Windows security

Exercise 1 – Module 1 – Section 1

June 2020  
V1.3

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only, and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third-party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

This training uses various tools and utilities downloaded from the Internet for the classroom environment.   
Downloading any tools, installing and using them should only be done at your own risk security checked the tools in a test environment.

© 2020 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are the property of their respective owners.

Contents

[Analysis of a memory dump 1](#_Toc526232823)

[Abstract and learning objectives 1](#_Toc526232824)

[Overview 2](#_Toc526232825)

[Requirements 2](#_Toc526232826)

[Before the exercise 3](#_Toc526232827)

[List of VM to start 3](#_Toc526232828)

[How to start and connect to a VM 4](#_Toc526232829)

[Exercise 1: Examining system’s current activity 6](#_Toc526232830)

[Task 1: Prepare the debugging environment 6](#_Toc526232831)

[Task 2: List currently running processes 8](#_Toc526232832)

[Task 3: Get details about a running process – part 1 9](#_Toc526232833)

[Task 4: Get details about a running process – part 2 10](#_Toc526232834)

[Task 5: Explore a process activity 11](#_Toc526232835)

[Exercise 2: Analyzing memory usage 12](#_Toc526232836)

[Task 1: Display general memory usage 12](#_Toc526232837)

[Task 2: Get insights about pools consumption 12](#_Toc526232838)

[Task 3: Examine the memory layout of an application 13](#_Toc526232839)

[Task 4: Examine Heap structures 13](#_Toc526232840)

[Exercise 3: Devices and drivers 14](#_Toc526232841)

[Task 1: List driver modules currently loaded by the system 14](#_Toc526232842)

[Task 2: Navigate the object manager 15](#_Toc526232843)

[Task 3: I/O Request Packets 15](#_Toc526232844)

[After the Lab 17](#_Toc526232845)

[Task 1: Stop and deallocated all the VMs 17](#_Toc526232846)

# Analysis of a memory dump

## Abstract and learning objectives

In this lab, you are going to demonstrate your knowledge about the Windows operating system by analyzing the content of the physical memory of a running system.

Under some conditions, Windows operating system permits to copy the whole content of the physical memory to a file for analysis purposes. This file is what is generally called a dump. Such files are used in a wide range of scenarios from system component development (eg. Drivers development) to computer security forensics. The dump gives a frozen picture of what the operating system was doing.

## Overview

In this Lab, attendees will perform basic interpretation of memory dump to inspect the state of a working Windows machine.

## Requirements

1. Attendee’s machine:
   1. Ideal resolution 1920 x 1080
   2. An Internet browser
   3. An RDP client
   4. Internet access without restriction on outbound connections.   
      The following outbound TCP port must be accessible :

* **TCP/80 and TCP/443** to reach Azure Portal
* **TCP/3389** to establish RDP remote connection to virtual machines exposed directly to Internet

or

* **TCP/(49152 to 65535)** to establish RDP remote connection to virtual machines exposed by a Load Balancer

