

Windows Debugging 101

Author: Ialle Teixeira {blog} <https://www.linkedin.com/in/isdebuggerpresent/>



Basicamente o processo de encontrar e reduzir defeitos numa aplicação de software ou hardware é denominada debugging. Mas em nosso caso (Exploitation, Malware Research, Kernel Debugging e afins), essa prática é um caminho para descobrir informações mais sensíveis sobre uma aplicação que estamos desejando encontrar alguma falha ou resolver algum bug.

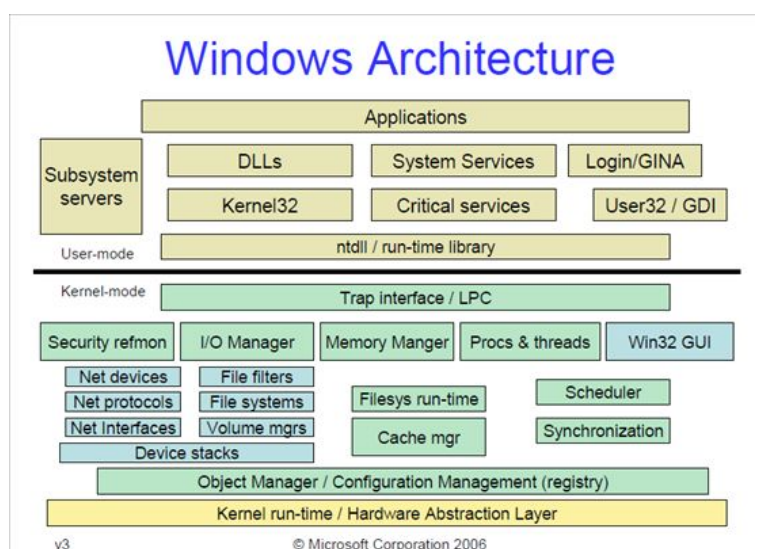
Ao decorrer desse paper iremos conhecer algumas ferramentas e entender o ambiente e situação correta para utilizar da uma.

Para começarmos, vamos deixar bem claro que debugging não se trata de Pentesting, Exploitation ou Code review, entenda isso como um mecanismo que dará a você as informações necessárias para execução de um pentesting, exploração ou code review. De modo tradicional, você executa o seu software no debugger, tenta verificar como ele se comporta com o sistema operacional, reúne todas as informações necessárias e as usa de acordo com sua necessidade. Isso é basicamente reconhecimento(recon). No Windows existem basicamente dois tipos de debuggers:

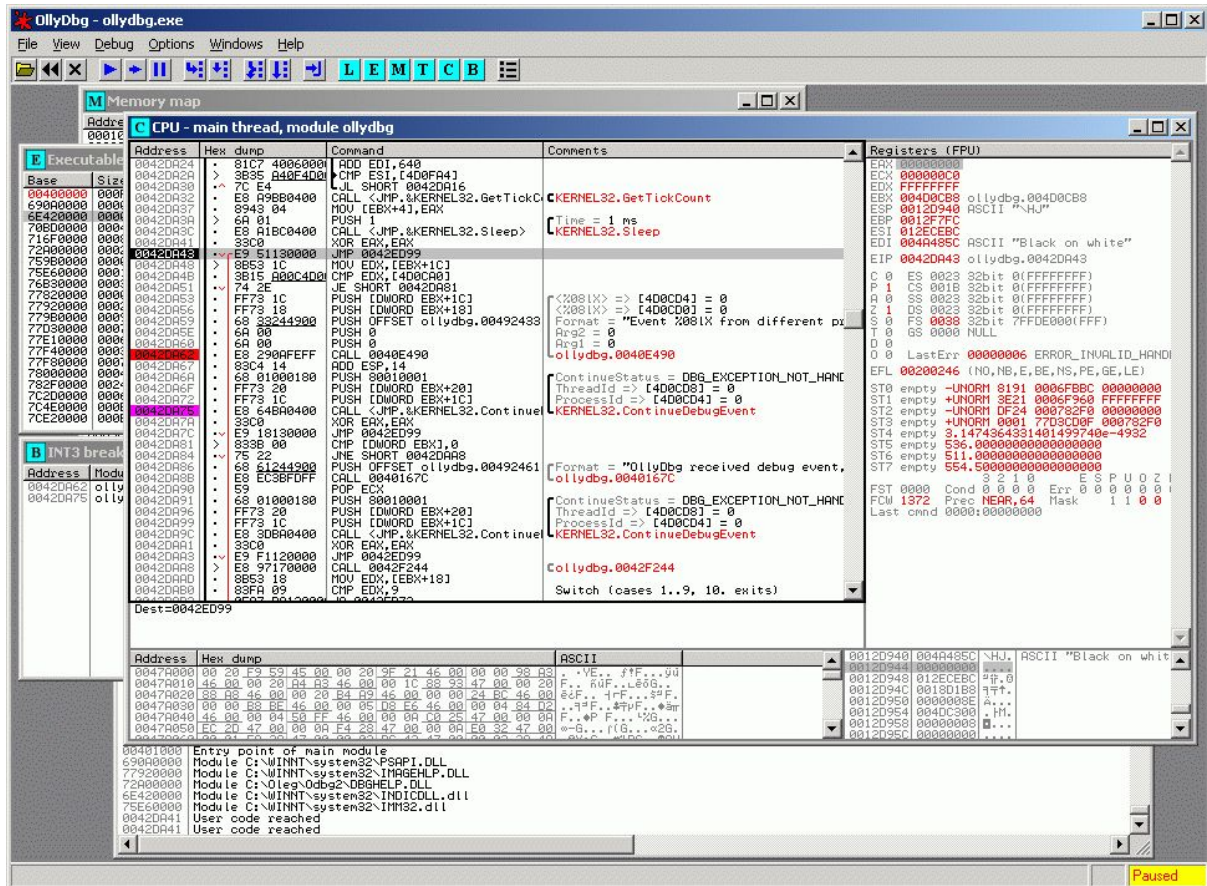
User-Mode Debugger : Este tipo de ação é capaz de mostrar o status de um processo em um determinado momento (execução de threads, registros da máquina, conteúdo da memória, etc.), ou seja, o debugger tem exatamente o mesmo visão da máquina que teria cada um dos processos em execução.

