Daniel Chamorro, Caleb Ci, Jarrett Kinsey

Jyh Liu

CSCE 451

Mar 24, 2023

C2 Report: Team 6.0's Crack Me Challenge

Overall Summary:

This challenge involved solving another team's CrackMe challenge. Specifically, in our case, team 6.0 submitted a binary file that, when run, prompts the user for words to input. The goal of the challenge, as stated in their ReadMe file, is to get the program to output "Congratulations, you win!!!". We employed static and dynamic analysis to find any weaknesses and develop attacks to gain access to the password that cracks the puzzle.

The First Contact Impression:

Initially, based on running the .out file and looking over the very limited ReadMe file, the program prompts the user to input words. Because the goal of the program is stated to get the output "Congratulations, you win!!!", we assumed the correct keyword would crack the program, and it was our goal to figure out the word. It should be noted that this initial assumption was wrong, as after solving the CrackMe, we discovered that the inputted word did not even matter to obtain the desired output. Our next steps involved putting the binary file into a decompiler to see the contents of the program.

Divide & Conquer/Solving the Challenge:

Putting the binary file into Ghidra, we found that there were eight primary functions called by the main function, make_unique<Funroom>(), fun(), party(), validate(), check2(), KalmanFilter(), NeuralNetwork(), and Goal(). We also found four classes used, “Friend,” “Funroom,” “Init,” and “SkyNet”. After running the program 4 times, it says “Out of attempts.”, the function that does this is check2, so that is what Jarrett worked on first.

Initial functions reversed:

- check2(string param1) - Jarrett: Solved the “Out of attempts.” issue
- Fun(string param1) - Daniel
- KalmanFilter(string param1) - Caleb

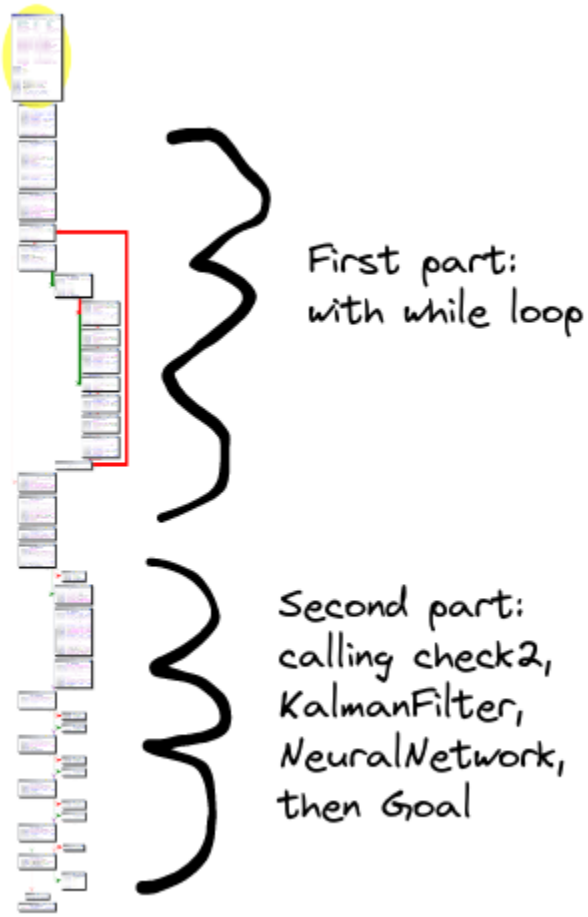
After doing the initial functions, Jarrett saw that you could reach the Goal function just without the right input easily and decided to work backwards from there.

Working Backwards:

- KalmanFilter(string param1) - Caleb: Determined that the sum of the values of the first argument needs to be 883
- check2(string param1) - Jarrett: actually solved it and make it return the correct value
- Deciphering purpose behind KalmanFilter and Fun functions. - Daniel

Summary of Such actions on Each of the D&C Steps:

Initial Reversing:



Initially looking at main, Jarrett found that there were two major areas. The first part was the while loop, and the second part was everything after it. The while loop looked complicated, for every loop it would take a line of input and, depending on if the increment variable was even or odd, it would pass that input to either the function “fun” or “party”, then add that to a vector. After the loop ends, that vector would be checked with the function “validate”.

The second part was a little more simple. It called check2 passing it the stdin line, then it called KalmanFilter passing it the first command line argument, then passing the outputs of these two called SkyNet::NeuralNetwork. Finally, it called the Goal function with the output of SkyNet::NeuralNetwork.

Running the program 4 times, it says “Out of attempts.”, so Jarrett used strace to find what it was using to store the number of attempts. It turns out that the function “check2” used these files to store the number of attempts:

- ~/.config.zzz
- ~/.config/.config.zzz

- ~/.ssh/.config.zzz
- ~/Documents/.config.zzz
- ~/Downloads/.config.zzz
- ~/Pictures/.config.zzz
- ~/tmp/attempts
- /tmp/attempts
- attempts

Check2 stores the number of attempts in these files and if any of them are 4 or greater it outputs out of attempts. To get around this Jarrett made a Makefile to remove the files so that it was easier to reset the attempts. This let the rest of the dynamic analysis continue without being blocked by the attempts.