## Before the exercise

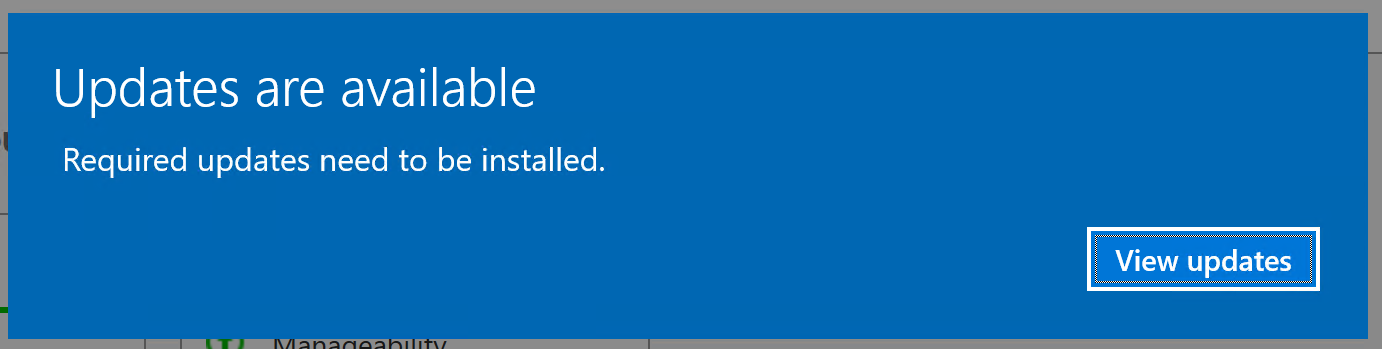
Duration: 10 minutes

Synopsis: In this section, you will set up your environment for use in the rest of the Lab. You should have the following environment.

