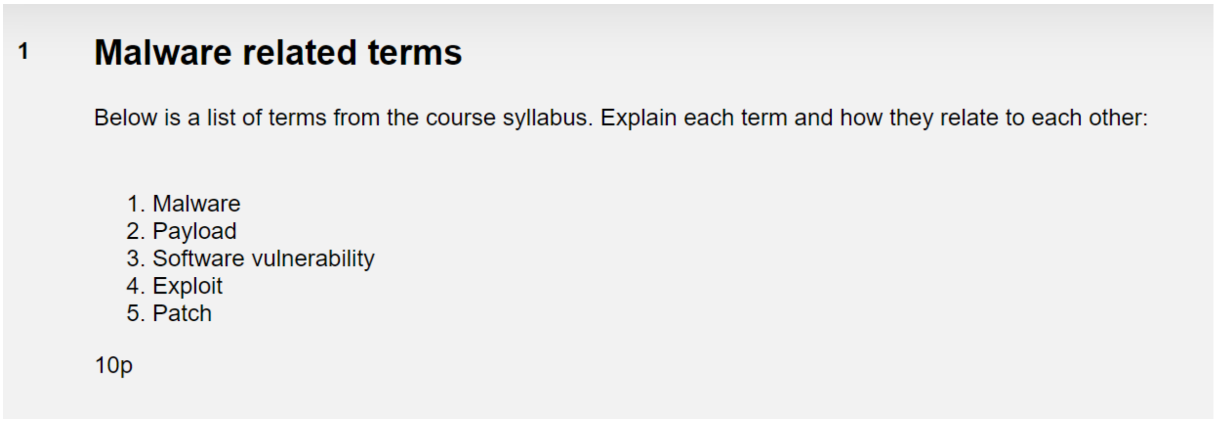


1 **Malware related terms**

Below is a list of terms from the course syllabus. Explain each term and how they relate to each other:

1. Malware
2. Payload
3. Software vulnerability
4. Exploit
5. Patch

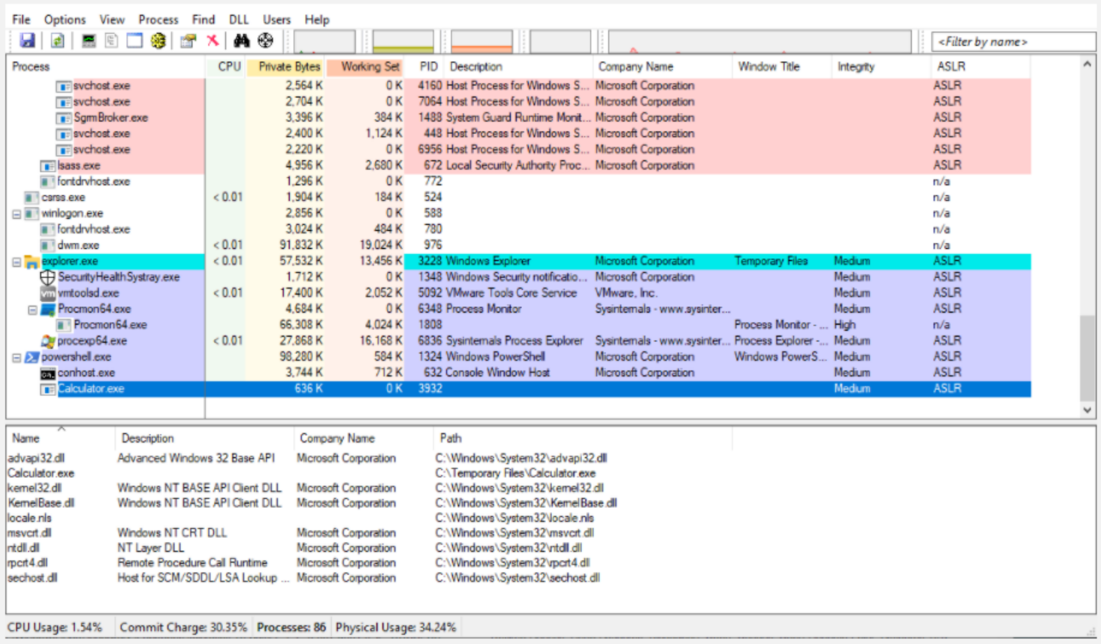
10p

1p for explaining each term, and 1p for relating each term to the other terms.

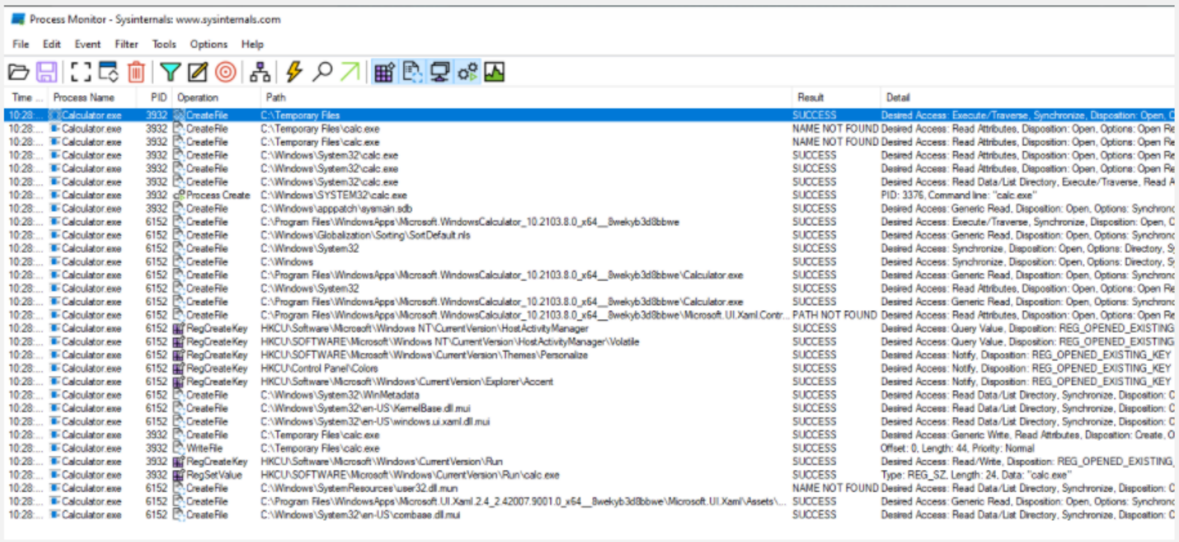
Example:

1. Malware is malicious software, and it typically consists of an exploit and a payload.
2. A payload is the part of a malware that contains the purpose of the malware, the actions the malware author wants to execute on a system. The payload runs after the exploit has gained access to run it. The payload can be reused with other exploits.
3. A software vulnerability is a flaw in a piece of software that makes it vulnerable for attack. Software vulnerabilities can be used by an exploit to gain access, and they can be patched (fixed) by software updates (patches).
4. Exploit is the piece of malware that uses vulnerabilities to gain access on a system. Its purpose is to gain access so that the payload can be run.
5. A patch is a software update that can patch (fix) vulnerabilities. When a patch removes a vulnerability, it will stop the exploits for that vulnerability.

