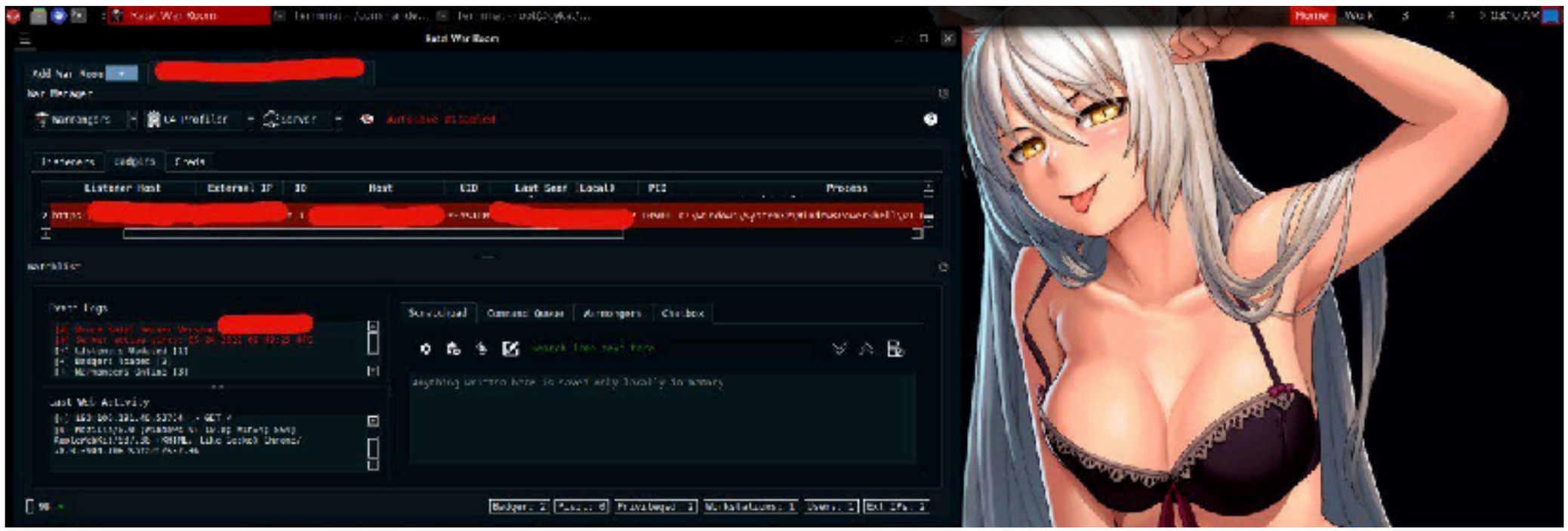

how i met your beacon
A LOVE STORY IN REVERSE.
SLIVER
FRAMEWORK
MATRIX

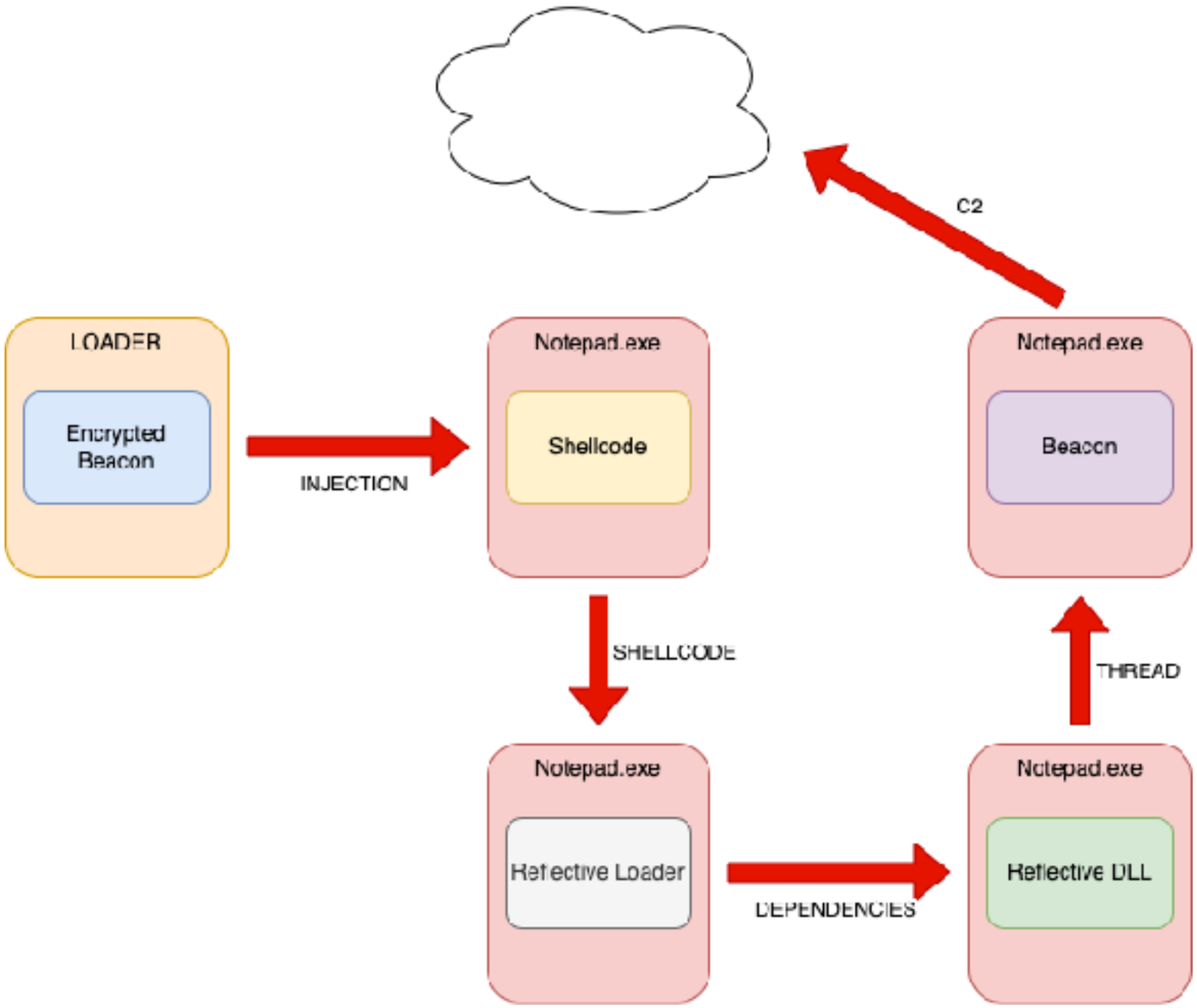
- Effective strategies for generic and targeted beacon detection, including techniques based on:
  - Behaviour,
  - In-memory analysis,
  - Network footprint
- Development of an open source BeaconHunter to incorporate these techniques
- Case studies of commercial and open source c2 frameworks - more in the blog posts!

- Cobalt Strike, commercial C2 developed by HelpSystems (https://www.cobaltstrike.com/)
  - Analysis performed on v4.6.1 (latest)
  - Popular among red teams and threat actors alike
  - Highly customisable and stable

Mandiant has observed FIN12 use a broad toolset that included the Powershell-based EMPIRE framework and the TRICKBOT banking Trojan. However, since February 2020, FIN12 has used Cobalt Strike BEACON payloads in nearly every one of its intrusions, from internal reconnaissance to ransomware deployment.

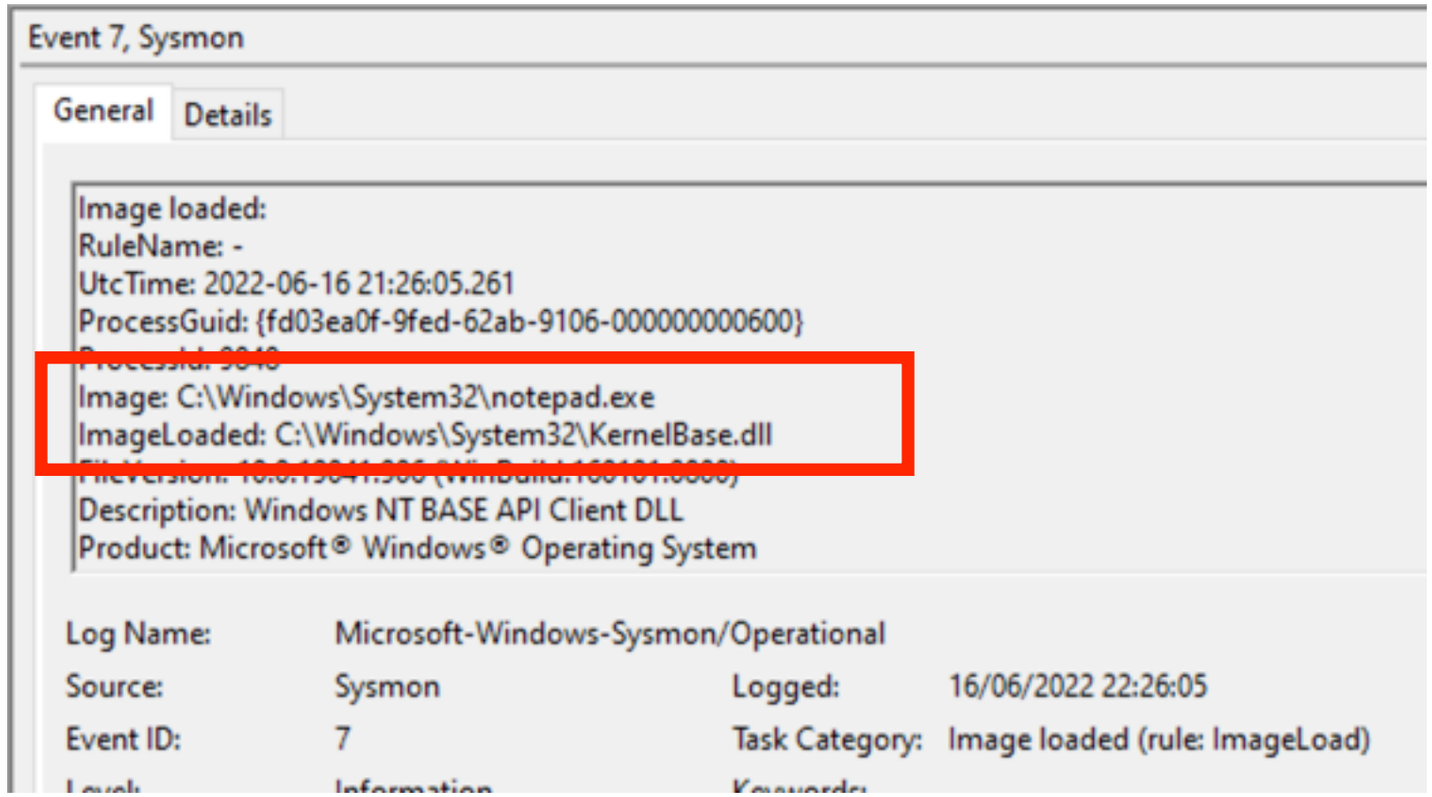
- Brute Ratel (BRC4), commercial C2 developed by Dark Vortex (https://bruteratel.com/)
  - Less popular but used by some actors
  - Analysis performed on v1.0.7 (latest)




C2
LOADER
Encrypted Beacon
INJECTION
Notepad.exe
Shellcode
Notepad.exe
Beacon
SHELLCODE
THREAD
Notepad.exe
Reflective Loader
DEPENDENCIES
Notepad.exe
Reflective DLL