Working Backwards:

Jarrett determined that it was easy to reach the Goal function by having any argument in the command line. This bypassed the complicated while loop and Funroom maze. Goal's input needed to be 1 for it to print the correct output and it was only dependent on the output, therefore, the only things that mattered in the main function were:

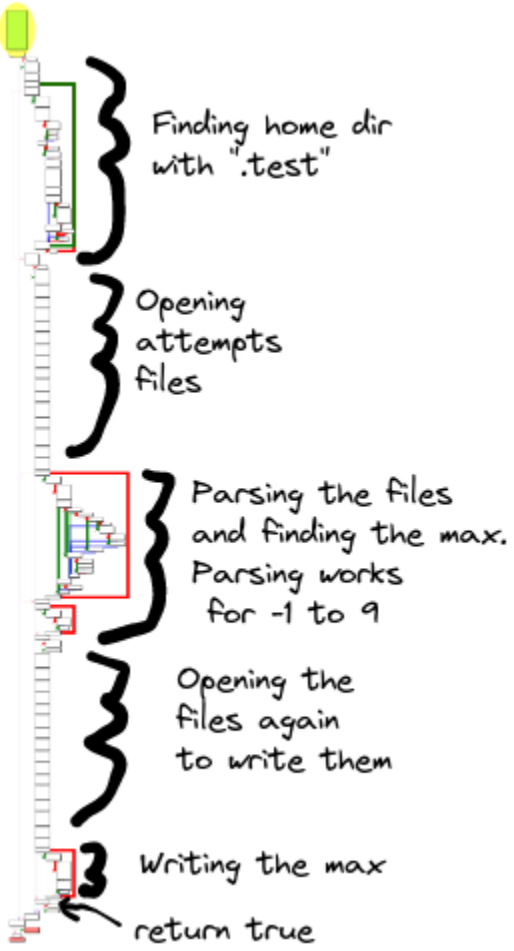
```
                /* try { // try from 00104fe1 to 00104fe5 has its CatchHandler @ 00105ice */
bVar2 = check2((basic_string)first_arg);
check2_output = (uint)bVar2;
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string
          ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)first_arg);
if (1 < argc) {
                /* try { // try from 00105032 to 00105068 has its CatchHandler @ 001051fe */
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=
            ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)argument[abi:cxx11]
             ,*(char **)(argv + 8));
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=
            ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)FalsAns[abi:cxx11],
             "win!");
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string
            (first_arg);
  kalman_output = KalmanFilter((basic_string)first_arg);
                /* try { // try from 00105090 to 001050a3 has its CatchHandler @ 001051e6 */
  kalman_output = SkyNet::NeuralNetwork(&skynet,check2_output,check2_output,kalman_output);
  Goal(kalman_output != 0);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string
            ((basic_string<char,std::char_traits<char>,std::allocator<char>> *)first_arg);
}
```

The rest of the code in the main function didn't matter because it did not control the input of Goal. Therefore, Jarrett decided to work backwards to find what inputs were needed. Using GDB debugger, Jarrett determined that when all values passed into the "SkyNet::NeuralNetwork" function were "1," the program returned the desired output. So our next step was to figure out how to get every variable, set by the other functions in main(), to return 1.

Daniel went through the process of deciphering the purpose of the Fun function and Funroom class. After analyzing the output of Ghidra the key takeaways from the fun function were that the function takes in two string parameters, modifies them, and then creates a new string object from the modified objects. Additionally, Daniel provided the team with some insight into the Kalman Filter function. Some notes provided on the Kalman Filter included that the function takes in a string object, initializes a local double to 883.0, and then calculates the difference between the ASCII values of the characters in the input string and the double variable 883.0. Daniel also provided some brief notes on the check2 function. Daniel researched tools such as GEF, BFBTester, and ANGR to solve the crack me and ultimately ended up using tools such as LLDB, and commands such as Strace, String, and Objdump to examine the contents of the read-only registers for clues as to what strings could be related to cracking the puzzle. Lastly, Daniel developed a python script to brute force a variety of inputs into the binary to find any vulnerabilities or information on how the program would react to different inputs.

Caleb took responsibility for deciphering the KalmanFilter function. Looking up the name of the function, he found that "Kalman Filter" is the name of a common estimation algorithm. Caleb had never used or come across this algorithm before so he spent some time looking into how it

worked and how it might be used to encrypt the CrackMe password. However, this was a mistake, as it turns out the name of the function was meant to mislead. This was made evident after looking into the decompiled function and translating it to C++, as the function was way too simple to be the Kalman Filter algorithm. From the translated code, the function takes a string input and returns an integer "1 – variable." Because we needed the program to return 1, this "variable" needed to be 0. We found that there was a hard-coded double value of "883.0" in the code. The program subtracted the ASCII values of each character from the input string and ran more calculations that would make "variable" 1 if 883 was not subtracted perfectly from 883. So from this, he determined that the solution would be to input a string with the sum of ASCII values as 883. This could be any string as long as all characters added up to exactly 883. He then derived the string "TestTesbE", which successfully made the function return 1.



Jarrett revisited "check2" to determine how to make it return true. The general steps of how "check2" works is it first finds the correct home directory such as "/home/<username>" trying to make the file "/home/<dir>/.test" for every directory in "/home" and check if it succeeds. After that, it opens all the attempt files. Then it loops through the files converting the string from the file into an integer and using that to find the max attempts in all the files. After that, it closes all the files. Then it checks if the max attempt is

less than 4. If it is it increments it, if it's not it prints out “Out of attempts.”. It continues by opening all the files again then writing the max attempt to all of them. Finally, it checks whether the max attempt is zero and returns true if it is otherwise false. Even though “check2” has a parameter, it never uses it and the output only depends on the attempt files and whether it can create “~/.test”. Since the max attempt is incremented, the number of attempts needs to be -1. So for “check2” to output true, all the attempt files need to contain “-1” and the process needs to create “~/.test” without an error.

Now that we were able to determine how to make “check2” and “KalmanFilter” output 1, we were able to get “Goal” to print the win string. Only the attempt files and the first command line argument mattered, the input word did not and you can type anything you want and still get the win. The only requirements are that the first command line argument’s characters add up to 883, the attempt files all contain “-1” and that “~/.test” doesn’t exist. Jarrett made a Make command to set up the file structure.

