BPF process for windows

<https://github.com/microsoft/ebpf-for-windows>

1. Installing VM -> Hyper-V, not with links

* Install ISO (disk image) windows 11
* Create VM (<https://www.youtube.com/watch?v=QvOmaf1jFbE>) (Development environment 11)

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/vm-setup.md>

* Network debugging
* <https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-network-debugging-of-a-virtual-machine-host>

1. Building bpf for windows

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/GettingStarted.md>

* Prerequisites: 1-2-3-4-5-6-7-8-9
* Add to path the .exe
* Copy repository (NO NEED) -> CMake

git clone --recurse-submodules <https://github.com/microsoft/ebpf-for-windows.git>

C:\Windows\System32

1. Installing BPF

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/GettingStarted.md#installing-ebpf-for-windows>

* Test signing mode -> debugger attached
* Driver test signed NO
  + Signtool needs to be added to the environment variables in Path

(windows kits / 10 / bin / build / x64)

* Test certificate installed NO

1. Installing eBPF into a test VM (MSI)

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/InstallEbpf.md>

* Method 1

1. eBPF tutorial

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/tutorial.md>

* done

1. Work in environment similar to libbpf-bootstrap

<https://blog.subcom.tech/ebpf-programming-on-windows/>

Followed process

1. Install VM -> DONE

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/vm-setup.md> -> DONE

* Setup VM -> DONE
  + Windows VM on Hyper-V
  + Enable test-signed binaries
* Install all prerequisites

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/GettingStarted.md>

* Debug VM -> DONE
  + Setting network debugging

<https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-network-debugging-of-a-virtual-machine-host> -> DONE

* + - Hyper-V VM setup (WDK on VM for debug point 4)

<https://learn.microsoft.com/en-us/windows-hardware/drivers/download-the-wdk>

* + - Setting up network debugging of VM

External VM Switch -> IPv4: 192.168.1.7 -> CAMBIA

* + - Setting up VM target

Port -> 50005

Comando -> kdnet 192.168.1.7 50005

“”””””””””””””””

Enabling network debugging on Network debugging is supported by this Microsoft Hypervisor Virtual Machine.

To debug this vm, run the following command on your debugger host machine.

windbg -k net:port=50005,key=3q98fqaazvxgb.9af14h7lvxls.2cxecb7hhc3y1.1id4mb4hwgr6w

Then restart this VM by running shutdown -r -t 0 from this command prompt.

“”””””””””””””””

* + - Install WinDbg

Installed in C:\Program Files (x86)\Windows Kits\10\Debuggers\x64

* + - Execute the command in “””…””” in host

Connection for debugging VM

Immagine che contiene testo, elettronica, schermata, schermo

Descrizione generata automaticamenteDO NOT ABILITATE SECURE BOOT

IF NOT RUNNING -> shutdown -r -t 0 (reboot VM)

* + Enable driver verifier on eBPF drivers -> DONE

“””””””””””””””””””””””””””””””””””””””

Each time, before starting VM, in host command prompt as administrator:

* Secure boot off
* Ipconfig -> then kdnet in VM
* Copy command in host prompt
  + cd C:\Program Files (x86)\Windows Kits\10\Debuggers\x64
  + windbg -k net:port=50005,key=3q98fqaazvxgb.9af14h7lvxls.2cxecb7hhc3y1.1id4mb4hwgr6w
* Check advanced session
* Restart VM: shutdown -r -t 0

secure boot still off (it says to turn it on).

“””””””””””””””””””””””””””””””””””””””

After restarting the VM: if it says “debugger not connected”, restart VM twice to see “Debugger is running”.

Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente

OS booted with a kernel debugger attached (?).

* Test signing mode
* Driver test signed
* Test certificate installed

1. Installing eBPF for windows -> DONE

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/InstallEbpf.md>

Method 1 is simplest.

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

1. Tutorial basic

<https://github.com/microsoft/ebpf-for-windows/blob/main/docs/tutorial.md>

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

When creating a new file:

Immagine che contiene testo, schermata, bianco e nero, documento

Descrizione generata automaticamente

But everything works as in the tutorial.

TODO: check maps -> solved with explicit cast

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

TUTORIAL LIKE libbpf-bootstrap

<https://blog.subcom.tech/ebpf-programming-on-windows/>

Clone repo at: <https://github.com/SubconsciousCompute/windows-ebpf-starter>

* Compile and load eBPF programs:

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

* Start tracing (printk or all prints)

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

TRACING: <https://github.com/microsoft/ebpf-for-windows/blob/main/docs/GettingStarted.md#using-tracing>

REMEMBER TO ADD THE PROGRAM AND DELETE IT AT THE END OF THE TRACING

* real time in prompt
  + tracelog -start MyTrace -guid "%ProgramFiles%\ebpf-for-windows\ebpf-printk.guid" -rt
  + tracefmt -rt MyTrace -displayonly -jsonMeta 0
  + tracelog -stop MyTrace
* in kernel debugger: also have a LogFile (.Etl)
  + tracelog -start MyTrace -guid "%ProgramFiles%\ebpf-for-windows\ebpf-printk.guid" -kd
  + tracelog -stop MyTrace
  + netsh trace convert LogFile.Etl Output.csv csv

file generated in C:\Program Files (x86)\Windows Kits\10\bin\10.0.22621.0\x64

Often when tracing eBPF programs, it is useful to only view output generated by the bpf\_printk helper. To do so, use ebpf-printk.guid instead of ebpf-all.guid when creating a trace session (first tracelog command).

Output on kernel debugger:

Immagine che contiene testo, bianco e nero, modello, punto

Descrizione generata automaticamente

Also an output on csv file.

* For all the kernelsrc programs do the same procedure and get the output in a csv file

Other commands (but conversion done using kernel debugger).

* wpr.exe -start "%ProgramFiles%\ebpf-for-windows\ebpfforwindows.wprp" -filemode
* netsh trace convert ebpfforwindows.etl overwrite=yes
* netsh trace convert ebpfforwindows.etl ebpfforwindows.csv csv
* wpr.exe -stop ebpfforwindows.etl
* User space:

Add nmake to environment variables:

Open a Developer Command Prompt for VS 2022 -> bat files run only there

* Do the tutorial for user space programs
* DOES NOT WORK: LOOK FOR THE CORRECT INCLUDES AND DIRECTORIES OF THE .h FILES
* Develop program from Linux examples

FOR THE KERNEL PROGRAMS

* Look for the helpers name and for the include
* DON’T KNOW: SEC STRINGS
* Lin\_simple: do not define global variables -> number
* Bpf\_printk -> bpf\_trace\_printk -> call 2-3-4-5 with uint64\_t parameters (1 to 3)

Only numeric arguments

Does not print strings or other arguments

* Include of files a bit strange -> need to explicitly declare all the files you need to include

See lin\_ring\_buf\_use

* ERROR on return from helpers

See examples that use array or hash map

* More strict on verification process -> \*variables only if variable != NULL

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

* Function calls all inline

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

FOR THE USER PROGRAMS

* All the same as in the tutorial

BPF2C

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

AFTER TRACING EVERY TIME

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente