## i Front Page

**UNIVERSITY OF OSLO**
**The Faculty of Mathematics and Natural Sciences**
**Written examination TEK5510/9510**
**2023 2nd SEMESTER**
**Duration: December 15th at 15:00 to 19:00 (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 Malware

There exists several types of malware. In this exercise consider these two types:

1. Ransomware
2. Keylogger

a. What are the three main security principles, and how do they relate to each of these two types of malware? (6p)
b. What type of attackers may use each of these types of malware, and what may the attackers achieve? (4p)

a

1p for each of the three security principles Confidentiality, Integrity and Availability.

For each of the three security principles 1p is given if the description of how the security principle relates to the malware types shows good understanding of the security principle.

There is not only one correct answer for how the principles relate to ransomware and keyloggers, but the candidate should at least mention that Availability is attacked by Ransomware and that Confidentiality is attacked by Keyloggers by leaking information.

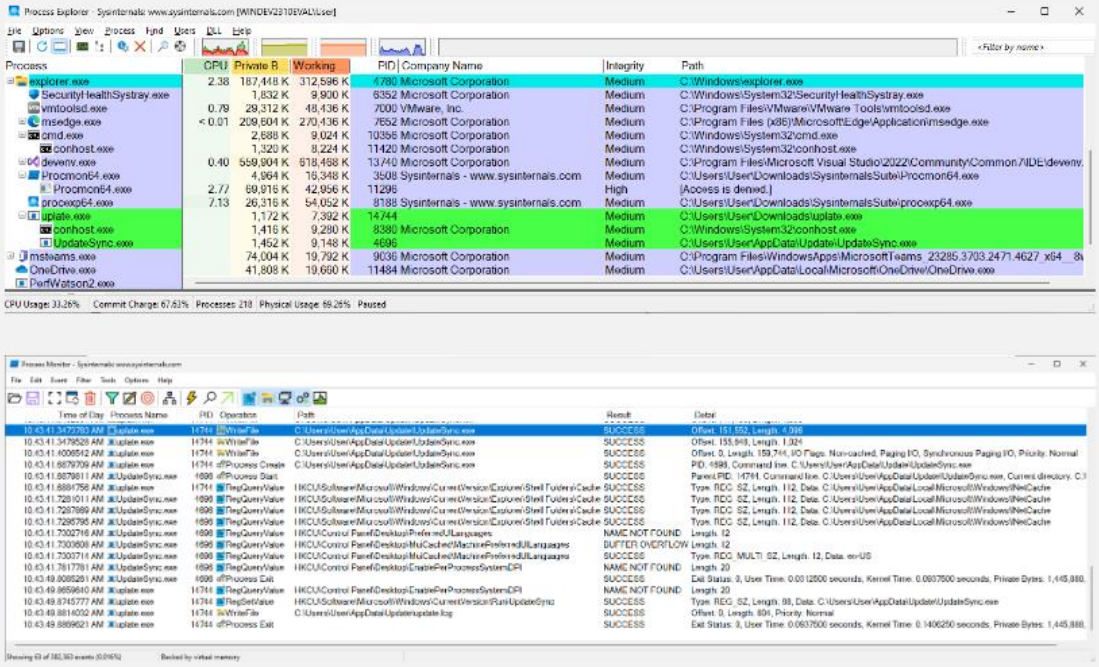
b

2p for Ransomware can be used by attackers that want money, which may be organized crimes or other attackers in need of money.

2p for Keyloggers can be used by attackers that are interested in information, including passwords. This may be advanced actors.

Other reasonable explanations should be accepted, if they show understanding of the malware types.

## 2 Analyzing an Executable







You are given screenshots from Process Explorer and Process Monitor.

a) What can you tell about the process with PID 14744 from the screenshots? (6p)

b) What does the tool RegShot do, and how could it be of use here? (2p)

c) Based on what you know about the process so far, what are your next steps, and why? (2p)

a

Up to 6 points for these observations:

The process name is uplate.exe, seen in both Process Explorer and Process Monitor.

