Practical Malware Development

x33fcon 2024

Whoami

rad[@]ish.wtf
x.com/rad9800
linkedin.com/in/onlymalware/

- Senior Security Engineer at Praetorian
  - Previously on the Red Team as Operator
    - Operator first, Developer second
  - Member of Labs team
    - Build tools to identify material risk for our clients
  - **We are hiring!**
- *Occasionally* publish/blog/tweet about malware and security

Workshop Goals

20% effort ➡ **80% results**

- ❖ Pareto Principle
  - ➢ Teach you the stuff you ACTUALLY need to know
- ❖ Give you enough information to:
  - ➢ Prepare, understand, debug your payload
  - ➢ Understand how EDRs detect malware
- ❖ <u>Host and deliver</u> payloads
- ❖ Explain payload technical details to the Blue Team

(Provide you with resources to do further exploration)

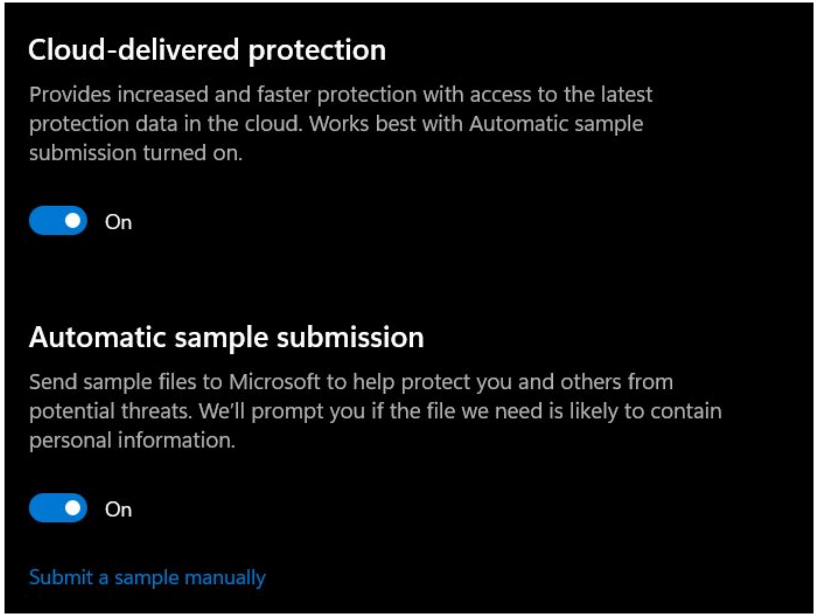
Agenda

I. Setup
II. PE Properties
III. DLL Hijacking
IV. Shellcode Loaders
V. Delivery
VI. CI/CD
VII. EDR Stuff

Section 1 - Setup

Malware Development Environment

- Base operating system - Windows
  - Run within hypervisor with limited network connectivity
  - Disable Windows Defender
- IDE - *Visual Studio 2022*
- Compiler - *MSVC*
- Debugging and Reversing
  - Procmon, Ghidra, PE Bear
- **Feel free to use whatever you are comfortable with**
  - The toolset used today was selected for <u>ease of setup</u>



Cloud-delivered protection

Provides increased and faster protection with access to the latest protection data in the cloud. Works best with Automatic sample submission turned on.

On

Automatic sample submission

Send sample files to Microsoft to help protect you and others from potential threats. We'll prompt you if the file we need is likely to contain personal information.

On

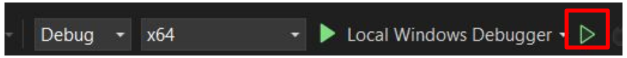
Submit a sample manually

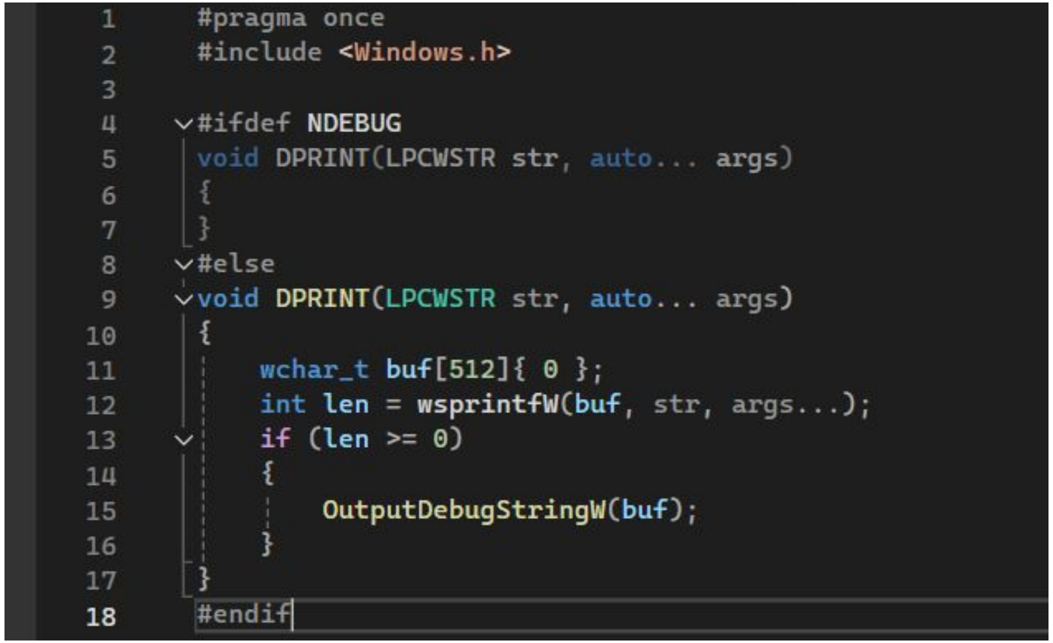
Exercise 1 - Environment Setup

<u>Solution 1</u> - Environment Setup

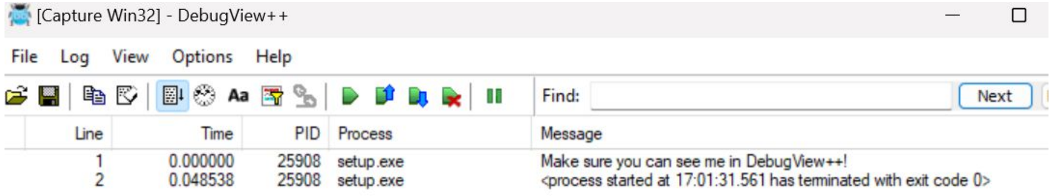
OutputDebugString

- Easy way to capture output from payload
- NDEBUG macro defined only in Debug mode





```
#pragma once
#include <Windows.h>

#ifdef NDEBUG
void DPRINT(LPCWSTR str, auto... args)
{
}
#else
void DPRINT(LPCWSTR str, auto... args)
{
    wchar_t buf[512]{ 0 };
    int len = wsprintfW(buf, str, args...);
    if (len >= 0)
    {
        OutputDebugStringW(buf);
    }
}
#endif
```



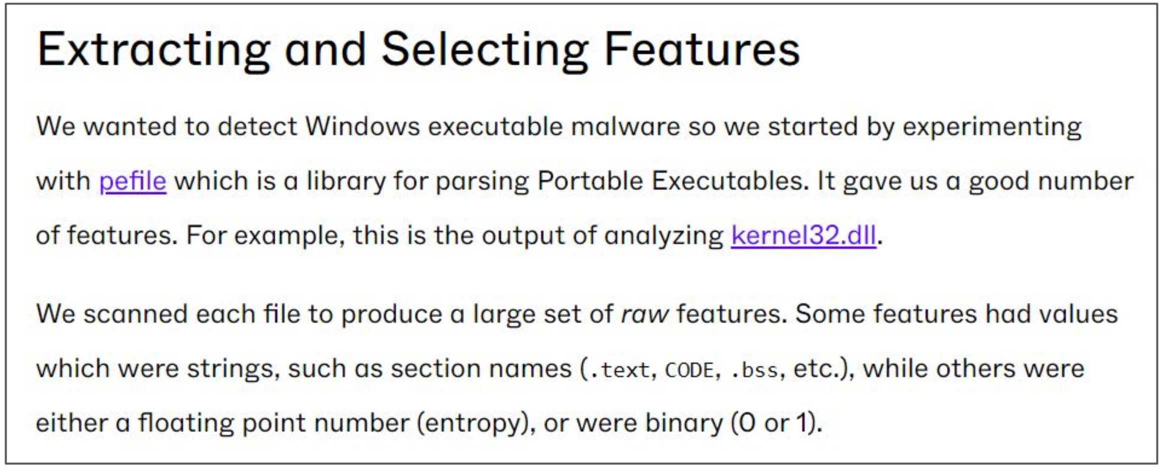
```
Line  Time      PID    Process    Message
1     0.000000  25908  setup.exe  Make sure you can see me in DebugView++!
2     0.048538  25908  setup.exe  <process started at 17:01:31.561 has terminated with exit code 0>
```

http://www.unixwiz.net/techtips/outputdebugstring.html

Section 2 - PE Properties

Investigating PE Properties

- pefile - Python module for working with PEs
- pe bear - Explore PE with nice GUI
- dumpbin - Binary File Dumper CLI tool



