ELEPHANT IN THE SANDBOX:

AN ANALYSIS OF DBATLOADER’S SANDBOX EVASION TECHNIQUES

Kyle Cucci
Staff Security Research Engineer @ Proofpoint

proofpoint. | Protect people. Defend data.

À PROPOS DE MOI

- Salut! I’m **Kyle Cucci**
- Staff Security Research Engineer @ Proofpoint
  - Malware analysis / reversing
  - Detection signatures
  - Malware sandbox
- Hobbies: malware, research, also malware

X: @d4rksystem
LinkedIn: https://linkedin.com/in/kylecucci

LET’S PLAY A GAME: BUG? OR FEATURE?

- DBatLoader (aka. ModiLoader, aka. NatsoLoader) uses “interesting” sandbox evasion techniques
- Executes these techniques in a **yolo-like*** manner – not much stealth
- Let’s talk about the “interesting” design decisions of DBatLoader

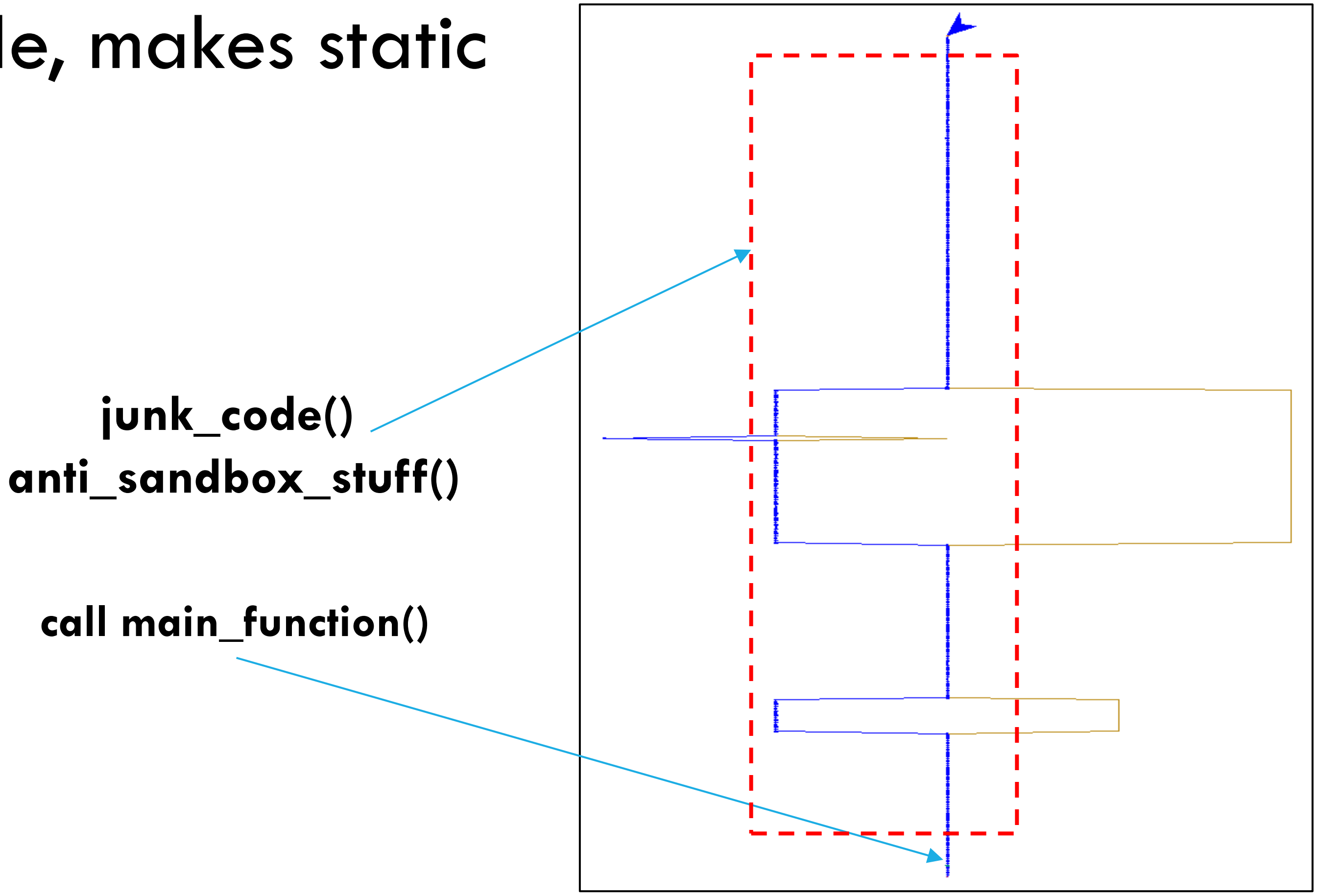
*__yolo-like__ = “you only live once”. Doing something without care or regard.

DBATLOADER: OVERVIEW & HISTORY

- DBatLoader functions mainly as a **loader/downloader**
- Loads Remcos, AveMaria, Formbook/XLoader and other RATs and stealers
- Multiple stages:
  - LNK >> Powershell >> DBatLoader
  - JS >> BAT Script >> DbatLoader
  - Payloads usually hosted on OneDrive, Google Drive, sometimes Discord

ANTI- STATIC ANALYSIS (JUNK CODE)

