

MODULE STOMPING

Aliz Hammond, F-Secure Countercept
BSidesSG 2019

ABOUT ME



- Aliz Hammond
- Based in Singapore
- Experience in fuzzing, binary exploitation, Windows kernel
- Researcher supporting Threat Hunters

MODULE STOMPING

- The attack itself
- Detection
 - Existing techniques
 - My new technique
 - Other things you can use this research for

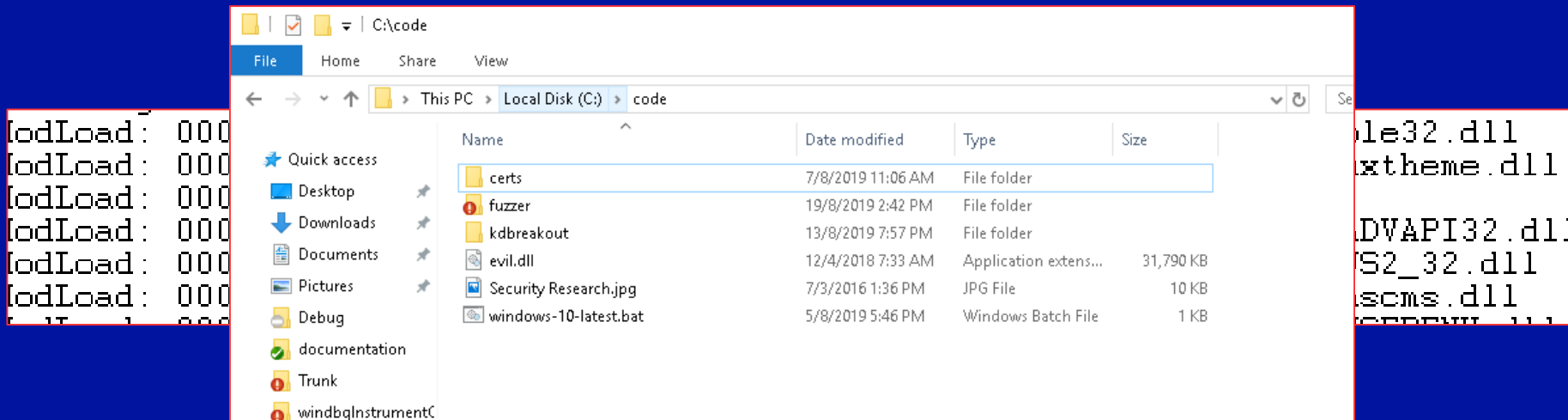
THE ATTACK

THE ATTACK

- Way of hiding malicious code

THE ATTACK

- Write DLL payload to disk, inject it into some legit process
 - Write your evil code into C:\windows\temp\evil.dll
 - OpenProcess
 - CreateRemoteThread(... LoadLibrary("C:\windows\temp\evil.dll"))



THE ATTACK

- Reflective load

1. Create a thread in a legit process as before
2. Allocate memory in the target process
3. Set this memory to be executable
4. Download your payload code into it

- No disk artefacts

- But does leave m

Windows 10 Creators Update can detect reflective Dynamic-Link Library (DLL) loading in a variety of high-risk processes, including browsers and productivity software, Microsoft says.

This is possible because of function calls (*VirtualAlloc* and *VirtualProtect*) related to procuring executable memory, which generate signals for Windows Defender Advanced Threat Protection (Windows Defender ATP).

Reflective DLL loading, the software giant explains, relies on loading a DLL into a process memory without using the Windows loader. First described in 2008, the method allows for the loading of a DLL into a process even if the DLL isn't registered with the process.

SECURITYWEEK.COM

THE ATTACK

- Enter – “Module stomping”
 1. Open a legit process
 2. Inject a thread via `CreateRemoteThread`
 3. Injected thread loads a legitimate but unnecessary system DLL
 4. Overwrite the module with our own malicious module

THE ATTACK

- All module loads are of legit files
- All executable memory backed by system DLLs
- No disk artefacts
- Easy to do yourself
- In threat emulation toolkits

