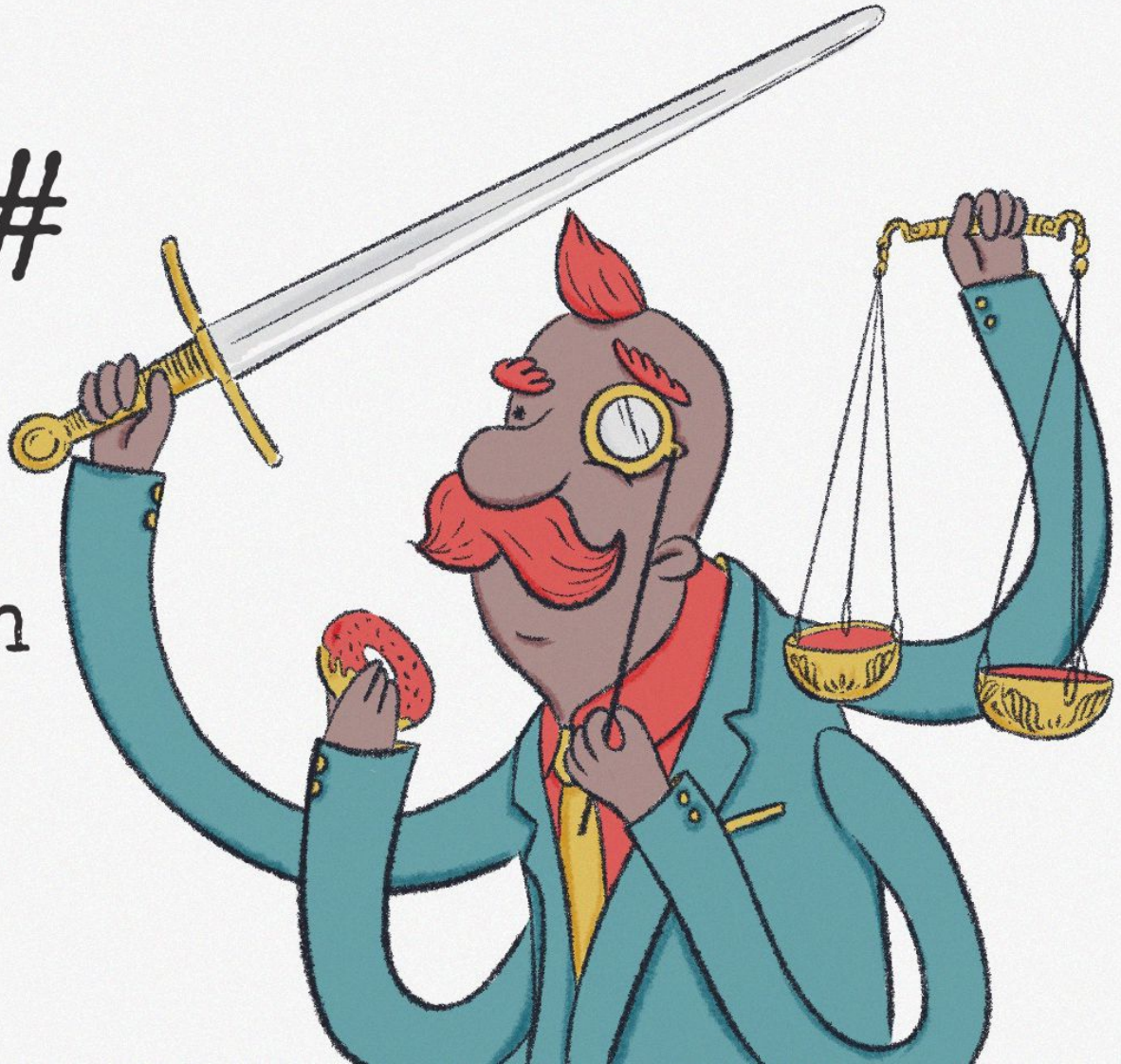
Staying

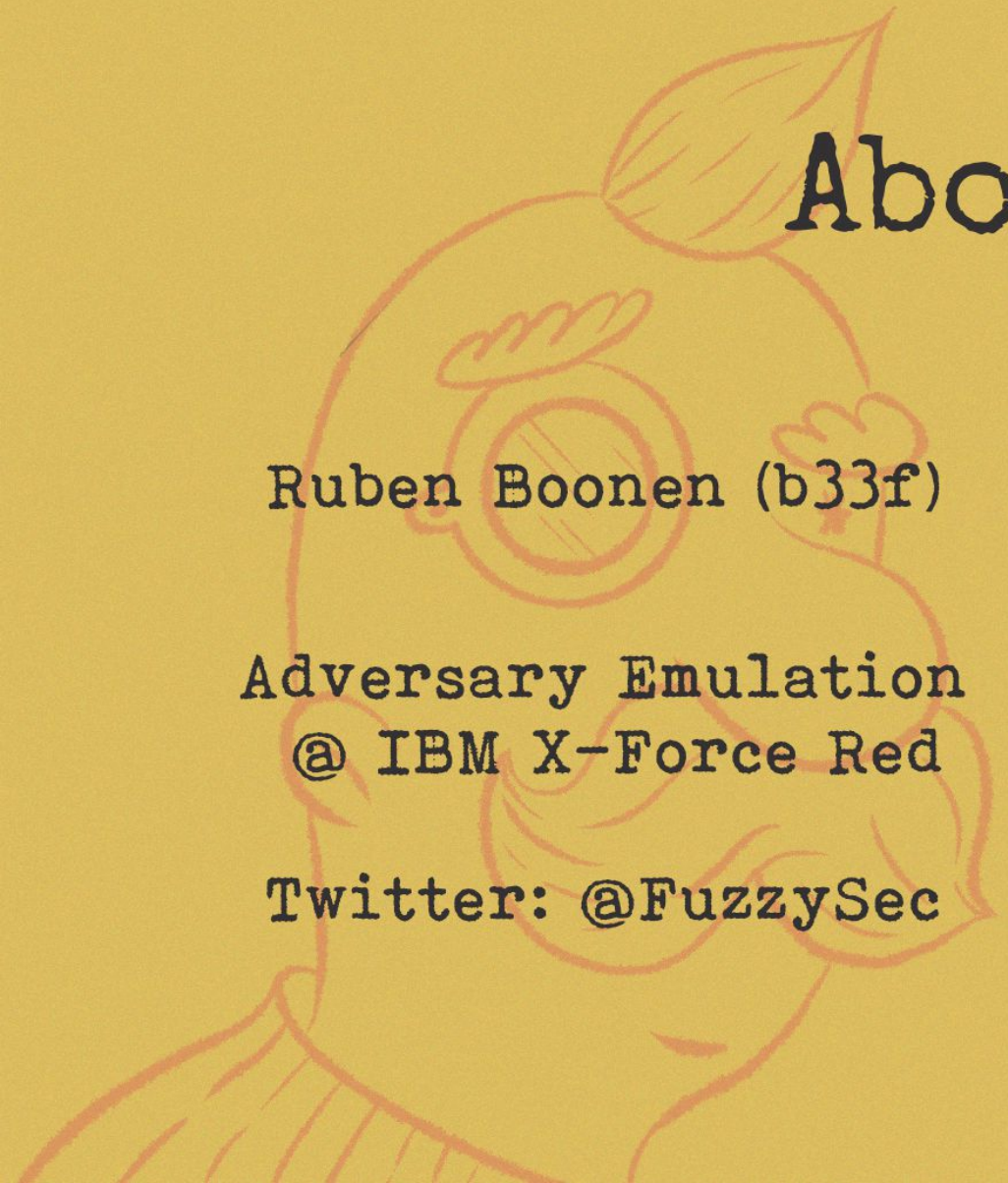
and Bringing