- The behaviour of a beacon can give rise to detection opportunities
- Commercial frameworks are closed source and behaviours cannot be trivially changed
- Predetermined behaviour presents detection opportunities, two examples:
  - Image load events
  - Named pipes

- Beacons often rely on libraries native to the OS, keeping the beacon small
- Image load events provide telemetry:



- Image load telemetry provides a hunting opportunity:
  - Analysis shows some beacon frameworks load all post-ex dependencies on start
  - Egress beacons typically load `winhttp` or `wininet` DLLs
  - Anomalies for processes loading irregular DLLs, e.g. `notepad.exe` loading `dbghelp.dll`

- Event Query Language can be used to build signatures around image loads to hunt for beacons:

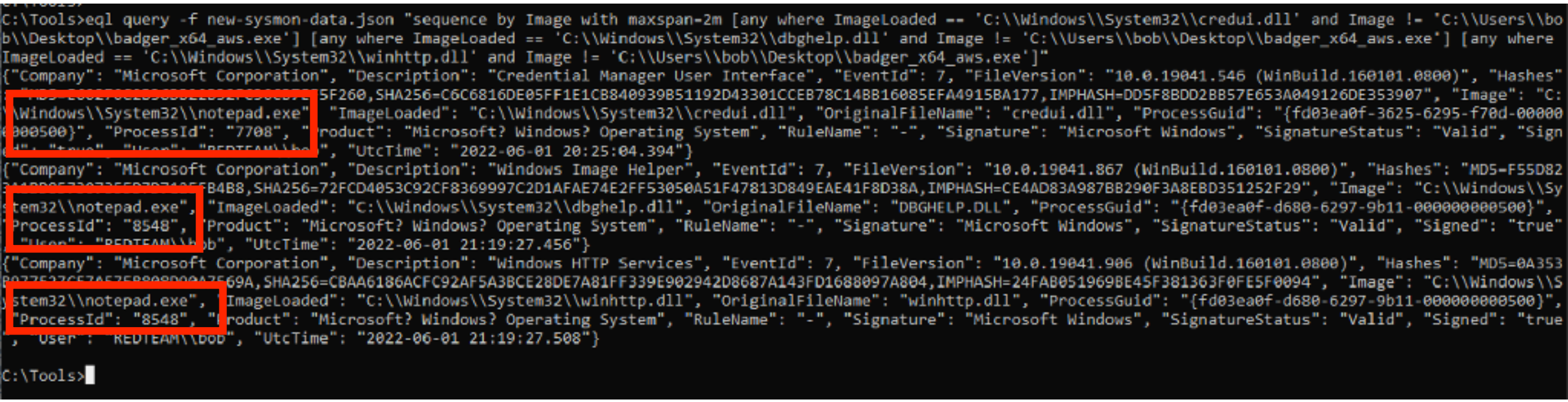
```
sequence by Image with maxspan=1m
   [any where ImageLoaded == 'C:\\Windows\\System32\\credui.dll']
   [any where ImageLoaded == ‘C:\\Windows\\System32\\winhttp.dll']
```

- Injecting Brute Ratel in to a process:

Time of Day	Process Name	PID	Operation	Path	Result	Detail
22:19:27.3294254	notepad.exe	8548	Load Image	C:\Windows\System32\cryptsp.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.3405444	notepad.exe	8548	Load Image	C:\Windows\System32\cryptbase.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.3472687	notepad.exe	8548	Load Image	C:\Windows\System32\cryptdll.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.3632003	notepad.exe	8548	Load Image	C:\Windows\System32\crypt32.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.3652582	notepad.exe	8548	Load Image	C:\Windows\System32\psapi.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4036815	notepad.exe	8548	Load Image	C:\Windows\WinSxS\amd64_microsoft.windows.gdi...	SUCCESS	Image Base: 0x7ffd...
22:19:27.4079732	notepad.exe	8548	Load Image	C:\Windows\System32\netapi32.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4107073	notepad.exe	8548	Load Image	C:\Windows\System32\samcli.dll	SUCCESS	Image Base: 0x7ffd
22:19:27.4133385	notepad.exe	8548	Load Image	C:\Windows\System32\logoncli.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4184777	notepad.exe	8548	Load Image	C:\Windows\System32\netutils.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4229742	notepad.exe	8548	Load Image	C:\Windows\System32\srvcli.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4252674	notepad.exe	8548	Load Image	C:\Windows\System32\olc32.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4323592	notepad.exe	8548	Load Image	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	Image Base: 0x7ffd...
22:19:27.4418068	notepad.exe	8548	Load Image	C:\Windows\System32\msasn1.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4467424	notepad.exe	8548	Load Image	C:\Windows\System32\secur32.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4532209	notepad.exe	8548	Load Image	C:\Windows\System32\sspicli.dll	SUCCESS	Image Base: 0x7ffd
22:19:27.4618411	notepad.exe	8548	Load Image	C:\Windows\System32\wtsapi32.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4650746	notepad.exe	8548	Load Image	C:\Windows\System32\dbghelp.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4691740	notepad.exe	8548	Load Image	C:\Windows\System32\version.dll	SUCCESS	Image Base: 0x7ffd
22:19:27.4718988	notepad.exe	8548	Load Image	C:\Windows\System32\dnsapi.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4741019	notepad.exe	8548	Load Image	C:\Windows\System32\iri.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4785349	notepad.exe	8548	Load Image	C:\Windows\System32\credui.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4840403	notepad.exe	8548	Load Image	C:\Windows\System32\wininet.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.4956147	notepad.exe	8548	Load Image	C:\Windows\System32\iertutil.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.5000154	notepad.exe	8548	Load Image	C:\Windows\System32\prnfapi.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.5139270	notepad.exe	8548	Load Image	C:\Windows\System32\OnDemandConnRouteHelpe...	SUCCESS	Image Base: 0x7ffd...
22:19:27.5166682	notepad.exe	8548	Load Image	C:\Windows\System32\winhttp.dll	SUCCESS	Image Base: 0x7ffd
22:19:27.5201859	notepad.exe	8548	Load Image	C:\Windows\System32\mswsock.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.5254157	notepad.exe	8548	Load Image	C:\Windows\System32\winrnr.dll	SUCCESS	Image Base: 0x7ffd...
22:19:27.7512557	notepad.exe	8548	Load Image	C:\Windows\System32\bcrypt.dll	SUCCESS	Image Base: 0x7ffd...
22:13:37.0402196	notepad.exe	8548	Load Image	C:\Windows\System32\notepad.exe	SUCCESS	Image Base: 0x7ffd...
22:13:37.0518518	notepad.exe	8548	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ffd...
22:13:37.0548401	notepad.exe	8548	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7ffd...
22:13:37.0561712	notepad.exe	8548	Load Image	C:\Windows\System32\KernelBase.dll	SUCCESS	Image Base: 0x7ffd...
22:13:37.0746007	notepad.exe	8548	Load Image	C:\Windows\System32\gdi32.dll	SUCCESS	Image Base: 0x7ffd...

- Using EQL we can trivially detect and BRc4 loads, including injected beacons:

```
sequence by Image with maxspan=1m [any where ImageLoaded == 'C:\
\Windows\\System32\\credui.dll'] [any where ImageLoaded == 'C:\
\Windows\\System32\\dbghelp.dll'] [any where ImageLoaded == ‘C:\
\Windows\\System32\\winhttp.dll']
```



- Beacons typically remain memory resident to avoid on disk detections
- Signatures provide a simple, yet effective means of detecting in-memory malware
- Signatures can be be built for content from known beacons
- Yara rules can be built to scan memory for known indicators[1]

1 https://www.elastic.co/blog/detecting-cobalt-strike-with-memory-signatures

```
rule helloworld
{
meta:
    author = "domchell"
    description = “Simple yara rule"
strings:
    $a = "hello"
    $b = "x33fcon"
    $c = “2022”
condition:
    2 of them
}
```

- Some beacons like Cobalt Strike offer techniques to evade these detections:

```
strrep "ReflectiveLoader" “RunFunc";
strrep "This program cannot be run in DOS mode" "";
strrep "beacon.dll" "";
```

- Instead of strings, code from the rDLL can be fingerprinted
- Sleepmask kit introduced in CS4.4 allows the user to control how the beacon obfuscates and sleeps

- Cobalt Strike offers three configurations for sleep obfuscation:
  - No sleep mask: the beacon, its strings and code will remain plaintext in memory
  - `sleep_mask=true`: obfuscate and sleep strategy masks strings and data using xor
  - User defined sleep mask: can obfuscate the beacon sections and heap records

- With a user defined sleep mask, but `userwx=false`:

```
C:\Tools\yara-v4.2.1-1934-win64>
C:\Tools\yara-v4.2.1-1934-win64>yara64.exe --print-strings sleepmask_norwx.yara 3220
CobaltStrike_sleepmask 3220
0x2c3a6270000:$sleep_mask: 48 8B C4 48 89 58 08 48 89 68 10 48 89 70 18 48 89 78 20 45 33 DB 45 33 D2 3
0F 84 81 00 00 00 0F B6 45

C:\Tools\yara-v4.2.1-1934-win64>
```

- The `.text` section cannot be obfuscated due to it being `PAGE_EXECUTE_READ`, leaving code ripe for signature

- With a user defined sleep mask and `userwx=true`:

0x69b1cec000	Private: Commit	12 kB	RW+G	Stack (thread 9672)
0x203b4e50000	Private: Commit	316 kB	RWX	
0x7ff674871000	Image: Commit	148 kB	RX	C:\Windows\System32\notepad.exe

notepad.exe (9604) (0x203b4e50000 - 0x203b4e9f000)

```
00000000 fa 45 55 e3 9e 58 2b 50 3b a6 81 e3 9e 70 3b 50 3b b6 b1 ee 24 db 66 2b 60 fd 6e 98 e1 48 a8 f1 .EU..X+P;....p;P;...$.f+`.n..H..
00000020 09 cd 91 ab 17 85 f1 17 36 4f 91 ab 17 0f 95 5d b2 86 1c a6 df 7b 20 18 38 c2 99 2b ee ff 57 79 ........6O.....]....{ .8..+..Wy
00000040 32 37 6f de 1a 32 ea e7 79 b7 9d 13 10 00 23 18 59 af 12 50 14 75 d7 59 73 2d 97 a4 a1 c1 dc df 27o..2..y.....#.Y..P.u.Ys-......
00000060 f6 c5 49 28 e8 04 56 21 f3 43 95 b1 56 3b 22 6f e5 8f 1a 60 d6 e9 33 5b 3a c2 93 ea e8 c2 a0 e3 ..I(..V!.C..V;"o...`..3[:.......
00000080 b3 b0 9c ea 9c cb e2 f1 ba 8d 19 a7 15 41 dc da 31 35 93 d5 10 47 ab 04 b0 8f 6e 69 52 33 f8 2b .............A..15...G....niR3.+
000000a0 4d 31 57 e3 e8 c5 18 ea c0 4d 14 54 62 9d 66 91 a3 fd 51 e3 9c 5c 07 10 fa 45 fd 8f 07 48 a8 6c M1W......M.Tb.f...Q..\...E...H.l
000000c0 96 d6 d9 20 6b 24 03 db 0a c8 91 ab 17 eb c7 d4 fa 47 cd 8f 1f 48 aa 74 96 d6 d9 22 63 24 03 4f ... k$..........G...H.t..."c$.O
000000e0 f3 9a d0 fe 56 56 62 4f fa 4d 7d 8b 52 33 c7 5d 81 38 a2 70 5a 8b cb 93 48 82 1a 52 9c c2 aa 4c ....VVbO.M}.R3.].8.pZ...H..R...L
```

Command Prompt

```
C:\Tools\yara-v4.2.1-1934-win64>
C:\Tools\yara-v4.2.1-1934-win64>
C:\Tools\yara-v4.2.1-1934-win64>yara64.exe --print-strings sleepmask_norwx.yara 9604

C:\Tools\yara-v4.2.1-1934-win64>
```

- The documentation indicates Brute Ratel has a complex obfuscate and sleep strategy:

New Encrypt and Sleep Mechanisms

In the release v0.7, BRc4 introduced Encrypting of the RX region and sleeping with the use of ROP gadgets and APCs which used the method found by Austin Hudson. However, upon further research, multiple other techniques were found which utilize Windows Event Creation, Wait Objects and Timers. Badger now comes with multiple anti-detection sleeping techniques, such as not using the usual Sleep API, encrypting the RX region with and without using ROP gadgets, and various different types of Wait Object Events and Timers to hide the badger during sleep. Each of these sleeping techniques are a part of all the badgers and the techniques are randomly switched everytime they go to sleep to avoid detection.

