Lab 10

Venkata Diwakar Reddy Kashireddy

***Abstract*—In this lab, we learned about Ghidra tool and how to perform binary analysis and manipulate assembly code to crack passwords and to convert assembly code to C code to have better understanding of the program.**

I. Introduction

IN this lab, we learned about Binary analyis. We were instructed to download and install Ghidra, an open-source program developed by the National Security Agency for reverse engineering. We have learned about Ghidra tool to perform binary analysis, understand the source code which is in assembly language, patch it to crack passwords, bypass without passwords and get access.

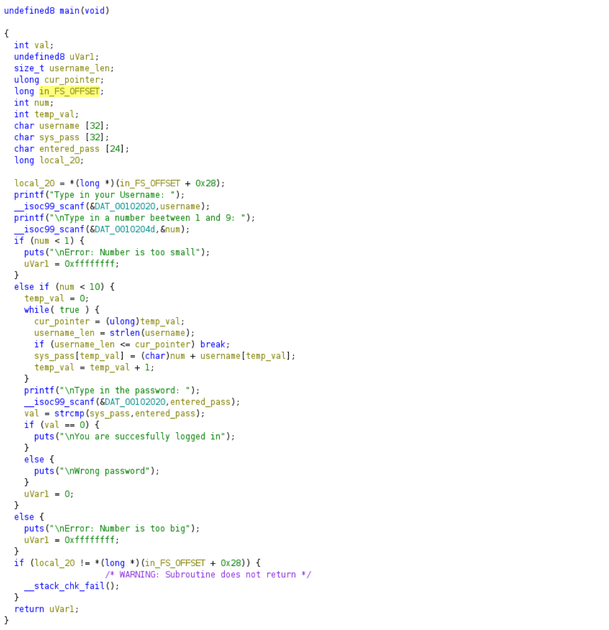
II. Tools

- KVM (Kernel-based Virtual Machine): It is a full virtualization solution for Linux. [1]
- Ghidra: Ghidra is a software reverse engineering (SRE) framework created and maintained by the National Security Agency Research Directorate. [2]
- Overleaf: It is a collaborative cloud-based LaTeX editor that helps to create documents easily by providing standard formats. [3]
- GitHub: GitHub is an Internet hosting service for software development and version control using Git. [4]
- RED: – IU Research Desktop (RED) is a virtual desktop service for users with accounts on the Carbonate research supercomputer at IU. [5]

III. Retrieving the password and the number from the binary using Ghidra

In our first task, we had to analyse the crackme.bin file using Ghidra tool. As mentioned before, Ghidra is a software reverse engineering (SRE) tool by National Security Agency, used to analyse malicious code to get better under standing of the potential vulnerabilities in networks and systems. So, in our task, the C generated code from the assembly code of the crackme.bin main's function by Ghidra contains variables that are not easy to understand. So, first we need to change the variables in such a way that they are easy to understand. From the program, we can understand that, the password is generated using the username and number provided.

The password is generate by shifting the username by provided number places. Suppose, the username is 'abc' and the number provided is 1, then the password generated would be 'bcd'.

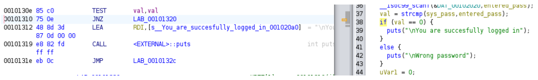


```c
undefined8 main(void)

{
  int val;
  undefined8 uVar1;
  size_t username_len;
  ulong cur_pointer;
  long in_FS_OFFSET;
  int num;
  int temp_val;
  char username [32];
  char sys_pass [32];
  char entered_pass [24];
  long local_20;

  local_20 = *(long *)(in_FS_OFFSET + 0x28);
  printf("Type in your Username: ");
  __isoc99_scanf(&DAT_00102020,username);
  printf("\nType in a number beetween 1 and 9: ");
  __isoc99_scanf(&DAT_0010204d,&num);
  if (num < 1) {
    puts("\nError: Number is too small");
    uVar1 = 0xffffffff;
  }
  else if (num < 10) {
    temp_val = 0;
    while( true ) {
      cur_pointer = (ulong)temp_val;
      username_len = strlen(username);
      if (username_len <= cur_pointer) break;
      sys_pass[temp_val] = (char)num + username[temp_val];
      temp_val = temp_val + 1;
    }
    printf("\nType in the password: ");
    __isoc99_scanf(&DAT_00102020,entered_pass);
    val = strcmp(sys_pass,entered_pass);
    if (val == 0) {
      puts("\nYou are succesfully logged in");
    }
    else {
      puts("\nWrong password");
    }
    uVar1 = 0;
  }
  else {
    puts("\nError: Number is too big");
    uVar1 = 0xffffffff;
  }
  if (local_20 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return uVar1;
}
```

Fig. 1. Main function C code generated by Ghidra with changed variables.

IV. Patch and Bypass the Password

In this task, we were asked to patch the code and bypass the password to gain access. When we analyse the C code, the password entered will be checked with the actual generated password. String compare method has been used to compare the entered password and the actual password. The output of the 'strcmp' method is assigned to a variable and if both the passwords match, then the variable will have a value of 0, else some value other than 0.

