UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	TEK 5510 — Security in operating systems and software.
Day of examination:	Tuesday November 27th 2018.
Examination hours:	09:00 – 13:00.
This problem set consists of 11 pages.	
Appendices:	None
Permitted aids:	Dictionary.

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

Answer all questions in this examination paper.

Answers can be written in English or in Norwegian.

Be concise when answering. It is often sufficient to write a single sentence, or at most a few sentences, to describe the concept that each sub-question asks for.

Question 1. Cryptography

a) Explain what public and private cryptographic keys are and give an example of an application of them. (2p)

b) What is salt, and how does the use of salt increase password security? (3p)

Answer

a) **1p** A private key is known only by the owner, used to decrypt messages.
1p The public key is publicly known, used to encrypt messages to the owner of the corresponding private key.
b) **1p** A salt is a string that is appended or prepended to the password before hashing it.
1p This is done to make precomputed rainbow tables unusable. The hash depends on the salt value, so one would need a rainbow table for the specific salt value to use the table for cracking.
1p The salt is known, so password salting does not increase the difficulty of cracking one password through bruteforcing.

Question 2. Secure development and Web security

You are hired by the university to design and develop a new web application for handing in and receiving markings and grades of home exams. Before

(Continued on page 2.)

you start you do a threat analysis of the project.

a) What are the assets and threats that needs to be considered in this project? (2p)

b) Explain the terms Confidentiality, Integrity and Availability. How do these principles relate to the assets and threats? (3p)

c) Use the OWASP principles of secure development to describe how you will design and develop the web application. (5p)

d) Explain the terms *SQL injection* and *Cross site scripting*. Will these types of attack be a threat to your web application? (2p)

Answer

a) **1p** The assets are the students' markings and grades (and possibly other personal data).
1p Possible threats are that the service becomes unavailable to the students, that the students' results fall in the wrong hands, and that the results are tampered with.
b) **1.5p** Confidentiality concerns protecting information from falling into wrong hands. Integrity concerns unauthorized tampering with data. Availability concerns that the service should be available to authorized users.
1.5p Relation to the assets and threats: the service becomes unavailable to the students (availability), that the students' results fall in the wrong hands (confidentiality), and that the results are tampered with (integrity).
c) **1p** for each security principle from the list below that is stated with a reasonable description of how it relates to the web application. Maximum 5p.

- Minimize attack surface
- Secure default settings
- Least privilege
- Defense in depth
- Fail securely
- Don't trust services
- Separation of duties
- Keep security simple
- Fix security issues correctly
- Avoid security by obscurity

d) **0,5p** SQL injection means that, because of missing input validation, additional SQL requests can be injected to do additional/unintended database queries.

(Continued on page 3.)

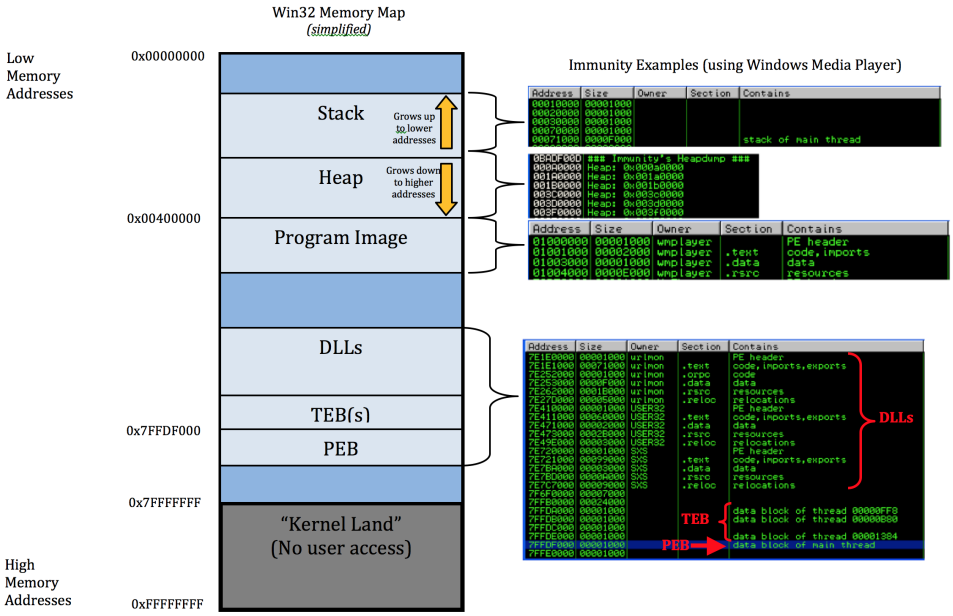


Figure 1: Figure from: www.securitysift.com/windows-exploit-development-part-1-basics/

0,5p Cross site scripting: Attacker can, because of missing input validation, provide any html element, for example to rewrite the document content or to redirect the page.
1p Good input validation can prevent both attacks. Limitation of the user input to pre-defined choices/commands for example. (Whether they apply to the web page depends on the design of the web application, for example whether the web page has a database)

Question 3. Analyzing executables
Figure 1 shows an executable file mapped into memory.

a) Explain the terms *executable* and *process*. (2p)

b) Explain the terms stack, heap, program image, DLLs, and Kernel Land that are shown in Figure 1.(4p)

c) Why can it be useful to find the strings contained in an executable? How can you find the strings? (2p)

d) In some cases the strings are *obfuscated*. What does this mean? How can you then find the actual strings? (2p)

Answer
a) **1p** An executable is a file that contains data and instructions on a format that can be executed.
1p A process is a running executable.
b) **1p** The **stack** is used for local variables and parameters for functions,

(Continued on page 4.)

and to help control program flow.
1p The **heap** is used for dynamic memory during program execution.
1p The **program image** is the executable file loaded into memory.
0.5p The **DLLs** are the libraries/dll-files that are loaded into memory (dynamically linked libraries).
0.5p Kernel Land is the most protected part of the memory, where for example core operating system components are running. As opposed to **User Land**, where normal user processes are run.
c) **1p** The strings can give information about what tasks an executable can perform. Error messages or debug strings, for example, or library names/file names.
1p The strings can be found either through the strings-view in IDA, the Strings tool, or by similar types of tools.
d) **1p** If the strings are obfuscated, they are encrypted or scrambled/encoded in order to make them difficult for an analyzer to decode. The strings are typically decrypted into clear text by a decryption function before use.
1p The actual strings can be found by feeding the encrypted strings to the decryption function. Either by writing a separate program containing a reimplementation of the decryption function, or by calling the decryption function in the executable directly. One of the alternatives is sufficient. (Or through logging during execution of the entire executable which only decrypts the strings actually used in that specific run.)

