

# Forum:[Hack Forums](#)

Thread: [\[Tutorial\] Windows 10 x86 Kernel Exploit Development - part 1](#)

apt1337 - December 18, 2020, 2:42 am

Windows 10 (1703) x86 Kernel Exploit Development Part 1 - setting up the environment

Hello everyone! This tutorial is part of a series I will be posting here, and this particular one is an kernel debugging environment setup which is a prerequisite to following along my future posts. Due to the complexity of the topic, I won't be writing any fancy thread design, all text will be like this throughout the thread to keep the letter limit below maximum, any code or important parts of the text may be highlighted.

I am an Offensive Security / Information Security student with a keen interest in exploit development, I am far from an expert, which is why I'm writing this thread, not only for you guys but also for myself, as I feel I need to reinforce what I learn and I think writing about it is the best way for me to learn.

In part 1, I will be walking you through setting up an environment for debugging. In usermode exploit development, if you mess things up and you crash the app you can just restart the application / debugger, however, Kernel debugging is slightly different. If you mess up here or you crash it, it will result in a blue screen of death and you will have to restart the device all together. Because of this, debugging of the kernel is often done remotely from another device, to avoid losing progress of your work by accidentally crashing the kernel.

Later on we are going to "crash" the kernel, however, not by accident! We will be targeting a vulnerable driver called HackSys Extreme Vulnerable Driver (I will refer to this as HEVD from this point onwards), HEVD is an intentionally vulnerable Windows driver developed by security researchers. HEVD runs in kernel space, and it contains multiple vulnerabilities, the specific vulnerability we will be targeting in this driver (for now) is a "simple" stack buffer overflow.

If you need any further help setting up the environment feel free to pm me or post below this thread, but if you follow carefully, everything should be okay.

Requirements:

1. Windows 10 x86 1703 Virtual Machine - You can download it here <https://www.getmyos.com/windows-10-1703-home-pro-education-32-64-bit-free-download> make sure you get the 32bit version.
2. Python / C
3. IDA Freeware 5.0 - (direct download link) <https://samsclass.info/126/proj/idafree50.exe>
4. HEVD - <https://github.com/hacksystem/HackSysExtremeVulnerableDriver/releases>
5. OSR loader - to "load" the driver <https://www.osronline.com/article.cfm%5Earticle=157.htm>
6. WinDbg - for remote kernel debugging  
<https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk/> you can just download the SDK, unselect everything apart from "Debugging tools for windows"
7. Pykd - <https://githomelab.ru/pykd/pykd>
8. Mona & WinDbglib - <https://github.com/corelan/windbglib>
9. A hypervisor - I suggest you use vmware player or vmware pro as I wil be using this throughout

this series.

There will be two virtual machines we will make, one machine will be called "Debugger" and the other will be called "Target".

### Setting up the environment

Install the windows 10 VM listed in requirement 1, name it debugger, and then download requirement 2 (python2 and 3), 3, and 6 yourself then come back here.

Now that you have those requirements, we will focus on the rest in a minute, first we will setup an environment variable for Windows symbols:

1. In the Windows start menu search bar type "Advanced system settings" --> Click Environment Variables --> New System variable.

Enter the following:

`_NT_SYMBOL_PATH`

`srv*https://msdl.microsoft.com/download/symbols`

Press ok and ok again, you can exit that now.

Now download pykd.zip from here <https://github.com/corelan/windbglib/blob/master/pykd/pykd.zip> this contains two files - vscredist\_x86.exe and pykd.pyd, you can leave vcaredist.exe where it is for now, but pykd.pyd should be placed in the following directory:

`C:\Program Files\Windows Kits\10\Debuggers\x86\winext\pykd.pyd <--`

You can now run vcaredist\_x86.exe and follow the installer it should only take a few seconds.

Download mona.py <https://github.com/corelan/mona>

Download windbglib.py <https://github.com/corelan/windbglib>

Once you've downloaded these two, move both mona.py and windbglib.py to the following directory:

`C:\Program Files\Windows Kits\10\Debuggers\x86`

You may come accross an issue with these 3 files when attempting to run them, because they're "downloaded from the internet" they may have "protected mode" on, to be sure this isn't the case, for pykd.pyd, mona.py, and windbglib.py, right click each one and select "properties" and mark it as safe manually, you will see it if it's enabled don't worry.

