This assignment will be done in groups of 2-3.

Answer the following questions:

1. Think of a time when you have reversed something (it doesn’t have to be code).
   a. What was your motivating factor?
   
   b. What steps did you take?
   
   c. How long did it take you?

2. You and your friends have developed a new video game, “Security Wars”. How are you going to protect your product? Why?

3. After dedicating your entire life to creating a human teleporter, you have thought of a way to build one. How are you going to protect this? Why?

4. Your music band just cut a new album and you’ve secured copyright, but you’re still worried about unauthorized duplication. What do you do?

5. You have a Spotify® account and you own a restaurant. Should it be legal to play music from Spotify® for your patrons? Why or why not?
6. The following is similar to what you may see when you use open source software. Describe the license in your own words.

```php
<?php

/* GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

/** Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.*> */

Joomla! derives from copyrighted works licensed under the GNU General Public License. This version has been modified pursuant to the GNU General Public License as of September 15, 2005, and as distributed,
It includes or is derivative of works licensed under the GNU General Public License or other free or open source software licenses. Please see the CREDITS.php for a non-exhaustive list of contributors and copyright holders. A full text version of the GNU GPL version 2 can be found in the LICENSE.php file. A full text version of the other licenses that Joomla! is derivative of or includes can be found in LICENSES.php.
In-Class Assignment – Low Level Software

This assignment will be done in groups of 2-3.

Answer the following questions:

1. The following three snippets accomplish the same task.

   a. Using a high-level language, write what these snippets are accomplishing.

   ```
   a
   mov eax, [0x00B0 0080]
   xor eax, 1
   jz loc_1
   mov ecx, 0
   END
   loc_1:
   mov ecx, 1
   END
   ```

   ```
   b
   mov eax, 0
   mov ebx, 1
   xor eax, ebx
   jnz loc_1
   mov ecx, 1
   END
   loc_1:
   mov ecx, 0
   END
   ```

   ```
   c
   mov edx, 0x00B0 0080
   mov eax, [edx]
   xor eax, [0x00B0 0084]
   jz loc_1
   mov ecx, 0
   END
   loc_1:
   mov ecx, 1
   END
   ```

Data Memory

<table>
<thead>
<tr>
<th>0x00B0 007C</th>
<th>0x00B0 0080</th>
<th>0x00B0 0084</th>
<th>0x00B0 0088</th>
<th>0x00B0 008C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 000A</td>
<td>0x0000 0000</td>
<td>0x0000 0001</td>
<td>0x0000 0004</td>
<td>0x0000 000B</td>
</tr>
</tbody>
</table>
b. For each line in snippets (b) and (c) write the type of the operand being used (use (r) for register, (i) for immediate, (m) for memory address).

c. Based only on this context, rank the above snippets by latency (1 is the longest, 3 is shortest). Explain your answers.

1. ____

2. ____

3. ____

2. Using a high-level language, write what these snippets are accomplishing (use the data memory from the previous question).

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov eax, 0x00B0 007C</td>
<td></td>
</tr>
<tr>
<td>mov ecx, 0</td>
<td></td>
</tr>
<tr>
<td>loc_1:</td>
<td></td>
</tr>
<tr>
<td>add eax, 4</td>
<td></td>
</tr>
<tr>
<td>add edx, [eax]</td>
<td></td>
</tr>
<tr>
<td>add ecx, 1</td>
<td></td>
</tr>
<tr>
<td>cmp ecx, 4</td>
<td></td>
</tr>
<tr>
<td>jz loc_2</td>
<td></td>
</tr>
<tr>
<td>jmp loc_1</td>
<td></td>
</tr>
<tr>
<td>mov ebx, 0</td>
<td></td>
</tr>
<tr>
<td>mov eax, 1</td>
<td></td>
</tr>
<tr>
<td>add ebx, eax</td>
<td></td>
</tr>
<tr>
<td>shl eax, 1</td>
<td></td>
</tr>
<tr>
<td>add ebx, eax</td>
<td></td>
</tr>
<tr>
<td>shl eax, 1</td>
<td></td>
</tr>
<tr>
<td>add ebx, eax</td>
<td></td>
</tr>
</tbody>
</table>
```
This assignment will be done in groups of 2-3.

Answer the following questions:

1. The following is a snapshot of the system before execution. For each of the following, show the contents (by scratching out/rewriting values) of the stack and registers (b) after executing the last instruction shown in the code snippet (a).

   1.)

   ![Data Stack Diagram]

   ![Code Snippet]

   a)

   0x00A0 0000  mov eax, 0
   0x00A0 0004  mov ebx, [esp+0x0C]
   0x00A0 0008  xor eax, ebx
   0x00A0 000C  pop ecx
   0x00A0 0010  mov [esp], 1
   0x00A0 0013  ..

   b)

   0xB000 0000 0x0000 000A
   0xB000 0004 0x0000 0000
   0xB000 0008 0x0000 0001
   0xB000 000C 0x0000 0004
   0xB000 0010 0xB000 0028
   0xB000 0014 0x00A0 0B00
   0xB000 0018 0x0000 0001

   ESP  
   0xB000 000C
   EBP  
   0xB000 0010
   EAX  
   0x0000 0000
   EBX  
   0x0000 0000
   ECX  
   0x0000 0000
   EIP  
   0x00A0 0000
II.)

(a)

0x00A0 0000  push eax
0x00A0 0004  push [ebx]
0x00A0 0008  call func ;resides at 0x00A0 B000

(b)

Data Stack

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
</table>
| 0xB000 0000 | 0x0000 000A | ESP
| 0xB000 0004 | 0x0000 0000 |
| 0xB000 0008 | 0x0000 0001 |
| 0xB000 000C | 0x0000 0004 |
| 0xB000 0010 | 0xB000 0028 |
| 0xB000 0014 | 0x00A0 0B00 |
| 0xB000 0018 | 0x0000 0001 |

EAX

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 0001</td>
<td></td>
</tr>
</tbody>
</table>

EBX

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB000 0018</td>
<td></td>
</tr>
</tbody>
</table>

ECX

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 0000</td>
<td></td>
</tr>
</tbody>
</table>

EIP

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00A0 0000</td>
<td></td>
</tr>
</tbody>
</table>
a.) In the lectures during class, we did not see the mov at line 0x00A0 0008. Why was this the case?

b.) Which instruction accomplishes the same task as the mov and pop at 0x00A0 0008 and 0x00A 000C? (hint: it was mentioned in a previous lecture)
This assignment will be done in groups of 2-3.

In class we talked about x86 and how IDA pro can be used to analyze binary files statically and at runtime. In this assignment you will become familiar with IDA pro and will use it to statically identify areas of interest within a binary file (a dynamically linked library or DLL) that lacks high-level source code\(^1\).

When answering each question, be very explicit describing your steps and include any screenshots you think will help evaluate your work.

All files required for this assignment can be downloaded from the course web page:
http://www.cs.utep.edu/DeptCS/courses/cs5390/

For the following questions open IDA Pro and load the file: HW1.dll

1. What is the address of the DllMain function?

2. Use the Imports window to browse to gethostbyname. Where is the import located?

3. How many functions call gethostbyname?

4. A Domain Name System Server is a machine on a network that converts names to IP addresses (e.g., google.com has an address 74.125.227.197). Focusing on the call to gethostbyname located at 0x10001757, can you figure out which DNS request will be made?

5. How many local variables and parameters has IDA Pro recognized for the subroutine at 0x10001656?

6. Locate the string `cmd.exe /c` in the disassembly. Where is it located? (Extra credit: What is happening in the area of code that references `cmd.exe /c`?)