Extracting and Selecting Features

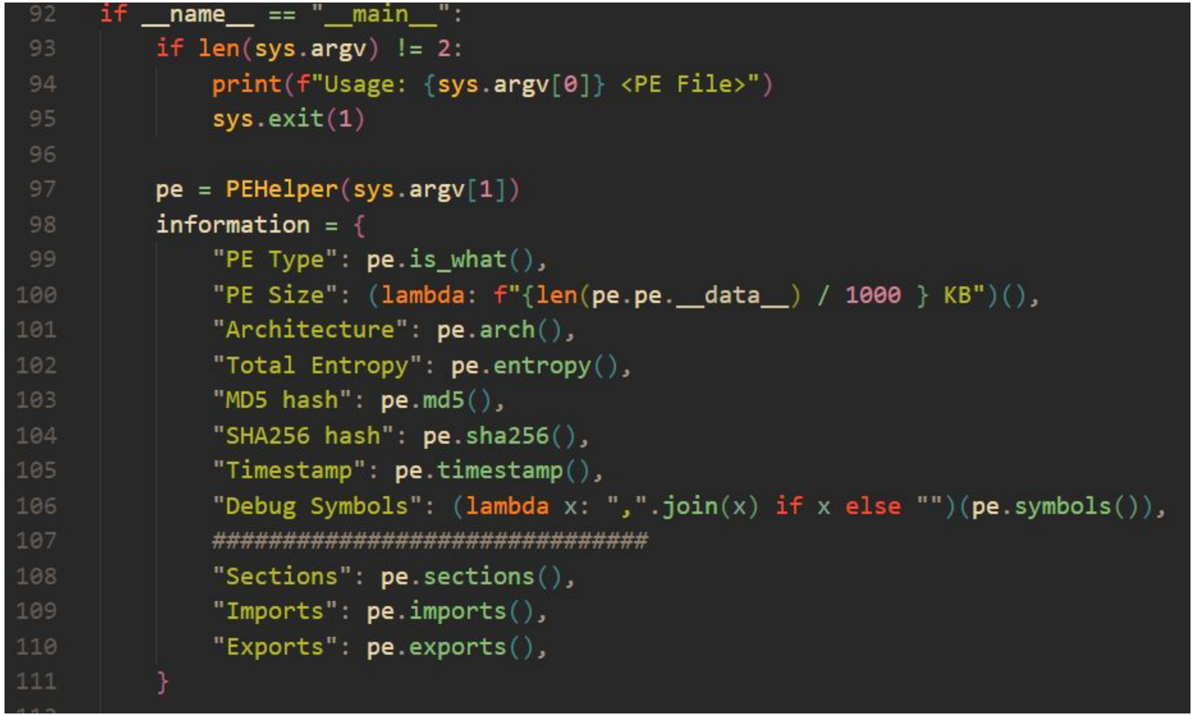
We wanted to detect Windows executable malware so we started by experimenting with pefile which is a library for parsing Portable Executables. It gave us a good number of features. For example, this is the output of analyzing kernel32.dll.

We scanned each file to produce a large set of *raw* features. Some features had values which were strings, such as section names (`.text`, `CODE`, `.bss`, etc.), while others were either a floating point number (entropy), or were binary (0 or 1).

https://www.sentinelone.com/blog/detecting-malware-pre-execution-static-analysis-machine-learning/

pefile wrapper

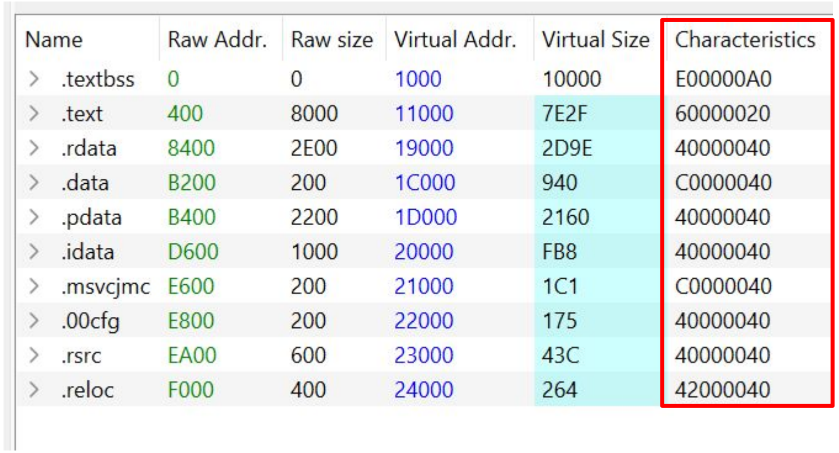
- scripts\props.py
- Dumps out all the information we'll cover in the next slides
- Can be used to check your deliverables



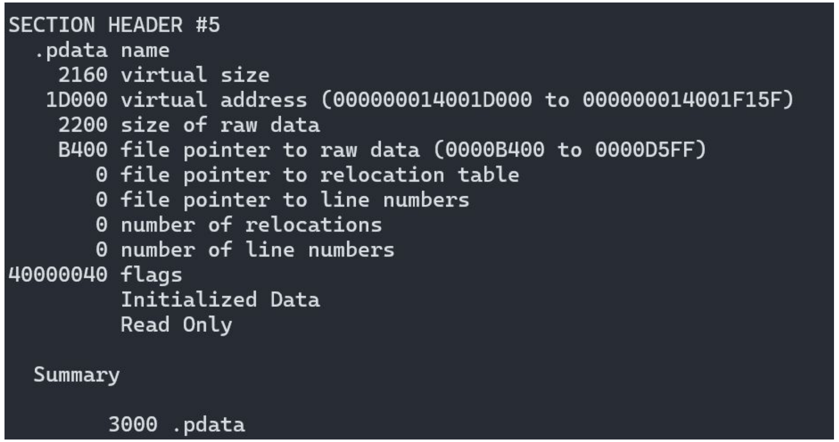
```python
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print(f"Usage: {sys.argv[0]} <PE File>")
        sys.exit(1)

    pe = PEHelper(sys.argv[1])
    information = {
        "PE Type": pe.is_what(),
        "PE Size": (lambda: f"{len(pe.pe.__data__) / 1000 } KB")(),
        "Architecture": pe.arch(),
        "Total Entropy": pe.entropy(),
        "MD5 hash": pe.md5(),
        "SHA256 hash": pe.sha256(),
        "Timestamp": pe.timestamp(),
        "Debug Symbols": (lambda x: ",".join(x) if x else "")(pe.symbols()),
        ################################
        "Sections": pe.sections(),
        "Imports": pe.imports(),
        "Exports": pe.exports(),
    }
```

Sections

- Contain either code or data
- Can be explicitly declared
- Some sections have a special meaning
  - .pdata/.idata/.reloc/.text and more…
- Sections can be given characteristics through a set of flags



Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics
.textbss	0	0	1000	10000	E00000A0
.text	400	8000	11000	7E2F	60000020
.rdata	8400	2E00	19000	2D9E	40000040
.data	B200	200	1C000	940	C0000040
.pdata	B400	2200	1D000	2160	40000040
.idata	D600	1000	20000	FB8	40000040
.msvcjmc	E600	200	21000	1C1	C0000040
.00cfg	E800	200	22000	175	40000040
.rsrc	EA00	600	23000	43C	40000040
.reloc	F000	400	24000	264	42000040



```
SECTION HEADER #5
  .pdata name
    2160 virtual size
   1D000 virtual address (000000014001D000 to 000000014001F15F)
    2200 size of raw data
    B400 file pointer to raw data (0000B400 to 0000D5FF)
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         Read Only

  Summary

        3000 .pdata
```

dumpbin /SECTION:.pdata setup.exe

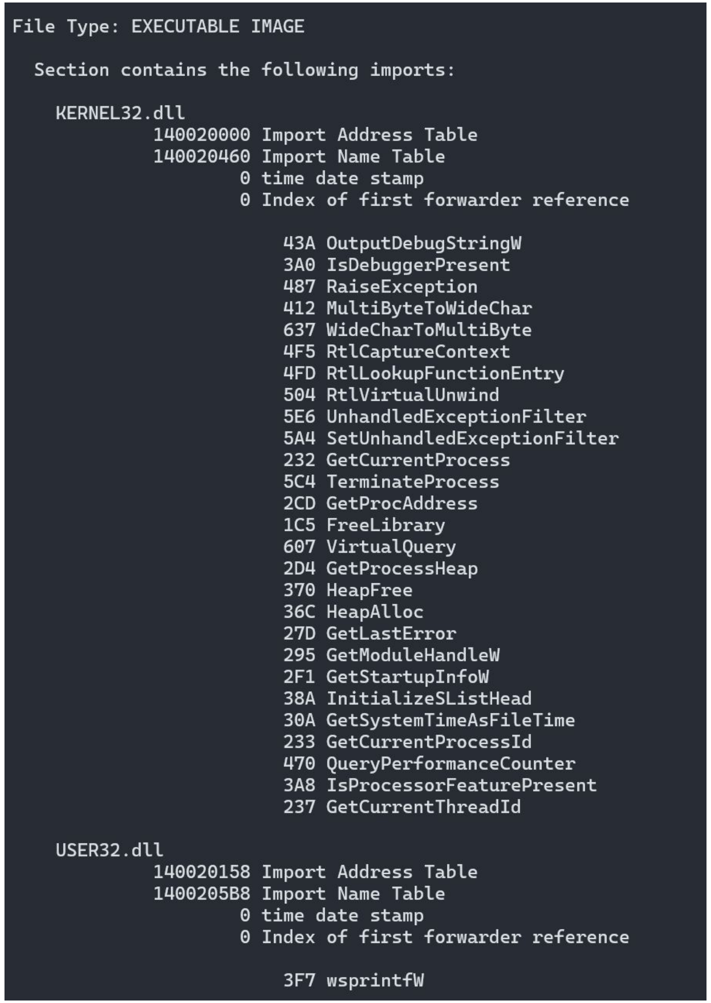
Import Address Table

- ❖ Stored in .idata section
- ❖ Array of pointers populated with addresses of imported functions
- ❖ If the loader doesn’t find a match when parsing the IAT, it will abort

