

i Front Page

UNIVERSITY OF OSLO

The Faculty of Mathematics and Natural Sciences

Written examination TEK5510/9510

2024 2nd SEMESTER

Duration: December 5th at 9:00 to 13:00 (4 hours)

Permitted aids: None

It is important that you read this front page before you start.

Remember to explain your answers. Explanations are required to receive full score.

If you need to zoom in on screenshots, press 'Ctrl' and '+' at the same time. Use the '+' in numpad (the right hand part of the keyboard).

Good luck!

1 Security Principles and Terminology

1. What are the three main security principles? (3p)
2. Explain what these two types of malware do. How do they relate to the three main security principles:
 - a. Credential stealer (2p)
 - b. Backdoor (2p)
3. Explain the difference between **full disclosure** and **responsible disclosure** of vulnerabilities. (2p)

1. **1p** for each of:

- Confidentiality
- Integrity
- Availability

2.

a. **0.5p** for: A credential stealer steals user credentials.

1.5p for an explanation like: This is an attack on confidentiality, as the credentials should be secret. The credentials can be used to attack both integrity and confidentiality by using the credentials to read or change information. If the attacker changes the password, availability is attacked.

b. **1p** for: A backdoor is a malware that provides the attacker with remote access to the infected machine.

1p for an explanation like: With remote access to the infected machine, the attacker can attack confidentiality by reading information, integrity by changing information and availability by deleting information.

3. **1p** for: Full disclosure is the practice of publishing information on vulnerabilities as early as possible, with few or no restrictions on the information published.

1p for: Responsible disclosure is the practice of first notifying the vendor, allowing them some time to fix the vulnerability before disclosing it to the public.

2 Analyzing an Executable

Process Explorer - Sysinternals: www.sysinternals.com [WINDEV2401EVAL\User] (Administrator)

Process	CPU	Private Bytes	Working Set	PID	Description	Path	Integrity
explorer.exe	0.66	133,140 K	284,212 K	5436	Windows Explorer	C:\Windows\explorer.exe	Medium
SecurityHealthSystray.exe		1,860 K	9,584 K	7620	Windows Security notification...	C:\Windows\System32\SecurityHealthSystray.exe	Medium
cmd.exe		2,088 K	4,652 K	7964	Windows Command Processor	C:\Windows\System32\cmd.exe	Medium
conhost.exe		1,324 K	7,684 K	7972	Console Window Host	C:\Windows\System32\conhost.exe	Medium
EdgeUpdater.exe		1,400 K	8,120 K	7884		C:\Users\User\Downloads\EdgeUpdater.exe	Medium
msedge.exe	1.32	57,996 K	147,188 K	664	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Medium
msedge.exe		2,040 K	8,456 K	1896	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Medium
msedge.exe		18,420 K	50,360 K	3292	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Low
msedge.exe		9,956 K	29,160 K	3236	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Medium
msedge.exe		6,564 K	17,556 K	3024	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Untrusted
msedge.exe	3.55	25,124 K	44,748 K	1712	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	AppContainer
msedge.exe	< 0.01	26,188 K	51,188 K	8124	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	AppContainer
msedge.exe	< 0.01	24,312 K	68,832 K	2220	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	AppContainer
msedge.exe		6,664 K	17,776 K	5484	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Untrusted
msedge.exe		8,648 K	20,160 K	7932	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Untrusted
msedge.exe		7,740 K	23,532 K	8952	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Low
identity_helper.exe		9,932 K	34,388 K	8676	PWA Identity Proxy Host	C:\Program Files (x86)\Microsoft\Edge\Application\122.0.2365.32...	Medium
msedge.exe		20,460 K	30,644 K	8336	Microsoft Edge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	AppContainer
vmtoolsd.exe	0.33	22,900 K	40,596 K	7296	VMware Tools Core Service	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe	Medium
OneDrive.exe		46,656 K	108,076 K	1020	Microsoft OneDrive	C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe	Medium