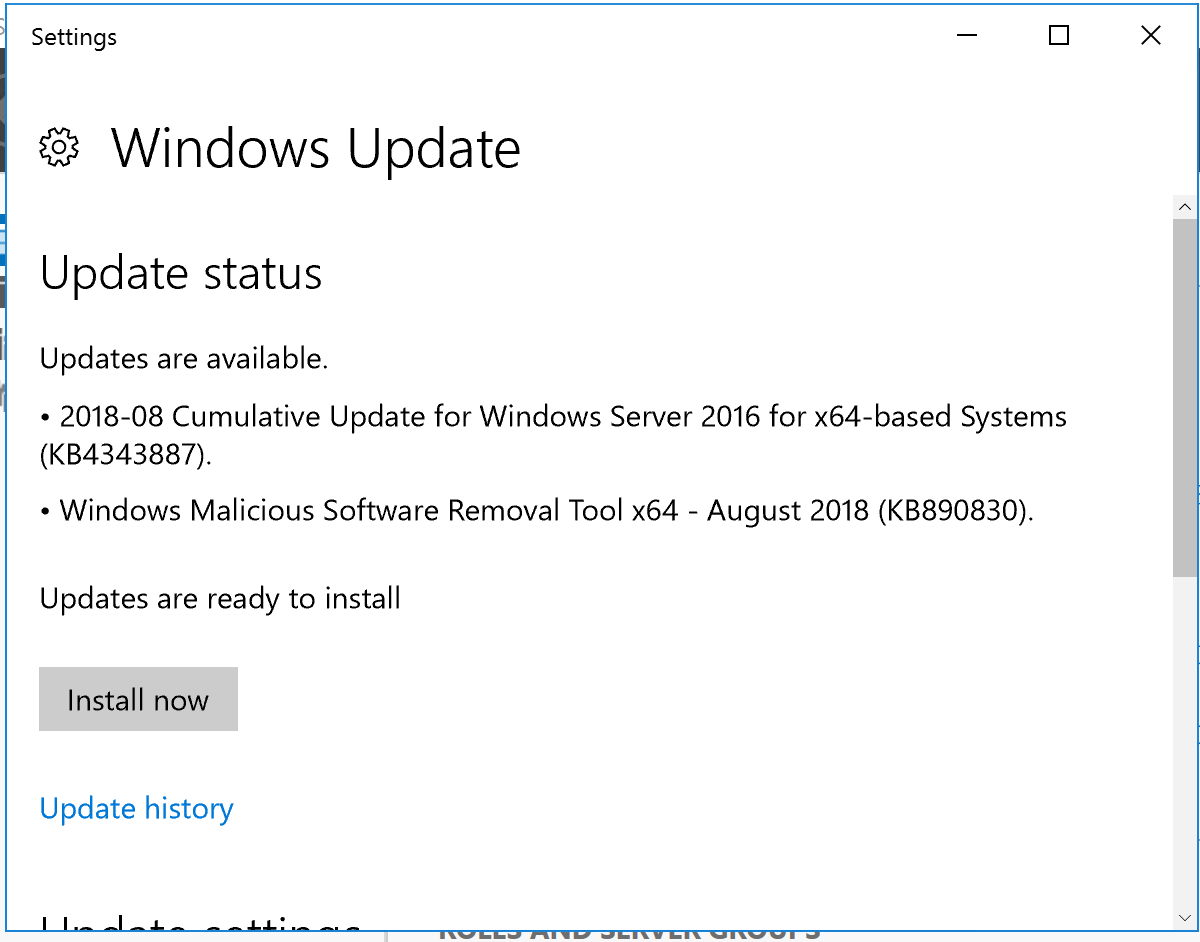
#### List of VM to start

|  |  |  |  |
| --- | --- | --- | --- |
| Name of VM | Hostname | OS Type | Role |
| CSW1-LAB | CSW1-LAB | Windows 10 Enterprise | Workstation |

Note that the machines have been provisioned in March 2020.   
Therefore, it is possible to see the following message while connecting for the first time to the servers:



In this case, click on View updates.



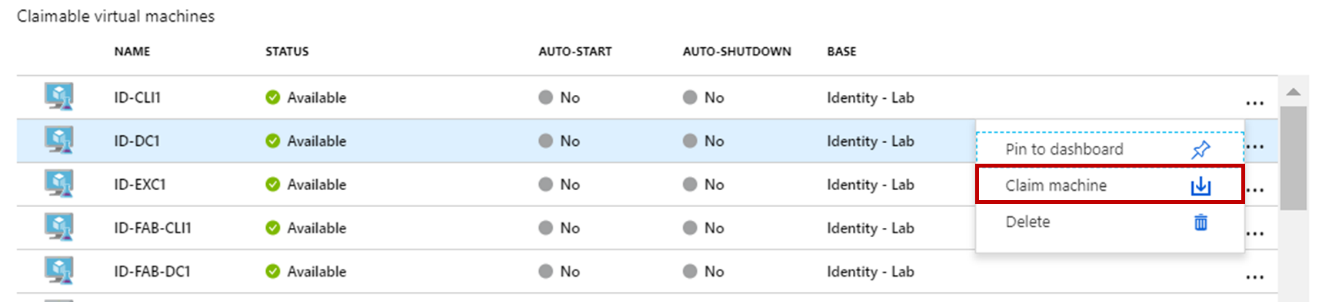
We do not need the latest updates for these labs so you can close this window.

#### How to start and connect to a VM

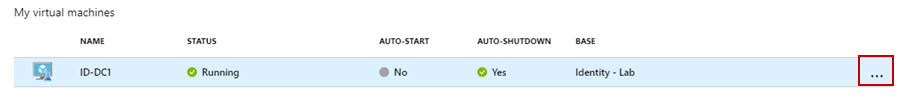
1. Go to Azure portal : <https://portal.azure.com>
2. Sign-in with your student or organizational account
3. Click on the Dev&Test Lab (Select the right subscription if the resource is not displayed)



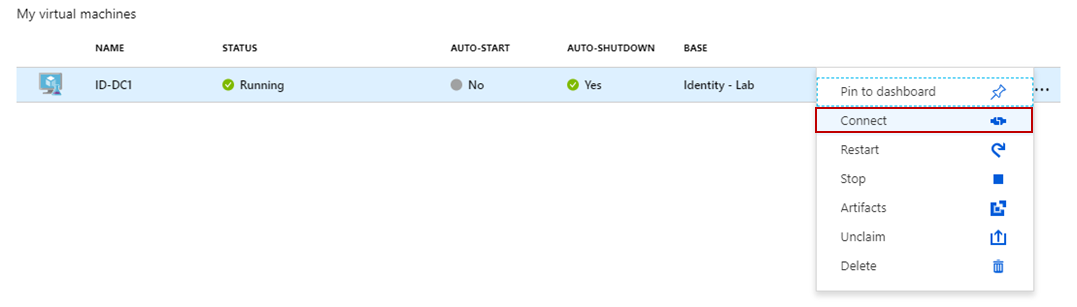
1. To start a VM, click on “Claim machine”