---

- ❖ Import Hashing
  - ➢ Used to track malware families
- ❖ Looked at for static analysis

https://cloud.google.com/blog/topics/threat-intelligence/tracking-malware-import-hashing/

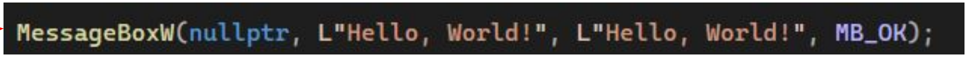


```
File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140020000 Import Address Table
             140020460 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         43A OutputDebugStringW
                         3A0 IsDebuggerPresent
                         487 RaiseException
                         412 MultiByteToWideChar
                         637 WideCharToMultiByte
                         4F5 RtlCaptureContext
                         4FD RtlLookupFunctionEntry
                         504 RtlVirtualUnwind
                         5E6 UnhandledExceptionFilter
                         5A4 SetUnhandledExceptionFilter
                         232 GetCurrentProcess
                         5C4 TerminateProcess
                         2CD GetProcAddress
                         1C5 FreeLibrary
                         607 VirtualQuery
                         2D4 GetProcessHeap
                         370 HeapFree
                         36C HeapAlloc
                         27D GetLastError
                         295 GetModuleHandleW
                         2F1 GetStartupInfoW
                         38A InitializeSListHead
                         30A GetSystemTimeAsFileTime
                         233 GetCurrentProcessId
                         470 QueryPerformanceCounter
                         3A8 IsProcessorFeaturePresent
                         237 GetCurrentThreadId

    USER32.dll
             140020158 Import Address Table
             1400205B8 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         3F7 wsprintfW
```

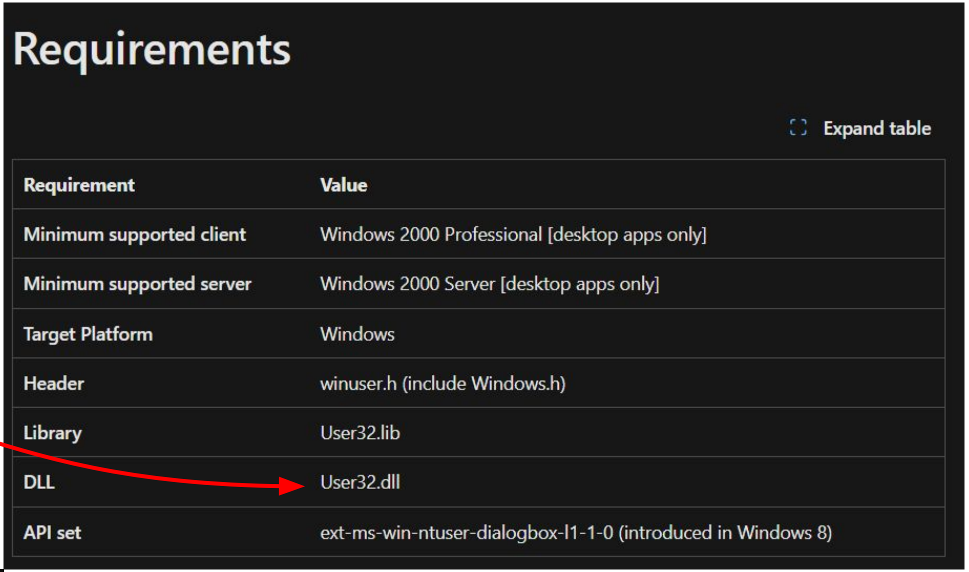
dumpbin /IMPORTS setup.exe

What gets imported?

- If we use MessageBoxW
- We can find what DLL implements this by searching msdn
- Checking our IAT again:



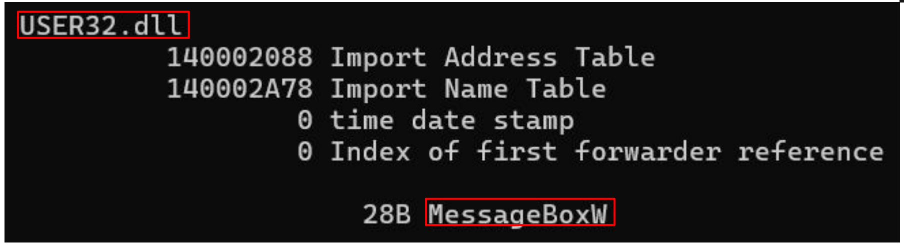
```
MessageBoxW(nullptr, L"Hello, World!", L"Hello, World!", MB_OK);
```



Requirements

Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll
API set	ext-ms-win-ntuser-dialogbox-l1-1-0 (introduced in Windows 8)



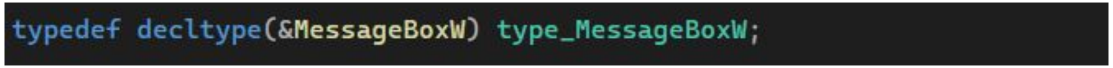
```
USER32.dll
         140002088 Import Address Table
         140002A78 Import Name Table
                 0 time date stamp
                 0 Index of first forwarder reference

                   28B MessageBoxW
```

dumpbin /IMPORTS template.exe

https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messageboxw

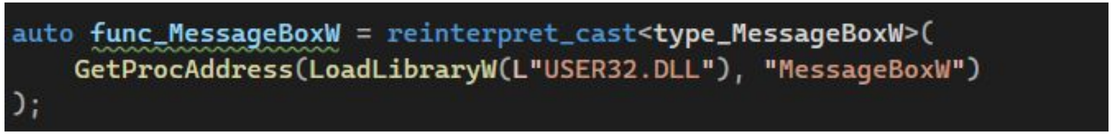
Dynamic API Resolution

- ❖ Define a new type based on the type of MessageBoxW



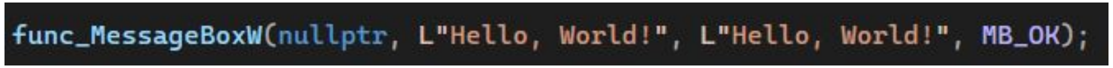
```cpp
typedef decltype(&MessageBoxW) type_MessageBoxW;
```

- ❖ Load User32.dll into our process
  a. Not a dependency
  b. Not guaranteed to be loaded
- ❖ Cast the address of MessageBoxW to our defined type



```cpp
auto func_MessageBoxW = reinterpret_cast<type_MessageBoxW>(
    GetProcAddress(LoadLibraryW(L"USER32.DLL"), "MessageBoxW")
);
```

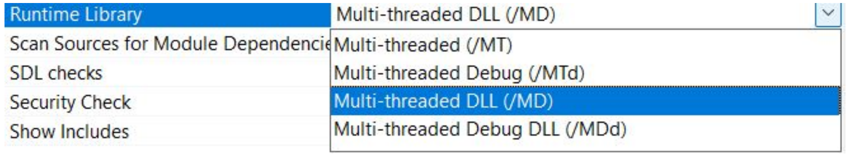
- ❖ Call it



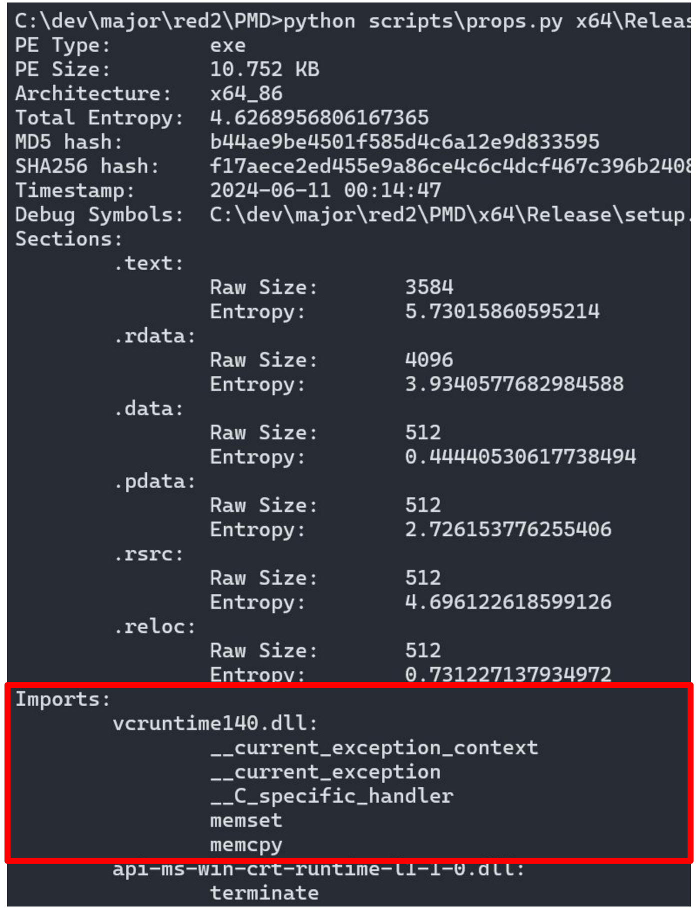
```cpp
func_MessageBoxW(nullptr, L"Hello, World!", L"Hello, World!", MB_OK);
```

CRT - C runtime

- You can link a PE to the CRT
- However, the target runtime <u>is not guaranteed</u> to be installed on the target machine
- <u>**You do not want this as a dependency in your final deliverable**</u>
- Configurable compiler option