2 Analyze an executable



```
Process                    CPU     Private Bytes  Working Set  PID   Description                      Company Name                  Window Title         Integrity  ASLR
        svchost.exe                2,564 K        0 K          4160  Host Process for Windows S...    Microsoft Corporation                                         ASLR
        svchost.exe                2,704 K        0 K          7064  Host Process for Windows S...    Microsoft Corporation                                         ASLR
        SgmBroker.exe              3,396 K        384 K        1488  System Guard Runtime Monit...    Microsoft Corporation                                         ASLR
        svchost.exe                2,400 K        1,124 K      448   Host Process for Windows S...    Microsoft Corporation                                         ASLR
        svchost.exe                2,220 K        0 K          6956  Host Process for Windows S...    Microsoft Corporation                                         ASLR
      lsass.exe                    4,956 K        2,680 K      672   Local Security Authority Proc... Microsoft Corporation                                         ASLR
      fontdrvhost.exe              1,296 K        0 K          772                                                                                              n/a
    csrss.exe              < 0.01  1,904 K        184 K        524                                                                                              n/a
  ⊟ winlogon.exe                   2,856 K        0 K          588                                                                                              n/a
      fontdrvhost.exe              3,024 K        484 K        780                                                                                              n/a
      dwm.exe              < 0.01  91,832 K       19,024 K     976                                                                                              n/a
  ⊟ explorer.exe           < 0.01  57,532 K       13,456 K     3228  Windows Explorer                 Microsoft Corporation         Temporary Files      Medium     ASLR
      SecurityHealthSystray.exe    1,712 K        0 K          1348  Windows Security notificatio...  Microsoft Corporation                              Medium     ASLR
      vmtoolsd.exe         < 0.01  17,400 K       2,052 K      5092  VMware Tools Core Service        VMware, Inc.                                       Medium     ASLR
    ⊟ Procmon64.exe                4,684 K        0 K          6348  Process Monitor                  Sysinternals - www.sysinter...                     Medium     ASLR
        Procmon64.exe              66,308 K       4,024 K      1808                                                                 Process Monitor - ... High       n/a
      procexp64.exe        < 0.01  27,868 K       16,168 K     6836  Sysinternals Process Explorer    Sysinternals - www.sysinter... Process Explorer -... Medium     ASLR
  ⊟ powershell.exe                 98,280 K       584 K        1324  Windows PowerShell               Microsoft Corporation         Windows PowerS...    Medium     ASLR
      conhost.exe                  3,744 K        712 K        632   Console Window Host              Microsoft Corporation                              Medium     ASLR
      Calculator.exe               636 K          0 K          3932                                                                                    Medium     ASLR

Name            Description                        Company Name           Path
advapi32.dll    Advanced Windows 32 Base API       Microsoft Corporation  C:\Windows\System32\advapi32.dll
Calculator.exe                                                            C:\Temporary Files\Calculator.exe
kernel32.dll    Windows NT BASE API Client DLL     Microsoft Corporation  C:\Windows\System32\kernel32.dll
KernelBase.dll  Windows NT BASE API Client DLL     Microsoft Corporation  C:\Windows\System32\KernelBase.dll
locale.nls                                                                C:\Windows\System32\locale.nls
msvcrt.dll      Windows NT CRT DLL                 Microsoft Corporation  C:\Windows\System32\msvcrt.dll
ntdll.dll       NT Layer DLL                       Microsoft Corporation  C:\Windows\System32\ntdll.dll
rpcrt4.dll      Remote Procedure Call Runtime      Microsoft Corporation  C:\Windows\System32\rpcrt4.dll
sechost.dll     Host for SCM/SDDL/LSA Lookup ...   Microsoft Corporation  C:\Windows\System32\sechost.dll

CPU Usage: 1.54% | Commit Charge: 30.35% | Processes: 86 | Physical Usage: 34.24%
```


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

```
Time ...  Process Name    PID   Operation       Path                                                                                               Result           Detail
10:28:... Calculator.exe  3932  CreateFile      C:\Temporary Files                                                                                 SUCCESS          Desired Access: Execute/Traverse, Synchronize, Disposition: Open, C
10:28:... Calculator.exe  3932  CreateFile      C:\Temporary Files\calc.exe                                                                        NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  3932  CreateFile      C:\Temporary Files\calc.exe                                                                        NAME NOT FOUND   Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  3932  CreateFile      C:\Windows\System32\calc.exe                                                                       SUCCESS          Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  3932  CreateFile      C:\Windows\System32\calc.exe                                                                       SUCCESS          Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  3932  CreateFile      C:\Windows\System32\calc.exe                                                                       SUCCESS          Desired Access: Read Data/List Directory, Execute/Traverse, Read A
10:28:... Calculator.exe  3932  Process Create  C:\Windows\SYSTEM32\calc.exe                                                                       SUCCESS          PID: 3376, Command line: "calc.exe"
10:28:... Calculator.exe  3932  CreateFile      C:\Windows\apppatch\sysmain.sdb                                                                    SUCCESS          Desired Access: Generic Read, Disposition: Open, Options: Synchronc
10:28:... Calculator.exe  6152  CreateFile      C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_10.2103.8.0_x64__8wekyb3d8bbwe            SUCCESS          Desired Access: Execute/Traverse, Synchronize, Disposition: Open, C
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\Globalization\Sorting\SortDefault.nls                                                   SUCCESS          Desired Access: Generic Read, Disposition: Open, Options: Synchronc
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32                                                                                SUCCESS          Desired Access: Synchronize, Disposition: Open, Options: Directory, Sy
10:28:... Calculator.exe  6152  CreateFile      C:\Windows                                                                                         SUCCESS          Desired Access: Synchronize, Disposition: Open, Options: Directory, Sy
10:28:... Calculator.exe  6152  CreateFile      C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_10.2103.8.0_x64__8wekyb3d8bbwe\Calculator.exe  SUCCESS     Desired Access: Generic Read, Disposition: Open, Options: Synchronc
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32                                                                                SUCCESS          Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  6152  CreateFile      C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_10.2103.8.0_x64__8wekyb3d8bbwe\Calculator.exe  SUCCESS     Desired Access: Generic Read, Disposition: Open, Options: Synchronc
10:28:... Calculator.exe  6152  CreateFile      C:\Program Files\WindowsApps\Microsoft.WindowsCalculator_10.2103.8.0_x64__8wekyb3d8bbwe\Microsoft.UI.Xaml.Contr...  PATH NOT FOUND  Desired Access: Read Attributes, Disposition: Open, Options: Open Re
10:28:... Calculator.exe  6152  RegCreateKey    HKCU\Software\Microsoft\Windows NT\CurrentVersion\HostActivityManager                               SUCCESS          Desired Access: Query Value, Disposition: REG_OPENED_EXISTING
10:28:... Calculator.exe  6152  RegCreateKey    HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\HostActivityManager\Volatile                      SUCCESS          Desired Access: Query Value, Disposition: REG_OPENED_EXISTING
10:28:... Calculator.exe  6152  RegCreateKey    HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Themes\Personalize                                  SUCCESS          Desired Access: Notify, Disposition: REG_OPENED_EXISTING_KEY
10:28:... Calculator.exe  6152  RegCreateKey    HKCU\Control Panel\Colors                                                                          SUCCESS          Desired Access: Notify, Disposition: REG_OPENED_EXISTING_KEY
10:28:... Calculator.exe  6152  RegCreateKey    HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Accent                                     SUCCESS          Desired Access: Notify, Disposition: REG_OPENED_EXISTING_KEY
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32\WinMetadata                                                                    SUCCESS          Desired Access: Read Data/List Directory, Synchronize, Disposition: C
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32\en-US\KernelBase.dll.mui                                                       SUCCESS          Desired Access: Read Data/List Directory, Synchronize, Disposition: C
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32\en-US\windows.ui.xaml.dll.mui                                                  SUCCESS          Desired Access: Read Data/List Directory, Synchronize, Disposition: C
10:28:... Calculator.exe  3932  CreateFile      C:\Temporary Files\calc.exe                                                                        SUCCESS          Desired Access: Generic Write, Read Attributes, Disposition: Create, O
10:28:... Calculator.exe  3932  WriteFile       C:\Temporary Files\calc.exe                                                                        SUCCESS          Offset: 0, Length: 44, Priority: Normal
10:28:... Calculator.exe  3932  RegCreateKey    HKCU\Software\Microsoft\Windows\CurrentVersion\Run                                                 SUCCESS          Desired Access: Read/Write, Disposition: REG_OPENED_EXISTING
10:28:... Calculator.exe  3932  RegSetValue     HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\calc.exe                                        SUCCESS          Type: REG_SZ, Length: 24, Data: "calc.exe"
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\SystemResources\user32.dll.mun                                                          NAME NOT FOUND   Desired Access: Read Data/List Directory, Synchronize, Disposition: C
10:28:... Calculator.exe  6152  CreateFile      C:\Program Files\WindowsApps\Microsoft.UI.Xaml.2.4_2.42007.9001.0_x64__8wekyb3d8bbwe\Microsoft.UI.Xaml\Assets\...  SUCCESS  Desired Access: Generic Read, Disposition: Open, Options: Synchronc
10:28:... Calculator.exe  6152  CreateFile      C:\Windows\System32\en-US\combase.dll.mui                                                          SUCCESS          Desired Access: Read Data/List Directory, Synchronize, Disposition: C
```

You have been asked to investigate an incident regarding a suspicious executable Calculator.exe. Use the screenshots from Process Explorer and Process Monitor to answer the questions below.