Covert Injection Tradecraft

to .NET



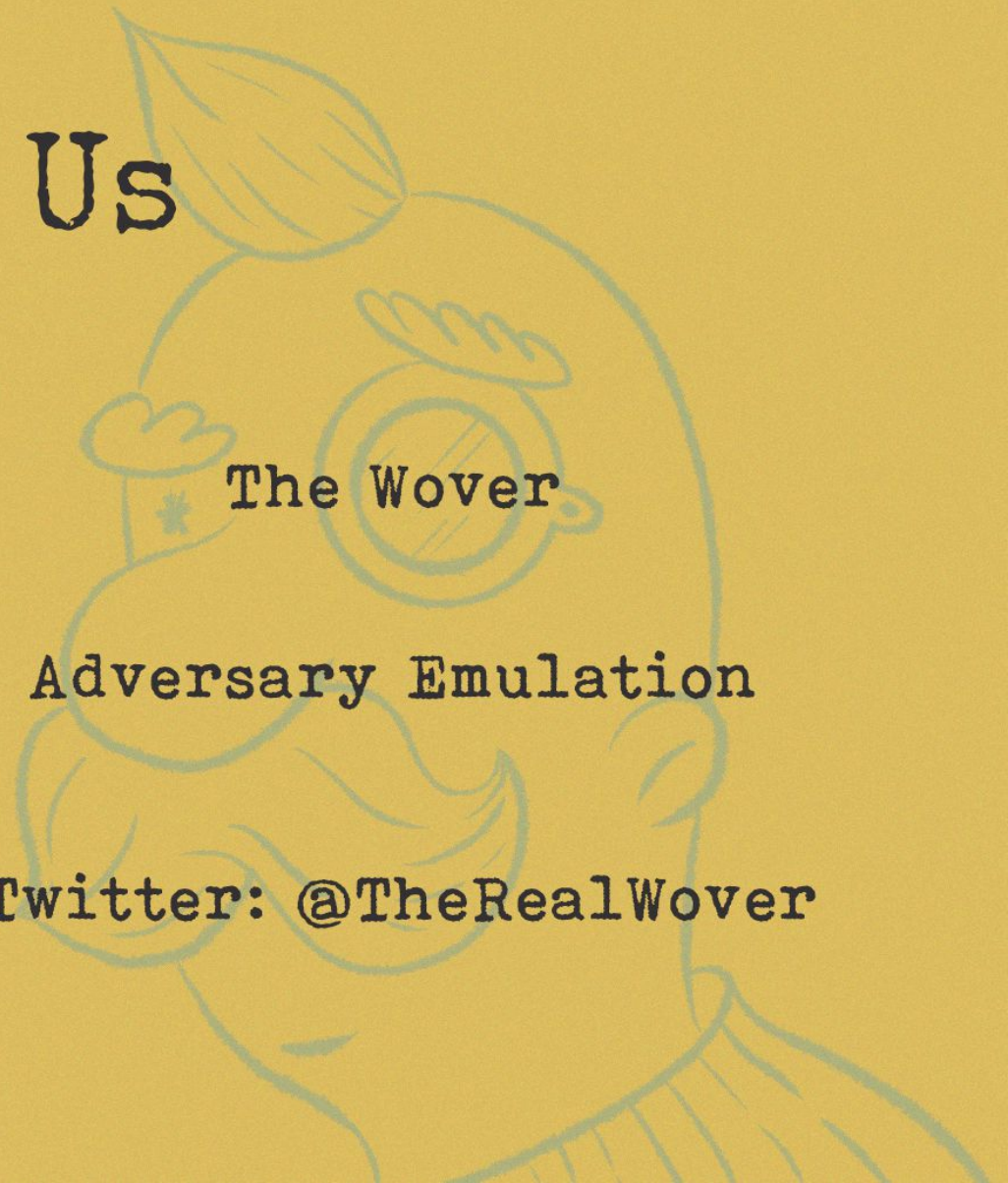
About Us



Ruben Boonen (b33f)

