Detecting CONTI CobaltStrike Lateral Movement Techniques - Part 2 | Cyb3rSn0rlax

Published: 2021-11-01 · Archived: 2026-04-05 22:08:54 UTC

⌘Ctrlk

1. ☢️ DEATH : Detection Engineering And Threat Hunting

2. 🦘TA0008 : Lateral Movement

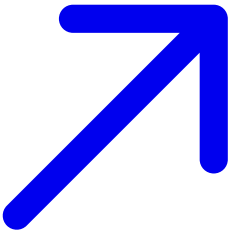
Detecting CONTI CobaltStrike Lateral Movement Techniques - Part 2

Detection opportunities on lateral movement techniques used by CONTI ransomware group using CobaltStrike.

In this second and last part of detecting CONTI lateral movement techniques I will go through the rest of CobaltStrike's built-in capabilities documented in the CONTI leak.

In the first blog post I tried to cover the `jump` command capabilities and detection opportunities where we compared them to some built-in windows utilities.

For the first part, please visit : Detecting CONTI CobaltStrike Lateral Movement Techniques - Part 1



WMI is Microsoft's implementation of **Web-Based Enterprise Management** (WBEM) which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment and **CIM** (Common Information Model) which is an open standard from the Distributed Management Task Force (**DMTF**). CIM provides a common definition of management information for systems, networks, applications, and services.

WMI can be used over `RPC/WinRM` protocol or `RPC/DCOM`. In this introduction I will be focusing on RPC/DCOM.

Data in WMI is grouped into WMI classes. WMI classes are then grouped into WMI namespaces. Most of the WMI classes exist under the root\cimv2 WMI namespace.

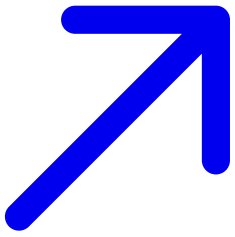
In summary each Namespace contains Classes which have:

- **Methods** : Actions that can be taken.
- **Properties** : Information that can be retrieved.
- **Instances** : Instances of the class objects (services, Processes, Disks) each instance with Methods and Properties.
- **Events** : Actions that WMI can monitor for and take action when they happen.

WMI leverages DCOM server and client interfaces to communicate over the network between Windows Management Instrumentation Remote Protocol clients and servers.

When it comes to lateral movement one of my favorite data sources to check first is Zeek. Upon running the simulated lateral movement attack using CobaltStrike built-in command `remote-exec wmi`, the following telemetry was generated by Zeek.

`zeek.dce_rpc.endpoint` column values are the interfaces while `zeek.dce_rpc.operation` are the methods defined in **WMI** and **DCOM** documentations. This is very helpful in order to understand how WMI looks like from a network perspective. Zeek can identify these GUIDs related to **IWbem** interfaces. A full list is documented



in GitHub source code here .

- **IObjectExporter::ServerAlive** : First we can see RPC binding information calls to the `IObjectExporter` interface using methods `ServerAlive` or `ServerAlive2` to determine server aliveness. Deciding the method is related to the `COMVERSION` in use.
- **IRemoteSCMActivator::RemoteCreateInstance** : The DCOM client MUST support the `Activation` and `OXID Resolution` DCOM mechanisms for creating and resolving object references. `Activation` mechanism can be achieved through two interfaces and three different methods, `IActivation::RemoteActivation` , `IRemoteSCMActivator::RemoteCreateInstance` , or `IRemoteSCMActivator::RemoteGetClassObject`.
- **IRemUnknown2::RemQueryInterface :** Every object can be bound to one or multiple interfaces. An Object reference counter is used to keep track of a Component Object Model (COM) objects. For acquiring

additional interfaces on the object `IRemUnknown::RemQueryInterface` and `IRemUnknown2::RemQueryInterface` calls are used.

- An object reference is represented on the wire by a marshaled form called `OBJREF`.
- **IWbemLevel1Login::NTLMLogin :** According to MS-WMI documentation, during protocol initialization, The client MUST call the `IWbemLevel1Login::NTLMLogin` method.
- **IWbemServices::ExecMethod** : This call will return an interface pointer to `IWbemServices` management services where methods like `GetObject` which retrieves a CIM class or a CIM instance and `ExecMethod` which executes a CIM method that is implemented by a CIM class or a CIM instance, can be used.
- **IRemUnknown2::RemRelease :** The release sequence is then called to decrement the reference counter

Bellow is a mind-map where I tried to summarize the different interfaces and method used during WMI remote calls. This will help understand the telemetry recorded by Zeek in order to identify the best calls to focus our detections on.

