

CSC 4100 “Operating Systems” Creative Inquiry Project

Buffer Overflow Exploitation and Protection Techniques

Grading and Submission Summary

Total Points	100
Due Date	01 May 2020, 11:59pm CDT
Submission System	iLearn
Submission Instructions	Submit your source code file and one pdf file with the report as one zip archive

Overview

This exercise teaches students about buffer overflow vulnerabilities, their exploitation and protection techniques. According to “Common Vulnerabilities and Exposures” database [1], buffer overflows are still a serious threat for OS kernels CVE-2019-14835, drivers (CVE-2019-14897, CVE-2019-14895) and utilities (CVE-2019-17652). An attacker can provide a malicious input to a process or an application that has a buffer overflow vulnerability and execute a malicious code. C/C++ programming languages that are widely used to develop OS kernels, drivers and applications, do not provide built-in memory protection. It is a programmer’s job to apply safe coding techniques and avoid potential buffer overflows. In this lab, you will explore a simple archiver that is written in C language and works under Linux OS. The archiver’s source code has a buffer overflow in it. Firstly, you need to exploit the buffer overflow vulnerability by providing an “unusual” input file name that you want to archive. The goal is to redirect the program’s execution flow to a “secret” function, which will print a secret message for you, i.e. a “flag” that you need to capture. Secondly, you will need to apply a fix to a vulnerable archiver so that your attack does not succeed anymore and the archiver can be included in the next OS release. We ask you to be creative and explore multiple ways to fix the buffer overflow vulnerability.

Learning Objectives

- Use digital information search tools
- Explore a buffer overflow problem by choosing an appropriate discovery process to address the problem, collecting information relevant to the problem and assessing collected information
- Apply critical thinking skills to compare alternative solutions to fix buffer overflow vulnerability; make analysis and conclusions
- Apply computer science theory to design and implement computing-based solution to protect a program, written in C language, against buffer overflow attacks
- Communicate and report your findings.

Note: this assignment was designed with the support of the TTU EDGE Curriculum Grant Program, to integrate creative inquiry ideas and activities, as part of Tech's Quality Enhancement Plan (QEP) program, EDGE: Enhanced Discovery through Guided Exploration.

Project Guidelines, Questions and Tasks

Guidelines

Step 1. Follow instructions from Appendix 1 and log into your CyberRange Virtual Machine instance (box), the username is **root** and the password is **toor**.

Step 2. Go to *Desktop/Compression* directory. You should see ***bufferoverflowX*** executable file, which is an archiver, where X is some number. You should also see the file *compress_me.dat* and a source code file *bufferoverflow-generic.c*

Step 3. Try to compress the file *compress_me.dat* by executing the archiver *bufferoverflowX*. Run it as

```
$ ./bufferoverflowX
```

When you are asked, type in the file name to archive. It should be *compress_me.dat*

See whether the archive is generated in the current directory *Desktop/Compression*.

Question 1: Can you archive the same file *compress_me.dat* again? What is the result? Insert the screenshot of the command line terminal and of the *Desktop/Compression* directory content and your answer into your report.

Question 2: What would you change in the archiver's user interface? Write your answer in your report.

Step 4. Explore the generic archiver's source code in *bufferoverflow-generic.c*. You archiver's binary executable file 'bufferoverflowX' is compiled from the different source code which you don't have access to. It has the same functions *FileCompress()* and *main()* as in the *bufferoverflow-generic.c* but different function *secretFunction()*.

Step 5. Read the tutorial "Buffer Overflow Exploit", published by Dhaval Kapil [2].

Question 3: Draw a stack memory layout for *FileCompress()* function in your report. It must include local variables declared in *FileCompress()* function, the Base pointer register (*%ebp*) and the return register.

Hint: look at the stack memory layout figure from the "Memory Layout of a C program" section in [2].

Step 6. Using your stack memory layout from step 5, think what should be your input file name, which you are asked to enter when archiver starts, so that the return register is overwritten with the address of *secretFunction()* from the archiver's binary executable *bufferoverflowX*.