Adversary Emulation
@ IBM X-Force Red

Twitter: @FuzzySec



The Wover

Adversary Emulation

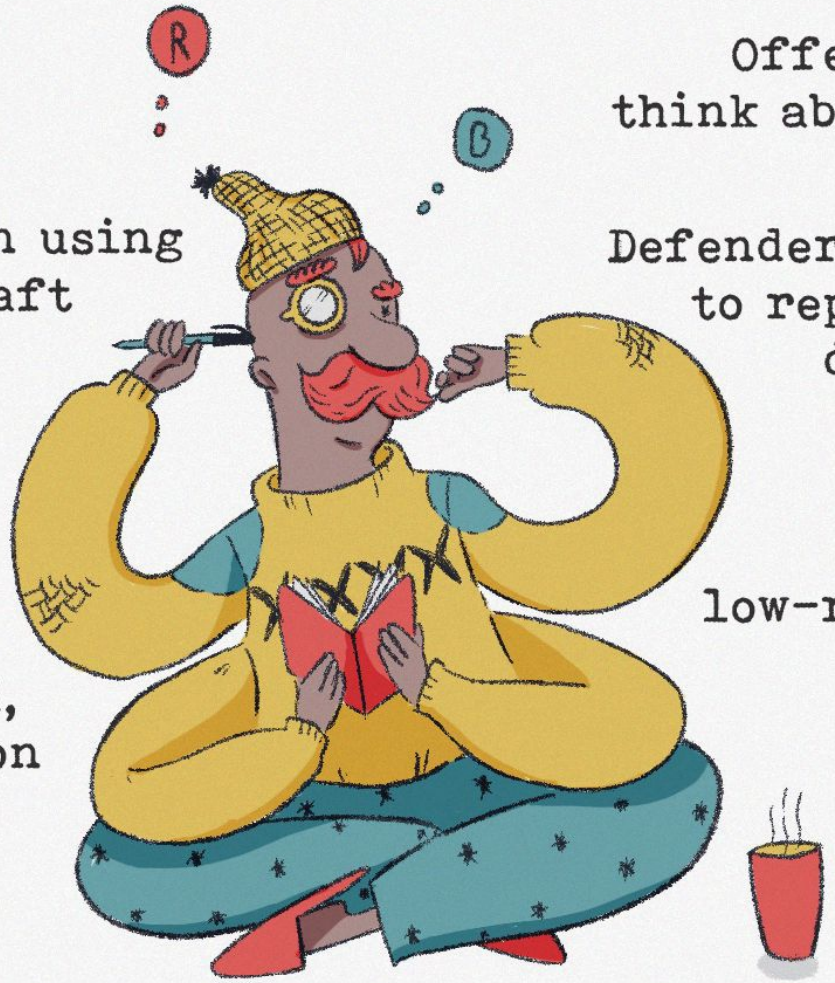
Twitter: @TheRealWover

Why talk about in-memory tradecraft?

In offense we have been using in-memory tradecraft

It's dated but it has been working (think Stephen Fewer)

Changes are needed, look at the evolution of Cobalt Strike



Offense informs defense, think about innovations like AMSI

Defenders need models they can use to reproduce tradecraft and develop detections

Threat groups are already doing this, low-rent crypto miners as well

Focus on principles and primitives to catch the behavior

Modern .NET Tradecraft

Many new, powerful .NET toolkits

Attackers leverage legitimate APIs;
using Microsoft libraries for post-ex

No longer just for skiddies

As .NET becomes more powerful,
so does .NET malware

Look for anomalous loading
of clr.dll;ETW can log
Assembly loads from memory

Can be run from memory,
hard to inspect at scale

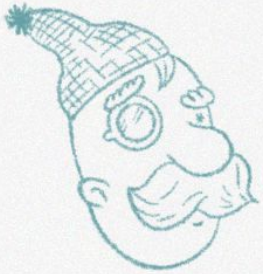
Same TTPs as earlier,
new delivery mechanism

Easy transition
from PowerShell

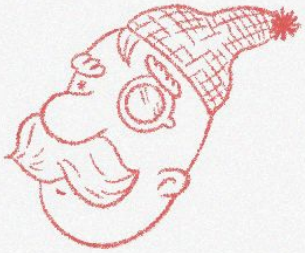
Increased reliance
makes it a point-of-failure
when caught

Easy to develop

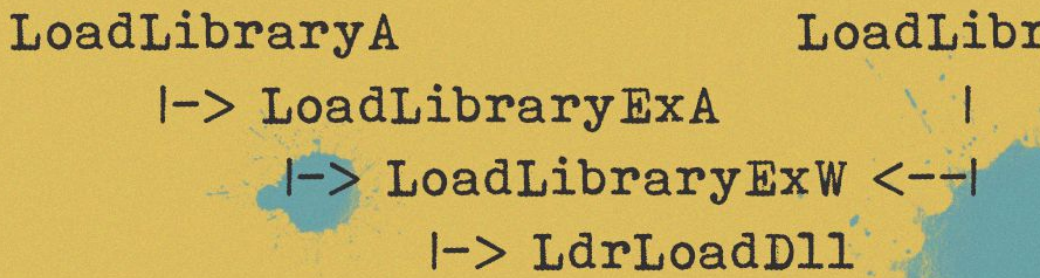
SharpSploit (@cobbr)



- Aims to highlight attack surface of .NET,
- Library of post-exploitation TTPs
- Loaded into many RATs to be used by tasks/modules
- Accessible in Covenant, a .NET C2 framework.
  We will use it with C# scripting for demos.



DynamicInvoke Principles: ++Undocumented



Why though?

DynamicInvoke Principles: GetDelegateForFunctionPointer

No LoadLibrary

PEB -> PEB_LDR_DATA -> InLoadOrderModuleList -> LDR_DATA_TABLE_ENTRY

No GetProcAddress

IMAGE_NT_HEADER -> IMAGE_OPTIONAL_HEADER -> IMAGE_EXPORT_DIRECTORY
|-> By: Name, Ordinal, HMACMD5(key)

DynamicInvoke Principles: Manual Mapping

- The ability to manually map an executable or DLL
  |-> Alloc SizeOfImage, write headers & sections,
      Relocate, Rewrite IAT, Set permissions

- Crude(ish), it does the job but needs more loving

DynamicInvoke Principles: Generic Syscall Wrapper

- What are the challenges to using direct Syscalls operationally?
- Manual map duplicate of ntdll -> RX copy of Syscall stub

Using the DInvoke API Covertly

Defenses

Anomalous process behavior

API Hooking

Memory scanners (e.g. pe-sieve)

Execution in unusual locations

Evasions

Avoid Image Load events

Manually map, or use PEB

Free when done, hide your code

Map into file-backed memory

=> https://www.forrest-orr.net/post/malicious-memory-artifacts-part-i-dll-hollowing

Covert Win32/Nt API Calling

(1) No P/Invoke: No static imports in IAT or Image Load events

(2) No LoadLibrary: MapModuleToMemory(filePath)

(3) No GetProcAddress: GetExportAddress
(moduleAddress, exportName)

(4) Execute with args ==> DynamicFunctionInvoke
(exportAddress, functionPrototype, parameters)

Module Overloading

- What if we map a legit, signed dll, then overwrite it?
- NtCreateSection(SEC_IMAGE) + NtMapViewOfSection
- Overwrite the Section with our payload, then have to map it ourselves :-(
- Code executed in the payload will run from file-backed memory

Module Overloading

- M = Random, legitimately signed module in System32/SysWOW64
- S = NtCreateSection(M) + SEC_IMAGE
- P = Payload, PE we want to use from memory
- Write P to baseAddress of a View of S and virtualize the module