1. How was Calculator.exe started? (2p)
2. What other information do you get about Calculator.exe from the screenshots provided? Explain how you get the information from the screenshots. (5p)
3. What would be your next steps to further analyze this incident? Which questions remain unanswered after the initial analysis in question 2? Which tools would you use and what would you use them for? (3p)

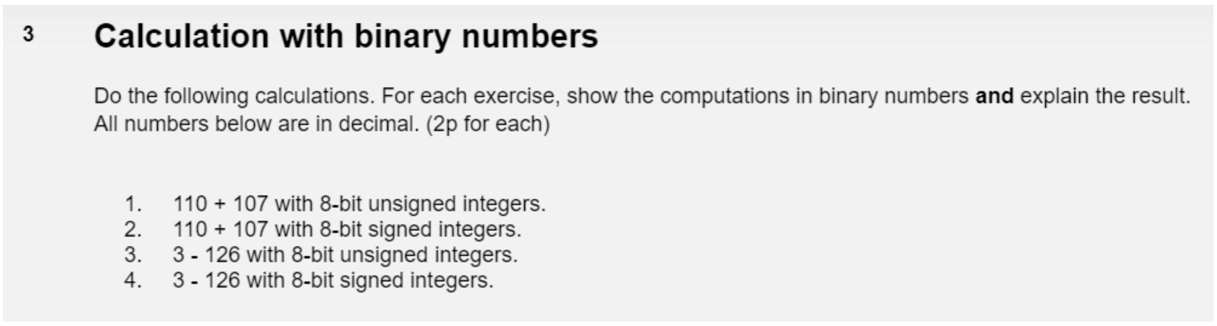
1. 1,5 p for: Calculator.exe was started by Powershell (PID 1324). 0,5p for: We note that Powershell is orphaned and we do not know how it was started.
2. 1p for each of the following. To get full score they must explain how the information is found in the screenshots.
   - Calculator.exe has no company name, runs with medium integrity and has ASLR enabled. Shown in Process Explorer upper pane
   - Calculator.exe is located in c:\Temporary Files. Shown in Process Explorer lower pane

- It writes to c:\Temporary Files\calc.exe (WriteFile event in ProcMon)
- It starts c:\windows\system32\calc.exe (the windows calculator) (ProcessCreate event in ProcMon)
- It creates and sets a registry key: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\calc.exe (RegCreateKey and RegSetValue in ProcMon)

The conclusion should be that Calculator.exe does not appear to be the real Windows calculator and should be investigated further.

3. Several possible answers. Examples:
    - Calc.exe in ProcMon and/or IDA to look for …
    - Investigate further events from ProcMon. For example the registry key that is set, what is it used for? Some may want to investigate the Windows Calcuator App that is started, as it is quite noisy in the ProcMon screenshot.
    - Calculator.exe in IDA to look for …
    - Other filters in ProcMon to look for …

   1p for each suggestion of further work/remaining questions to answer, maximum 3. It is accepted if a student focuses on one or two tracks (e.g. analysis in IDA) and gives detailed steps in this direction)



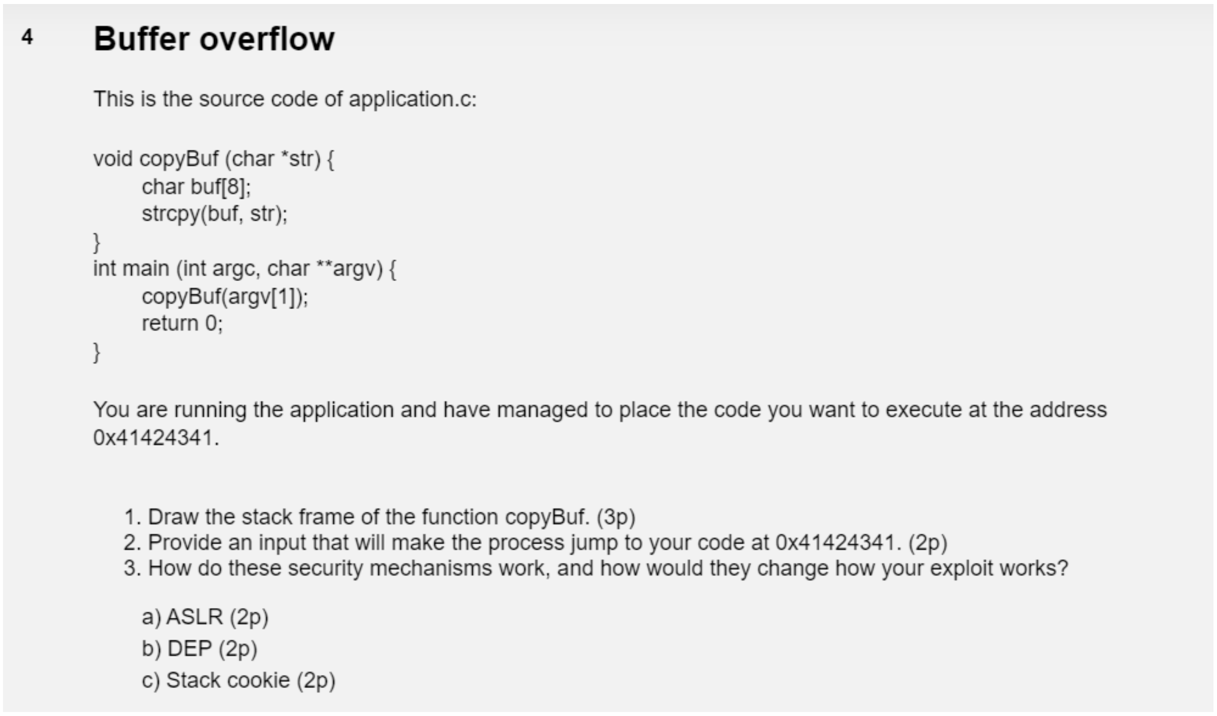
3 Calculation with binary numbers

Do the following calculations. For each exercise, show the computations in binary numbers **and** explain the result. All numbers below are in decimal. (2p for each)

1. 110 + 107 with 8-bit unsigned integers.
2. 110 + 107 with 8-bit signed integers.
3. 3 - 126 with 8-bit unsigned integers.
4. 3 - 126 with 8-bit signed integers.

1p for each correct result, 1p for each correct binary computation and explanation.

1. $110+107 = 01101110_2 + 01101011_2 = 11011001_2 = 217$ No problems.
2. $110+107 = 01101110_2 + 01101011_2 = 11011001_2 = -39$ Overflow.
3. $3 - 126 = 0011_2 - 01111110_2 = 10000101_2 = 133$ Underflow.
4. $3 - 126 = 0011_2 - 01111110_2 = 10000101_2 = -123$ No problems.


4 Buffer overflow

This is the source code of application.c:

```
void copyBuf (char *str) {
      char buf[8];
      strcpy(buf, str);
}
int main (int argc, char **argv) {
      copyBuf(argv[1]);
      return 0;
}
```

You are running the application and have managed to place the code you want to execute at the address 0x41424341.