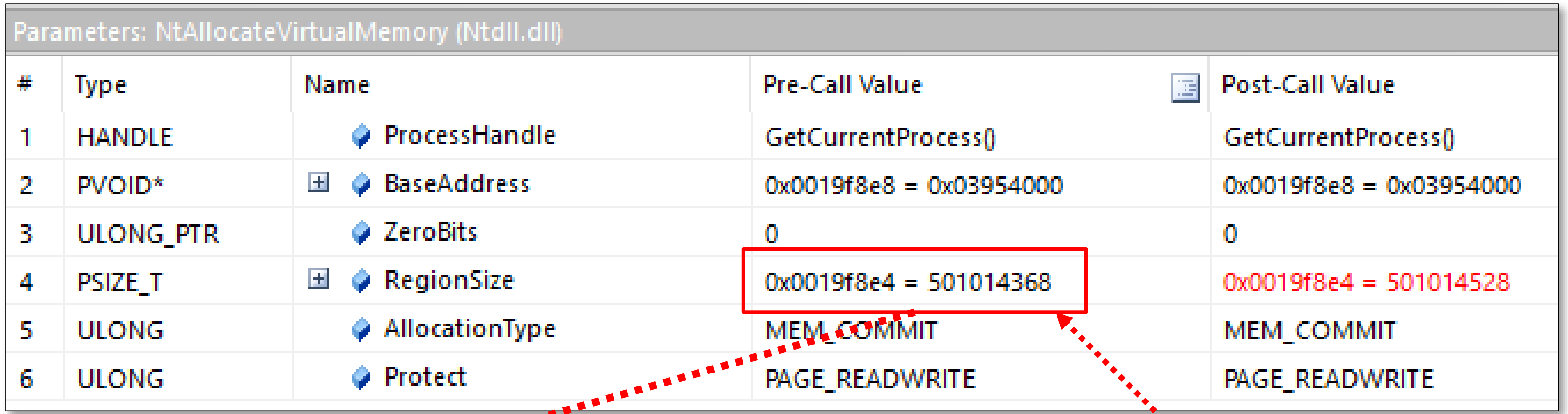
- Contains lots of junk code, makes static analysis annoying.



TECHNIQUE 1: MEMORY BOMBING

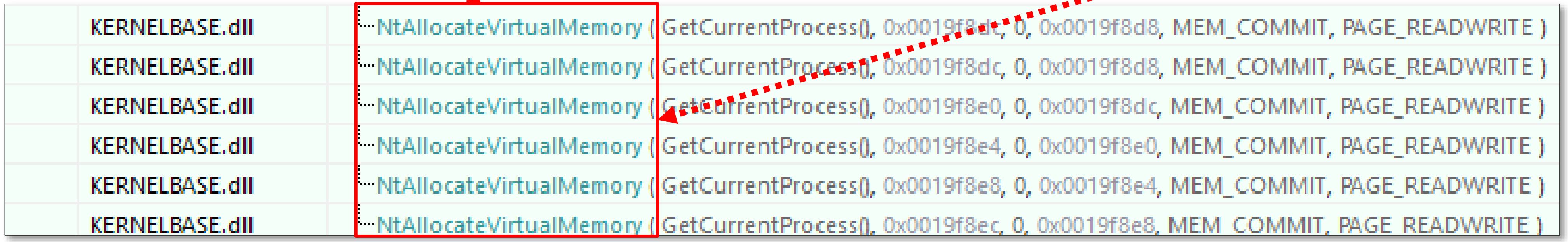
- Allocates more memory than most sandboxes have.
- Not enough RAM == "commitment_limit" error.

NtAllocateVirtualMemory



Parameters: NtAllocateVirtualMemory (Ntdll.dll)

#	Type	Name	Pre-Call Value	Post-Call Value
1	HANDLE	ProcessHandle	GetCurrentProcess()	GetCurrentProcess()
2	PVOID*	BaseAddress	0x0019f8e8 = 0x03954000	0x0019f8e8 = 0x03954000
3	ULONG_PTR	ZeroBits	0	0
4	PSIZE_T	RegionSize	0x0019f8e4 = 501014368	0x0019f8e4 = 501014528
5	ULONG	AllocationType	MEM_COMMIT	MEM_COMMIT
6	ULONG	Protect	PAGE_READWRITE	PAGE_READWRITE

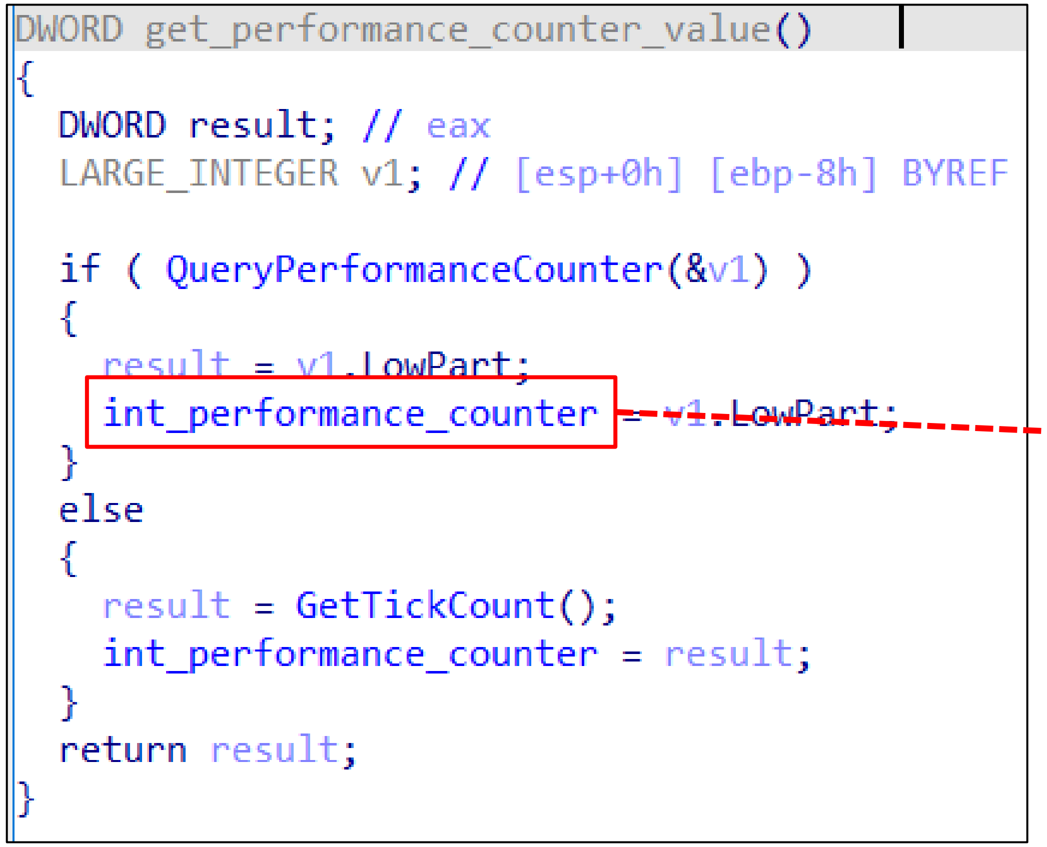


```
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8d..., 0, 0x0019f8d8, MEM_COMMIT, PAGE_READWRITE )
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8dc, 0, 0x0019f8d8, MEM_COMMIT, PAGE_READWRITE )
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8e0, 0, 0x0019f8dc, MEM_COMMIT, PAGE_READWRITE )
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8e4, 0, 0x0019f8e0, MEM_COMMIT, PAGE_READWRITE )
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8e8, 0, 0x0019f8e4, MEM_COMMIT, PAGE_READWRITE )
KERNELBASE.dll   NtAllocateVirtualMemory ( GetCurrentProcess(), 0x0019f8ec, 0, 0x0019f8e8, MEM_COMMIT, PAGE_READWRITE )
```