- New thread's start address is in file-backed memory
- Appears to be executing in a legitimate, signed DLL

0x7fffac3fc000	Image: Commit	4 kB	RW	C:\Windows\System32\umpdc.dll
0x7fffac3fd000	Image: Commit	12 kB	R	C:\Windows\System32\umpdc.dll
0x230f2180000	Image: Commit	4 kB	R	C:\Windows\System32\user32.dll
0x230f2181000	Image: Commit	772 kB	RX	C:\Windows\System32\user32.dll
0x230f2242000	Image: Commit	348 kB	R	C:\Windows\System32\user32.dll
0x230f2299000	Image: Commit	32 kB	RW	C:\Windows\System32\user32.dll
0x230f22a1000	Image: Commit	56 kB	R	C:\Windows\System32\user32.dll
0x230f22af000	Image: Commit	404 kB	WCX	C:\Windows\System32\user32.dll
0x7fffaef10000	Image: Commit	4 kB	R	C:\Windows\System32\user32.dll
0x7fffaef11000	Image: Commit	536 kB	RX	C:\Windows\System32\user32.dll
0x7fffaef97000	Image: Commit	128 kB	R	C:\Windows\System32\user32.dll
0x7fffaefb7000	Image: Commit	8 kB	RW	C:\Windows\System32\user32.dll

mimikatz 2.2.0 x64 (oe.eo)

```
[+] Module Address: 2409243344896
[+] Module Backing File: C:\Windows\System32\user32.dll
Hold fire!

Firing!
[+] Thread: 0

  .#####.   mimikatz 2.2.0 (x64) #18362 Jan  4 2020 18:59:26
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz #
```

Module Overloading: Covert Execution

Process Injection Word Soup

```
     NtMapViewOfSection \                / NtQueueApcThread ----------|
NtAllocateVirtualMemory  \ _ .... _/   NtCreateThreadEx (RtlExitUserThread) -> NtAlertResumeThread
          GlobalAddAtom  /             \  SetWindowsHookEx
   NtUpdateWnfStateData /               \ PROPagate / WNF / Conhost / ExtraBytes
```

=> Also hybrid techniques like Ghost-Writing

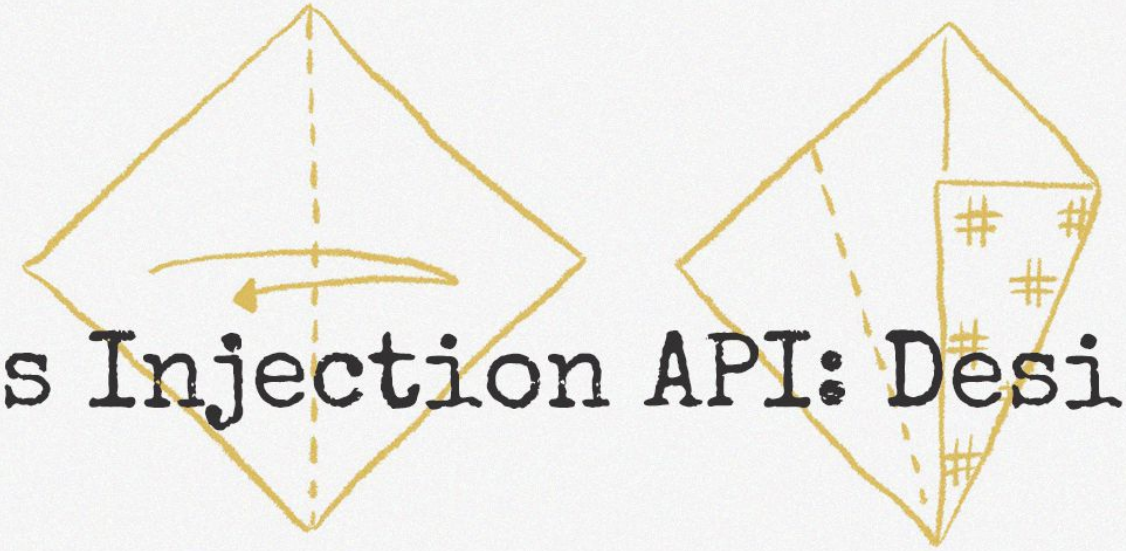
The reality is different

Allocation & Execution -> A many to many relationship
-> Not a totally accurate picture

Component Chains

- Injection techniques are built out of distinct components
- Putting sets of components into SharpSploit gives users flexibility
- For now the available building blocks are limited
- A pick your own adventure/poison type deal <3

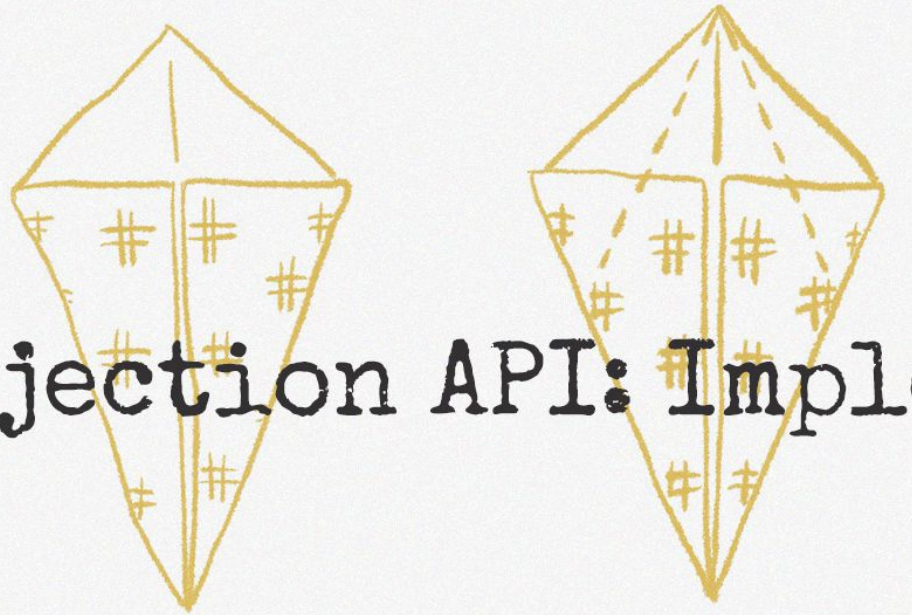
==> Similar to the approach of Pinjectra