To verify that all of this has worked correctly, type WinDbg in the start menu search bar and open it, in the top left of WinDbg go to:

File --> Open executable --> go to system32 and select calc.exe

Now calc should be loaded, in the command window type:

```
.load pykd.pyd
```

```
!py mona modules
```

Should look something like this - I may have a different font for screenshots purposes.

Great!! Now let's set up the second VM, Target!

Setting up the vulnerable target

Create a new virtual machine with the same version, name it "target" - Windows 10 1703

Once target has installed and booted up download and unzip the following:

HEVD - <https://github.com/hacksystem/HackSysExtremeVulnerableDriver/releases>

OSR Loader - <https://www.osronline.com/article.cfm%5Earticle=157.htm>

Once you have downloaded these two, drag them to the desktop or a directory you think is suitable.

We need to start OSR loader, go to WNET/i386/FRE/ and start OSRLOADER as an Administrator.

Click browse within OSRLOADER, select & load HEVD.sys from VULNERABLE/x86/HEVD.sys.

Now, click "register" --> and "start service".

You should get a "The operation completed successfully" message box.

Now we're going to make sure that HEVD is loaded at startup everytime.

Open cmd as Administrator and type the following:

```
sc config HEVD start=system
```

If that was successful, you should see this:

Go back to the DEBUGGER machine, open cmd and type "ipconfig".

Copy the IPv4 address, now go back to the TARGET machine, and open a NEW CMD prompt as Administrator and enter the following:

```
bcdedit.exe /dbgsettings NET HOSTIP:192.168.x.x PORT:50000
```

Replace HOSTIP with the DEBUGGER VM IP that you just copied.

The output of this command should be a key, which we will use in windbg on the DEBUGGER machine to listen and receive a connection from TARGET.

Now copy this key, and go back to your debugger machine, close WinDbg if it's open still.

Open WinDbg --> in the top left, click file --> click Kernel Debugging -> Enter key and set port to 50000 --> click ok.

Okay, now WinDbg should be listening for a connection :) now we just need to make the TARGET windows start in debug mode, type the following:

```
bcdedit /debug ON
```

```
shutdown -r -t 0
```

When it boots up you should see a connection in WinDbg! Pause execution by selecting the "break" option, Debug --> break.

Should look something like this after.

Verifying HEVD

Now we need to verify that the HEVD "module" has been loaded. Type in the debugger:

!m m H\* - This takes the letter "H" and queries all modules that begin with it.

Now we just need to quickly Download HEVD on our DEBUGGER machine as we need to load the symbols.

<https://github.com/hacksystem/HackSysExtremeVulnerableDriver/releases>

Once downloaded, enter the following commands on the DEBUGGER machine:

```
mkdir C:\projects\hevd\build\driver\vulnerable\x86\HEVD\
```

```
cd C:\projects\hevd\build\driver\vulnerable\x86\HEVD\
```

Now you need to copy HEVD.pdb to this directory, you will need to replace the user in the path with whatever your user is called:

```
xcopy C:\Users\size_t\Desktop\driver\vulnerable\x86\HEVD.pdb .
```

Use the "dir" command to check HEVD.pdb was copied.

Now go back to WinDbg, and let's check that the HEVD symbols have been mapped:

```
.reload  
x /D HEVD!a*
```

You should be able to see HEVD functions now, if you can't then you need to make sure HEVD.pdb file is in the correct directory on DEBUGGER.

If you followed this and everything has gone well, you have now setup the debugging environment and are ready to start debugging the TARGETS kernel space! :)

Due to the character limit on threads, you will need to wait for part 2 to start doing hacky stuff.

In part 2 we will analyze the driver in IDA, and maybe walk through the source code, we'll use the information gathered during the analysis to start building an exploit!

DI?E - December 18, 2020, 2:48 am

Really nice write up. Will be awaiting the rest of it. ?

apt1337 - December 18, 2020, 3:14 am

(Dec 18, 2020 03:48 AM)DI?E Wrote: [\[CITE\]](#) Really nice write up. Will be awaiting the rest of it. ?  
Thankyou :)

If anyone has any problems setting up the debugging environment, feel free to PM me with a detailed explanation of what it's doing + screenshots.

