Tu Lam

CS 373 (Defense Against the Dark Arts)

Dr. Bram Lewis

October 20th, 2021

Homework #4 Write-Up

Week #4 of the class have a special guest; his name was Brad Antoniewicz. He works for Foundstone, which is a sub company of McAfee. They mostly deal with attacking the software to test their ability to protect well or not with their security and also see what type bugs can be found during an attack. Through this week lecture, we will be dealing with the ***software vulnerabilities*** and ***common exploits*** that happen in cybersecurity.

First, we dive into an example of how an attack would look like. Below is an image of what the example is about. In there, we were giving an instruction to enter what happen if enter a direction to go. Brad explains why not go outside of the box and head straight instead of left or right, because the code tells us if the box is blank, go to the direction that we tell them too. Through this, we could see the code vulnerably in the software as there is no protection that tell the user going straight is not a good idea. From this, we can at two ideas of software manipulation, one is ***software vulnerabilities*** where we look at the code to find the weakness and take control of the program, and then ***configuration vulnerabilities*** where user let their guard down and make their system unsecure and easy to break in with an attack.

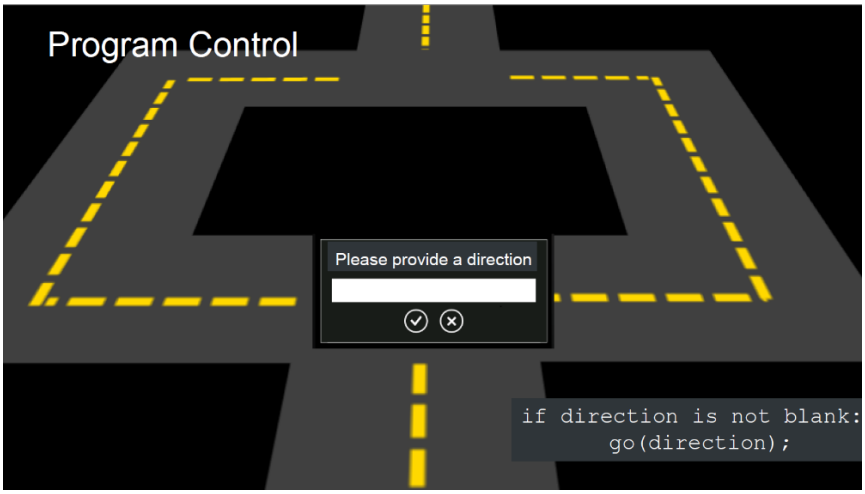

Figure #1: [**From Lecture**] An Exercise on what to do with Direction

Then, Brad went into describing the past about software vulnerability and that it is good to learn it. Many have found flaw in many systems that we use today and reported back to the company giving them time to tackle on that idea and fix it before anyone bad could attack it. Next, Brad introduce the concept of the ***Bug Bounty Program*** is where you are given a piece of software, test it and find if there are any vulnerability that you discover and report it back to the company and get some money. Brad then continues give a brief history that vulnerability attack goes back to five to seven years ago (during the time of the lecture recoding) where most of those attack happen on the internet at the ***DMZ (the border of the system)*** and these could be access through via email server or any server that connect to the internet. So, most company solution out there is treating those DMZ with updates to help it get protected. On top of that barrier, there is a firewall block the connection if an attack is trying to access the system giving it restriction.

Then, Brad move onto the concept of ***exploitation vulnerability*** this is where we found one or more vulnerabilities, then use it and exploit the vulnerability and take it over and control it. We first got introduce to ***WinGDB*** as a type of a basic debugger. He uses a simple program to test WinGDB by opening a program called "simplestack". Then he went on to explain how the module load up the program and how memory is store in a disassembly code. Next, he demonstrates the "***lmf***" command to list out all the module in WinGDB or a specific one base on the name that user want to search. Other commands were mention to learn about the memory instruction are "***u***", "***db***", and "***bp***". This is part of the lab as he exposes the students into learning about WinGDB and how to set breakpoint, find the EIP, and other stuffs to find the vulnerability in the code. Brad also introduce the concept of register and taking a look at ***EIP*** where the it points in memory where the instruction is, and looking at ***EAX*** as a register that store the return value. Below is the figure that go into depth of it more.