Process Injection API: Design Goals

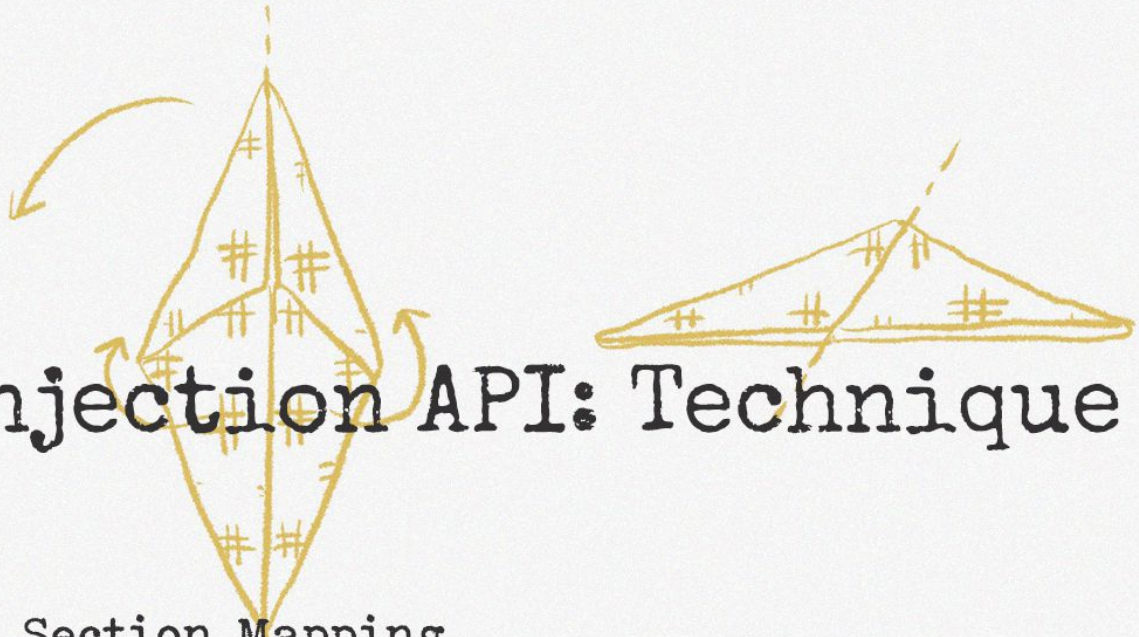
- Build an API on top of these primitives
- Modular, implementation-agnostic, object-oriented
- Easily extensible; just implement a subclass
- Build your own injector from components

=> https://www.blackhat.com/us-19/briefings/schedule/#process-injection-techniques---gotta-catch-them-all-16010



Process Injection API: Implementation

- PayloadType
- ExecutionTechnique
- AllocationTechnique
- Implement functionality in subclasses, call it via polymorphism

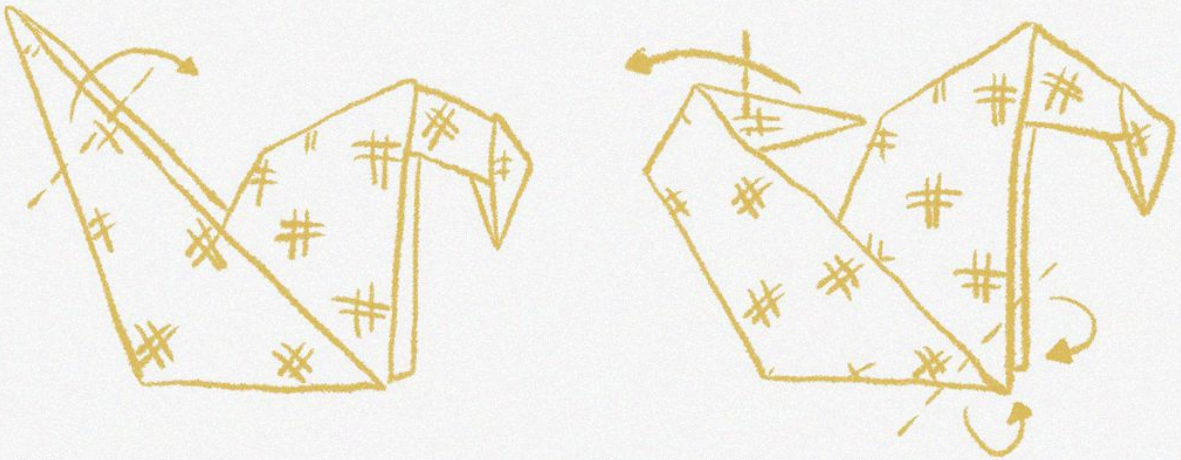


Process Injection API: Technique Examples

- Allocation: Section Mapping
- Can set permissions, copy from a local section
- Execution: Remote Thread Creation
- Can specify which thread creation API call to use, start suspended

Process Injection API: Building an Injector

- Build an injector by combining techniques and a payload
- Specify options for each
- Injector.Inject(injectionTechnique, allocationTechnique, payload, process);



Process Injection API: Demonstration

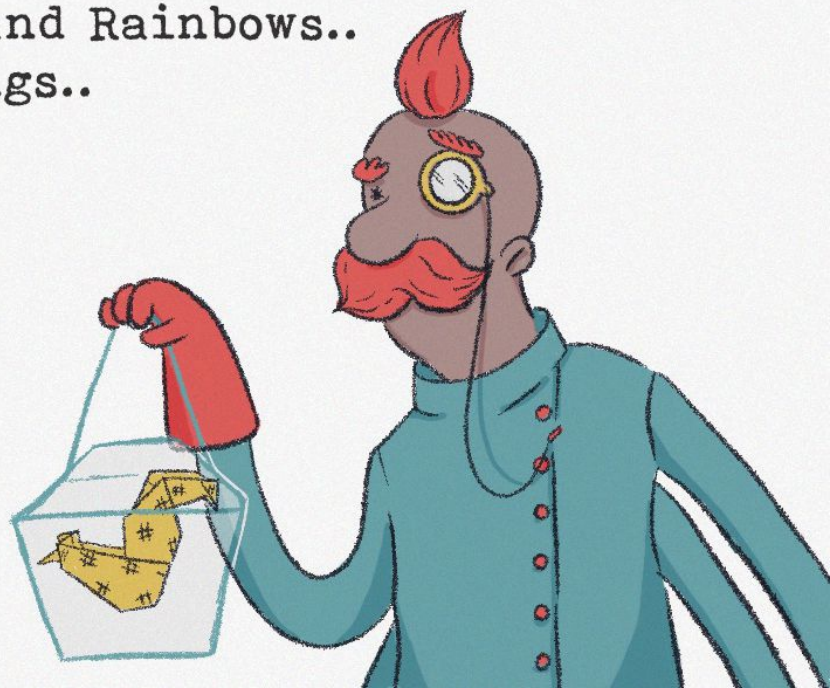
- Through an existing implant, rapidly build an injector

