Thanks to your work, the field agents were able to locate and obtain a copy of the server. A Dynamic Linked Library was found alongside the server, which may serve a significant purpose regarding the cryptography involved in the communication. Please prepare your testbed once again for analysis.

Your final goal is to obtain the passwords needed to decrypt the archive.
Skip to Step 2 if you are using EmuBox

Step 1 – Connectivity to the testbed

NOTE: Ensure the laptop’s wireless interface is turned off.

1. Connect the laptop to the testbed with the provided Ethernet cable.

Windows users:

2. Click on the Windows Logo > Control Panel > View Network status and tasks under Network and Internet. (Windows 10: ⊞ + “network status”)

3. In the new window, click on Local Area Connection.

4. Click on Details and make sure the IPv4 Address field is 192.168.1.x, where x is an integer (shown above).

5. Open Remote Desktop Connection on your computer by clicking on the Windows Logo > Search programs and files > type Remote Desktop Connection. (Windows 10: ⊞ + “remote desktop”)

a. Click on the 
*Show Options* drop down menu and under the *Display* tab select the following settings:

![Remote Desktop Connection with settings](image1.png)

b. In the *Computer* field type in the Remote Computer address provided on your ticket.

<IP address>::<port>

Example: 192.168.1.xxx:xxxx

![Remote Desktop Connection](image2.png)

c. Click on *Connect*.

d. Click *Yes* on the security prompt.

e. After ~10 seconds you will see the desktop.
OSX users:

2. Click on the **Apple logo** on the top left corner > **System Preferences**.
3. Click on **Network**.
4. Verify that the Ethernet adapter IP Address matches **192.168.1.x**, where x is an integer (shown below).

5. Open **Microsoft Remote Desktop** (can be obtained from the **APP Store**) by holding **Command + Space** keys and typing **Microsoft Remote Desktop**.
6. Click on **New**.
7. Enter the **Remote Computer** address in the **PC name** field.

    `<IP address>:`:<port>

Example: 192.168.1.xxx:xxxx

8. Close the window, double click the new entry under **My Desktops**.

9. After ~10 seconds you will see the desktop.
Step 2 – Patch the Client

Using the information you gathered from performing a static analysis of the Client binary, you will now patch the Client to use localhost in the communication protocol instead.

1. **Start** the *HexEdit* application from the Task bar menu.

2. **Open** the *Client.exe* binary.

3. **Enable editing** by clicking the *Allow Changes* icon in the menu:

   ![Allow Changes Icon](image)

4. In the *Jump Hex* text box, type **64B4** and **push Enter**:

   ![Jump Hex Text Box](image)

5. This will move the cursor to the location of the IP address. You should **modify** the series of hex values to read **127.0.0.1** (*localhost*).

   **Before:**
   
   | 31 39 32 2E | 31 2E 31 2E | 32 00 00 | r · · · 92.1.1.2 |

   **After:**
   
   | 31 32 37 2E | 30 2E 30 2E | 31 00 00 | r · · · 127.0.0.1 |

6. **Save** the binary.

7. **Close** the application.
Step 3 – Using IDA Pro’s Debugger

IDA Pro is a combination of disassembler and debugger, facilitating both static and dynamic analysis. It is a powerful tool commonly used by professionals to perform malware analysis and much more. Needless to say, definitely an excellent asset to add to your knowledge of tools.

1. Start the **IDA Pro** application from the Task Bar menu.
2. Click the **New** button to begin a new disassembly.
   a. Select the **PE Dynamic Library** option and click **Ok**.
   b. Your target for analysis is **ServerLibrary.dll**. Browse to it’s location on the **Desktop** and open it.
3. Click through the loading wizard by selecting **Next, Next**, and then **Finish**.
4. Select **Yes** on the confirmation dialog.
5. Once IDA finishes loading the DLL, you will be presented with the entry point:

   ```
   ; Attributes: think
   ; BOM; _stdcall _thisEntryPoint[INSTANCE hinstDLL, WORD fUnicode, LPVOID lpReserved]
   public _stdcall _thisEntryPoint
   public _thisEntryPoint
   extern _thisEntryPoint proc near
   jmp sub_18F555A0
   _thisEntryPoint endp
   ```

3. You will now instruct IDA to use the server executable when starting a debugging process on the DLL. From IDA’s menu, select **Debugger**, and then **Process Options**...
4. Change the **Application path** to point to **ServerLibraryApp.exe** instead of the DLL:

   ![Application setup dialog](image)
5. Click **Ok** to confirm the change.

