CodeMachine Training

System Setup Instructions

Please read carefully and email **info@codemachine.com** with any questions regarding system setup. CodeMachine courses are very intensive and run on a very tight schedule. There will be absolutely no time during the class to deal with system or software configuration issues.

You must bring your own personal laptop that is virtualization capable and must have the necessary software installed. The hands-on lab exercises will not damage or negatively impact your laptop. All the hands-on labs exercises will be performed inside the virtual machine (VM) guest OS.

Host System

1. **Host System (your laptop)**
   Host system must be powerful enough (in terms of CPU/Disk/Memory) to run at least one virtual machine guest OS. CPU must support hardware virtualization for performance reasons. System must have a working USB 2.0/3.0 port for the attendees to copy files from USB drives.

2. **Host OS**
   The preferred host OS for this course is Windows 10 Version 1703 (RS2) X64 Enterprise with the latest updates applied. However, any version of Windows including or later than Windows 7 SP1 X64 Professional or Enterprise with all the latest updates will work. OS installation image can be obtained from MSDN or TechNet. Windows need not be activated. Trial version of Windows will work just fine.

3. **Internet Access**
   Host systems must have internet access throughout the duration of the course to look up MSDN and for the debugger to download symbols from Microsoft's public symbol server.

4. **Enterprise WDK**
   Download the free Enterprise Windows Driver Kit 1703 (for Windows 10 RS2) [EnterpriseWDK_rs2_release_15063_20170317-1834.zip] (~2GB size) from the following URL: https://msdn.microsoft.com/en-us/windows/hardware/drivers/develop/installing-the-enterprise-wdk
   Unzip the contents of the file in to c:\EWDK. This package contains all the tools including the kernel debugger that are required to build and debug kernel mode code on Windows.

5. **Debugging Symbols**

Run the following command in an administrative command prompt (cmd.exe) to setup the environment variable for the debugger to download symbols from Microsoft's public symbol server.

```
setx _NT_SYMBOL_PATH SRV*c:\symbols*https://msdl.microsoft.com/download/symbols
```

6. **Debugger Shortcut for VM guest OS kernel debugging**

   Create a debugger shortcut on your host system desktop to debug the guest OS kernel. The target of the shortcut should be as follows:

```
"C:\EWDK\Program Files\Windows Kits\10\Debuggers\x64\windbg.exe" -Q -QSY -k
"com:port=\\.\pipe\WINVM,baud=115200,pipe,resets=0,reconnect"
```

7. **Source Code Editor**

   The hands-on labs involve extensive programming in C/C++, so a good text editor must be installed on your host system. You can use your favorite source code editor. Good open source options are Notepad++ available at https://notepad-plus-plus.org and Visual Studio Code available at https://code.visualstudio.com.

8. **Virtualization Software**

   Any virtualization software that is capable of running Window as a guest OS will work i.e. Microsoft Hyper-V, Oracle Virtual Box or VMWare. We recommend using Hyper-V because it is built in Windows 10 Enterprise Edition.

9. **Virtual Machine Configuration**

   Virtual Machine configuration options vary depending on the virtualization software being used. Please note the following if you choose to use serial debugging over named pipe:

   - Virtual Serial Port configuration is extremely important for the hands-on lab exercises. Please make sure you test your kernel debugging setup BEFORE the class.
   - If you are using VMWare, make sure that your guest VM COM ports are configured properly for kernel debugging. You may have to edit your .**vmx** file directly.
   - If you are using Hyper-V and creating a Generation 2 VM, please note that guest VM COM port must be configured manually using the power shell script

     ```
     Set-VMCOMPort -VMName <VirtualMachineName> 1 \\.\pipe\WINVM
     ```

     where <VirtualMachineName> is the name of the guest VM.

   Alternative you can use Virtual KD for kernel debugging if you are running VMWare or Virtual Box.

   Virtual Machine Configuration Options:

Option	Configuration
Virtual Hard Drive	20GB (dynamically expanding)
RAM	1 GB
Virtual CPUs	1 (some labs may not work with multiple CPUs)
Virtual Network	Bridged (VM obtains IP address on the same subnet as host)
Virtual Serial Port	COM1 emulated over host named pipe \\.\pipe\WINVM

Shared Folders (Virtual Box only)	Create a directory c:\Course on the host system. Ensure that this directory on the host is shared with the guest VM either using the virtualization software's shared folder or over SMB. SMB sharing assumes that host and guest systems are on the same subnet.

VM Guest System

10. Windows on VM Guest System

Install Windows 10 Version 1703 (RS2) X64 Professional or Enterprise Edition in the guest VM. OS installation image (.ISO) can be obtained from MSDN or TechNet. Windows need not be activated. Trial version of Windows will also work just fine. The Guest OS system must be in a predictable state for the hands-on lab exercises, so ensure that **no updates are applied** after installing the guest OS. You can install the **virtual machine integration software** after installing the guest OS.

11. Disable Windows Update

Please make sure that Windows Update is disabled in the guest OS. You can turn off Windows update by running the following command in an administrative command prompt (CMD.exe). Also note that the guest OS does NOT need internet access.

```
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU" /v NoAuto Update
/t REG_DWORD /d 1 /f
```

12. Endpoint Protection Software

Anti-Virus, Anti-Malware or any form of host endpoint protection software will interfere with the hands-on lab exercises. Please ensure that no Anti-Malware, Anti-Virus or Endpoint Protection Software is installed on the Guest OS. Also, please ensure that Windows Defender "Real-time Protection" is disabled.

13. Kernel Debugger Configuration

The following commands must be run in an administrative command prompt (CMD.exe) to setup the guest OS for kernel debugging over a serial connection on COM1.

```
bcdedit /copy {current} /d "Windows [debugger disabled]"
bcdedit /debug {current} ON
bcdedit /set {current} debugtype SERIAL
bcdedit /set {current} debugport 1
bcdedit /set {current} baudrate 115200
```

14. Memory Dump Configuration

The following commands must be run in an administrative command prompt (CMD.exe) to configure the guest OS for memory dump generation and **enable driver debug output to show up in the debugger**, this step is critical for the labs.

```
wmic recoveros set DebugInfoType = 1
reg add "HKLM\SYSTEM\CurrentControlSet\Control\CrashControl" /v AlwaysKeepMemoryDump
/t REG_DWORD /d 0x1
reg add "HKLM\SYSTEM\CurrentControlSet\Control\CrashControl" /v NMICrashDump /t
REG_DWORD /d 0x1
```



```
reg add "HKLM\SYSTEM\CurrentControlSet\Services\kbdhid\Parameters" /v
CrashOnCtrlScroll /t REG_DWORD /d 0x1
reg add "HKLM\SYSTEM\CurrentControlSet\Services\i8042prt\Parameters" /v
CrashOnCtrlScroll /t REG_DWORD /d 0x1
reg add "HKLM\SYSTEM\CurrentControlSet\Services\hypervkbd\Parameters" /v
CrashOnCtrlScroll /t REG_DWORD /d 0x1
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Debug Print Filter" /v
DEFAULT /t REG_DWORD /d 0xffffffff
```

Pre-Training Reading Material

It is good to brush up your Windows internals knowledge before the class. We have also made a collection of public papers and slides available for review at
https://1drv.ms/f/s!AtvUy24DJoLGb8y3PDfL-6ECbgY

Lab Material

Software required for the course including tools, configuration scripts, reference documents, source code for the labs exercises, solutions to the lab exercises, memory images and memory dumps will be provided by the instructor on the first day of the class.

We look forward to seeing you in class!