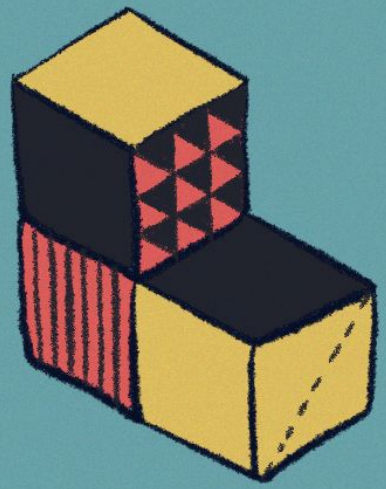
- Operator specifies at runtime how their tool behaves

[Demo: Use SharpShell in a Grunt to write and use an injector in a few lines of code]

Detection Strategies

Unfortunately it is not all Sunshine, Lollipops and Rainbows..
..but we can try some things..

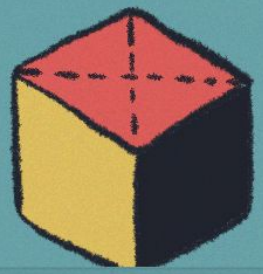




Correlating Module Load Events

- Injecting .NET assemblies into memory causes a number of CLR modules to be loaded by the application.

- Here we see notepad pre-injection.



Untitled - Notepad

File Edit Format View Help

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

notepad.exe (17832) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment | Windows

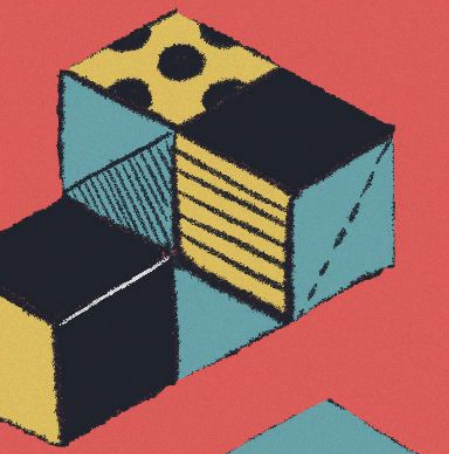
Options | Search Modules (Ctrl+K)

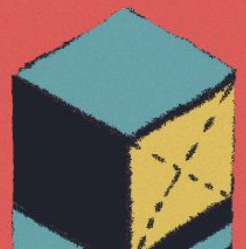
Name	ase address	Size	Description	File name
R000000000001.clb	31b35b0000	24 kB		C:\Windows\Registration\R0000
profapi.dll	ffb4cdf0000	124 kB	User Profile Basic API	C:\Windows\System32\profapi.d
powrprof.dll	fb4ce30000	296 kB	Power Profile Helper DLL	C:\Windows\System32\powrprof
oleaut32.dll	fb4e8b0000	784 kB	OLEAUT32.DLL	C:\Windows\System32\oleaut32
oleaccrc.dll	81b35f0000	8 kB	Active Accessibility Resourc...	C:\Windows\System32\oleaccrc.
oleacc.dll	fb3e830000	404 kB	Active Accessibility Core Co...	C:\Windows\System32\oleacc.dll
ntmarta.dll	fb4be40000	196 kB	Windows NT MARTA provider	C:\Windows\System32\ntmarta.d
ntdll.dll	ffb4ff40000	1.94 MB	NT Layer DLL	C:\Windows\System32\ntdll.dll
notepad.exe.mun	31b3420000	104 kB	Notepad	C:\Windows\SystemResources\n
notepad.exe.mui	31b19e0000	12 kB	Notepad	C:\Windows\System32\en-US\nc
notepad.exe	**6f0250000**	**200 kB**	**Notepad**	**C:\Windows\System32\not**
msvcrt.dll	ffb4e9f0000	632 kB	Windows NT CRT DLL	C:\Windows\System32\msvcrt.dl
msvcp_win.dll	fb4db70000	632 kB	Microsoft® C Runtime Library	C:\Windows\System32\msvcp_w
msctf.dll	ffb4f580000	1.21 MB	MSCTF Server DLL	C:\Windows\System32\msctf.dll
MrmCoreR.dll	fb44640000	1.07 MB	Microsoft Windows MRM	C:\Windows\System32\MrmCore
mpr.dll	ffb35fa0000	108 kB	Multiple Provider Router DLL	C:\Windows\System32\mpr.dll
locale.nls	31b1b20000	796 kB		C:\Windows\System32\locale.nls
KernelBase.dll	fb4dc10000	2.64 MB	Windows NT BASE API Clien...	C:\Windows\System32\KernelBa
kernel32.dll	fb4e6d0000	712 kB	Windows NT BASE API Clien...	C:\Windows\System32\kernel32.
kernel.appcore.dll	fb4ce10000	68 kB	AppModel API Host	C:\Windows\System32\kernel.ap
imm32.dll	ffb4fb60000	184 kB	Multi-User Windows IMM32 ...	C:\Windows\System32\imm32.dll
iertutil.dll	ffb3f760000	2.65 MB	Run time utility for Internet ...	C:\Windows\System32\iertutil.dll
gdi32full.dll	fb4d080000	1.58 MB	GDI Client DLL	C:\Windows\System32\gdi32full.
gdi32.dll	ffb4f6c0000	152 kB	GDI Client DLL	C:\Windows\System32\gdi32.dll
efswrt.dll	fb295c0000	860 kB	Storage Protection Windows...	C:\Windows\System32\efswrt.dl
cryptsp.dll	ffb4def0000	92 kB	Cryptographic Service Provi...	C:\Windows\System32\cryptsp.d
CoreUIComponents.dll	fb46de0000	3.16 MB	Microsoft Core UI Compone...	C:\Windows\System32\CoreUIC
CoreMessaging.dll	fb4a6d0000	848 kB	Microsoft CoreMessaging Dll	C:\Windows\System32\CoreMes
comctl32.dll	fb3eb60000	2.52 MB	User Experience Controls Li...	C:\Windows\WinSxS\amd64_mic
combase.dll	ffb4f780000	3.21 MB	Microsoft COM for Windows	C:\Windows\System32\combase
clbcatq.dll	ffb4fb90000	648 kB	COM+ Configuration Catalog	C:\Windows\System32\clbcatq.d
cfgmgr32.dll	ffb4cf00000	296 kB	Configuration Manager DLL	C:\Windows\System32\cfgmgr32
bcryptprimitives.dll				

Close

- When an Assembly is loaded by the process a whole set of new modules appears.
- This approach requires silent testing & FP tuning.