EXISTING DETECTION METHODS

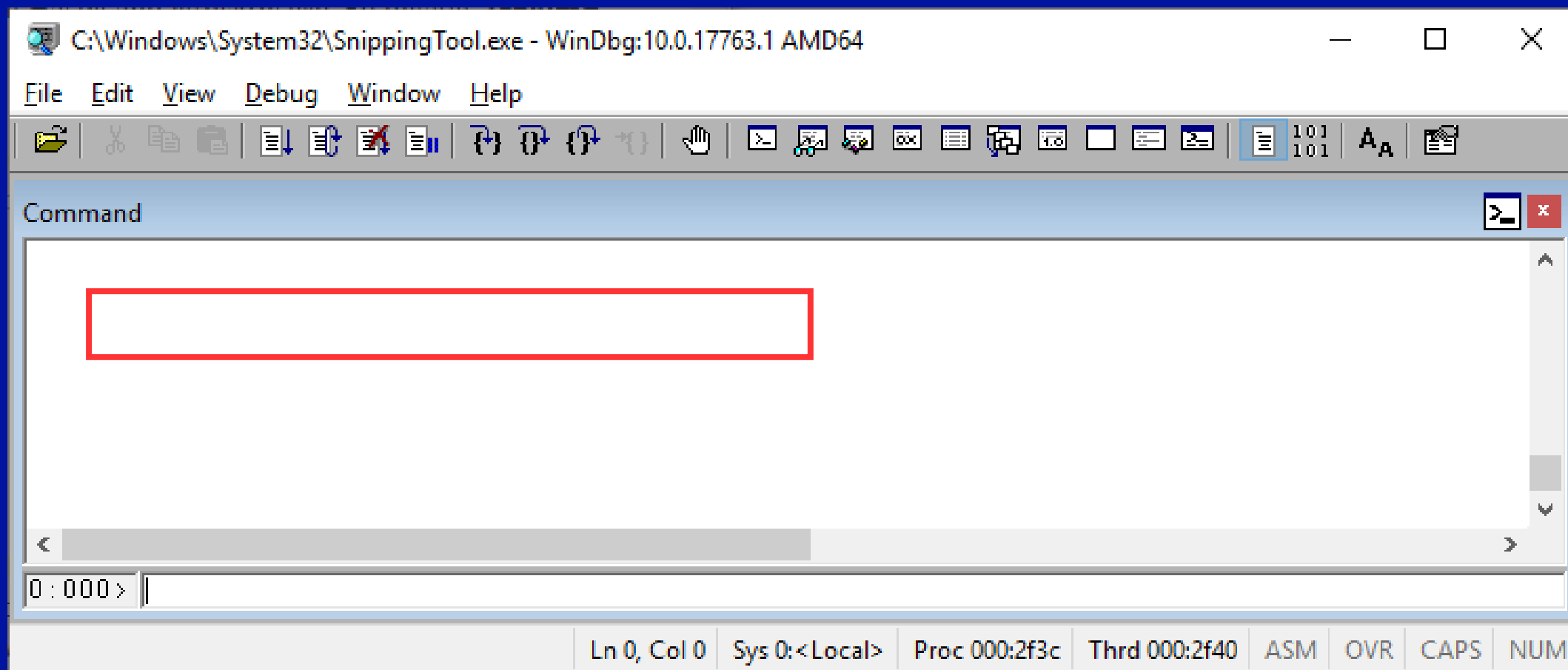
DETECTION

- Compare all modules in memory to their counterparts on disk
 - Very slow
 - Impossible in some cases
- Download legit version
 - Even slower
 - Impossible in some cases