Process Monitor - Sysinternals: www.sysinternals.com

Time of Day	Process Name	PID	Operation	Path	Result	Detail
9:41:53.4347132 AM	EdgeUpdater.exe	7884	RegQuery\Value	HKLM\SOFTWARE\Microsoft\Windows Defender\Symantec\SubmitSamplesConsent	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
9:41:53.4350135 AM	EdgeUpdater.exe	7884	RegSet\Value	HKCU\Software\Microsoft\Edge\SmartScreenEnabled\Default	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
9:41:53.4351515 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 0, Length: 4,096, Priority: Normal
9:41:53.4362058 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 4,096, Length: 4,096
9:41:53.4362355 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 0, Length: 4,096, Priority: Normal
9:41:53.4368405 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 8,192, Length: 4,096
9:41:53.4368733 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 4,096, Length: 4,096, Priority: Normal
9:41:53.4373749 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 12,288, Length: 4,096
9:41:53.4374555 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 8,192, Length: 4,096
9:41:53.4380119 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 16,384, Length: 4,096
9:41:53.4380279 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 12,288, Length: 4,096
9:41:53.4384963 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 20,480, Length: 4,096
9:41:53.4385285 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 16,384, Length: 4,096, Priority: Normal
9:41:53.4390172 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 24,576, Length: 4,096
9:41:53.4390361 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 20,480, Length: 4,096
9:41:53.4395319 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	SUCCESS	Offset: 28,672, Length: 2,216
9:41:53.4395475 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 24,576, Length: 4,096
9:41:53.4397940 AM	EdgeUpdater.exe	7884	ReadFile	C:\Users\User\Documents\Private\backup_ssh_keys.txt	END OF FILE	Offset: 30,880, Length: 4,096
9:41:53.4398356 AM	EdgeUpdater.exe	7884	WriteFile	C:\Users\User\AppData\Local\Microsoft\Edge\UpdateStatus.log	SUCCESS	Offset: 28,672, Length: 2,216
9:41:53.4438153 AM	EdgeUpdater.exe	7884	Process Create	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	SUCCESS	PID: 564, Command line: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe"

You are given the screenshots above from Process Explorer and Process Monitor.

- What can you tell about the process with PID 7884 from the screenshots? (5p)
- Based on the screenshots, what would be your next steps in the analysis, and why? (2p)

- 1p for each of these observations, max 5:

- The application EdgeUpdater.exe reads a registry value related to Windows Defender
- The application EdgeUpdater.exe changes a registry value related to Microsoft Edge
- The application EdgeUpdater.exe reads the file backup_ssh_keys.txt
- The application EdgeUpdater.exe writes to the file UpdateStatus.log
- The application EdgeUpdater.exe starts Edge (msedge.exe)

- 2p for any reasonable next steps. For example: 1p for Analyzing EdgeUpdater.exe in IDA, an additional 1p for what to look for in IDA. Or: 1p for Check network activity in Process Monitor or Wireshark and 1p for a description of other filters in Process Monitor. Or reading what is written to UpdateStatus.log.

3 A Vulnerable Function

Consider this code:

```
void copyName(char* str) {
    char buf[12];
    strcpy(buf, str);
}

int main(int argc, char** argv) {
    copyName(argv[1]);
    return 0;
}
```

1. What is this vulnerability called? (1p)
2. What is EIP? (1p)
3. Write out the stack frame for the copyName function. (3p)
4. Give an input that executes shellcode placed at 0x00414243. (2p)
5. Write out the resulting stack frame after providing your input (2p)
6. Explain DEP and ASLR. How would these mechanisms influence the result of providing your input? (4p)

1. 1p for: Stack buffer overflow
2. 1p for: The general register EIP is the instruction pointer, which contains the address of the next instruction to be executed.
3. The stack frame should be:

buf
buf
buf
Saved EBP
Saved EIP
argv[1]

3p can be given. Deduct 1p if saved EIP and EBP are switched.