~500 MB

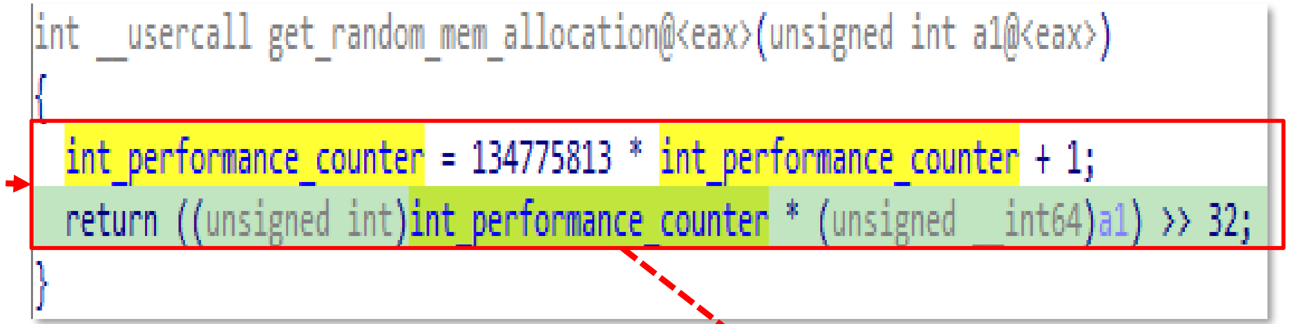
TECHNIQUE 1: MEMORY BOMBING

Calculation of random memory allocation sizes (wut?)



```
DWORD get_performance_counter_value()
{
  DWORD result; // eax
  LARGE_INTEGER v1; // [esp+0h] [ebp-8h] BYREF

  if ( QueryPerformanceCounter(&v1) )
  {
    result = v1.LowPart;
    int_performance_counter = v1.LowPart;
  }
  else
  {
    result = GetTickCount();
    int_performance_counter = result;
  }
  return result;
}
```

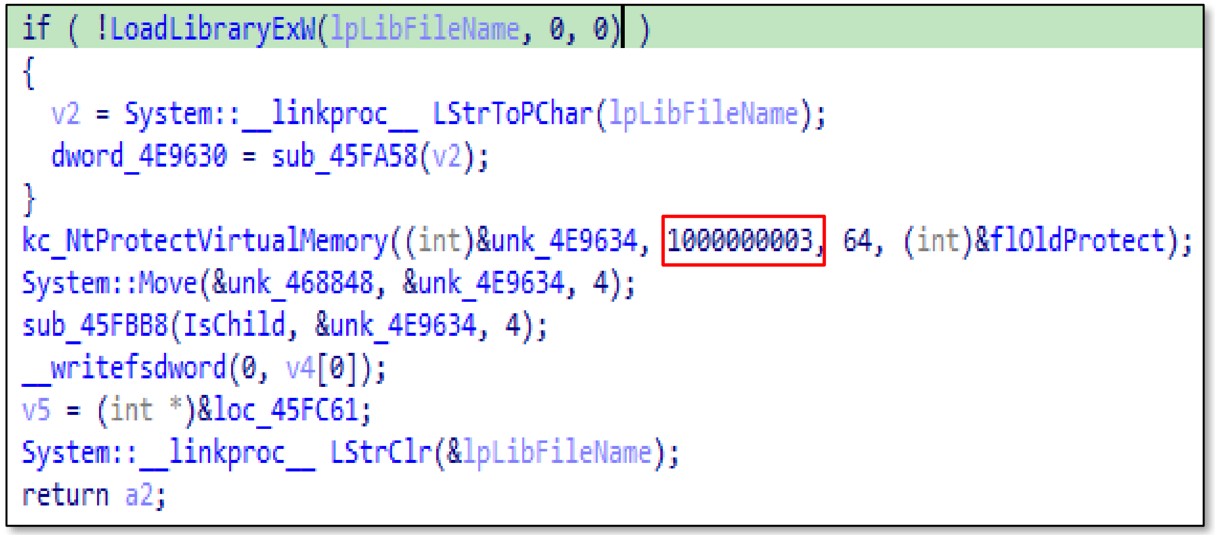


```
int __usercall get_random_mem_allocation@<eax>(unsigned int a1@<eax>)
{
  int_performance_counter = 134775813 * int_performance_counter + 1;
  return ((unsigned int)int_performance_counter * (unsigned __int64)a1) >> 32;
}
```

