MOBIA is a business technology integrator with over thirty years of experience and 500+ employees across Canada and USA. Our talented bench of technical engineers and trusted advisors deliver process improvements and business transformations within our core pillars of Cloud, Infrastructure, Software Development, Cybersecurity and Broadband & Wireless Services.

Our inside-out approach allows us to understand business challenges from deep within the organization, mapping the impact as it ripples outward. This insight enables us to create future-proof solutions that maximize results and repeatedly exceed our client's expectations.

Adversary Simulation Workshop

Lab Guide:
Specify Custom Input Arguments

Objective:

Specify custom input arguments during atomic test execution.

Instructions:

To learn about input arguments we will take a closer look at one of the tests under T1016.

```
Invoke-AtomicTest T1016 -ShowDetailsBrief
```

```
PS C:\users\art> Invoke-AtomicTest T1016 -ShowDetailsBrief
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

T1016-1 System Network Configuration Discovery on Windows
T1016-2 List Windows Firewall Rules
T1016-4 System Network Configuration Discovery (TrickBot Style)
T1016-5 List Open Egress Ports
```

If you notice, the test numbers might look a little strange, as it leaves out test number 3. Let's find out why. Go to the following page to see the Information about T1016.

https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1016/T1016.md#atomic-tests

You can see that there are 5 total Atomic Tests for this one T#.

Atomic Tests

- Atomic Test #1 - System Network Configuration Discovery on Windows
- Atomic Test #2 - List Windows Firewall Rules
- Atomic Test #3 - System Network Configuration Discovery
- Atomic Test #4 - System Network Configuration Discovery (TrickBot Style)
- Atomic Test #5 - List Open Egress Ports

So why don't they all show up when we look at it in PowerShell? Look at more specific details for Atomic Test #3 to see why it didn't show up earlier.

Atomic Test #3 - System Network Configuration Discovery

Identify network configuration information.

Upon successful execution, sh will spawn multiple commands and output will be via stdout.

Supported Platforms: macOS, Linux

Attack Commands: Run with `sh` !

```
if [ -x "$(command -v arp)" ]; then arp -a; else echo "arp is missing from the machine. skipping..."; fi;
if [ -x "$(command -v ifconfig)" ]; then ifconfig; else echo "ifconfig is missing from the machine. skipping...";
if [ -x "$(command -v ip)" ]; then ip addr; else echo "ip is missing from the machine. skipping..."; fi;
if [ -x "$(command -v netstat)" ]; then netstat -ant | awk '{print $NF}' | grep -v '[a-z]' | sort | uniq -c; else
```

The supported platforms for test #3 are Linux and macOS. Since we are on a Windows machine, these tests either don't apply and are not listed.

Let's look at the information for test number 5.

Atomic Test #5 - List Open Egress Ports

This is to test for what ports are open outbound. The technique used was taken from the following blog: https://www.blackhillsinfosec.com/poking-holes-in-the-firewall-egress-testing-with-allports-exposed/

Upon successful execution, powershell will read top-128.txt (ports) and contact each port to confirm if open or not. Output will be to Desktop\open-ports.txt.

Supported Platforms: Windows

Inputs:

Name	Description	Type	Default Value
output_file	Path of file to write port scan results	Path	$env:USERPROFILE\Desktop\open-ports.txt
portfile_url	URL to top-128.txt	Url	https://github.com/redcanaryco/atomic-red-team/raw/master/atomics/T1016/src/top-128.txt
port_file	The path to a text file containing ports to be scanned, one port per line. The default list uses the top 128 ports as defined by Nmap.	Path	PathToAtomicsFolder\T1016\src\top-128.txt

Notice that this test has input arguments (inputs). Look at the table and you can see the name of the input, as well as its default value.

Let's use the PowerShell Execution Framework to look at the details for test #5.

```
Invoke-AtomicTest T1016 -TestNumbers 5 -ShowDetails
```