With the make file showing the commands to setup the files:

```
[username@computer Reversing]$ make win
rm -f ~/.test ~/.config.zzz ~/.config/.config.zzz ~/.ssh/.config.zzz ~/Documents/.config.zzz
~/Downloads/.config.zzz ~/Pictures/.config.zzz /tmp/attempts attempts
echo -n -1 > ~/.config.zzz
echo -n -1 > ~/.config/.config.zzz
echo -n -1 > ~/.ssh/.config.zzz
echo -n -1 > ~/Documents/.config.zzz
echo -n -1 > ~/Downloads/.config.zzz
echo -n -1 > ~/Pictures/.config.zzz
echo -n -1 > /tmp/attempts
#echo -n -1 > ~/tmp/attempts
echo -n -1 > attempts
rm -f ~/.test
# input does not matter, command line arg char value has to add up to 883
echo a | ./Program.out TestTesbE
Goal is to print 'Congratulations you win!!!'
Congratulations you win!!!
[username@computer Reversing]$
```

Just running the program showing that the input doesn’t matter:

```
[username@computer Reversing]$ ./Program.out TestTesbE
Goal is to print 'Congratulations you win!!!'
asdlkasdkfjj35hnfsda
Congratulations you win!!!
```

Fuzzing:

Jarrett and Daniel decided to try fuzzing the binary to see if it would find anything. If the Goal function was passed a 1 it would print out “Congratulations you win!!!”. Jarrett decided to use AFL to fuzz the binary. To do this he used Ghidra to patch the binary to crash when it reaches that point. By changing the call instruction that prints the win string to call 0x0 as seen in the image below. This would cause a segmentation fault which AFL could be used to fuzz for.

```
void Goal(bool param_1)

{
  basic_ostream *pbVar1;

  if (param_1) {
    pbVar1 = std::operator<<((basic_ostream *)std::cout,(basic_string *)goalString[abi:cxx11]);
    func_0x00000000(pbVar1,std::endl<char,std::char_traits<char>>);
  }
  return;
}
```

Initially, Jarrett ran AFL overnight for a couple of nights, however, it was set up wrong. The command was

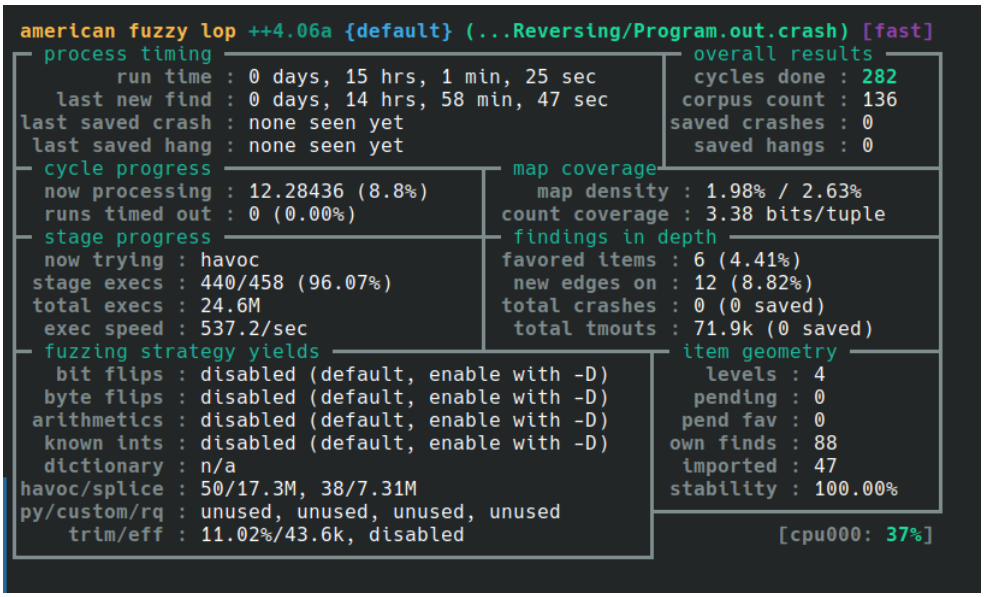
```
> AFL_SKIP_CPUFREQ=1 AFL_BENCH_UNTIL_CRASH=1 afl-fuzz -i testcase_dir/ -o findings_dir2/ -Q -
"/home/username/college/CSCE402/C2/Program.out.crash"
```

Which was only fuzzing the standard input, but the binary takes a command line argument as well. So it would never have found an input that would cause it to crash.

After Caleb found the right input for the KalmanFilter, Jarrett decided to run the fuzzing again. KalmanFilter was checking the command line argument and there were no other functions that used the arguments, so we know that we had the arguments correct. The new command line was

```
> AFL_SKIP_CPUFREQ=1 AFL_BENCH_UNTIL_CRASH=1 afl-fuzz -i testcase_dir/ -o findings_dir2/ -Q -
"/home/username/college/CSCE402/C2/Program.out.crash" TestTesbE
```

It still did not find anything after running it overnight.



In hindsight, this was expected since the check2 function was checking files instead of the stdin and AFL was only fuzzing stdin.

Summary of Key Challenges:

Key challenges in this assignment included navigating unique encryption techniques that we had not come across before. Another key challenge for this project was running the Linux x86 binary because Caleb and Daniel had ARM processors on the MacOS operating system which does not support Linux x86 through a standard virtual machine but instead requires emulation in order to run the x86 binary. Emulation for x86 Linux on silicon M1 chips currently has very little support. In order to solve this problem, Daniel tried to set up an ArchLinux-emulated VM on UTM but the VM continuously returned an error when attempting to boot. Parallels was another program that was used in an attempt to run x86 Linux but it does not support any other Linux architectures other than ARM for an M1 computer.