Question 4. Assembly
Consider the screenshot in Figure 2.

a) Explain the contents of the red, orange and green rectangles. (3p)

b) The next instruction to be executed is a call. To which function, and with which parameters? (2p)

c) Explain in detail how the data at address 0x00F8F968 on the stack was placed there. Which instructions, registers, and addresses were involved? (3p)

Answer
a) **1p** The red rectangle contains the addresses in memory of the corresponding bytes in the orange rectangle.
1p The byte values in the orange rectangle are the op codes for the instructions that are to be executed when the process is running.
1p The green rectangle contains the assembly instructions (human readable form) corresponding to the op codes.
b) **1p** The function to be called is strncmp (string comparison).
1p The arguments are pointers to the strings "-w" and "-r", and the number 2.
c) At 0x00F8F968 on the stack is the pointer (address) to the string "-w", 0x013B95A2. This address was pushed to the stack by the previously executed instruction *push dword ptr DS:[ESI+4]*. ESI = 0x013B9528B and ESI+4 = 0x013B952B. At address ESI+4 shown in blue in the memory dump at the lower left is the address 0x013B95A2, which points to the string "-w".

(Continued on page 5.)

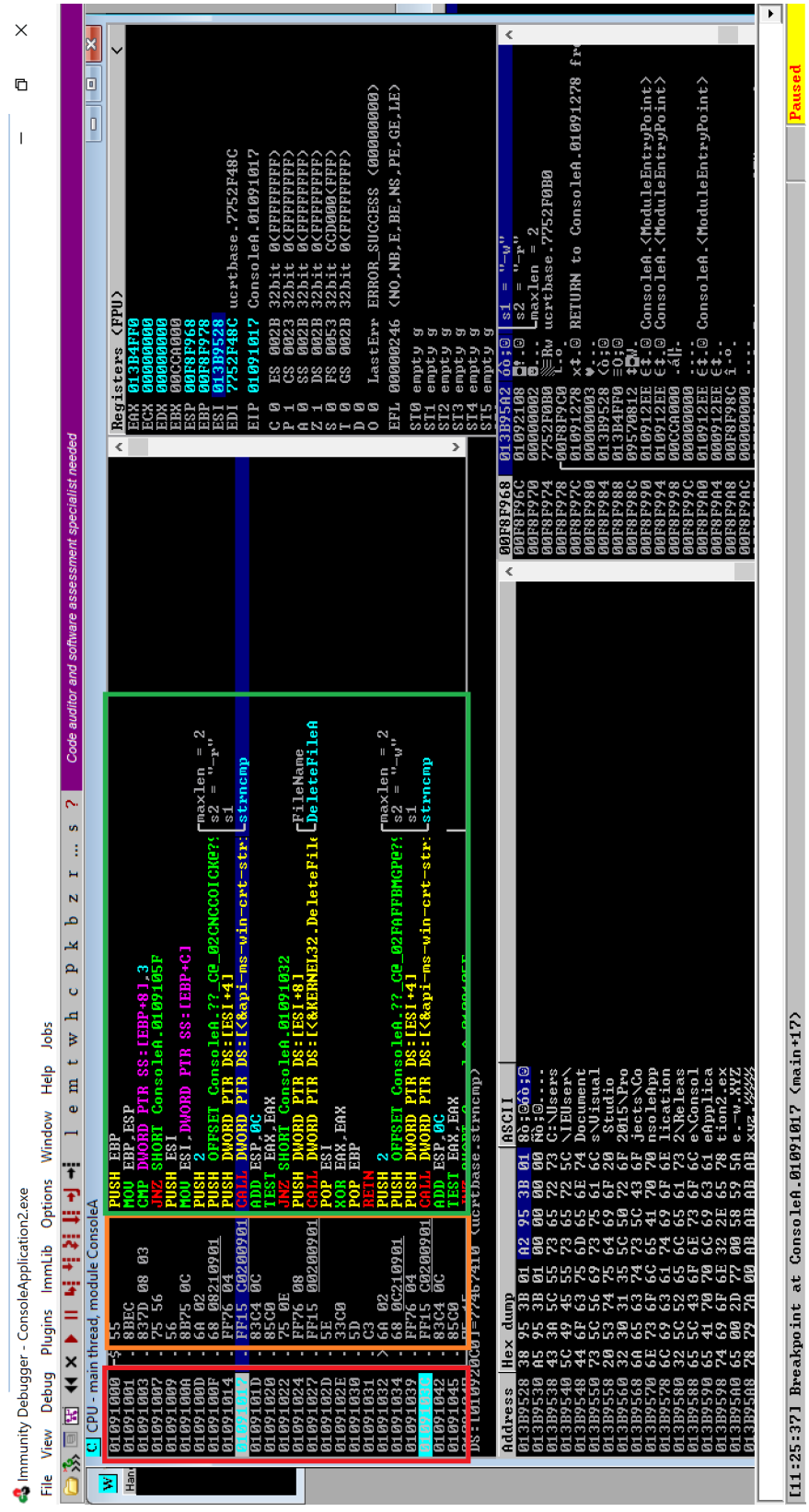


Figure 2: Immunity Debugger

(Continued on page 6.)

1p for stating that the previous instruction pushed the value onto the stack. **2p** for explaining the role of ESI, using the register value and the memory dump.

Question 5. Malware and vulnerabilities

a) Give a short explanation of typical steps involved when a malware takes control over a computer. (2p)

b) Malware are created for different purposes by different types of attackers. Give examples of three types of attackers and what they typically aim to achieve with their malware. (3p)

c) What is a vulnerability? Give two examples of different types of vulnerabilities. (3p)

d) "Always update your software to the latest version as quickly as possible" is a common security guideline. Explain why this is important. (2p)

Answer

a) **1p** A *vulnerability* is exploited to gain control over the execution flow. **1p** Thereafter a *payload* is executed, containing the actions that the attacker is interested in executing (e.g. encrypting the disk if it were ransomware). In other words: The vulnerability is the way in. The payload is the reason that the attacker was interested in getting in.

b) **1p** for each example, maximum 3p. Possible examples:

- Script kiddies - because they can/for fun/...
- Hacktivists - to promote ideological or political agenda
- Organized crime - for money
- Advanced persistent threats - targeted attacks

c) **1p** A vulnerability is an exploitable bug. **2p** for giving two examples of vulnerabilities. 1p for each. Possible examples: Buffer overflow, use after free, double free,...

d) **2p** The latest version contains the latest security patches. After a vulnerability of a piece of software has been published, anyone could use that vulnerability to attack anyone who has not yet updated to a fixed version. It is a bad idea to leave oneself vulnerable to publicly known attacks.