DETECTION

- Windbg's “!ChkImg”
 - Downloads original image, checks against it

DETECTION



DETECTION

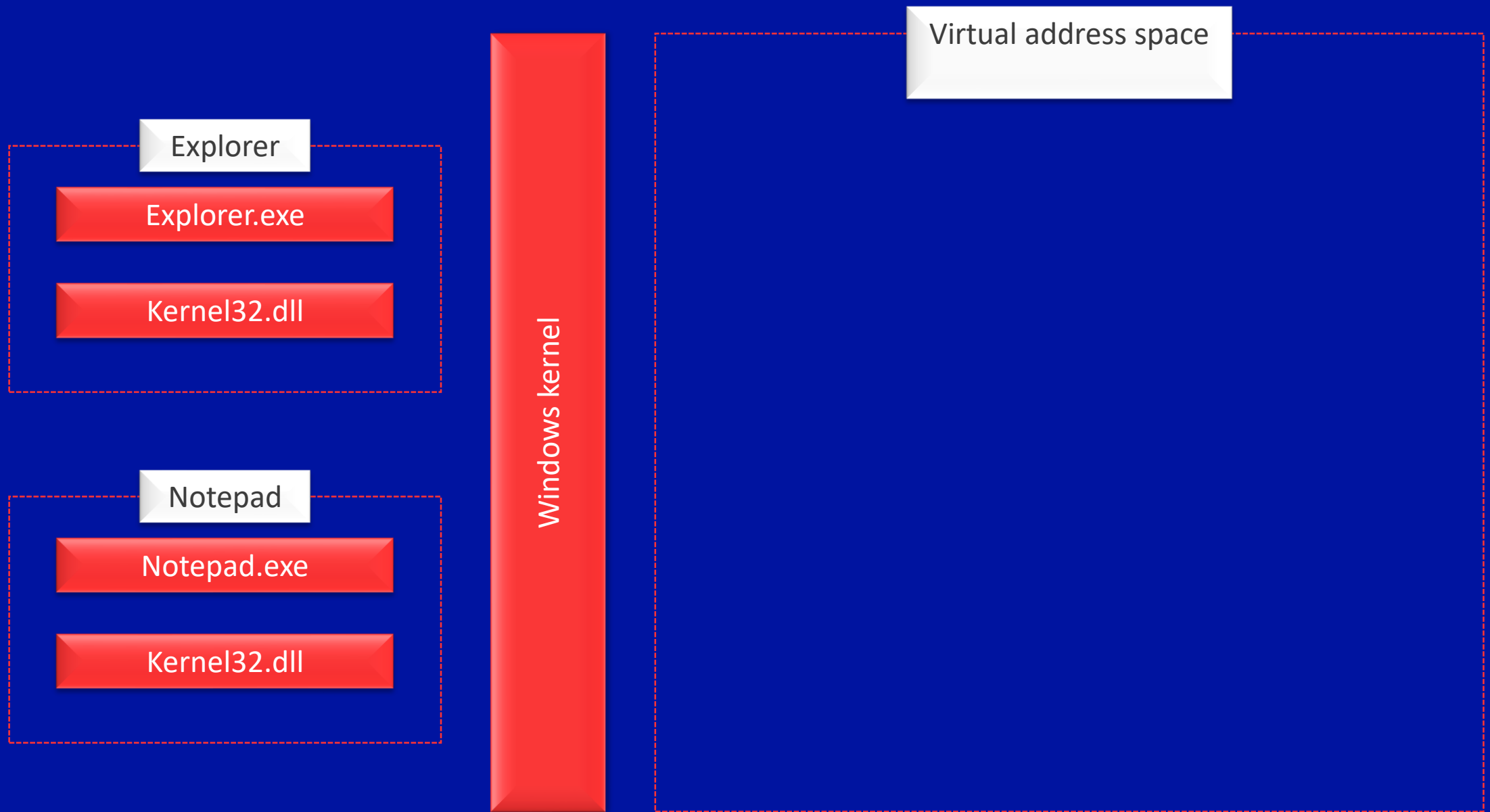
- Windbg's “!ChkImg”
 - Downloads original image, checks against it
 - Only useful when debug symbols are available
 - Almost all of Windows
 - Chrome
 - Firefox
 - Not designed for adversarial use
- Still useful for offline analysis