1. Draw the stack frame of the function copyBuf. (3p)
2. Provide an input that will make the process jump to your code at 0x41424341. (2p)
3. How do these security mechanisms work, and how would they change how your exploit works?

   a) ASLR (2p)
   b) DEP (2p)
   c) Stack cookie (2p)

1. Stack frame of copy buf:

```
__ __ __ __    // Local buffer, size 8
__ __ __ __
__ __ __ __   // Saved frame pointer (EBP)
__ __ __ __   // Saved instruction pointer (EIP)
__ __ __ __   // Argument (argv[1]/str)
```

1p for correct position of ebp,
1p for correct position of eip,
0.5p for correct position and length of local buffer
and 0.5p for correct position of argument

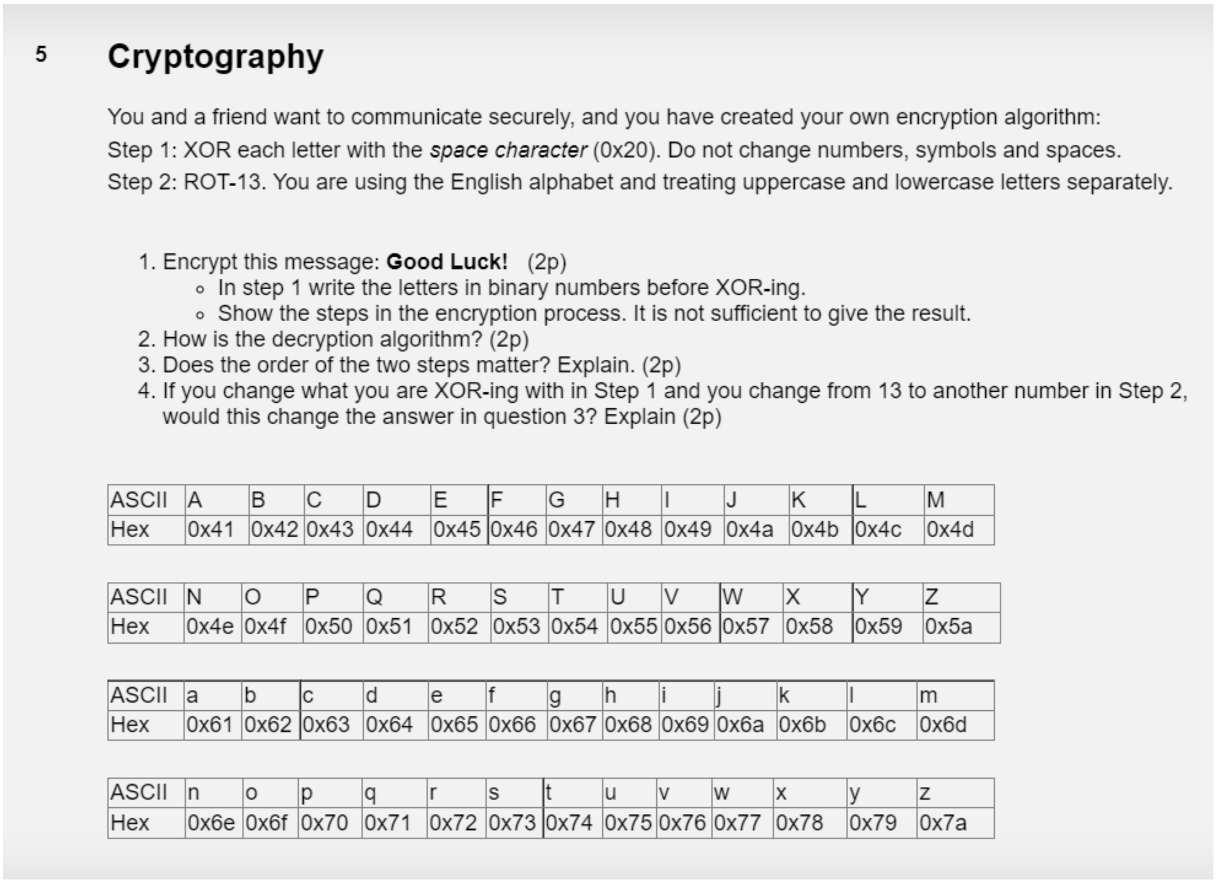
2. AAAABBBBCCCCACBA
   AAAABBBB fills the buffer
   CCCC overwrites the saved frame pointer
   ACBA (0x41434241) overwrites the saved instruction pointer, and returns to 0x41424341 because of endianness.
   It is not necessary to change from hex to letters, but it is necessary to have the correct endianness for full score (2p)

-0.5p for wrong endianness
-1p for forgotten EBP

3. For each: 1p for explaining the mechanism. 1p for explaining how it influences the exploit
   a) ASLR is Address Space Layout Randomization and makes the addresses of stack, instructions, … unpredictable. ASLR would not prevent the overflow, but would make it more difficult to predict where your code is located.

b) DEP is Data Execution Prevention and makes for example the stack non-executable. This means that you can still overflow, but your code cannot be located on the stack or other non-executable memory. (It may not have been in the first place).

c) Stack cookie is an unpredictable value that is placed on the stack below the local buffer and it is checked before the function returns. It would not prevent the overflow, but it would prevent the function from returning to the address you overwrote EIP with.



5 Cryptography

You and a friend want to communicate securely, and you have created your own encryption algorithm:
Step 1: XOR each letter with the *space character* (0x20). Do not change numbers, symbols and spaces.
Step 2: ROT-13. You are using the English alphabet and treating uppercase and lowercase letters separately.

1. Encrypt this message: **Good Luck!** (2p)
   - In step 1 write the letters in binary numbers before XOR-ing.
   - Show the steps in the encryption process. It is not sufficient to give the result.
2. How is the decryption algorithm? (2p)
3. Does the order of the two steps matter? Explain. (2p)
4. If you change what you are XOR-ing with in Step 1 and you change from 13 to another number in Step 2, would this change the answer in question 3? Explain (2p)

ASCII	A	B	C	D	E	F	G	H	I	J	K	L	M
Hex	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48	0x49	0x4a	0x4b	0x4c	0x4d

ASCII	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Hex	0x4e	0x4f	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57	0x58	0x59	0x5a

ASCII	a	b	c	d	e	f	g	h	i	j	k	l	m
Hex	0x61	0x62	0x63	0x64	0x65	0x66	0x67	0x68	0x69	0x6a	0x6b	0x6c	0x6d