6. From the menu again, select **Debugger**, and then **Start Process** to start the debugger.
   a. Select **Yes** in the warning dialog that opens.
   b. Select **Ok** in the next warning dialog about a software breakpoint.
   c. Click the **Play** button near to top of the window to continue execution.
   d. Select **Change Exception Definition** on the dialog about exception handling.
   e. Check **Pass to Application** and click **Ok**.
   f. Click **Yes**.

ServerLibraryApp.exe should now be running and listening for an incoming connection:

7. Start a new **Windows Command Prompt** by clicking the icon in the bottom menu.

8. Using the command prompt, you want to navigate to the location of the client and then run the program. In your command prompt, type the following:

```
cd C:\Users\cs5357\Desktop
Client.exe
```

9. Notice two things here: the server accepted the connection while being debugged and the client terminated due to an insufficient number of parameters. What is the correct usage for the client?

___________________________________________________________________

10. Using your **command prompt**, try running **Client.exe** again but this time with **1111** as your first password and **2222** as your second password.

You should receive a different message this time; however, it seems nothing happened and the connection closed (**probably wrong passwords**):

```
Initializing connection...Connected
Connection closed!
```

11. Stop the debugging process using the **Stop** button near the top of the window in IDA Pro.
Step 4 – Using Static Analysis to Set Breakpoints

Recall from the previous workshop that static analysis is the process of analyzing a binary to determine its functionality without actual execution of the program. You will use this process to identify locations in the binary to set breakpoints to assist your dynamic analysis.

12. Using the Names window, identify the two functions that are responsible for validating the passwords.
   a. __________________________________________
   b. __________________________________________

13. Double click on the first function name.

14. Follow the jump instruction to the function implementation by double clicking on the address label in yellow (you may have a slightly different hex address):

   jmp  loc_550F18F0

IDA is smart enough to identify method signatures from commonly used libraries, making your life as a reverse engineer much easier.

15. Notice the call to the atoi method.

   atoi - converts a string to an integer.
   (i.e. “6969” => 6969 or “abc” => 0)

   Set a breakpoint on the push eax instruction by first clicking on the instruction, and then pushing the F2 key.

A breakpoint tells the debugger to halt execution. This allows you to examine the current state of the program.
Step 5 – Dynamic Analysis of a Function

Dynamic analysis allows you to analyze a program’s functional behavior and monitor its interaction with the system’s memory through execution. This is quite useful when a program is too complex or purposely obfuscated to perform a static analysis.

16. Start the debugger as you did in Step 6.

17. Using your command prompt, run the Client.exe as you did in Step 8 using 1111 and 2222 as your passcodes.

18. Execution should halt at the breakpoint you set (highlighted in purple):

19. The eax register contains a pointer to the string being passed as a parameter to the atoi function.

   Click on the General registers window so it is the active window and mouse over the value (shown in the red box below).

   Write down the string you see (reading downward until you see a 0):__________________________

20. Click on the IDA View-A window so that is it the active window.

21. Continue the program’s execution line-by-line by using the Step Over (F8 key) command. Stop after executing the mov [ebp+var_8], eax instruction (the blue line should be below the mov instruction).

22. Ebp+var_8 contains the result of the atoi function. What value does it contain? (mouse over Ebp+var_8 in the window)

23. The following xor instruction clears the eax register. After executing the instruction, you will notice the value in eax is 0.
24. Next, ebp+var_8 is compared (the cmp instruction) against a certain constant value. What is the value of the constant? (the ‘h’ denotes that it is a hex value)

25. Convert the value from Step 24 into decimal using the calculator in IDA. To do this, click on the constant and use the menu option View > Calculator.

Write it down:

26. Congratulations! You have your first password.

27. Now stop the debugger.

28. Clear the breakpoint you set by pushing F2 on the instruction again.

29. Repeat the process to obtain the second password (starting from Step 12 again).

Write it down:

30. Run the client with the correct passwords to obtain the secret password to the archive on the desktop (UBER_1.rar):

Uber Challenge: Discover the password for the second archive.
(hint: there may be an alternative usage of the client binary)

Thanks for attending!