Question 6. Exploitation

a) Explain the terms *local exploit* and *remote exploit*. (1p)

b) What is meant by *privilege escalation*? (1p)

c) What is a Return Oriented Programming (ROP) chain? (2p).

d) What is Address Space Layout Randomization (ASLR)? (2p)

Answer

a) **1p** Local exploit is an attack that is done locally on a machine,

(Continued on page 7.)

through physical access. Remote exploit is an attack delivered over a network/without physical access to the computer.
b) **1p** Privilege escalation means to increase ones privileges on a computer, for example from normal user privileges to administrator privileges.
c) A ROP chain consists of a sequence of *gadgets*. Each gadget is a (often short) sequence of bytes that are interpreted as instructions, ending with a return. The addresses of the gadgets are placed in sequence in memory, such that the return at the end of one gadget makes the instruction pointer return into the next gadget. In total the gadgets perform a sequence of instructions, for example to make a piece of memory executable, and to execute a payload located there. The addresses in a ROP chains do not need to be stored in executable memory to work, but the addresses must point to executable memory.
d) ASLR is a security mechanism that aims to make the addresses of loaded modules (and thereby gadgets in the modules) less predictable. When a module is loaded into memory, the address on which it is loaded is randomized, instead of always loading it at the same address.

Question 7. Computing with integers

Show the computations with binary numbers and explain the results. With unsigned 8-bits integers:

a) 255 + 3 = ? (1p)

b) 126+4 = ? (1p)

With signed 8-bits integers:

c) 126+4 = ? (1p)

Answer

To get full score, the binary calculations must be included. a) **1p** 255+3 = (in binary:) 1111 1111 + 11 = 1 0000 0010 =(8bits unsigned:) 10 =2.
In 8bits integers, the first bit overflows, and the result is 255+3 = 2.
b) **1p** 126+4= (binary:) 0111 1110 + 100 = 1000 0010 = (8bits unsigned:) 130.
No overflow. Just as in normal mathematics.
c) **1p** 126+4= (binary:) 0111 1110 + 100 = 1000 0010 = (8bits signed:) -126
The first bit is used for sign. If the first bit is 1, the magnitude is found by the two's complement of the last 7 bits.

Question 8. Windows

a) How is a token used in the Windows access control model? (2p)

b) Explain the concept of User Account Control (UAC). Why does it exist? What is it intended to mitigate? How is it related to tokens? (3p)

You are asked to investigate a piece of malware, Lab03-03.exe. You decide to run it in a virtual machine first to get an overview of the executable. Figure 3 contains a screenshot of Process Explorer during the execution of

(Continued on page 8.)

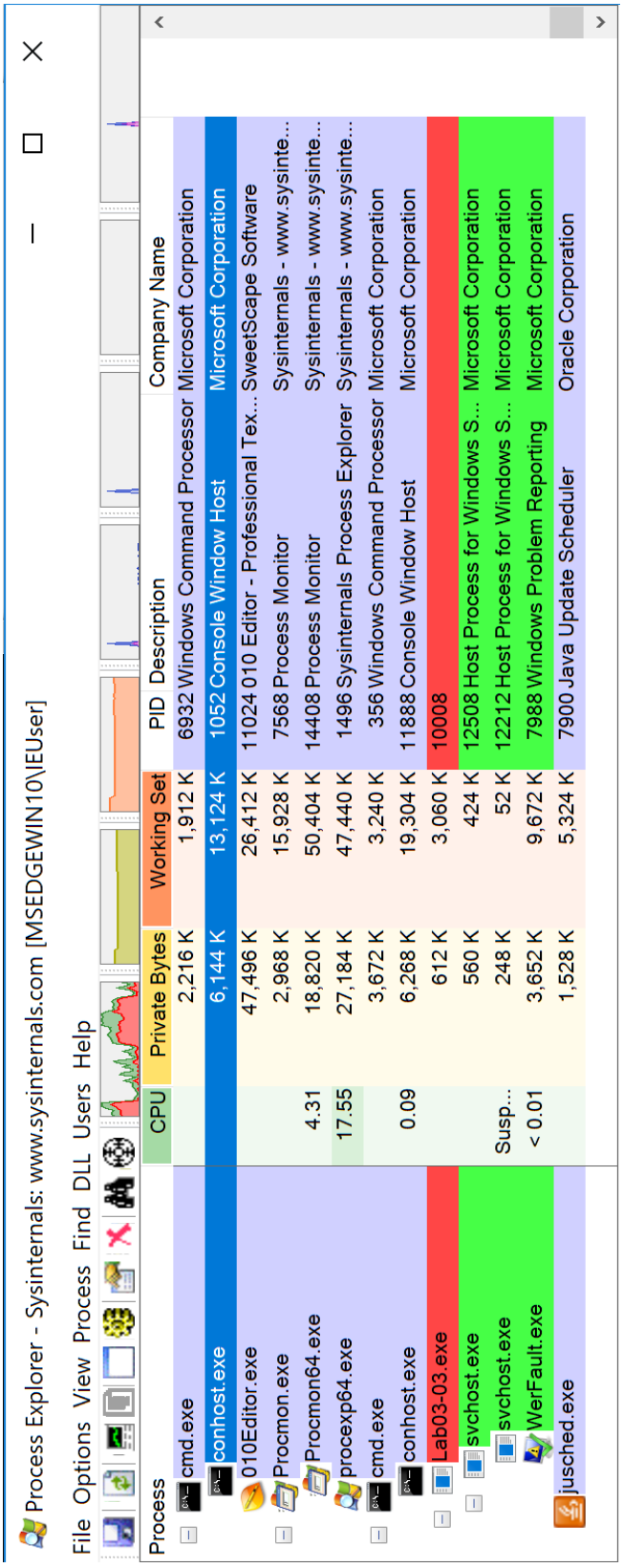


Figure 3: Process Explorer

(Continued on page 9.)

the malware.

c) What information does the screenshot in Figure 3 give about the malware? (3p).

d) How would you proceed to investigate the malware further? (3p)

Answer

a) **2p** The security credentials of a subject is stored in its token. An access token contains user SID, group SIDs, alias SIDs, and privileges. The Discretionary Access Control List (DACL) of an object consists of Access Control Entries (ACEs). Each ACE contains the SID it applies to, the type of access, and whether access is denied or granted. If the access token of the subject requesting a specific access to an object contains SIDs that according to the DACL of the object are granted the requested access, access is granted.

b) **3p** UAC is a mechanism for letting administrator users have as low privileges as possible, according to the security principle of least privilege. A standard token is used by default, and the admin token is prompted for (with the UAC pop-up) when needed to perform specific tasks. The idea is that if an attacker manages to take control over a process with the user's token, the attacker does not automatically have all administrator privileges, only the standard user privileges. This may introduce the extra step of privilege escalation in the attack, thereby increasing the difficulty. Before UAC was introduced, users tended to use administrator users (and administrator tokens) at all times, even though the administrator privileges were only needed for a few tasks, because it was too tedious to be logged in as normal user, since they needed to log out and in again as administrator if they needed to perform a task requiring administrator privileges.

c) **1p** The process Lab03-03.exe creates a new process called svchost.exe, which again creates another process called svchost.exe.

