July 5th, 2017 | Garland Joseph

This script is part of a series of scripts that perform packet capture between two endpoints.  In this case, the endpoints are a Unix machine and a windows machine. This script was tested with the “source endpoint” as a Redhat Linux and the “target endpoint” a Windows 2016 Server machine.

The circular traces are started on each machine and stopped whenever an event is detected on the Unix side.  In this case the event is to monitor a file for a particular string.

Requirements: Wireshark installed on Windows.  OpenSSH installed on Linux.

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

# Date: May 2017

#

# This script is offered as is. It is designed to

# run a circular trace using tcpdump on UNIX system

# and wireshark on Windows systems.

#

# You will either have to manually enter the password

# for the root account on the remote system or setup

# ssh keys from promptless access.

#

# The traces will stop once a key string SEARCH\_STRING is

# found in LOG\_FILE.

#

# Note: Some UNIX systems like LINUX Fedora will

# result in permsission denied when using

# tcpdump -W and -C options and writing to / or /root.

#

# Modify the REM\_INTERFACE parameter below to fix the interface number

# on the windows system. Do a tshark -D to determine the interface number.

#

#

# -----

#

# Defaults

#

USAGE="u2wcap [-v] [ -c capture\_file ] [ -w secs ] -h remote\_host -l log\_file -s search\_string"

DEBUG=false

SLEEP\_TIME="5" #seconds

LOCAL\_CAPTURE\_FILE="/tmp/capture"

TCPDUMPCMD="tcpdump -C 1 -W 2 -w ${LOCAL\_CAPTURE\_FILE}"

#

# Options for remote tracing

#

REM\_CAP\_FILE="capture.windows"

REM\_USER="wireshark"

REM\_INTERFACE="4"

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="c:\progra~1\wireshark\tshark"

#TRACECMD="$TSHARK\_LOCATION -b filesize:$FILESIZE -b files:$FILECOUNT -w ${REM\_CAP\_FILE}"

TRACECMD="$TSHARK\_LOCATION -b filesize:$FILESIZE -b files:$FILECOUNT -w ${REM\_CAP\_FILE} -i ${REM\_INTERFACE}"

#

# Process command line arguments

#

while getopts ":vc:w:l:s:h:" opts

do

case ${opts} in

v) DEBUG=true ;;

c) CAPTURE\_FILE=${OPTARG} ;;

w) SLEEP\_TIME=${OPTARG} ;;

s) SEARCH\_STRING=${OPTARG} ;;

l) LOG\_FILE=${OPTARG} ;;

h) REMOTE\_HOST=${OPTARG} ;;

":") echo "Please specify a value for ${OPTARG}" ; exit ;;

\?) echo "${OPTARG} is not a valid switch" ; echo "${USAGE}" ; exit;;

esac

done

#

# Insure required values have been specified, check for existence of

# log file, getops should handle case of no values for -l and -s.

# A sanity check in the event getopts varies per unix

#

if [[ -z ${SEARCH\_STRING} || -z ${LOG\_FILE} || -z ${REMOTE\_HOST} ]]

then

echo ${USAGE}

exit

fi

if ! [[ -f ${LOG\_FILE} ]]

then

echo "File ${LOG\_FILE} does not exist"

exit

fi

#

# Start trace on remote host

#

$(ssh ${REM\_USER}@${REMOTE\_HOST} ${TRACECMD})& 2>&1 > /dev/null

#

# Start trace on this host

#

${TCPDUMPCMD} 2>/dev/null 1>/dev/null &

LOCAL\_PID=$!

${DEBUG} && echo "${0}-I-LOCAL\_PID, local pid is ${LOCAL\_PID}."

#

# Monitor log file

#

old\_count=`grep -c ${SEARCH\_STRING} ${LOG\_FILE}`

(( new\_count=old\_count ))

(( i = 0 ))

while (( old\_count == new\_count ))

do

(( i++ ))

${DEBUG} && echo "${0}-F-SLEEP, sleeping ${SLEEP\_TIME}, iternation ${i}."

sleep ${SLEEP\_TIME}

new\_count=`grep -c ${SEARCH\_STRING} ${LOG\_FILE}`

done

#

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

#

kill ${LOCAL\_PID}

ssh ${REM\_USER}@${REMOTE\_HOST} taskkill /f /fi \"imagename eq tshark\*\"

#

# Reminders

#

echo "Consult files ${REM\_CAP\_FILE} on remote host ${REMOTE\_HOST} and ${LOC\_CAP\_FILE} on local host."

exit