4. AAAABBBBCCCCDDDDCBA
(strings are terminated with a zero byte)
1p for correct length to overwrite EIP, 1p for overwriting with the correct value.
If the input is given in hexadecimal, the same amount of points should be given, if this is explained.
5. The stack frame depends on the answer given in 4, but should be similar to:
2p if correct, 1p is given if partly correct.

buf	AAAA
buf	BBBB
buf	CCCC
Saved EBP	DDDD
Saved EIP	\00ABC
argv[1]	unchanged

6. 1p for explaining that DEP is marking some memory regions as non-executable.
1p for: Whether this matters to our example, depends on where 0x00414243 is and whether it is marked as non-executable.
1p for ASLR is Address Space Layout Randomization, which is a randomization of the addresses where applications and libraries are loaded in.
1p for: This means that we no longer know the address of the shellcode, so we do not what we will jump to.

4 Integer Computation

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. 95 + 40 as 8-bit unsigned integers
2. 95 + 40 as 8-bit signed integers
3. 32 - 42 as 8-bit unsigned integers
4. 32 - 42 as 8-bit signed integers

1. 95+40 -> 0101 1111 + 0010 1000 -> 1000 0111 -> 135
2. 95+40 -> 0101 1111 + 0010 1000 -> 1000 0111 -> -121 overflow
3. 32-42 -> 0010 0000 - 0010 1010 -> 1111 0110 -> 246 underflow
4. 32-42 -> 0010 0000 - 0010 1010 -> 1111 0110 -> -10

2p per exercise. Deduct a point on each exercise where the student has not converted to binary expressions.

5 Cryptography

We have used XOR in this course. What would happen with the plain text if you used each of the following keys in an XOR scheme?

1. All ones. (2p)
2. All zeroes. (2p)
3. The plain text itself. (2p)

We also used the ROT-x cipher, where x is the amount of letters in the alphabet you should shift your plain text by.

4. What is special about the case when x=13? (2p)
5. What OWASP Security principle tells you to not use your own crypto? (2p)

1. Flipping the bits, i.e. 0 -> 1, 1->0 (2p)
2. Stays the same, i.e. 0 -> 0, 1 -> 1 (2p)
3. The plain text is zeroed (2p)
4. It is symmetric, i.e. the same key can be used for decryption (2p)
5. 2p for Leveraging Existing Components. 1p if the student cannot remember the name of the principle, but explains that you should use components already made and tested instead of making your own.

6 Passwords

This is an excerpt from an etc/shadow file With the users Ole and Dole.

```
ole:$y$j9T$QIIDE623CYq2Gcdb.QZV..$9bAK412uGY5qCLj0P70ARyotcS1br9IantxlwFwqASA:20022:0:99999:7:::  
dole:$y$j9T$9ZVRTfCOGTgn0wfQQMoHn0$atlF2arJCD8WP/ZsvfITmALh10Ah3bYV0My1DDCgFdB:20022:0:99999:7:::
```

1. How can you tell what hashing algorithm has been used? (1p)
2. Ole and Dole have the same password. Why are the hashes different? (2p)
3. Where are passwords stored in Windows for a local user? (1p)
4. Ole and Dole also have the same password on a Windows computer. How could you detect that their passwords are equal? (2p)

1. 1p for the field after the first dollar sign or the field containing the y.
2. The passwords are salted (2p)
3. In Windows passwords are locally stored in HKLM\SAM (or the Security Account Manager Database) in the registry. It is not required to mention the registry key name 'V'.
4. 2p for Windows does not salt passwords, so the hash values will be equal.