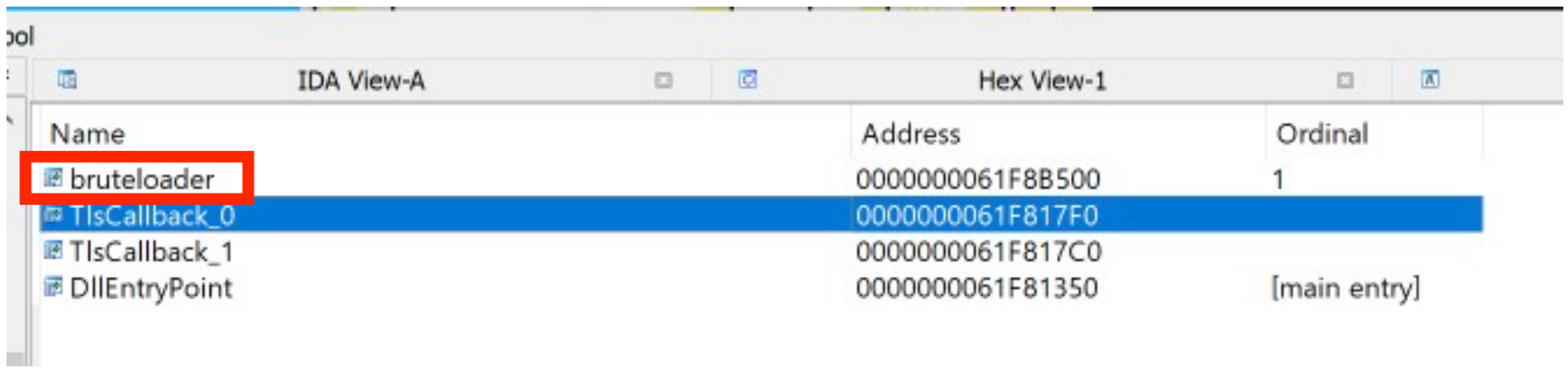
- Consequently, the beacon should be more complex to identify in memory?

- Sleeping the badger and recovering strings is a little surprising...

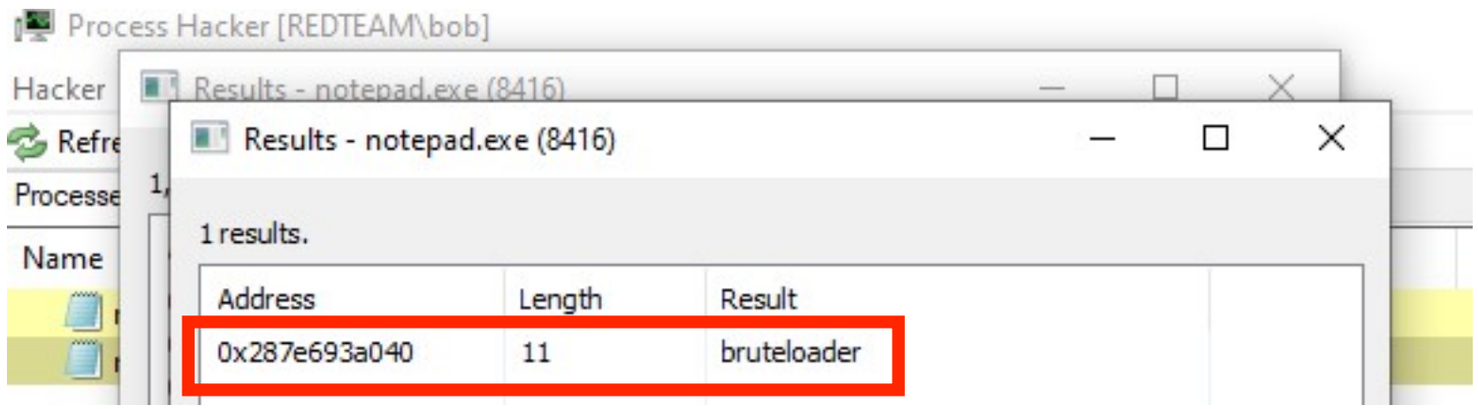
13 results.

Address	Length	Result
0x1bc789705cc	50	[+] AMSI and ETW patched
0x1bc7897067c	34	[+] Patched AMSI
0x1bc789706a0	50	[-] Unable to patch AMSI
0x1bc789706d4	62	[+] AMSI patching not required
0x7ff690d6ce35	7	amsiPI

- Reversing the rDLL we can identify the loader export:

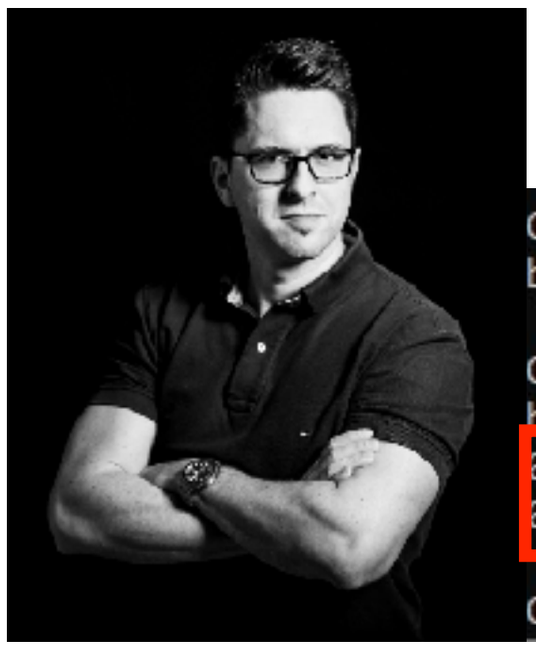


- We oddly find these strings in the memory of a sleeping badger:



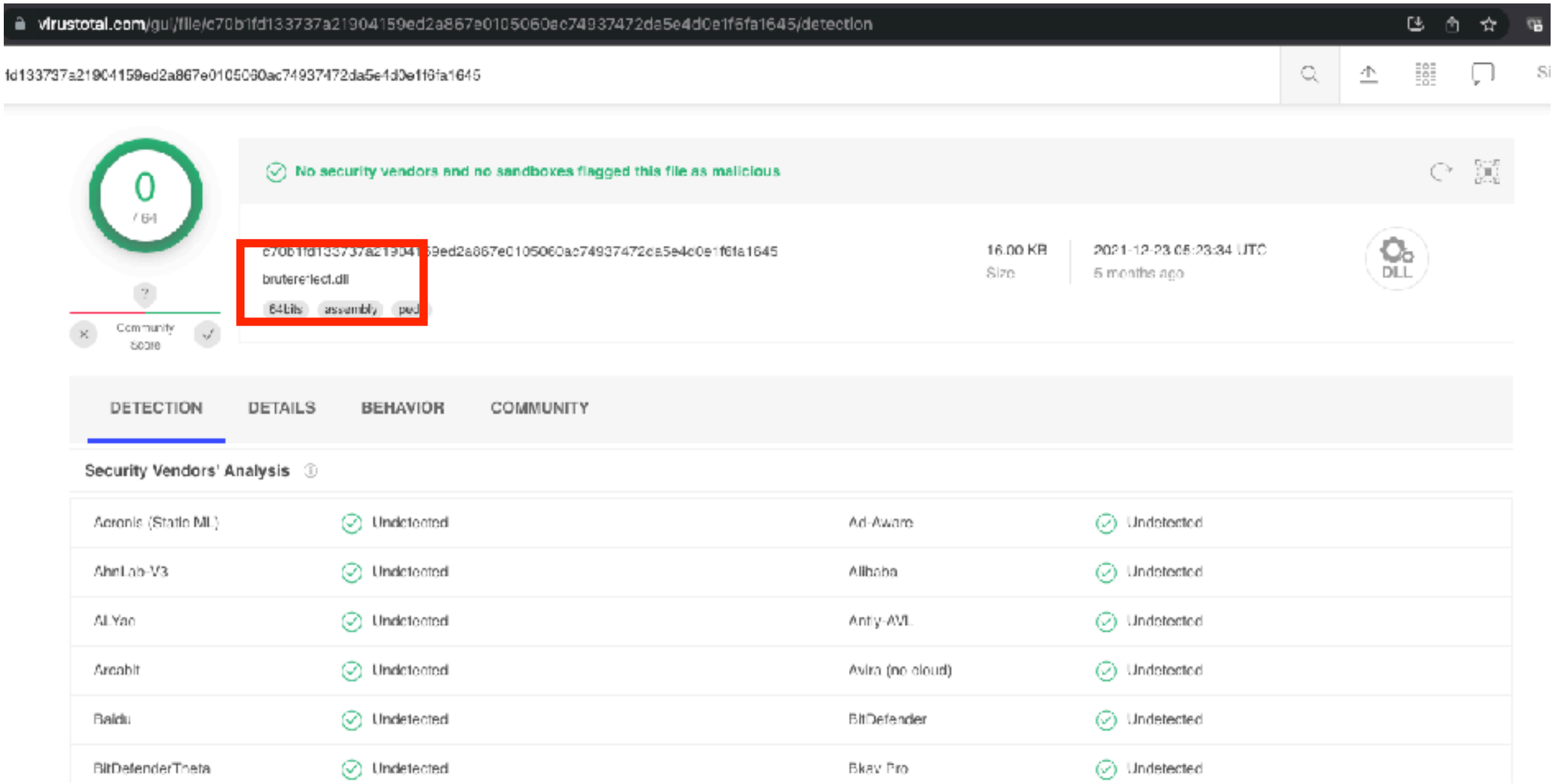
- Yara rules can spot all running BRc4 instances:

```
rule brc4_badger_strings
{
meta:
    author = "@domchell"
    description = "Identifies strings used in Badger rDLL"
strings:
    $a = "bruteloader"
    $b = "bhttp_x64.dll"
condition:
    1 of them
}
```