Correlating Module Load Events





```
Command Prompt

C:\Users\b33f\Tools\donut>C:\Users\b33f\Tools\Dev\UrbanBishop\bin\Releas
e\UrbanBishop.exe -p C:\Users\b33f\Tools\Dev\DonutSc.bin -i 17832

                 _o
               /  //\
              {     }
               \   /
                (_)
                 |_
               /   \
              (_____)
              (_____)
              /_____\
                ~b33f~

|--------
| Process     : notepad
| Handle      : 796
| Is x32      : False
| Sc binpath  : C:\Users\b33f\Tools\Dev\DonutSc.bin
|--------

[>] Creating local section..
    |-> hSection: 0x320
    |-> Size: 35479
    |-> pBase: 0x2FC0000
[>] Map RX section to remote proc..
    |-> pRemoteBase: 0x181B3600000
[>] Write shellcode to local section..
    |-> Size: 35479
[>] Seek export offset..
    |-> pRemoteNtDllBase: 0x7FFB4FF40000
    |-> LdrGetDllHandle OK
    |-> RtlExitUserThread: 0x7FFB4FFACF00
    |-> Offset: 0x6CF00
[>] NtCreateThreadEx -> RtlExitUserThread <- Suspended..
    |-> Success
[>] Set APC trigger & resume thread..
    |-> NtQueueApcThread
    |-> NtAlertResumeThread

C:\Users\b33f\Tools\donut>
```

```
In-Memory
Hello Donut!
OK
```

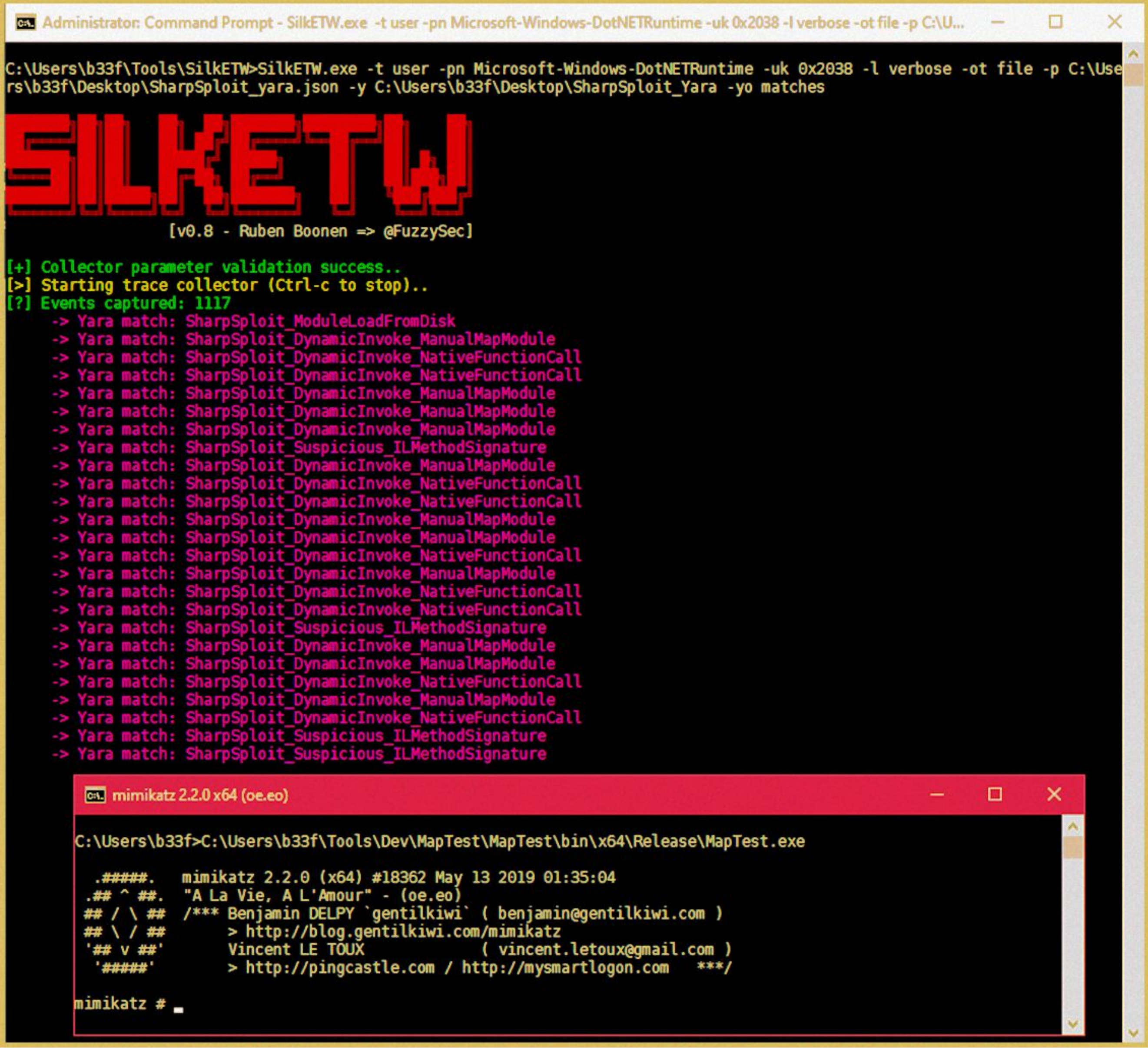
```
notepad.exe (17832) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | GPU | Comment | Windows

Options                                                    Search Modules (Ctrl+K)

Name                ase address   Size      Description                  File name
oleaccrc.dll        81b35f0000    8 kB      Active Accessibility Resourc...  C:\Windows\System32\oleaccrc.
oleacc.dll          7fb3e830000   404 kB    Active Accessibility Core Co...  C:\Windows\System32\oleacc.dll
ole32.dll           7fb4f420000   1.34 MB   Microsoft OLE for Windows    C:\Windows\System32\ole32.dll
ntmarta.dll         7fb4be40000   196 kB    Windows NT MARTA provider    C:\Windows\System32\ntmarta.
ntdll.dll           7fb4ff40000   1.94 MB   NT Layer DLL                 C:\Windows\System32\ntdll.dll
notepad.exe.mun     81b3420000    104 kB    Notepad                      C:\Windows\SystemResources\n
notepad.exe.mui     81b19e0000    12 kB     Notepad                      C:\Windows\System32\en-US\no
notepad.exe         0f0250000     200 kB    Notepad                      C:\Windows\System32\not
msvcrt.dll          7fb4e9f0000   632 kB    Windows NT CRT DLL           C:\Windows\System32\msvcrt.dl
msvcr80.dll         0x56430000    804 kB    Microsoft® C Runtime Library  C:\Windows\WinSxS\amd64_mic
msvcp_win.dll       7fb4db70000   632 kB    Microsoft® C Runtime Library  C:\Windows\System32\msvcp_w
msctf.dll           7fb4f580000   1.21 MB   MSCTF Server DLL             C:\Windows\System32\msctf.dll
mscorwks.dll        7fafc3c0000   9.63 MB   Microsoft .NET Runtime Com...  C:\Windows\Microsoft.NET\fram
mscorlib.ni.dll     7fafb4d0000   14.89 MB  Microsoft Common Languag...  C:\Windows\assembly\NativeIma
mscorjit.dll        7fafec10000   1.51 MB   Microsoft .NET Runtime Just...  C:\Windows\Microsoft.NET\fram
mscoreei.dll        7fb18820000   676 kB    Microsoft .NET Runtime Exe...  C:\Windows\Microsoft.NET\fram
mscoree.dll         7fb35f10000   400 kB    Microsoft .NET Runtime Exe...  C:\Windows\System32\mscoree.
msasn1.dll          7fb4ce80000   72 kB     ASN.1 Runtime APIs           C:\Windows\System32\msasn1.c
MrmCoreR.dll        7fb44640000   1.07 MB   Microsoft Windows MRM        C:\Windows\System32\MrmCore
mpr.dll             7fb35fa0000   108 kB    Multiple Provider Router DLL  C:\Windows\System32\mpr.dll
l_intl.nls          81b3710000    12 kB                                  C:\Windows\System32\l_intl.nls
locale.nls          81b1b20000    796 kB                                 C:\Windows\System32\locale.nls
KernelBase.dll      7fb4dc10000   2.64 MB   Windows NT BASE API Clien...  C:\Windows\System32\KernelBa
kernel32.dll        7fb4e6d0000   712 kB    Windows NT BASE API Clien...  C:\Windows\System32\kernel32.
kernel.appcore.dll  7fb4ce10000   68 kB     AppModel API Host            C:\Windows\System32\kernel.ap
imm32.dll           7fb4fb60000   184 kB    Multi-User Windows IMM32 ...  C:\Windows\System32\imm32.dll
iertutil.dll        7fb3f760000   2.65 MB   Run time utility for Internet ...  C:\Windows\System32\iertutil.dll
gdi32full.dll       7fb4d080000   1.58 MB   GDI Client DLL               C:\Windows\System32\gdi32full.
gdi32.dll           7fb4f6c0000   152 kB    GDI Client DLL               C:\Windows\System32\gdi32.dll
efswrt.dll          7fb295c0000   860 kB    Storage Protection Windows...  C:\Windows\System32\efswrt.dll
cryptsp.dll         7fb4def0000   92 kB     Cryptographic Service Provi...  C:\Windows\System32\cryptsp.c
cryptbase.dll       7fb4c820000   48 kB     Base cryptographic API DLL   C:\Windows\System32\cryptbase
crypt32.dll

Close
```