As stated in the MS-WMI documentation, during protocol initialization, the client **MUST** call the `IWbemLevel1Login::NTLMLogin` method. This is a good indication of WMI usage over the network. However, a good baseline of users and assets with authorization to use WMI accompanied with a well defined change management process will significantly improve your detection success rate. `IWbemServices::ExecMethod` and `IWbemServices::GetObject` calls are also good indications of WMI accessing web-based management services.

- Zeek Telemetry:

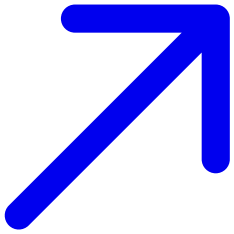
CobaltStrike has a built-in lateral movement module called `remote-exec` which supports three commands : `wmi`, `winrm`, and `psexec`. Remote-Exec module is used to execute a command on a host remotely and doesn't pop a beacon unless it is used for that particular purpose by first uploading a script or a beacon file then execute it via remote-exec commands and use `link` or `connect` commands to assume control of the target.

In this section I will be exploring some generated telemetries from the endpoint perspective using `wmi` command.

- `wmiprvse.exe` process is spawned with the command line `C:\\Windows\\system32\\wbem\\wmiprvse.exe -secured -Embedding` and parent command line `C:\\Windows\\system32\\svchost.exe -k DcomLaunch`.
- EID `5857` was generated to report the start of WMI provider `cimwin32.dll`. There are several WMI providers. This is not very useful because WMI usage can be verbose.
- The command is executed within the context of `*WmiPrvSE.exe*`.
- By default, WMI uses a randomly selected dynamic port range for TCP **between** `49152` **and** `65535`**.**

Detecting malicious usage of WMI relies heavily on WmiPrvse.exe abnormal child processes behavior. However, some approaches can be taken to improve your detections. For example if you have a SCCM server, you might

consider whitelisting the following paths in your process command arguments (Reference



):

```
 C:\\Windows\\CCM\\SystemTemp\\
C:\\Windows\\CCMCache\\
C:\\CCM\\\\Cache\\
```

Keep in mind that attackers might still abuse these paths to evade detections so baselining your assets, source IPs and users that are allowed to use WMI remotely is recommended to increase detection resilience.

By default only Local Administrators or Domain Admins can read WMI class information so in order to further refine your access control policies you can limit regular users permissions by adding them to the Distributed COM Users group and the Performance Monitor Users group.

In the leaked CONTI documentation, we noticed a lot of wmic.exe usage for remote command execution across multiple assets. For example, they use a batch file called WMI.BAT with the following command to spread a binary file across multiple hosts.

```
start wmic /node:@C:\\share$\\comps1.txt /user:"DOMAIN\\Administrator" /password:"PASSWORD" process call create
```

Or interact with beacon through `shell` command to dump credentials :

```
shell wmic /node:[target] process call create "cmd /c rundll32.exe C:\\windows\\System32\\comsvcs.dll, MiniDump
```

WMIC.EXE is one of the Windows built-in utilities that leverages WMI protocol for command execution. For detection opportunities we can look for :

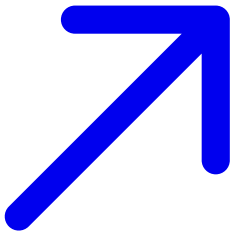
-
- EID `4648` **A logon was attempted using explicit credentials** where the process name is `svchost.EXE` and service class `RPCSS*`. This event is a good DFIR artifact for differentiating between the original account and the account specified in the wmic command (In my case I didn't specify any credentials).

A service principal name (SPN) is the name by which a Kerberos client uniquely identifies an instance of a service for a given Kerberos target computer. There are multiple SPN registrations :

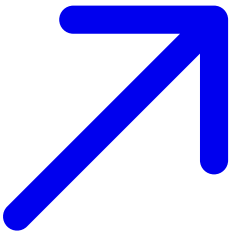
- `HTTP/hostname.contoso.com` like when using PowerShell Remoting via `Enter-PSSession`

- `WSMAN/hostname.contoso.com` like when using WinRM for Remoting
- `CIFS/hostname.contoso.com` like when using PsExec
- `HOST/hostname.contoso.com` for any service running on the computer with hostname `HOSTNAME`

The **RPCSS** service is the Service Control Manager for COM and DCOM servers. It performs object activations requests, object exporter resolutions and distributed garbage collection for COM and DCOM servers (source



). **HOST** service can also be used for remotely executing



commands on the target system via WMI (source).

- On the destination, as previously explained, looking for abnormal behavior of `WmiPvSE.exe` like spawning `PowerShell.exe` and `Cmd.exe` with suspicious arguments would be effective. (see previous table Endpoint for more details)

The table bellow displays WMIC related telemetry generated from the source host :

The following rules present some ideas about detecting malicious WMI behavior.