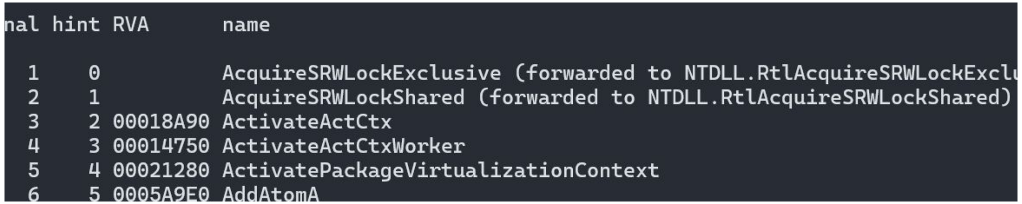


```
Runtime Library                          Multi-threaded DLL (/MD)
Scan Sources for Module Dependencie      Multi-threaded (/MT)
SDL checks                               Multi-threaded Debug (/MTd)
Security Check                           Multi-threaded DLL (/MD)
Show Includes                            Multi-threaded Debug DLL (/MDd)
```



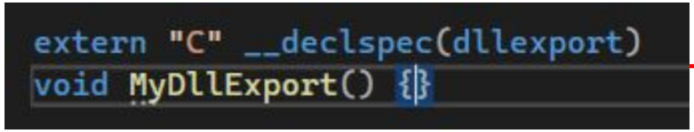
```
C:\dev\major\red2\PMD>python scripts\props.py x64\Releas
PE Type:        exe
PE Size:        10.752 KB
Architecture:   x64_86
Total Entropy:  4.6268956806167365
MD5 hash:       b44ae9be4501f585d4c6a12e9d833595
SHA256 hash:    f17aece2ed455e9a86ce4c6c4dcf467c396b2408
Timestamp:      2024-06-11 00:14:47
Debug Symbols:  C:\dev\major\red2\PMD\x64\Release\setup.
Sections:
        .text:
                Raw Size:       3584
                Entropy:        5.73015860595214
        .rdata:
                Raw Size:       4096
                Entropy:        3.9340577682984588
        .data:
                Raw Size:       512
                Entropy:        0.44440530617738494
        .pdata:
                Raw Size:       512
                Entropy:        2.726153776255406
        .rsrc:
                Raw Size:       512
                Entropy:        4.696122618599126
        .reloc:
                Raw Size:       512
                Entropy:        0.731227137934972
Imports:
        vcruntime140.dll:
                __current_exception_context
                __current_exception
                __C_specific_handler
                memset
                memcpy
        api-ms-win-crt-runtime-l1-1-0.dll:
                terminate
```

Export Address Table



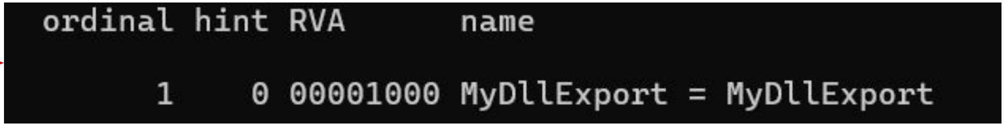
```
nal hint RVA      name

  1    0          AcquireSRWLockExclusive (forwarded to NTDLL.RtlAcquireSRWLockExclu
  2    1          AcquireSRWLockShared (forwarded to NTDLL.RtlAcquireSRWLockShared)
  3    2 00018A90 ActivateActCtx
  4    3 00014750 ActivateActCtxWorker
  5    4 00021280 ActivatePackageVirtualizationContext
  6    5 0005A9E0 AddAtomA
```

- ❖ Stored in .edata
- ❖ Contains
  - ➢ exported table of function names
  - ➢ entry point addresses
    - ■ Export RVA - Relative address of symbol when loaded
    - ■ Forwarder RVA - e.g. DLLNAME.funcname



```
extern "C" __declspec(dllexport)
void MyDllExport() {}
```

Exporting a function

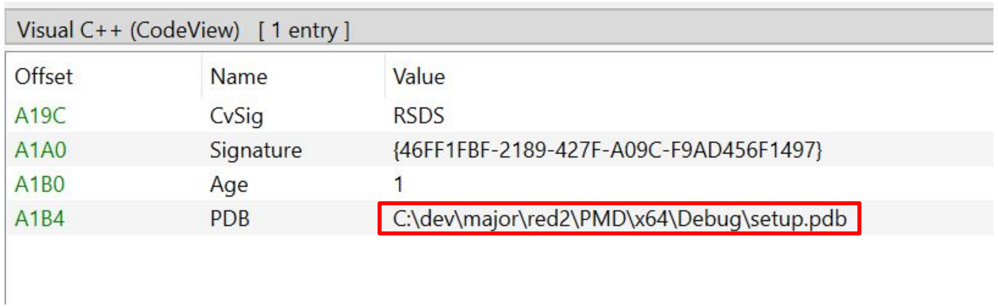


```
ordinal hint RVA      name

      1    0 00001000 MyDllExport = MyDllExport
```

dumpbin /EXPORTS dll_template.dll

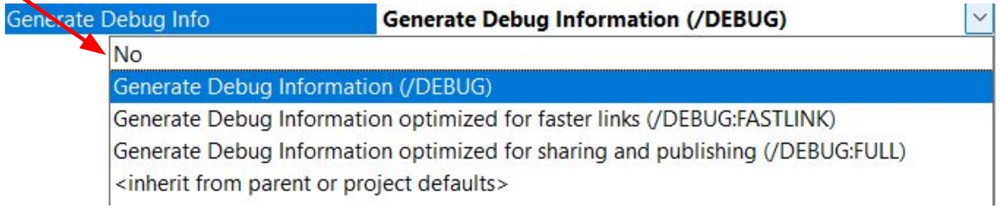
Debug Information

- Leaving debug information may often reveal that this is a red team
  - E.g. path contains your full name or other dead giveaway information
  - Luckily, can be disabled with compiler option
- Both properties can be edited



Visual C++ (CodeView) [1 entry]

Offset	Name	Value
A19C	CvSig	RSDS
A1A0	Signature	{46FF1FBF-2189-427F-A09C-F9AD456F1497}
A1B0	Age	1
A1B4	PDB	C:\dev\major\red2\PMD\x64\Debug\setup.pdb

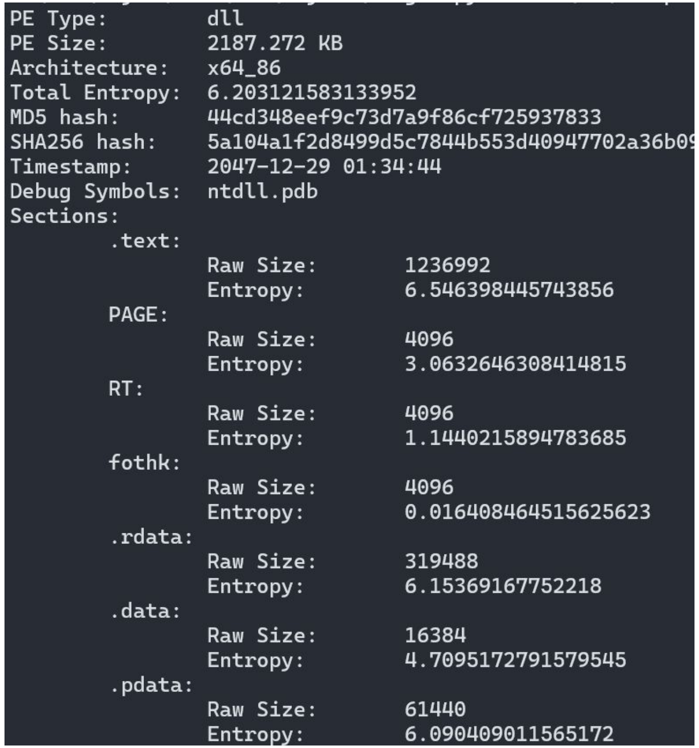


Generate Debug Info: **Generate Debug Information (/DEBUG)**

- No
- Generate Debug Information (/DEBUG)
- Generate Debug Information optimized for faster links (/DEBUG:FASTLINK)
- Generate Debug Information optimized for sharing and publishing (/DEBUG:FULL)
- <inherit from parent or project defaults>

Entropy

- ❖ Measure of randomness
- ❖ Encryption/Compression naturally increases entropy
- ❖ Good indicator of whether something is “packed”
- ❖ EDRs will look at both:
  - ➢ individual sections (and the size of the section)
  - ➢ overall executable
- ❖ E.g. a high entropy AND large section may be a good indicator of something suspicious



```
PE Type:         dll
PE Size:         2187.272 KB
Architecture:    x64_86
Total Entropy:   6.203121583133952
MD5 hash:        44cd348eef9c73d7a9f86cf725937833
SHA256 hash:     5a104a1f2d8499d5c7844b553d40947702a36b0
Timestamp:       2047-12-29 01:34:44
Debug Symbols:   ntdll.pdb
Sections:
        .text:
                 Raw Size:        1236992
                 Entropy:         6.546398445743856
        PAGE:
                 Raw Size:        4096
                 Entropy:         3.0632646308414815
        RT:
                 Raw Size:        4096
                 Entropy:         1.1440215894783685
        fothk:
                 Raw Size:        4096
                 Entropy:         0.016408464515625623
        .rdata:
                 Raw Size:        319488
                 Entropy:         6.15369167752218
        .data:
                 Raw Size:        16384
                 Entropy:         4.7095172791579545
        .pdata:
                 Raw Size:        61440
                 Entropy:         6.090409011565172
```

scripts\props.py ntdll.dll

File Size

- EDRs and AV scanners often place a limit on the maximum file size they will scan
  - Larger => Takes longer to scan
  - Thus causes a slow, unusable system
- Sandboxes may also limit file sizes or restrict access behind a paywall
- If done right, PITA for malware analysts