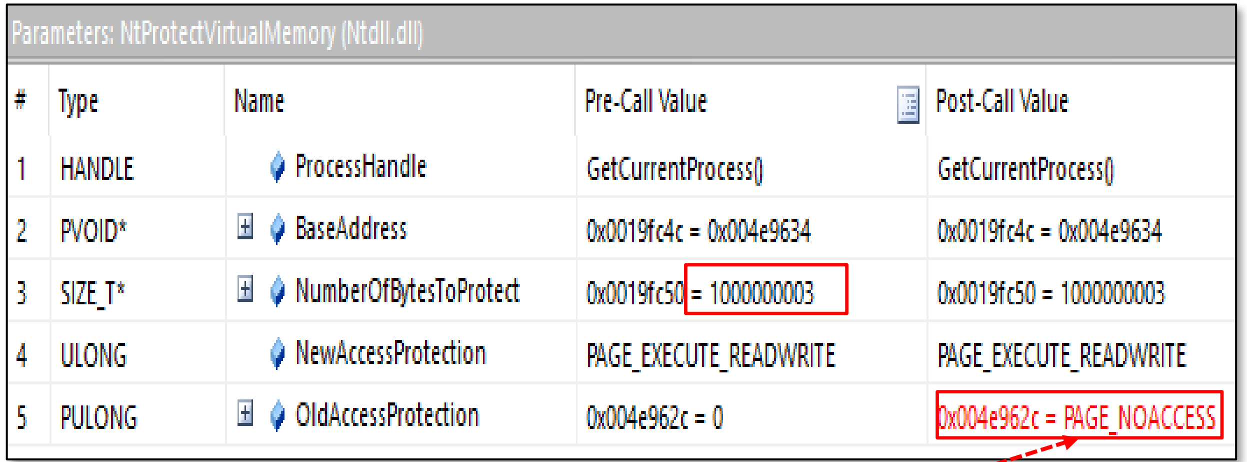
NtAllocateVirtualMemory (..., ..., ..., **int_performance_counter**,...)

TECHNIQUE 2: YOLO MEMORY PROTECTS

- "Let's just try to change the protection of memory I don't have access to":



```
if ( !LoadLibraryExW(lpLibFileName, 0, 0) )
{
  v2 = System::__linkproc__ LStrToPChar(lpLibFileName);
  dword_4E9630 = sub_45FA58(v2);
}
kc_NtProtectVirtualMemory((int)&unk_4E9634, 1000000003, 64, (int)&flOldProtect);
System::Move(&unk_468848, &unk_4E9634, 4);
sub_45FBB8(IsChild, &unk_4E9634, 4);
__writefsdword(0, v4[0]);
v5 = (int *)&loc_45FC61;
System::__linkproc__ LStrClr(&lpLibFileName);
return a2;
```

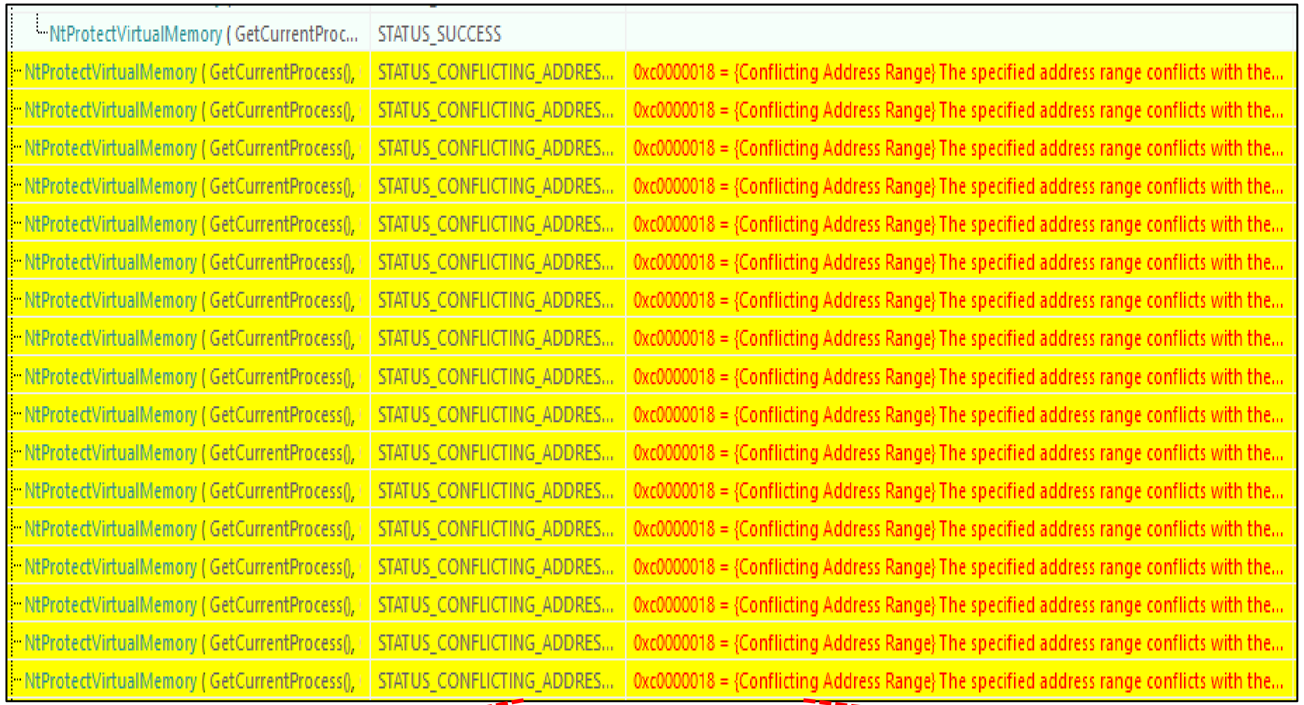


Parameters: NtProtectVirtualMemory (Ntdll.dll)