7. At 0x100101C8, it looks like dword_1008E5C4 is a global variable that helps decide which path to take. How does the DLL set dword_1008E5C4? (Hint: Use dword_1008E5C4’s cross-references.)

8. At 0x10001701 is a call to socket. What are the three parameters?

---

\(^1\) Portions of this assignment were taken from: Honig, Andrew. *Practical Malware Analysis*. No Starch Press, 2012.
9. Using the MSDN page for socket and the named symbolic constants functionality in IDA Pro, can you make the parameters to the call at 0x10001701 more meaningful? What are the parameters after you apply changes?

10. At 0x10001358, there is a call to Sleep (an API function that takes one parameter containing the number of milliseconds to sleep). Looking backward through the code, how long will the program sleep if this code executes? (Keep in mind that IDA Pro is not perfect).

**Deliverables:** Email a zip file with the subject **Assignment-StaticBinaryAnalysis** to sp14cs4390cs5390@gmail.com. The zip file must contain the following:

- A write-up with steps detailing your thoughts and actions during your completion of each question. Also include any screenshots you deem necessary.

- Your idb file. This is the file saved by IDA Pro that contains your comments, renamed functions, etc.)
This assignment will be done in groups of 2-3.

**Part 1**

Answer the following questions:

1. Recall from the lecture that when entering 7 As, the dynamic memory trace showed the following:

   ![Image of memory trace]

   Why does IDA Pro show the 0A (newline character) on the left-most side instead of the right-most? What memory location really holds the “0A”?

2. Given the following data stack, label (1) the return address, (2) old base pointer and (3) realistic local variables and parameters.

   **Data Stack**

<table>
<thead>
<tr>
<th>ESP</th>
<th>EBP</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB00 0004</td>
<td>0xB00 000C</td>
<td>0xB00 0000</td>
</tr>
<tr>
<td>0xB00 0008</td>
<td>0xB00 0001</td>
<td>0xB00 0000A</td>
</tr>
<tr>
<td>0xB00 0010</td>
<td>0xB00 0004</td>
<td>0xB00 0000</td>
</tr>
<tr>
<td>0xB00 0014</td>
<td>0xB00 00028</td>
<td>0xB00 0000</td>
</tr>
<tr>
<td>0xB00 0018</td>
<td>0xB00 00A0</td>
<td>0xB00 0001</td>
</tr>
</tbody>
</table>

   **EBP**

   | 0xB00 0010 |
   | 0xB00 000C |

   **ESP**

   | 0xB00 0000 |
   | 0xB00 0004 |
   | 0xB00 0008 |
   | 0xB00 0010 |
   | 0xB00 0014 |
   | 0xB00 0018 |

   | 0xB00 0000A |
   | 0xB00 0001 |
   | 0xB00 0004 |
   | 0xB00 00028 |
   | 0xB00 00A0 |
   | 0xB00 0001 |
3. Use the following instruction snippet to answer the following questions.

```
; Attributes: bp-based frame
; int cdecl sub_10211B0(char *)
sub_10211B0 proc near

var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_4= dword ptr -4
arg_0= dword ptr 8

push ebp
mov ebp, esp
```

a. What are the values (-20h, -1Ch, -4, and 8) an offset from?

b. Draw a picture showing the stack frame for this function.

c. What is the size of each local variable and parameter (in bytes)?

d. Assuming var_4 is filled by user command-line input, how many bytes would have to be entered to completely overwrite the old EBP address and nothing further? (assume no defenses)
Part 2

```c
#include <stdio.h>

int i; //assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
        copy[i] = name[i];
    copy[i] = '\0';
}

void main()
{
    char input[16]; //assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); //parameters pushed in reverse order
    doCopy(input);
    printf("done copying: ");
    printf(input);
}
```

1. If the code above were written in Java, would you ever receive an error? If so, when would you receive the error?

2. Describe some trade-offs of using Java vs. using C to implement this code segment.
3. Based on the following code snippet, answer the following question.

```assembly
; Attributes: bp-based frame
sub_1021870 proc near
var_C= dword ptr -0Ch
var_8= dword ptr 8h
var_4= dword ptr 4h
push ebp
mov ebp, esp
sub esp, 0Ch            ; Integer Subtraction
```

a. What is being done during the first three instructions?

b. Assuming var_8 is filled by user input, how many bytes are there available for a malicious payload (code injected by a malicious user), assuming no defenses are in place.

c. What are some other ways that a malicious user can inject larger payloads?
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Assume the code in (a) has executed. Fill in possible values for the heap and stack in (b).

(a)

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main()
{
    u_int diff; // 4 byte unsigned integer
    char *string1 = (char *) malloc(4);
    char *string2 = (char *) malloc(4);
    char *string3 = (char *) malloc(4);

    ...
}
```

(b)
2. Assume the code in (a) has executed. Fill in possible values for the heap and stack in (b). Write your assumptions.

(a)

```java
Foot f1 = new Foot(5); // 5 toes
Foot f2 = new Foot(5); // 5 toes
Leg leg1 = new Leg();
leg1.foot = f1;
Leg leg2 = new Leg();
leg2.foot = f2;
Torso t = new Torso();
t.leg1 = leg1;
t.leg2 = leg2;
```

(b)
This assignment will be done in groups of 2-3. Late work will only be accepted until March 1st at 11:59pm and will be deducted two letter grades.

Scenario: Because of your incredibly great endorsements by your System Security Assurance professor, you have been selected by the NSA to analyze a never before seen virus. Experts claim that the virus will execute mass destruction if it is not defused in time. In this assignment you will use the IDA Pro debugger to execute and trace through a program in order to defuse a “destructor”.

When answering each question, be very explicit describing your steps and include any screenshots or other materials you think will help evaluate your work.

All files required for this assignment can be downloaded from the course web page:
http://www.cs.utep.edu/DeptCS/courses/cs5390/

Instructions:

1. Download the file called destructor.exe from the course web page.

2. Open IDA Pro (it is highly recommended that you use your own system, but you may also use a computer in the CS lab or on the Citrix server (http://my.apps.utep.edu)

3. Click on Go

4. Click on File->Open

5. Navigate to the directory where you saved destructor.exe

6. Select to view all file types and select the destructor.exe file.

7. Disassemble!
Important Note if you are using Windows 7, 8, or 8.1

When you start debugging you will receive the following prompt:

Press **OK** and then continue the debugging process (press F9 or press the play button).

Next, you will receive the following prompt:

Click **Change exception definition** and then make the following selections:

Press **OK** and then **Yes**. From here, IDA will work as intended.
Your task:

Answer the following questions:

1. What is the first secret passphrase?

2. What is the second secret passphrase?

3. What is the third secret passphrase?

4. Write java code to implement the 1st passphrase check in the same way as the assembly code.

5. Write java code to implement the 2nd passphrase check in the same way as the assembly code.

6. Write java code to implement the 3rd passphrase check in the same way as the assembly code.

Extra Credit:
Find the 4th secret passphrase (hint: write pseudo-code while you are debugging the code) and describe your steps. Include comments in your idb file.

Deliverables: Email a zip file with the subject Assignment-DynamicAnalysis to sp14cs4390cs5390@gmail.com. The zip file must contain the following:

- For questions 1-3, a write-up with steps detailing your thoughts and actions during your completion of each question. Also include any screenshots you deem necessary.

- For questions 1-3, your idb file (containing your comments, renamed functions, etc.)

- For question 4-6, a java file called Passphrase<#>.java, where <#> is the passphrase number.

Note: there will be no credit given if you simply supply an answer, you must document your steps and your code in sufficient detail.
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Using the source code in Figure 1, what would a malicious entity have to enter as input to force the program to execute code at memory location “0x0400 0000” when returning from the call to the foo function?

2. How much space is available for malicious instructions? Write down your assumptions.

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main()
{
    foo();
}

int foo()
{
    int a; // 4 byte integers
    int b;
    char c[16];

    a = 12;
    b = 10;

    // read unbounded input from command line into c...

    return a;
}
```

Figure 1
3. Using the code in Figure 2, fill in realistic values for the stack and heap after executing line 13. Assume that heap memory is allocated contiguously (i.e., there are no gaps between the space for string1 and string2).