- Leverage a language that compiles to large binary sizes
  - E.g. rust, golang
- Or store garbage content in your sections

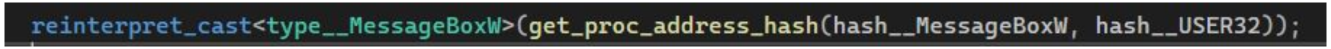
- Network bandwidth is no longer an issue for most orgs

Exercise 2 - Understanding PE Properties

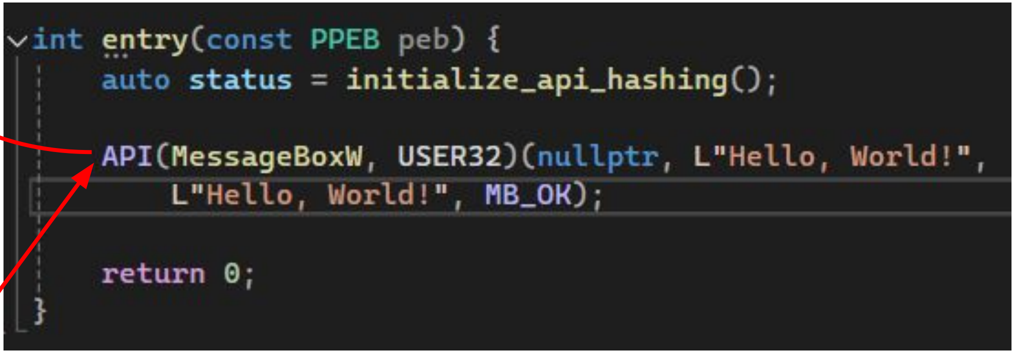
Solution 2 - Understanding PE Properties

Compile Time API Hashing Library

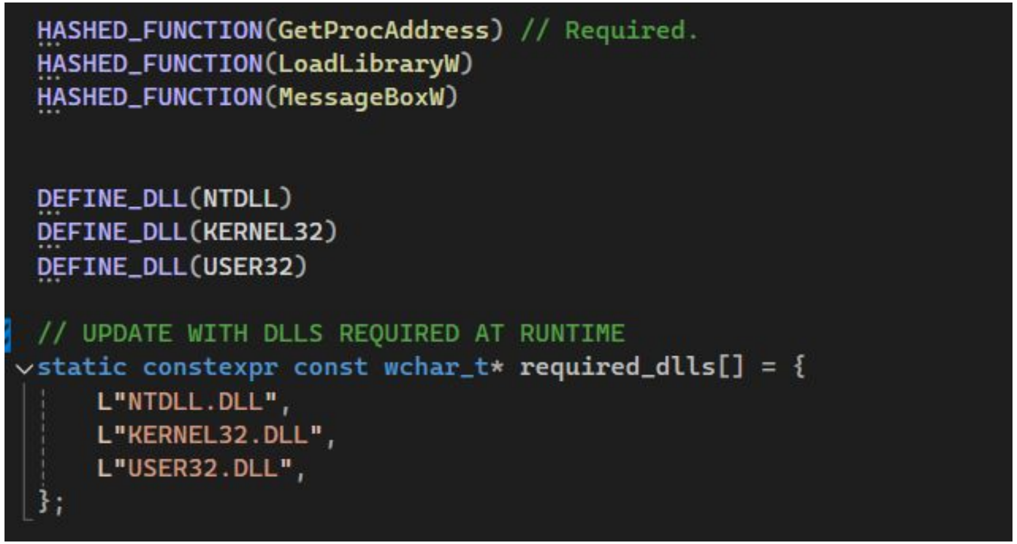
- Requires C++20
- No strings of function names
- Header only
- No CRT dependencies
- Compile time hashes
  - Update hashing algorithm quickly
- Nice helper macros for
  - Invocation `API()()`
  - Easy to define new DLLs/functions



```
reinterpret_cast<type__MessageBoxW>(get_proc_address_hash(hash__MessageBoxW, hash__USER32));
```



```
int entry(const PPEB peb) {
    auto status = initialize_api_hashing();

    API(MessageBoxW, USER32)(nullptr, L"Hello, World!",
        L"Hello, World!", MB_OK);

    return 0;
}
```



```
HASHED_FUNCTION(GetProcAddress) // Required.
HASHED_FUNCTION(LoadLibraryW)
HASHED_FUNCTION(MessageBoxW)


DEFINE_DLL(NTDLL)
DEFINE_DLL(KERNEL32)
DEFINE_DLL(USER32)

// UPDATE WITH DLLS REQUIRED AT RUNTIME
static constexpr const wchar_t* required_dlls[] = {
    L"NTDLL.DLL",
    L"KERNEL32.DLL",
    L"USER32.DLL",
};
```

Exercise 3 - Compile Time API Hashing

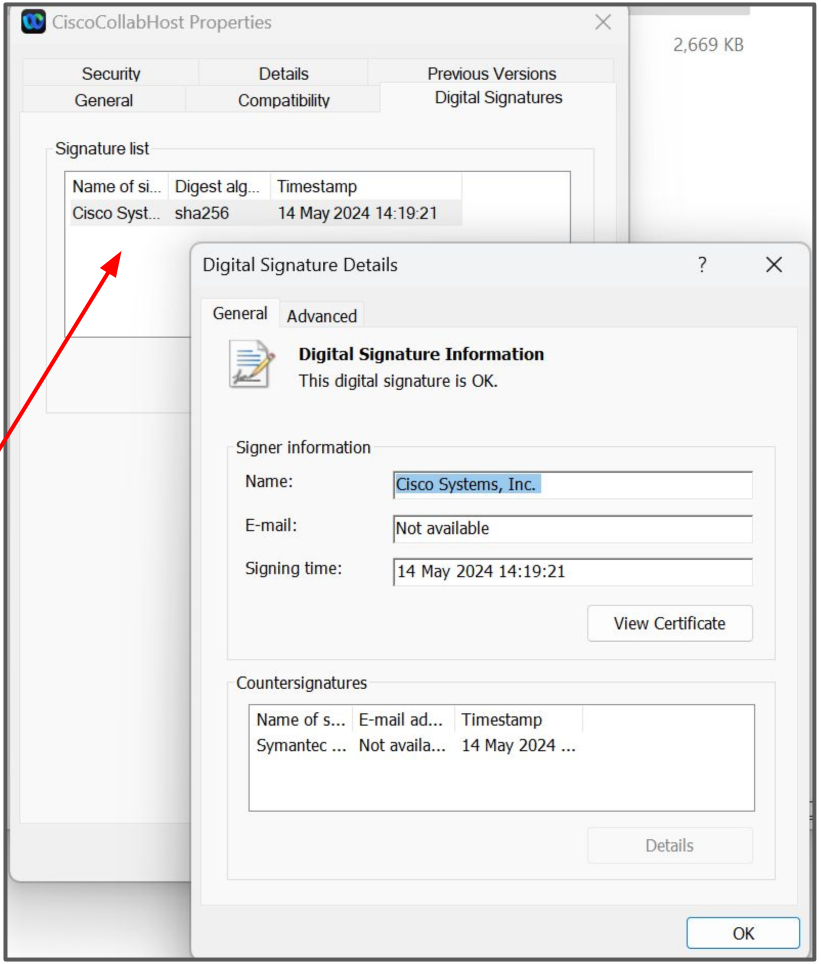
Solution 3 - Compile Time API Hashing

Section 3 - DLL Hijacking

Why?

- Can bundle for Initial Access
- Run our code within signed process
- Face less scrutiny by EDR

- Can leverage software provider for lure
- Uneducated malware analysts may misinterpret alert and give you a free pass


CiscoCollabHost Properties

2,669 KB

Security | Details | Previous Versions
General | Compatibility | Digital Signatures

Signature list

Name of si...	Digest alg...	Timestamp
Cisco Syst...	sha256	14 May 2024 14:19:21

Digital Signature Details

General | Advanced

Digital Signature Information
This digital signature is OK.