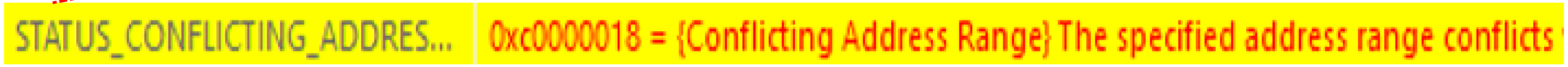
#	Type	Name	Pre-Call Value	Post-Call Value
1	HANDLE	ProcessHandle	GetCurrentProcess()	GetCurrentProcess()
2	PVOID*	BaseAddress	0x0019fc4c = 0x004e9634	0x0019fc4c = 0x004e9634
3	SIZE_T*	NumberOfBytesToProtect	0x0019fc50 = 1000000003	0x0019fc50 = 1000000003
4	ULONG	NewAccessProtection	PAGE_EXECUTE_READWRITE	PAGE_EXECUTE_READWRITE
5	PULONG	OldAccessProtection	0x004e962c = 0	0x004e962c = PAGE_NOACCESS

Poor DBatLoader ☹.
We get a "PAGE_NOACCESS" error.

TECHNIQUE 2: YOLO MEMORY PROTECTS



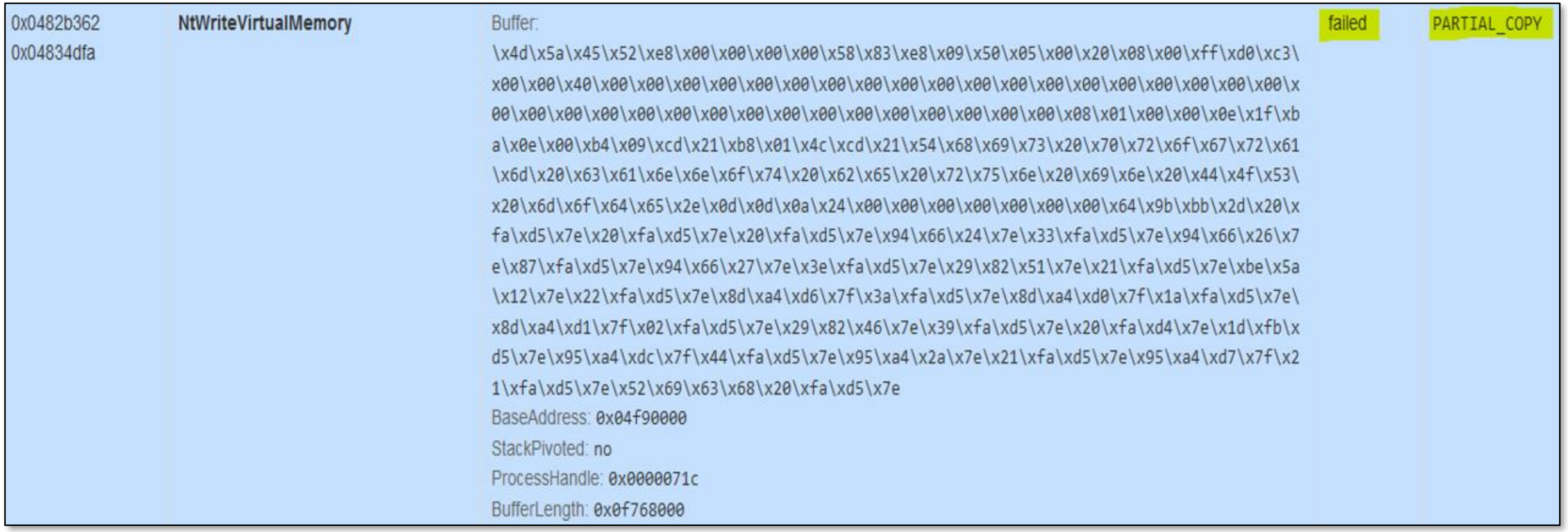
```
└NtProtectVirtualMemory ( GetCurrentProc...   STATUS_SUCCESS
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
NtProtectVirtualMemory ( GetCurrentProcess(),   STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts with the...
```



```
STATUS_CONFLICTING_ADDRES...   0xc0000018 = {Conflicting Address Range} The specified address range conflicts
```

TECHNIQUE 3: NO MEMORY? NO PROBLEM (OR, "DRUNK PROCESS INJECTION"

- And if we can't change the protection class? Let's just yolo that too ☺
- Results in funny access errors, or errors like "PARTIAL_COPY"