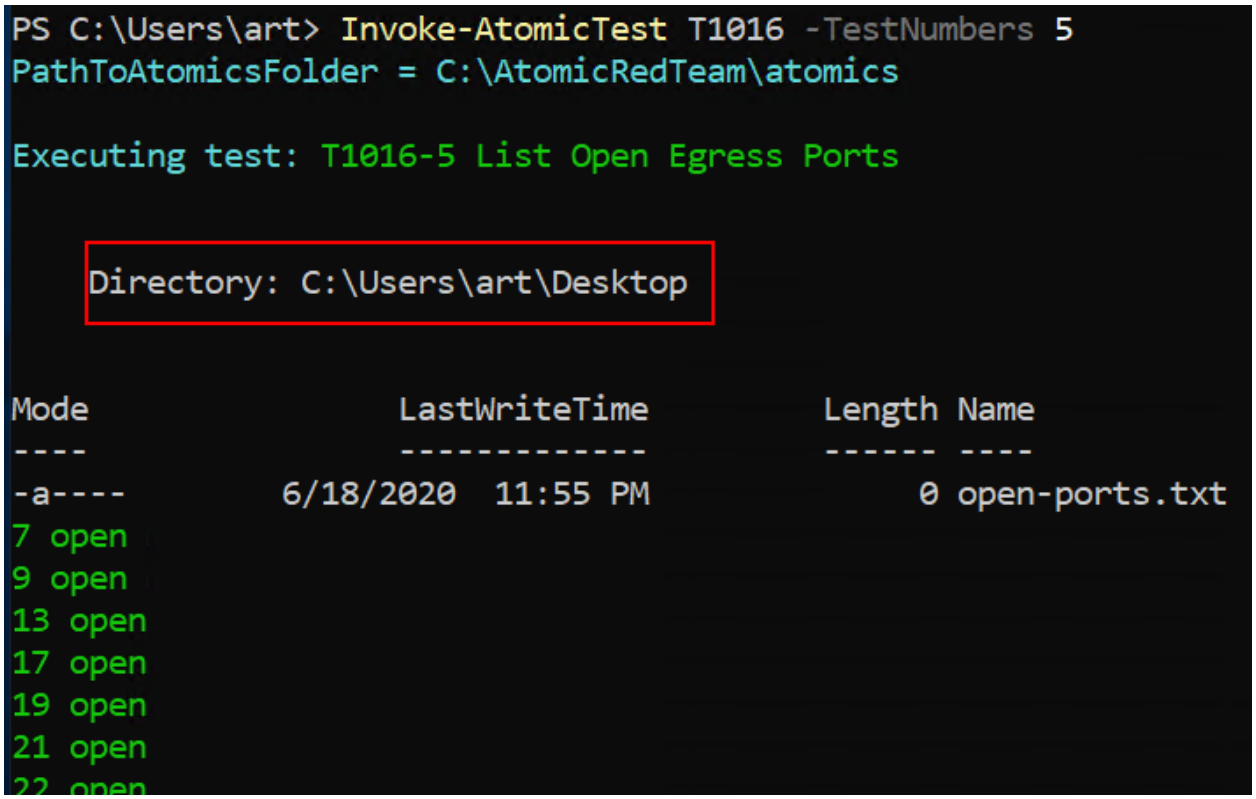
Scroll down to the "Command" section.

```
Attack Commands:
Executor: powershell
ElevationRequired: False
Command:
$ports = Get-content #{port_file}
$file = "#{output_file}"
$totalopen = 0
$totalports = 0
New-Item $file -Force
foreach ($port in $ports) {
    $test = new-object system.Net.Sockets.TcpClient
    $wait = $test.beginConnect("allports.exposed", $port, $null, $null)
    $wait.asyncwaithandle.waitone(250, $false) | Out-Null
    $totalports++ | Out-Null
    if ($test.Connected) {
        $result = "$port open"
        Write-Host -ForegroundColor Green $result
        $result | Out-File -Encoding ASCII -append $file
        $totalopen++ | Out-Null
    }
    else {
        $result = "$port closed"
        Write-Host -ForegroundColor Red $result
        $totalclosed++ | Out-Null
        $result | Out-File -Encoding ASCII -append $file
    }
}
$results = "There were a total of $totalopen open ports out of $totalports ports tested."
$results | Out-File -Encoding ASCII -append $file
Write-Host $results
Command (with inputs):
$ports = Get-content C:\AtomicRedTeam\atomics\T1016\src\top-128.txt
$file = "$env:USERPROFILE\Desktop\open-ports.txt"
$totalopen = 0
$totalports = 0
New-Item $file -Force
foreach ($port in $ports) {
    $test = new-object system.Net.Sockets.TcpClient
    $wait = $test.beginConnect("allports.exposed", $port, $null, $null)
    $wait.asyncwaithandle.waitone(250, $false) | Out-Null
    $totalports++ | Out-Null
    if ($test.Connected) {
```

There are two versions of the commands listed, one with and one without the input arguments substituted in, “Command” vs “Command (with inputs)”. The field outlined in red in the “Command” section is the input argument, the default value is outlined in red in “Command (with inputs)” section. The same is true for the variables outlined in blue.

