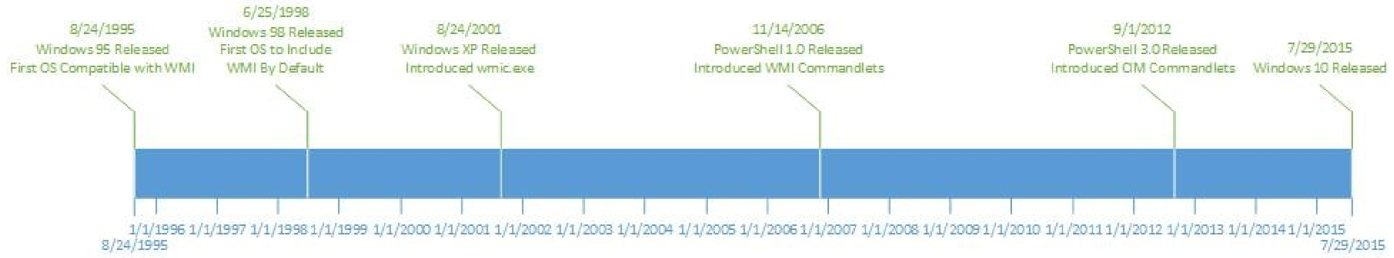
Getting Started with WMI Weaponization – Part 1

A Brief History of WMI

What is WMI?

Windows Management Instrumentation (WMI) is a Microsoft management protocol derived from the Web-Based Enterprise Management (WBEM) protocol. WMI is a web service that can perform management operations on the host operating system. It has also been a part of Windows since Windows 95 where it was available as an optional feature. Since Windows 98, WMI has been included by default. WMI primarily operates through Windows Management Instrumentation Query Language (WQL), which is a SQL like language that is used to access WMI. WMI being a web service, it can be accessed remotely on any system running the winmgmt service.



How can WMI be accessed?

VBScript (1996)

Originally, the only way to easily access WMI was via VBScript or similar Microsoft scripting. Below is a simple VBScript that uses the Win32_Process class to create a text file that contains the string netspi.

```
C:\> type wmi.vbs
strProcess = "cmd.exe /c echo 'netspi' > C:\text.txt"

Set objWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!"_
       & "\\.\root\cimv2:Win32_Process")
Error = objWMI.Create(strProcess, null, null, intProcessID)

Wscript.Echo "Process Id = " & intProcessID
Wscript.Echo "ReturnValue = " & Error

C:> cscript.exe wmi.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Process Id = 14040
```

```
ReturnValue = 0
```

wmic.exe (2001)

With Windows XP / 2003, Microsoft began shipping wmic.exe with the OS. wmic is a command line interface for use with WMI. WMIC can be run in an interactive mode or via one liners. People in the offensive security field might be familiar with the one liner command:

```
wmic.exe process call create "cmd.exe /c echo 'netspi' > C:\text.txt"
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 910124;
        ReturnValue = 0;
};
```

Breaking this command down, the Win32_Process (process) class is being invoked, while calling (call) and the create (create) method. The command cmd.exe /c echo ‘netspi’ > C:\text.txt is being supplied as an argument to the create method and will be run.

PowerShell Version 1+ (2006)

With Windows XP / 2003 / Vista / 2008, PowerShell started being introduced. With PowerShell 1.0, several WMI commands were introduced. For now, we are going to focus on just two of them: Get-WmiObject and Invoke-WmiMethod. Get-WmiObject is used to access class properties (read things) and Invoke-WmiMethod is used to invoke the methods (change things).

The previous command could have been replaced with:

```
$Command = "powershell.exe -Command Set-Content -Path C:\text.txt -Value 
netspi";

Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList $Command

__GENUS : 2
__CLASS : __PARAMETERS
__SUPERCLASS :
__DYNASTY : __PARAMETERS
__RELPATH :
__PROPERTY_COUNT : 2
__DERIVATION : {}
__SERVER :
__NAMESPACE :
__PATH :
ProcessId : 892796
ReturnValue : 0
PSComputerName :
```

In this command, we largely follow the procedure that was used in wmic to access the Win32_Process class to invoke the create method. One important note for this command is that with Invoke-WmiMethod the argument are positional parameters.

PowerShell Version 3+ (2012)

In PowerShell Version 3, CIM commands were introduced that even further simplified the use of WMI/CIM, and introduced the concept of reusable CIM sessions and named arguments.

```
$Command = "powershell.exe -Command Set-Content -Path C:\text.txt -Value
netspi";

Invoke-CimMethod -ClassName Win32_Process -MethodName Create -Arguments @{
   CommandLine = $Command
}

ProcessId ReturnValue PSComputerName
--------- ----------- --------------
 911900 0
```

In the next blog, we will cover the basics of using WMI, how to discover useful classes, and how commands can be remotely run.