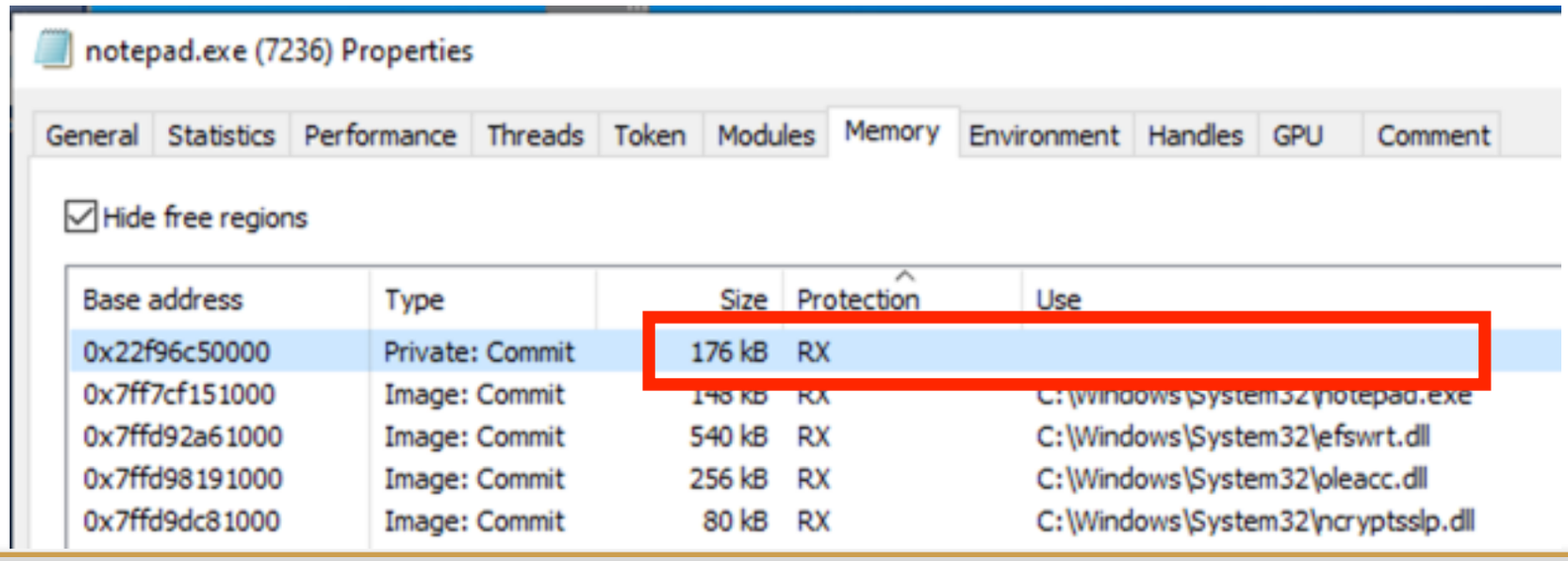


```
C:\Tools\yara-v4.2.1-1934-win64>yara64.exe z:\brc4.yara 9700
brc4_badger_strings 9700

C:\Tools\yara-v4.2.1-1934-win64>yara64.exe --print-strings z:\brc4.yara 9700
brc4_badger_strings 9700
0x1de73dca040:$a: bruteloader
0x1de73dca032:$b: bhttp_x64.dll

C:\Tools\yara-v4.2.1-1934-win64>
```

- Can we use this to find VT samples?....



- The beacon will typically be operating from either virtual memory or a stomped DLL
- If running from virtual memory, the memory must be executable for the beacon to operate, creating a detection opportunity
- Irregular to see RX or RWX unmapped pages



- Hunting for unmapped executable memory, with caution of CLR modules, can prove fruitful

- Some beacons will use event driven execution to shuffle the page permissions when sleeping

- Various public techniques exist to achieve this, typically involve using ROP with asynchronous events such as APCs, or event timers, or triggering a VEH on an access violation

In Memory Detections: Page Permissions

Base address	Size	Protect...	Use	Total WS	Private WS
0x20835340000	252 kB	RW		248 kB	184 kB
0x20835340000	4 kB	RW		4 kB	
0x20835341000	156 kB	RX		156 kB	156 kB
0x20835368000	8 kB	RW		8 kB	4 kB
0x2083536a000	52 kB	R		52 kB	8 kB
0x20835377000	12 kB	RW		12 kB	12 kB
0x2083537a000	4 kB	R		4 kB	

Performing tasking

Sleeping

Base address	Type	Size	Protect...	Use	Total WS	Private WS
0x20835340000	Private	252 kB	RW		248 kB	184 kB
0x20835340000	Private: Commit	168 kB	RW		168 kB	160 kB
0x2083536a000	Private: Commit	52 kB	R		52 kB	8 kB
0x20835377000	Private: Commit	12 kB	RW		12 kB	12 kB
0x2083537a000	Private: Commit	4 kB	R		4 kB	
0x2083537b000	Private: Commit	12 kB	RW		8 kB	4 kB
0x2083537e000	Private: Commit	4 kB	R		4 kB	
0x208353b0000	Private	64 kB	RW	Heap (ID 4)	40 kB	40 kB
0x208353c0000	Mapped	3,296 kB	R	C:\Windows\Globalizati...	36 kB	

- The page permissions of Cobalt Strike are always executable:
  - Pages remain either RX or RWX
  - Without module stomping will be backed by unmapped memory

Type	Size	Protect...	Use
Image: Commit	44 kB	RX	C:\Windows\System32\OnDemandConnRouteHelper.dll
Image: Commit	1,996 kB	RX	C:\Windows\System32\wininet.dll
Image: Commit	80 kB	RX	C:\Windows\System32\ncryptsslp.dll
Image: Commit	44 kB	RX	C:\Windows\System32\mskeyprotect.dll
Image: Commit	256 kB	RX	C:\Windows\System32\oleacc.dll
Image: Commit	540 kB	RX	C:\Windows\System32\efswrt.dll
Image: Commit	148 kB	RX	C:\Windows\System32\notepad.exe
Private: Commit	176 kB	RX	

- Brute Ratel does shuffle its page permissions when sleeping:

notepad.exe (1684) Properties

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

Hide free regions — Strings... — Refresh

Base address	Type	Size	Protect...	Use
0x304b3ec000	Private: Commit	12 kB	RW+G	Stack (thread 6624)
0x21586f01000	Private: Commit	156 kB	RX	
0x7ff7548d1000	Image: Commit	148 kB	RX	C:\Windows\System32\notepad.exe
0x7ff835f21000	Image: Commit	1,304 kB	RX	C:\Windows\WinSxS\amd64_microsoft.windows.gdi
0x7ff83b2a1000	Image: Commit	540 kB	RX	C:\Windows\System32\efswrt.dll
0x7ff83b4a1000	Image: Commit	256 kB	RX	C:\Windows\System32\pleacc.dll

notepad.exe (1684) Properties

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

Hide free regions — Strings... — Refresh

Base address	Type	Size	Protect...	Use
0x21586ef0000	Mapped: Commit	4 kB	RW	
0x21586f00000	Private: Commit	168 kB	RW	
0x21586f37000	Private: Commit	12 kB	RW	

- ...unless you have a p2p badger linked:

Listeners | Badgers | Creds

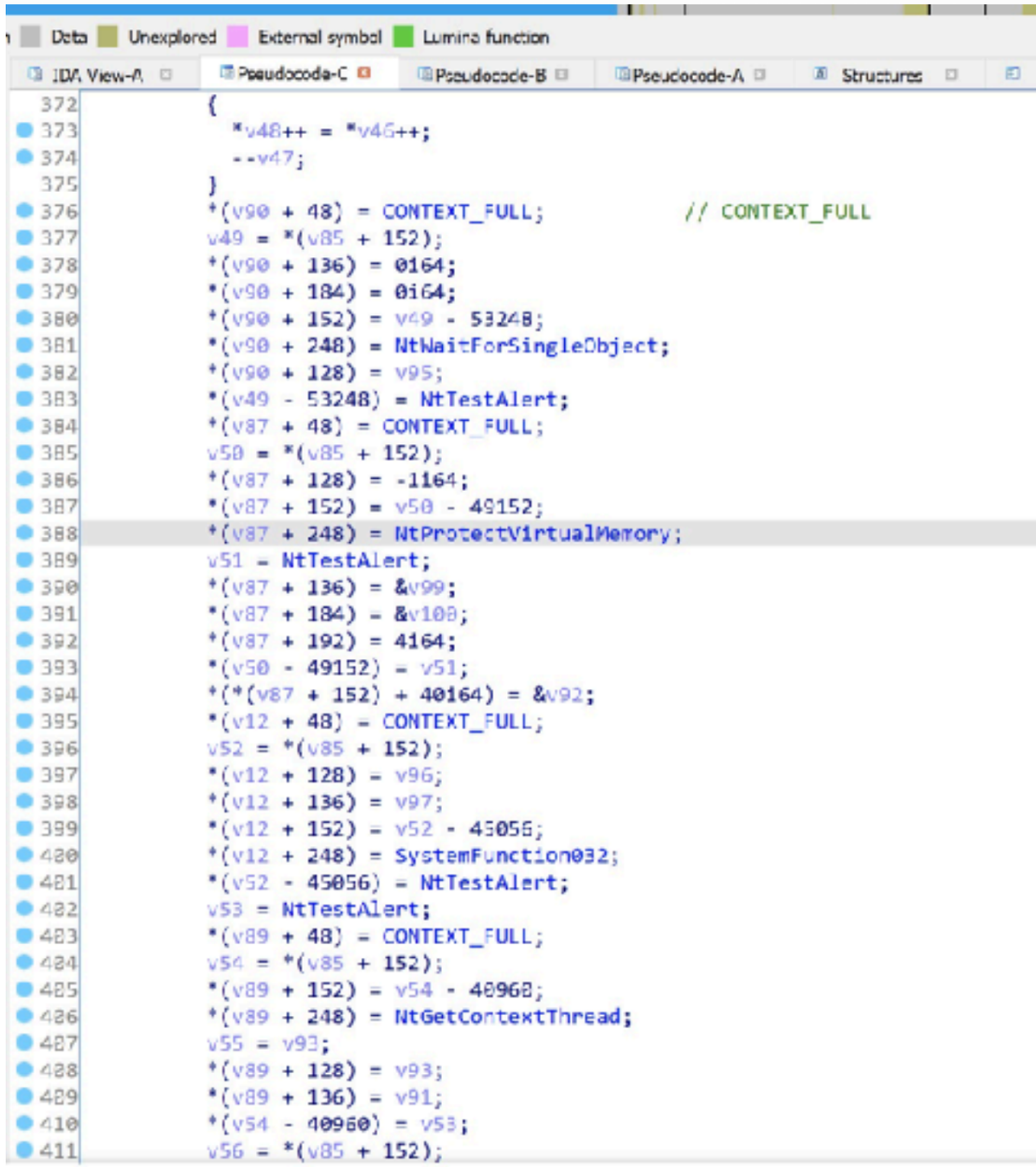
	Listener ID	Listener Host	External IP	ID	Host	UID	Last Seen (Local)	PID	Process	Arch/OS (Build)	Payload Arch	ivot Strea
1	http	http://172.31.30.191:80	[redacted]	b-3	[redacted]				\badger_x64_aws.exe	x64/10.0 (19043)	x64	Direct
2	http	http://172.31.30.191:80	[redacted]	b-7	[redacted]				\notepad.exe	x64/10.0 (19043)	x64	b-3-<>-b-7