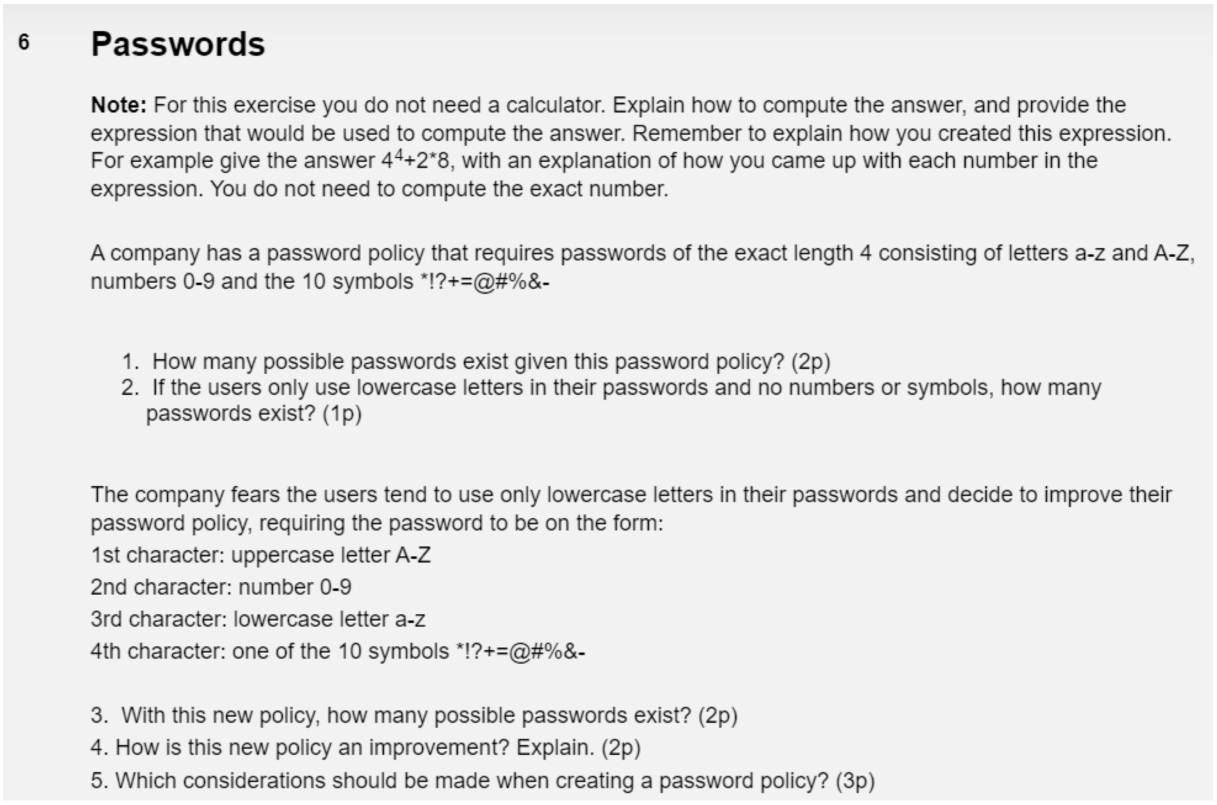
ASCII	n	o	p	q	r	s	t	u	v	w	x	y	z
Hex	0x6e	0x6f	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78	0x79	0x7a

1. 1p for correct XOR, 1p for correct ROT-13. It is not sufficient to show the result, but detailed calculations for a few letters is sufficient.
   Result of XOR: gOOD lUCK
   Result of ROT-13: tBBQ yHPX
2. 2p for either one of these:
   a. The decryption algorithm is the same as the encryption algorithm.
   b. One could also reverse the order of XOR and ROT-13
3. 2p for: The order does not matter. XOR switches between upper and lower case letters. ROT-13 replaces the letter with the letter 13 places after in the alphabet (the letter in the other row of the table provided). The order of these two operations is irrelevant.

   It is necessary to explain the answer to get points.

4. 2p for for example: Depending on how you change step 1, step 2 may not even make sense. The result after step 1 may not be within the ASCII range of the alphabet.

   No more details are required.



6 **Passwords**

Note: For this exercise you do not need a calculator. Explain how to compute the answer, and provide the expression that would be used to compute the answer. Remember to explain how you created this expression. For example give the answer 4^4+2*8, with an explanation of how you came up with each number in the expression. You do not need to compute the exact number.

A company has a password policy that requires passwords of the exact length 4 consisting of letters a-z and A-Z, numbers 0-9 and the 10 symbols *!?+=@#%&-

1. How many possible passwords exist given this password policy? (2p)
2. If the users only use lowercase letters in their passwords and no numbers or symbols, how many passwords exist? (1p)

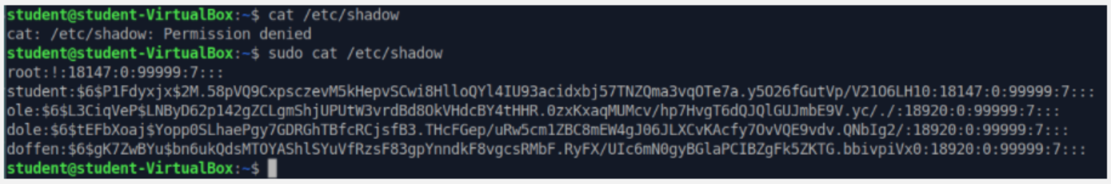
The company fears the users tend to use only lowercase letters in their passwords and decide to improve their password policy, requiring the password to be on the form:
1st character: uppercase letter A-Z
2nd character: number 0-9
3rd character: lowercase letter a-z
4th character: one of the 10 symbols *!?+=@#%&-

3. With this new policy, how many possible passwords exist? (2p)
4. How is this new policy an improvement? Explain. (2p)
5. Which considerations should be made when creating a password policy? (3p)

1. 1p for correct expression, 1p for explanation.
   The number of allowed characters is 26+26+10+10=72. Four characters can then be chosen in 72^4 ways.
2. 1p for correct expression and explanation.
   The number of lowercase letters is 26. Four characters can then be chosen in 26^4 ways.
3. There are 26 ways to choose the first character, 10 ways to choose the second character, 26 ways to choose the third character, and 10 ways to choose the fourth character. The order of the characters is set. The number of possible passwords is then $26*10*26*10 = 26^2*10^2$
4. 2p for reasonable argumentation, for example: It is not an improvement, since it reduces the number of possible passwords, and it makes it more difficult to make memorable passwords. An alternative argumentation is that if people use only lowercase letters, they are likely to choose combinations that form words, and that the actual passwords will only be a subset of the possible combinations. The new policy has probably less obvious choices. This can be accepted as an answer, but to get full score it is necessary to state that the new policy has obvious drawbacks since it reduces the possible passwords significantly. (But the first policy wasn't good either).
5. 1p for each reasonable consideration, max 3. Considerations may be:
   a. Encourage users to make sufficiently long passwords or passphrases
   b. A policy that requires frequent change of passwords may make users choose more predictable passwords
   c. Require characters from different character groups, but length may be more important than characters from several groups
   d. Not allow known common passwords or passphrases, or previously used passwords.

7 Users

This screenshot shows the result of the command
sudo cat /etc/shadow



```
student@student-VirtualBox:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
student@student-VirtualBox:~$ sudo cat /etc/shadow
root:!:18147:0:99999:7:::
student:$6$P1Fdyxjx$2M.58pVQ9CxpsczevM5kHepvSCwi8HlloQYl4IU93acidxbj57TNZQma3vqOTe7a.y5O26fGutVp/V21O6LH10:18147:0:99999:7:::
ole:$6$L3CiqVeP$LNByD62p142gZCLgmShjUPUtW3vrdBd8OkVHdcBY4tHHR.0zxKxaqMUMcv/hp7HvgT6dQJQlGUJmbE9V.yc/./:18920:0:99999:7:::
dole:$6$tEFbXoaj$Yopp0SLhaePgy7GDRGhTBfcRCjsfB3.THcFGep/uRw5cm1ZBC8mEW4gJ06JLXCvKAcfy7OvVQE9vdv.QNbIg2/:18920:0:99999:7:::
doffen:$6$gK7ZwBYu$bn6ukQdsMTOYAShlSYuVfRzsF83gpYnndkF8vgcsRMbF.RyFX/UIc6mN0gyBGlaPCIBZgFk5ZKTG.bbivpiVx0:18920:0:99999:7:::
student@student-VirtualBox:~$ 
```

1. What is shown here? (1p)
2. Why does the second command start with *sudo*? (2p)
3. You add another user *donald* with the same password as the existing user *dole*. Explain what would be equal and what would be different in the lines for *dole* and *donald*. Why? (2p)
4. If you had the same users on a Windows computer, how would the corresponding entries for *dole* and *donald* be? Why? (2p)

1. 1p for: the file /etc/shadow, which contains the passwords hashes in Linux.
2. 2p for: /etc/shadow is only readable by root, and sudo is used to perform the command with root privileges.
3. 1p for: the salt is randomly chosen and will be different. 1p for stating that this would also make the password hash different, even though the passwords are equal.
4. 2p for: Windows does not salt the password hashes, therefore the password hashes would be equal if the passwords are equal.