It writes a file UpdateSync.exe to the folder C:\Users\User\AppData\Update, shown by the first lines with WriteFile events in Process Monitor.

uplate.exe then starts the process UpdateSync.exe, shown by the Process Start event in Process Monitor.

Both uplate.exe and UpdateSync.exe are missing Company Name in Process Explorer.

UpdateSync.exe is reading registry values, including values related to language, before exiting, shown in Process Monitor.

uplate.exe is setting a registry value HKCU\Software\Microsoft\Windows\CurrentVersion\Run\UpdateSync shown by the RegSetValue in Process Monitor.

uplate.exe is writing a file update.log in the folder C:\Users\User\AppData\Update shown by the last WriteFile event in Process Monitor.

b

1p for RegShot can take a snapshot of the registry and the file system at two different times and compare the contents of the snapshots. Thereby it gives an overview of changes in the registry and the file system between the two snapshots.

1p for Regshot can be used to take snapshots of the registry and filesystem before and after uplate.exe is run to get a complete overview of the changes that has been made to the file system and the registry.

c

1p for each reasonable described next step, max 2p.

It is not required to know what the registry value HKCU\Software\Microsoft\Windows\CurrentVersion\Run\UpdateSync is, or the other registry values shown in Process Monitor, and to propose checking what these values are related to should be accepted as a next step.

It should also be accepted to propose other filters in Process Monitor to see other events. Then the candidate should also mention briefly how the filters should be changed.

Also, further research in IDA, of both uplate.exe and UpdateSync.exe should be accepted.

3 **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. 100 + 29 as 8-bit unsigned integers
2. 100 + 29 as 8-bit signed integers
3. 23 - 25 as 8-bit unsigned integers
4. 23 - 25 as 8-bit signed integers

1. 100 + 29 -> 0110 0100 + 0001 1101 -> 1000 0001 -> 129
2. 100 + 29 -> 0110 0100 + 0001 1101 -> 1000 0001 -> -127
3. 23 - 25 -> 0001 0111 + 1110 0111 -> 1111 1110 -> 254
4. 23 - 25 -> 0001 0111 + 1110 0111 -> 1111 1110 -> -2