7 Linux

1. Convert these triples of access rights to triples of octal numbers (1p per)
 - a. rw-rw-r--
 - b. rwxr-x---
2. Given the correct access right, what would the consequence of the following command be? (2p)
chown ole:scrimshankers tent.make
3. What does the s signify in the following Access triple rwsr-xr-x and explain why it is used? (2p)

1. 1p for each:

A) 664

B) 750

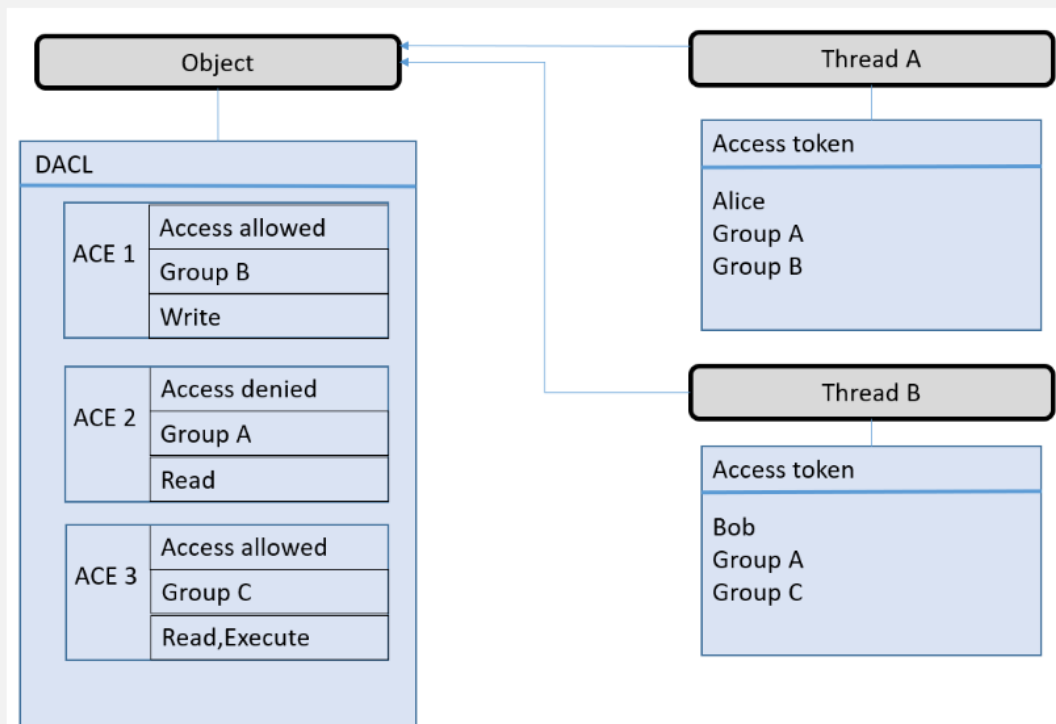
2. 1p for making ole the owner of the file

1p for making scrimshankers the owning group of the file

3. 1p for SUID

1p for the program will be executed with the permissions of the owner of the file

8 Windows



For each case below, what is the result of the request? Explain how each Access Control Entry influences the result.

1. Thread A requests read and write access to the object. (2p)
2. Thread B requests write and execute access to the object. (2p)
3. Thread B requests read and execute access to the object. (2p)
4. If Group A is removed from the access tokens for Thread A and Thread B, how would that influence the results above? (2p)

1. ACE 1 applies, and write access can be granted while read access is still needed. ACE 2 applies, and read access is denied, and the result is that **no access is granted**. The rest of the list is not checked. 2p for the right conclusion with explanation for each ACE.
2. ACE 1 does not apply, since it is for group B. ACE 2 does not apply, as read access is not requested. ACE 3 applies, which can give execute access, but write access is not given, and the end of the list is reached. Therefore **no access is granted**. 2p for the right conclusion with explanation for each ACE.
3. ACE 1 does not apply, since it is for group B. ACE 2 applies and denies read access. The rest of the list is not checked, and **no access is granted**. 2p for the right conclusion with explanation for each ACE.
4. 2p for an explanation like:
If Group A is removed from the tokens, ACE 2 no longer applies to the threads. This will not change the result of 1, but the procedure. ACE 2 does not deny read access, but ACE 3 does not grant read access to Thread A, and the end is reached without all access granted, so **no access is granted**.

This will not change the result of 2, as ACE 2 was not used to determine this result.

This will change the result of 3. ACE 2 no longer applies and does not deny read access. Both **read and execute access are granted** to Thread B by ACE 3.