Step 7. Find out the address of *secretFunction()* in your executable *bufferoverflowX*. Write the address in your report

Hint: objdump utility might be helpful

Question 4. Does OS on your VM instance use Big Endian or Little Endian system? How will you figure it out? Write your answers in your report.

Note: the question on how to find out whether the OS uses Big Endian or Little Endian is a very popular question for coding interviews

Step 8: now you are ready to try a buffer overflow attack on your archiver's executable `bufferoverflowX`. You need to run the executable `bufferoverflowX` and when you are asked to enter the file name, provide a malicious input string so that it overwrites in stack memory the return register with the address of `secretFunction()`

Hint: you can use the following python command:

```
$ python -c 'print "a"*NN + "\xDD\xCC\xBB\xAA"' | ./bufferoverflowX
```

Explain, what does this command do. Write your answer in your report.

Find correct values of NN, AA, BB, CC, DD so that the python call from above overwrites the return register in stack memory and jumps to the `secretFunction()` from your binary executable file `bufferoverflowX`.

Once you succeed, you will see the following message, in addition to Segmentation Fault error:

“You have entered in the secret function! The”

The end of this message is different for each user. It is the “flag” that you need to capture. Write your flag (message from the `secretFunction()`) in your report.

Step 9. Do a literature review on buffer overflow attacks and protection mechanisms.

a) go to cve.mitre.org web site and count how many buffer overflow vulnerabilities are reported there. Pick any two of them which you think are critical and describe them. Write your answers in your report

b) go to scholar.google.com and search for academic publications on buffer overflow attacks and protection mechanisms. Select one publication you like, read it carefully and write a brief summary of it in your report.

c) Using any web search engine, search for methods to protect against buffer overflow attacks. Write one-page summary in your report. Cite source references which you used to write a summary.

Step 10. Fix the buffer overflow vulnerability in the executable *bufferoverflowX*. Remember that you don't have a real source code of this executable file. Explain your solution. Discuss at least two methods to protect against buffer overflow problem in binary executables that were developed in C/C++ languages. Compare pros and cons of the protection methods that you selected. Which method would you prefer?

Step 11. Fix the buffer overflow vulnerability in the source code *bufferoverflow-generic.c*. Describe your solution in your report and attach the modified source code file *bufferoverflow-generic.c* with the comments in the code.

Discuss at least two methods to protect against buffer overflow problem when you have access to the C/C++ source code of the program with buffer overflow vulnerability. Compare pros and cons of the protection methods that you selected. Which method would you prefer?

Step 12. Try the same buffer overflow attack from step 8 on the executable *bufferoverflowX* and make sure it does not succeed anymore after you applied your fix in step 10.

Step 13. If you changed the source code in *bufferoverflow-generic.c*, recompile the source code and try the buffer overflow attack. Make sure it does not succeed. Insert the compilation command in your report.

Question 5: What would you change in the implementation of the main archiver's functionality, besides fixing the buffer overflow vulnerability? Write your answer in your report.

Question 6. As you saw, C/C++ are memory unsafe languages. Why don't we stop using these languages? Insert your thoughts in your report.

Question 7. Find out whether any source code level tools exist to prevent buffer overflows in C/C++ programs. Include the description of at least one tool/method in your report.

Question 8: Explain each local variable and its purpose in FileCompress() function. Explain each of the following calls:

```
gets(buffer);
```

```
strcat(exists, buffer);
```

```
status = system(exists);
```

```
strcat(cp, buffer);
```

```
strcat(cp, " ");
```

```
strcat(cp, buffer);
```

```
strcat(cp, "1");
```

```
system(cp);
```

```
strcat(zip, buffer);
```

```
status = system(zip);
```

```
strcat(mv, buffer);
```

```
strcat(mv, "1");
```

```
strcat(mv, " ");
```

```
strcat(mv, buffer);
```

```
system(mv);
```

Include your answers either in your report or as comments for each function call in the source code file `bufferoverflow-generic.c` in `FileCompress()` function

Step 14. List C/C++ functions that are potentially vulnerable to buffer overflows. Discuss the C/C++ functions that can be their safe replacement

Step 15. Briefly (no more than 10 sentences) explain the meaning and protection mechanisms for the following types of attacks:

- Return-to-libc
- Return-oriented programming
- Integer overflow
- Format string vulnerability

If you cite some source, do not forget to insert source reference. If you insert the exact quote, do not forget to put quotes sign and insert the reference with the source of the quote.