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

## 4 A vulnerable function

**Consider this code:**

```
void copyBuf(char* str) {
   char buf[8];
   strcpy(buf,str);
}

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

1. What is this type of vulnerability called? (1p)
2. Give an input that would make the program crash. Explain. (2p)
3. After running the program in a debugger you can see that the address of the buffer buf on the stack is 0x01CD3350. Draw the stack frame with addresses. (2p)
4. You want to execute some shellcode which is 16 bytes. Explain how this can be achieved. Draw the resulting stack frame after providing your input. (2p)
5. When trying to run the exploit you discover it does not work, as stack cookies and DEP are activated. How do these defense mechanisms work, and why do they stop our exploit? (3p)

1. Buffer overflow 1p
2. 1p if the student gives an input with a length of at least 16.
   1p if the student says that the reason for the crash is overwriting EIP.
3. The stack frame should be:
   0x01CD3350 - CopyBuf
   0x01CD3354 - Copybuf
   0x01CD3358 - Saved EBP
   0x01CD335C - Saved EIP

   2p can be given. Deduct 1p if the addresses are not correct. Deduct 1p if EIP and EBP are exchanged.

4. 2p can be given. 1p is given for overwriting the EIP with the address of the shellcode. 1p if the shellcode is placed so it can be executed. Since the shellcode is 16 bytes long it cannot be placed at the address of CopyBuf, since it will not fit. Could be placed after EIP, and return there, as shown in the example below, or return to the beginning of the input and change the last instruction of the shellcode before the saved EIP to jump over the EIP and continue execution. Example stack frame:

   0x01CD3350 - 0x90909090
   0x01CD3354 - 0x90909090
   0x01CD3358 - 0x90909090
   0x01CD335C - 0x01CD3360
   0x01CD3360 - Shellcode
   0x01CD3364 – Shellcode

5. 3p can be given.
   1p for explaining that DEP is marking some memory regions as non-executable.
   1p for explaining that stack cookies are inserted before the saved EBP and EIP and checked if altered before returning.

1p for explaining that the stack cookie would be overwritten and the program would crash or/and that the shellcode could not be executed on the stack.

## 5 Cryptography

| 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 |

After reversing a malware you find a homemade crypto to obfuscate some of the malware. The algorithm takes in a string "qgogbd" and XORs it with the hex value "0x010203040506".

1. Do the computation in bit values using the supplied table and give the output of the operation. (3p)

After XORing the two values, it takes the output and rotates it by 13.
2. Do this ROT-13 computation and give the output. (2p)

Someone suggests that they should have used a hashing algorithm to encrypt the information instead.

3. What is a hashing algorithm? (2p)
4. Would a hashing algorithm be suitable to encrypt the information? Why/why not? (3p)

1. 3p for pelcgb. Deduct one point if the computation is not shown in binary. Deduct one point for minor mistakes.
2. 2p for crypto. If the answer from the previous exercise was wrong, but the ROT-13 in this exercise is correct, full points are given.
3. 2p for Encoding method that by design should be infeasible to reverse.
4. 1p for no.
   2p for Because you can not reverse the encoding the information is not decryptable.

## 6 Passwords

1. In what files are the user account information and the passwords stored in **Linux**? (2p)
2. Where are passwords stored in **Windows**? (2p)
3. Explain what a salt is and what salts are meant to defend against. (2p)
4. How is salt used to store passwords in **Linux** and **Windows**, respectively? Explain. (4p)

1. 1p for /etc/passwd. 1p for /etc/shadow

2. 2p for In Windows passwords are stored in HKLM\SAM (or the Security Account Manager Database) in the registry. It is not required to mention the registry key name 'V'.