Reversed functions/Code Blocks:

Ghidra Decompiler Output	C++ Translation	Notes/Explanation

`undefined4 main(int argc, char **argv)`
`{`
`  bool iostream_status;`
`  byte bVar1;`
`  int iVar2;`
`  basic_ostream *output;`
`  basic_istream *input_stream;`
`  undefined4 unaff_EBX;`
`  long in_FS_OFFSET;`
`  SkyNet skynet;`
`  int index;`
`  uint local_270;`
`  uint check2_output;`
`  vector list[8];`
`  unique_ptr funroom[32];`
`  basic_string input_line[8];`
`  basic_string size[8];`
`  basic_string tmp_str[8];`
`  basic_string other_tmp_str[8];`
`  basic_string cin[98];`
`  long canary;`
 	
`  canary = *(long *)(in_FS_OFFSET + 0x28);`
`  output = std::operator<<((basic_ostream *)std::cout,`
`  "Goal is to print`
`\'Congratulations you win!!!\'");`
`  std::basic_ostream<char,`
`std::char_traits<char>>::operator<<((basic_ostream<char,`
`std::char_traits<char>> *)output,`
 	
`std::endl<char, std::char_traits<char>>);`
`  std::__cxx11::basic_string<char, std::char_traits<char>,`
`std::allocator<char>>::basic_string();`
`  /* try { // try from 00104dbf to 00104ded has its`
`CatchHandler @ 00105234 */`
`  std::getline<char, std::char_traits<char>,`
`std::allocator<char>>((basic_istream *)std::cin, input_line);`
`  std::operator|(16, 8);`
`  std::__cxx11::basic_stringstream<char,`
`std::char_traits<char>, std::allocator<char>>::`
`  basic_stringstream(cin, (_Ios_Openmode)input_line);`
`  std::`
`  vector<std::__cxx11::basic_string<char,`
`std::char_traits<char>, std::allocator<char>>,`
`std::allocator<std::__cxx11::basic_string<char,`
`std::char_traits<char>,`
`std::allocator<char>>>>::vector((vector <`
`std::__cxx11::basic_string<char, std::char_traits<char>,`
`std::allocator<char>>, std`
 	
`:`
 	
`: allocator < std::__cxx11::basic_string < char,`
`std::char_traits<char>, std::allocator < char >>>> *)list);`
`  std::__cxx11::basic_string<char, std::char_traits<char>,`
`std::allocator<char>>::basic_string();`
`  index = 0;`
`  /* try { // try from 00104e20 to 00104e80 has its`
`CatchHandler @ 001051fe */`
`  std::make_unique<Funroom>();`
`  std::unique_ptr<Funroom,`
`std::default_delete<Funroom>>::operator=((unique_ptr<Funroom,`
`std::default_delete<Funroom>> *)&funptr, funroom);`
`  std::unique_ptr<Funroom,`
	`cout << "Goal is to print`
	`'Congratulations you win!!!'" <<`
	`endl;`
	`  string other_tmp_string;`
	`  getline(cin,`
	`other_tmp_string);int main(int argc,`
	`char **argv) {`
	`  bool iostream_status;`
	`  uint check2_output;`
	`  SkyNet skynet;`
	`  uint local_270;`
	`  unique_ptr<string> funroom[32];`
	`  vector<string> list[8];`
	`  string input_line[8];`
	`  string size[8];`
	`  string tmp_str[8];`
	`  string other_tmp_str;`
	`  string cin[98];`
	
	`  cout << "Goal is to print`
	`'Congratulations you win!!!'" <<`
	`endl;`
	`  string other_tmp_string;`
	`  getline(cin, other_tmp_string);`
	
	`  bool bVar1 =`
	`validate((vector<unique_ptr<string>>&`
	`)funroom);`
	`  local_270 = (uint)bVar1;`
	
	`  // while loop and validate in`
	`ghidra output were unnecessary to`
	`solve the challenge`
	
	`  iostream_status =`
	`check2(other_tmp_str);`
	`  check2_output =`
	`(uint)iostream_status;`
	
	`  if (argc > 1) {`
	`  other_tmp_str = argv[1];`
	`  other_tmp_str += "win!";`
	`  int iVar2 =`
	`KalmanFilter(other_tmp_str);`
	`  iVar2 =`
	`skynet.NeuralNetwork(check2_output,`
	`check2_output, iVar2);`
	`  Goal(iVar2 != 0);`
	`  }`
	
	`  return 0;`
	`}`
		Main Function -
		The function starts out with a print statement that describes the goal of the program.
		
		After that, the main function has two primary parts. The first part was a large while loop that ended up being a maze. This part essentially isn't required to solve the CrackMe, so we have excluded it in our C++ translation.
		