8 Access control in Linux

Look at the permissions of tent.make by using ls -l:

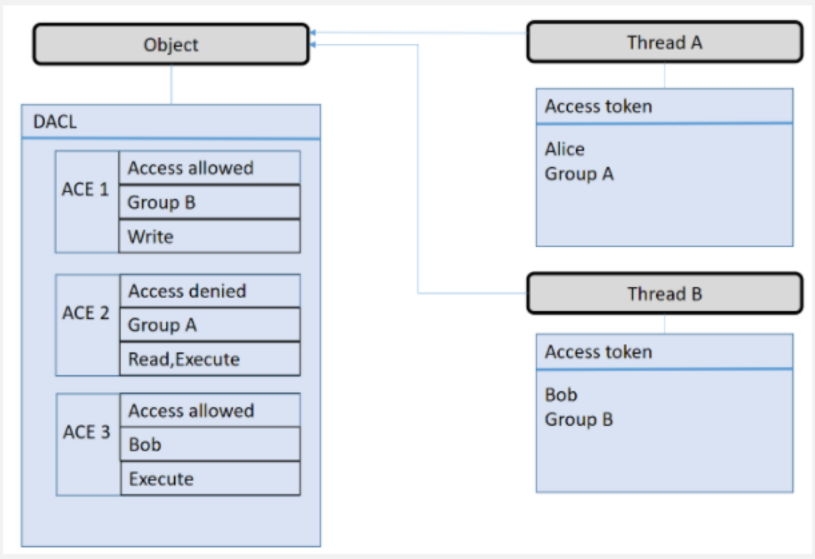
```
-rw-r--r-- 1 Doffen student 1617 Oct 28 11:01 tent.make
```

In these exercises, explain how you think, do not just give a command.

1. How can Doffen make sure that only himself has access to tent.make? (2p)
2. How can Doffen give everyone access to write to tent.make? (2p)
3. How can Doffen give access to all users except one user, Donald? (2p)

1. 2p for: removing the read access for group and others.
2. 2p for: adding write access for both group and others. Only 1p is granted for: Adding write only for others, since this will not give write access to those in the student group.
3. 2p for: No obvious way in Linux, but he could create a group with everybody except Donald, change the file's group to this group, and set the desired access privileges for this group, while removing all access for others.

9 **Access control in Windows**



The figure above shows an object with a discretionary access control list (DACL) and two threads with corresponding access tokens.

For each case below, explain what access the thread will be granted **and** how each access control entry (ACE) influences the access control decision.

1. Thread A requests write access to the object. (2p)
2. Thread B requests read and write access to the object. (2p)
3. Thread A requests execute access to the object. (1p)
4. Thread B requests execute access to the object. (1p)
5. You want both threads to have execute access to the object. If this is not the case at the moment, create one new ACE to make both have access, and explain where in the DACL the entry should be placed. (2p)

The answer must be explained to get full score.

1. 2p for: ACE1 does not apply (not group B), ACE 2 does not apply (not write), ACE3 does not apply (not write). Access is not granted.
2. 2p for: ACE1 allows write, ACE2 does not apply (not group A), ACE3 does not apply (not read/write). Access is not granted, since read is not allowed. Therefore no access is granted.
3. 1p for: ACE1 does not apply (not group A), ACE 2 denies execute. Access is not granted.
4. 1p for: ACE1 does not apply (not execute), ACE2 does not apply (not group B), ACE3 allows execute. Access is granted.
5. Thread A is not granted execute. 2p for suggesting adding an entry like ACE3, but for Alice/Group A, before ACE2 (before or after ACE1). (1p for the entry, 1p for the correct position.)

10 **Sandboxing**

1. What is sandboxing? (2p)
2. Give two examples of applications or types of applications that typically use sandboxing. (2p)
3. Sandboxing is an example of an implementation of some of the OWASP security principles. Which security principles?
4. For each security principle you mentioned in question 2:
   1. Explain the principle
   2. How is sandboxing an example of this principle?

Total: 10p

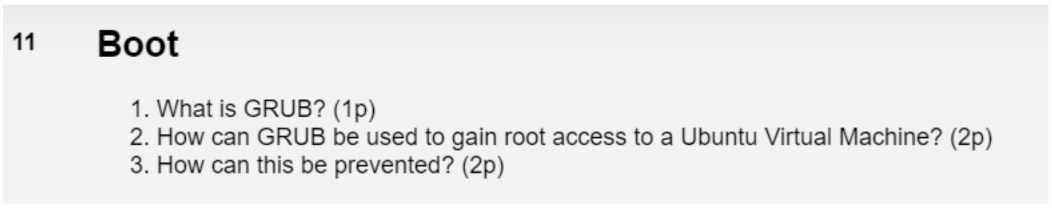
1. 2p for A sandbox is a security mechanism for isolating programs and processes, to make it more difficult for malware to spread on a system.
2. 1p (max 2) for each of
    a. Document readers
    b. Web browsers
    c. Most Windows Store Apps

    or other suitable types of applications.

3. 1p for each of:
    a. Security in depth
    b. Least privilege

    Other reasonable and well explained answers are accepted.

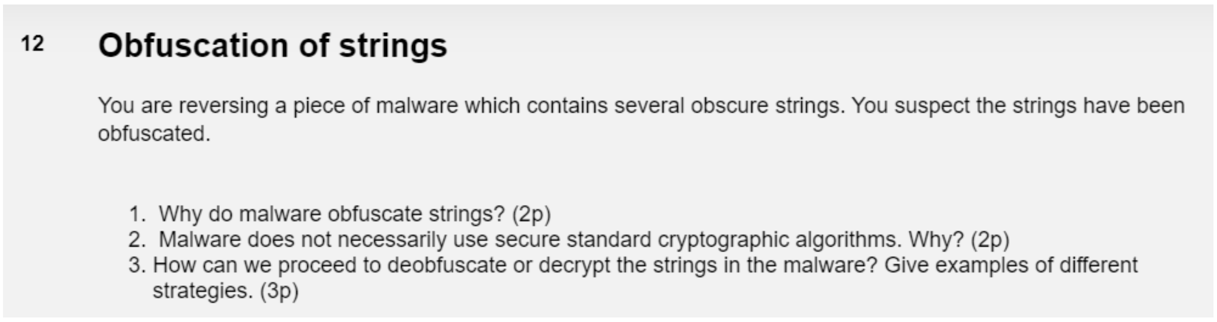
4. For each principle: 1p for explaining the principle, 1p for relating it to sandboxing.
    a. Security in depth
        i. Control mechanisms that approach risks in different ways are better than only one. A vulnerability in one part is then not enough to take control over the entire system.
        ii. If the sandboxed process gets compromised, the attacker will need an additional vulnerability to break out of the sandbox
    b. Least privilege
        i. No principal should have more privileges than needed at any time. Each principal should have access to the required objects and resources and be allowed a suitable set of actions.
        ii. The process running in a sandbox will have very limited access to the system, as sandboxed processes are stripped for most privileges



11 **Boot**

1. What is GRUB? (1p)
2. How can GRUB be used to gain root access to a Ubuntu Virtual Machine? (2p)
3. How can this be prevented? (2p)

1. 1p for: A bootloader, runs before the operating system is loaded after boot.
2. 2p for one of a) or b), 1.5p for c):
    a. Enter the GRUB menu during boot and select to boot into single-user/recovery mode. In the Ubuntu recovery menu, select the root entry. You can then change the password for later root access.
    b. Enter the GRUB menu during boot and select to edit the Ubuntu entry. Change the initial process from init to /bin/bash, then boot
    c. Boot via removable media
3. 2p for a), 1.5p for one of b) or c):
    a. Password protect GRUB
    b. Password protect the single-user/recovery mode (which will prevent 2a, but not b).
    c. Disable booting from removable media/BIOS password/encrypt harddrive