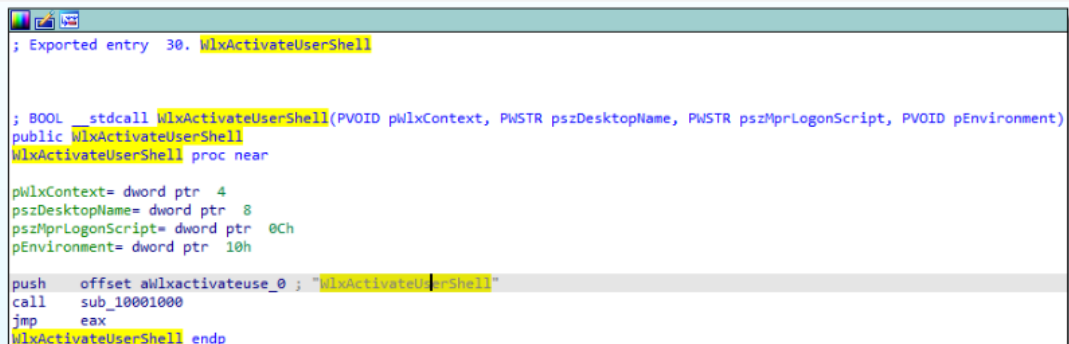
9 Securing Software

1. As we have seen, modern systems implement both ASLR, NX/DEP, and stack cookies to mitigate exploitation of vulnerabilities. One of the OWASP Security principles states that you should use multiple layers of such mitigations. Which security principle is this? (1p)
2. You should not use an administrator/superuser account as your main account. Several of the Security principles mention this, name one. (1p)
3. In the weekly exercises you used AFL++. What is this type of tool called? (1p)
4. For ranking threats we could use the DREAD-model. What is DREAD an acronym for? (1p for each, max 5)

1. 1p for “defense in depth”
2. 1p for “separation of duties”, “compartmentalize” or “least privilege”
3. 1p for fuzzer
4. 1p for each of the following:
 - a. Damage
 - b. Reproducibility
 - c. Exploitability
 - d. Affected users
 - e. Discoverability

10 Reversing

1. You are reversing a new piece of malware. You detect that the executable contains the string "This program cannot be run in DOS mode" twice. What does this indicate? (2p)
2. What is obfuscation? Give two examples of how malware uses obfuscation. (3p)
3. While reversing a piece of malware, you encounter the small function shown below which jumps to another function and never returns. Explain how this works when the function is called. What is the purpose of this function? (3p)



```
; Exported entry 30. WlxActivateUserShell

; BOOL stdcall WlxActivateUserShell(PVOID pWlxContext, PWSTR pszDesktopName, PWSTR pszMprLogonScript, PVOID pEnvironment)
public WlxActivateUserShell
WlxActivateUserShell proc near

    pWlxContext= dword ptr 4
    pszDesktopName= dword ptr 8
    pszMprLogonScript= dword ptr 0Ch
    pEnvironment= dword ptr 10h

    push offset aWlxactivateuse_0 ; "WlxActivateUserShell"
    call sub_10001000
    jmp eax
WlxActivateUserShell endp
```

1. 2p for: This indicates that the malware contains another executable that is to be dropped to the system by the malware. The malware seems to be a dropper, which has the purpose to install the next stage of malware on the victim.
2. 1p for: Obfuscation is to encrypt or scramble code or strings to make it difficult to read or reverse.
1p for each example like, max 2p:
 - Malware can use obfuscation to hide malicious-looking strings or function calls to appear as legitimate applications.
 - Malware can use obfuscation to hide a second executable that is to be dropped to the system, so that there would not be two strings from the PE header visible, like in the first question.
3. 1p for: By jumping to the address instead of calling the function, the stack frame is kept.
1p for: This avoids creating a new stack frame and makes the function that is jumped to return directly to where `WlxActivateUserShell` was called from.
1p for: This function (is from a malicious dll and) has the purpose to forward the call to the legitimate implementation of `WlxActivateUserShell`, which then will proceed as normal.

