

RED TEAMER AND SECURITY ADDICT

ENIGMA0X3

« BYPASSING APPLICATION WHITELISTING BY
USING RCSI.EXE

LATERAL MOVEMENT VIA DCOM: ROUND 2 »

LATERAL MOVEMENT USING THE MMC20.APPLICATION

COM OBJECT

January 5, 2017 by [enigma0x3](#)

For those of you who conduct pentests or red team assessments, you are probably aware that there are only so many ways to pivot, or conduct lateral movement to a Windows system. Some of those techniques include psexec, WMI, at, Scheduled Tasks, and WinRM (if enabled). Since there are only a handful of techniques, more mature defenders are likely able to prepare for and detect attackers using them. Due to this, I set out to find an alternate way of pivoting to a remote system.

Recently, I have been digging into COM (Component Object Model) internals. My interest in researching new lateral movement techniques led me to DCOM (Distributed Component Object Model), due to the ability to interact with the objects over the network. Microsoft has some good documentation on DCOM [here](#) and on COM [here](#). You can find a solid list of DCOM applications using PowerShell, by running "Get-CimInstance Win32_DCOMApplication".

While enumerating the different DCOM applications, I came across the [MMC Application Class \(MMC20.Application\)](#). This COM object allows you to script components of MMC snap-in operations. While enumerating the different methods and properties within this COM object, I noticed that there is a method named "ExecuteShellCommand" under Document.ActiveView.

You can read more on that method [here](#). So far, we have a DCOM application that we can access over the network and can execute commands. The final piece is to leverage this DCOM application and the `ExecuteShellCommand` method to obtain code execution on a remote host.

Fortunately, as an admin, you can remotely interact with DCOM with PowerShell by using “[`activator`]::CreateInstance([`type`]::GetTypeFromProgID)”. All you need to do is provide it a DCOM ProgID and an IP address. It will then provide you back an instance of that COM object remotely:

```
PS C:\Users\Watt> $com = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.Application","192.168.99.132"))
PS C:\Users\Watt> $com | Get-Member

TypeName: System.__ComObject#{a3afb9cc-b653-4741-86ab-f0470ec1384c}

Name      MemberType Definition
-----
Help      Method      void Help ()
Hide      Method      void Hide ()
Load      Method      void Load (string)
Quit      Method      void Quit ()
Show      Method      void Show ()
Document  Property    Document Document () {get}
Frame     Property    Frame Frame () {get}
UserControl Property    int UserControl () {get} {set}
VersionMajor Property    int VersionMajor () {get}
VersionMinor Property    int VersionMinor () {get}
Visible   Property    int Visible () {get}

PS C:\Users\Watt>
```

It is then possible to invoke the “`ExecuteShellCommand`” method to start a process on the remote host:

As you can see, calc.exe is running under Matt while the user “Jason” is logged in:

By using this DCOM application and the associated method, it is possible to pivot to a remote host without using psexec, WMI, or other well-known techniques.

To further demonstrate this, we can use this technique to execute an agent, such as [Cobalt Strike's Beacon](#), on a remote host. Since this is a lateral movement technique, it requires administrative privileges on the remote host:

As you can see, the user “Matt” has local admin rights on “192.168.99.132”. You can then use the ExecuteShellCommand method of MMC20.Application to execute staging code on the remote host. For this example, a simple encoded PowerShell download cradle is specified. Be sure to pay attention to the requirements of “ExecuteShellCommand” as the program and its parameters are separated:

The result of executing this through an agent results in obtaining access to the remote target:

To detect/mitigate this, defenders can [disable DCOM](#), block RPC traffic between workstations, and look for a child process spawning off of “mmc.exe”.

Edit: After some investigating and [back & forth](#) with [James Forshaw](#), it appears that the Windows Firewall will block this technique by default. As an additional mitigation, ensure the windows firewall is enabled and “Microsoft Management Console” isn’t an enabled rule.

Cheers!
Matt N.

SHARE THIS:

 Twitter

 Facebook

Loading...

Bookmark the [permalink](#).

LEAVE A COMMENT

ARCHIVES

- [October 2023](#)
- [January 2020](#)
- [December 2019](#)
- [August 2019](#)
- [July 2019](#)
- [March 2019](#)
- [January 2019](#)
- [October 2018](#)
- [June 2018](#)
- [January 2018](#)
- [November 2017](#)
- [October 2017](#)
- [September 2017](#)
- [August 2017](#)
- [July 2017](#)
- [April 2017](#)
- [March 2017](#)
- [January 2017](#)
- [November 2016](#)
- [August 2016](#)
- [July 2016](#)
- [May 2016](#)
- [March 2016](#)
- [February 2016](#)
- [January 2016](#)
- [October 2015](#)
- [August 2015](#)
- [April 2015](#)
- [March 2015](#)
- [January 2015](#)
- [October 2014](#)
- [July 2014](#)
- [June 2014](#)
- [March 2014](#)
- [January 2014](#)

RECENT POSTS

- [CVE-2023-4632: Local Privilege Escalation in Lenovo System Updater](#)
- [Avira VPN Local Privilege Escalation via Insecure Update Location](#)
- [CVE-2019-19248: Local Privilege Escalation in EA's Origin Client](#)
- [Avira Optimizer Local Privilege Escalation](#)
- [CVE-2019-13382: Local Privilege Escalation in Snagit](#)

CATEGORIES

- [Uncategorized](#)

RECENT COMMENTS

- Ron on [CVE-2019-13382](#)
- enigma0x3 on [CVE-2019-13382: Local Privilege Escalation in Snagit](#)
- Ron on [CVE-2019-13382: Local Privilege Escalation in Snagit](#)
- Soc on [Defeating DeviceGuard](#)
- "Fileless..." on ["Fileless" Lateral Movement](#)

META

- [Register](#)
- [Log in](#)
- [Entries feed](#)
- [Comments feed](#)
- [WordPress.com](#)

[Blog at WordPress.com.](#)