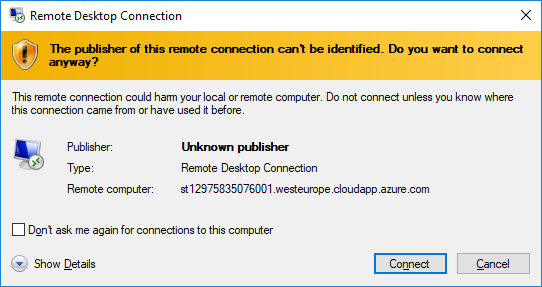
1. When the machine is started, it will be displayed in the “My Virtual Machines” pane.   
   After one minute, the status will be Running. You can wait 30 seconds more before trying to connect on it.



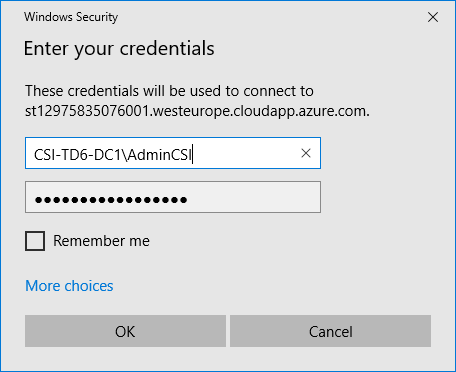
1. Select the running Virtual Machine and at the end of line, click on “…” then select Connect



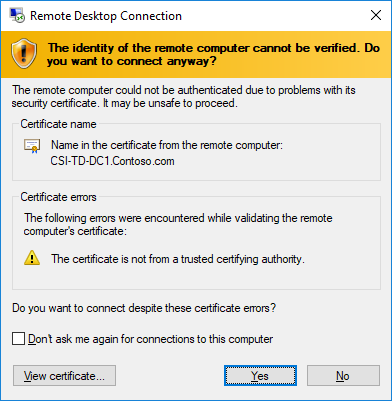
1. A warning is displayed about the publisher. You can ignore the warning and click on Connect.



1. Enter the user name and password to connect to the Virtual Machine detailed in each exercise below.   
   (Do not use your student or organizational account.)



1. A warning on the self-issued certificate is displayed. You can safely ignore this warning by clicking on Yes.



## Exercise 1: Examining system’s current activity

Duration: 1 hour

Synopsis: In this exercise, you will learn how to use the debugger to extract information about processes and threads currently running on a system.

For all the assignments in this exercise, you will work on the machine named **CSW1-LAB** using the following credentials:  
 Username: CSW1-LAB\local\_student  
 Password: 08Fc720C!0eK2

#### Task 1: Prepare the debugging environment

1. Start **Windbg.exe**  
   Note: Windbg.exe is located in the **C:\LabContent\Tools\WinDbg** folder
2. Select **File** \ **Open crash dump** or hit **CTRL-D**
3. Select the **MEMORY.DMP** file from **C:\LabContent\Material\Dump**
4. In order to ease the work inside Windbg, the window must be docked. This is accomplished by selecting the **Dock All** item in the **Window** menu

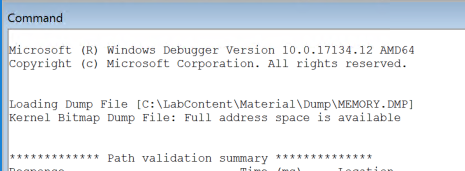
Before moving on with this lab exercise, there are some basic concepts of WinDBG that you need to understand.

WinDBG is a command-line based debugger. The main window is divided into 2 parts:

* **The command text box**. This is basically the location where you will type all the commands to the debugger. It is located at the very bottom of the main window. Most of the time, the text location is prefixed with ***n*: kd>** where n is the current processor ID. On a 2-processor machine, **n** will be either 0 or 1.