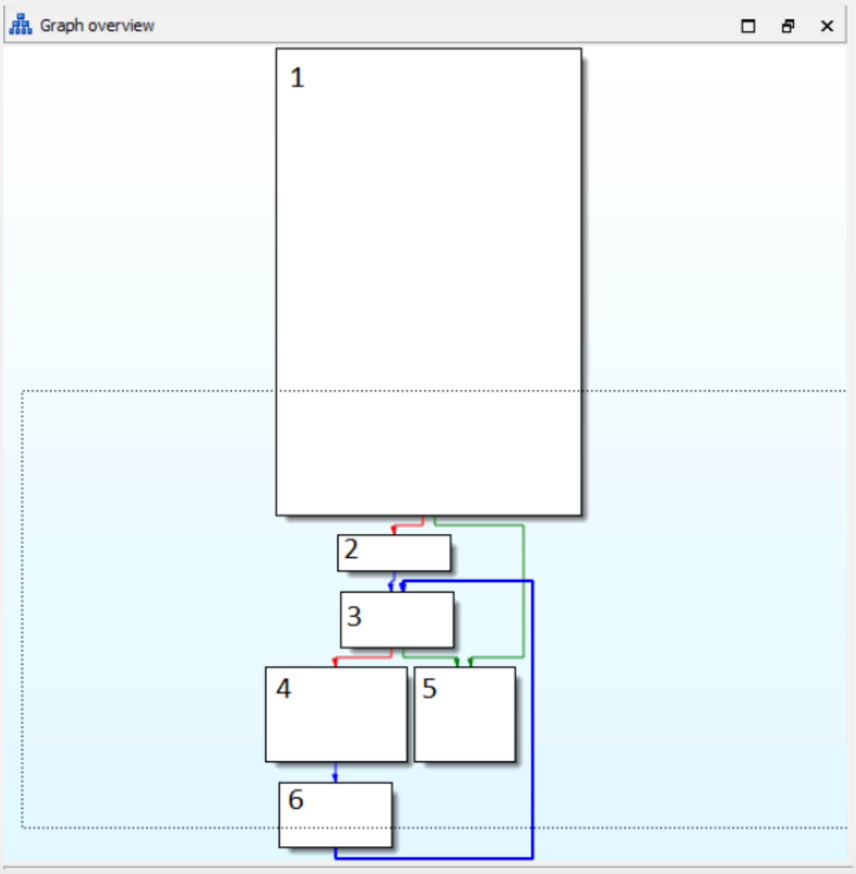
12 **Obfuscation of strings**

You are reversing a piece of malware which contains several obscure strings. You suspect the strings have been obfuscated.

1. Why do malware obfuscate strings? (2p)
2. Malware does not necessarily use secure standard cryptographic algorithms. Why? (2p)
3. How can we proceed to deobfuscate or decrypt the strings in the malware? Give examples of different strategies. (3p)

1. 1p for each of:
   a. To hide what it does/make reversing it harder
   b. To hide that it is malicious/avoid detection
2. 2p for: Malware may use cryptographic algorithms to hide or obfuscate, not necessarily to be cryptographically secure. Therefore, lightweight homemade cryptography can be more suitable than larger, more easily detected and recognizable cryptographic libraries/algorithms.
3. Possible strategies are: (1p for each)
   a. Reverse the decryption function and reimplement it.
   b. Set a breakpoint after the decryption function to see the cleartext when running the malware.
   c. Call the decryption function within the malware directly, without running the entire malware.

13 **Reversing in IDA**



The screenshot shows the graph overview of a function in IDA.
What can you tell about the function from the graph? Where does it start? Which code structures do you recognize? Where does it return? (5p)

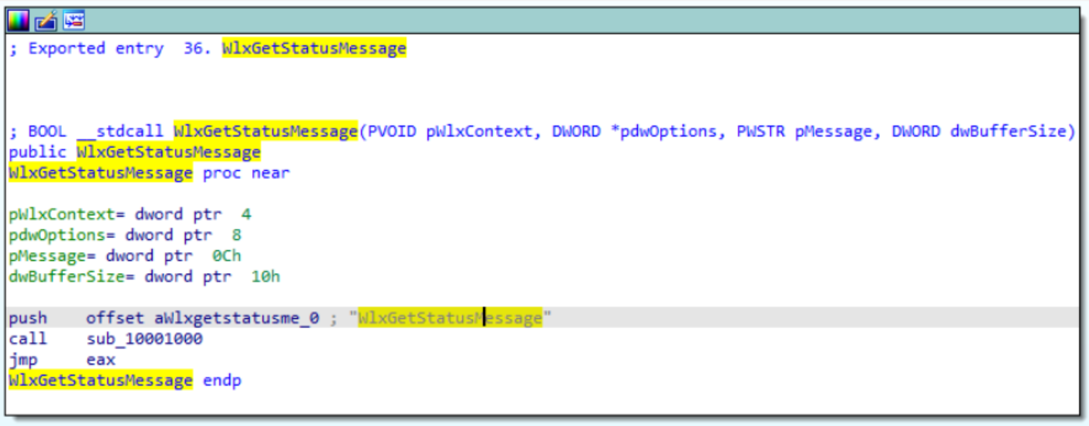
Note:
The numbers are inserted to make it easier to reference to the code blocks in your answer and have no other function.

1p for each of these:

1. The function starts in 1.
2. At the end of 1 there is an if-test that either goes directly to the return in 5 or through the rest of the function.
3. 3, 4 and 6 are part of a loop (while/for). At the end of 6 it goes to the beginning of 3.
4. At the end of 3, the end criteria for the loop is checked. If the end criteria is met, it goes to 5, otherwise it goes through the loop again.
5. The function returns in 5.

14 Malicious dll

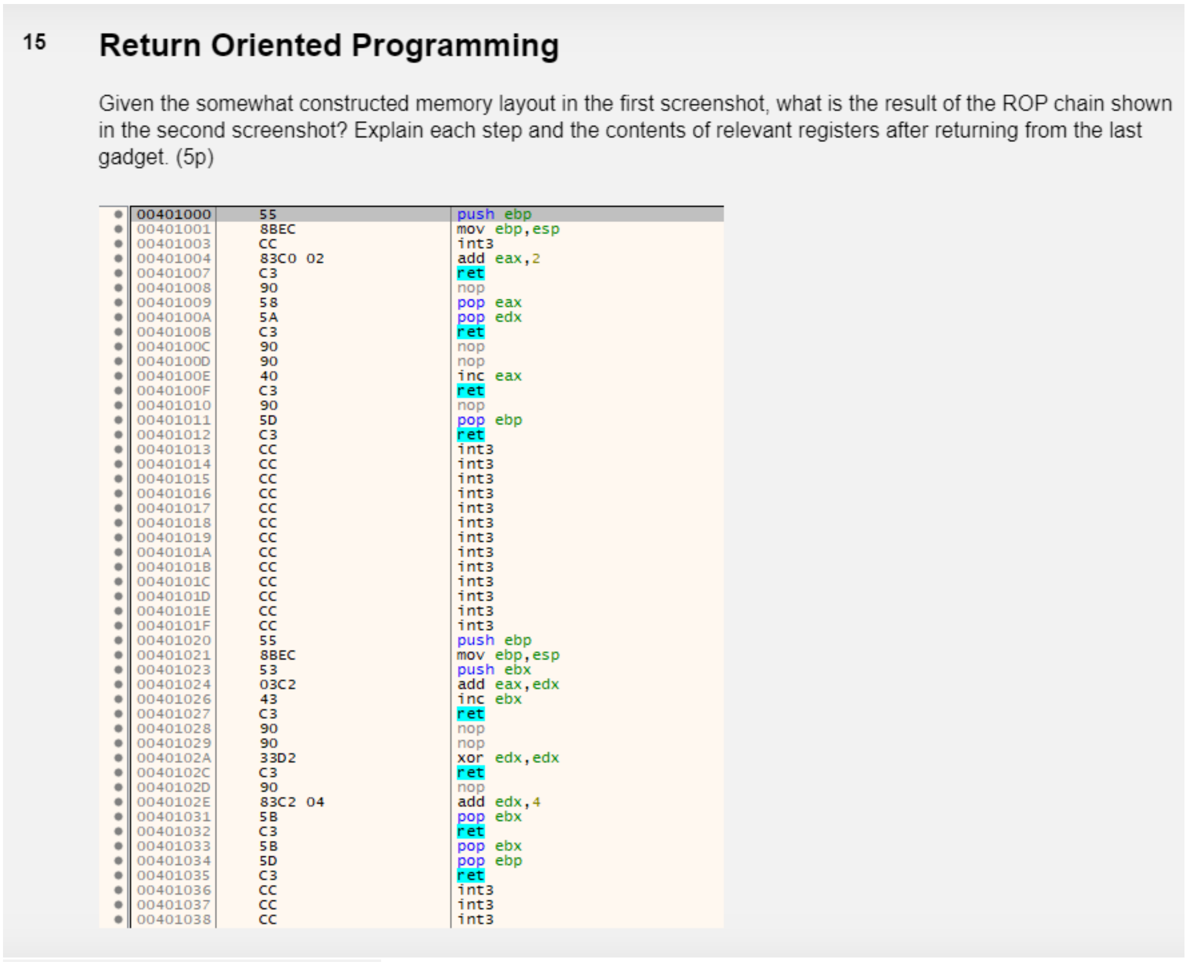
This screenshot shows a function from a malicious dll.



```
; Exported entry  36. WlxGetStatusMessage


; BOOL __stdcall WlxGetStatusMessage(PVOID pWlxContext, DWORD *pdwOptions, PWSTR pMessage, DWORD dwBufferSize)
public WlxGetStatusMessage
WlxGetStatusMessage proc near

pWlxContext= dword ptr  4
pdwOptions= dword ptr  8
pMessage= dword ptr  0Ch
dwBufferSize= dword ptr  10h

push    offset aWlxgetstatusme_0 ; "WlxGetStatusMessage"
call    sub_10001000
jmp     eax
WlxGetStatusMessage endp
```