3.

1p for salts are a random string/value stored with the password.

1p for defending against rainbow/precomputed hash tables.

4.

2p for In Windows salts are not used to store passwords. The passwords are hashed without a salt, which means that the password hash for two users will be equal if the passwords are equal, making a Rainbow Table attack suitable.

2p for In Linux the passwords are salted and stored with the password hash in the /etc/shadow file

7 **Linux Access Control**

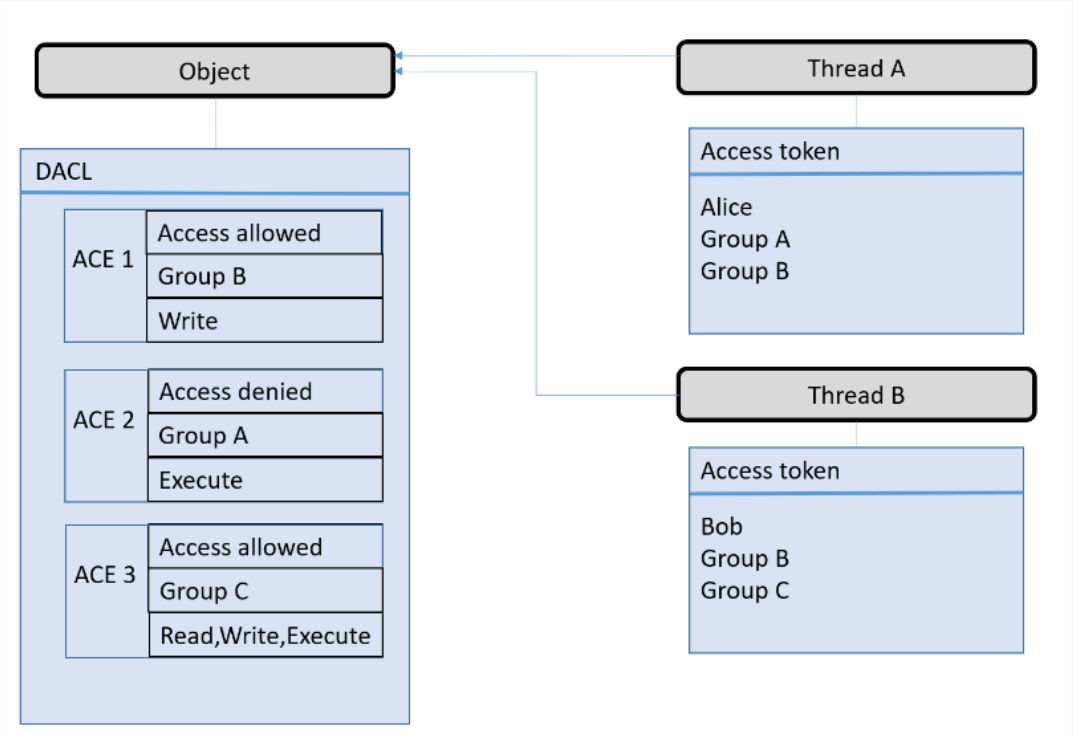
Consider this listing, provided by the command ls -l

-rw-rw-r-- 1 Ole student 1000 Nov 18 13.45 tek5510

1. Is tek5510 a file or a directory? Why? (2p)
2. Dole is also a member of the student group. What access to tek5510 does he have? Why? (2p)
3. Doffen is not a part of the student group. What access to tek5510 does he have? Why? (2p)
4. What does Ole have to do to give Doffen the same access to tek5510 as Dole? Also write any commands Ole would have to run. (2p)

1. 1p for saying it's a file. 1p for saying its because the first symbol in the ls -l output is a "-".
2. 1p for He has read and write access. 1p for Because the second triple in the access control is rw-.
3. 1p for He has read access. 1p for Because the last triple is r--.
4. 1p for He can give write access to others. Alternatively, if Ole has permission to do so, he can add Doffen to the student group.
   1p for chmod o+w tek5510 or chmod 666 tek5510. Alternatively, if the student wants to add Doffen to the student group and has permission to do so, usermod +aG student Doffen.

8 **Access Control in 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 write access to the object. (2p)
2. Thread B requests read and write access to the object. (2p)
3. Thread A requests read and write access to the object. (2p)
4. Thread A requests write and execute access to the object. (2p)
5. What would be the consequences of removing ACE2 from the list? (2p)

1.

A requests write. ACE1 applies and grants write. Access is granted. 2p for the right conclusion with explanation for each ACE.

2.

B requests read and write. ACE1 applies and grants write. Read is still needed, so keeps checking. ACE2 does not apply. ACE3 applies and grants read. Access is granted. 2p for the right conclusion with explanation for each ACE.

3.

A requests read and write. ACE1 is applied and grants write. Read is still needed. ACE2 denies execute, which is not asked for. Checking continues. ACE3 does not apply to thread A. The end of the list is reached without read access granted. Therefore no access is given. 2p for the right conclusion with explanation for each ACE.

4.

A requests write and execute.  ACE1 is applied and grants write. Execute is still needed. ACE2 denies execute, and no access is given. 2p for the right conclusion with explanation for each ACE.

5.

2p for an explanation like: ACE2 denies execute access for group A. ACE3 grants execute access for group C, and ACE3 is the only way to get execute access. Removing ACE2 will not change the access for the two threads shown, since none of the tokens have both group A and group C. If a thread had a token with both group A and C, removing ACE2 would mean the thread would get execute right, instead of being denied execute by ACE2.

**9** **Reversing**

a.
Four possible ways to start when reversing a piece of software are:

1. The main function
2. Strings
3. Imports
4. Exports

What are the benefits of each alternative, and when are they suitable? (8p)

b.
You have reversed one version of a software before. How can the reversing work on a previous version of a software be useful when reversing a new version of the same software? (2p)

a

2p for a description showing an understanding of each of the starting points. For example:

1. The main function is where an application starts. Not suitable for reversing libraries without a main function. If the executable is small, it can be a good place to start. If the executable is large, it may be too much happening before reaching the function you are interested in.

2. Strings may contain human readable information giving an idea of what the executable is doing and which functions that are related to what you are interested in. Using cross references to find the functions that use the strings you find interesting is a good way to find the relevant code. If strings are obfuscated or there are few strings, this strategy may not be the best choice. For obfuscated strings, one possible strategy is to find the functions reading the strings in order to locate the deobfuscation function and deobfuscate the strings.

3. Imports show functionality that is imported from other libraries. This can give information about what the software is doing. And cross references to the imported functions help finding the parts of the software related to that functionality.

4. Exports can be useful when analyzing a library. The exported functions show the functionality this library contains. Some of these exported functions may be good starting points for an analysis.

b

2p for an explanation like: There are tools for binary comparison like BinDiff that can be used to help finding the changes from one version to a new version. These tools compare the binaries and match functions in the two versions. If you have reversed interesting functions in the old version, you can identify the corresponding functions in the new version and use your understanding from the old version to reverse the new version. This works best if there are some modifications, but not a total restructuring of the software between the versions.

## 10 Hardware and virtualization

1. What is virtualization, and what can it protect against? (3p)
2. Explain the two types of hypervisors/virtual machine monitors. (2p)
3. Explain how attacks like Spectre and Meltdown are affected by virtualization. (3p)

1.

1p for Virtualization is using a hypervisor or virtual machine monitor to create and control virtual machines and providing an environment for virtual machines to run. Answers like this also give 1p: Virtualization can be used to run virtual machines inside a host machine.

2p for mentioning some of the reasons for using virtualization that have been described in this course:

- Efficient use of hardware and resources
  - Improved management and resource utilization
- Improved security
  - Malware can only infect the VM
  - Safe testing and analysis of malware
  - Isolates VMs from each other
- Distributed applications bundled with OS
  - Allows optimal combination of OS and application
  - Ideal for cloud services
- Powerful debugging
  - Snapshot of the current state of the OS
  - Step through program and OS execution
  - Reset system state

2.

1p for Type 1 hypervisors run below the operating system, are usually very small, have less attack surface and high performance. Hardware support can be an issue.

1p for Type 2 hypervisors run on a host operating system and uses the services of the host for e.g. memory management, scheduling, and drivers. Type 2 have a performance penalty, but are easier to use. An example is the Oracle VirtualBox, which probably most students have used in this course.

3. 3p for an explanation like: Spectre and Meltdown work by leaking information between two processes running on the same physical processor core by using time as a side channel. The attacks can be used to break the isolation provided by virtualization between processes running on the same physical processor core, if it is vulnerable to Spectre and Meltdown. Virtualization does not protect against Spectre and Meltdown.

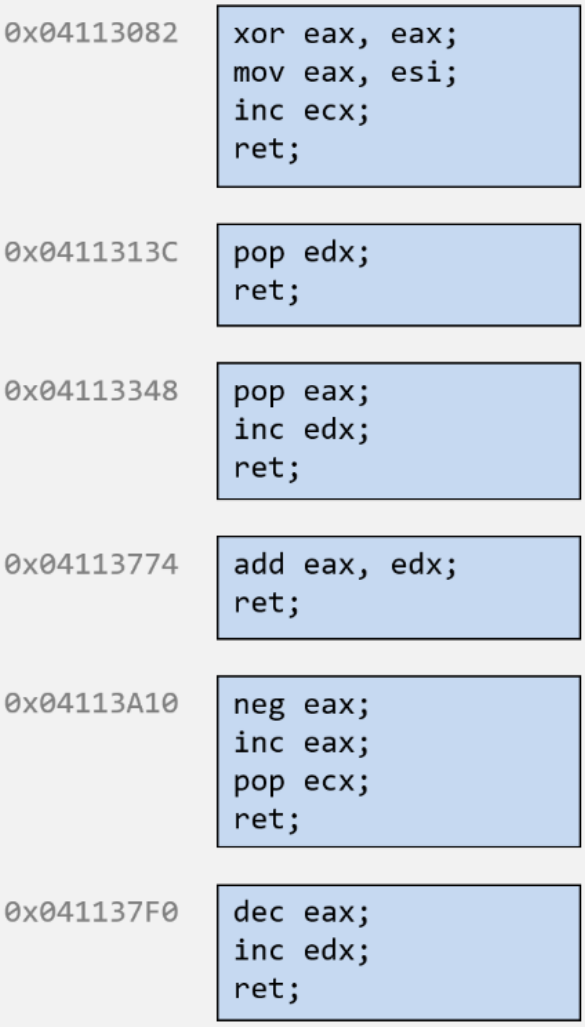
