directory
    - If files can be added, these are candidates for privilege escalation
    - If not, these programs are good for application whitelisting bypass

Apply What You Have Learned Today (2)

- For blue team and defensive security practitioners:
  - Follow the red team slides and hunt these same conditions
    - Or better yet, engage the red team to do it **with** you
  - Examine your tooling - can you detect an abnormal DLL loaded into a legitimate process?
    - If not, try to correct that - (don't forget dlllist from SysInternals)
  - Determine whether all signed binaries should be allowed to load per application whitelisting rules (in most cases the answer is no)
  - Examine EDR capabilities and determine if rules can be structured to detect abnormal DLL loads (**caution:** very heavy lift)

Closing Thoughts

- Proxy DLLs have been used as a persistence method in advanced attacks for years
- Because tooling is getting better, we're likely to see an increase in this attack technique
- If you haven't seen these yet, it may be that attackers are able to maintain persistence using easier methods