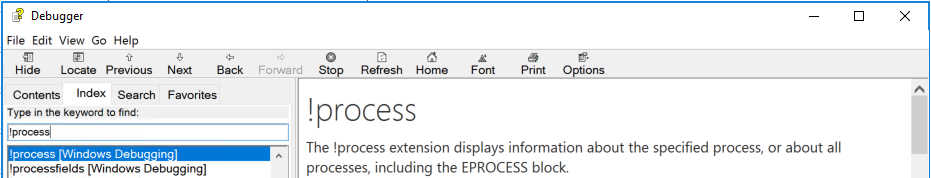


* **The output panel**. It is basically where the result of the commands you type is printed on. It takes all the upper region of the main widow, sitting just above the command text box. All the commands you type will be copied in the output panel prior to display the command result.



There are 3 types of commands in WinDBG:

* **Native commands**: they do not have any prefix character in their name.
* **Meta-commands**: they all start with a ‘**.’** character.
* **Extension commands**: they all start with a ‘**!’** character

At any time, you can consult the documentation of any command by hitting the **F1** key or opening the **Help** \ **Window F1** menu item. Use the **Index** tab an type the command full name to display the help content for that particular command. Example below is for the **!process** command:

Now, you can continue to setup the lab.

1. In the command section, type the following command:

**.sympath**

1. Ensure you get something similar to:

0: kd> .sympath

Symbol search path is: SRV\*C:\LabContent\Symbols\symbols\*https://msdl.microsoft.com/download/symbols

Expanded Symbol search path is: srv\*c:\labcontent\symbols\symbols\*https://msdl.microsoft.com/download/symbols

\*\*\*\*\*\*\*\*\*\*\*\*\* Path validation summary \*\*\*\*\*\*\*\*\*\*\*\*\*\*

Response Time (ms) Location

Deferred SRV\*C:\LabContent\Symbols\symbols\*https://msdl.microsoft.com/download/symbols

#### Task 2: List currently running processes

In this Task, you will display the list of currently running processes by leveraging the !process WinDBG command.

1. Review the help of the **!process** command
2. Determine the proper parameter list to display all the running processes with minimal details for each one.
3. Run your command.

If you ran the correct command, you should see an output similar to:

\*\*\*\* NT ACTIVE PROCESS DUMP \*\*\*\*

PROCESS ffffa7043f68e440

SessionId: none Cid: 0004 Peb: 00000000 ParentCid: 0000

DirBase: 001ca000 ObjectTable: ffff960b7e615000 HandleCount: 2036.

Image: System

PROCESS ffffa7043f7a6040

SessionId: none Cid: 0058 Peb: 00000000 ParentCid: 0004

DirBase: 00200000 ObjectTable: ffff960b7e63d000 HandleCount: 0.

Image: Registry

Additional Questions:

1. Write down the command you used:
2. What is the address of the following processes?
   1. Lsass.exe:
   2. Winlogon.exe:
   3. Services.exe:
3. What is the PID of the following processes?
   1. Wininit.exe:
   2. Explorer.exe:
   3. Powershell.exe:
4. For each of the following processes, give the name of their respective parent process:
   1. Powershell.exe:
   2. Onedrive.exe:
   3. Spoolsv.exe:

#### Task 3: Get details about a running process – part 1

In this task, you will focus on a single process and attempt to get information about it. One location where you can find some interesting data is the PEB *– Process Environment Block*. The PEB is a subset of Windows’s internal control structure for processes which is available in user mode. The process chosen for study is winlogon.exe but, the principles applies to any other process.

1. Run !process ffffa70442304400 0 in order to get the general information regarding winlogon.exe  
    *Note: ffffa70442304400 is the address of winlogon.exe*
2. Copy the address displayed right after **Peb:**, this is the virtual address of the PEB.
3. Run !peb addr where addr is the virtual address of the PEB.  
    **!peb** e934519000
4. You should see an error similar to :

0: kd> !peb e934519000

PEB at 000000e934519000

error 1 InitTypeRead( nt!\_PEB at 000000e934519000)...

This is an error triggered by the debugger because it is unable to read the PEB structure from target’s memory.

You must fix the issue by setting the current process context.