Microsoft Windows DotNETRuntime

- MSFT has implemented .NET visibility through ETW, though it is not exposed to end-users.
- Enter SilkETW/ SilkService: subscribe to any provider, filter data, tag events with Yara, serialized to JSON

AMSI for .NET v4.8

- This is a great addition to the AMSI family <3
- If enabled, support is backported to v4.0
- v3.* would still remain unprotected but needs to be installed on the system

=> AMSI's attack surface remains intact -> If a language has the capability to re-write memory it can render AMSI inoperable

Application Introspection

- Hooking is like a taboo, I know, I know <3

- It remains a very powerful tool to detect suspicious API calls / sequences of calls / aberrant parameters

- Hooking inherently brings blocking capabilities to the table

Fermion

Device

local

Process ID

ID

Process Name

Name

Attach

Process Path

C:\Users\b33f\Tools\De

Process Arguments

Args

Start

Detach

Reload Script

Process Info

Open Save DevTools About Exit

Wrap

idleFingers

```
                var isPE = peHeader.readU32();
                if (isPE == 0x4550) {
                    send("[!] WARNING DETECTED: NtWriteVirtualMemory -> PE");
                    var optHeader = peHeader.add(0x18);
                    if (optHeader.readU16() == 0x020b) {
                        send("    |-> PE is x64..");
                    } else {
                        send("    |-> PE is x86..");
                    }
                    var addressOfEntryPoint = optHeader.add(0x10);
                    var entryPointOffet = args[1].add(addressOfEntryPoint.readU32());

                    // Add entrypoint to an array, we can use
```

mimikatz 2.2.0 x64 (oe.eo)

```
  .#####.   mimikatz 2.2.0 (x64) #18362 May 13 2019 01:35:04
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz #
```

```
[?] Attempting process start..
[+] Injecting => PID: 5896, Name: C:\Users\b33f\Tools\Dev\MapTest\MapTest\bin\x64\Release\MapTest.exe
[+] Process start success
[!] WARNING DETECTED: NtWriteVirtualMemory -> PE
    |-> PE is x64..
    |-> lpEntryPoint: 0x1e75fa38
    |-> Hexdump:
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
1e5d0080  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  MZ..............
1e5d0090  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
1e5d00a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1e5d00b0  00 00 00 00 00 00 00 00 00 00 00 00 28 01 00 00  ............(...
1e5d00c0  0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68  ........!..L.!Th
1e5d00d0  69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f  is program canno
1e5d00e0  74 20 62 65                                     t be

[!] WARNING DETECTED: NtWriteVirtualMemory -> PE -> NtCreateThreadEx
    |-> lpStartAddress: 0x1e75fa38
```

- Contribute components to the process injection API
- Implement post-exploitation TTPs in C#
- Share detection techniques

How Can You Contribute?

Conclusion

- Release will be coordinated with the recording going live
- We will also release blog posts with more details
- In the meantime, the code will be in the dev branch
- If you find detection strategies in meantime,
  message us and we'll add them to the blog and credit you

Q & A