Kernel-Mode Debugging : Esse tipo de debugging possui uma visão completa da máquina na qual eles estão sendo executados. Eles geralmente são usados para detectar erros em device drivers, kernel exploitation, e afins.

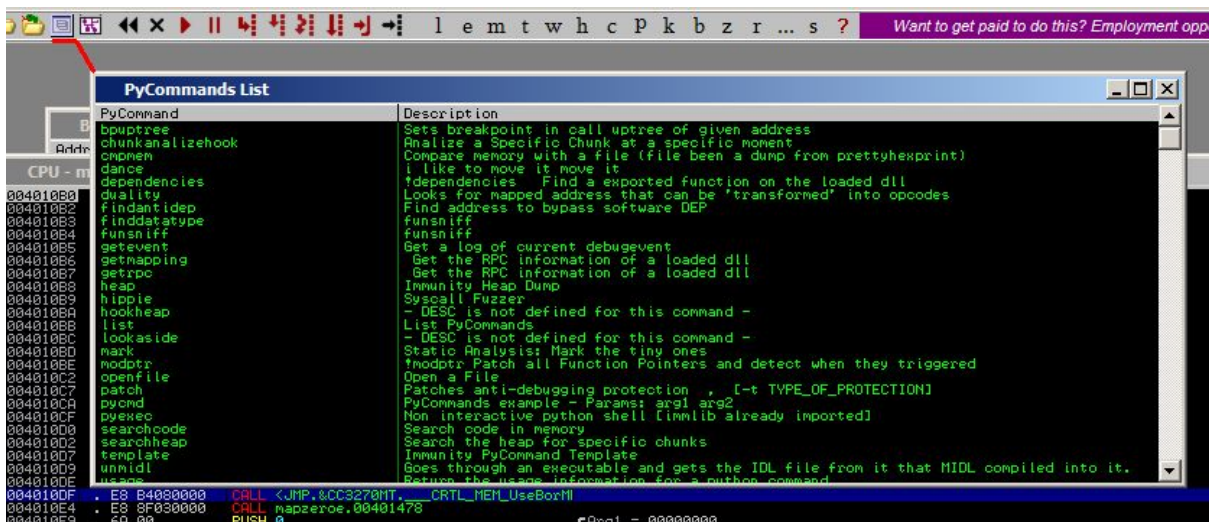
Abaixo, temos uma lista e o descritivo dos debuggers mais usados em Windows env:



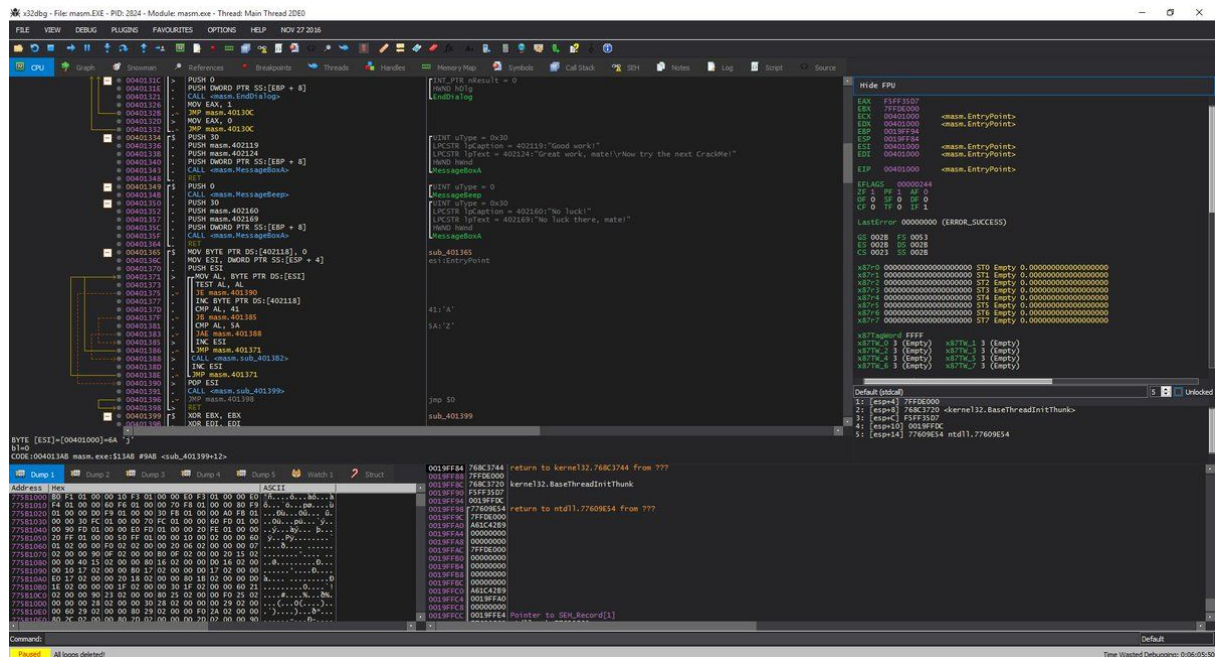
[OllyDBG](#) - Um dos debuggers mais conhecidos. Infelizmente o Ollydbg é antigo e suporta apenas sistemas operacionais x86 e não recebe atualizações relevantes faz um bom tempo