Are any of these 4 types applicable to our *bufferoverflow-generic.c*?

Step 16. Do you know any hardware that has a built-in protection against buffer overflows in the software that runs on that hardware? Include the brief (no more than half page) description of the hardware and hardware-based protection mechanism in your report.

Question 9: Explore the Virtual Machine instance and describe the things that drew your attention (no more than one page) in your report. What operating system does the Virtual Machine use? How about the time format? What text editor did you use to modify the source code file `bufferoverflow-generic.c`? Write your answers in your report.

Submission

You are required to submit two files: *bufferoverflow-generic.c* source code file with fixed buffer overflow vulnerability and a report file. **For the name of your report file, follow this naming convention:**

<your_username>_{CSC4100 or CSC5100}_OverflowLab.pdf

Example: **johndoe_CSC4100_OverflowLab.pdf**

If you have technical problems with the project, contact Dr. Denis Ulybyshev (dulybyshev@tntech.edu).

References

[1] “Common Vulnerabilities and Exposures”, [Online]: <http://cve.mitre.org/>

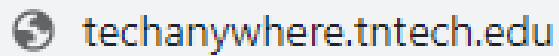
[2] Dhaval Kapil, “Buffer Overflow Exploit”, [Online]:

<https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/>

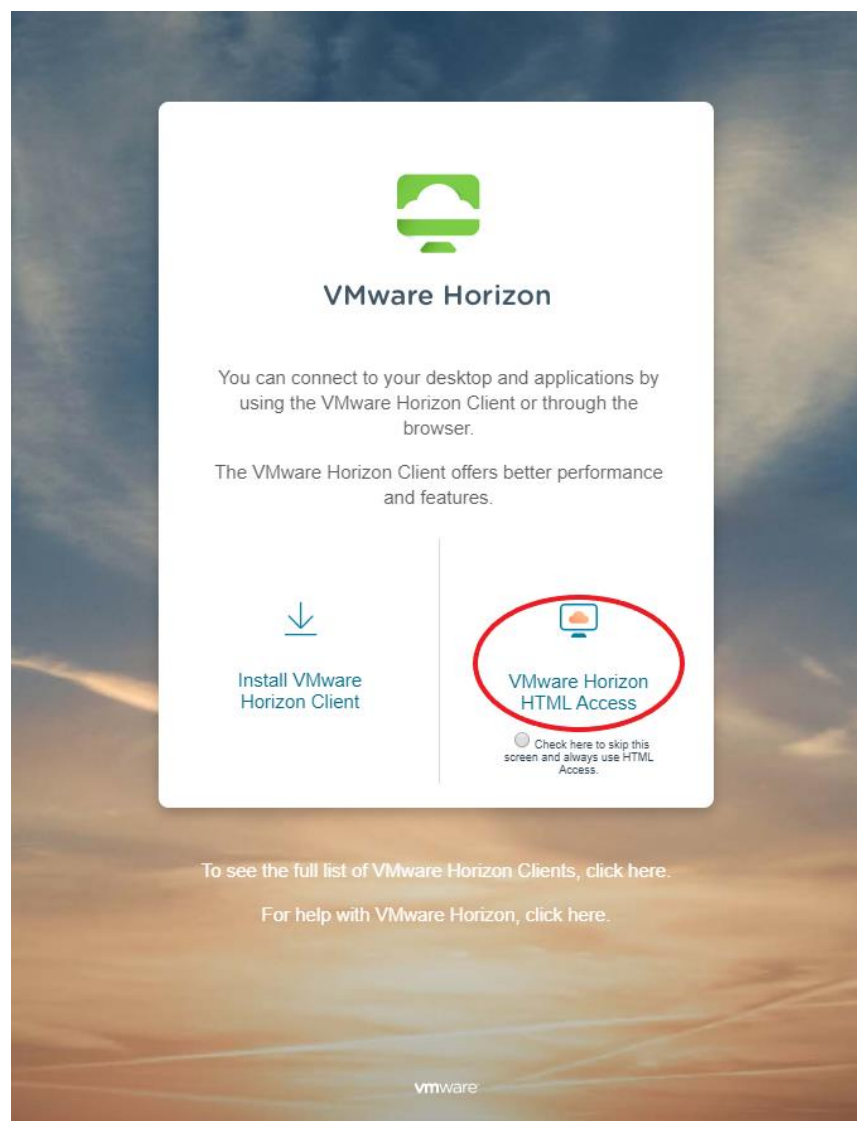