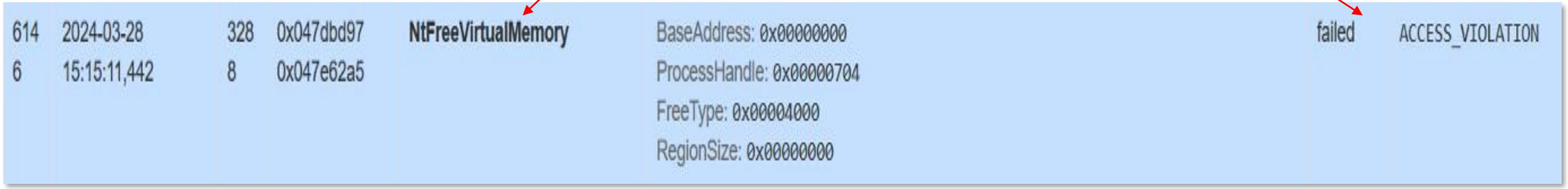
```
0x0482b362    NtWriteVirtualMemory    Buffer:                                                                                    failed    PARTIAL_COPY
0x04834dfa                            \x4d\x5a\x45\x52\xe8\x00\x00\x00\x00\x58\x83\xe8\x09\x50\x05\x00\x20\x08\x00\xff\xd0\xc3\
                                      x00\x00\x40\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
                                      00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x01\x00\x00\x0e\x1f\xb
                                      a\x0e\x00\xb4\x09\xcd\x21\xb8\x01\x4c\xcd\x21\x54\x68\x69\x73\x20\x70\x72\x6f\x67\x72\x61
                                      \x6d\x20\x63\x61\x6e\x6e\x6f\x74\x20\x62\x65\x20\x72\x75\x6e\x20\x69\x6e\x20\x44\x4f\x53\
                                      x20\x6d\x6f\x64\x65\x2e\x0d\x0d\x0a\x24\x00\x00\x00\x00\x00\x00\x00\x64\x9b\xbb\x2d\x20\x
                                      fa\xd5\x7e\x20\xfa\xd5\x7e\x20\xfa\xd5\x7e\x94\x66\x24\x7e\x33\xfa\xd5\x7e\x94\x66\x26\x7
                                      e\x87\xfa\xd5\x7e\x94\x66\x27\x7e\x3e\xfa\xd5\x7e\x29\x82\x51\x7e\x21\xfa\xd5\x7e\xbe\x5a
                                      \x12\x7e\x22\xfa\xd5\x7e\x8d\xa4\xd6\x7f\x3a\xfa\xd5\x7e\x8d\xa4\xd0\x7f\x1a\xfa\xd5\x7e\
                                      x8d\xa4\xd1\x7f\x02\xfa\xd5\x7e\x29\x82\x46\x7e\x39\xfa\xd5\x7e\x20\xfa\xd4\x7e\x1d\xfb\x
                                      d5\x7e\x95\xa4\xdc\x7f\x44\xfa\xd5\x7e\x95\xa4\x2a\x7e\x21\xfa\xd5\x7e\x95\xa4\xd7\x7f\x2
                                      1\xfa\xd5\x7e\x52\x69\x63\x68\x20\xfa\xd5\x7e
                                      BaseAddress: 0x04f90000
                                      StackPivoted: no
                                      ProcessHandle: 0x0000071c
                                      BufferLength: 0x0f768000
```

TECHNIQUE 3: YOLO MEMORY WRITES (OR, “DRUNK PROCESS INJECTION”

- … And then DBatLoader decided to free its virtual memory… that it doesn’t have access to:

NtFreeVirtualMemory

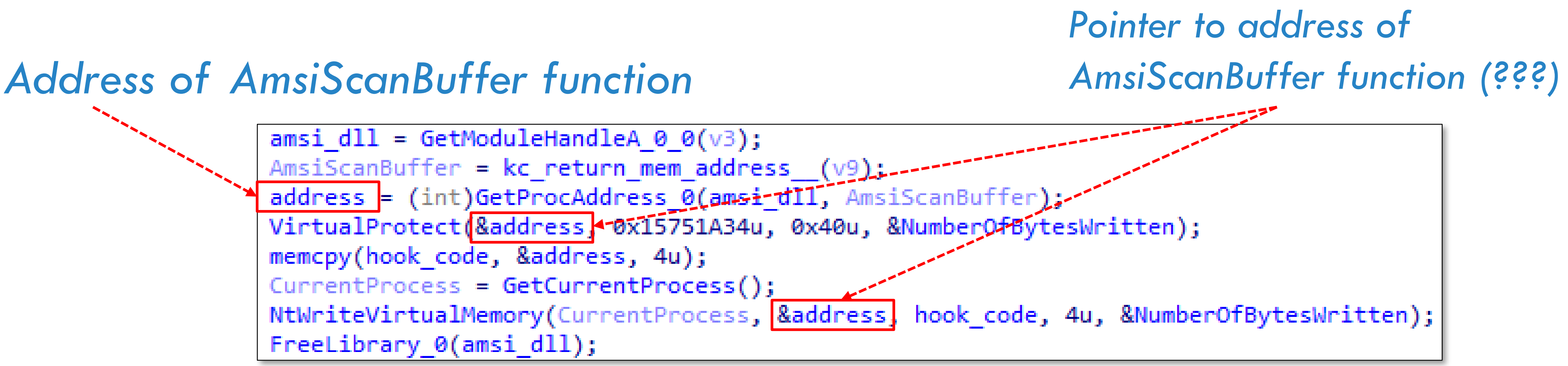
ACCESS_VIOLATION



```
6146  2024-03-28 15:15:11,442  3288  0x047dbd97 0x047e62a5  NtFreeVirtualMemory  BaseAddress: 0x00000000     failed  ACCESS_VIOLATION
                                                                                 ProcessHandle: 0x00000704
                                                                                 FreeType: 0x00004000
                                                                                 RegionSize: 0x00000000
```

TECHNIQUE 4: BUGGY AMSI PATCHING

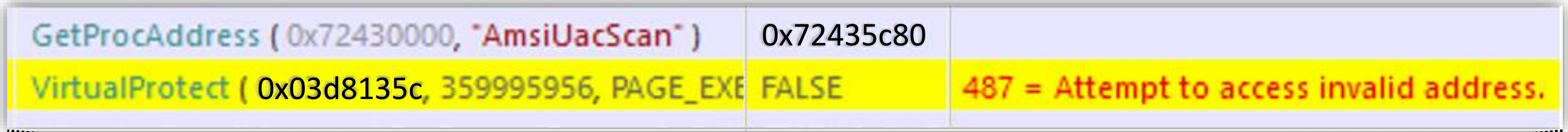
- The Anti-malware Scan Interface (or, AMSI) lets endpoint defenses scan potentially malicious code.
- Malware often tries to patch AMSI functions.
- DBatLoader uses a flawed patching mechanism.