		The second part that passed check2 the first line of stdin and passed KalmanFilter the first command line argument. The results of check2 and KalmanFilter were then passed to SkyNet::NeuralNetwork and finally the output of that was passed to the Goal function which would print out the desired win string.

```
std::default_delete<Funroom>>::~unique_ptr((unique_ptr<Funroo
m, std::default_delete<Funroom>> *)funroom);
    while (true)
    {
        /* try { // try from 00104f5c to 00104f96 has its
CatchHandler @ 001051fe */
        input_stream = std::getline<char,
std::char_traits<char>, std::allocator<char>>((basic_istream
*)cin, size, ' ');
        iostream_status =
            std::basic_ios::operator.cast.to.bool((basic_ios
*)(input_stream + *(long *)(*(long *)input_stream + -24)));
        if (!iostream_status)
            break;
        if (index % 2 == 1)
        {
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::basic_string(tmp_str);
            /* try { // try from 00104e95 to 00104e99 has its
CatchHandler @ 0010516e */
            fun((basic_string)other_tmp_str);
            /* try { // try from 00104eae to 00104eb2 has its
CatchHandler @ 00105156 */
            std::
                vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>, std::alloc
ator<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>>>::push_back((vector<std::__cxx11::basi
c_string<char, std::char_traits<char>, std::allocator<char>>,
std::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>>>

*)list,

other_tmp_str);
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>>
*)other_tmp_str);
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)tmp_str);
        }
        else
        {
            /* try { // try from 00104ee7 to 00104eeb has its
CatchHandler @ 001051fe */
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::basic_string(tmp_str);
            /* try { // try from 00104f00 to 00104f04 has its
CatchHandler @ 0010519e */
            party((basic_string)other_tmp_str);
            /* try { // try from 00104f19 to 00104f1d has its
CatchHandler @ 00105186 */
            std::
                vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>, std::alloc
ator<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>>>::push_back((vector<std::__cxx11::basi
c_string<char, std::char_traits<char>, std::allocator<char>>,
std::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>>>

*)list,

other_tmp_str);
```

```
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>>
*)other_tmp_str);
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)tmp_str);
       }
       index = index + 1;
   }
   std::
       vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>,
std::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>>>::vector((vector <
std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>, std

:

: allocator < std::__cxx11::basic_string < char,
std::char_traits<char>, std::allocator < char >>>> *)funroom,

list);
   /* try { // try from 00104fa1 to 00104fa5 has its
CatchHandler @ 001051b6 */
   bVar1 = validate((vector)funroom);
   local_270 = (uint)bVar1;
   std::
       vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>,
std::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>>>::~vector((vector<std::__cxx11::basic_
string<char, std::char_traits<char>, std::allocator<char>>,
std ::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>>>

*)funroom);
   /* try { // try from 00104fd2 to 00104fd6 has its
CatchHandler @ 001051fe */
   std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>::basic_string(other_tmp_str);
   /* try { // try from 00104fe1 to 00104fe5 has its
CatchHandler @ 001051ce */
   iostream_status = check2((basic_string)other_tmp_str);
   check2_output = (uint)iostream_status;
   std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>>
*)other_tmp_str);
   if (1 < argc)
   {
       /* try { // try from 00105032 to 00105068 has its
CatchHandler @ 001051fe */
       std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::operator=((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)argument
[abi:cxx11], argv[1]);
       /* first argument needs to be "win!" */
       std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::operator+=((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)FalsAns
[abi:cxx11],
```

```
"win!");
       std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::basic_string(other_tmp_str);
       iVar2 = KalmanFilter((basic_string)other_tmp_str);
       /* try { // try from 00105090 to 001050a3 has its
CatchHandler @ 001051e6 */
       iVar2 = SkyNet::NeuralNetwork(&skynet, check2_output,
check2_output, iVar2);
       Goal(iVar2 != 0);
       std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>>
*)other_tmp_str);
   }
   else
   {
       unaff_EBX = 0;
   }
   std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)size);
   std::
       vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>,
std::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>>>::~vector((vector<std::__cxx11::basic_
string<char, std::char_traits<char>, std::allocator<char>>,
std ::allocator<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>>>

*)list);
   std::__cxx11::basic_stringstream<char,
std::char_traits<char>, std::allocator<char>>::
       ~basic_stringstream((basic_stringstream<char,
std::char_traits<char>, std::allocator<char>> *)cin);
   std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)input_line);
   if (1 < argc)
   {
       unaff_EBX = 0;
   }
   if (canary != *(long *)(in_FS_OFFSET + 0x28))
   {
       /* WARNING: Subroutine does not return */
       __stack_chk_fail();
   }
   return unaff_EBX;
}
```

```
/* check2(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >) */

bool check2(basic_string param_1)

{
   long lVar1;
   char *path;
   size_t read_size;
   bool ret_output;
   long in_FS_OFFSET;
   allocator alloc;
   int num_attempts_max;
   int index2;
   int num_attempts_curr;
   int index3;
```

```
// Jarrett
int check2(string not_used)
{
   // .test, looks for home directory
by traversing folders in /home
   // and trying to make a file
called `.test` if it can it uses that
directory name
   // This gets the home directory of
the current user
   DIR *dirFP = opendir("/home");
   if (dirFP == nullptr)
   {
       return 0;
   }
```

Check2 Function
Jarrett -
Check2 stores the number of attempts in these files and if any of them are 4 or greater it outputs out of attempts. Specifically, check2 finds the correct home directory such as

```
   int index;
   int local_e0;
   int file_pointer;
   DIR *dir_fp;
   dirent *directory;
   basic_string local_c8[8];
   basic_string d_name[8];
   basic_string local_88[8];
   FILE *files[5];
   char file_attempts[2];

   lVar1 = *(long *)(in_FS_OFFSET + 0x28);
   dir_fp = opendir("/home");
   if (dir_fp == (DIR *)0x0)
   {
       ret_output = false;
   }
   else
   {
       std::allocator<char>::allocator();
       /* try { // try from 00103e63 to 00103e67 has its
CatchHandler @ 00104851 */
       std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::

basic_string<std::allocator<char>>((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_c8, "",
&alloc);
       std::allocator<char>::~allocator((allocator<char>
*)&alloc);
       /* try { // try from 0010403e to 001040af has its
CatchHandler @ 00104a8b */
       while (directory = readdir(dir_fp), directory !=
(dirent *)0x0)
       {
           std::allocator<char>::allocator();
           /* try { // try from 00103eaa to 00103eae has its
CatchHandler @ 00104872 */
           std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::

basic_string<std::allocator<char>>((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)d_name,

directory->d_name, &alloc);
           std::allocator<char>::~allocator((allocator<char>
*)&alloc);
           ret_output = std::operator==(d_name, ".");
           if ((ret_output) || (ret_output =
std::operator==(d_name, ".."), ret_output))
           {
               ret_output = true;
           }
           else
           {
               ret_output = false;
           }
           if (!ret_output)
           {
               /* try { // try from 00103f2a to 00103f2e has
its CatchHandler @ 001048b7 */
               std::operator+((char *)files, (basic_string
*)"/home/");
               /* try { // try from 00103f44 to 00103f48 has
its CatchHandler @ 0010488d */
               std::operator+(local_88, (char *)files);
               std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::operator=((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)d_name,
```

```
   string test_path;
   dirent *directory;
   while ((directory =
readdir(dirFP)) != nullptr)
   {
       string dir_name =
directory->d_name;
       if (!(dir_name == "." ||
dir_name == ".."))
       {
           test_path = "/home";
           test_path += "/";
           test_path += dir_name +
"/";

           int fildes =
creat((test_path + ".test").c_str(),
1);
           if (-1 < fildes)
           {
               close(fildes);

remove(test_path.c_str());
           }
       }
   }
   closedir(dirFP);
   if (test_path == "")
   {
       return 0;
   }

   string dir_name = test_path;
   // read files and get max attempt
   FILE *files[8];

   files[0] = fopen((dir_name +
".config.zzz").c_str(), "a+");
   files[1] = fopen((dir_name +
".config/.config.zzz").c_str(),
"a+");
   files[2] = fopen((dir_name +
".ssh/.config.zzz").c_str(), "a+");
   files[3] = fopen((dir_name +
"Documents/.config.zzz").c_str(),
"a+");
   files[4] = fopen((dir_name +
"Downloads/.config.zzz").c_str(),
"a+");
   files[5] = fopen((dir_name +
"Pictures/.config.zzz").c_str(),
"a+");
   files[6] = fopen("attempts",
"a+");
   files[7] = fopen((dir_name +
"/tmp/attempts").c_str(), "a+");

   // get max attempt
   int max_attempt = -1;
   int curr_attempt;
   char read_buf[2];
   for (int i = 0; i < 8; i++)
   {
       if (files[i] != nullptr)
       {
           fseek(files[i], 0, 0);
           int size = fread(read_buf,
1, 2, files[i]);
           if (size == 2)
           {
```