APPENDIX 1. HOW TO USE TECHANYWHERE TO ACCESS THE CYBERRANGE

(created by Kendall Land, modified by Denis Ulybyshev)

1. Open a browser, type **techanywhere.tntech.edu** into the address bar, and press enter.



2. Once the page loads, click **VMware Horizon HTML Access**.



- Next you will be taken to Tech's IDme single sign on page, if you are not already logged in. If this is the case, just log in with the first part of your school email and your school email password, otherwise move onto the next step. (Note: if it also prompts for you to enter a One Time Passcode, just go to wherever it tells you it was delivered and enter it in)

IDme

Username

jdsmith42

Password

.....

*First Time User? Click **Login** to Register.*

Login

[Change Password](#) | [Forgot Password](#)

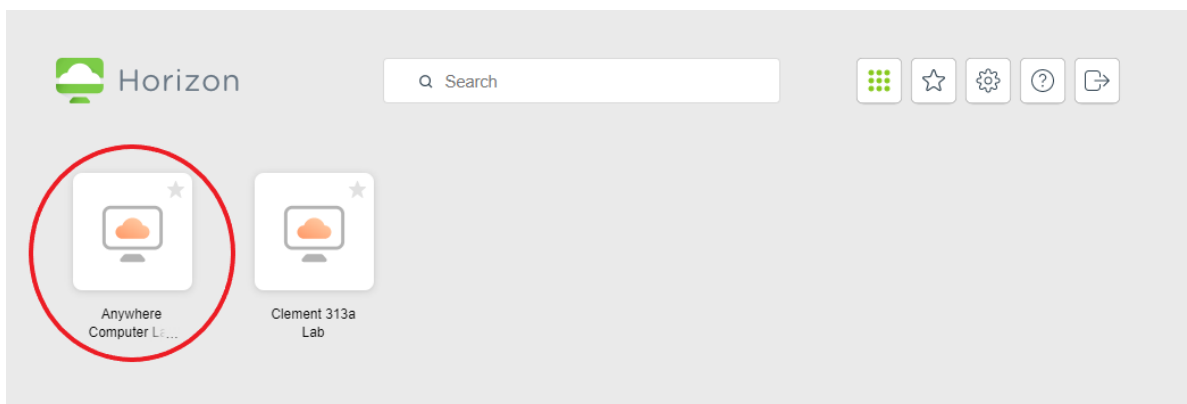
Need help? We have a [self-help guide!](#)

This system may contain Government, University or Student information, which is restricted to authorized users ONLY. Unauthorized access, use, misuse, or modification of this computer system or of the data contained herein or in transit to/from this system constitutes a violation of state and federal laws including, but not limited to Title 18, United States Code, Section 1030, and may subject the individual to Criminal and Civil penalties pursuant to Title 26, United States Code, Sections 7213(a), 7213A (the Taxpayer Browsing Protection Act), and 7431.