1. Run .process ffffa70442304400 in order to set the correct process context
2. Run again !peb e934519000. You should obtain an output similar to:

0: kd> !peb e934519000

PEB at 000000e934519000

InheritedAddressSpace: No

ReadImageFileExecOptions: No

BeingDebugged: No

ImageBaseAddress: 00007ff705190000

The PEB contains list of loaded DLLs (called modules), the environment variables and the full command line.

Additional Questions:

1. Explain the failure:  
   *Hint: It is highly recommended to read the description of the .process command*
2. Based on the output of the **!peb** command:
   1. What is the value of the **USERNAME** environment variable?
   2. How many modules (main binary + loaded DLLs) were loaded by winlogon.exe?
   3. What is the command line which was used to launch the application?
3. Use the same technique to analyze the svchost.exe process which PID is 0x0174 (PID is in hexadecimal form) and answer the following questions, based on the output of the **!peb** command:
   1. What is the value of the **USERNAME** environment variable?
   2. Did svchost.exe load WinSCard.dll?
   3. What is the command line which was used to launch this svchost.exe process?

#### Task 4: Get details about a running process – part 2

In this task, you will display more details about a running process by dumping the content of Kernel-mode structures. The process studied in this exercise is an instance of powershell.exe.

One of the additional entries brought by **!process** with greater detail level is the address of the process’s token. Process’s token represents the identity of the process when accessing resources and is generally inherited from the user or account which has started it.

Assignments:

1. Run !process ffffa704432a1580 1 in order to get details about the PowerShell process
2. Get the address of the token object
3. Run !token -n *addr* where *addr* is the address of the token object  
    !token -n ffff960b821f2720

Additional Questions:

1. Based on the output of the !token command, answer following questions:
   1. What is the SID of the user?
   2. Does the user belong to the local Administrators group?
   3. What is the Mandatory Integrity Level of that user?
   4. Does the user is granted the SeTcb privilege?
   5. Does the user is granted the SeDebug privilege?
2. Based on the output of the !process command output, how long does the process have been running for?

#### Task 5: Explore a process activity

As you dig deeper in your analysis, you may want to understand what a process is doing. Windows does not schedule processes. Processes are mainly a container and an isolation layer between applications. Windows schedule threads. In this task, you will learn

Assignments:

1. Read the help content regarding the !process and !thread commands
2. Run .process ffffa70442304400 to set the current process to winlogon.exe
3. Run .reload /f /user in order to load winlogon’s symbols  
    Note: Symbols are files which help the debugger to map memory areas with functions and variables names in the source code.
4. Run !process ffffa70442304400 7 in order to list all threads running inside the process.

You should get an output like:

0: kd> !process ffffa70442304400 7

PROCESS ffffa70442304400

SessionId: 1 Cid: 0248 Peb: e934519000 ParentCid: 01e0

DirBase: 08f80000 ObjectTable: ffff960b7f991b80 HandleCount: 256.

Image: winlogon.exe

VadRoot ffffa70442303f70 Vads 77 Clone 0 Private 392. Modified 1786. Locked 0.

DeviceMap ffff960b7e61fd90

<content omitted>

THREAD ffffa70441ffd080 Cid 0248.024c Teb: 000000e93451a000 Win32Thread: ffffa70441e01520 WAIT: (UserRequest) UserMode Non-Alertable

ffffa7044272cf80 SynchronizationEvent

Not impersonating

Additional information about the meaning of some !thread fields:

* Wait Start TickCount : representation in internal time format (ticks) when this thread entered a wait state
* Ticks : difference between the current time as stored in the dump file and Wait Start TickCount. It is expressed in ticks and in usual time format (days:hours:minutes:seconds.milliseconds)
* User Time : how much time the thread ran in user-mode
* Kernel Time : how much time the thread ran in kernel-mode

Additional Questions:

1. How many threads does the winlogon process own?
2. List all thread IDs in hexadecimal form
3. In that instance of winlogon.exe, the thread 0x24c is currently in the WAIT state meaning it iss not running but rather waiting on a resource to be signaled or available. How long this thread has been waiting for?

