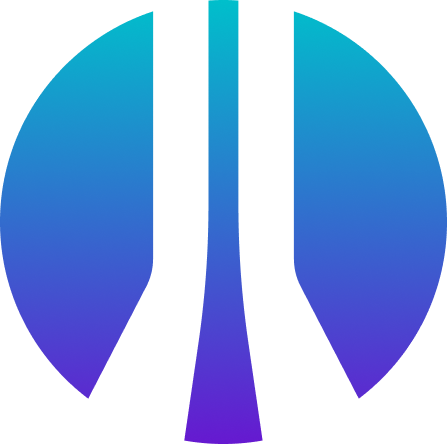
EXP-301 Lab Report

v.2.0

student@youremailaddress.com

OSID: XXXXX



Copyright © 2023 Offsec Ltd. All rights reserved.

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from Offsec.

Table of Contents

1.0 Offsec EXP-301 Lab Documentation [3](#_gjdgxs)

1.1 Objective [3](#_30j0zll)

Note that copy-pasting code from the book modules into a script may result in unintended whitespace or newlines due to formatting. [3](#_1fob9te)

1.2 Extra Miles [3](#_3znysh7)

1.3 Requirements [4](#_2et92p0)

2.0 Exercises [5](#_tyjcwt)

2.1 Summary Overview [5](#_3dy6vkm)

2.2 Accessing and Manipulating Memory from WinDbg (Example Exercise) [5](#_1t3h5sf)

2.3.1 Unassemble from Memory (Example) [6](#_4d34og8)

2.3.2 Reading from Memory (Example) [7](#_2s8eyo1)

13 Challenges [8](#_17dp8vu)

13.1 Challenge 1 (Example) [8](#_3rdcrjn)

## 1.0 Offsec EXP-301 Lab Documentation

The following pages contains the lab exercises for the WUMED course and should be attempted only inside the Offsec hosted lab environment. Please note that most of the attacks described in the lab guide would be illegal if attempted on machines that you do not have explicit permission to test and attack. Since the Offsec lab environment is segregated from the Internet, it is safe to perform the attacks inside the lab. Offsec does not authorize you to perform these attacks outside its own hosted lab environment and disclaims all liability or responsibility for any such actions

## 1.1 Objective

We recommend that you fully complete the exercises for each module prior to moving on to the next module. They will test your understanding of the material and build your confidence to move forward.

Depending on your existing skillset, it may take considerable time and effort to complete the exercises. Nevertheless, we encourage you to be persistent, especially with tougher exercises. Persistence is an essential trait to develop as part of the OffSec "Try Harder" mindset.

To aid in your studying the dedicated student vm contains the folder **C:\proof\_of\_concepts**. Inside this folder you will find exploit code for relevant exercises as marked by module, section, and exercise numbers. Only exercises that result in an updated exploit code have entries in the list.

We encourage you to attempt to solve the exercises on your own before you read the solutions, as this will greatly increase your learning.

**Note that copy-pasting code from the book modules into a script may result in unintended whitespace or newlines due to formatting.**

## 1.2 Extra Miles

Some modules include extra mile exercises, which are more difficult and time-consuming than regular exercises. These exercises are not required to learn the material, but they will you help develop extra skills and succeed on the exam. Also note that solutions to these extra miles are not given on your student vm.

## 1.3 Requirements

The student will be required to fill out this lab report fully and to include the following sections:

* High-Level summary of assignment solutions.
* Methodology walkthrough and detailed outline of steps taken through analysis and all written code.
* Each finding with included screenshots, walkthrough, sample code or reference.
* Screenshots of the final working exploit against your target.