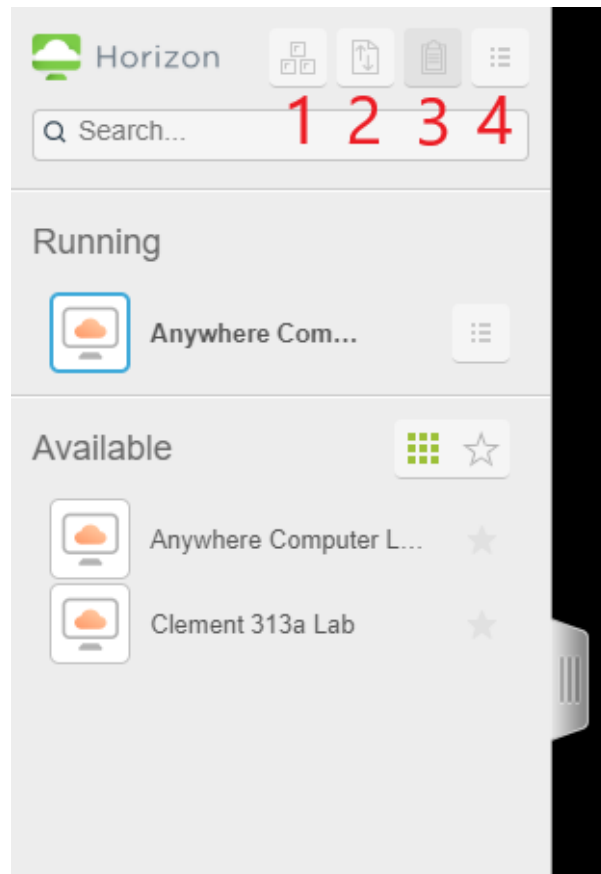
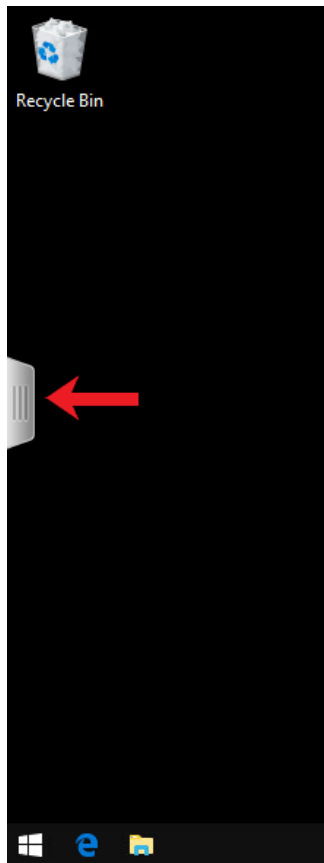
This system and equipment are subject to monitoring. Such monitoring may result in the acquisition, recording and analysis of all data being communicated, transmitted, processed or stored in this system by a user. If monitoring reveals possible evidence of misuse or criminal activity, such evidence may be provided to appropriate officials including but not limited to law enforcement personnel.

ANYONE USING THIS SYSTEM EXPRESSLY CONSENTS TO SUCH MONITORING and HAS NO EXPECTATION OF PRIVACY for any activity, access, use or information stored or communicated via this system.

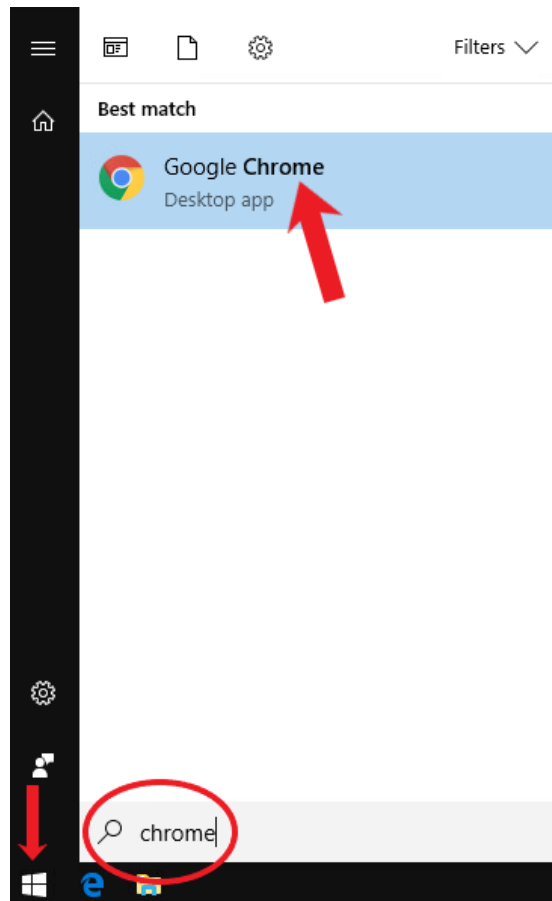
- Once you are logged in, you will be taken to the following screen that shows two different Virtual Machine's (VM's). Click the one on the left that is labeled **Anywhere Computer Lab II**.