1p The name svchost is also the name of some normal, non-malicious windows processes. The fact that the malware spawns processes with this name is probably to hide and be difficult to detect. We can see from the screenshot that the Lab03-03.exe process is terminating, meaning that the two svchosts will soon be orphaned, and it will be more difficult to spot that these two processes were created by the malware and are not normal svchost processes. **1p** The WerFault.exe means that the malware has failed during execution. Probably because it was created for an older/other version of windows. This means that doing dynamic analysis like this will not show the actual purpose/functionality of this malware.

d) **3p** To investigate the malware further, we could try to run the malware on older/other versions of windows, we could try to investigate what makes it fail (for example through debugging it), or we could do static analysis. Static analysis is the only way to be sure that you can find/analyze all possible functionalities of the malware, not only the functionality that is run under the current conditions. We can use IDA to do static analysis: Find strings, export and import functions. Look for functions that interact with the system, like writing to or reading from files or the registry, or network activity. If the strings are obfuscated, try to decrypt them.

(Continued on page 10.)

Question 9. Linux

Consider the folling directory listing:

```
-rw-r--r-x 2  dole woodpeckers 8789 Feb 20 12:49  myfile
```

a) Briefly describe **who** can do **what** with *myfile* (3p)

In Linux systems users have a UID and username.

b) Explain the relationship between usernames and UIDs. (2p)

c) Which identifier is used for authentication, and which identifier is used to make access control decisions? (2p)

Answer

a) **1p** The user dole can read and write to myfile.
1p Members of the group woodpeckers can read myfile.
1p Others can read and execute myfile.
b) **1p** In Linux systems each user has a username which is mapped to a UID (User Identity).
1p Multiple users can be mapped to the same UID, but a single username can only have a single UID.
c) **1p** The username is the key to find the hashed password which is used for user authentication.
1p The UID is used when the system approves access to resources.

Question 10. Hardware security and Android

a) Briefly explain the difference between hardware and software vulnerabilities. (1p)

b) Describe the relationship between the BIOS/UEFI and the operating system. (2p)

c) Describe some security benefits and risks from using virtualization. (2p)

Android's access control model is based on Linux.

d) Explain the role of permissions in the Android security model. (2p)

e) Explain how traditional Linux users are utilized for security in Android. (2p)

f) Briefly describe the particular security challenges related to mobile devices (2p)

Answer

a) **1p** A hardware vulnerability is a vulnerability in the hardware on which software is running, while a software vulnerability is a vulnerability in the code/instructions of the software.
b) **2p** The BIOS/UEFI is responsible for initializing the system when it

(Continued on page 11.)

boots, and it loads the operating system.
c) **2p** Virtualization provides isolation between virtual machines and between virtual machine and host. This isolation provides a safe(r) environment for testing and analyzing malware. Still, the VMs and the host run on the same hardware, and they would still be vulnerable to hardware-level attacks like Spectre and Meltdown. Also the virtualization layer itself is an attack surface, and enabling features like shared clipboard weakens the isolation.
d) **2p** Applications have a set of permissions, which describe what the application is allowed to access. There are three types of permissions: Normal (not show to the user), Dangerous (user is alerted, can cost money/access personal data), and Signature (Only granted to apps that have the same key as the app that declared the permission. Primarily used to give system permissions to system apps).
e) **2p** Android typically uses one Linux user (UID) per application. Otherwise it uses a traditional Unix filesystem security model, with UIDs and GIDs and access masks for all file objects.
f) **2p** Mobile devices are always online, on different networks and multiple network types. They are often used in public and across security boundaries (private/work/meetings...). They are easy to lose/steal.

Good luck!

Front page

TEK5510 – Security in Operating Systems and Software
Time: Thursday November 28th at 14:30 (4 hours).

All questions should be answered. The questions are weighted differently. The maximum score is shown for each question. The maximum total score is 107.

Answers can be given in Norwegian or English.

1 Types of exploits

One way to classify exploits is by which abilities they give the attacker:

1. Arbitrary code execution
2. Information disclosure
3. Denial of service

Describe these three types of exploits, and explain their differences.

(6 points)

Answer:

- Arbitrary code execution: the ability to execute any command of the attacker's choice.
- Information disclosure: the ability to access information the vulnerable software did not intend to disclose.
- Denial of service: the ability to make the vulnerable software unavailable to its legitimate users.

2 points for each

Arbitrary code execution may be used both to obtain information and for denial of service and for running any other code.

2 Security principles

Continue with the same classification of exploits:

1. Arbitrary code execution
2. Information disclosure
3. Denial of service

Explain the three security principles *Confidentiality*, *Integrity* and *Availability and* how these principles relate to the three types of exploits above.

(6 points)

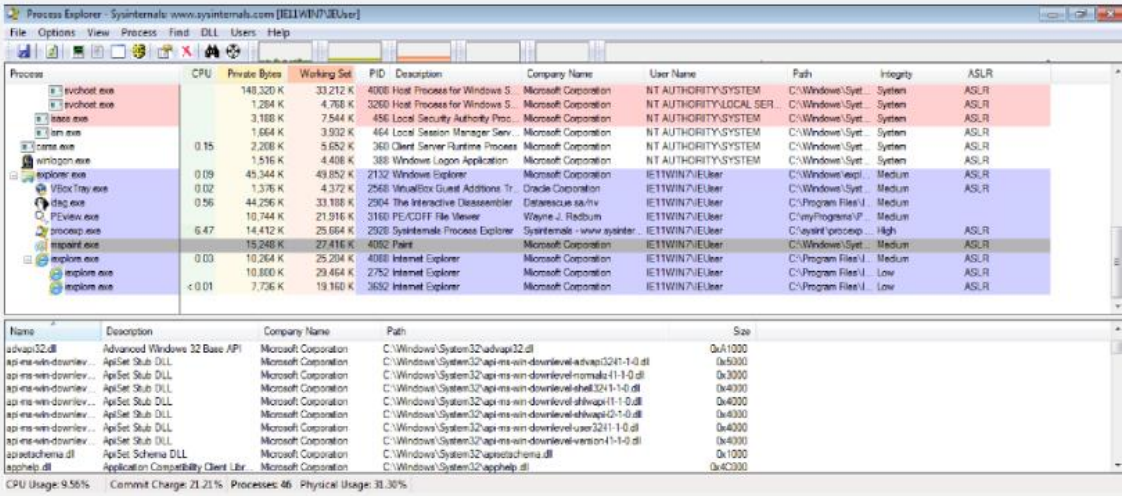
Answer:

- Confidentiality is about preventing unauthorized disclosure of information (unauthorized reading)
    - An information disclosure exploit is an attack on confidentiality
    - Arbitrary code execution can be used for an attack on this principle

- Integrity is about preventing unauthorized modification of information (unauthorized writing)
    - Arbitrary code execution can be used for an attack on this principle
- Availability is the property of being accessible and usable upon demand by an authorized entity
    - A denial of service exploit is an attack on availability
    - Arbitrary code execution can be used for an attack on this principle

1p for each correct definition of confidentiality, integrity and availability
1p for each principle associated with the correct exploit types

3 **Process Explorer**



This is a screenshot from Process Explorer. Explain what information this gives us about the running process mspaint.exe. Don't forget to mention the lower pane.

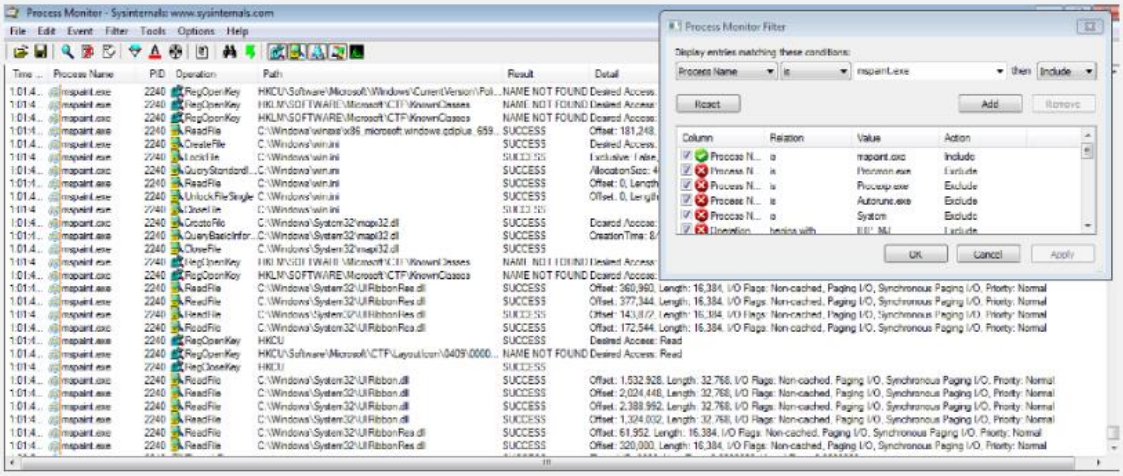
(5 points)

Answer should contain:

- PID 4092, process identity, which can be used as an ID number for finding the correct process for example in process monitor or a debugger.
- This process runs as IEUser. Integrity level Medium shows that it is a non-admin token. (The non-admin token of IEUser)
- The path to the executable (C:\Windows\Syst..), name of the company (Microsoft corporation), description (Paint)
- ASLR is on for this process, meaning that the address space layout of the process is not deterministic. Makes it more difficult for an attacker to predict addresses.
- The lower pane shows the dlls (libraries) currently loaded by this process. These libraries contain additional functionality used/needed by the executable, but not included in the executable itself.

1p for each of these. If one is missing, and private bytes/working set are well described, a point should be given for those instead.

4 Process Monitor



This is a screenshot from Process Monitor. Explain what information this gives us about mspaint.exe. How would you proceed with analyzing the executable in Process Monitor?

(5 points)

Answer:

- Shows the activity of the process (registry activity, network activity, file system activity, process/thread activity)
- The screenshot has a filter which makes only the activity of mspaint.exe show.
- Shows registry access requests. Most are not found, but HKCU is opened with read access.
- Shows files that are read. An initialization file win.ini. Several dlls
- Only a fraction of all events of paint.exe are visible in the screenshot. Next step would typically be to add a filter for only write file activity and afterwards a filter for only set registry value activity. In other words: Add specific filters for interesting, more specific, events and study these more closely.

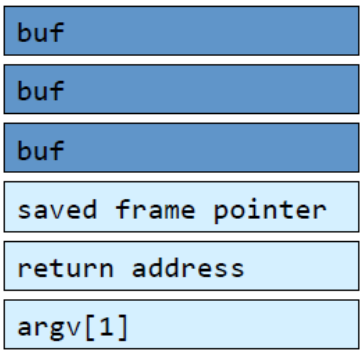
1 p for each of these. Similar or other reasonable answers should be accepted, but maximum 5 p total. Filters must be mentioned to achieve full score.

5 A vulnerable function

foo.c:

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

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



This figure shows the code of a vulnerable function and a part of the stack for this function.
Answer the following questions:

1. What is this type of vulnerability called?
2. How would you exploit this function?
3. What is a stack cookie, and would it prevent your exploit?
4. What is ASLR and would it prevent your exploit?

(10 points)

Answer:

1. 1p for (Stack) buffer overflow
2. 3p for sending in a too long buffer, overwriting the return address, making the function return to code you want to run (shell code in the buffer or ROP)
3.
   - 2p for a stack cookie is a value placed on the stack as a canary at the beginning of the function, and this value is checked before returning to the return address. If the cookie is changed (e.g. overwritten by an overflow), the program fails instead of returning to the value in the return address field.
   - 2p for this would prevent a simple stack overflow which overwrites the return address and makes the function return to the new address in this field. (Descriptions of how to bypass a cookie are of course accepted)
4.
   - 1p for ASLR is Address Space Layout Randomization, which means that the addresses of stack, executables and libraries are randomized.
   - 1p for this would make it difficult to find a static address to return to. (Descriptions of how to bypass ASLR are of course accepted)

6 Computing with integers

Show the computations with binary numbers and explain the results.
With unsigned 8-bit integers:

1. 128+8=?
2. 253+6=?

With signed 8-bit integers:

3. 128+8=?
4. -123-8=?

(8 points)

Answer: 1p for each correct answer, 1p for each correct bit calculation.