0xf3f4bfa000	Private: Commit	12 kB	RW+G	Stack (thread 5368)				
0xf3f50fb000	Private: Commit	12 kB	RW+G	Stack (thread 10112)				
0x1ff82d11000	Private: Commit	156 kB	RX		156 kB	156 kB		
0x7ff6d8581000	Image: Commit	288 kB	RX	C:\Users\bob\Desktop\badger_x64_...	288 kB		288 kB	288 kB
0x7ffdc2291000	Image: Commit	1,320 kB	RX	C:\Windows\WinSxS\amd64_microso...	20 kB		20 kB	20 kB
0x7ffdc6591000	Image: Commit	1,996 kB	RX	C:\Windows\System32\wininet.dll	756 kB		756 kB	756 kB
0x7ffdcb2e1000	Image: Commit	44 kB	RX	C:\Windows\System32\netapi32.dll	8 kB		8 kB	8 kB
0x7ffdce481000	Image: Commit	56 kB	RX	C:\Windows\System32\srvcli.dll	12 kB		12 kB	12 kB
0x7ffdce621000	Image: Commit	60 kB	RX	C:\Windows\System32\samcli.dll	16 kB		16 kB	16 kB
0x7ffdcee41000	Image: Commit	12 kB	RX	C:\Windows\System32\version.dll	4 kB		4 kB	4 kB
0x7ffdceeb1000	Image: Commit	1,368 kB	RX	C:\Windows\System32\dbghelp.dll	60 kB		60 kB	60 kB

- First a new thread with the start address spoofed (`TpReleaseCleanupGroupMembers+550`) is created:

```
Data  Unexplored  External symbol  Lumina function
IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A | Structures | Enums
121   sub_61F8B090(v3, NtTestAlert);
122   sub_61F8B090(v3, NtGetContextThread);
123   sub_61F8B090(v3, NtSetContextThread);
124   sub_61F8B090(v3, NtWaitForSingleObject);
125   sub_61F8B090(v3, NtProtectVirtualMemory);
126 }
127 v4 = *(*(__readgsqword(0x30u) + 96) + 16i64);
128 v5 = *(v4 + 60);
129 LODWORD(TpReleaseCleanupGroupMembers) = HashLookup(0x77D0E3B5, v3);
130 if ( TpReleaseCleanupGroupMembers )
131   ThreadStartAddress = TpReleaseCleanupGroupMembers + 0x550;
132 else
133   ThreadStartAddress = v4 + *(v5 + v4 + 40);
134 if ( NtCreateEvent_1 )
135   result = sub_61F81491(&v95, 2031619, 0, 1, 0, NtCreateEvent_1);
136 else
137   result = NtCreateEvent(&v95, 2031619i64, 0i64, 1i64, 0);
138 if ( result < 0
139   || (!NtCreateThreadEx_1 ? (result = NtCreateThreadEx(
140                                         &v94,
141                                         2032639i64,
142                                         0i64,
143                                         -1i64,
144                                         ThreadStartAddress,
145                                         0i64,
146                                         1,
147                                         0i64,
148                                         81920i64,
149                                         81920i64,
150                                         0i64)) : (result = sub_61F81410(
151                                                       &v94,
152                                                       2032639,
153                                                       0,
154                                                       -1,
155                                                       ThreadStartAddress,
156                                                       0,
157                                                       1,
158                                                       0,
159                                                       81920,
160                                                       81920,
```

- A series of context structures are then created for a number of function calls, to `NtWaitForSingleObject`, `NtProtectVirtualMemory`, `SystemFunction032`, `NtGetContextThread` and `SetThreadContext`:



- APCs queued against `NtContinue` to proxy calls to context structures
- Appears to be Foliage[1]

```
Data  Unexplored  External symbol  Lumina function
IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A | Structures | Enums | Imports
474                 if ( sub_61F8141B(v94, NtContinue, v89, 0i64, 0, v77) >= 0 )
475                 {
476                   LODWORD(v78) = NtQueueApcThread;
477                   if ( sub_61F8141B(v94, NtContinue, v88, 0i64, 0, v78) >= 0 )
478                   {
479                     LODWORD(v79) = NtQueueApcThread;
480                     if ( sub_61F8141B(v94, NtContinue, v86, 0i64, 0, v79) >= 0 )
481                     {
482                       LODWORD(v80) = NtQueueApcThread;
483                       if ( sub_61F8141B(v94, NtContinue, v14, 0i64, 0, v80) >= 0 )
484                       {
485                         LODWORD(v81) = NtQueueApcThread;
486                         if ( sub_61F8141B(v94, NtContinue, v10, 0i64, 0, v81) >= 0 )
487                         {
488                           LODWORD(v82) = NtQueueApcThread;
489                           if ( sub_61F8141B(v94, NtContinue, v9, 0i64, 0, v82) >= 0 )
490                           {
491                             LODWORD(v83) = NtQueueApcThread;
492                             v68 = sub_61F8141B(v94, NtContinue, v11, 0i64, 0, v83);
493 LABEL_76:
494                             if ( v68 >= 0 )
495                             {
496                               v69 = NtAlertResumeThread_1 ? sub_61F8142F(v94, 0i64) : NtAlertResumeTh
497                               if ( v69 >= 0 )
498                               {
499                                 v70 = *WaitForSingleObjectEx;
500                                 *(v13 + 48) = 1048587;
501                                 v71 = v94;
502                                 v72 = v95;
503                                 *(v13 + 248) = v70;
504                                 *(v13 + 152) = *(__readgsqword(0x30u) + 8);
505                                 if ( NtSignalAndWaitForSingleObject_1 )
506                                 {
507                                   LODWORD(v73) = NtSignalAndWaitForSingleObject_1;
508                                   sub_61F8149C(v72, v71, 0i64, 0i64, v73);
509                                 }
510                                 else
511                                 {
512                                   NtSignalAndWaitForSingleObject(v72, v71, 0i64, 0i64);
513                                 }
```

1 https://github.com/SecIdiot/FOLIAGE

- Beacons (or the operator) often apply hooks to certain functions to circumvent controls, such as:
  - ETW
  - AMSI
- Hunting for these patches can reveal hidden beacons

github.com/sliverermory/injectEtwBypass/blob/main/injectEtwBypass

```
216         //__debugbreak();
217         // nt.NtOpenProcess(&hProc, 0x1FFFFF, &oa, &cid);
218         HellsGate(nt.NtOpenProcessSyscall);
219         HellDescent(&hProc, 0x1FFFFF, &oa, &cid);
220         // ETW Bypass
221         CHAR etwbypass[] = { 0xC3 }; // ret
223         PVOID aligedETW = pageAlign(nt.pEtwEventWrite);
224         unsigned __int64 memPage = 0x1000;
```

github.com/sliverarmory/injectAmsiBypass/blob/main/inject-amsiBypass.c

```
27          // The addresses of the symbols for DLLs are the same across all processes
28          PVOID amsiOpenSessAddr = KERNEL32$GetProcAddress(KERNEL32$LoadLibraryA("amsi.dll"), "AmsiOpenSession");
29          // This is the payload we will inject into the start of the AmsiOpenSession symbol within the target process
30          unsigned char amsibypass[] = { 0x48, 0x31, 0xC0 }; // xor rax, rax
31          // Write the AMSI bypass payload to the remote process
32          BOOL success = KERNEL32$WriteProcessMemory(hProc, amsiOpenSessAddr, (PVOID)amsibypass, sizeof(amsibypass), &bytesWritte
33          KERNEL32$CloseHandle(hProc);
```

- Some hooks may be temporary and reverted on execution
- To avoid duplication Windows backs common DLLs to physical memory and is shared across processes
- Patching these DLLs triggers a copy on write, making the page private to the process
- Querying the working set (`QueryWorkingSetEx`) to see if the shared bit is cleared will indicate if a copy on write occurred[1]
- Mapping the exports in the modified page can give an indicator to which functions were patched

1 https://www.forrest-orr.net/post/masking-malicious-memory-artifacts-part-ii-insights-from-moneta

- To execute assemblies, BRC4 has the `sharpinline` command:
- EtW and AMSI patched are always applied:

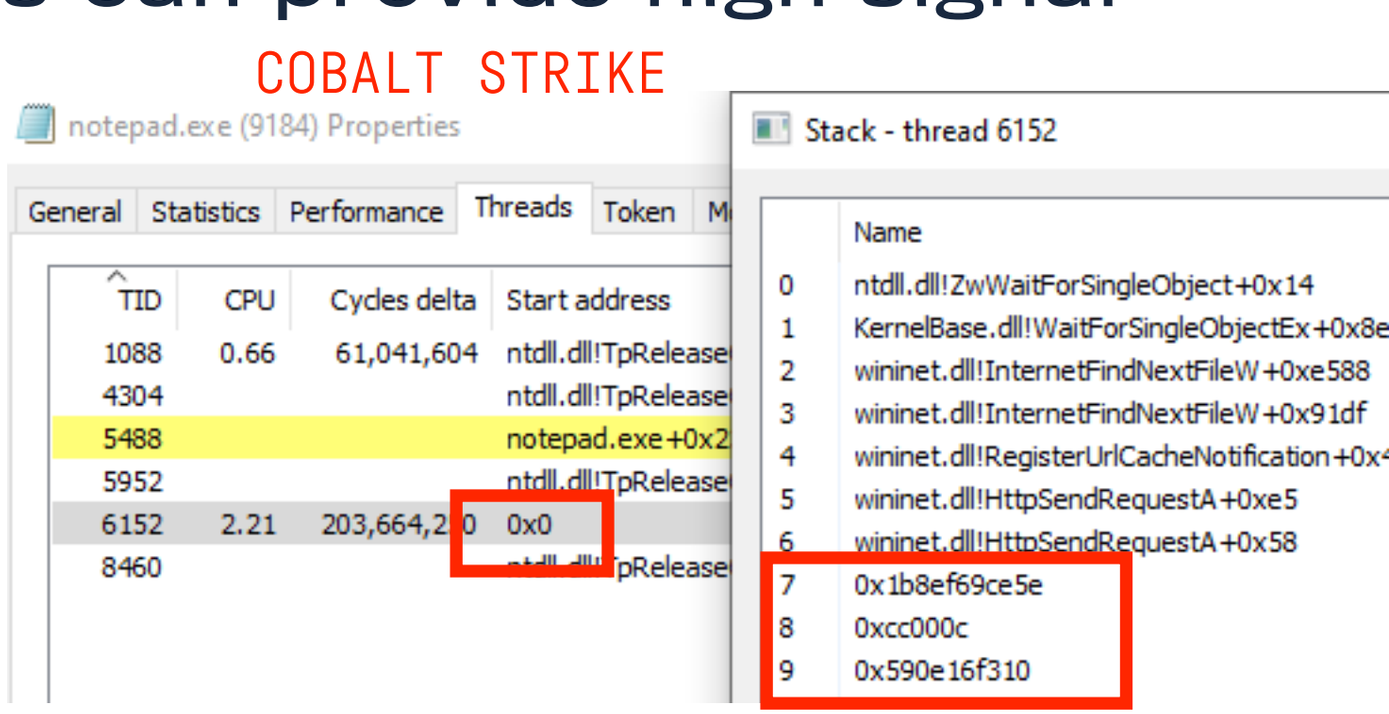
```
2022/06/20 14:03:19 BST [input] admin => sharpinline /home/parallels/brc4/bruteratel-10jun22/server_confs/Seatbelt.exe all

2022/06/20 14:03:53 BST [sent 284912 bytes]
[+] Running dotnet_v2 in CLR v4.0.30319
[+] Patched EtwEventWrite
[+] Patched AMSI
```