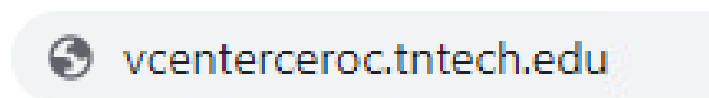
5. The virtual machine should start up and you should see a blank windows desktop. From here if we click the tab on the left side of the screen, we can see what VM we are running, other possible VM's to run, and some buttons at the top. These buttons can be used to send Ctrl+Alt+Delete to the VM (1), transfer files between your host machine and the VM (2), enable/disable copy & paste between your host machine and the VM (3), and open a menu with various settings. As shown in the screen shot, copy & paste should be enabled by default.




6. Now that we are in the VM, we need to open Google Chrome in it, so click the windows symbol in the bottom left of the VM, type **chrome** (you do not need to click a search button you can just start typing), and then click **Google Chrome**.



7. Now type into the address bar of the Google Chrome window you opened in the VM **vcenterceroc.tntech.edu** and hit enter.



8. Click **LAUNCH VSPHERE CLIENT (HTML5)**.



Getting Started

The vSphere Flash-based Web Client is deprecated in vSphere 6.7. We recommend switching to the all-new modern HTML5-based vSphere client as the primary client and only reverting to the Flash-based Web Client when necessary.

LAUNCH VSPHERE CLIENT (HTML5)

LAUNCH VSPHERE WEB CLIENT (FLEX)
Deprecated

Documentation

[VMware vSphere Documentation Center](#)
[Functionality Updates for the vSphere Client \(HTML5\)](#)