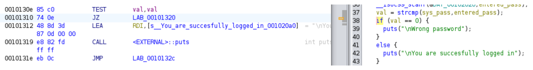


```
0010130e 85 c0          TEST     val,val
00101310 75 0e          JNZ      LAB_00101320
00101312 48 8d 3d       LEA      RDI,[s__You_are_succesfully_logged_in_001020a0]  = "\nYou
         87 0d 00 00
00101319 e8 82 fd       CALL     <EXTERNAL>::puts                                  int puts
         ff ff
0010131e eb 0c          JMP      LAB_0010132c
```

```c
  __isoc99_scanf(&DAT_00102020,entered_pass);
  val = strcmp(sys_pass,entered_pass);
  if (val == 0) {
    puts("\nYou are succesfully logged in");
  }
  else {
    puts("\nWrong password");
  }
  uVar1 = 0;
```

Fig. 2. Orginal Code

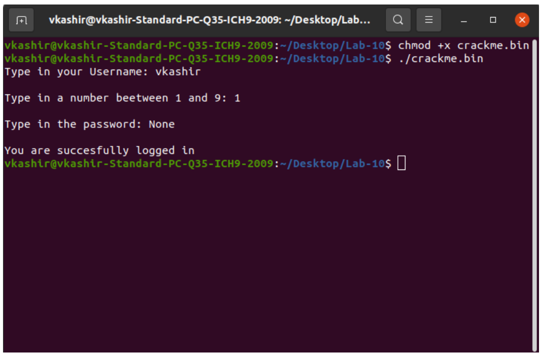
This was done using if else statement. I have patched the logic in such a way that if the varible is anything other than 0, we have gained access.



```
0010130e 85 c0          TEST     val,val
00101310 74 0e          JZ       LAB_00101320
00101312 48 8d 3d       LEA      RDI,[s__You_are_succesfully_logged_in_001020a0]  = "\nYou
         87 0d 00 00
00101319 e8 82 fd       CALL     <EXTERNAL>::puts                                  int puts
         ff ff
0010131e eb 0c          JMP      LAB_0010132c
```

```c
  __isoc99_scanf(&DAT_00102020,entered_pass);
  val = strcmp(sys_pass,entered_pass);
  if (val == 0) {
    puts("\nWrong password");
  }
  else {
    puts("\nYou are succesfully logged in");
  }
```

Fig. 3. Patched Code

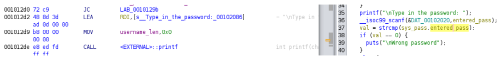


```
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ chmod +x crackme.bin
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ ./crackme.bin
Type in your Username: vkashir

Type in a number beetween 1 and 9: 1

Type in the password: None

You are succesfully logged in
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ 
```

Fig. 4. Bypass password by patching the assembly code

V. Patch and Print the Password (Bonus)

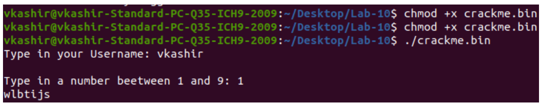
In the last bonus task, we were asked to print the actual system generated password, so we analyse the Ghidra generated C code, we can identify the the variable that maintains the actual system generated password. So, I used the variable and patched the assembly code to print the generated password, after the username and number is entered but before the program asks the user to enter the password.



```
001012d0 72 c9          JC      LAB_0010129b
001012d2 48 8d 3d       LEA     RDI,[s__Type_in_the_password:_00102086]     = "\nType in
         ad 0d 00 00
001012d9 b8 00 00       MOV     username_len,0x0
         00 00
001012de e8 ed fd       CALL    <EXTERNAL>::printf                          int printf(ch
         ff ff
```

```
}
printf("\nType in the password: ");
__isoc99_scanf(&DAT_00102020,entered_pass);
val = strcmp(sys_pass,entered_pass);
if (val == 0) {
  puts("\nWrong password");
}
```

Fig. 5. Original Code



```
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ chmod +x crackme.bin
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ chmod +x crackme.bin
vkashir@vkashir-Standard-PC-Q35-ICH9-2009:~/Desktop/Lab-10$ ./crackme.bin
Type in your Username: vkashir

Type in a number beetween 1 and 9: 1
wlbtijs
```

Fig. 6. Printed the system generated password.

VI. Conclusion

This lab highlighted the complicated aspects of binary analysis and the vulnerabilities inherent in binary codes, especially in terms of password security. Through the tasks, we not only learned the technical aspects of using Ghidra for reverse engineering but also gained insights into the practical applications of these skills in cybersecurity. The tasks showed how to crack passwords by understanding and modifying assembly code, highlighting the risks in simplistic password generation algorithms. The ability to patch code and bypass authentication processes without the correct credentials was particularly alarming, as it exposed significant flaws in authentication verification. It is important to understand and secure binary codes in our increasingly digital world, where such skills are invaluable in protecting against cyber threats.

References

[1] KVM
https://www.linux-kvm.org/page/MainPage..

[2] Ghidra
https://github.com/NationalSecurityAgency/ghidra/blob/master/README.md

[3] OverLeaf
https://www.overleaf.com/.

[4] Github
https://en.wikipedia.org/wiki/GitHub.

[5] RED
https://kb.iu.edu/d/apum.