1. $128 + 8 = 1000\ 0000_2 + 1000_2 = 1000\ 1000_2 = 136$
2. $253 + 6 = 1111\ 1101_2 + 110_2 = 1\ 0000\ 0011_2$ (first bit removed, since 8-bit int) => $11_2 = 3$
3. 2p for: 128 is not possible to write in signed 8-bit integers. The highest possible number is 127.
4. $-123 - 8 = 1000\ 0101_2 + 1111\ 1000_2 = 1\ 0111\ 1101_2$ (first bit removed, since 8-bit int) => $0111\ 1101_2 = 125$

7 Passwords

Explain how passwords are stored

1. in Linux
2. in Windows

(6 points)

Answer:

1. Linux
   - 1p for passwords are not stored in /etc/passwd but in a shadow file (/etc/shadow) that can only be accessed by root.
   - 2p for passwords are hashed with a salt and stored together with the salt (and info about which hashing algorithm that was used).
2. Windows
   - 1p for passwords are stored in the SAM (Security Account Manager), a hive in the registry. (SAM is stored in a file on disk. This file is not readable while windows is running)
   - 2p for passwords are stored as unsalted NTLM hashes.

8 Password cracking

1. What is the difference between a hash function and an encryption function?
2. Explain how password cracking is done.
3. What can be done to make it more difficult for an attacker to crack our passwords?

(6 points)

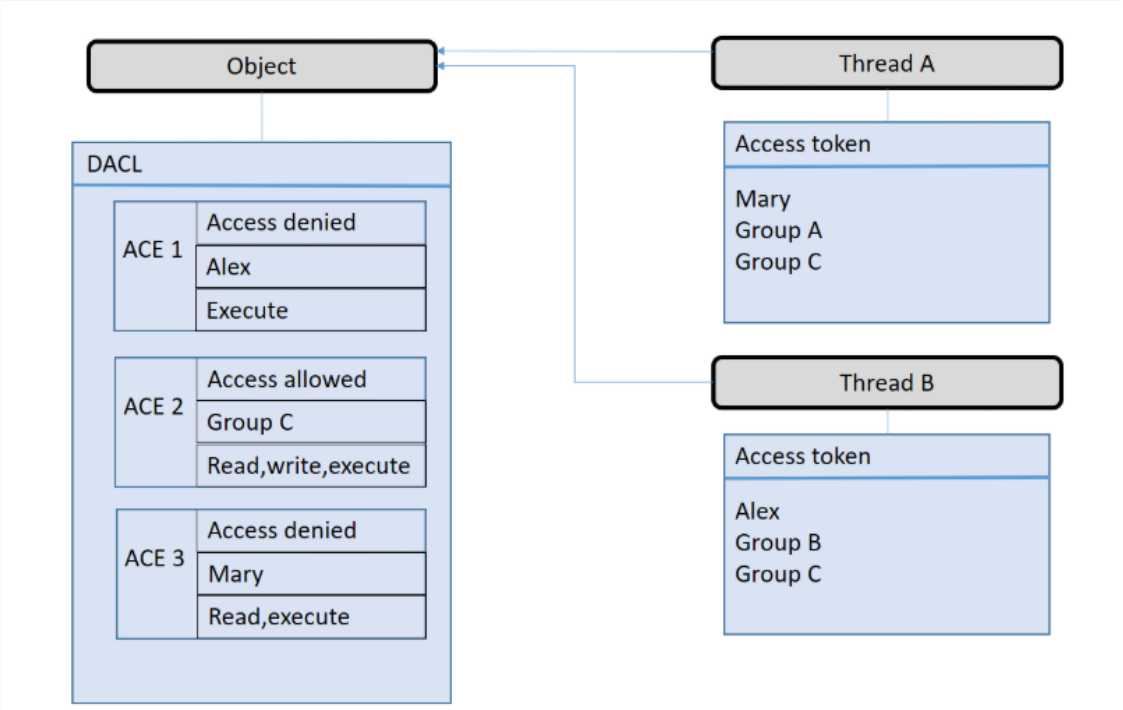
Answer:

1. 2p for a hash function is a one-way function, an encryption function has a corresponding decryption function, such that encryption can be reversed (if the key is known).

2. 2p for when a password is stored as a hashed value, the hash function cannot be reversed to obtain the password. Password cracking is therefore done by guessing a password, hashing it and comparing the value with the stored hash value. Several strategies can be used to guess passwords: brute force, word lists, word lists with rules.
3. 2p for storing the passwords as salted hash values together with a random salt makes it more difficult to crack passwords because
    - Precomputed hash tables (or rainbow tables) cannot be used
    - Each password must be cracked individually

   Answers mentioning using strong passwords with large character set/passphrases or other relevant answers should also be accepted, but some explanation is needed for full score.

9 Access Control Lists in Windows



The figure above shows an object with a discretionary access control list 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) in the DACL influences the access control decision.

1. Thread A requests read and write access to the object.
2. Thread B requests read access to the object.
3. Thread B instead requests read and execute access to the object.
4. Could adding a fourth ACE change the results above?

(10 points)

Answer:

For 1-2: 1p for correct access control decision. 2p for right description of how the ACEs influence the decision.

1. ACE1 does not apply. Alex is not in the thread token. ACE2 grants read and write access since group C is in the thread token. The requested access, read and write, is given.

2. ACE1 does not apply. Alex is in the thread token, but execute access is not requested. ACE2 grants read access since group C is in the thread token. The requested access, read, is given.
3. 2p for: ACE1 denies access. Alex is in the thread token, and execution access is requested. No access is given.
4. 2p for: Adding an ACE4 at the end of the list would not change the decisions. In fact all the above decisions were made without using the ACE3 as well. If a new ACE is added on top of the list, the decision may change in all cases, depending on the added ACE. If a new ACE is added between ACE1 and ACE2, it may change the decision in the first two cases.

10 **Access Control in Linux**

Consider the following output from the command *ls -l*

```
-rw-r--r-- 1 Doffen student 1617 Oct 28 11:01 tent.make
drwx------ 2 Doffen student  512 Oct 25 17:44 tricks
```

1. Explain what the output tells us about the two items. In particular, describe Doffen's access rights to these items.
2. The user Ole is also member of the student group. What are his access rights to these items?
3. The user Dole is member of the teacher group. What are his access rights to these items?
4. Doffen runs the following command: *chmod u+x,g-rx tent.make*. How do the access rights change for Ole, Dole and Doffen?

(8 points)

Answer:

1.
   - 1p for directory or not, access mask, user name of owner (Doffen), group (student), size, date and time, name of file (tent.make)/folder (tricks)
   - 1p for Doffen can read and write to tent.make, and Doffen can read, write and execute the folder tricks
2. 2p for Ole can read the file tent.make, but has no access to the folder tricks.
3. 1p to Dole can read the file tent.make, but has no access to the folder tricks. (same as above since the same access mask for group and others)
4. 3p for Doffen gains execute access to tent.make. Ole loses read access, since the group loses read access. No changes for Dole, since there is no changes for “others”.