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main()
{
    u_int diff; // 4 bytes for unsigned integer
    char *string1 = (char *) malloc(4);
    char *string2 = (char *) malloc(4);
    string1[0] = '\0';
    string2[0] = '\0';
    diff = (u_int) string2 - (u_int) string1;
    strcat(string2, "AA");
    printf(string2);
    printf("\n");
    memset(string1, 'B', (uint)(diff + 1));
    printf(string2);
}
```

Figure 2
4. In Figure 2, fill in **line 17** with a value that will overflow string 1 to write the first character in string2.

5. In class we saw an example of a heap spray attack that used Adobe PDF Reader and exploited the fact that Adobe PDF Reader executes user code. Name another piece of software that works in a similar way as Adobe PDF Reader. Include a simple diagram to show how it works.

6. Under what circumstances will a heap spray eventually lead to malicious code execution?

7. On its own is a heap spray dangerous? Explain.

8. What defenses may help protect against heap spray?
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Two processes are running on the same computer at the same time. A debugger indicates that their .text section (executable code) starts at 0x00D00 0000. Is there a potential for the two to clash? Why or why not?

2. 10 processes are running concurrently on a PC running a modern version of Windows. How much address space is available to each process on a 32-bit system with 8GB of RAM and a 500GB external disk? Explain your answer.

3. List one trade-off of using statically linked libraries vs. dynamic linked libraries.

4. Evil Carlos already has administrator privileges on machine A. Is there any reason why he might want to use dllInjection for a malicious purpose? Give a scenario.

5. What is the difference between NtCreateThread and CreateRemoteThread?
Read over the MSDN documentation for the GetProcAddress function below.

**GetProcAddress function**

Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).

**Syntax**

```cpp
FARPROC WINAPI GetProcAddress(
    _In_  HMODULE hModule,
    _In_  LPCSTR lpProcName
);
```

**Parameters**

- **hModule [in]**
  A handle to the DLL module that contains the function or variable. The `LoadLibrary`, `LoadLibraryEx`, `LoadPackagedLibrary`, or `GetModuleHandle` function returns this handle.

  The `GetProcAddress` function does not retrieve addresses from modules that were loaded using the `LOAD_LIBRARY_AS_DATAFILE` flag. For more information, see `LoadLibraryEx`.

- **lpProcName [in]**
  The function or variable name, or the function's ordinal value. If this parameter is an ordinal value, it must be in the low-order word; the high-order word must be zero.

**Return value**

If the function succeeds, the return value is the address of the exported function or variable.

6. In the following pseudocode, what is the `GetProcAddress` call doing? How is it related to the GetProcAddress function?

```c
hVictimProcess = OpenProcess(PROCESS_ALL_ACCESS, 0, victimProcessID);
pNameInVictimProcess = VirtualAllocEx(hVictimProcess, ..., sizeof(maliciousLibraryName), ..., ...);
WriteProcessMemory(hVictimProcess, ..., maliciousLibraryName, sizeof(maliciousLibraryName), ...);
GetProcAddress("Kernel32.dll");
GetProcAddress("LoadLibraryA");
CreateRemoteThread(hVictimProcess, ..., LoadLibraryAddress, pNameInVictimProcess, ..., ...);
```
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Fill in the following Figure to show how DLL injection works:

```
<table>
<thead>
<tr>
<th>Hard Drive</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Process</td>
<td>User Process</td>
</tr>
<tr>
<td>User DLL</td>
<td></td>
</tr>
<tr>
<td>iexplore.exe</td>
<td></td>
</tr>
</tbody>
</table>

```

2. Fill in the following Figure to show how direct injection works:

```
<table>
<thead>
<tr>
<th>Hard Drive</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Process</td>
<td>User Process</td>
</tr>
<tr>
<td>User DLL</td>
<td></td>
</tr>
<tr>
<td>iexplore.exe</td>
<td></td>
</tr>
</tbody>
</table>
```

Internet

Internet
3. Does Evil Carlos have to get the address of Kernel32.dll when using direct injection? Why or why not?

4. Write/Explain the steps needed to conduct process replacement.

5. Would Evil Carlos have the same impact if he creates a malicious file, renames it the same as a legitimate file and then replaces the original file?

6. What is the difference between a local hook and a remote hook?

7. What is a legitimate reason for installing a keylogger on a system?

8. Alice wants to give Bob a copy of her new game called securitywars.exe, but there is no way she can reach Bob. She copies the file on a flash drive and gives it to Evil Carlos (Bob’s best friend). Evil Carlos has developed keylogger.dll and placed it in the same directory as securitywars.exe. Evil Carlos used detours to force securitywars.exe to load keylogger.dll. Bob runs securitywars.exe, and doesn’t notice the keylogger is running. Describe some countermeasures.
This assignment will be completed individually.

In class we talked about several ways to conduct process injection. In this assignment, you will use IDA Pro to analyze executables and a DLL file both statically and dynamically to determine their behavior.

When answering each question, be very explicit in describing your steps and include any screenshots you think will help evaluate your work.

All files required for this assignment can be downloaded from the course web page: http://www.cs.utep.edu/DeptCS/courses/cs5390/

For the following questions download two files: Annoying.exe and Annoying.dll. Make sure that both files are in the same directory.

1. What happens when you run the executable?
2. What process is a victim of this executable? (use IDA Pro to find out)
3. How can you stop the strange behavior?
4. Describe in detail how Annoying.exe operates. (use IDA Pro)

For the following questions download the following file: NotSure.exe.

5. What is the purpose of this executable? (use IDA Pro)
6. How does the payload inject itself? (use IDA Pro)
7. What filesystem residue does this program create? (use IDA Pro)
**Deliverables:** Email a zip file with the subject **Assignment-ProcessInjection** to sp14cs4390cs5390@gmail.com. The zip file must contain the following:

- A write-up with steps detailing your thoughts and actions during your completion of each question. Also include any screenshots you deem necessary.
- Your idb files. This is the file saved by IDA Pro that contains your comments, renamed functions, etc.)
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Will the following result in a successful operation? If not, change something so that it will result in a successful operation. Explain.

2. Assume that John is part of the “Guests” group that has only (r) access to securitywars.exe. Will the following result in a successful operation? If not, change something so that it will result in a successful operation. Explain.
3. Mary has installed the *Doors* operating system. *Doors* does not store user credentials, but instead stores only the hashes of salted passwords. The hash algorithm has yet to have any collisions found. Are Mary’s credentials guaranteed to be safe? Explain.

4. Explain in your own words why random numbers are so important for one-way hash functions.

5. Software-generated random numbers are usually called pseudo-random number generators. Why?

6. Describe some ways that a truly random number can be generated. Explain.
This assignment will be done in groups of 2-3.

Answer the following questions:

1. Fill in the following figure showing the basic data flow for windows authentication.

2. List some reasons why passwords should be hashed, not encrypted, and kept on the system.

3. Explain why sometimes the system keeps a password and not just the hash.

4. Come up with a small analogy story to describe the PassThePass attack (the version that reads directly from memory).
You are analyzing malware and trying to determine its behavior. You pinpoint this area of code in the assembly and start your analysis.