- The permanent patches can be confirmed in a debugger:

```
0:016> uf ntdll!EtWEventWrite
ntdll!EtwEventWrite:
00007ffa`bb05f1a0 c3                 ret
0:016> uf amsi!AmsiScanBuffer
DBGHELP: downstreamstore*https://msdl.microsoft.com/down
amsi!AmsiScanBuffer:
00007ffa`a78635e0 b857000780         mov     eax,80070057h
00007ffa`a78635e5 c3                 ret
0:016>
```

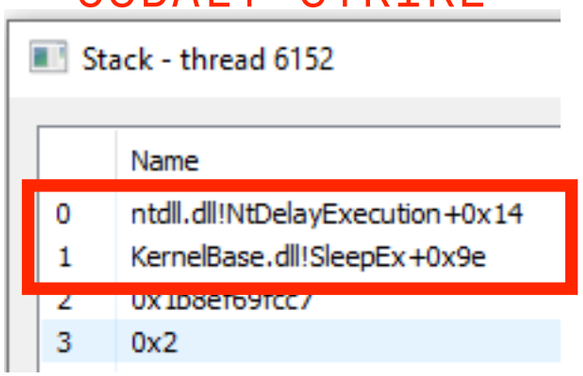
- Threat hunting for these patches will likely provide high signal detections

- Beacons will operate in one or more threads depending on if it's synchronous or asynchronous
- Anomalies in threads can provide high signal indicators
- Threads originating from virtual memory are highly suspicious

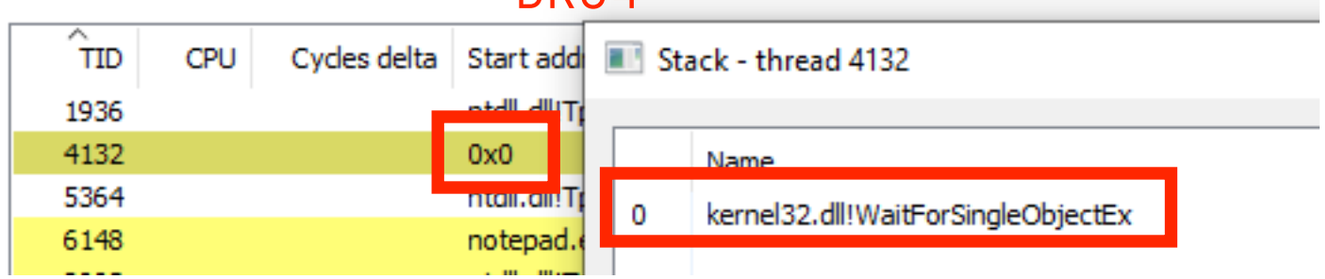


- Threads with techniques to delay execution in the call stack are another strong indicator of a sleeping beacon[1]:

COBALT STRIKE



BRC4



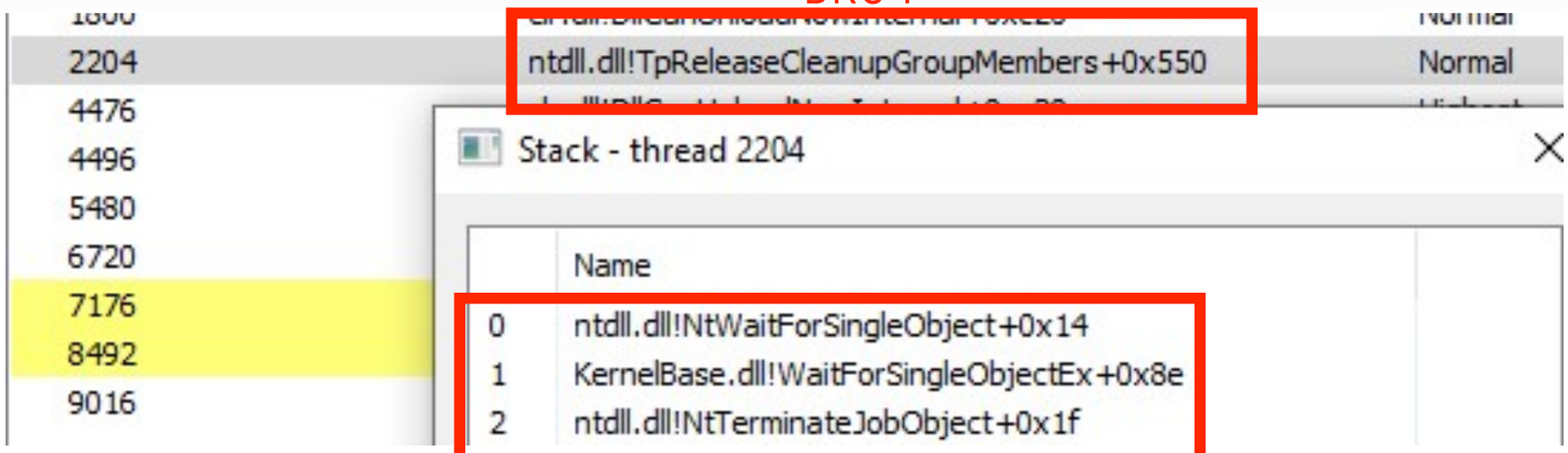
- Hunting for threads in a state of delay execution (ie `SleepEx` -> `NtDelayExecution`) when combined with other indicators provides high signal

1 https://github.com/thefLink/Hunt-Sleeping-Beacons

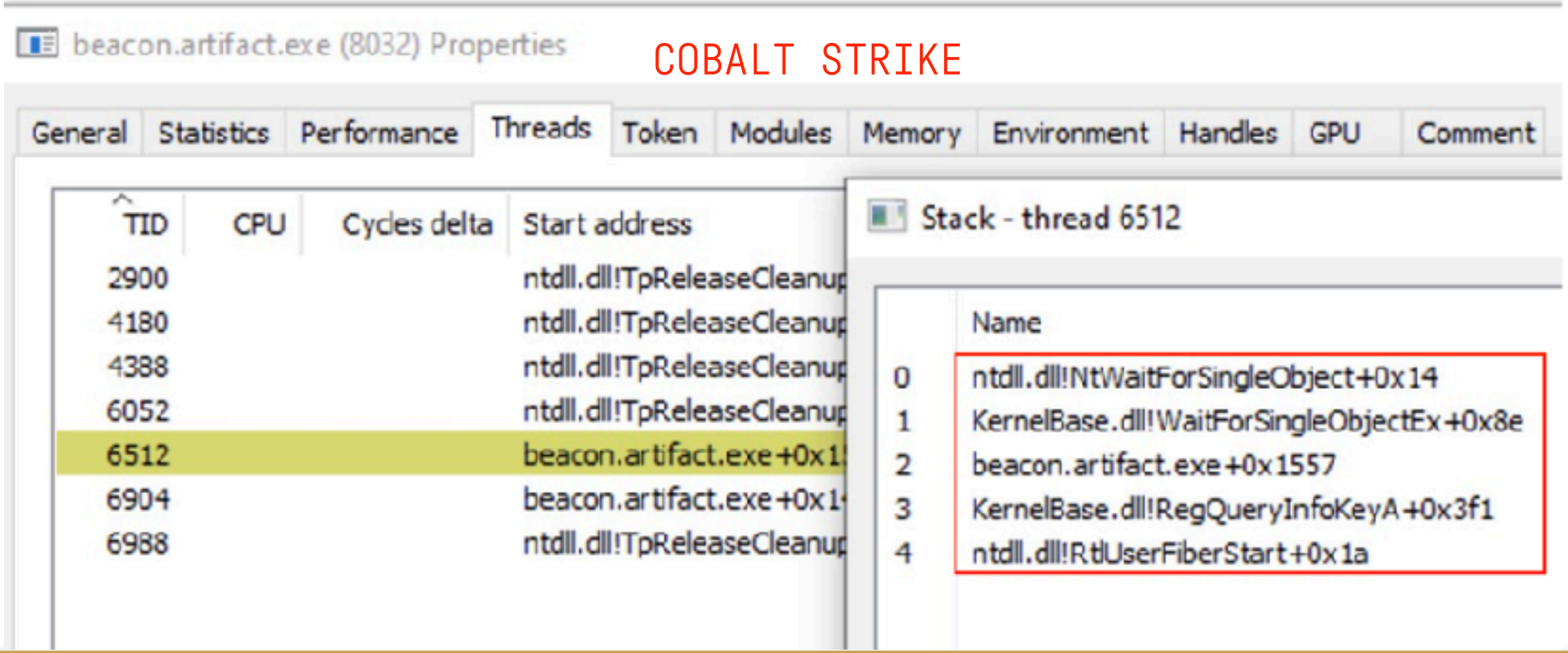

CALL STACK SPOOFING
SO HOT RIGHT NOW
imgflip.com

- Typically achieved by:
  - Truncating the stack, for example by setting `0x0` to the frames return address
  - Cloning the context of a legitimate thread
  - Switching the thread to a fiber, `CreateFiberEx` creates a new stack

BRC4



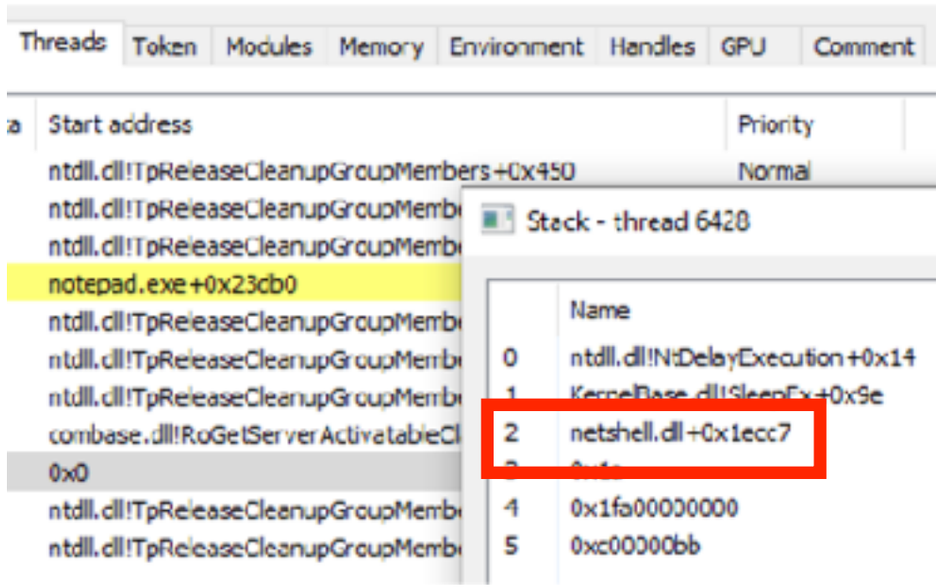
COBALT STRIKE



- Suspicious indicators include irregular start addresses (e.g. 0x0) or a truncated call stack
- Tracing the thread will typically show it originating from a set of “good” start addresses, eg:
  - `BaseThreadInitThunk`
  - `RtlUserThreadStart`
  - `RtlGetAppContainerNamedObjectPath`
- A cloned context can leave traces within the TIB, such as duplicated Stack Base and Stack Limit
- Fibers do not appear to be common, hunt of start address of `RtlUserFiberStart`

- Module stomping avoids the beacon running from unmapped memory
- A legitimate module is loaded and the beacon copies itself over the module
- A thread is then created backed by the stomped code

```
set module_x64 "netshell.dll";

set module_x86 "netshell.dll";
```