## Exercise 2: Analyzing memory usage

Duration: 1 hour

Synopsis: In this exercise, you will learn how to use the debugger to extract information about how memory is used by the system.

For all the assignments in this exercise, you will work on the machine named **CSW1-LAB** using the following credentials:  
 Username: CSW1-LAB\local\_student  
 Password: 08Fc720C!0eK2

#### Task 1: Display general memory usage

In this task, you will discover how to get information about general memory usage on the system.

1. Using the help content of WinDbg, find the command which displays general virtual memory information and answer the additional questions

Additional Questions:

1. How much physical memory is installed on this system?
2. How much memory is used by the Non Paged pool?
3. How much memory is used by the Paged Pool?
4. How much memory is Committed by the powershell.exe program?

#### Task 2: Get insights about pools consumption

In this task, you will learn basic commands to visualize the Paged and NonPaged pools allocations

1. Using the help content of WinDBG, familiarize yourself with the !poolused command
2. Run the command: !poolused 1

Additional Questions:

1. Briefly explain what the !poolused 1 command does.
2. How much memory is allocated under the Toke tag?
3. What kind of kernel object does the Toke tag relate to?
4. Find a WinDBG command to list all allocations for a given tag.

#### Task 3: Examine the memory layout of an application

In this task, you will learn how to display the memory layout of an application. Windows Memory Manager keeps internal structure to describe the virtual address space of a process. These structure are called VAD *– Virtual Address Descriptors* and describes the current status of a memory region in the process’s address space. Any valid virtual address belongs to a VAD. In this example, the application is winlogon.exe

1. Run .process ffffa70442304400 to change the process context to Winlogon.exe
2. Run !process ffffa70442304400 1 to display detail process information:
3. Retrieve the address of the root VAD. This address is located right after the VadRoot tag

In the case of winlogon.exe, root VAD is located at address 0xffffa70442303f70

1. Run !vad ffffa70442303f70

You should get an output like:

0: kd> !vad ffffa70442303f70

VAD Level Start End Commit

ffffa70441fe4980 6 7ffe0 7ffe0 1 Private READONLY

ffffa70442304150 5 7ffef 7ffef 1 Private READONLY

ffffa70441ffbce0 6 e934400 e9345ff 13 Private READWRITE

ffffa70441ffdbe0 4 e934600 e93467f 17 Private READWRITE

ffffa7044318a450 6 e934680 e9346ff 17 Private READWRITE

Additional Questions:

1. What is the address of the VAD which correspond to the mapping of profext.dll in winlogon’s virtual address space?
2. What is the start address of profext.dll?
3. What is the physical address which corresponds to the virtual address found in previous question?

#### Task 4: Examine Heap structures

In this task, you will use WinDBG commands to display details about the heaps allocated in a given process. Heap are the mechanism which allow an application to dynamically allocates a randomly sized area in memory. Heap are a user-mode only feature. Heaps exist because the memory manager can only allocate pages, which are generally 4kB large. Heaps allow finer granularity in allocation size. For this example, you will work in the context of the powershell.exe process.

1. Run .process ffffa704432a1580 to change the process context to powershell.exe
2. Run .reload /f to ensure symbols are loaded (will take some time).
3. Using WinDBG help content, find a command to list all heaps created by the process.
4. Dump the details of heap at address 0x1e94eb20000

Note: You may encounter the error ‘Failed to read heap keySEGMENT’ when running your commands. You can safely ignore those errors.

Additional Questions:

1. How many heaps are owned by powershell.exe?
2. How many segments does heap 0x1e94eb20000 own?
3. What is the allocation granularity of this heap?

## Exercise 3: Devices and drivers

Duration: 1 hour

Synopsis: In this exercise, you will learn how to use the debugger to extract information about drivers and devices.

For all the assignments in this exercise, you will work on the machine named **CSW1-LAB** using the following credentials:  
 Username: CSW1-LAB\local\_student  
 Password: 08Fc720C!0eK2

#### Task 1: List driver modules currently loaded by the system

In this task, you will learn how to extract the list of loaded kernel modules. This list is also the list of drivers (in the meaning driver .sys files) leaded by the system. A driver module must not be confused with a driver object as there is no guaranty of a 1:1 mapping.

