

# BOHOPS

A blog about cybersecurity research, education, and news

WRITTEN BY BOHOPS  
OCTOBER 8, 2021

## QUICK LINKS

- [Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence \(Part 2\)](#)
- [Abusing .NET Core CLR Diagnostic Features \(+ CVE-2023-33127\)](#)
- [Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence](#)
- [Abusing the COM Registry Structure \(Part 2\): Hijacking & Loading Techniques](#)
- [Abusing the COM Registry Structure: CLSID, LocalServer32, & InprocServer32](#)
- [WS-Management COM: Another Approach for WinRM Lateral Movement](#)
- [DiskShadow: The Return of VSS Evasion, Persistence, and Active Directory Database Extraction](#)
- [Analyzing and Detecting a VMTools Persistence Technique](#)
- [Investigating .NET CLR Usage Log Tampering Techniques For EDR Evasion \(Part 2\)](#)
- [Executing Commands and Bypassing AppLocker with PowerShell Diagnostic Scripts](#)

## ANALYZING AND DETECTING A VMTOOLS PERSISTENCE TECHNIQUE

### INTRODUCTION

It is always fun to reexplore previously discovered techniques or pick back on old research that was put on the wayside in hopes to maybe finding something new or different. Recently, I stood up an ESXi server at home and decided to take a quick peak at the VMware directory structure after installing the VMware Tools (vmtools) package in a Windows 10 Virtual Machine.

Among the directory contents were some batch files that I forgot about and the very interesting binary – VMwareToolBoxCmd.exe. After some quick Googling, it did not take long to land on Adam’s (@Hexacorn) incredible blog to find these two very informative post about VMwareToolBoxCmd.exe, OS fingerprinting, and a privileged persistence technique with VMware Tools:

- [Beyond good ol’ Run key, Part 53](#)
- [Using Virtual Machine tools for Guest OS fingerprinting](#)

In this quick post, we will analyze this persistence technique and discuss a few strategies for detecting potential abuse.

### THE TECHNIQUE

As Adam describes, VMwareToolBoxCmd.exe is a utility command for capturing VM information or changing the configuration of various and sundry virtual machine settings. One feature is to control batch scripts that can be configured to run based on VM state operations including *power* (power on), *shutdown* (power off), *resume* (from

suspended state), and and *suspend* (entering suspended stated) as noted in the command utilities *script* help subcommand:




```
c:\Program Files\VMware\VMware Tools>VMwareToolboxCmd.exe help script
script: control the scripts run in response to power operations
Usage: VMwareToolboxCmd.exe script <power|resume|suspend|shutdown> <subcommand> [args]

Subcommands:
  enable: enable the given script and restore its path to the default
  disable: disable the given script
  set <full_path>: set the given script to the given path
  default: print the default path of the given script
  current: print the current path of the given script
  NOTE: If the path is not present in tools.conf, its
  value from the global configuration is returned if present
```

There are several built-in batch scripts in the VMware Tools directory, but this does not preclude someone from using and enabling a custom script. For example, the following command script can be used to specify the execution of a custom script when the VM is powered on:

```
VMwareToolboxCmd.exe script power set "c:\evil\evilscrip.bat"
VMwareToolboxCmd.exe script power enable
```

The command sequence itself is not as interesting as what it actually does. In the following Sysmon screenshot, we can see that content is actually written to the *tools.conf* file in \ProgramData:

 Event	 Process	 Stack
Date:	10/3/2021 12:13:40.9646514 PM	
Thread:	3900	
Class:	File System	
Operation:	WriteFile	
Result:	SUCCESS	
Path:	C:\ProgramData\VMware\VMware Tools\tools.conf	
Duration:	0.0000324	
Offset:	0	
Length:	103	
Priority:	Normal	