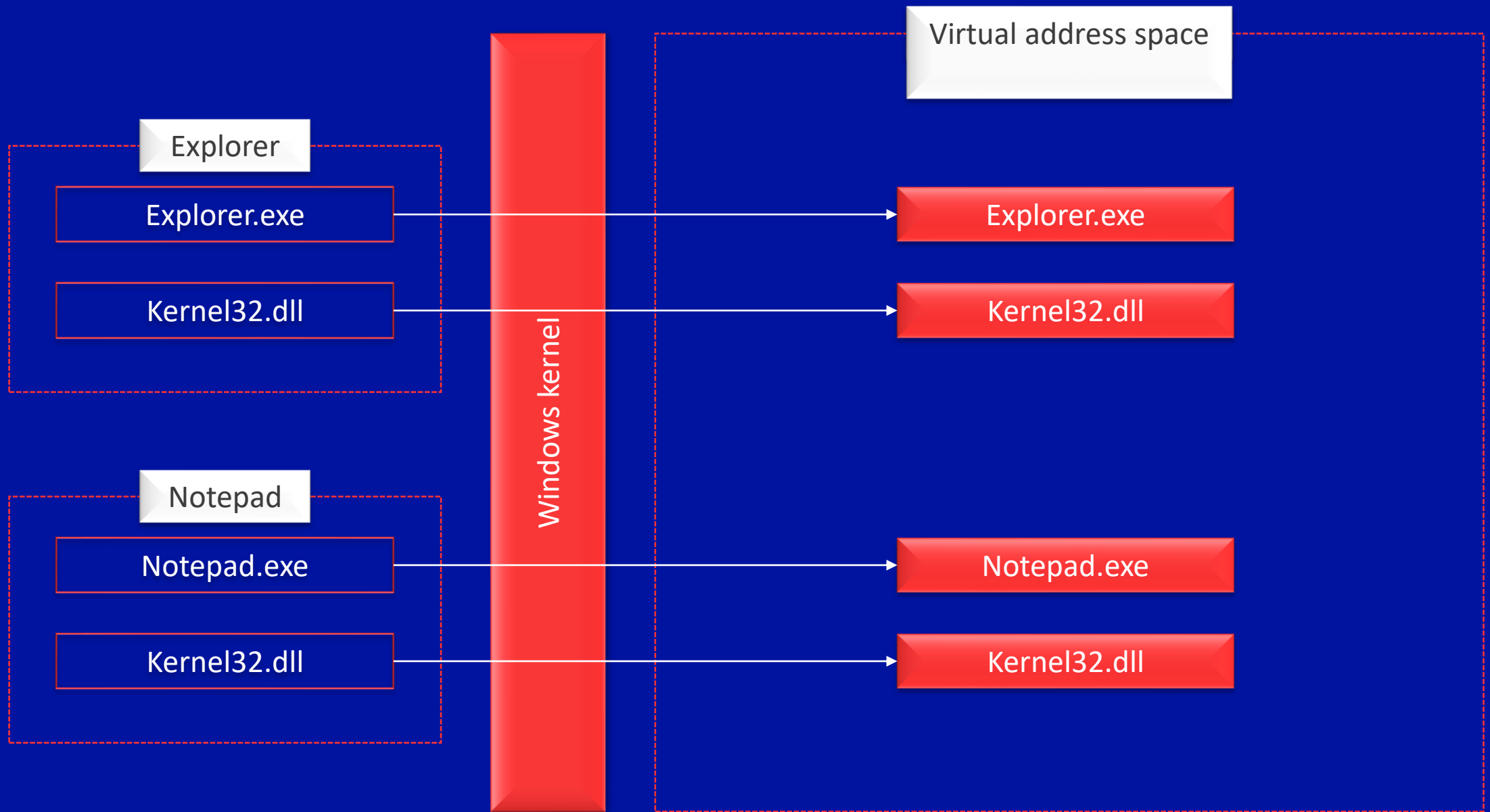
DETECTION

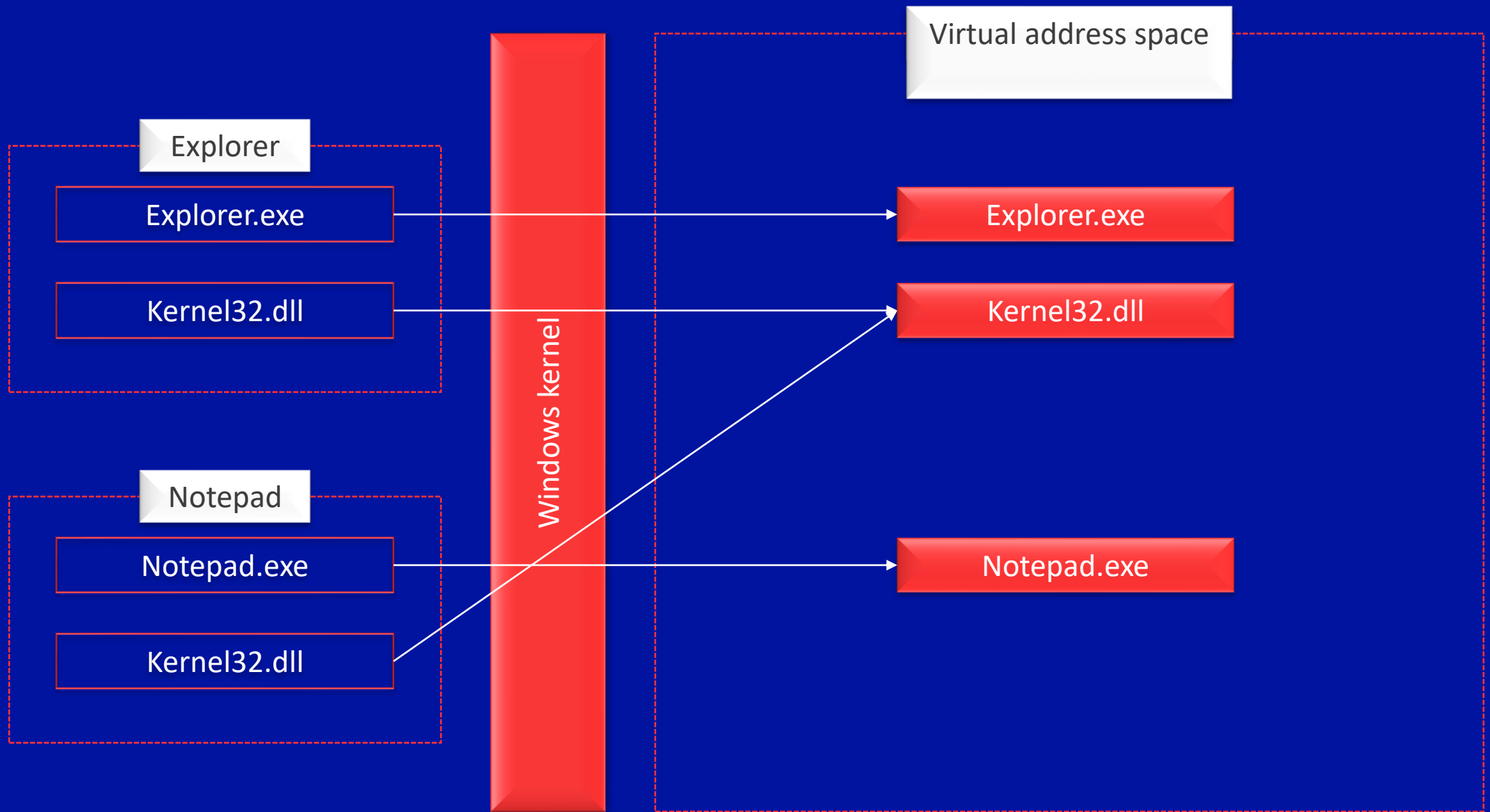
DETECTION

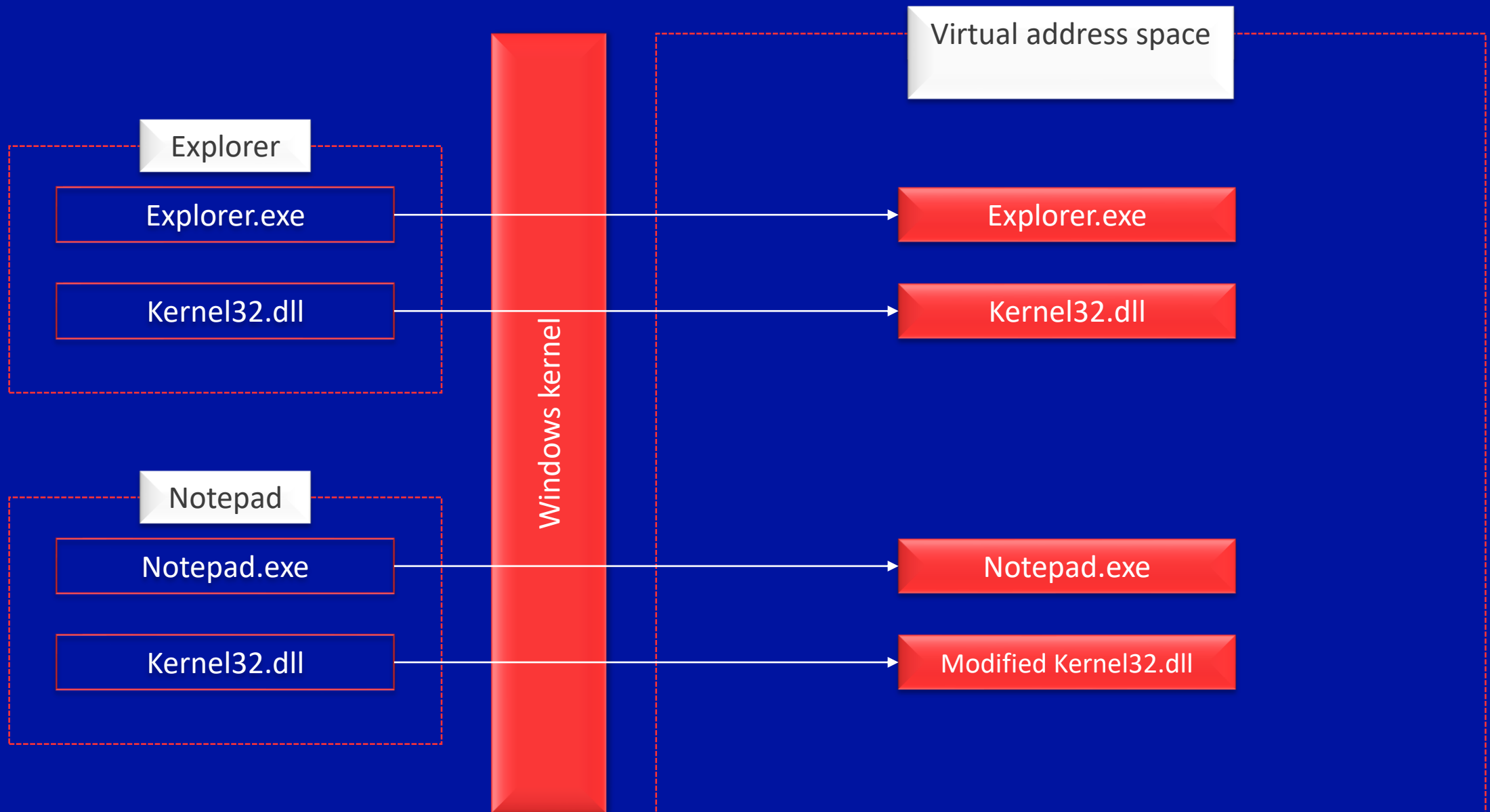
- Wouldn't it be nice if we could:
 - Detect in real-time
 - Fast enough for a background scanner
 - In a reasonably secure manner
 - Hardened against usermode attackers
 - Without needing debug symbols
 - Without needing the original module on disk