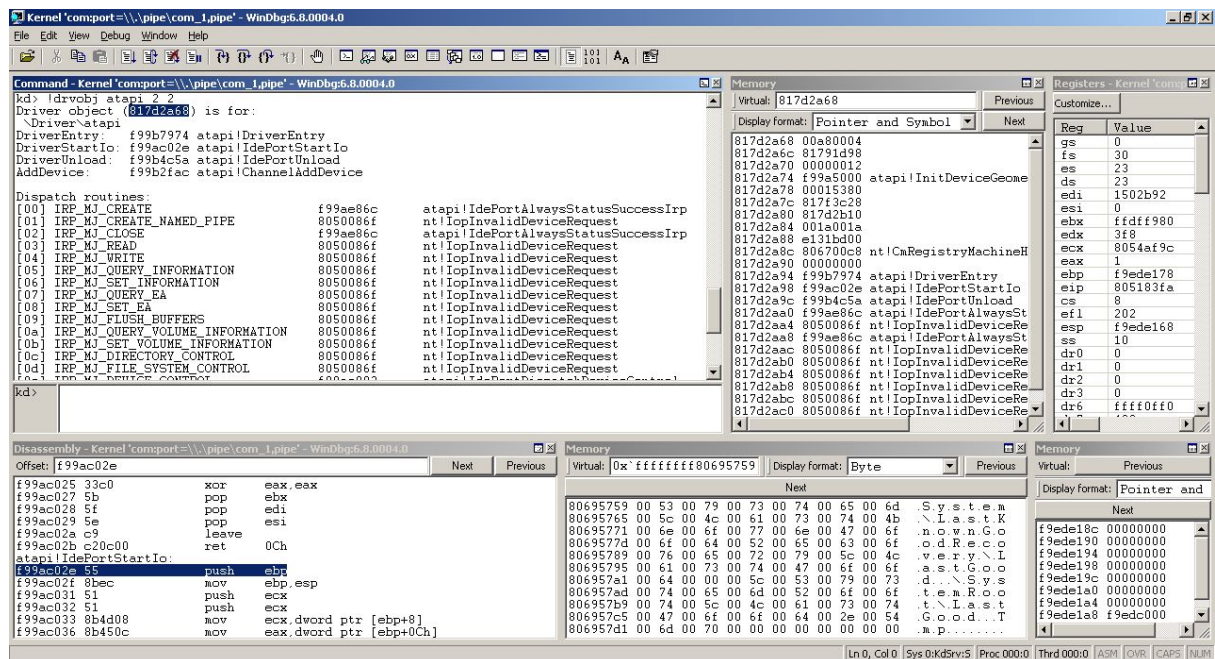
[Immunity Debugger](#) - Immunity Debugge é um dos mais poderosos debuggers e escolha número um entre os desenvolvedores de exploits, usado também para análise de malwares e engenharia reversa de binários. Basicamente foi desenvolvido em uma sólida interface com funções gráficas que vão facilitar sua vida. Obs: suporte apenas para 32bit.