9. Now login to vSphere with the same credentials as you did for the single sign on page.

VMware[®] vSphere

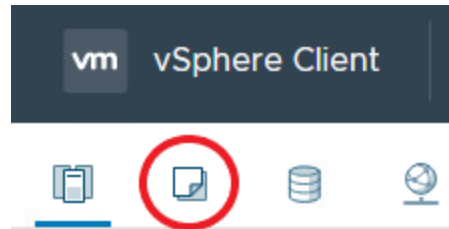
jdsmith42

.....|

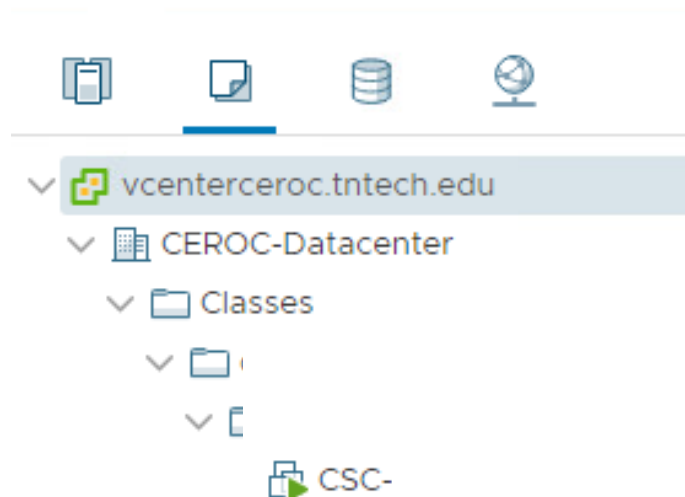
☐ Use Windows session authentication

LOGIN

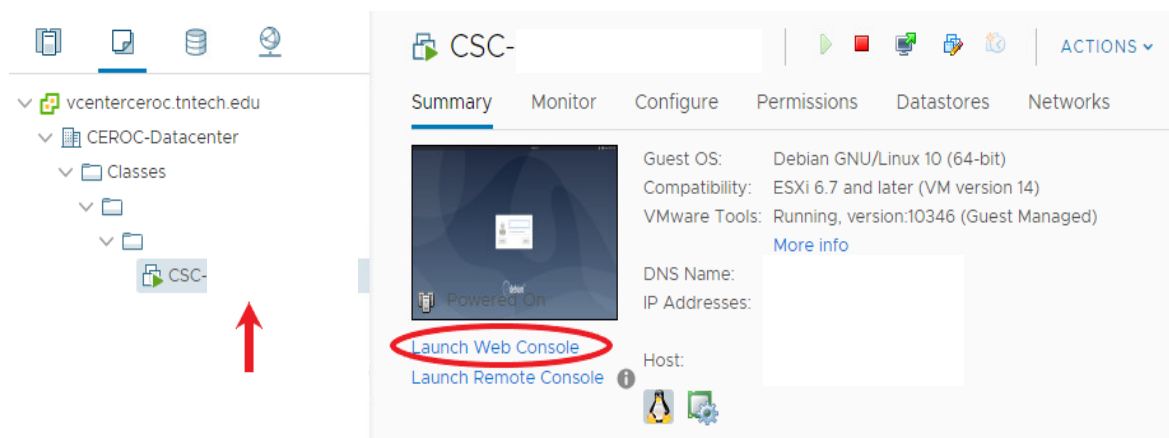
10. In the top left you will see the following 4 symbols, click the second one.



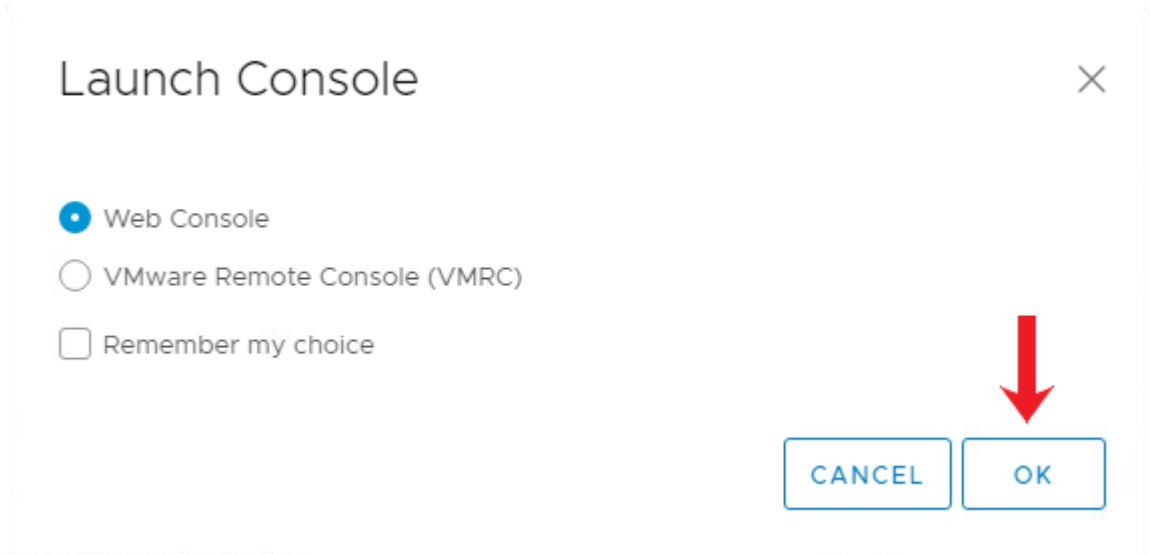
11. From here, we can select drop down to navigate to the path of the virtual machine in vSphere we want to use. (In the screenshot is just an example path. Your path will differ. You need to find **“Overflow”** directory under **Classes**)



12. If you then click the virtual machine you want to use, we can see a summary of the machine. From which we can open the virtual machine by clicking **Launch Web Console**.



13. You will the get the following pop up window, just click **OK**.



14. It will then put you into a new tab with VM in it, where you can then use your virtual machine.
The credentials to login in your VM: root / toor