Signer information

Name:	Cisco Systems, Inc.
E-mail:	Not available
Signing time:	14 May 2024 14:19:21

View Certificate

Countersignatures

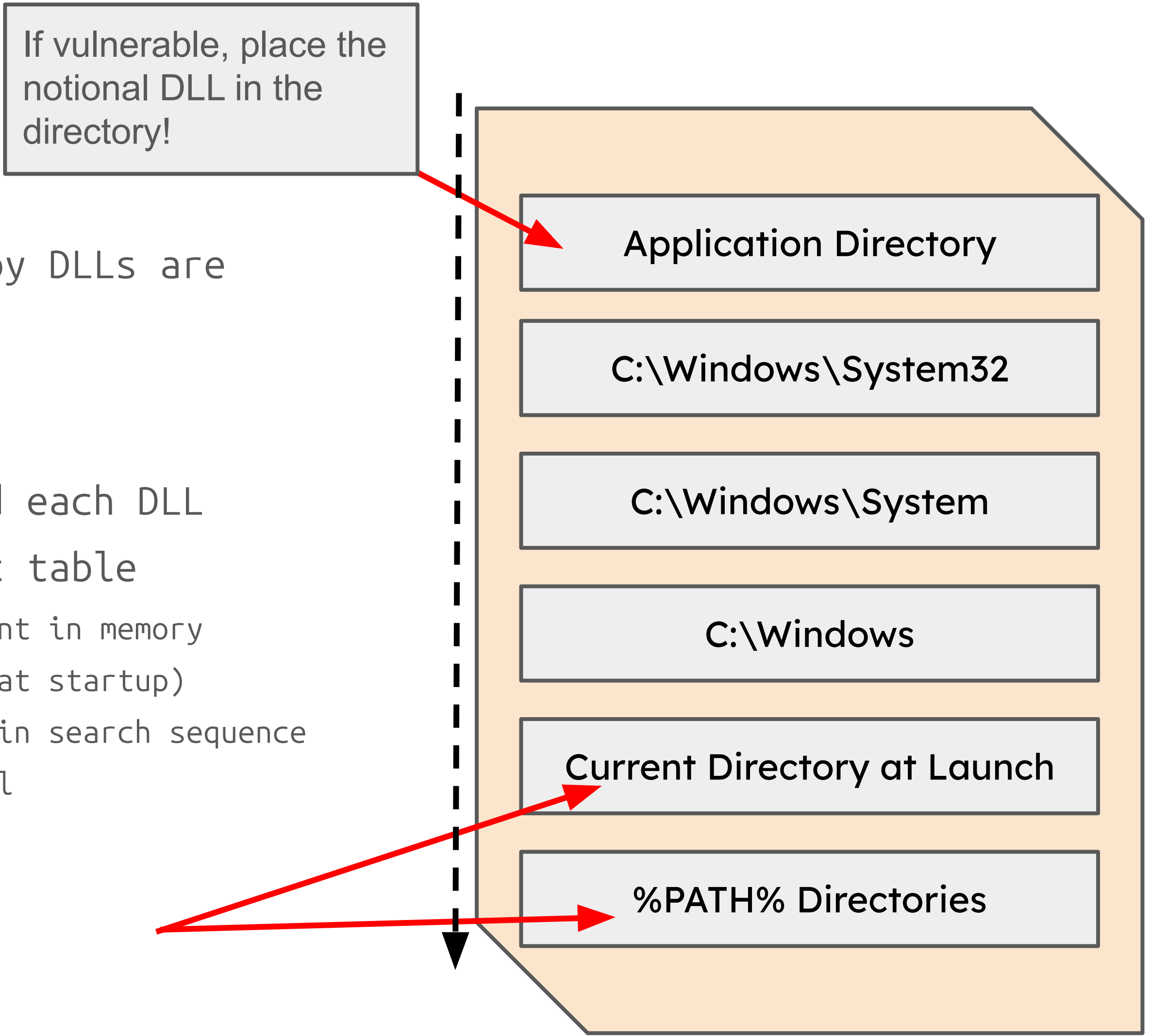
Name of s...	E-mail ad...	Timestamp
Symantec ...	Not availa...	14 May 2024 ...

Details

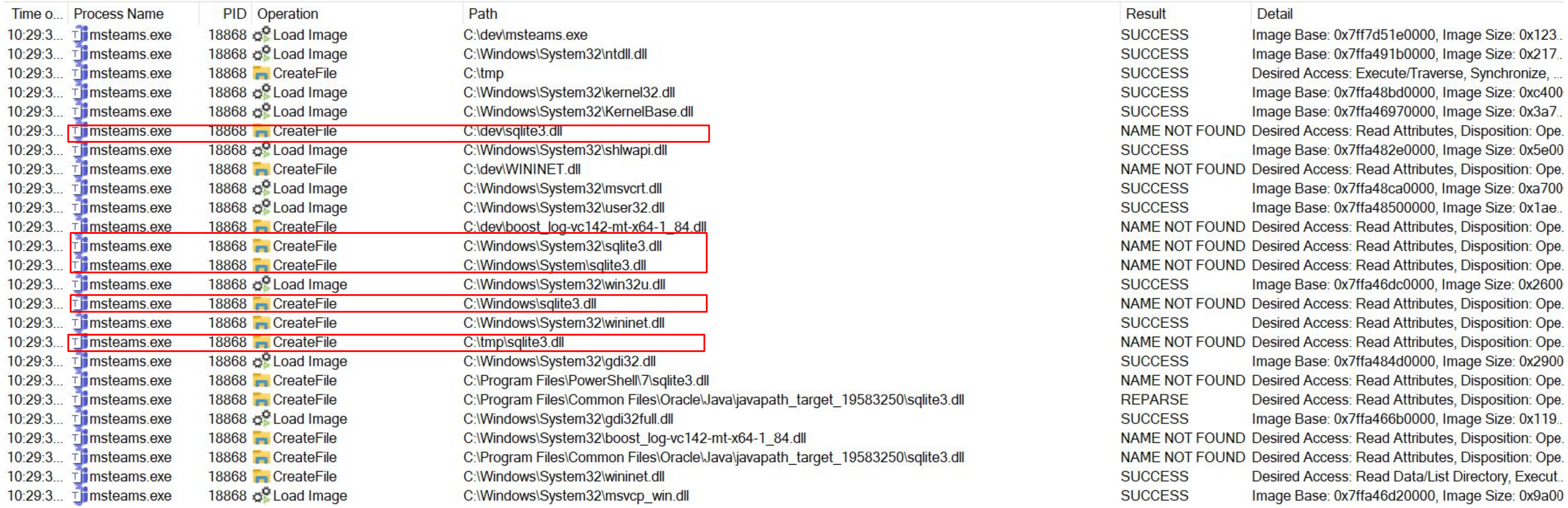
OK

DLL Search Order

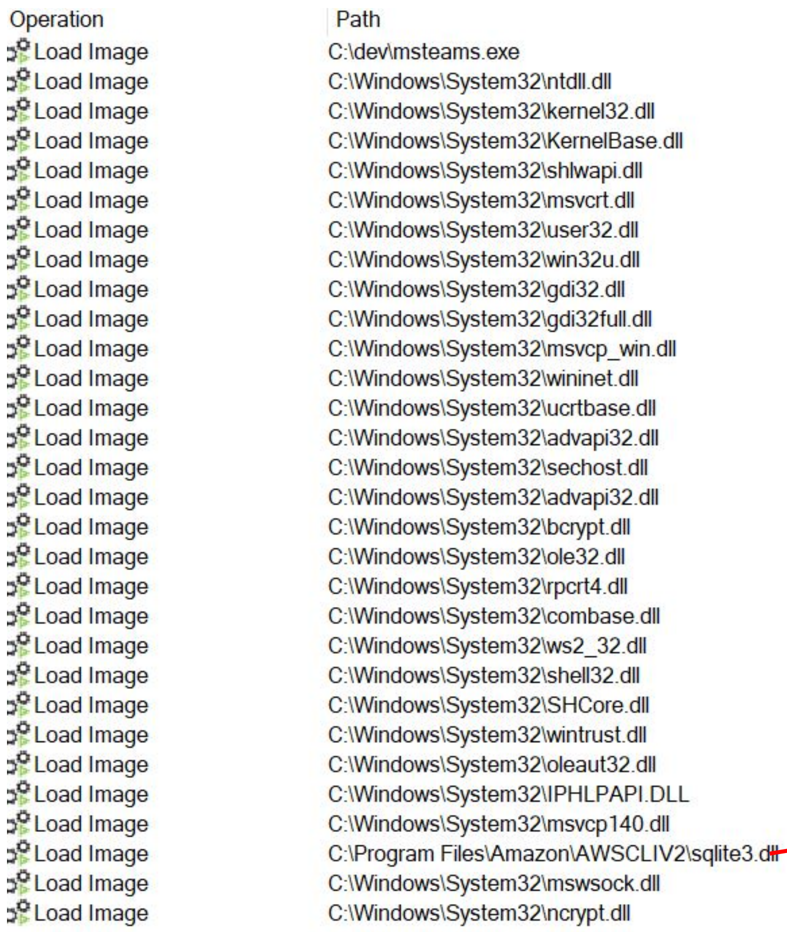
- Functionality exported by DLLs are either required at:
  - Link Time in the IAT
  - Runtime with LoadLibrary…
- Windows loader will load each DLL referenced in the import table
  - First check whether present in memory
  - Or known DLL (pre-loaded at startup)
  - Else probe each location in search sequence until DLL is found OR fail