```
xor edx, edx
lea eax, [esp+404h+DueTime]
mov dword ptr [esp+404h+SystemTime.wYear], edx
lea ecx, [esp+404h+SystemTime]
mov dword ptr [esp+404h+SystemTime.wDayOfWeek], edx
push eax ; lpFileTime
mov dword ptr [esp+408h+SystemTime.wHour], edx
push ecx ; lpSystemTime
mov dword ptr [esp+40Ch+SystemTime.wSecond], edx
mov [esp+40Ch+SystemTime.wYear], 834h
call ds:SystemTimeToFileTime
```

and then later ...

```
call ds:CreateWaitableTimerA ; store resulting object in eax
push 0 ; FResume
push 0 ; lpArgToCompletionRoutine
push 0 ; pfnCompletionRoutine
lea edx, [esp+410h+DueTime]
mov esi, eax ; move address of timer object into esi register
push 0 ; lPeriod
push edx ; lpDueTime
push esi ; hTimer
call ds:SetWaitableTimer
push 0FFFFFFFFh ; dwMilliseconds
push esi ; hHandle
call ds:WaitForSingleObject
```

and then...

```
mov esi, 14h
```

```
loc_401126: ; lpThreadId
push 0
push 0 ; dwCreationFlags
push 0 ; lpParameter
push offset StartAddress ; lpStartAddress
push 0 ; dwStackSize
push 0 ; lpThreadAttributes
call edi ; CreateThread
dec esi
jnz short loc_401126
```
1. Add more comments to each line indicating the purpose of the line (look at the appendix for function descriptions).

2. As a whole what is this malware doing?

3. Into which categories that we discussed in the lecture could this malware fit? Explain.
Appendix
The following are descriptions of some of the functions called in the code.

**SetWaitableTimer function**

Activates the specified waitable timer. When the due time arrives, the timer is signaled and the thread that set the timer calls the optional completion routine.

**Syntax**

```cpp
BOOL WINAPI SetWaitableTimer(
    _In_ HANDLE hTimer,
    _In_ const LARGE_INTEGER *pDueTime,
    _In_ LONG lPeriod,
    _In_opt_ PTIMERAPCROUTINE pfnCompletionRoutine,
    _In_opt_ LPVOID lpArgToCompletionRoutine,
    _In_ BOOL fResume
);
```
WaitForSingleObject function

124 out of 175 rated this helpful - Rate this topic

Waits until the specified object is in the signaled state or the time-out interval elapses.

To enter an alertable wait state, use the **WaitForSingleObjectEx** function. To wait for multiple objects, use **WaitForMultipleObjects**.

Syntax