"/home/<username>" trying to make the file "/home/<dir>/.test" for every directory in "/home" and checks if it succeeds. After that, it opens all the attempt files. Then it loops through the files converting the string from the file into an integer and using that to find the max attempts in all the files. After that, it closes all the files. Then it checks if the max attempt is less than 4. If it is it increments it, if it's not it prints out "Out of attempts.". It continues by opening all the files again then writing the max attempt to all of them. Finally, it checks whether the max attempt is zero and returns true if it is otherwise false.

```
local_88);
                std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
                std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)files);
                /* try { // try from 00103f8f to 00103f93 has
its CatchHandler @ 001048b7 */
                std::operator+((basic_string *)files, (char
*)d_name);
                path = (char
*)std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>::c_str();
                /* try { // try from 00103fa8 to 00103ffc has
its CatchHandler @ 001048a2 */
                file_pointer = creat(path, 1);
                if (-1 < file_pointer)
                {
                    close(file_pointer);
                    path = (char *)std::__cxx11::
                        basic_string<char,
std::char_traits<char>, std::allocator<char>>::c_str();
                    remove(path);
                    std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::operator=((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_c8,

d_name);
                }
                std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)files);
            }
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)d_name);
        }
        closedir(dir_fp);
        ret_output = std::operator==(local_c8, "");
        if (ret_output)
        {
            ret_output = false;
        }
        else
        {
            /* start reading files */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                c_str();
            /* try { // try from 001040cc to 001040d0 has its
CatchHandler @ 001048d2 */
            files[0] = fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 001040f9 to 001040fd has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                c_str();
            /* try { // try from 0010411a to 0010411e has its
```

```
                if (read_buf[0] == '-'
&& read_buf[1] == '1')
                {
                    curr_attempt = -1;
                }
                else
                {
                    curr_attempt = 0;
                }
            }
            else if (size < 3)
            {
                if (size == 0)
                {
                    curr_attempt = 0;
                }
                else if (size == 1)
                {
                    if (read_buf[0] -
'0' < 10)
                    {
                        curr_attempt =
read_buf[0] - '0';
                    }
                    else
                    {
                        curr_attempt =
0;
                    }
                }
            }

            if (max_attempt <
curr_attempt)
            {
                max_attempt =
curr_attempt;
            }
        }
    }
    for (int i = 0; i < 8; i++)
    {
        if (files[i] != nullptr)
        {
            fclose(files[i]);
        }
    }

    // check max attempts
    if (max_attempt < 4)
    {
        max_attempt = max_attempt + 1;
    }
    else
    {
        cout << "Out of attempts.\n";
    }
    // write new attempt
    files[0] = fopen((dir_name +
".config.zzz").c_str(), "w+");
    files[1] = fopen((dir_name +
".config/.config.zzz").c_str(),
"w+");
    files[2] = fopen((dir_name +
".ssh/.config.zzz").c_str(), "w+");
    files[3] = fopen((dir_name +
"Documents/.config.zzz").c_str(),
"w+");
    files[4] = fopen((dir_name +
"Downloads/.config.zzz").c_str(),
```

```
CatchHandler @ 001048ea */
            files[1] = fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 00104147 to 0010414b has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
            /* try { // try from 00104168 to 0010416c has its
CatchHandler @ 00104902 */
            files[2] = fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 00104195 to 00104199 has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
            /* try { // try from 001041b6 to 001041ba has its
CatchHandler @ 0010491a */
            files[3] = fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 001041e3 to 001041e7 has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
            /* try { // try from 00104204 to 00104208 has its
CatchHandler @ 00104932 */
            files[4] = fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 00104231 to 00104235 has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
            /* try { // try from 00104252 to 00104256 has its
CatchHandler @ 0010494a */
            fopen(path, "a+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            std::allocator<char>::allocator();
            /* try { // try from 0010428e to 00104292 has its
CatchHandler @ 00104977 */
            std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::

basic_string<std::allocator<char>>((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88,
                                                    "attempts",
&alloc);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
```

```
"w+");
   files[5] = fopen((dir_name +
"Pictures/.config.zzz").c_str(),
"w+");
   files[6] = fopen("attempts",
"w+");
   files[7] = fopen((dir_name +
"tmp/attempts").c_str(), "w+");
   string s;
   for (int i = 0; i < 8; i++)
   {
       if (files[i] != nullptr)
       {
           fseek(files[i], 0, 0);
           char s = (char)max_attempt
+ '0';
           fwrite(&s, 1, 1,
files[i]);
           fclose(files[i]);
       }
   }
   if (max_attempt == 0)
   {
       return 1;
   }
   else
   {
       return 0;
   }
}
```

```
            c_str();
        /* try { // try from 001042af to 001042b3 has its
CatchHandler @ 00104962 */
        fopen(path, "a+");
        std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
        std::allocator<char>::~allocator((allocator<char>
*)&alloc);
        std::allocator<char>::allocator();
        /* try { // try from 001042fa to 001042fe has its
CatchHandler @ 001049a7 */
        std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::

basic_string<std::allocator<char>>((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88,

"/tmp/attempts", &alloc);
        path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
            c_str();