MEMORY MANAGEMENT THEORY









DETECTION

- Does Windows keep track of this information anywhere?
 - Yes it does!
 - It's in the kernel
 - But it's undocumented 😞
 - But it's pretty safe to access 😊
- The “PFN Database”
 - PFN meaning “Page Frame Number”
- A flat array of structs
- Windbg has “!pte” and “!pfn” commands

```

1: kd> !pte KERNEL32!BeepImplementation
                                VA 00007ff9fbbe2160
PXE at FFFF89C4E27137F8    PPE at FFFF89C4E26FFF38    PDE at FFFF89C4DFFE7EE8    PTE at FFFF89BFFCFDDDF10
contains 0A00000002C44867 contains 0A00000070EDB867 contains 0A0000006430F867    0000005098C025
pfn 2c44    ---DA--UWEV    pfn 70edb    ---DA--UWEV    pfn 6430f    ---DA--UWEV    pfn 5098c    ----A--UREV

1: kd> !pfn 5098c
  PFN 000    address FFFFE0000F1CA40
  flink      00000001    blink / share count 00000001    pteaddress FFFF89BFFCFDDDF10
  reference count 0001    used entry count 0000    Cached    color 0    Priority 5
  restore pte B3C1300002070    containing page 06430F    Active    M
  Modified

```

```

1: kd>

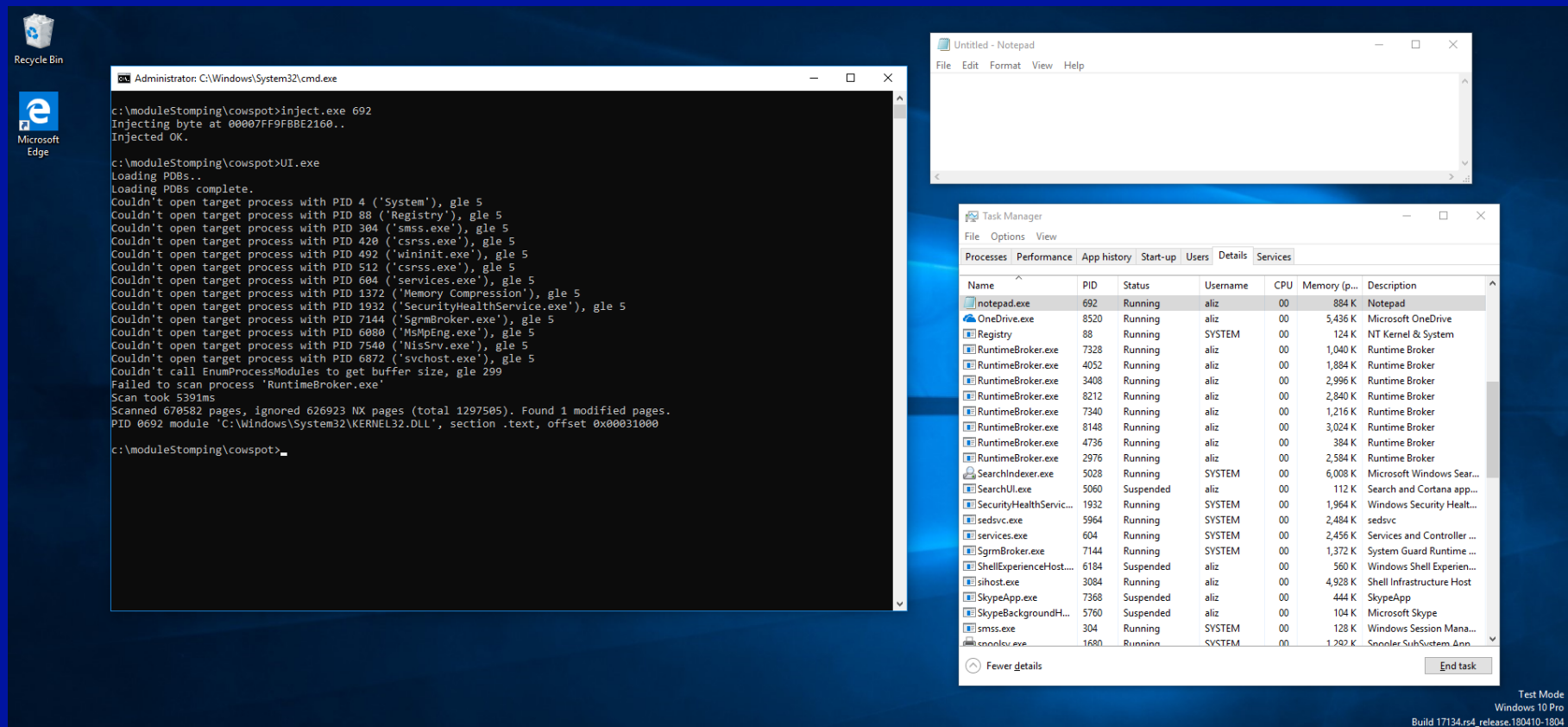
```

LIVE DETECTION

LIVE DETECTION

- All we need to do is query the PFN database
- It's in kernel space, so we write a simple driver to query it
- The address and structure is undocumented
 - Use debug symbols to acquire the address
 - "DIA SDK" is installed with VS, underutilised capability IMO!
 - This part done by userspace component
- Surprisingly easy to implement

LIVE DETECTION



The screenshot shows a Windows 10 desktop with a blue background. In the bottom-left corner, there is a Recycle Bin icon and a Microsoft Edge icon. The main area of the desktop is occupied by two windows.