```cpp
DWORD WINAPI WaitForSingleObject(
    _In_ HANDLE hObject,
    _In_ DWORD dwMilliseconds
);
```

Parameters

*hHandle [in]*

A handle to the object. For a list of the object types whose handles can be specified, see the following Remarks section.

If this handle is closed while the wait is still pending, the function’s behavior is undefined.

The handle must have the **SYNCHRONIZE** access right. For more information, see Standard Access Rights.

*dwMilliseconds [in]*

The time-out interval, in milliseconds. If a nonzero value is specified, the function waits until the object is signaled or the interval elapses. If *dwMilliseconds* is zero, the function does not enter a wait state if the object is not signaled; it always returns immediately. If *dwMilliseconds* is **INFINITE**, the function will return only when the object is signaled.
You suspect an executable on your system is malware.

1. Analyze the assembly in Figure 1 to determine what the malware is likely doing. The parameter labeled `buffer` in Figure 1 is a pointer to the memory address 0x00408060 (see Figure 2).
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab13_01.exe:0040805E</td>
<td>db 0</td>
<td></td>
</tr>
<tr>
<td>Lab13_01.exe:0040805F</td>
<td>db 0</td>
<td></td>
</tr>
<tr>
<td>Lab13_01.exe:00408060</td>
<td>unk_408060 db 4Ch</td>
<td>L</td>
</tr>
<tr>
<td>Lab13_01.exe:00408061</td>
<td>db 4Ch</td>
<td>L</td>
</tr>
<tr>
<td>Lab13_01.exe:00408062</td>
<td>db 4Ch</td>
<td>L</td>
</tr>
<tr>
<td>Lab13_01.exe:00408063</td>
<td>db 15h</td>
<td></td>
</tr>
<tr>
<td>Lab13_01.exe:00408064</td>
<td>db 48h</td>
<td>K</td>
</tr>
<tr>
<td>Lab13_01.exe:00408065</td>
<td>db 49h</td>
<td>I</td>
</tr>
<tr>
<td>Lab13_01.exe:00408066</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:00408067</td>
<td>db 58h</td>
<td>X</td>
</tr>
<tr>
<td>Lab13_01.exe:00408068</td>
<td>db 4Fh</td>
<td>O</td>
</tr>
<tr>
<td>Lab13_01.exe:00408069</td>
<td>db 52h</td>
<td>R</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806A</td>
<td>db 58h</td>
<td>X</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806B</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806C</td>
<td>db 57h</td>
<td>W</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806D</td>
<td>db 56h</td>
<td>U</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806E</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:0040806F</td>
<td>db 57h</td>
<td>W</td>
</tr>
<tr>
<td>Lab13_01.exe:00408070</td>
<td>db 4Ch</td>
<td>L</td>
</tr>
<tr>
<td>Lab13_01.exe:00408071</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:00408072</td>
<td>db 49h</td>
<td>I</td>
</tr>
<tr>
<td>Lab13_01.exe:00408073</td>
<td>db 5Eh</td>
<td></td>
</tr>
<tr>
<td>Lab13_01.exe:00408074</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:00408075</td>
<td>db 55h</td>
<td>U</td>
</tr>
<tr>
<td>Lab13_01.exe:00408076</td>
<td>db 5Ah</td>
<td>Z</td>
</tr>
<tr>
<td>Lab13_01.exe:00408077</td>
<td>db 57h</td>
<td>W</td>
</tr>
<tr>
<td>Lab13_01.exe:00408078</td>
<td>db 42h</td>
<td>B</td>
</tr>
<tr>
<td>Lab13_01.exe:00408079</td>
<td>db 48h</td>
<td>H</td>
</tr>
<tr>
<td>Lab13_01.exe:0040807A</td>
<td>db 52h</td>
<td>R</td>
</tr>
<tr>
<td>Lab13_01.exe:0040807B</td>
<td>db 48h</td>
<td>H</td>
</tr>
<tr>
<td>Lab13_01.exe:0040807C</td>
<td>db 15h</td>
<td></td>
</tr>
<tr>
<td>Lab13_01.exe:0040807D</td>
<td>db 58h</td>
<td>X</td>
</tr>
<tr>
<td>Lab13_01.exe:0040807E</td>
<td>db 54h</td>
<td>T</td>
</tr>
<tr>
<td>Lab13_01.exe:0040807F</td>
<td>db 56h</td>
<td>U</td>
</tr>
</tbody>
</table>
2. Assume that the standard index is used for base64 encoding (see Figure 3) and decode the string Y3Mx.

<table>
<thead>
<tr>
<th>Value</th>
<th>Char</th>
<th>Value</th>
<th>Char</th>
<th>Value</th>
<th>Char</th>
<th>Value</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>16</td>
<td>Q</td>
<td>32</td>
<td>g</td>
<td>48</td>
<td>w</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>17</td>
<td>R</td>
<td>33</td>
<td>h</td>
<td>49</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>18</td>
<td>S</td>
<td>34</td>
<td>i</td>
<td>50</td>
<td>y</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>19</td>
<td>T</td>
<td>35</td>
<td>j</td>
<td>51</td>
<td>z</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>20</td>
<td>U</td>
<td>36</td>
<td>k</td>
<td>52</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>21</td>
<td>V</td>
<td>37</td>
<td>l</td>
<td>53</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>22</td>
<td>W</td>
<td>38</td>
<td>m</td>
<td>54</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>H</td>
<td>23</td>
<td>X</td>
<td>39</td>
<td>n</td>
<td>55</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
<td>24</td>
<td>Y</td>
<td>40</td>
<td>o</td>
<td>56</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>J</td>
<td>25</td>
<td>Z</td>
<td>41</td>
<td>p</td>
<td>57</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>K</td>
<td>26</td>
<td>a</td>
<td>42</td>
<td>q</td>
<td>58</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>L</td>
<td>27</td>
<td>b</td>
<td>43</td>
<td>r</td>
<td>59</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>28</td>
<td>c</td>
<td>44</td>
<td>s</td>
<td>60</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>N</td>
<td>29</td>
<td>d</td>
<td>45</td>
<td>t</td>
<td>61</td>
<td>9</td>
</tr>
<tr>
<td>14</td>
<td>O</td>
<td>30</td>
<td>e</td>
<td>46</td>
<td>u</td>
<td>62</td>
<td>+</td>
</tr>
<tr>
<td>15</td>
<td>P</td>
<td>31</td>
<td>f</td>
<td>47</td>
<td>v</td>
<td>63</td>
<td>/</td>
</tr>
</tbody>
</table>

http://en.wikipedia.org/wiki/Base64

Figure 3
### Table of ASCII Characters

This table lists the ASCII characters and their decimal, octal and hexadecimal numbers. Characters which appear as names in parentheses (e.g., (nl)) are non-printing characters. A table of the common non-printing characters appears after this table.

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul)</td>