Address of AmsiScanBuffer function
Pointer to address of AmsiScanBuffer function (???)

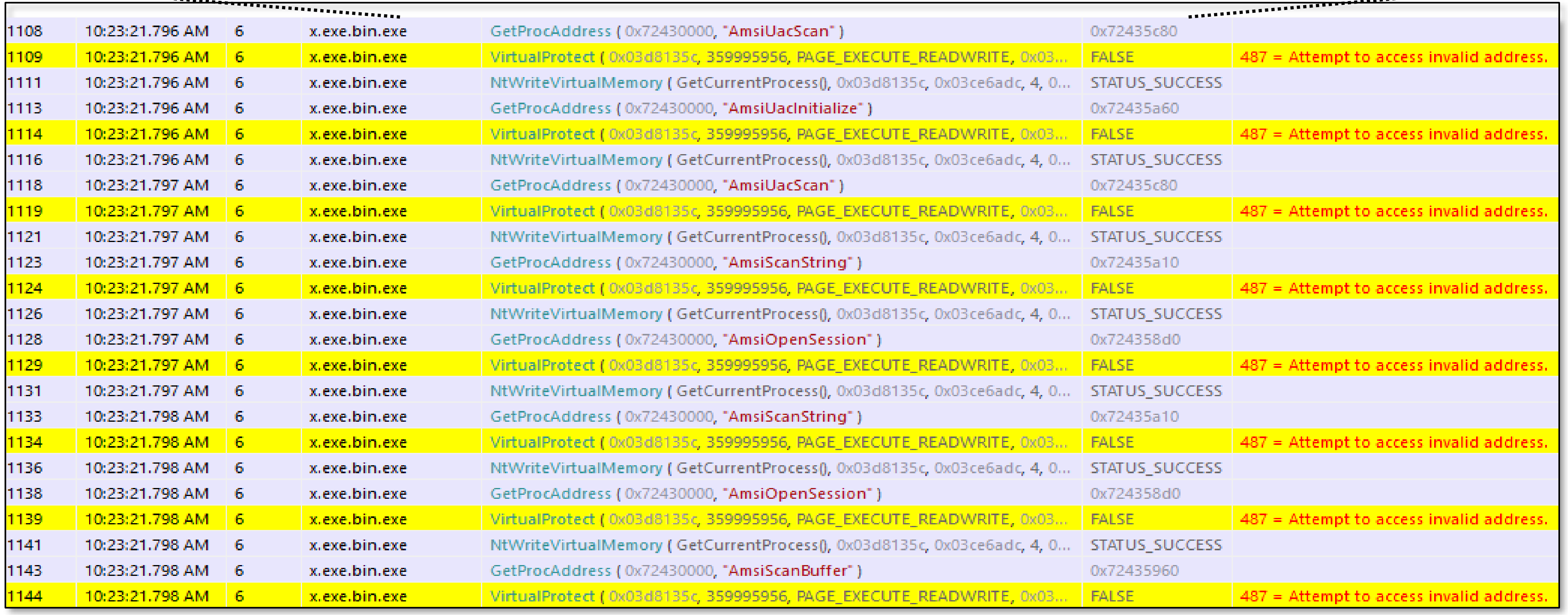
```
amsi_dll = GetModuleHandleA_0_0(v3);
AmsiScanBuffer = kc_return_mem_address__(v9);
address = (int)GetProcAddress_0(amsi_dll, AmsiScanBuffer);
VirtualProtect(&address, 0x15751A34u, 0x40u, &NumberOfBytesWritten);
memcpy(hook_code, &address, 4u);
CurrentProcess = GetCurrentProcess();
NtWriteVirtualMemory(CurrentProcess, &address, hook_code, 4u, &NumberOfBytesWritten);
FreeLibrary_0(amsi_dll);
```

https://www.ibm.com/think/x-force/email-campaigns-leverage-updated-dbatloader-deliver-rats-stealers

TECHNIQUE 4: DECOY/FLAWED AMSI PATCHING



```
GetProcAddress ( 0x72430000, "AmsiUacScan" )                       0x72435c80
VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXE                   FALSE       487 = Attempt to access invalid address.
```