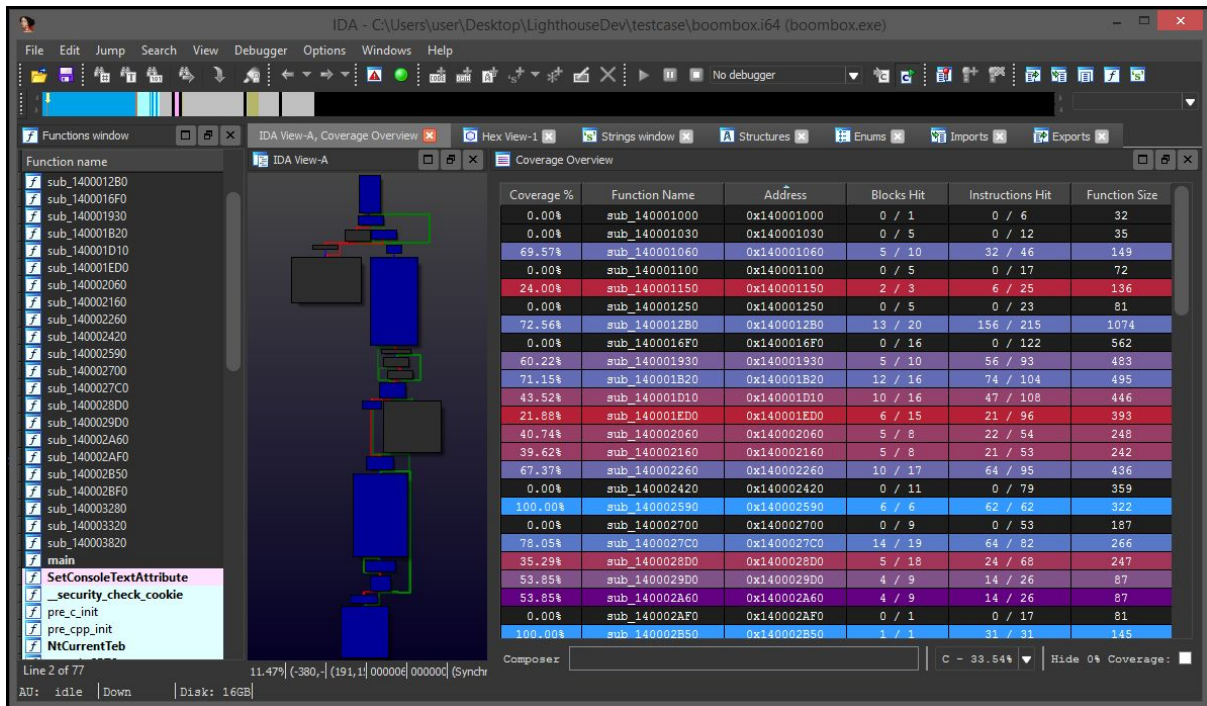
x64dbg - Um puco mais completo que os debuggers anteriores, tem suporte a x64/x32 bit. Este debugger open source é parecido com o Ollydbg/Immunity. Esse te permite realizar debugging no Win10 x64.



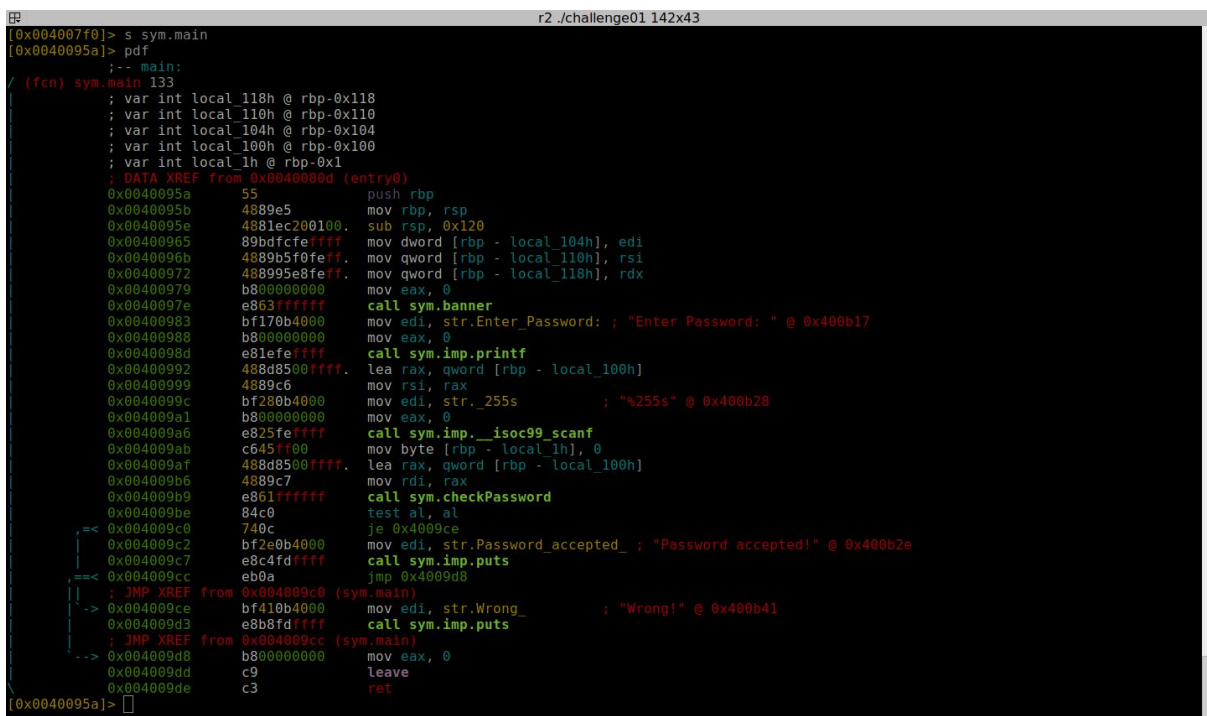
WinDBG - Este é o debugger oficial fornecido pela Microsoft, considerando a forma como o WinDbg é construído, ele não é realmente um debugger. O que ele faz é prover uma interface, bem mais intuitiva, para os debuggers com interface linha de comando. Um vasta documentação nos portais da Microsoft e updates frequentes.



[IDA Pro](#) - Um dos debuggers e desassemblers mais conhecidos e versáteis. O único problema é que, ao contrário de outros debuggers, ele é caro. Tem uma versão freeware, mas suporta apenas x86 e funcionalidade bem limitadas.



[Radare2](#) - Imagine o IDA Pro open source, esse é o r2. No entanto, como muitas pessoas afirmam, a curva de aprendizado é bem maior e mais complexa, e sua opção de local e remote(gdb, rap, webui, r2pipe, winedbg, windbg).