1. Familiarize yourself with the documentation of the lm command
2. Using this command, find a way to list all drivers loaded by the system
3. Using the same command, find the syntax to display details on a specific module.

Additional Questions:

1. Write down the command you’ve built for step #2 and #3
2. Is the appid.sys driver loaded?
3. What Is the full path of the ntfs.sys driver?
4. What is the file version of ntfs.sys?

#### Task 2: Navigate the object manager

All drivers and devices are represented internally as objects. The object manager is responsible for storing objects, keeping reference counts accurate and organize objects in a tree-like structure.

In this task, you will dump some of the object manager structure in order to list all driver and device objects.

1. Run command .reload /f in order to ensure symbols are loaded.
2. Using the WinDBG help content, find a command to dump objects. Familiarize yourself with command’s options.
3. Dump the list of all objects in the driver namespace \Driver
4. Find a command specially designed to dump the details of a driver object and dump the details of the HTTP driver, including driver’s dispatch routines and device objects owned by the driver.
5. Find a command specially designed to dump the details of a device and dump the details of device 0xffffa704428b1c20
6. Dump the security descriptor of this device

Additional Questions:

1. What is the address of the HTTP driver object?
2. How many devices are owned by this driver?
3. What is the name of the HTTP driver’s routine handling IRP\_MJ\_CREATE IRPs?
4. Regarding the security descriptor of device 0xffffa704428b1c20, which principal is the owner of this device?

#### Task 3: I/O Request Packets

In this task, you’ll learn how the object manager and driver works together to access a file in a storage location. You will start with the C: drive which is the usual location where Windows and applications are installed. The true object name for the C: drive is \Global??\C:

1. Using learnings from previous task, dump the \Global??\C: object
2. Note this object is like a redirector to another object. Dump the true target object.
3. As this object is a device, use the device specific command to dump the device object.
4. When an application opens a file, the device which handles the request is generally an instance of the volmgr driver (volmgr stands for Volume Manager). So, opening a file means asking the volume manager to open the file from the particular volume. But, volmgr does not have the knowledge of the layout of filesystems. It must request some help from a filesystem driver. The link between a specific volume device and the filesystem device is stored in the VPB – *Volume Parameter Block*.  
   Using the help content of WinDBG, find the command which can dump the content of a VPB. Then, use the result from previous step to dump the VPB of the volume. The address of the filesystem device is specified as the DeviceObject field.
5. Now, you have the address of the Ntfs device which handles the C: volume. We want to know all I/O requests which are processed by this volume.  
   Using the help content of WinDBG, find a command which can search IRP using search criterias. Then, run a command to list all IRP which are related to the Ntfs device.
6. We are interested by the IRP which has its minor and major codes equals to 0. This one should appear as (0, 0) in the IRP list.  
   Using the help content of WinDBG, find a way to dump the content of the IRP.
7. In the IRP stack, you can find the address of the file object which is manipulated by the IRP. Use the !fileobj command to dump this file object.
8. The IRP also contains the address of the thread which is currently processing the IRP. The thread address is displayed at the very top of the irp details.  
   Using command from previous exercises, find the process which this thread belongs to.

Additional Questions:

1. Write down the output of command used in step #1.
2. What is the exact object type of \Global??\C:?
3. Write down the output of command used in step #2
4. Write down the output of command used in step #3
5. Write down the output of command used in step #4
6. Write down the command you used for step #5
7. What is the functional description of an IRP with major code equals to 0?
8. Which file is being accessed?
9. Which application is making the call?

## After the Lab

Duration: 10 minutes

In this exercise, attendees will deprovision any Azure resources that were created in support of the lab.

#### Task 1: Stop and deallocated all the VMs

1. Properly shutdown all the VMs
2. Deallocate the VM in the Azure Portal
3. To Stop a VM, simply click on Unclaim.