Let’s run the test and view the output. Because we aren’t specifying any input arguments, the default values will be used.

```
Invoke-AtomicTest T1016 -TestNumbers 5
```



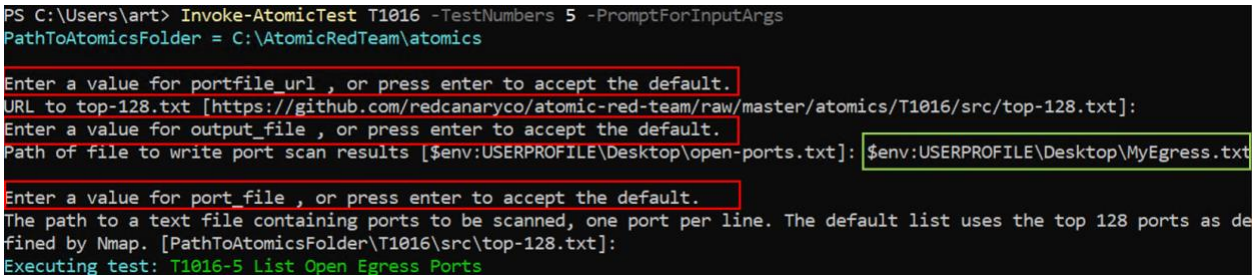
The output_file (c:\Users\art\Desktop\open-ports.txt) should have shown up on your desktop.

A note about “PowerShell ISE”: PowerShell ISE handles output redirection funny and you won’t see the output from the atomics run, such as the list of open ports above. We suggest not using PowerShell ISE for this reason, and instead using the plain “PowerShell” prompt.

If we want to be prompted to enter our own values for the input arguments we can use the “PromptForInputArgs” flag.

```
Invoke-AtomicTest T1016 -TestNumbers 5 -PromptForInputArgs
```

Here we are prompted to enter a value for each of the three input arguments as shown in red. The default value for the input argument is shown inside the square brackets and can be accepted by just pressing Enter or Return. In this example we set one custom argument for the output_file argument as shown in green.

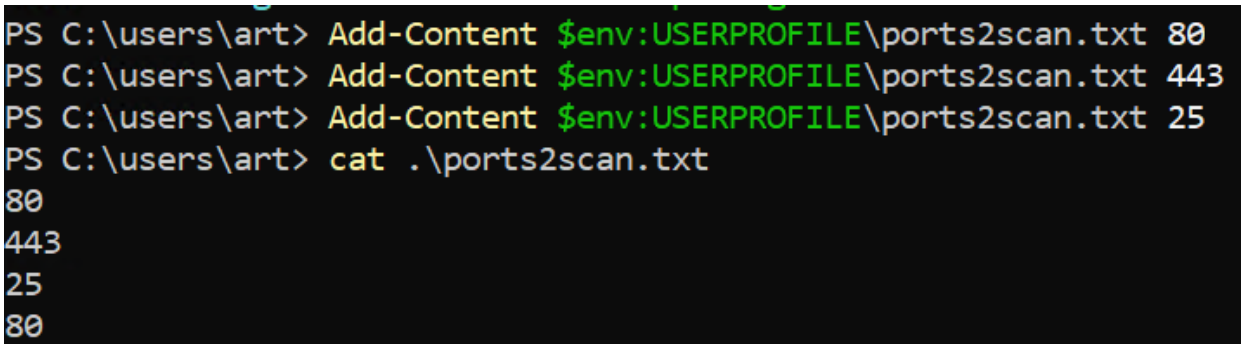


Now when we execute the test it will write the open ports to a file on the Desktop called “MyEgress.txt”

Being prompted interactively to enter values for input arguments is handy for one-off interactive execution but there is also another way to specify input arguments which is especially useful when using a non-interactive script to execute the test.