No entanto o debugger mais recomendado para ambiente Windows é o Windbg, ele é nativo e muito simples de usar, e com algumas vantagens sobre os demais:

Desenvolvimento pela Microsoft: O fato de ser oficial da Microsoft, ele suporta quase todas as versões do Windows, você pode debuggar C, C ++, C #, VB, UWP, aplicativos móveis, Azure, XBOX, aplicativos da plataforma Kinect etc. Além disso, ele está disponível para x64, x86, ARM e etc.

Sem nenhum custo financeiro: Ele vem com o Windows Software Development Toolkit, pode ser usado para realizar debugging em kernel mode também. Juntamente com o Windbg, a MS também fornece alguns itens adicionais: KD, NTKD, CDB, NTSD, a variedade de extensões é gigantesca.

A Microsoft tem um livro chamado "[Windows Internals](#)" que é lançado a cada nova versão do Windows, o que facilita seu entendimento sobre o funcionamento interno do mesmo. O Windbg (junto com o Sysinternals) é a ferramenta principal para executar os exercícios mencionados no livro ou pode você optar por ler a documentação que a MS disponibiliza gratuitamente.

Filter by title

Home

Learn

Mark's Webcasts

Windows Internals Book

Troubleshooting with the Windows Sysinternals Tools

Inside Native Applications

Downloads

> File and Disk Utilities

> Networking Utilities

> Process Utilities

> Security Utilities

> System Information

> Miscellaneous

Sysinternals Suite

Community

Software License Terms

Licensing FAQ

Download PDF

Windows Internals Book

02/06/2017 • 3 minutes to read • Contributors

Windows Internals 7th edition (Part 1) covers the architecture and core internals of Windows 10 and Windows Server 2016. This book helps you:

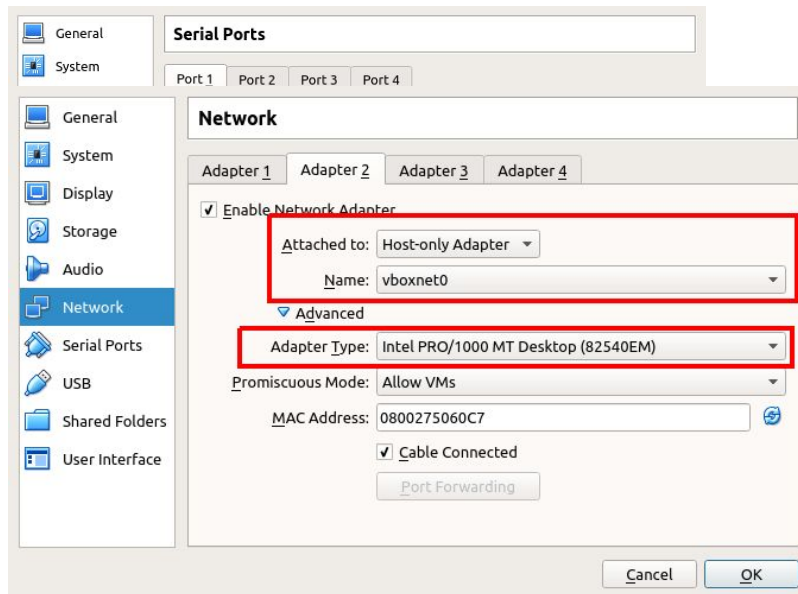
- Understand the Windows system architecture and its general components
- Explore internal data structures using tools like the kernel debugger
- Understand how Windows uses processes for management and isolation
- Understand and view thread scheduling and how CPU resources are managed
- Dig into the Windows security model including recent advances in security mitigations
- Understand how Windows manages memory
- Understand how the I/O system works

The 7th edition was written by Pavel Yosifovich, Mark Russinovich, David Solomon, and Alex Ionescu. New material has been added since the 6th edition (written by Mark Russinovich, David Solomon, and Alex Ionescu) (written by Mark Russinovich, David Solomon, and Alex Ionescu) topics from the first part of the 7th edition include: process and job mechanisms, networking, file systems, and more.

Table of contents of the 7th edition:

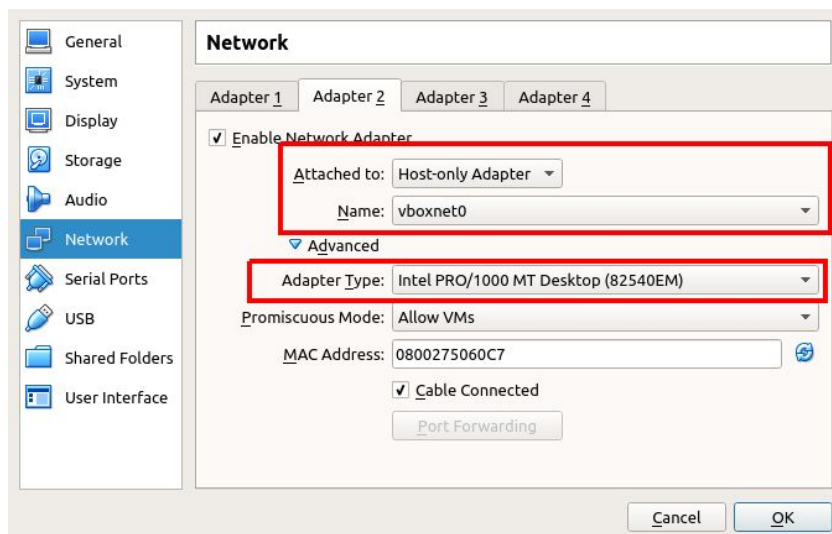
- Chapter 1: Concepts and Tools
- Chapter 2: System Architecture
- Chapter 3: Processes and Jobs