<td>0 0000</td>
<td>0x00</td>
<td></td>
<td>(sp)</td>
<td>32 0040</td>
<td>0x20</td>
<td></td>
<td>@</td>
<td>64 0100</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td>(sch)</td>
<td>1 0001</td>
<td>0x01</td>
<td></td>
<td>(sh)</td>
<td>33 0041</td>
<td>0x21</td>
<td>A</td>
<td>65 0101</td>
<td>0x41</td>
<td>a</td>
<td>97 0141</td>
</tr>
<tr>
<td>(stx)</td>
<td>2 0002</td>
<td>0x02</td>
<td>&quot;</td>
<td>(bx)</td>
<td>34 0042</td>
<td>0x22</td>
<td>B</td>
<td>66 0102</td>
<td>0x42</td>
<td>b</td>
<td>98 0142</td>
</tr>
<tr>
<td>(etx)</td>
<td>3 0003</td>
<td>0x03</td>
<td></td>
<td>(ex)</td>
<td>35 0043</td>
<td>0x23</td>
<td>C</td>
<td>67 0103</td>
<td>0x43</td>
<td>c</td>
<td>99 0143</td>
</tr>
<tr>
<td>(eot)</td>
<td>4 0004</td>
<td>0x04</td>
<td>$</td>
<td>(enq)</td>
<td>36 0044</td>
<td>0x24</td>
<td>D</td>
<td>68 0104</td>
<td>0x44</td>
<td>d</td>
<td>100 0144</td>
</tr>
<tr>
<td>(eng)</td>
<td>5 0005</td>
<td>0x05</td>
<td>%</td>
<td>(ack)</td>
<td>37 0045</td>
<td>0x25</td>
<td>E</td>
<td>69 0105</td>
<td>0x45</td>
<td>e</td>
<td>101 0145</td>
</tr>
<tr>
<td>(bEL)</td>
<td>6 0006</td>
<td>0x06</td>
<td>&amp;</td>
<td>(bel)</td>
<td>38 0046</td>
<td>0x26</td>
<td>F</td>
<td>70 0106</td>
<td>0x46</td>
<td>f</td>
<td>102 0146</td>
</tr>
<tr>
<td>(bs)</td>
<td>7 0007</td>
<td>0x07</td>
<td></td>
<td>(bt)</td>
<td>39 0047</td>
<td>0x27</td>
<td>G</td>
<td>71 0107</td>
<td>0x47</td>
<td>g</td>
<td>103 0147</td>
</tr>
<tr>
<td>(ht)</td>
<td>8 0010</td>
<td>0x08</td>
<td></td>
<td>(ls)</td>
<td>40 0050</td>
<td>0x28</td>
<td>H</td>
<td>72 0110</td>
<td>0x48</td>
<td>h</td>
<td>104 0150</td>
</tr>
<tr>
<td>(nl)</td>
<td>9 0011</td>
<td>0x09</td>
<td></td>
<td>(os)</td>
<td>41 0051</td>
<td>0x29</td>
<td>I</td>
<td>73 0111</td>
<td>0x49</td>
<td>i</td>
<td>105 0151</td>
</tr>
<tr>
<td>(vt)</td>
<td>10 0012</td>
<td>0x0a</td>
<td></td>
<td>(ps)</td>
<td>42 0052</td>
<td>0x2a</td>
<td>J</td>
<td>74 0112</td>
<td>0x4a</td>
<td>j</td>
<td>106 0152</td>
</tr>
<tr>
<td>(mp)</td>
<td>11 0013</td>
<td>0x0b</td>
<td></td>
<td>(ch)</td>
<td>43 0053</td>
<td>0x2b</td>
<td>K</td>
<td>75 0113</td>
<td>0x4b</td>
<td>k</td>
<td>107 0153</td>
</tr>
<tr>
<td>(cr)</td>
<td>12 0014</td>
<td>0x0c</td>
<td></td>
<td>(ns)</td>
<td>44 0054</td>
<td>0x2c</td>
<td>L</td>
<td>76 0114</td>
<td>0x4c</td>
<td>l</td>
<td>108 0154</td>
</tr>
<tr>
<td>(so)</td>
<td>13 0015</td>
<td>0x0d</td>
<td></td>
<td>(si)</td>
<td>45 0055</td>
<td>0x2d</td>
<td>M</td>
<td>77 0115</td>
<td>0x4d</td>
<td>m</td>
<td>109 0155</td>
</tr>
<tr>
<td>(do)</td>
<td>14 0016</td>
<td>0x0e</td>
<td></td>
<td>(di)</td>
<td>46 0056</td>
<td>0x2e</td>
<td>N</td>
<td>78 0116</td>
<td>0x4e</td>
<td>n</td>
<td>110 0156</td>
</tr>
<tr>
<td>(nl)</td>
<td>15 0017</td>
<td>0x0f</td>
<td></td>
<td>(dl)</td>
<td>47 0057</td>
<td>0x2f</td>
<td>O</td>
<td>79 0117</td>
<td>0x4f</td>
<td>o</td>
<td>111 0157</td>
</tr>
<tr>
<td>(dc1)</td>
<td>16 0020</td>
<td>0x10</td>
<td></td>
<td>(h)</td>
<td>48 0060</td>
<td>0x30</td>
<td>P</td>
<td>80 0120</td>
<td>0x50</td>
<td>p</td>
<td>112 0160</td>
</tr>
<tr>
<td>(dc2)</td>
<td>17 0021</td>
<td>0x11</td>
<td></td>
<td>(t)</td>
<td>49 0061</td>
<td>0x31</td>
<td>Q</td>
<td>81 0121</td>
<td>0x51</td>
<td>q</td>
<td>113 0161</td>
</tr>
<tr>
<td>(dc3)</td>
<td>18 0022</td>
<td>0x12</td>
<td></td>
<td>(u)</td>
<td>50 0062</td>
<td>0x32</td>
<td>R</td>
<td>82 0122</td>
<td>0x52</td>
<td>r</td>
<td>114 0162</td>
</tr>
<tr>
<td>(dc4)</td>
<td>19 0023</td>
<td>0x13</td>
<td></td>
<td>(v)</td>
<td>51 0063</td>
<td>0x33</td>
<td>S</td>
<td>83 0123</td>
<td>0x53</td>
<td>s</td>
<td>115 0163</td>
</tr>
<tr>
<td>(dc5)</td>
<td>20 0024</td>
<td>0x14</td>
<td></td>
<td>(w)</td>
<td>52 0064</td>
<td>0x34</td>
<td>T</td>
<td>84 0124</td>
<td>0x54</td>
<td>t</td>
<td>116 0164</td>
</tr>
<tr>
<td>(dc6)</td>
<td>21 0025</td>
<td>0x15</td>
<td></td>
<td>(x)</td>
<td>53 0065</td>
<td>0x35</td>
<td>U</td>
<td>85 0125</td>
<td>0x55</td>
<td>u</td>
<td>117 0165</td>
</tr>
<tr>
<td>(dc7)</td>
<td>22 0026</td>
<td>0x16</td>
<td></td>
<td>(y)</td>
<td>54 0066</td>
<td>0x36</td>
<td>V</td>
<td>86 0126</td>
<td>0x56</td>
<td>y</td>
<td>118 0166</td>
</tr>
<tr>
<td>(dc8)</td>
<td>23 0027</td>
<td>0x17</td>
<td></td>
<td>(z)</td>
<td>55 0067</td>
<td>0x37</td>
<td>W</td>
<td>87 0127</td>
<td>0x57</td>
<td>z</td>
<td>119 0167</td>
</tr>
<tr>
<td>(dc9)</td>
<td>24 0030</td>
<td>0x18</td>
<td></td>
<td>(del)</td>
<td>56 0070</td>
<td>0x3a</td>
<td>X</td>
<td>88 0130</td>
<td>0x5a</td>
<td>x</td>
<td>120 0170</td>
</tr>
<tr>
<td>(dc10)</td>
<td>25 0031</td>
<td>0x19</td>
<td></td>
<td>(sub)</td>
<td>57 0071</td>
<td>0x39</td>
<td>Y</td>
<td>89 0131</td>
<td>0x59</td>
<td>y</td>
<td>121 0171</td>
</tr>
<tr>
<td>(dc11)</td>
<td>26 0032</td>
<td>0x1a</td>
<td></td>
<td>(esc)</td>
<td>58 0072</td>
<td>0x3a</td>
<td>Z</td>
<td>90 0132</td>
<td>0x5a</td>
<td>z</td>
<td>122 0172</td>
</tr>
<tr>
<td>(dc12)</td>
<td>27 0033</td>
<td>0x1b</td>
<td></td>
<td>(fs)</td>
<td>59 0073</td>
<td>0x3b</td>
<td>{</td>
<td>91 0133</td>
<td>0x5b</td>
<td>{</td>
<td>123 0173</td>
</tr>
<tr>
<td>(dc13)</td>
<td>28 0034</td>
<td>0x1c</td>
<td></td>
<td>(gm)</td>
<td>60 0074</td>
<td>0x3c</td>
<td></td>
<td>92 0134</td>
<td>0x5c</td>
<td></td>
<td>124 0174</td>
</tr>
<tr>
<td>(dc14)</td>
<td>29 0035</td>
<td>0x1d</td>
<td></td>
<td>(rs)</td>
<td>61 0075</td>
<td>0x3d</td>
<td>}</td>
<td>93 0135</td>
<td>0x5d</td>
<td>}</td>
<td>125 0175</td>
</tr>
<tr>
<td>(dc15)</td>
<td>30 0036</td>
<td>0x1e</td>
<td></td>
<td>(us)</td>
<td>62 0076</td>
<td>0x3e</td>
<td>^</td>
<td>94 0136</td>
<td>0x5e</td>
<td>^</td>
<td>126 0176</td>
</tr>
</tbody>
</table>

Figure 4 ASCII Chart

### Figure 4 Conversion Chart

http://ascii.cl/conversion.htm
For each of the following, state the security code practices that are violated. Explain the problems and explain ways to fix the problems.

1.