1. What does this function do? (2p)
2. When is it called? (2p)
3. Where does it return? Explain. (2p)
4. What are its arguments? Explain. (2p)

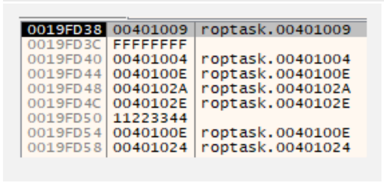
1. 2p for: We are told that this function is from a malicious dll. The function does a function call with the function name as parameter and there after jumps to the return value of the function call. This appears to be a hook, intercepting calls to a function with the same name in a legitimate dll. The function it calls is likely to return the address of the corresponding function in the legitimate dll.
2. 2p for: The malicious dll wants to be loaded by a process instead of a legitimate dll. This function will be called when the process that has loaded the malicious dll attempts to call the legitimate function with the same name in the legitimate dll that the process was supposed to have loaded.
3. 2p for: This function has no return. It jumps to the legitimate function with the same name, and that function will return directly to where this function was called from.
4. 1p for: The function arguments are pWlxContext, pdwOptions, pMessage, and dwBufferSize.
   1p for: These are not needed/used by this functions but are the parameters of the legitimate function with the same name. Because of the jmp eax instead of a call these parameters will be the parameters of the legitimate function as well when the process jumps there, as no new stack frame is created.



15 **Return Oriented Programming**

Given the somewhat constructed memory layout in the first screenshot, what is the result of the ROP chain shown in the second screenshot? Explain each step and the contents of relevant registers after returning from the last gadget. (5p)

```
00401000  55       push ebp
00401001  8BEC     mov ebp,esp
00401003  CC       int3
00401004  83C0 02  add eax,2
00401007  C3       ret
00401008  90       nop
00401009  58       pop eax
0040100A  5A       pop edx
0040100B  C3       ret
0040100C  90       nop
0040100D  90       nop
0040100E  40       inc eax
0040100F  C3       ret
00401010  90       nop
00401011  5D       pop ebp
00401012  C3       ret
00401013  CC       int3
00401014  CC       int3
00401015  CC       int3
00401016  CC       int3
00401017  CC       int3
00401018  CC       int3
00401019  CC       int3
0040101A  CC       int3
0040101B  CC       int3
0040101C  CC       int3
0040101D  CC       int3
0040101E  CC       int3
0040101F  CC       int3
00401020  55       push ebp
00401021  8BEC     mov ebp,esp
00401023  53       push ebx
00401024  03C2     add eax,edx
00401026  43       inc ebx
00401027  C3       ret
00401028  90       nop
00401029  90       nop
0040102A  33D2     xor edx,edx
0040102C  C3       ret
0040102D  90       nop
0040102E  83C2 04  add edx,4
00401031  5B       pop ebx
00401032  C3       ret
00401033  5B       pop ebx
00401034  5D       pop ebp
00401035  C3       ret
00401036  CC       int3
00401037  CC       int3
00401038  CC       int3
```



```
0019FD38  00401009  roptask.00401009
0019FD3C  FFFFFFFF
0019FD40  00401004  roptask.00401004
0019FD44  0040100E  roptask.0040100E
0019FD48  0040102A  roptask.0040102A
0019FD4C  0040102E  roptask.0040102E
0019FD50  11223344
0019FD54  0040100E  roptask.0040100E
0019FD58  00401024  roptask.00401024
```

5p for all steps correct and explained.

Minus 1p if the only mistake is that gadget 00401004 is "executed", since it is actually popped from the stack without being executed. 00401004 is "add eax, 2; ret", which would give the end result eax =7 instead.

Overview of the steps:

The first gadget is: 00401009

pop eax

pop edx

ret

The result of this:

eax = FFFFFFFF (=-1)

edx = 00401004

When it returns, it returns to 0040100E, not 00401004 which was popped into edx.

Next gadget: 0040100E

inc eax

ret

The result of this:

eax = 0

edx = 00401004

Next gadget: 0040102A

xor edx, edx

ret

The result of this:

eax = 0

edx = 0

Next gadget: 0040102E

add edx, 4

pop ebx

ret

The result of this:

eax = 0

ebx = 11223344

edx = 4

Next gadget: 0040100E

```
inc eax
ret
```

The result of this:

eax = 1

ebx = 11223344

edx = 4

Last gadget: 00401024

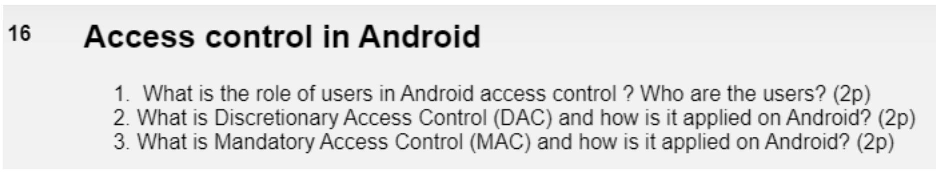
```
add eax, edx
inc ebx
ret
```

The result:

eax = 5

ebx = 11223345

edx = 4



16 **Access control in Android**

1. What is the role of users in Android access control ? Who are the users? (2p)
2. What is Discretionary Access Control (DAC) and how is it applied on Android? (2p)
3. What is Mandatory Access Control (MAC) and how is it applied on Android? (2p)

1. 2p for: Android is based on Linux. Each app is a unique Linux user, and the apps are separated (sandboxed) like users in Linux.
2. 2p for: Access control based on policies that refer to user identities. Can be configured and set by the users, and is what is shown as the rwx flags in a file listing.
3. 2p for: Access control based on policies that refer to security labels, and is enforced by the kernel. Part of SELinux. To allow access, the access must be allowed by both DAC and MAC.