2.0 Exercises

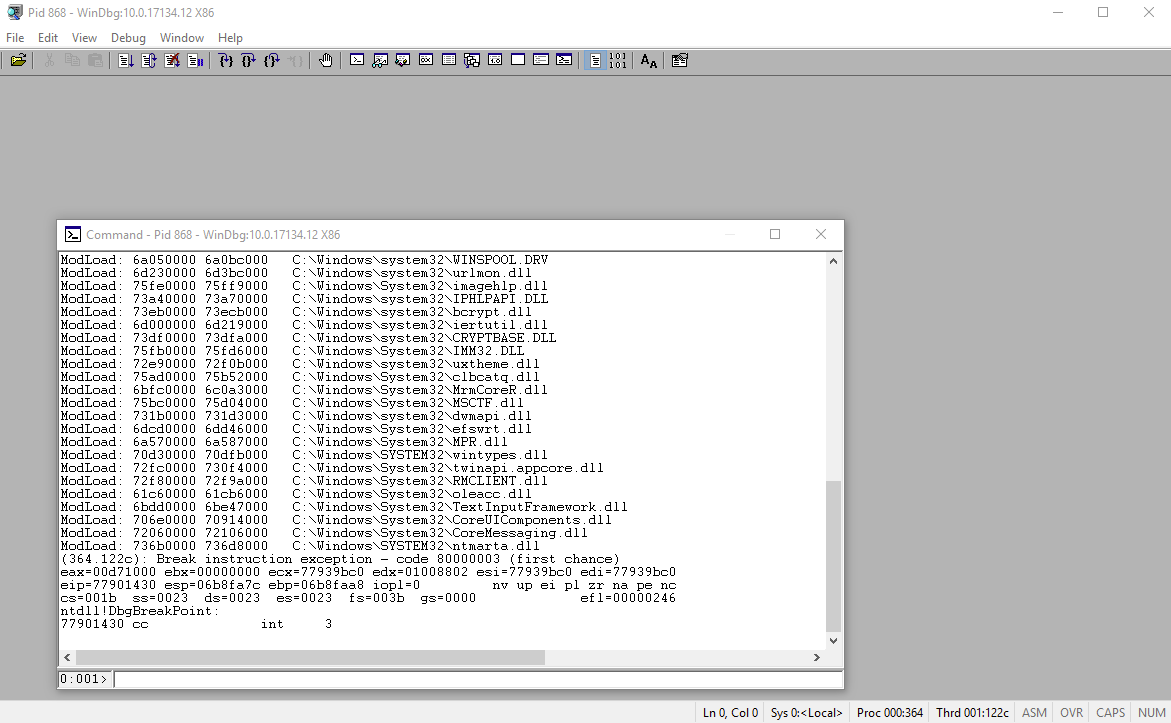
## 2.1 Summary Overview

A brief description of the assignments that were solved, including the overall exploitation / development steps.

E.g.: The supplied lab systems each had a vulnerable application installed, our goal was to disassemble or reverse engineer these applications. The following exercises were solved using Windbg and other relevant software namely IDA Pro. The following was achieved using Python and its libraries along with the C language and Assembly. Each section below contains the steps taken for each module.

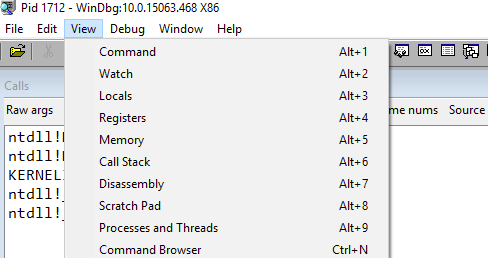
## 2.2 Accessing and Manipulating Memory from WinDbg (Example Exercise)

2.2.1 Open WinDbg and attach it to the Notepad process.



2.2.1 Explore different WinDbg windows and get a feel for the layout.

I like the default look because it is clear to read....

****

## 2.3.1 Unassemble from Memory (Example)

2.3.1.1 Use the *u* command to unassemble the *kernel32!GetCurrentThread* Windows API.

0:003> u kernel32!GetCurrentThread

KERNEL32!GetCurrentThread:

770b5910 6afe push 0FFFFFFFEh

770b5912 58 pop eax

770b5913 c3 ret

770b5914 cc int 3

770b5915 cc int 3

770b5916 cc int 3

770b5917 cc int 3

770b5918 cc int 3

2.3.1.2 Can you explain the assembly code? What is the result of this function and how it is returned to the caller?

*Short Descriptive answer here for each command...*

## 2.3.2 Reading from Memory (Example)

2.3.2.1 Use different versions of dd to dump data from memory and attempt to combine the display commands with poi.

0:000> db esp

00faf974 89 ab 1b 77 78 68 1f c1-50 ab 1b 77 50 ab 1b 77 ...wxh..P..wP..w

00faf984 00 00 00 00 78 f9 fa 00-00 00 00 00 ec f9 fa 00 ....x...........

00faf994 80 a3 18 77 90 ae fa b6-00 00 00 00 b4 f9 fa 00 ...w............

00faf9a4 a4 de e2 76 00 00 00 00-80 de e2 76 8a ae aa ca ...v.......v....

00faf9b4 fc f9 fa 00 be 00 15 77-00 00 00 00 24 68 1f c1 .......w....$h..

00faf9c4 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................

00faf9d4 00 00 00 00 00 00 00 00-00 00 00 00 24 68 1f c1 ............$h..

00faf9e4 c0 f9 fa 00 00 00 00 00-04 fa fa 00 80 a3 18 77 ...............w

13 Challenges

Take the time to work on these challenges and develop a methodology for enumeration, reverse engineering, and exploit development.

## 13.1 Challenge 1 (Example)

Challenge 1 makes use of the Intelligent Management Center (iMC) application portfolio by HP Enterprise. It contains a multitude of applications listening on more than 15 different network ports, offering a wide attack surface.

Several hundred vulnerabilities have been found in the application over the last 5 years. In this challenge, the TFTP server that comes packaged with iMC is the target.

These are the steps followed below:

* Running an NMAP scan against the target showed the port closed

#➤ [wumed] sudo nmap -sC -sV -sU -T2 -p 69 192.168.70.40

Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-09 07:32 EDT

Nmap scan report for 192.168.70.40

Host is up (0.17s latency).

PORT STATE SERVICE VERSION

69/udp closed tufts

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 1.81 seconds

* Unable to see the port from our vm we rdesktop into the system:

sudo rdesktop -u admin -p lab -g 90% 192.168.78.100

* and attach the process running the IMC service
* Once attached we able to start reversing the application