```c
#include <ctype.h> // tolower
#include <string.h> // strcmp
#include <stdio.h> // fgets, fputs

void reveal_secret()
{
    fputs("SUPER SECRET = 42\n", stdout);
}

int verify(const char* name)
{
    char user[256];
    int i;
    for (i = 0; name[i] != '\0'; ++i)
        user[i] = tolower(name[i]);
    user[i] = '\0';
    return strcmp(user, "xyzzy") == 0;
}

int main()
{
    char login[512];
    fgets(login, 512, stdin);
    if (!verify(login))
        return 1;
    reveal_secret();
    return 0;
}
```
2.

Example config file /etc/vsftpd/vsftpd.conf

The default compiled in settings are fairly paranoid. This sample file loosens things up a bit, to make the ftp daemon more usable. Please see vsftpd.conf.5 for all compiled in defaults.

READ THIS: This example file is NOT an exhaustive list of vsftpd options. Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd’s capabilities.

Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES

Uncomment this to allow local users to log in.
local_enable=YES

Uncomment this to enable any form of FTP write command.
write_enable=YES

Default umask for local users is 077. You may wish to change this to 022, if your users expect that (022 is used by most other ftpd’s)
local_umask=022

Uncomment this to allow the anonymous FTP user to upload files. This only
"/etc/vsftpd/vsftpd.conf" 119L, 4607C


3.

input.txt

john.smith
angel.avery
...
*
...
joe.smalls

Application code:

```python
def main():
    userList = []
    databaseConn = openDBCConn("localhost, ...")
    f = open("input.txt")
    lines = f.readlines()
    for line in lines:
        userList.append(line)
        database.addUser(line)
```

import java.io.*;

public class JavaFile {
    public static void main(String args[])
    {
        if(args.length != 1)
            System.exit(0);
        try
        {
            Process p=Runtime.getRuntime().exec(args[0]);
            p.waitFor();
            BufferedReader reader=new BufferedReader(new InputStreamReader(p.getInputStream()));
            String line=reader.readLine();
            while(line!=null)
            {
                System.out.println(line);
                line=reader.readLine();
            }
        } catch(IOException e1) {} catch(InterruptedException e2) {} System.out.println("Done");
    }
}
5.2.2. Enable root login over SSH

Now that virt-v2v is installed, the conversion server must be prepared to accept P2V client connections. The P2V client connects to the conversion server as root using SSH, so root login over SSH must be allowed on the conversion server.

Enable root login over SSH:

1. As root, edit the sshd_config file in /etc/ssh/sshd_config:
   ```
   nano /etc/ssh/sshd_config
   ```

2. Add a line in the Authentication section of the file that says `PermitRootLogin yes`. This line may already exist and be commented out with a "#". In this case, remove the "#".
   ```
   # Authentication:
   #LoginGraceTime 2m
   #PermitRootLogin yes
   #StrictModes yes
   #MaxAuthTries 6
   #MaxSessions 10
   ```

3. Save the updated /etc/ssh/sshd_config file.

4. Restart the SSH server:
   ```
   service sshd restart
   ```

You can now connect to the conversion server as root over SSH.

1. Construct a table showing with the following:
   Rows: protocol layers (i.e., physical, data link, etc.)
   Columns: layer data (layer name, protocol, source value, destination value)

2. Fill in your table from #1 using the information you can gather from the packet highlighted dark blue in Figure 1.

3. What is happening in the 3rd, 4th, and 5th packets in Figure 1?

4. In the lecture the sequence numbers started at a random value (not at 0 as they seemingly do in Figure 1). List a reason why it is not a good idea to always start the sequence numbers at 0.

5. Write the values that should be at a, b, c, d, and e in Figure 1.
6. Construct another table showing with the following:

   Rows: protocol layers (i.e., physical, data link, etc.)

   Columns: layer data (layer name, protocol, source value, destination value)

7. Fill in your table from #6 using the information you can gather from the packet highlighted dark blue in Figure 2.

8. In the 2nd packet, why do you think I intentionally placed black bars to cover some of the data?

9. Name 2 things that are provided in the communication in Figure 1 that are not provided in the communication in Figure 2.

10. Write the purpose of ARP messages. Do you think there exist some ARP messages above the traffic shown in Figure 1? Figure 2?
Frame 10: 182 bytes on wire (1456 bits), 182 bytes captured (1456 bits) on interface 0
Ethernet II, Src: 00:00:00:00:00:00:aa (00:00:00:00:00:00:aa), Dst: NovatelW_54:bd:80 (00:15:ff:54:bd:80)
Transmission Control Protocol, Src Port: 36079 (36079), Dst Port: http (80), Seq: 1, Ack: 1, Len: 116

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.1</td>
<td>192.168.1.5</td>
<td>DNS</td>
<td>Standard query response 0xfd2</td>
</tr>
<tr>
<td>192.168.1.1</td>
<td>192.168.1.5</td>
<td>DNS</td>
<td>Standard query response 0x9356 A 66.96.147.115</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>66.96.147.115</td>
<td>TCP</td>
<td>36079 &gt; http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=4294937844 TSecr=4225</td>
</tr>
<tr>
<td>66.96.147.115</td>
<td>192.168.1.5</td>
<td>TCP</td>
<td>http &gt; 36079 [SYN, ACK] Seq=0 Ack=117 Win=65535 Len=0 MSS=1388 SACK_PERM=1 TSecr=4225</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>66.96.147.115</td>
<td>TCP</td>
<td>36079 &gt; http [ACK] Seq=117 Ack=1377 Win=17536 Len=0 TSval=4294938164 TSecr=4225</td>
</tr>
<tr>
<td>66.96.147.115</td>
<td>192.168.1.5</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>66.96.147.115</td>
<td>TCP</td>
<td>36079 &gt; http [ACK] Seq=117 Ack=1449 Win=17536 Len=0 TSval=4294938164 TSecr=4225</td>
</tr>
<tr>
<td>66.96.147.115</td>
<td>192.168.1.5</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
</tbody>
</table>

Figure 1.

...P...  .......
......  ..........
<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00:00:00:00:aa</td>
<td>Broadcast</td>
<td>ARP</td>
<td>Who has 192.168.1.2? Tell 192.168.1.5</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>192.168.1.2</td>
<td>ARP</td>
<td>192.168.1.2 is at [redacted]</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>192.168.1.2</td>
<td>UDP</td>
<td>Source port: 33858 Destination port: http-alt</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>192.168.1.2</td>
<td>UDP</td>
<td>Source port: 33858 Destination port: http-alt</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>192.168.1.2</td>
<td>UDP</td>
<td>Source port: 33858 Destination port: http-alt</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>192.168.1.2</td>
<td>UDP</td>
<td>Source port: 33858 Destination port: http-alt</td>
</tr>
</tbody>
</table>

Frame 13: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
- Ethernet II, Src: 00:00:00:00:00:aa (00:00:00:00:00:aa), Dst: [redacted] [redacted]
- User Datagram Protocol, Src Port: 33858 (33858), Dst Port: http-alt (8080)
- Data (14 bytes)

0000 64 80 99 4b 1f 34 00 00 00 00 00 aa 08 00 45 0c d..K.4... ....E.
0010 00 2a 61 a2 40 00 40 11 55 c9 c0 a8 01 05 c0 a8 .*a.0.0. U....... 
0020 01 02 84 42 1f 90 00 16 83 7f 77 68 65 72 65 20 ...B.... ..where
0030 61 72 65 20 79 6f 75 0a are you.

Figure 2
1. The following is an incomplete network topology:

a. Fill in the topology so that node $a$ can communicate with node $c$.

b. What should be the gateway address for node $a$?

c. Assign a physical address to each interface on each node in the topology.
d. Node $a$ sends a UDP packet to node $c$ with source port 8080 and destination port 8081. At the application layer, the UDP packet contains the text “Hello”. Fill in the following tables to describe the address resolution and the UDP packet as it traverses the network. Assume routes have already been determined.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>Ethernet Cable</td>
<td></td>
<td>RAW binary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>Ethernet Cable</td>
<td></td>
<td>RAW binary</td>
</tr>
<tr>
<td>Layer</td>
<td>Protocol</td>
<td>Source</td>
<td>Destination</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>Ethernet</td>
<td>RAW binary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cable</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>Ethernet</td>
<td>RAW binary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3/5
<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ethernet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td>Ethernet</td>
<td>RAW binary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. The Fakenet corporation has obtained the following public IP addresses: 80.78.0.0/16.
   a. How many IP addresses has Fakenet obtained?

   b. What are these IP addresses? (specify a range, not every single IP address)

   c. How many IP addresses are available for assignment assuming Fakenet does not further subnet these addresses?

3. Fakenet wants to separate their IP addresses space into 4 subnetworks.
   a. What subnet mask should each host on each subnetwork have?

   b. What are the 4 subnet numbers associated with these subnetworks?

   c. How many addresses are available for assignment in each of these subnets?
This assignment will be completed in groups of size 3. At least one group member must have a personal laptop with administrator (or equivalent) permissions.

In class we talked about networking and buffer overflow. In this assignment, you will exploit a vulnerable ftp program on a remote machine. You will need to use the computers in the rack in room CCSB1.0510. You will need your UTEP miners ID card to enter this room. This room will be available for your use anytime except:

M, W 1:30pm – 3:00pm
T, R 3:00pm – 4:30pm and 6:00pm – 7:30pm

Please contact Jon Ramirez if you have technical issues. His contact information is below:

jvramirez@utep.edu
(915) 747-5568
CS Dept 1.0610

Note that if you work outside of regular business hours (8:00am – 5:00pm M-F), you will not have access to IT support.

When answering each question, be very explicit in describing your steps and include any screenshots you think will help evaluate your work.

This document contains several embedded links to references that will help you answer the questions. A digital version of this assignment can be downloaded from the course web page: http://www.cs.utep.edu/DeptCS/courses/cs5390/
Part 1 – Network Connectivity

1. Using `windows network config` as a reference, assign to your laptop the following network settings.

   IP address: 11.0.0.<group#+200>  (e.g. if you are group 10, then you will use `11.0.0.210`)
   Subnet mask: 255.255.255.0
   and leave the rest of the parameters blank (see the image below).
2. Open a Remote Desktop Connection to the attacker machine: 11.0.0.<group#> (replace <group#> with your group number). username: cs5390 password: cs5390

3. On the attacker machine, implement a network client (or you may use java client socket) and add comments to every line of code to describe in detail what is happening. Save the code to a file named Client.java (alternatively, you may create an annotated network client in any language of your choice – adjust the file extension as needed) and compile.

You may now close all programs on the victim and attacker machines and close connections and all remote desktop sessions.

**Part II – FTP Communication**

4. Open a Remote Desktop Connection to the victim machine: 11.0.0.<group#+10> username: cs5390 password: cs5390

5. On the victim machine, use IDA Pro to launch “C:\Program Files\War-ftp\war-ftp.exe” On the war-ftp GUI, click on the icon on the upper left side to set the server “Online”.

6. Open a Remote Desktop Connection to the attacker machine: 11.0.0.<group#> username: cs5390 password: cs5390

7. On the attacker machine:

   a. use your Client.java program to send the following string to the victim machine using TCP on port 21:

   “USER <group member names>” followed by two bytes: 0x0a0x0d (don’t include quotes)

   Show the traffic using wireshark (create a filter) and save a screenshot to 7a.jpg. (use wireshark ref as a reference)

   b. In the above command what are the two bytes 0x0a0x0d and why are they needed? Write your answers in a file named 7b.txt

   c. If you did this correctly, you will see the string you entered appear in the war-ftp GUI. Take a screenshot and name it 7c.jpg

You may now close all programs on the victim and attacker machines and close connections and all remote desktop sessions.
Part III - Vulnerability Hunting

9. Open a Remote Desktop Connection to the *victim* machine: 11.0.0.<group#+10>
   username: cs5390 password: cs5390

10. On the *victim* machine, use IDA Pro to launch C:\Program Files\War-ftp\war-ftp.exe. On
    the war-ftp GUI, click on the icon on the upper left side to set the server “Online”.

11. Open a Remote Desktop Connection to the *attacker* machine: 11.0.0.<group#>.
    username: cs5390 password: cs5390

12. On the *attacker* machine:

   a. Modify your *Client.java* program. Replace the portion of the code that sends
      “USER <group member names>” followed by two bytes: 0x0a0x0d (don’t include quotes)
      to instead send
      “USER <x bytes of garbage data>” followed by two bytes: 0x0a0x0d (don’t include quotes)
      e.g., if x=4, then the sent data would like similar to:
      USER 0x410x410x410x410x0a0x0d
      Save this program to a file named *FuzzerFtpClient.java* and compile.

   b. Use *FuzzerFtpClient.java* to identify at what memory location the vulnerable buffer
      starts (hint: this will require several trials: switching between the *attacker* and *victim
      machine, restarting the debugging, etc.). Create a folder named *12b* that includes
      screenshots and a detailed description of your process.

   c. Use *FuzzerFtpClient.java* to identify what memory location contains the overwritten
      return address. Create a folder named *12c* that includes screenshots and a detailed
      description of your process.

   d. Exactly how many bytes must you send (including the 5 bytes for “USER ”) to
      overwrite the saved ebp (and nothing further)? Write your answer in a file named *12d.txt*
Part IV – Setting Up the Exploit

13. On the attacker machine:

   a. Modify your FuzzerFtpClient.java program. Replace the portion of the code that sends
      “USER <x bytes of garbage data>” followed by two bytes: 0x0a0x0d (don’t include quotes)
      to instead send
      “USER <y bytes of garbage data><0x540x1d0xab0x71><z bytes of garbage data>” followed by two bytes: 0x0a0x0d
      (don’t include quotes)

      e.g., if y=4 and z=4 then the sent data would like similar to:

      USER 0x410x410x410x410x540x1d0xab0x710x410x410x410x0a0x0d

      Change y to the answer you have in 12d.txt

      Save this program to a file named ExploitA.java and compile.

14. On the victim machine, and with IDA Pro running war-ftpd.exe,

   a. What bytes are located at the memory address 0x71ab1d54? Which x86 opcodes
      represented by the bytes located at this memory address? Use x86 opcodes as a reference.
      Write your answers in a file named 14a.txt

   b. In your ExploitA.java program, why did you have to insert these bytes in reverse
      order? Write your answer in a file named 14b.txt
Part V - Generate Shellcode

15. On the attacker machine, open metasploit console (there is an icon on the desktop).
16. Type the following into the msfconsole:

use payload/windows/shell_bind_tcp
generate -b "\x00\x0a\x0d\x40" -t java -f 16.txt

(you can substitute -java with -python, -perl, -c, etc. depending on the language you are using)

These commands will automatically copy the result into a file named C:\metasploit\16.txt

17. Using metasploit unleashed as a reference, describe what the two commands in 16 are doing
(include a description of the flags: -b, -t, -f). Write your answer in a file named 17.txt

18. Take the shellcode from 16.txt and prepend 12 additional nop opcodes: 0x90. Modify your
ExploitA.java program by placing this padded shellcode into the data that is sent to the victim
machine. Make sure that you place the shellcode in a location so that it is executed when the
buffer overflow occurs. Save your program as Exploit.java and compile.

Part VI - Exploit

19. On the victim machine, make sure that war-ftp.exe is running and is online. Execute your
Exploit.java and use wireshark to capture your traffic to a file named 19.pcapng

20. If your exploit succeeded, a TCP backdoor server has been created on the victim machine
listening on port 4444. On the attacker machine open a new cygwin terminal and use nc.exe to
connect to this backdoor server (use netcat ref as a reference). Capture a screenshot and save it to
a file named 20.jpg

Extra Credit:

21. Why do you need to prepend bytes to the shellcode? Write a single paragraph answer to
21.txt

Deliverables: Create a README.txt on the desktop of the attacker machine that indicates the
location of each of your solution files associated with the questions above (you may save
your solution files on the attacker machine, the victim machine or on your personal
computer. If stored on your local computer, I expect you to email me the solution files).
1. Identify and describe at least one encryption mechanism for layers 3-4 in the network model.

2. Describe a specific example showing how encryption at layers 2, 3 and 4 could potentially (or have been) compromised.
XenServer has been installed in the lab. In this assignment you will add several virtual machines and create a network architecture. At the completion of this assignment, you will have an environment suitable to design and conduct simple and complex penetration tests.

1. Research at least 3 publicly available virtual machines (these can be designed either as native installation packages or VMWare or VirtualBox packages – these can be converted to the XenServer format). Write a paragraph about each of the virtual machines and what they are why they are typically used. Some examples include virtual machines for security testing (metasploitable, beebbox, OWASP, etc.), routing software, email servers, and more.

2. On paper, design an environment consisting of 2 subnets with a total of at least 5 nodes (at least two nodes per subnet; hint: one of these must be a router [or at least have IP forwarding enabled]). Include an entry point for a user to plug in a laptop/PC.

3. Investigate some vulnerabilities with publicly known exploits at Exploit-DB (http://www.exploit-db.com/), and for each node in #2, include at least one vulnerability. Write a description of your scenario and the purpose of each node.

4. Create the scenario you described in #3 and #4 and include screenshots showing how an attacker could infiltrate the networks. Include screenshots and detailed descriptions. Use the document titled dynamic testbeds from the course website as a reference.

5. Describe how firewalls, and IDS/IPS, and defense in depth could help to prevent, detect and help a human react to the attacker in your scenario.

6. Describe how your scenario, could be expanded to create a distributed/cloud. Also include discussion on some of the security ramifications of these systems.