        /* try { // try from 0010431b to 0010431f has its
CatchHandler @ 00104992 */
        fopen(path, "a+");
        std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
        std::allocator<char>::~allocator((allocator<char>
*)&alloc);
        num_attempts_max = -1;
        for (index2 = 0; index2 < 8; index2 = index2 + 1)
        {
            if (files[index2] != (FILE *)0x0)
            {
                fseek(files[index2], 0, 0);
                /* try { // try from 001043ae to 001044fd
has its CatchHandler @ 00104a8b */
                read_size = fread(file_attempts, 1, 2,
files[index2]);
                local_e0 = (int)read_size;
                if (local_e0 == 2)
                {
                    if ((file_attempts[0] == '-') &&
(file_attempts[1] == '1'))
                    {
                        num_attempts_curr = -1;
                    }
                    else
                    {
                        num_attempts_curr = 0;
                    }
                }
                else if (local_e0 < 3)
                {
                    if (local_e0 == 0)
                    {
                        num_attempts_curr = 0;
                    }
                    else if (local_e0 == 1)
                    {
                        if ((int)file_attempts[0] - L'0' U
< 10)
                        {
                            num_attempts_curr =
file_attempts[0] + -48;
                        }
                        else
```

```
                            {
                                num_attempts_curr = 0;
                            }
                        }
                    }
                    if (num_attempts_max < num_attempts_curr)
                    {
                        num_attempts_max = num_attempts_curr;
                    }
                }
            }
            for (index3 = 0; index3 < 8; index3 = index3 + 1)
            {
                if (files[index3] != (FILE *)0x0)
                {
                    fclose(files[index3]);
                }
            }
            /* check if max attempts in the files < 4 or not
*/
            if (num_attempts_max < 4)
            {
                num_attempts_max = num_attempts_max + 1;
            }
            else
            {
                std::operator<<((basic_ostream *)std::cout,
"Out of attempts.\n");
            }
            /* start writing file */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                   c_str();
            /* try { // try from 0010451a to 0010451e has its
CatchHandler @ 001049c2 */
            files[0] = fopen(path, "w+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 00104547 to 0010454b has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                   c_str();
            /* try { // try from 00104568 to 0010456c has its
CatchHandler @ 001049da */
            files[1] = fopen(path, "w+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 00104595 to 00104599 has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
            path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                   c_str();
            /* try { // try from 001045b6 to 001045ba has its
CatchHandler @ 001049f2 */
            files[2] = fopen(path, "w+");
            std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
            /* try { // try from 001045e3 to 001045e7 has its
CatchHandler @ 00104a8b */
            std::operator+(local_88, (char *)local_c8);
```

```
          path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
          /* try { // try from 00104604 to 00104608 has its
CatchHandler @ 00104a0a */
          files[3] = fopen(path, "w+");
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
          /* try { // try from 00104631 to 00104635 has its
CatchHandler @ 00104a8b */
          std::operator+(local_88, (char *)local_c8);
          path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
          /* try { // try from 00104652 to 00104656 has its
CatchHandler @ 00104a1f */
          files[4] = fopen(path, "w+");
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
          /* try { // try from 0010467f to 00104683 has its
CatchHandler @ 00104a8b */
          std::operator+(local_88, (char *)local_c8);
          path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
          /* try { // try from 001046a0 to 001046a4 has its
CatchHandler @ 00104a34 */
          fopen(path, "w+");
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
          std::allocator<char>::allocator();
          /* try { // try from 001046dc to 001046e0 has its
CatchHandler @ 00104a5e */
          std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::

basic_string<std::allocator<char>>((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88,
                                                 "attempts",
&alloc);
          path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
          /* try { // try from 001046fd to 00104701 has its
CatchHandler @ 00104a49 */
          fopen(path, "w+");
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_88);
          std::allocator<char>::~allocator((allocator<char>
*)&alloc);
          /* try { // try from 00104739 to 0010473d has its
CatchHandler @ 00104a8b */
          std::operator+(local_88, (char *)local_c8);
          path = (char *)std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::
                 c_str();
          /* try { // try from 0010475a to 0010475e has its
CatchHandler @ 00104a76 */
          fopen(path, "w+");
          std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
```

```
std::char_traits<char>, std::allocator<char>> *)local_88);
          /* Write max attempts to all the files */
          for (index = 0; index < 8; index = index + 1)
          {
              if (files[index] != (FILE *)0x0)
              {
                  fseek(files[index], 0, 0);
                  alloc = (allocator)((char)num_attempts_max
+ '0');
                  /* try { // try from 001047e2 to 001047fb
has its CatchHandler @ 00104a8b */
                  fwrite(&alloc, 1, 1, files[index]);
                  fclose(files[index]);
              }
          }
          /* returns true only if max attempts is 0 */
          if (num_attempts_max == 0)
          {
              ret_output = true;
          }
          else
          {
              ret_output = false;
          }
      }
      std::__cxx11::basic_string<char,
std::char_traits<char>,
std::allocator<char>>::~basic_string((basic_string<char,
std::char_traits<char>, std::allocator<char>> *)local_c8);
  }
  if (lVar1 == *(long *)(in_FS_OFFSET + 0x28))
  {
      return ret_output;
  }
  /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