- A number of strategies exist for detecting module stomping, including:
  - Comparing the in-memory module to that on disk,
  - A modified working set may indicate changes to the DLL,
  - Indicators from the implementation used, eg LoadLibrary vs custom loader

- The Cobalt Strike implementation of module stomping leaves some IoCs in the PEB
- The module is loaded using LoadLibraryEx

```
LoadLibraryExA(moduleName, NULL, DONT_RESOLVE_DLL_REFERENCES);
```

- This avoids the loader calling DllMain and processing imports
- Leaves traces in the `LDR_DATA_TABLE_ENTRY` structure of the PEB for the stomped module
- `EntryPoint` attribute is null and `ImageDll` bit set to false

1 https://github.com/slaeryan/DetectCobaltStomp

- Outside of the endpoint, detecting the team server may assist in obtaining the beacon
- Indicators can exist on the team server to fingerprint it
- Examples include:
  - The extraneous space affecting Cobalt Strike <3.13
  - Servers exposing default staging URIs
  - Default landing page exposed

1 https://blog.fox-it.com/2019/02/26/identifying-cobalt-strike-team-servers-in-the-wild/

- The Cobalt Strike C2 server is based on NanoHttpd
- The underlying web server is likely less prevalent than Cobalt Strike, and easy to fingerprint:

Request

Pretty Raw Hex

```
1 GET
  /api/devices/6fd24c22-4e3a-4ce2-9ba3/telemetry/logging
  HTTP/1.1
2 User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0;
  rv:11.0) like Gecko
3 Host: 10.211.55.22:8443
4 Content-Length: 0
5 Range: bytes=0-9999999999
6 Connection: Keep-Alive
7 Cache-Control: no-cache
8
9
```

Response

Pretty Raw Hex Render

```
1
```

- Unhandled exception when handling Range:

```
        at c2profile.Profile.recover(Unknown Source)
        at beacon.BeaconHTTP$_A.serve(Unknown Source)
        at c2profile.MalleableHook.serve(Unknown Source)
        at cloudstrike.WebServer._serve(WebServer.java:308)
        at cloudstrike.WebServer.serve(WebServer.java:246)
        at cloudstrike.NanoHTTPD$HTTPSession.run(NanoHTTPD.java:372)
        at java.lang.Thread.run(Thread.java:833)
        at com.oracle.svm.core.thread.JavaThreads.threadStartRoutine(JavaThreads.java:597)
        at com.oracle.svm.core.posix.thread.PosixJavaThreads.pthreadStartRoutine(PosixJavaThreads.java:194)
[-] Invalid session id
[-] A Malleable C2 attempt to recover data from a '.http-get.client.metadata' transaction failed. This could be due to a bug in the profile, a
 change made to the profile after this Beacon was run, or a change made to the transaction by some device between your target and your Cobalt
Strike controller. The following information will (hopefully) help narrow down what happened.

From    '10.211.55.2'
URI     '/api/devices/0fd24c22-4e3a-4ce2-9ba3/telemetry/logging'

Headers
-------
'REMOTE_ADDRESS' = '/10.211.55.2'
'Cache-Control' = 'no-cache'
'User-Agent' = 'Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko'
'Connection' = 'Keep-Alive'
'Host' = '10.211.55.22:8443'
'Content-Length' = '0'
'Range' = 'bytes=0-9999999999'

Exception in thread "HTTP session handler" java.lang.NumberFormatException: For input string: "9999999999"
        at java.lang.Integer.parseInt(Integer.java:668)
        at java.lang.Integer.parseInt(Integer.java:786)
        at cloudstrike.WebServer.handleRanges(WebServer.java:205)
        at cloudstrike.WebServer.serve(WebServer.java:246)
        at cloudstrike.NanoHTTPD$HTTPSession.run(NanoHTTPD.java:372)
        at java.lang.Thread.run(Thread.java:833)
        at com.oracle.svm.core.thread.JavaThreads.threadStartRoutine(JavaThreads.java:597)
        at com.oracle.svm.core.posix.thread.PosixJavaThreads.pthreadStartRoutine(PosixJavaThreads.java:194)
```

- src/main/java/cloudstrike/WebServer.java:

```
public Response handleRanges(String method, Properties header, Response original) {
   if (header.containsKey("Range") && "GET".equals(method) && original.size > 0L && original.data != null && "200 OK".equals(original.status)) {
      Pattern p = Pattern.compile("bytes=(\\d+)-(\\d+)");
      Matcher m = p.matcher((String)header.get("Range"));
      if (m.matches()) {
         int start = Integer.parseInt(m.group(1));
         int end = Integer.parseInt(m.group(2)) + 1;

         try {
            if (start < end && (long)end <= original.size) {
               byte[] rdata = new byte[end - start];
               original.data.skip((long)start);
```

- Range Not Satisfiable fingerprint (shared with NanoHttpd) when compared with responses of server in `Server` header:

Request

Pretty Raw Hex

```
1 GET
  /api/devices/6fd24c22-4e3a-4ce2-9ba3/telemetry/logging
  HTTP/1.1
2 User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0;
  rv:11.0) like Gecko
3 Host: 10.211.55.22:8443
4 Content-Length: 0
5 Range: bytes=1-0
6 Connection: Keep-Alive
7 Cache-Control: no-cache
8
9
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 416 Range Not Satisfiable
2 Content-Type: text/plain
3 Date: Sun, 19 Jun 2022 22:03:03 GMT
4 Server: Microsoft-IIS/8.5
5 Content-Length: 21
6 Connection: keep-alive
7 Content-Range: bytes */406
8 Cache-Control: max-age=0, no-cache
9 Pragma: no-cache
10 Access-Control-Allow-Origin: *
11 Timing-Allow-Origin: *
12 x-ms-content-source: ContentPackageReader
13 X-UA-Compatible: IE=edge
14
15 Range Not Satisfiable
```

- src/main/java/cloudstrike/NanoHTTPD.java:

```
private String decodePercent(String str) throws InterruptedException {
    try {
        return URLDecoder.decode(str, "UTF-8");
    } catch (Exception var3) {
        this.sendError("400 Bad Request", "BAD REQUEST: Bad percent-encoding.");
        return null;
    }
}
```

Request

Pretty Raw Hex \n

```
GET /%0 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0;
rv:11.0) like Gecko
Host: 10.211.55.22:8443
Content-Length: 0
Range: bytes=1-0
Connection: Keep-Alive
Cache-Control: no-cache
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 400 Bad Request
Date: Sun, 19 Jun 2022 22:06:50 GMT
Content-Type: text/plain

BAD REQUEST: Bad percent-encoding.
```

- Fingerprint via base64 POST:

Request

Pretty Raw Hex \n

```
1 POST / HTTP/2
2 Host: 10.211.55.22
3 User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0;
  rv:11.0) like Gecko
4 Content-Length: 16
5 Cache-Control: no-cache
6
7 YnJ1dGVjcmFwdGVs
```

Response

Request

Pretty Raw Hex \n

```
1 POST / HTTP/2
2 Host: 10.211.55.22
3 User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0;
  rv:11.0) like Gecko
4 Content-Length: 6
5 Cache-Control: no-cache
6
7 foobar
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 49
4 Date: Mon, 20 Jun 2022 13:45:09 GMT
5
6 <html>
7   <body>
8     Nothing to see here
9   </body>
10 </html>
```

- Active scanning for Brute Ratel is being performed:

Hello,

Your instance has been reported for Command and Control (C)2 activity related to a large scale botnet. Operation of a C2 is a violation of the AWS Acceptable Use policy.

Instance: i-07c9cae07cee3e825

Please terminate your infected resource. If you feel this abuse report was sent in error, please provide a clear explanation with details on why your resources was reported for this activity.

Regards,
AWS Trust & Safety

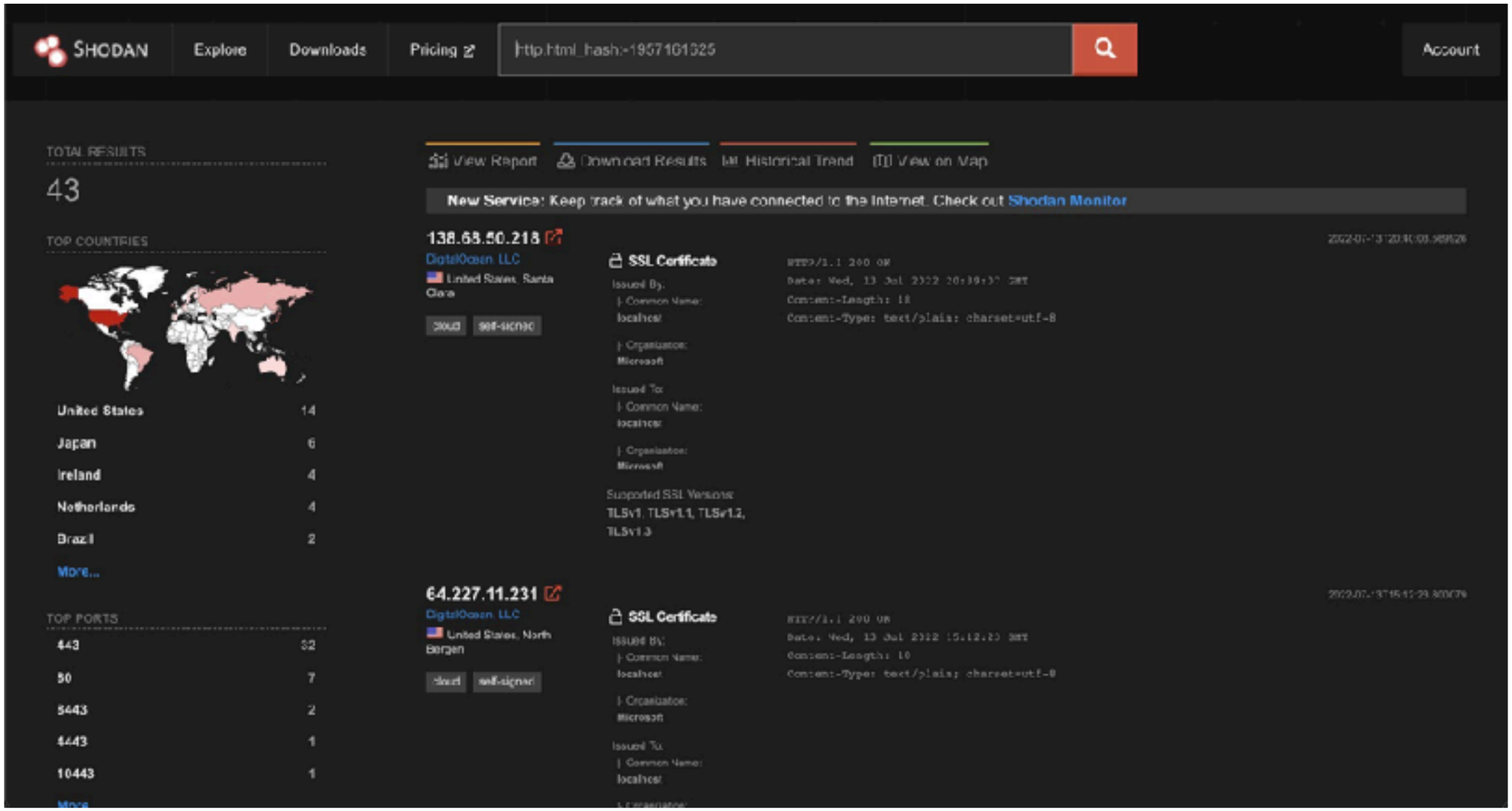
Case Number: 15927365252

How can I contact a member of the AWS abuse team or the reporter?
Reply to this email with the original subject line.

