## w2wcap: circular trace, stop when event detected

June 13th, 2017[Garland](http://web.archive.org/web/20200810084702/http:/www.garlandjoseph.org/author/admin/)[Leave a comment](http://web.archive.org/web/20200810084702/http:/www.garlandjoseph.org/w2wcap/#respond)[Go to comments](http://web.archive.org/web/20200810084702/http:/www.garlandjoseph.org/w2wcap/#comments)

This script is part of a series of scripts that perform packet capture between two endpoints.  In this case, the endpoints are two windows machines.  This script was tested with the “source endpoint” as a Windows 10 machine and the “target endpoint” a Windows 2016 Server machine.   The circular traces are started on each machine and stopped whenever an event is detected.  In this case the event is to monitor a file for a particular string.

Requirements: Wireshark installed on WindowsCaveats: Target endpoint must be Windows Server (Source endpoint can be windows client).

<#

################################################################################################

.SYNOPSIS

Author: Garland R.Joseph, garland.joseph@gmail.com, July 2016

Simple Packet capture from windows to windows based on monitoring a file.

.DESCRIPTION

Side-effects

Uses an interim x.x and local\_trace.ps1 file, but removes them at end.

.EXAMPLE

w2wcap [ -verbose | -v ] [ -c "capture-file" ] [ -s "seconds" ] [ -u "remote user" ] [ -p "password" ]

-r "remote host" -l "log file" -t "text string" -a "local interface" -b "remote interface"

.PARAMETER v | verbose

Optional. This option prints additional informational messages during processing.

.PARAMETER c

Optional. This options specifies the name of the capture file. The default is set

to capture. The script is set to only create 2 files per endpoint and the default

size is 512 MB. So the 2 files total will be less than a 1 GB. You can modify this

behavior by changing the FILESIZE and FILECOUNT variables below. The filesname will

look similar to...

capture\_00049\_20170612214134

capture\_00050\_20170612215042

.PARAMETER s

Optional. Range is 1 second to 18000 seconds (5 hours) for parsing the log file. The default

is 5 seconds. Please examine the process to gauge impact on system resources.

The lower the number, the more system resources are consumed.

.PARAMETER u

Optional. This is the username for the remote user on the target endpoint. You can hard

code the value in the script (see the $u=Wireshark line below) or

optionally specify the name on the command line. The account should have

sufficient privilege to run the tshark command tool for the Wireshark facility.

This parameter is optional, but recommended so you don't store the username in a file.

.PARAMETER p

Optional. This is the password for the remote user. This parameter is optional, but

recommended so you don't store the password in a file.

.PARAMETER r

Required. This is the hostname or ip address of the target endpoint.

.PARAMETER l

Required. This is the log file that will be parsed for the string as specified

by parameter t. The traces will stop once a new string specified by parameter t

is found in the log file. The script parses for an initial count so the log file

does not have to be "zeroed-out".

.PARAMETER t

Required. This the text string used in parsing the log file.

.PARAMETER a

Required. This is the ip address of the local interface on the source

endpoint that "talks" to the target endpoint.

.PARAMETER b

Required. This is the ip address of the remote interface on the target

endpoint.

###############################################################################################

#>

#

# Setup Command Line Options

#

[CmdletBinding()]

Param (

[string]$c = "capture",

[ValidateRange(1,18000)]

[int]$s = 5,

[string]$u = "Wireshark",

[string]$p = "Pr1nceH1ll",

[Parameter(Mandatory=$True)]

[string]$r,

[Parameter(Mandatory=$True)]

[string]$l,

[Parameter(Mandatory=$True)]

[string]$t,

[Parameter(Mandatory=$True)]

[int]$a,

[Parameter(Mandatory=$False)]

[int]$b

)

If (-Not (Test-Path $l ) ) {

Write-Host "Log file $l does not exist"

exit

}

#

# Default options, can be changed to increase capture file size, count, etc

#

#$FILESIZE=1000 #units or kB, so this means 1 Meg

$FILESIZE=500000 #512 Meg

#$FILESIZE=1000000 #units or kB, so this means 1 Gig

$FILECOUNT=2 #creates a count of FILECOUNT of trace files at most of size FILESIZE

$TSHARK\_LOCATION\_REMOTE="c:\progra~1\wireshark\tshark"

$TSHARK\_LOCATION\_LOCAL="c:\progra~1\wireshark\tshark"

$TRACECMD\_REMOTE="$TSHARK\_LOCATION\_REMOTE -b filesize:$FILESIZE -b files:$FILECOUNT -w $c -i $b"

$TRACECMD\_LOCAL="$TSHARK\_LOCATION\_REMOTE -b filesize:$FILESIZE -b files:$FILECOUNT -w $c -i $a"

#

# Set up script block in order to evaluate parameters for remote command

# We use an interim file for asynchronously running the local trace

#

$sb\_remote = {

param ($p1,$p2,$p3,$p4,$p5,$p6)

winrs /r:$p1 /u:$p2 /p:$p3 $p4

}

$TRACECMD\_LOCAL &gt; ./local\_trace.ps1

#

# Start trace on remote host, then on local host

#

Write-Verbose "Starting remote trace with ${TRACECMD\_REMOTE}"

Write-Verbose "(winrs /r:$r /u:$u /p:$p $TRACECMD\_REMOTE)"

Start-Job -Scriptblock $sb\_remote -ArgumentList $r,$u,$p,$TRACECMD\_REMOTE

Write-Verbose "Starting local trace with ${TRACECMD\_LOCAL}"

Start-Process powershell.exe -ArgumentList "-file ./local\_trace.ps1"

#

# Monitor log file

#

#

Write-Verbose "Start of monitoring file $l ever $s seconds for string $t..."

#init counters

Select-string -path "$l" -pattern $t | Measure-object -line | ft -hidetableheaders &gt; x.x ; $old\_count = cat x.x | where {$\_ -ne ""} | %{$\_ -replace '\s+','' }

$new\_count=$old\_count

#loop until match found

$i=0

while ($new\_count -eq $old\_count) {

$i++

write-verbose "Iteration: $i, sleeping $s seconds..."

start-sleep -s $s

Write-verbose "Searching $l for $t..."

Select-string -path "$l" -pattern $t | Measure-object -line | ft -hidetableheaders &gt; x.x ; $new\_count = cat x.x | where {$\_ -ne ""} | %{$\_ -replace '\s+','' }

}

#

# At this point, search string has been found, stop traces

#

#remote

taskkill /f /s $r /u $u /p $p /fi "imagename eq tshark\*"

#local

taskkill /f /fi "imagename eq tshark\*"

Write-Verbose "Traces completed after $i iterations. Examine the set of $FILECOUNT files on each endpoint with file name: $l on both endpoints."

#

# Clean up

#

if (Test-Path "./x.x") { rm "./x.x" }

if (Test-Path "./local\_trace.ps1") { rm ./local\_trace.ps1 }

exit