Atomic Red Team provides a good resource to test your WMI detections

EDR Testing Script :

> *Test the accuracy of Endpoint Detection and Response (EDR) software with simple script which executes various ATT&CK/LOLBAS/Invoke-CradleCrafter/Invoke-DOSfuscation payloads*

To provide more details about the WMI activity for your DFIR engagements, you can use ETW. To enable the event tracing of WMI, you can use the command line:

```
PS C:\> wevtutil.exe sl Microsoft-Windows-WMI-Activity/Trace /e:true
```

Be aware that ETW was made for debugging and enabling WMI event tracing features might generate a lot of data which will be stopped after reaching a certain size/duration limit.

`remote-exec winrm` command is similar to `jump winrm64` in command execution under the context of wmsprovhost.exe except that it was not made for creating and maintaining a remote session hence `wsmprovhost.exe` terminates after execution.

Generated telemetry on the destination :

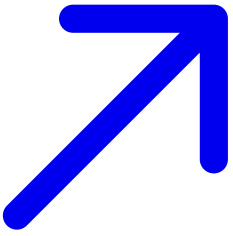
`remote-exec psexec` command creates and start a service remotely with random Service Name and the passed on command as Service File Name. The main difference between this feature and `jump psexec` or `jump psexec64` is that `remote-exec psexec` does not generate a service executable and upload it to the target. As noticed before, CobaltStrike's service file spawns `rundll32.exe` with no arguments which is suspicious.

Monitoring `services.exe` child process for malicious behavior like spawning system shells `cmd.exe` and `powershell.exe` or other discovery binaries like `whoami.exe` , `systeminfo.exe` , `net.exe` ,...etc would be effective against this type of attack.

In the CONTI leaked documentation, the playbook shows the usage of this module to dump `lsass.exe` memory via `comsvcs.dll`

```
remote-exec psexec [target] cmd /c rundll32.exe C:\\windows\\System32\\comsvcs.dll, MiniDump PID C:\\ProgramDa
```

This detection rule from Elastic should be enough to detect such behavior.



See [previous blog] for more details on CobaltStrike `psexec` built-in capabilities detection.

As defined by MITRE in ATT&CK framework:

> *Adversaries may "pass the hash" using stolen password hashes to move laterally within an environment, bypassing normal system access controls. Pass the hash (PtH) is a method of authenticating as a user without having access to the user's cleartext password. This method bypasses standard authentication steps that require a cleartext password, moving directly into the portion of the authentication that uses the password hash.*

CobaltStrike has a built-in module called `pth` to perform pass-the-hash attack using Mimikatz's `sekurlsa:pth` module. As stated by CobaltStrike creator himself this is not OpSec safe since it presents low hanging detection opportunities for defenders.

PTH module has a hardcoded command that contains suspicious sequence of arguments such as `*cmd.exe` `/c` `echo` `>` `\\.\\pipe\*` **.** Monitoring process creation events with such arguments would be effective against

CobaltStrike's way of implementing and automating **pass-the-hash** attack. Keep in mind attackers can always use **Mimikatz PTH** module where they can change these properties.

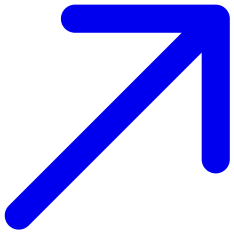
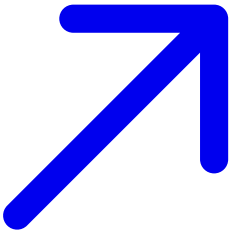
Another key event for detecting pass the hash is `EID 4624` with logon type `9` (**NewCredentials**), logon process `seclogo` and Authentication Package `Negotiate`.

PTH detection observations :

The CONTI leaked documentation shows RDP being used several time for manual access whether to dump `lsass` process memory using task manager or export credentials from users profiles and keyloggers data. This is not an exploitation of the RDP service itself since the attacker already got their hands on user's credentials, so in this case maintaining a good RDP users policy will help creating a baseline and detecting related violations. EID `4825` **A user was denied the access to Remote Desktop** can be helpful in this matter.

I previously created this mind map for **RDP DFIR Authentication** event logs that can be observed in your environment when using RDP with and without NLA enabled.

The mind map was pushed to a great GitHub project started by **Andrew Rathbun** (@bunsofwrath12

) here.

The RDP mind map can be found following this link :

GitHub Project Repository
RDP DFIR Authentication Event Logs PDF

Last updated 4 years ago