11 Hardware Security

1. Explain briefly these three terms:
 - a. Secure boot (1p)
 - b. Trusted boot (1p)
 - c. Measured boot (1p)
2. In Spectre and Meltdown, what was used as a side channel? (1p)
3. Explain briefly how the side channel was used to leak information in Spectre and Meltdown. (2p)

1. 1p for each of:
 - Secure Boot loads only trusted operating system bootloaders. It prevents loading of drivers and OS loaders that are not signed with an acceptable digital signature.
 - Trusted Boot checks the integrity of every component of the startup process before loading it.
 - Measured Boot sends the boot process measurements to a trusted server.
2. 1p for: Timing was used as a side channel. The technique is called *Flush and reload* or *Evict and reload*.
3. 2p for an explanation like: The victim writes a secret to an internal storage unit on the physical processor core, while the attacker uses the secret indirectly to speculatively read elements of an array. Training the branch predictor to make the victim access a secret is also an accepted explanation. Timing is used to

determine which array element that was stored when the speculative read was performed, since this array element will take the shortest time to read.

12

Return Oriented Programming

Make a stack layout for a ROP-chain that ends with the 0x04113774 gadget calculating $0xffffffff4 + 0x10$, putting the answer into `eax`. Show the values of each (relevant) register after each gadget.

Note: The following restriction applies: The primitive that lets you write the ROP-chain into the stack does not handle **0-bytes**.

(6p)

0x04113082

```
xor eax, eax;  
mov eax, esi;  
inc ecx;  
ret;
```

0x0411313C

```
pop edx;  
ret;
```

0x04113348

```
pop eax;  
inc edx;  
ret;
```

0x04113774

```
add eax, edx;  
ret;
```

0x04113A10

```
neg eax;  
inc eax;  
pop ecx;  
ret;
```

0x041137F0

```
dec eax;  
inc edx;  
ret;
```

Several possible answers. An example:

0x0411313c #pop edx

0xffffffff #edx = -1

0x04113348 #pop eax, inc edx

```

0xffffffff #eax = 0xffffffff, edx = 0
0x041137f0 #dec eax, inc edx. Eax = 0xfffffffffe, edx = 1
0x041137f0 #dec eax, inc edx. Eax = 0xfffffffffd, edx = 2
0x041137f0 #dec eax, inc edx. Eax = 0xfffffffffc, edx = 3
0x041137f0 #dec eax, inc edx. Eax = 0xfffffffffb, edx = 4
0x04113348 #pop eax, inc edx
0xffffffff #eax = 0xffffffff, edx = 5
0x041137f0 #dec eax, inc edx. eax = 0xfffffffffe, edx = 6
0x041137f0 #dec eax, inc edx. eax = 0xfffffffffd, edx = 7
0x041137f0 #dec eax, inc edx. eax = 0xfffffffffc, edx = 8
0x041137f0 #dec eax, inc edx. eax = 0xfffffffffb, edx = 9
0x041137f0 #dec eax, inc edx. eax = 0xfffffffffa, edx = a
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff9, edx = b
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff8, edx = c
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff7, edx = d
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff6, edx = e
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff5, edx = f
0x041137f0 #dec eax, inc edx. eax = 0xfffffffff4, edx = 0x10
0x04113774 #add eax, edx. eax = 0x4 = 4, edx = 0x10

```

6p for all steps correct and explained.

5p if only one wrong computation.

4p if the concept is understood and well explained, including how pop works, but computations are wrong.

1p in total if the concept of rop is understood, but the rest is wrong.

13

Android

Android uses several methods for sandboxing.

1. What is SELinux? (2p)
2. Explain how Chrome is sandboxed in Android. (2p)
3. How does Android make sure that the data for an app is isolated from other apps? (2p)

1. 2p for mandatory access control (MAC) enforced by the kernel

2. 2p for saying that each webpage is run in its own process

3. 1p for that each app is its own user and 1p for saying that it only has access to its own directory