DI?E - December 18, 2020, 3:28 am

(Dec 18, 2020 04:14 AM)size\_t Wrote: [\[CITE\]](#) Thankyou :)

If anyone has any problems setting up the debugging environment, feel free to PM me with a detailed explanation of what it's doing + screenshots.

No problem. This section needs to be more active in my opinion.

Soon in this section I'm going to write up some AMSI/AV bypass and evasion methods. With holidays coming up I don't have a date set in mind but soon lol. Hopefully we can get more people to hang around in this section of the forum.

apt1337 - December 18, 2020, 3:30 am

(Dec 18, 2020 04:28 AM)DI?E Wrote: [\[CITE\]](#) Soon in this section I'm going to write up some AMSI/AV bypass methods. With holidays coming up I don't have a date set in mind but soon lol. Hopefully we can get more people to hang around in this section of the forum.

That's awesome you should totally do that! Yea that is my motive as well I want this section to be really high quality, and also (selfishly) for myself to reinforce what I've learned ? - I think if I go to far into this series though, it will get to a point where no one understands anything at all, so I might do an alternative basics series

DI?E - December 18, 2020, 3:35 am

(Dec 18, 2020 04:30 AM)size\_t Wrote: [\[CITE\]](#) That's awesome you should totally do that! Yea that is my motive as well I want this section to be really high quality, and also (selfishly) for myself to reinforce what I've learned ? - I think if I go to far into this series though, it will get to a point where no one understands anything at all, so I might do an alternative basics series

I agree. Too many people buying tools etc and not even knowing what it does or how it does it. Gets pretty pointless when all the information is everywhere on the web to find yourself for free. Trial

and error. ?

**Xphile** - December 18, 2020, 9:40 pm

Really good thread dude, love the motivation behind it as well:

(Dec 18, 2020 03:42 AM)size\_t Wrote: [\[CITE\]](#) I am an Offensive Security / Information Security student with a keen interest in exploit development, I am far from an expert, which is why I'm writing this thread, not only for you guys but also for myself, as I feel I need to reinforce what I learn and I think writing about it is the best way for me to learn.

I might dip my toe into exploit dev after this series. Looking forward to the next one. :)

**Don\_Julio** - December 19, 2020, 10:20 pm

Great tutorial. I know what I'll bet getting myself into over the holidays.

Cheers.

**The Grand Oracle** - December 20, 2020, 9:10 pm

Do you have any idea how to download a specific .PDB from the symbol server, for example ntoskrnl.pdb, so I can load symbols during disassembly with IDA?

**apt1337** - December 28, 2020, 11:30 pm

(Dec 20, 2020 10:10 PM)The Grand Oracle Wrote: [\[CITE\]](#) Do you have any idea how to download a specific .PDB from the symbol server, for example ntoskrnl.pdb, so I can load symbols during disassembly with IDA?

I might be misunderstanding your question but IDA does that automatically when you analyze it. However, If you know the exact path of the pdb (the md5 hash) you can just wget it.

**compooter** - January 1, 2021, 4:53 pm

great thread! looking forward to part two :D

**2020Sucked** - January 1, 2021, 6:24 pm

Thanks for the write up. This must've taken a lot of time! I will be trying this soon.

**apt1337** - January 9, 2021, 2:47 am

(Jan 1, 2021 07:24 PM)2020Sucked Wrote: [\[CITE\]](#) Thanks for the write up. This must've taken a lot of time! I will be trying this soon.

It did take quite abit of time to put together but it's actually okay because I did learn a few things even just writing it.

I'm releasing part 2 soon, I've been very busy over Christmas / new year so I've not had a chance, in part 2 I will be going through static analysis on the vulnerable driver in IDA, and if there is room I'll write a (stage1) proof of concept exploit to demonstrate that it is indeed vulnerable. After that, possibly in part 3, we will write a more complex exploit with some token stealing shellcode or something like that, to escalate privileges to the highest level. The kernel uses alot more complex mitigations than userspace, so I may have to include an entire thread on just theory around bypassing (some) of the kernel exploit mitigations, and that will take a while because as I said I'm not an expert security researcher or anything.