The first window is a command prompt titled "Administrator: C:\Windows\System32\cmd.exe". It displays the following text:

```
c:\moduleStomping\cowspot>inject.exe 692
Injecting byte at 00007FF9FBBE2160..
Injected OK.

c:\moduleStomping\cowspot>UI.exe
Loading PDbs..
Loading PDbs complete.
Couldn't open target process with PID 4 ('System'), gle 5
Couldn't open target process with PID 88 ('Registry'), gle 5
Couldn't open target process with PID 304 ('smss.exe'), gle 5
Couldn't open target process with PID 420 ('csrss.exe'), gle 5
Couldn't open target process with PID 492 ('wininit.exe'), gle 5
Couldn't open target process with PID 512 ('csrss.exe'), gle 5
Couldn't open target process with PID 604 ('services.exe'), gle 5
Couldn't open target process with PID 1372 ('Memory Compression'), gle 5
Couldn't open target process with PID 1932 ('SecurityHealthService.exe'), gle 5
Couldn't open target process with PID 7144 ('SgrmBroker.exe'), gle 5
Couldn't open target process with PID 6080 ('MsMpEng.exe'), gle 5
Couldn't open target process with PID 7540 ('NisSrv.exe'), gle 5
Couldn't open target process with PID 6872 ('svchost.exe'), gle 5
Couldn't call EnumProcessModules to get buffer size, gle 299
Failed to scan process 'RuntimeBroker.exe'
Scan took 5381ms
Scanned 670582 pages, ignored 626923 NX pages (total 1297505). Found 1 modified pages.
PID 0692 module 'C:\Windows\System32\KERNEL32.DLL', section .text, offset 0x00031000

c:\moduleStomping\cowspot>
```

The second window is the Task Manager, showing the "Processes" tab. It lists the following processes:

Name	PID	Status	Username	CPU	Memory (p...)	Description
notepad.exe	692	Running	aliz	00	884 K	Notepad
OneDrive.exe	8520	Running	aliz	00	5,436 K	Microsoft OneDrive
Registry	88	Running	SYSTEM	00	124 K	NT Kernel & System
RuntimeBroker.exe	7328	Running	aliz	00	1,040 K	Runtime Broker
RuntimeBroker.exe	4052	Running	aliz	00	1,884 K	Runtime Broker
RuntimeBroker.exe	3408	Running	aliz	00	2,996 K	Runtime Broker
RuntimeBroker.exe	8212	Running	aliz	00	2,840 K	Runtime Broker
RuntimeBroker.exe	7340	Running	aliz	00	1,216 K	Runtime Broker
RuntimeBroker.exe	8148	Running	aliz	00	3,024 K	Runtime Broker
RuntimeBroker.exe	4736	Running	aliz	00	384 K	Runtime Broker
RuntimeBroker.exe	2976	Running	aliz	00	2,584 K	Runtime Broker
SearchIndexer.exe	5028	Running	SYSTEM	00	6,008 K	Microsoft Windows Sear...
SearchUI.exe	5060	Suspended	aliz	00	112 K	Search and Cortana app...
SecurityHealthServ...	1932	Running	SYSTEM	00	1,964 K	Windows Security Healt...
sedsv.exe	5964	Running	SYSTEM	00	2,484 K	sedsv
services.exe	604	Running	SYSTEM	00	2,456 K	Services and Controller ...
SgrmBroker.exe	7144	Running	SYSTEM	00	1,372 K	System Guard Runtime ...
ShellExperienceHost...	6184	Suspended	aliz	00	560 K	Windows Shell Experien...
shost.exe	3084	Running	aliz	00	4,928 K	Shell Infrastructure Host
SkypeApp.exe	7368	Suspended	aliz	00	444 K	SkypeApp
SkypeBackgroundH...	5760	Suspended	aliz	00	104 K	Microsoft Skype
smss.exe	304	Running	SYSTEM	00	128 K	Windows Session Mana...
smss.exe	1680	Running	SYSTEM	00	1,292 K	Smss.exe SubSystem Ann...