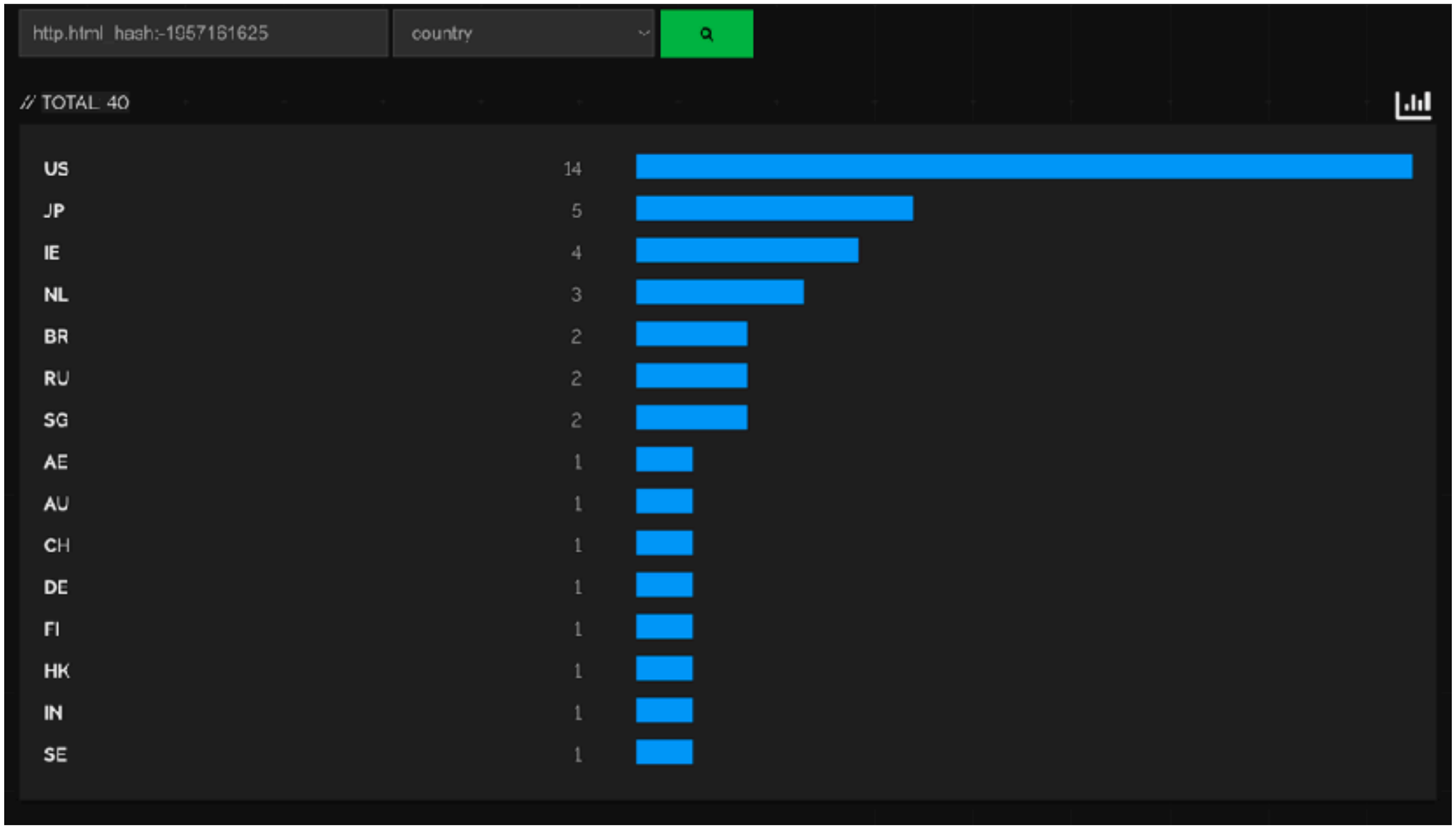
Amazon Web Services

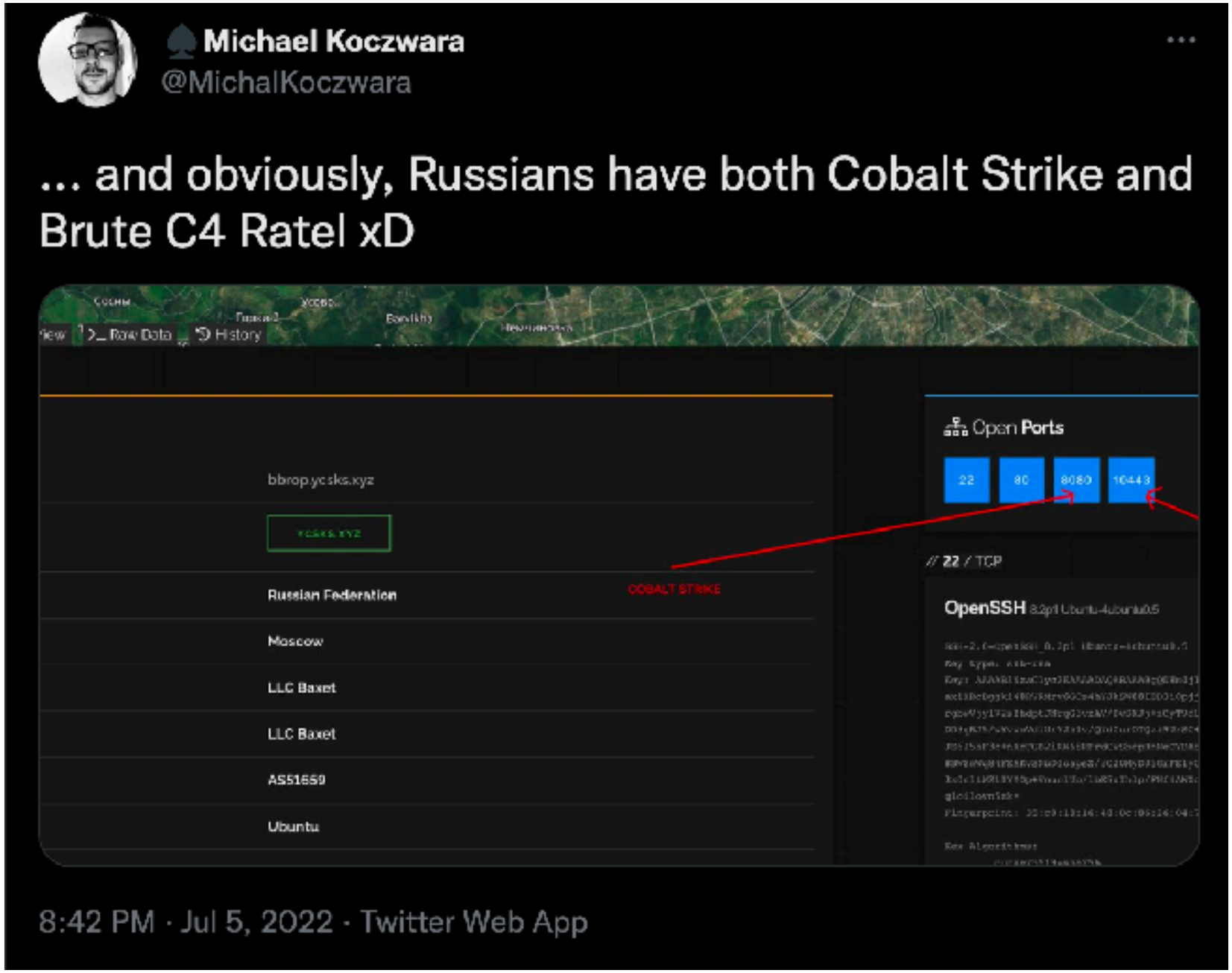
Name	Instance ID	Instance state	Instance type	Status check	Alarm
Test: BRC4	i-07c9cae07cee3e825	Running	t2.micro	2/2 checks passed	No ala

- Trivial to find using Shodan with `http.html_hash=-1957161625`:



- Distributed over the following regions:




Michael Koczwara
@MichalKoczwara
... and obviously, Russians have both Cobalt Strike and Brute C4 Ratel xD
Raw Data
History
bbrop.ycsks.xyz
Russian Federation
COBALT STRIKE
Moscow
LLC Baxet
LLC Baxet
AS51659
Ubuntu
Open Ports
22
80
8080
10443
// 22 / TCP
OpenSSH
8:42 PM · Jul 5, 2022 · Twitter Web App

- Bonus: What can we do once we've spotted a badger?

```
C:\Tools>BRC4ConfigExtractor.exe 836
BruteRatel v1.x Config Extractor
[+] Analysing process with ID 836 (1615618884 bytes)
[+] Searching process memory for badger state structure ...
[+] Found badger state structure at 0x000000A619C7ED70
[+] Badger 'b-1\\3D32N921PFJUBMBFD6NQSPG4UTF6RCGF' found...
    : 10.211.55.22
    : /content.php
    : /admin.php
    : x64/10.0
    : 19043
    : bob
    : QwA6AFwAVwBpAG4AZABvAHcAcwBcAHMAeQBzAHQAZQBtADMAMgBcAG4AbwB0AGUAcABhAGQALgBlAHgAZQA=
    : foobar123
    : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36
    : "},"mtdt":{"h_name":"
    : ","p_name":"
    : ","uid":"
    : ","pid":"
    : "}}

    : "},"dl":{"chkin":"
    : {"
    : ":
    : }

    : ","wver":"
    : ","arch":"x64","bld":"
    : ROOT\CIMV2
    :     :     :     :
C:\Tools>
```

- In order to better understand some of these detection techniques, we decided to build Beacon Hunter
- Currently supports:
  - Analysing call stacks for unmapped memory, suspicious start addresses,
  - Detection of suspicious executable pages
  - Detection of Cobalt Strike module stomping,
  - Querying the working set and detecting modified exports

Hunting Beacons

- https://medium.com/threatpunter/detecting-adversary-tradecraft-with-image-load-event-logging-and-eql-8de93338c16
- https://suspicious.actor/2022/05/05/mdsec-nighthawk-study.html
- https://github.com/SecIdiot/FOLIAGE
- https://github.com/Cracked5pider/Ekko
- https://github.com/mgeeky/ShellcodeFluctuation
- https://github.com/JLospinoso/gargoyle
- https://www.elastic.co/blog/detecting-cobalt-strike-with-memory-signatures
- https://codex-7.gitbook.io/codexs-terminal-window/blue-team/detecting-cobalt-strike/sleep-mask-kit-iocs
- https://github.com/slaeryan/DetectCobaltStomp
- https://blog.fox-it.com/2019/02/26/identifying-cobalt-strike-team-servers-in-the-wild/
- https://unit42.paloaltonetworks.com/brute-ratel-c4-tool/
- https://news.sophos.com/en-us/2022/07/14/blackcat-ransomware-attacks-not-merely-a-byproduct-of-bad-luck/
- https://www.sentinelone.com/blog/research-paper-emulating-phineas-phisher-attacks-in-modern-edr-environments/

Thanks to @peterwintrsmith, @modexpblog and the rest of the @MDSecLabs team