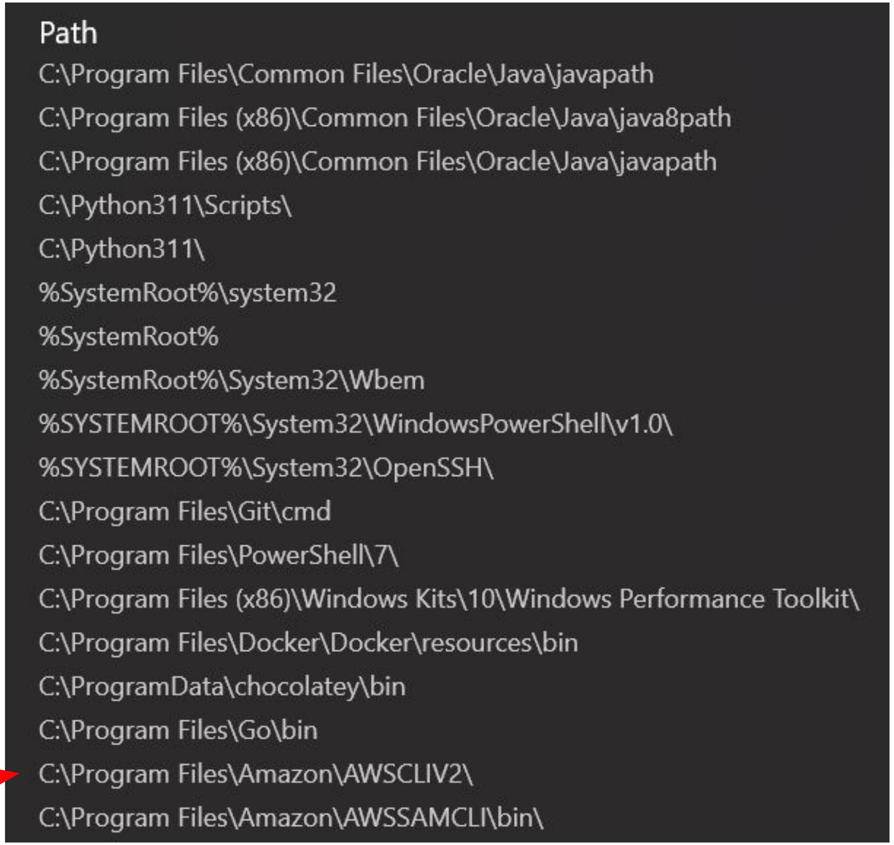
DLL Search Order in Action



Time o...	Process Name	PID	Operation	Path	Result	Detail
10:29:3...	msteams.exe	18868	Load Image	C:\dev\msteams.exe	SUCCESS	Image Base: 0x7ff7d51e0000, Image Size: 0x123..
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ffa491b0000, Image Size: 0x217..
10:29:3...	msteams.exe	18868	CreateFile	C:\tmp	SUCCESS	Desired Access: Execute/Traverse, Synchronize, ...
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7ffa48bd0000, Image Size: 0xc400
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\KernelBase.dll	SUCCESS	Image Base: 0x7ffa46970000, Image Size: 0x3a7..
10:29:3...	msteams.exe	18868	CreateFile	C:\dev\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\shlwapi.dll	SUCCESS	Image Base: 0x7ffa482e0000, Image Size: 0x5e00
10:29:3...	msteams.exe	18868	CreateFile	C:\dev\WININET.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\msvcrt.dll	SUCCESS	Image Base: 0x7ffa48ca0000, Image Size: 0xa700
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\user32.dll	SUCCESS	Image Base: 0x7ffa48500000, Image Size: 0x1ae..
10:29:3...	msteams.exe	18868	CreateFile	C:\dev\boost_log-vc142-mt-x64-1_84.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\System32\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\System\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\win32u.dll	SUCCESS	Image Base: 0x7ffa46dc0000, Image Size: 0x2600
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\System32\wininet.dll	SUCCESS	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\tmp\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\gdi32.dll	SUCCESS	Image Base: 0x7ffa484d0000, Image Size: 0x2900
10:29:3...	msteams.exe	18868	CreateFile	C:\Program Files\PowerShell\7\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Program Files\Common Files\Oracle\Java\javapath_target_19583250\sqlite3.dll	REPARSE	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\gdi32full.dll	SUCCESS	Image Base: 0x7ffa466b0000, Image Size: 0x119..
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\System32\boost_log-vc142-mt-x64-1_84.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Program Files\Common Files\Oracle\Java\javapath_target_19583250\sqlite3.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Ope.
10:29:3...	msteams.exe	18868	CreateFile	C:\Windows\System32\wininet.dll	SUCCESS	Desired Access: Read Data/List Directory, Execut..
10:29:3...	msteams.exe	18868	Load Image	C:\Windows\System32\msvcp_win.dll	SUCCESS	Image Base: 0x7ffa46d20000, Image Size: 0x9a00



Operation	Path
Load Image	C:\dev\msteams.exe
Load Image	C:\Windows\System32\ntdll.dll
Load Image	C:\Windows\System32\kernel32.dll
Load Image	C:\Windows\System32\KernelBase.dll
Load Image	C:\Windows\System32\shlwapi.dll
Load Image	C:\Windows\System32\msvcrt.dll
Load Image	C:\Windows\System32\user32.dll
Load Image	C:\Windows\System32\win32u.dll
Load Image	C:\Windows\System32\gdi32.dll
Load Image	C:\Windows\System32\gdi32full.dll
Load Image	C:\Windows\System32\msvcp_win.dll
Load Image	C:\Windows\System32\wininet.dll
Load Image	C:\Windows\System32\ucrtbase.dll
Load Image	C:\Windows\System32\advapi32.dll
Load Image	C:\Windows\System32\sechost.dll
Load Image	C:\Windows\System32\advapi32.dll
Load Image	C:\Windows\System32\bcrypt.dll
Load Image	C:\Windows\System32\ole32.dll
Load Image	C:\Windows\System32\rpcrt4.dll
Load Image	C:\Windows\System32\combase.dll
Load Image	C:\Windows\System32\ws2_32.dll
Load Image	C:\Windows\System32\shell32.dll
Load Image	C:\Windows\System32\SHCore.dll
Load Image	C:\Windows\System32\wintrust.dll
Load Image	C:\Windows\System32\oleaut32.dll
Load Image	C:\Windows\System32\IPHLPAPI.DLL
Load Image	C:\Windows\System32\msvcp140.dll
Load Image	C:\Program Files\Amazon\AWSCLIV2\sqlite3.dll
Load Image	C:\Windows\System32\mswsock.dll
Load Image	C:\Windows\System32\ncrypt.dll



```
Path
C:\Program Files\Common Files\Oracle\Java\javapath
C:\Program Files (x86)\Common Files\Oracle\Java\java8path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
C:\Python311\Scripts\
C:\Python311\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files\PowerShell\7\
C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\
C:\Program Files\Docker\Docker\resources\bin
C:\ProgramData\chocolatey\bin
C:\Program Files\Go\bin
C:\Program Files\Amazon\AWSCLIV2\
C:\Program Files\Amazon\AWSSAMCLI\bin\
```

Loading DLL from %PATH% directories

Finding DLL Sideloading Opportunities

- Download a lot of software
  - E.g. printer software or other hardware applications

1. Find where it’s installed
2. Look at the imports/exports of the PEs/DLLs
3. Run the signed EXEs in their their own folder
4. See what DLLs are required either at link/compile time
   - Leverage ProcMon
   - https://github.com/sadreck/Spartacus
5. If interesting, throw it in Ghidra
   - where is the DLL being loaded and why
   - Ensure undesired behaviour is not occuring

everything - https://www.voidtools.com/

What to avoid?

- Avoid sideloading Microsoft binaries:
  - vcruntime140.dll
  - version.dll
- Hard for vendors to profile the million of signed software packages and DLLs that they load
- Not following DLL Best Practices
  - https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-best-practices
  - Can cause instability => crash
- Having too many 3rd party dependencies
  - No idea what these DLLs are actually doing in the background
  - May be undesired behaviour (e.g. network requests/file system interaction/process exit)

Exercise 4 - Find and Leverage DLL Hijacking

Solution 4 - Find and Leverage DLL Hijacking

Section 5 - Shellcode Loaders

Storing Shellcode

- Encode it
- Encrypt it
  - Hardcode symmetric key
  - Use environment variable as key
    - Requires knowledge upfront
- Store it as:
  - PE Resource
  - In PE Sections
  - Local file
  - Remote file
- Loader needs to know how to *reverse* the storage process

Executing Shellcode

- Memory allocation
  - RWX
  - Suitable size
- Transfer code execution to shellcode
  - Create thread
  - Function pointer
  - Callback function

Exercise 5 - Shellcode Execution

Solution 5 - Shellcode Execution

Exercise 6 - Putting it Together

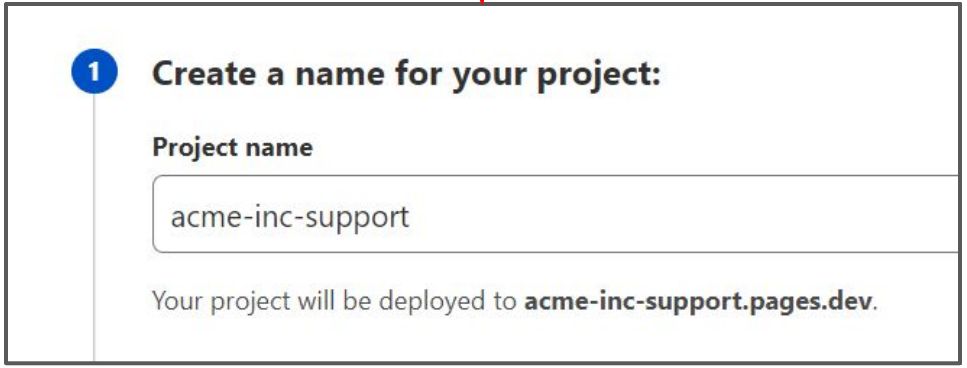
Solution 6 - Putting it Together

Section 6 - Delivery

Payload Hosting

- Cloud Storage Blobs
  - Blobs are publicly accessible via DNS
  - Name <company>-it or something that matches your pretext
  - Easy to navigate to, and convincing!
- If public, lack of control into who can download so…
  - ZIP it with a password
  - Share password with target
- Keep an eye on VirusTotal for your hashes

- **Endless** opportunities to leverage trusted SaaS infrastructure for payload hosting
  - Ensure you are authorized to do so - or you may get in trouble!


1 Create a name for your project:

Project name

acme-inc-support

Your project will be deployed to **acme-inc-support.pages.dev**.