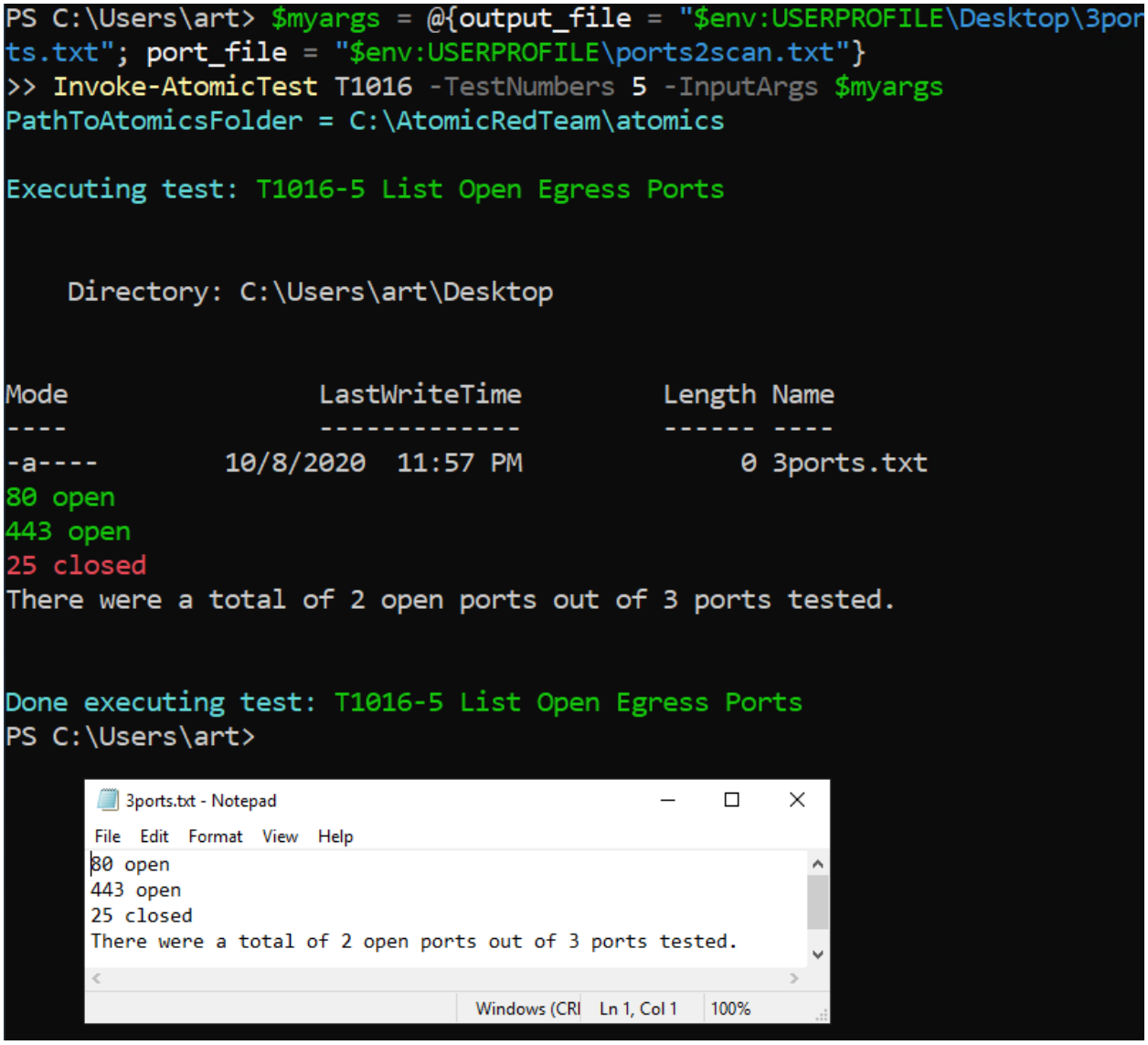
For this example, we will create a new ports.txt file with only a few ports in the list. The three commands below will add 80, 443 and 25 to a file called ports2scan.txt.

```
Add-Content $env:USERPROFILE\ports2scan.txt 80
Add-Content $env:USERPROFILE\ports2scan.txt 443
Add-Content $env:USERPROFILE\ports2scan.txt 25
```

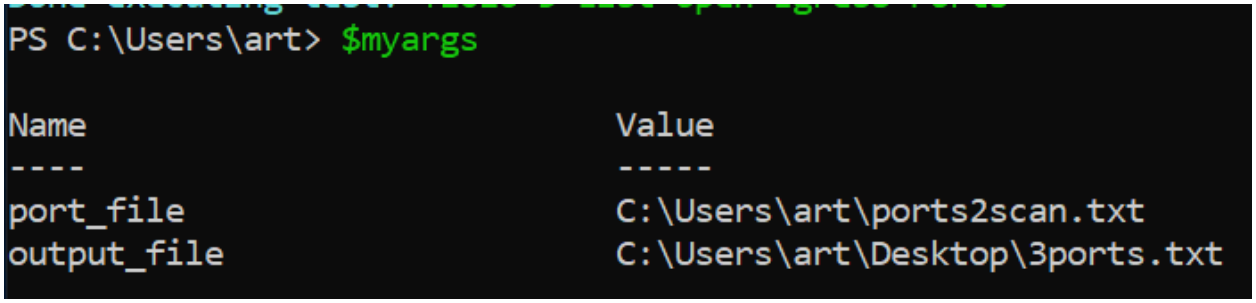


We will use a variable called “myargs” to store our custom arguments which we will then pass to the Invoke-AtomicTest call.

```
$myargs = @{output_file =
"$env:USERPROFILE\Desktop\3ports.txt"; port_file =
"$env:USERPROFILE\ports2scan.txt"}
Invoke-AtomicTest T1016 -TestNumbers 5 -InputArgs $myargs
```



We specified two of the three input arguments to use our values instead of the default.



With the $myArgs variable, we specified a port_file of C:\Users\art\port2scan.txt and a output_file of C:\Users\art\Desktop\3ports.txt

This caused the test to port scan 3 ports (80, 443, and 25) and to write its output to “3ports.txt” on the Desktop. This was all done without being interactively prompted for the input argument values.

This completes the custom input arguments lab. In the next lab we will be learning how to run the clean up commands in order to do some post-execution clean up.

End of Lab