Veja abaixo como realizar o setup do ambiente para iniciar seus estudos, inicialmente você precisará de uma ferramenta de virtualização como Virtualbox, Vmware e afins; no setup abaixo iremos demonstrar com o Virtualbox:



Na aba “Serial Ports” habilite uma porta. Você pode usar qualquer “Port Number” que precisar, não esqueça de guardar o endereço porque ele será necessário em um próximo passo. Selecione “Host Pipe” como “Port Mode” e insira o local path “Path/Address”(por exemplo (/tmp/win7-kd-pipe). Esse pipe será usado para comunicação remota durante o processo de debugging sobre UART. Por

fim, tenha certeza de deixar a opção “Connect to existing pipe/socket” desativada.



Na aba de “Network”, habilite outro adaptador como “Host-Only”. Configure à uma interface de rede existente e cick em “OK”

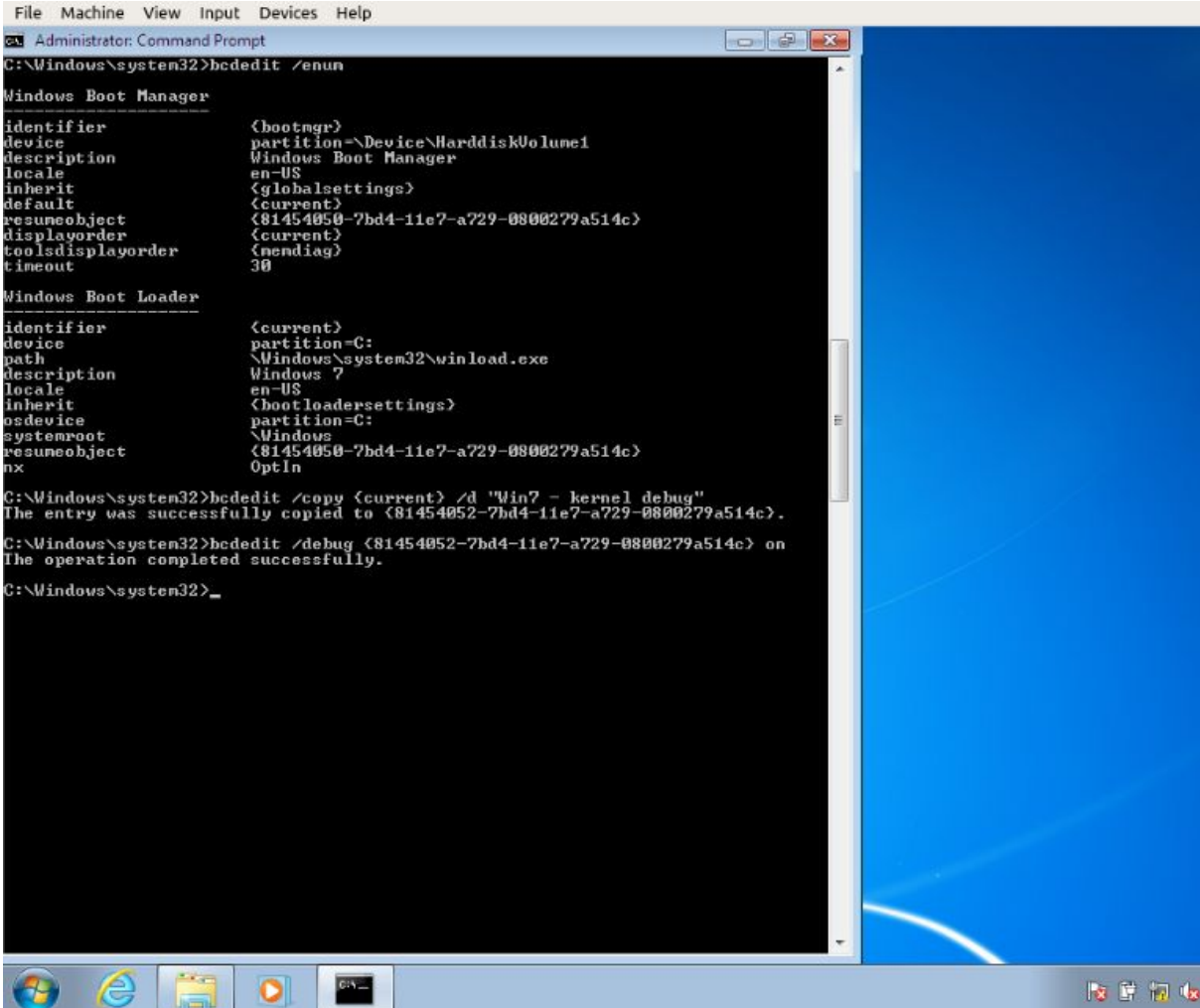
Agora, sua vm está quase pronta para habilitar o processo de remote debugging, você precisa instalar o WindDBG e usar a opção de kernel debugging. Uma forma rápida de instalar o windbg é usando o “chocolatey”:

```
C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
<chocolatey is being installed...>
C:\> choco install -y --force windbg
```