CloudFlare Pages

Environmental Keying

- Check environment information & exit if it does not match
- Hostname
- Domain Name
- Username
- IP address
- MAC Address

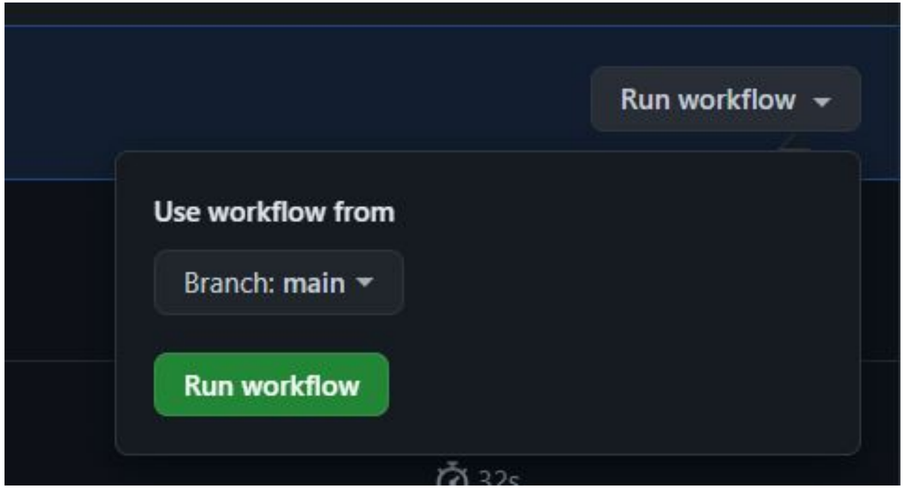
- Hard to get a lot of this information from external without prior knowledge
  - Not impossible…

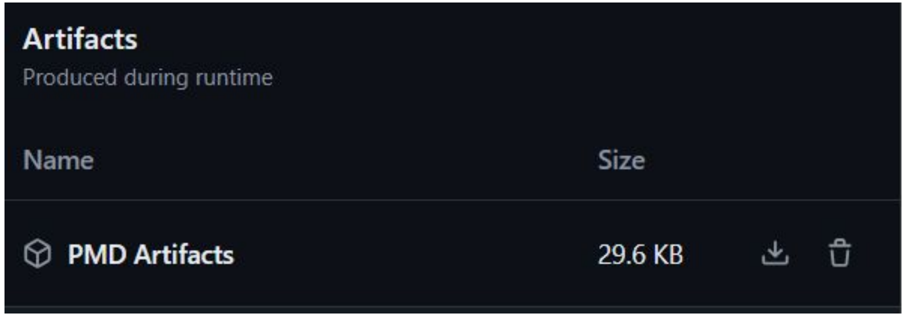
Exercise 7 - Payload Hosting

Section 7 - CI/CD

Why?

- Save time
  - Automate build process for other operators
  - Compile payloads without developer environment
- Add checks in pipeline for rules etc
  - Any manual work you do with a payload that you can automate, you should automate
- Automatically package payloads ready for delivery





Artifacts
Produced during runtime

Name	Size
PMD Artifacts	29.6 KB

Exercise 8 - Building Payloads with Github Actions

Section 8 - EDR Stuff

Known Knowns	Known Unknowns
Unknown Knowns	Unknown Unknowns

Telemetry

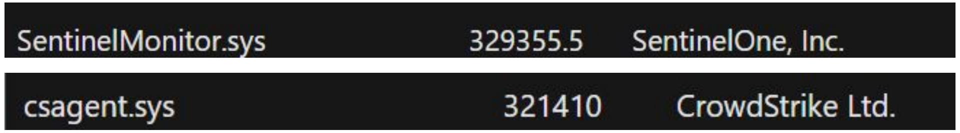
- ❖ Kernel Driver
  - ➢ Filesystem/Registry
  - ➢ Image/Process
  - ➢ Network
- ❖ Usermode DLL
  - ➢ Hook functions
  - ➢ Register callbacks
- ❖ ETW
  - ➢ EDR can consume different providers from user mode
  - ➢ Implemented at kernel level

https://github.com/tsale/EDR-Telemetry

Telemetry Feature Category	Sub-Category	Carbon Black	Cortex XDR	CrowdStrike	Cybereason	ESET Inspect	Elastic	Harfanglab	LimaCharlie	MDE
Process Activity	Process Creation	✅	✅	✅	✅	✅	✅	✅	✅	✅
	Process Termination	⚠️	✅	✅	✅	✅	✅	❌	✅	✅
	Process Access	✅	✅	✅	✅	⚠️	✅	✅	✅	✅
	Image/Library Loaded	✅	✅	✅	✅	✅	✅	✅	✅	✅
	Remote Thread Creation	✅	✅	✅	✅	✅	✅	✅	✅	✅
	Process Tampering Activity	⚠️	⚠️	✅	❓	❌	✅	✅	✅	✅
File Manipulation	File Creation	✅	✅	✅	✅	⚠️	✅	✅	✅	✅
	File Opened	✅	❌	✅	❌	❌	✅	✅	⚠️	❌
	File Deletion	✅	✅	✅	✅	✅	✅	❌	✅	✅
	File Modification	✅	✅	✅	❌	✅	✅	✅	✅	✅
	File Renaming	✅	✅	✅	✅	✅	✅	✅	⚠️	✅
User Account Activity	Local Account Creation	❌	🪵	✅	❌	✅	🪵	🪵	🪵	✅
	Local Account Modification	❌	🪵	⚠️	❌	✅	🪵	🪵	🪵	✅
	Local Account Deletion	❌	🪵	✅	❌	✅	🪵	🪵	🪵	✅
	Account Login	🪵	✅	✅	✅	✅	✅	✅	⚠️	✅
	Account Logoff	🪵	✅	✅	✅	✅	✅	✅	🪵	❌
Network Activity	TCP Connection	✅	✅	✅	✅	✅	✅	✅	✅	✅
	UDP Connection	✅	✅	✅	✅	❌	✅	🪵	✅	✅
	URL	❌	❌	✅	❌	✅	⚠️	✅	⚠️	✅
	DNS Query	✅	✅	✅	✅	✅	✅	✅	✅	✅
	File Downloaded	❌	❌	✅	⚠️	⚠️	❌	❌	⚠️	✅
Hash Algorithms	MD5	✅	✅	✅	✅	✅	✅	✅	✅	✅
	SHA	✅	✅	✅	✅	✅	✅	✅	✅	✅
	IMPHASH	❌	❌	❌	❌	❌	⚠️	✅	❌	❌
Registry Activity	Key/Value Creation	✅	✅	⚠️	⚠️	✅	✅	✅	✅	✅
	Key/Value Modification	✅	✅	⚠️	⚠️	✅	✅	✅	✅	✅
	Key/Value Deletion	✅	✅	❌	⚠️	✅	✅	✅	✅	✅
Schedule Task Activity	Scheduled Task Creation	❌	🪵	✅	✅	❌	🪵	🪵	🪵	✅
	Scheduled Task Modification	❌	🪵	✅	✅	❌	🪵	🪵	🪵	✅
	Scheduled Task Deletion	❌	🪵	✅	❌	❌	🪵	🪵	🪵	✅
Service Activity	Service Creation	⚠️	🪵	✅	✅	❌	🪵	🪵	✅	🪵
	Service Modification	❌	🪵	⚠️	❌	❌	🪵	🪵	✅	❌
	Service Deletion	❌	❌	❌	❌	❌	🪵	❌	❓	❌
Driver/Module Activity	Driver Loaded	❌	✅	✅	✅	✅	✅	✅	✅	✅
	Driver Modification	❌	❌	✅	❌	❌	❌	❌	✅	❌

Kernel Driver

- ❖ Filter Driver
  - ➢ Communicates with userland process through filter communication port
  - ➢ Driver intercepts and awaits decision ⇔ Userland process contains detection logic
- ❖ PsSet*NotifyRoutine
  - ➢ Notifes driver-supplied callback whenever image/thread/process is created/deleted
- ❖ WFP Callout/NDIS driver
  - ➢ Packet Inspection/Streaming/Modification
- ❖ Anti-tampering



SentinelMonitor.sys	329355.5	SentinelOne, Inc.
csagent.sys	321410	CrowdStrike Ltd.

https://learn.microsoft.com/en-us/windows-hardware/drivers/ifs/allocated-altitudes

Usermode DLL

- Injected by EDR Driver into process
- Increased introspection <u>that is not easily achieved</u> in kernel
- Hook Native (Nt*) API + other functions
  - Capture function arguments
  - Determine whether malicious or not
- Register Instrumentation Callbacks
  - `NtSetInformationProcess`
  - `ProcessInstrumentationCallback`
    - Syscall call stack analysis
- Register userland DLL load notifications
  - `LdrRegisterDllNotification`

Unhooking

- Refresh the “section” with an unmodified copy
  - We can get the unmodified copy:
    - Reading from disk
    - Or from mapping Known DLLs if present there
- EDRs will check if hooks are removed occasionally
  - Can restore hooks once “malicious” activity is done
- EDRs will STILL have some level of telemetry into your activity as this only deals with the userland

EDR Testing Lab Guidelines

- EDR testing lab
  - **Online**
    - Increased confidence on payload success
    - Bypass in lab != Bypass in target environment
  - **Offline**
    - Limited analysis
    - less % of burning payload
- Different environments, different EDR configurations
  - Business > Security
  - alerts != game over
- If you feel like you’ve been burnt, change <u>known known</u> stuff they will look for
  - Payload hashes
  - C2 Infra

Thank You. Any Questions?