## 11 Securing Software

You work at the security department for a company making an online banking application. They have decided to add a social media component to the application where their customers can interact and share their life with each other. You argue that this is a bad idea, and that it breaks with the OWASP security principle **Economy of mechanism**.

1. What does this principle say? (2p)
2. You are tasked with testing the security of the component. You start doing manual code review. What is this? What are the drawbacks of manual code review? (3p)
3. Because of the drawbacks discussed in the previous exercise you decide to start fuzzing the application. What is fuzz testing and give an example of a fuzzing software you could use? (3p)
4. When testing the software you find out that if the component crashes you suddenly gain admin privileges. Which security principle may not have been followed by the developers when writing the software? (2p)

1. 2p for More complex software often has more vulnerabilities. Keeping the code, design and implementation simple the attack surface is reduced.
2. 1p for Manual code review is a programmer and/or security experts manually reviewing code to look for vulnerabilities.
   1p for slow and/resource intensive
   1p for often limited to a small part of the code
3. 2p for mentioning giving unexpected and invalid input to a program and monitoring for exceptions.
   1p for giving any fuzzer. Examples mentioned in class are AFL, AFL++, Sage, Peach, Project Springfield, OSS-fuzz.
4. 2p for Fail Safe.

## 12 Return Oriented Programming




What are the results of executing this ROP chain? Explain the values of the relevant registers after each line. (10p)

0x041137F0
0x0411313C
0x4245414D
0x04113348
0xFFFFFFFB
0x04113A10
0x04113774
0x04113774

5 of these addresses are instructions that are executed. 2p for explaining each of these correctly. If the candidate states that both the two last 0x04113774 are executed even though one of them is popped into ecx, 2p are lost.

| | |
|---|---|
| 0x041137F0 # dec eax; inc edx; ret | Irrelevant, unknown values of eax and edx. |
| 0x0411313C # pop edx; ret; | edx = 0x4245414D |

| | | |
|---|---|---|
| 0x4245414D | # value for edx | popped |
| 0x04113348 | # pop eax; inc edx; ret | eax = 0xFFFFFFFB, edx = 0x4245414E |
| 0xFFFFFFFB | # value for eax | popped |
| 0x04113A10 | # neg eax; inc eax; pop ecx; ret | eax = 6, ecx = 0x04113774, edx = 0x4245414E |
| 0x04113774 | # dummy value for ecx (equals pop edx; inc edx; ret;) | popped |
| 0x04113774 | # add eax, edx; | eax = 0x42454154, ecx = 0x04113774, edx = 0x4245414E |

13 **Android Security**

**1. Hardware Based Security**

To use an isolated environment for executing security critical code is a hardware based security mitigation. Mention three examples of security critical functionality where Android uses this. (3p)

**2. Exploit Mitigation on Android**

a) Explain these two terms:
   i) Execute-Only Memory (2p)
   ii) ASLR (2p)

b) Explain the exploit mitigation effect of Execute-Only Memory and ASLR (3p)

Answers

1

The candidate gets 1 point for each of the following:

- Lock screen passcode verification
- Fingerprint template matching/fingerprint matching
- KeyStore management

2

a

i) Execute-Only Memory means that a section of the memory is marked as only executable, not readable or writeable. Up to 2 points can be given.

ii) ASLR is Address Space Layout Randomization, which is a randomization of the addresses where applications and libraries are loaded in. Up to 2 points can be given.

b Execute-Only Memory prevents an attacker from writing code into an address and executing it afterwards, while ASLR prevents an attacker from knowing the address of code the attacker wants to reuse. The location of the code must be leaked before reuse. This often requires two vulnerabilities: One read primitive, and one write primitive.

Up to 3 points can be given.