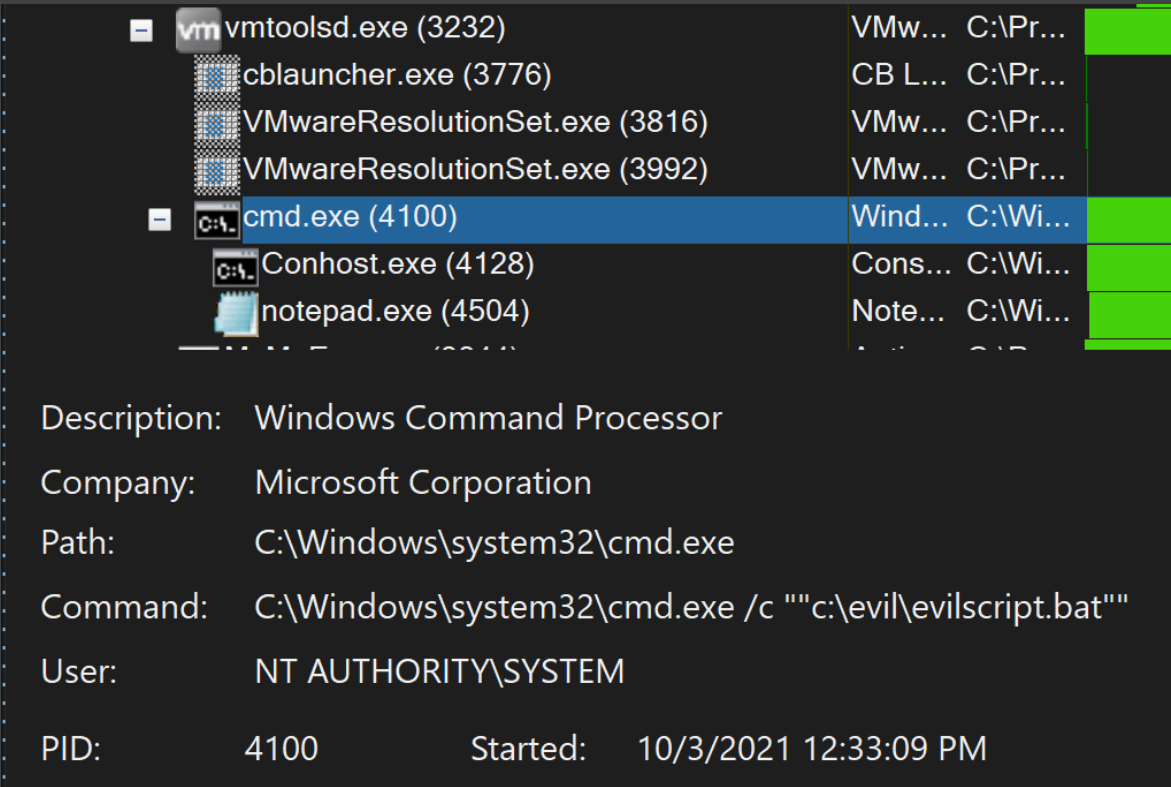
Upon further inspection, the contents of this file appear as follows:

```
c:\Program Files\VMware\VMware Tools>type "C:\ProgramData\VMware\VMware Tools\tools.conf"
[powerops]
poweron-script=c:\\evil\\evilscrip.bat
resume-script=c:\\windows\\system32\\notepad.exe
```

Coincidentally, there is another operation for *resume* under the *powerops* section directive. This was added previously by me to show that 1) batch files are not the only thing that can be configured and 2) the *tools.conf* file is the key component what enables the script execution functionality.

**Note:** For a complete example of what a configuration file may look like, take a look at *tools.conf.example* in the same \ProgramData directory or this [sample](#) file in VMware’s open-vm-tools repository.

After a quick shutdown and power-on, we can see our batch file payload (notepad.exe) is executed by cmd.exe as a child process of *vmtoolsd.exe* under the context of NT AUTHORITY\SYSTEM:



## DEFENSIVE CONSIDERATIONS

Consider the following detection opportunities:

### Sysmon

For event collection with Sysmon, consider monitoring *tools.conf* write (modification) events with an experimental rule. The following rule can be added to [@SwiftOnSecurity’s Sysmon-Config](#) under the *EVENT 11: “File created”* section or under [@olafhartong’s Sysmon-Modular](#) “11\_file\_create” rules:

```
<TargetFilename>C:\ProgramData\VMware\VMware
Tools\tools.conf</TargetFilename>
```

### Elastic Security

I’ve been digging into the Elastic Stack in recent months and felt that it would be a great opportunity to build a simple rule in Elastic Security. Conveniently, Elastic was kind enough to implement a rule creation wizard. Leverage this by selecting Elastic Security in Kibana, navigating to “Rules”, then selecting “Create New Rule”:

I created this ‘custom’ rule based on the Event Query Language (EQL) of another [rule](#) [License: [Elastic License v2](#)]:

```
file where event.type != "deletion" and
file.path :
(
"C:\ProgramData\VMware\VMware Tools\tools.conf"
)
```

After walking through the wizard and enabling the rule, I modified the *tools.conf* file which triggered this alert:

Of note, the community can contribute to Elastic’s open-source Detection Rules [repository](#). There is a set of instructions to leverage a Python utility to help with the creation and validation process (outlined [here](#)).

## Other Detection Opportunities

\***Environment:** In some environments, it is very plausible that operational power scripts/commands may already be enabled for legitimate reasons. If such is the case, audit the *tools.conf* file for target scripts and monitor accordingly. Although custom scripts can be specified, the following (default) operational state scripts are included with VMware Tools (in the \VMware Tools directory) and may be worth monitoring:

- poweroff-vm-default.bat
- poweron-vm-default.bat
- resume-vm-default.bat
- suspend-vm-default.bat

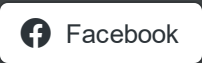
\***Hunt:** As shown in a previous screenshot, the parent process for the launched process is *vmtoolsd.exe*. Consider monitoring or hunting for suspicious child processes. Additionally, monitoring for VMwareToolBoxCmd.exe command usage could be opportunistic in some environments.

# CONCLUSION

As always, thank you for taking the time to read this post.

~ bohops

SHARE THIS:



Loading...

RELATED

Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence (Part 2)  
March 10, 2018  
In "applocker"

Abusing the COM Registry Structure: CLSID, LocalServer32, & InprocServer32  
June 28, 2018  
Liked by 1 person

Abusing the COM Registry Structure (Part 2): Hijacking & Loading Techniques  
August 18, 2018  
With 1 comment

PREVIOUS POST

NEXT POST