11 **OWASP Security Principles**

Consider these three OWASP security principles:

1. Minimize attack surface
2. Fail securely
3. Avoid security by obscurity

For each of these security principles, explain:

- What does this security principle mean?
- How would not following this security principle make an application vulnerable?
- How could this security principle be implemented in practice.

If you want to use an example application when answering these questions, you can use the Inspera digital exam application you are now using.

(9 points)

Answer:

1p for each bullet point (or similar answers)

1. Minimize attack surface
    - To minimize the externally accessible interface (e.g. input – only allow specific types of input/features – not more accessible features or functions than necessary.)
    - If there is a large external interface/many possible types of input, the probability that there will be an exploitable bug in code that can be reached by an attacker increases.
    - Few functions/features externally accessible, and good input validation. Input validation is easier if only specific types of input are allowed. For example not text, only allowed to pick elements from lists.
2. Fail securely
    - If an error/exception occurs, make sure to handle it. Don't continue execution if an error is detected.
    - An error may be exploitable (or caused by an attacker), and continuing may give the attacker the possibility to use the error to own advantage.
    - Abort the process if an error is detected, or go to an error handling routine which secures assets and cleans up.
3. Avoid security by obscurity
    - Security by obscurity means that if the code/executable is difficult to understand for example because of obfuscation or secret source code, it is difficult to find out how to exploit it. This should be avoided.
    - Security by obscurity should be avoided, since it may also be difficult to understand how to secure it, and it may become vulnerable for example if source code is leaked and security relies on the source code being secret. (It should particularly be avoided as the only security mechanism, but of course keeping the source code secret if the source code is well written is not a security risk.)
    - Write tidy and clean code, not too complicated. Makes it easier to review the code and prevents logical errors.

12 Processor vulnerabilities

In the processor vulnerabilities like Spectre and Meltdown, measurement of load time was essential in the attacks. Explain how.

(3 points)

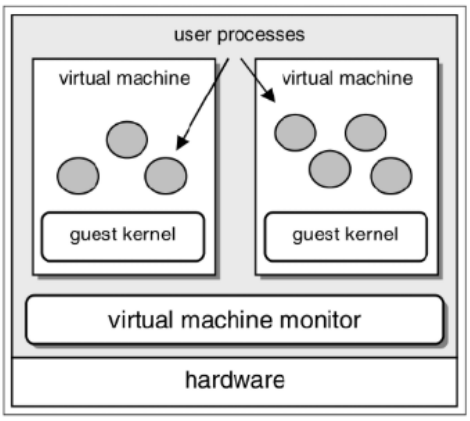
Answer:

An answer with the following detail level is sufficient. Both elements in bold must be included to obtain full score:

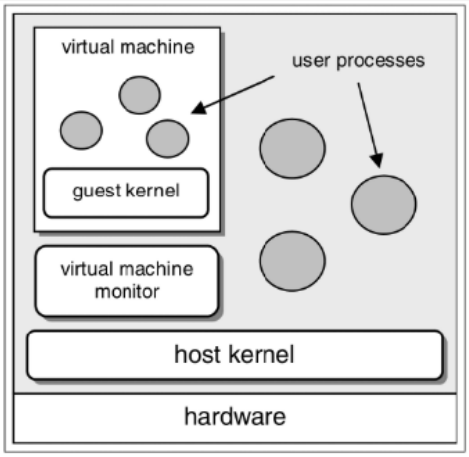
The attacks want to gain access to a secret. The attacker makes the secret be used indirectly in a **speculatively executed instruction (also accepted: out of order execution)**, for example as an index of an array. This instruction stores an array element into an internal cache on the processor. The index of the array element depends on the secret. We then looping through all array elements,

measuring the load time of each element. The index of the array element stored in cache by the speculative instruction is **the one with shortest load time**.

13 Virtualization



Type I hypervisor



Type II hypervisor

1. The figures illustrate two types of hypervisors. Explain how each type work and point out the significant differences.
2. Describe the virtualization solution you have used in this course. Include type of virtual machine monitor, guest kernel and host kernel (if applicable).
3. Give three reasons for using virtualization.

(7 points)

Answer:

1. 1p for each
   - Type I hypervisor runs below the operating system, is usually very small and has high performance. Hardware support can be an issue.
   - Type II runs on a host operating system, uses the host services for memory management, scheduling and drivers, has a performance penalty and is easier to use.
2. 2p for something like Type II hypervisor on a laptop with Windows 10 as host kernel/operating system, Virtual box as virtual machine monitor and the Windows course VM as the virtual machine (i.e Windows 10 as guest kernel)
   - We expect most students to have used type II hypervisor, but of course type I answers are accepted if the answer makes sense.
   - The students need not answer their actual virtualization solution, as long as they can describe a possible virtualization solution correctly.
3. 1p for each of these, maximum 3 p:
   - Efficient use of hardware and resources
   - Improved security (Isolated VMs, safe testing and analysis of malware)
   - Distributed applications bundled with the operating system
   - Powerful debugging (Snapshot of current state, step through program and operating system execution, reset system state)

14 Web security

1. What is SQL-injection?
2. What can an attacker do with XSS? Give four examples.
3. What is session hijacking
4. How can an attacker get the session cookie of a victim? Give two examples.

(10 points)

Answer:

1. 2p for: The attacker takes advantage of poor input validation by providing an input that influences the SQL query of the web application, either by trying to execute his own SQL query or by modifying the original query to gain access or information.
2. 1p for each example. 4p maximum:
    - Provide any html element, including javascript
    - Redirect the page to another site to mislead the user
    - Rewrite the document content/deface the site
    - Get cookie/session variables
    - Key logging
    - Phishing
    - Launch browser exploits
3. 2p for: The attacker modifies the session variables through the cookies to get access to unavailable sites. The attacker gains access to or guesses the session cookies of another user which has access.
4. 1p for each bullet point, maximum 2p:
    - Through information disclosure (e.g. session variable in the url)
    - Steal session cookie through vulnerability (e.g. XSS)
    - Steal through social engineering (e.g. link in e-mail)
    - Predictable session variable makes it possible to guess
    - Brute forcing

15 Mobile

1. Explain the patching process of Android devices when Google publishes a new version of Android Open Source Project.
2. Explain briefly how a) the Linux kernel and b) sandboxing provide security mechanisms for Android.

(8 points)

Answer:

1. 4p if these steps are described:
    a. Google publishes AOSP release
    b. Silicon manufacturers modify chip related code
    c. OEMs add their own apps and overlays
    d. Carriers test OEM updates, add their own apps
    e. Users receive OTA updates

Also accept answers considering newer versions (Project Treble, from Android 8.0), where step 2 was removed.