At the bottom right of the Task Manager window, there is a "Fewer details" button and an "End task" button. The Task Manager window also shows the "Performance" tab with a "Test Mode" warning: "Test Mode Windows 10 Pro Build 17134.rs4_release.180410-1804".

```
c:\moduleStomping\cowspot>inject.exe 692
```

```
Injecting byte at 00007FF9FBBE2160..
```

```
Injected OK.
```

```
c:\moduleStomping\cowspot>UI.exe
```

```
Loading PDBs..
```

```
Loading PDBs complete.
```

```
Couldn't open target process with PID 4 ('System'), gle 5
```

```
Couldn't open target process with PID 88 ('Registry'), gle 5
```

```
Couldn't open target process with PID 304 ('smss.exe'), gle 5
```

```
Couldn't open target process with PID 420 ('csrss.exe'), gle 5
```

```
Couldn't open target process with PID 492 ('wininit.exe'), gle 5
```

```
Couldn't open target process with PID 512 ('csrss.exe'), gle 5
```

```
Couldn't open target process with PID 604 ('services.exe'), gle 5
```

```
Couldn't open target process with PID 1372 ('Memory Compression'), gle 5
```

```
Couldn't open target process with PID 1932 ('SecurityHealthService.exe'), gle 5
```

```
Couldn't open target process with PID 7144 ('SgrmBroker.exe'), gle 5
```

```
Couldn't open target process with PID 6080 ('MsMpEng.exe'), gle 5
```

```
Couldn't open target process with PID 7540 ('NisSrv.exe'), gle 5
```

```
Couldn't open target process with PID 6872 ('svchost.exe'), gle 5
```

```
Couldn't call EnumProcessModules to get buffer size, gle 299
```

```
Failed to scan process 'RuntimeBroker.exe'
```

```
Scan took 5391ms
```

```
Scanned 670582 pages, ignored 626923 NX pages (total 1297505). Found 1 modified pages.
```

```
PID 0692 module 'C:\Windows\System32\KERNEL32.DLL', section .text, offset 0x00031000
```

```
c:\moduleStomping\cowspot>_
```

LIVE DETECTION

Administrator: Command Prompt

```
c:\code>C:\code\moduleStomping\cowspot\x64\Debug\UI.exe
```

```
Loading PDBs..
```

```
Loading PDBs complete.
```

```
Couldn't open target process with PID 4 ('System'), gle 5
```

```
Couldn't open target process with PID 120 ('Registry'), gle 5
```

```
Couldn't open target process with PID 616 ('smss.exe'), gle 5
```

```
Couldn't open target process with PID 712 ('csrss.exe'), gle 5
```

```
Couldn't open target process with PID 812 ('wininit.exe'), gle 5
```

```
Couldn't open target process with PID 820 ('csrss.exe'), gle 5
```

```
Couldn't open target process with PID 884 ('services.exe'), gle 5
```

```
Couldn't open target process with PID 2608 ('Memory Compression'), gle 5
```

```
Couldn't open target process with PID 3632 ('svchost.exe'), gle 5
```

```
Couldn't open target process with PID 5508 ('SecurityHealthService.exe'), gle 5
```

```
Couldn't open target process with PID 5568 ('SgrmBroker.exe'), gle 5
```

```
Couldn't open target process with PID 10708 ('svchost.exe'), gle 5
```

```
Scan took 4172ms
```

```
Scanned 1955628 pages (7639 MB), ignored 1133662 NX pages (7639 MB), totalling 3089290 pages (12067 MB). Found 130 modified pages (0.51 MB).
```

LIVE DETECTION

- 12GB RAM in use at time of scan
 - ~7GB executable
- Scan took 4.1 seconds
- Found 130 modified pages (~500KB)

FUTURE WORK

FUTURE WORK

- You can download and run the driver
- You can call it from your own C code
- Probably useful for more than just module stomping
 - Specific code patches can become signatures

```
"Name": "Generic_amsi_bypass":  
    modules = [ "amsi.dll" ]  
    offsets = [ 0x1234 ]
```

- It'd be great if this was useful to people

FUTURE WORK

- Temporary downsides
 - Driver is self-signed right now
 - You need to boot in Test Mode, which means disabling Secure Boot
 - Not heavily tested
 - It's in kernel space, it's possible it'll BSoD your box
 - Pretty unlikely because the kernel component is so thin and simple
- These problems will go away

FUTURE WORK

- Inherent downsides
 - We need to pull down symbols to locate the PFN database
 - We can build a database of offsets and build it into the binary if need be
 - Passing location of PFN to driver introduces a security weakness
 - We depend on PFN not changing
 - There are false positives
 - AVs that patch things, JIT engines, ...
 - Uncommon enough that we can memcpy these ranges

SUMMARY

SUMMARY

- We can now scan for modified file-backed memory in real-time
 - So can you
- Use it to detect patches in code quickly
- Code is available online
- Blogpost is available online
 - In three parts – first two all about the attack, third about detection
- Please do things with it so I can justify making it better

<https://blog.f-secure.com/category/threats-research>

<https://github.com/countercept/ModuleStomping>

@AlizTheHax0r



F-Secure®