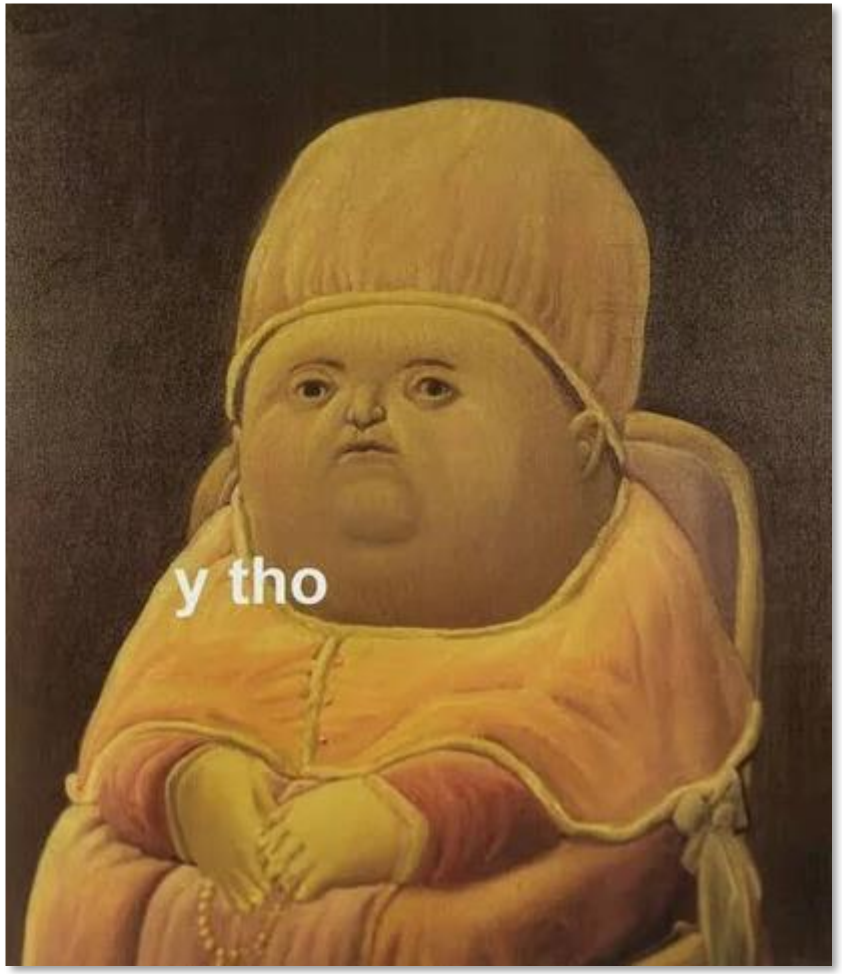


```
1108  10:23:21.796 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiUacScan" )                                   0x72435c80
1109  10:23:21.796 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1111  10:23:21.796 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1113  10:23:21.796 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiUacInitialize" )                             0x72435a60
1114  10:23:21.796 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1116  10:23:21.796 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1118  10:23:21.797 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiUacScan" )                                   0x72435c80
1119  10:23:21.797 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1121  10:23:21.797 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1123  10:23:21.797 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiScanString" )                                0x72435a10
1124  10:23:21.797 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1126  10:23:21.797 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1128  10:23:21.797 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiOpenSession" )                               0x724358d0
1129  10:23:21.797 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1131  10:23:21.797 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1133  10:23:21.798 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiScanString" )                                0x72435a10
1134  10:23:21.798 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1136  10:23:21.798 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1138  10:23:21.798 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiOpenSession" )                               0x724358d0
1139  10:23:21.798 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
1141  10:23:21.798 AM  6  x.exe.bin.exe  NtWriteVirtualMemory ( GetCurrentProcess(), 0x03d8135c, 0x03ce6adc, 4, 0...  STATUS_SUCCESS
1143  10:23:21.798 AM  6  x.exe.bin.exe  GetProcAddress ( 0x72430000, "AmsiScanBuffer" )                                0x72435960
1144  10:23:21.798 AM  6  x.exe.bin.exe  VirtualProtect ( 0x03d8135c, 359995956, PAGE_EXECUTE_READWRITE, 0x03...       FALSE           487 = Attempt to access invalid address.
```

WAIT, BUT WHY?

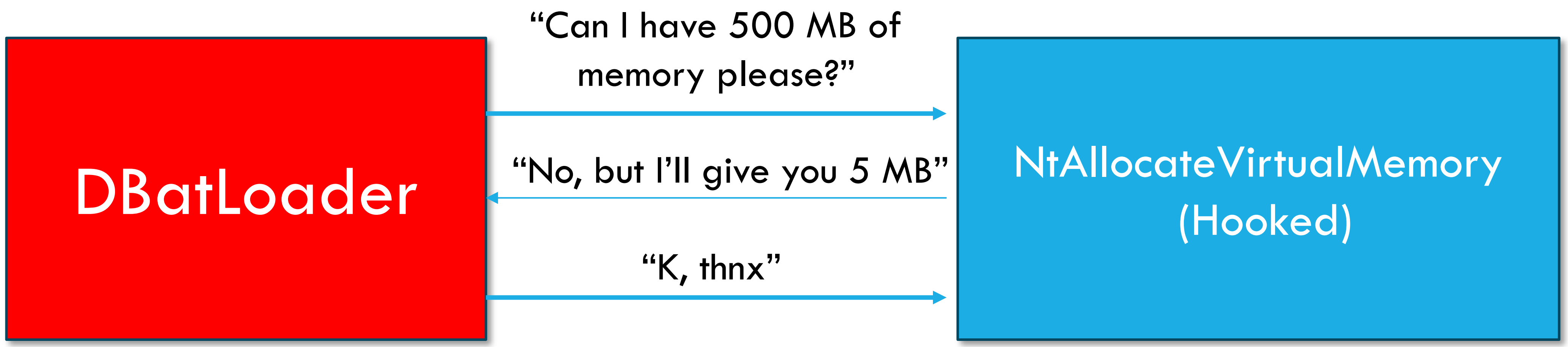
Why does DBatLoader employ such noisy tactics and what can we learn from this?

- Author(s) sacrifice stealth for sandbox smashing.
- They care greatly about sandbox avoidance.
- Sometimes difficult to understand why malware authors make the choices they do.



MITIGATIONS & DETECTIONS

- Lots of detection opportunities: multiple large memory allocations, "yolo" memory protection changes..
- Can be problematic in a sandbox.
- Hook **NtAllocateVirtualMemory**, **NtProtectVirtualMemory**, and **NtWriteVirtualMemory** to bypass some of these techniques:


DBatLoader
"Can I have 500 MB of memory please?"
"No, but I'll give you 5 MB"
"K, thnx"
NtAllocateVirtualMemory (Hooked)

REFERENCES & FURTHER READING

- https://malpedia.caad.fkie.fraunhofer.de/details/win.dbatloader
- https://www.ibm.com/think/x-force/email-campaigns-leverage-updated-dbatloader-deliver-rats-stealers
- https://www.sonicwall.com/blog/latest-dbatloader-uses-driver-module-to-disable-av-edr-software

QUESTIONS?