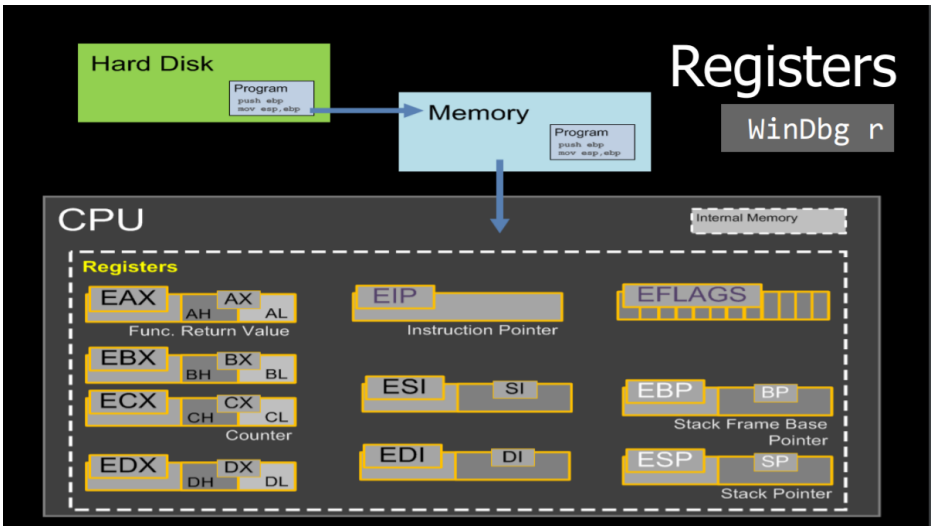

Figure #2: The layout of the entire register in WinGDB

Follow up with a quote from lecture, we have "Accessing memory in an invalid way which results in an **undefined behavior**". This means that the undefined behavior is what we are looking for to control the code. With this we can understand we the code break and take control of the exploit of the vulnerability. Moving on the exploit content, we were able to learn two new vocab words related to the exploit. First word is ***Vulnerability Trigger*** where it describes as "Invokes the software bug to obtain control of the program", and then the word ***Payload*** as an "Action to be performed when control is obtained". We then explore more on the stack and learn more about it and talk about stack overflow. Below is an example of the stack overflow that we look at and from there we can use this as a program control. From there we can follow four steps to allow us to use the exploitation of it. The steps layout as: 1. ***Crash Triage*** where we determine the way on how it is control, 2. ***Determine the return address offset*** where we find the address of the final result, 3. ***Position our shellcode*** where we would implement our code into the program, and 4. ***Find the address of our shellcode*** where we locate the exploit we did on the code.

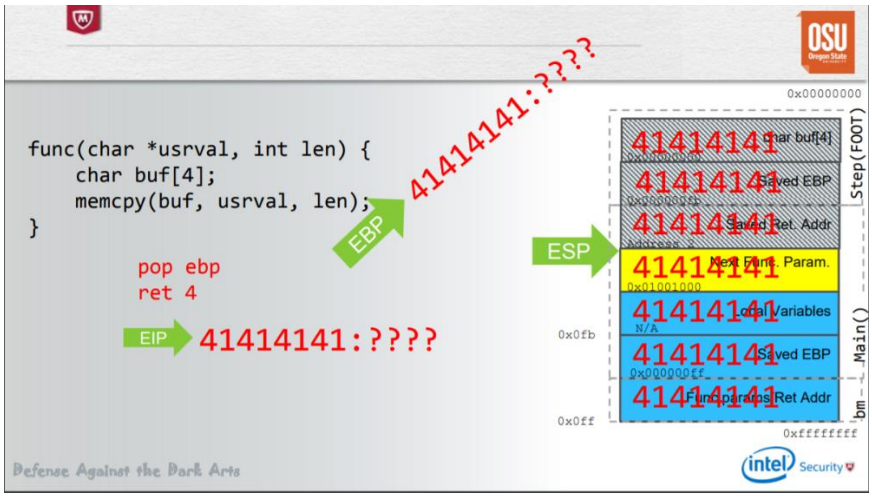

Figure #3: The stack overflow example on how to exploit the vulnerability

Lastly, we move on to the topic of heap where we implement the exploit by using the method of ***use-after-free***. This method includes: 1. ***Free the object***, 2. ***Replace the object with ours*** (a. Figure out the size & b. Make allocations of the same size), 3. ***Position our shellcode***, 4. ***Use the object again***. Through this process, we can exploit the content on the heap when allocating data.

Throughout this week, we are able to learn about the vulnerability program could be use as an advantage of exploitation on our hand to test and see how could couple vulnerability can break the program. Overall, it is a great concept to learn and with the practice of this, anyone could find bugs and report if any system they found could have flaw in it before hacker could take advantage of it.