Todas as versões do Windows pode ser debuggados via Seria Port (UART). Esse método é universal, para habilitar no Windows 7, abra um cmd como Administrador e insira uma entrada no bootloader usando "bcdedit":

```
C:\> bcdedit /copy {current} /d
```

```
C:\> bcdedit /debug {UUID-RETORNADO-NO-COMANDO-ANTERIOR} on
```

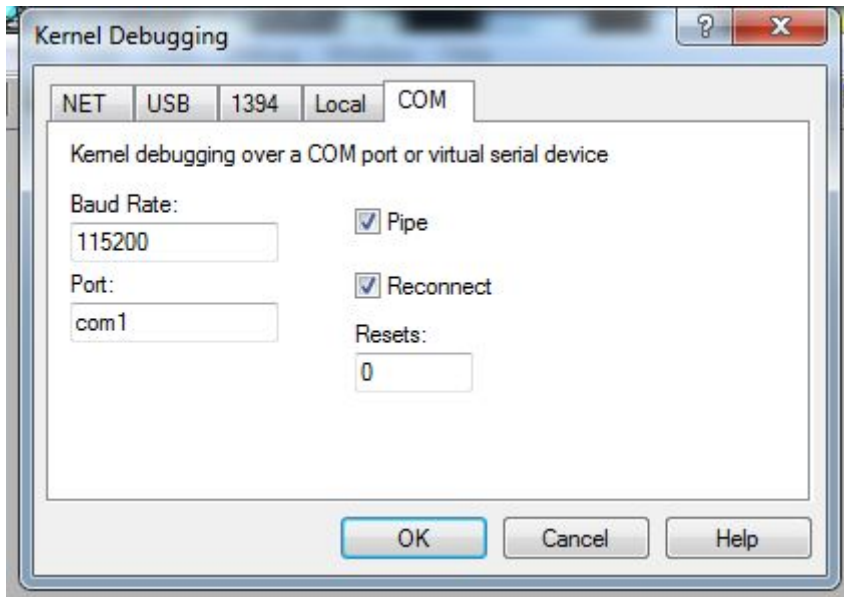


```
File Machine View Input Devices Help
Administrator: Command Prompt
C:\Windows\system32>bcdedit /enun
Windows Boot Manager
-----
identifier          <bootmgr>
device              partition=\Device\Harddisk0\lun1
description         Windows Boot Manager
locale              en-US
inherit              <globalsettings>
default             <current>
resumeobject        <81454050-7bd4-11e7-a729-0800279a514c>
displayorder        <current>
toolsdisplayorder   <mendiag>
timeout             30
-----
Windows Boot Loader
-----
identifier          <current>
device              partition=C:
path                \Windows\system32\winload.exe
description         Windows 7
locale              en-US
inherit              <bootloadersettings>
osdevice            partition=C:
systemroot          \Windows
resumeobject        <81454050-7bd4-11e7-a729-0800279a514c>
nx                  OptIn
C:\Windows\system32>bcdedit /copy <current> /d "Win7 - kernel debug"
The entry was successfully copied to <81454052-7bd4-11e7-a729-0800279a514c>.
C:\Windows\system32>bcdedit /debug <81454052-7bd4-11e7-a729-0800279a514c> on
The operation completed successfully.
C:\Windows\system32>_
```

Agora vamos setar a comunicação serial do Windows como "debugging medium", iremos usar o bcdedit novamente com /dbgsettings global switch, e em seguida reinicie sua vm:

```
C:\> bcdedit /dbgsettings serial debugport:1 baudrate:115200
```

```
C:\> bcdedit /set {UUID-RETORNADO-NO-COMANDO-ANTERIOR} debugtype serial
```

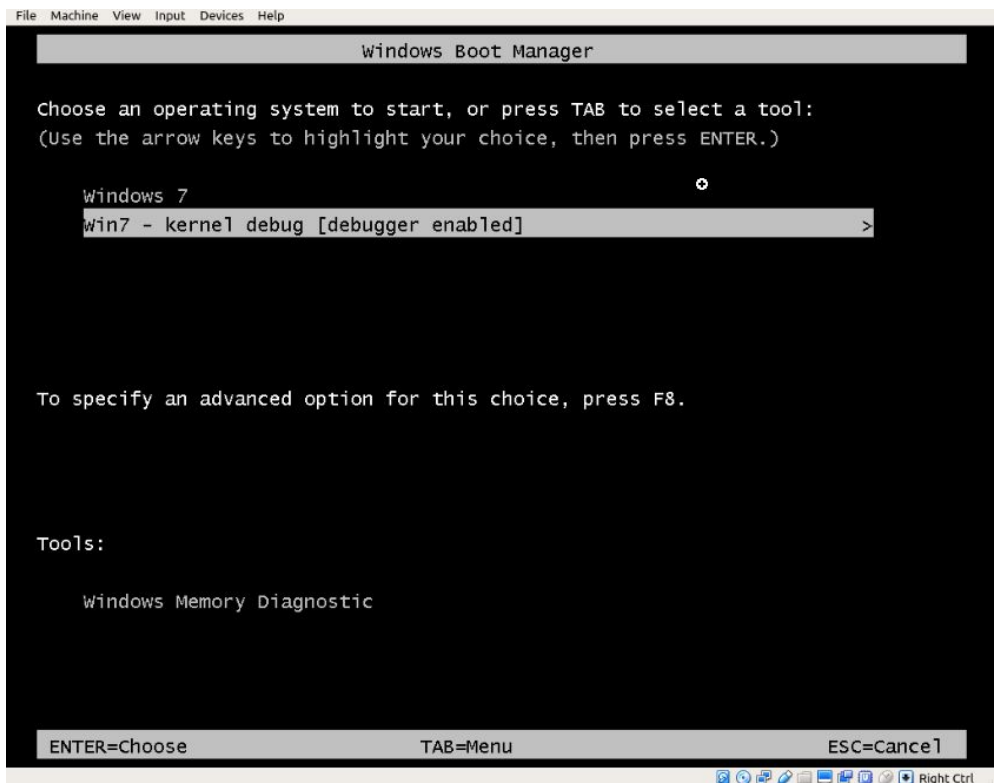
Para iniciarmos uma sessão de debugging no windbg, vá até a aba "COM" e insira as configurações setadas na máquina alvo, por e dê um "OK"