```
/* KalmanFilter(std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >) */

int KalmanFilter(basic_string param_1)

{
  bool has_ended;
  char *curr_string;
  int __x;
  undefined4 in_register_0000003c;
  long in_FS_OFFSET;
  double extraout_XMM0_Qa;
  double extraout_XMM0_Qa_00;
  double dVar1;
  undefined8 begin;
  undefined8 end;
  double local_30;
  undefined8 local_28;
  long canary;
  undefined8 *begin_ref;

  local_28 = CONCAT44(in_register_0000003c, param_1);
  canary = *(long *)(in_FS_OFFSET + 0x28);
  local_30 = 883.0;
  begin = std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::begin();
  end = std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>>::end();
```

```
int KalmanFilter(std::string str)
{
  double sum = 883.0;
  auto it_begin = str.begin();
  auto it_end = str.end();

  // Looping through the string
  while (it_begin != it_end)
  {
      // Subtracting the ASCII value
of each character in the string from
sum
      sum -=
static_cast<double>(*it_begin);
      ++it_begin;
  }

  // Calculating the absolute value
of the sum
  double dVar3 = std::abs(sum);
  double dVar4 = std::abs(sum);

  // Calculating a threshold value
  dVar3 = ceil(dVar3 / (dVar4 +
1.0));

  // Returning the result
  return 1 -
static_cast<int>(dVar3);
```

Kalman filter - The function subtracts ASCII values of each character from the input string from the hard coded double “sum” with value 883.0. The absolute value is taken after the subtraction and these variables are divided by each other, plus 1 in the denominator, and rounded up.

Essentially, because it divides the absolute values and rounds up,

```
   while (true)
   {
       begin_ref = &begin;
       has_ended = __gnu_cxx::operator!=((__normal_iterator
*)begin_ref, (__normal_iterator *)&end);
       __x = (int)begin_ref;
       if (!has_ended)
           break;
       curr_string = (char *)__gnu_cxx::
           __normal_iterator<char *,
std::__cxx11::basic_string<char, std::char_traits<c har>,
std::allocator<char>>>::operator*((__normal_iterator<char *,
std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char>>>

*)&begin);
       local_30 = local_30 - (double)(int)*curr_string;
       __gnu_cxx::
           __normal_iterator<char *,
std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<ch ar>>>::operator++((__normal_iterator<char
*, std::__cxx11::basic_string<char, std::char_traits<char>,
std ::allocator<char>>>

*)&begin);
   }
   std::abs(__x);
   std::abs(__x);
   dVar1 = ceil(extraout_XMM0_Qa / (extraout_XMM0_Qa_00 +
1.0));
   if (canary != *(long *)(in_FS_OFFSET + 0x28))
   {
       /* WARNING: Subroutine does not return */
       __stack_chk_fail();
   }
   return 1 - (int)dVar1;
}
```

```
}
```

if 883 is not subtracted perfectly from “sum,” the final variable subtracted by 1 in the return statement will be 1, so 1–1=0 will be returned.

Conversely, If all the characters in the input string add up to 883, “dVar3” will 0, and then 1 is returned.

```
void Goal(bool param_1)
{
 basic_ostream *this;
  if (param_1) {
   this = std::operator<<((basic_ostream
*)std::cout,(basic_string *)goalString[abi:cxx11]);

std::basic_ostream<char,std::char_traits<char>>::operator<<
             ((basic_ostream<char,std::char_traits<char>>
*)this,
              std::endl<char,std::char_traits<char>>);
 }
 return;
}
```

```
void Goal(bool param_1) {
 if (param_1) {
   cout << "Congratulations you
win!!!" << endl;
 }
}
```

Goal -
Just prints out “Congratulations you win!!!” if it’s parameter is true.

SkyNet::NeuralNetwork -
Jarrett found that passing all “1” to every parameter of the function produced the desired win output, so we focused on figuring out how to get each variable to return 1 to crack the challenge.

Validation/Verification:

Validation and verification were implemented by running the binary on a Linux x86 system with the argv being a string that sums to 883.0, in our case, “TestTesbE,” and then entering any input such as asdf while also making sure that the number of attempts in each of the files created does not go over 4. If the maximum number in the files created by the check2 function goes over four, then the program locks out the user and states that no more attempts are allowed. After entering a string in which the characters added up to 883 as an argument and then typing in any string following that, the program returns the desired congratulations message as expected.
Additionally, we ran our reversed files as well to make sure that they behaved in the same manner as the provided binary to ensure that we had reversed the binary correctly.
We checked that when the attempt files contain numbers -1 to 10 that both binaries return the same thing for “check2”. Additionally, for the command line arguments we checked strings that added up to 416 (Test), 883 (TestTesbE, TestTesaF), 897 (TestTestA). As expected both binaries only printed out the win string when the attempt files were -1 and the command line argument added up to 883.

The given binary:

```
[username@computer Reversing]$ ./Program.out TestTesbE
Goal is to print 'Congratulations you win!!!'
asdlkasdkfjj35hnfsda
Congratulations you win!!!
```

Our reversed binary:

```
[username@computer Reversing]$ make setup-files
rm -f ~/.test ~/.config.zzz ~/.config/.config.zzz ~/.ssh/.config.zzz ~/Documents/.config.zzz
~/Downloads/.config.zzz ~/Pictures/.config.zzz /tmp/attempts attempts
echo -n -1 > ~/.config.zzz
echo -n -1 > ~/.config/.config.zzz
echo -n -1 > ~/.ssh/.config.zzz
echo -n -1 > ~/Documents/.config.zzz
echo -n -1 > ~/Downloads/.config.zzz
echo -n -1 > ~/Pictures/.config.zzz
echo -n -1 > /tmp/attempts
#echo -n -1 > ~/tmp/attempts
echo -n -1 > attempts
rm -f ~/.test
[username@computer Reversing]$ ./src/main TestTesbE
Goal is to print 'Congratulations you win!!!'
asdfjhasdfhjesaf
Congratulations you win!!!
```

Task Summary:

In order to successfully crack the challenge, our team examined the reversed output of Ghidra and ran the gdb debugger to explore the behavior of functions, and we found specific inputs for NeuralNetwork would trigger the win output. After a thorough examination of the disassembled code, the team was able to decipher how to get every function to return the required value. This included piecing together a phrase whose character's ASCII values added up to 883, which needed to be passed into the program as an argument. We also found that there were only a limited number of attempts to guess the correct configuration before being locked out, and that the inputted word did not matter to obtain the win message. In the end, the team successfully cracked the challenge and managed to print "Congratulations, you win!!!".