Posteriormente o WinDBG irá aguardar com uma conexão na porta "COM1":

```
Microsoft (R) Windows Debugger Version 6.3.9600.17298 X86
Copyright (c) Microsoft Corporation. All rights reserved.

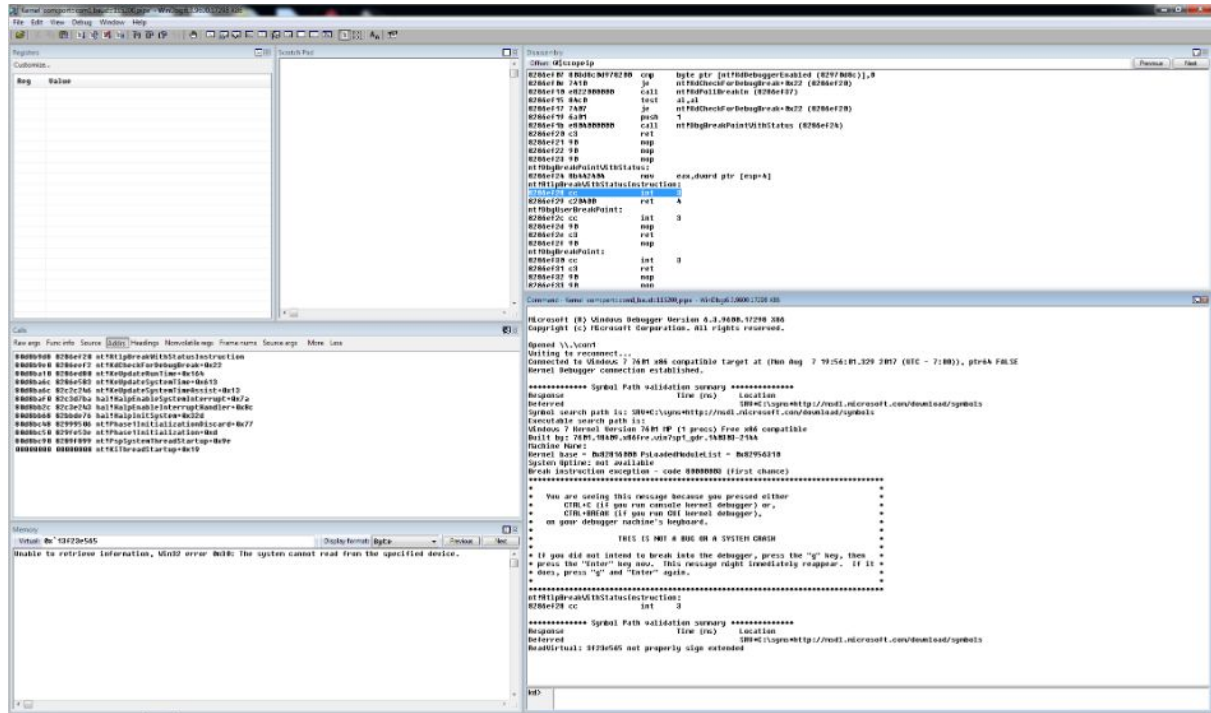
Opened \\.\com1
Waiting to reconnect...

***** Symbol Path validation summary *****
Response           Time (ms)  Location
Deferred           0          srv*c:\syms*http://msdl.microsoft.com/download/symbols
```



Essa opção será exibida quando a máquina alvo for reiniciada, selecione a opção "Kernel Debug [debugger enabled]"

Por fim, o windbg carregará a máquina em debug mode e quando você prescionar “enter”, a vm será inserida com sucesso:



Agora você está debugando o kernel do Windows 7, e você estará pronto para executar seus primeiros passos com kernel debugging, até a próxima ;)

Referências:

¹Symbols for Windows debugging (WinDbg, KD, CDB, NTSD) <https://docs.microsoft.com/pt-br/windows-hardware/drivers/debugger/symbols>

² Tales of a binary encoded life... <https://blahcat.github.io>

³Pattern-Oriented Software Diagnostics <http://www.windbg.org/>