2.

a. 1p for each bullet point
   - Isolated processes
   - User based permissions

b. 1p for each bullet point
   - Each app has its own UID, meaning that apps are isolated as users are in Linux.
   - SELinux extends the android sandbox with MAC.

   Also accept answers explaining Seccomp (secure computing mode) or Chrome/Webview.

UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	UNIK 4270 — Security in operating systems and software.
Day of examination:	Tuesday December 5th 2017.
Examination hours:	14:30 – 18:30.
This problem set consists of 10 pages.	
Appendices:	None
Permitted aids:	Dictionary.

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

Answer all questions in this examination paper.

Answers can be written in English or in Norwegian.

Be concise when answering. It is often sufficient to write a single sentence, or at most a few sentences, to describe the concept that each sub-question asks for.

Question 1. Cryptography, hash functions, and passwords

a) Explain the concepts of *symmetric* and *asymmetric* cryptographic methods. (2p)

b) Give two properties of hash functions, and explain why hash functions need to have these properties. (2p)

c) Explain the concepts *exhaustive search* and *intelligent search* in password cracking. (2p)

d) What is a *rainbow table*, and how is it used in password cracking? (2p)

e) What is password salting and why should it be done? (2p)

Answer

a) **1p** for stating that symmetric cryptographic methods use the same key for encryption and decryption.
1p for stating that asymmetric cryptographic methods use different keys for encryption and decryption.
b) **1p** for stating that hash functions should be slow to reverse and that small difference in input gives large difference in output. (In the context of passwords, it's also good if the hash function is slow (either a slow algorithm, or multiple rounds of hashing)).
1p These properties make hash cracking (including password cracking) time consuming.

(Continued on page 2.)

c) **1p** Exhaustive search: Brute force, trying all possibilities.
1p Intelligent search: Dictionary attack, using relevant information, for example from the password file, to guess likely passwords and variants of them.
d) **1p** A rainbow table is a table of passwords and their corresponding hashes.
1p Rainbow tables can be used for password cracking of (unsalted) password hashes by looking up a hash in the table to find the corresponding password, without needing to crack the hash.
e) **1p** Password salting means to append or prepend a salt (string) to the password before hashing it.
1p This is done to make precomputed rainbow tables unusable. The hash depends on the salt value, so one would need a rainbow table for the specific salt value to use the table for cracking.

Question 2. Malware and vulnerabilities

a) Simplistically, malware consists of two parts: An *exploit* and a *payload*. Explain what these two parts are. (2p)

b) Explain these terms for malware behavior: (1) Backdoor, (2) Credential stealer, (3) Ransomware. (3p)

c) How and why does malware use cryptography? (2p)

d) Vulnerabilities can be disclosed in different ways. Explain the concepts *full disclosure* and *responsible disclosure*. (2p)

e) What is a 0-day? (1p)

Answer
a) **1p** The exploit is the part that triggers a vulnerability (or several vulnerabilities) in the target program and uses this to take control of the execution flow. The purpose of this part is to enable the payload to run.
1p The payload is the part that contains the actions the malware writer wants to be executed, for example encrypting the files on the computer. The payload could be said to contain the purpose of the malware, while the exploit enables it to be run.
b) **1p** A backdoor is a functionality that gives those who know about it the ability to gain access when they desire it. If a malware installs a backdoor, the malware writer can use the backdoor to gain access later, without using malware to trigger a vulnerability again.
1p A credential stealer steals credentials, like user names and passwords, so that they can be used to gain access to the same or other systems later.
1p Ransomware encrypts files and demands money to decrypt them again.
c) **2p** Malware can use cryptography to hide its malicious functionality and disguise as a legit program, both the contents of the executable and the data sent through a communication channel over the internet used to send information home. The aim can then be not to be detected rather than being impossible to crack. (An exception is ransomware, which doesn't want the crypto it used to encrypt your files to be cracked)
d) **1p** Full disclosure: Publish everything about the vulnerability to everyone.
1p Responsible disclosure: Tell the vendor first and give them time to fix the issue. When it is fixed, or they have had sufficient time to fix it if they cared, then publish it to the world.

(Continued on page 3.)

e **1p** A vulnerability that has not been published/made known to the vendor. The name indicates that the vendor has had zero days to fix the vulnerability.

Question 3. Analyzing executables

a) Explain the terms *static analysis* and *dynamic analysis*. (2p)

b) If you receive a suspicious executable, how would you proceed to analyze it? Explain what tools like Process Monitor, Process Explorer, RegEdit, and IDA can be used for in this process. It is sufficient to explain the main steps and concepts. (8p)

Answer

a) **1p** Static analysis means to analyze the code of the executable without running it, for example by using IDA or another disassembler.
1p Dynamic analysis is to analyze the running executable, with a debugger attached and/or by using tools for monitoring its interactions with the operating system and other processes.
b) Several approaches are possible here, but this should be included:
1p A Virtual Machine (or similar security measures) should be used when running the executable.
2p Process Monitor monitors the interactions between processes and with the OS, including the registry, the file system, and network traffic. It can be used to determine where one should continue the analysis, eg. files, registry values accessed.
1p Process Explorer shows the running processes on the system, including parent/child relations.
2p IDA can be used to analyze the executable statically and for debugging the running process.
1p RegEdit (and RegShot) can be used to explore the registry values related to the malware.
1p The last point is not pinpointed to anything specific, but should be given if the description as a whole makes sense.

Question 4. Assembly

Consider these x86 assembly instructions:

```
xor ebx, ebx
mov ebx, 8
add ebx, 2
xor eax, eax
mov eax, 7FFFFFFEh
push eax
push 1
push ebx
call sub_58F52C
```

a) Explain what each of these assembly instructions does. (6p)

b) What is the value of *ebx* after these instructions are executed? (2p)

c) What seems to be the purpose of this set of assembly instructions? (2p)

Answer

a) **1p** *xor ebx,ebx* and *xor eax, eax* set the registry values ebx and eax,

(Continued on page 4.)

respectively, to 0.
1p *mov ebx, 8*: Set the value of ebx to 8.
1p *add ebx, 2*: Add 2 to ebx.
1p *mov eax, 7FFFFFFEh* Set the value of eax to 7FFFFFFEh. The h at the end means that the value is in hex.
1p *push eax*: Push the value of eax to the stack. Similarly for the next two.
1p *call sub_58f52c* Call a function (the function identified by IDA as located at 58f52c)
b) **1p** for 10. An additional **1p** for saying that there are more possibilities.
c **2p** Calling a function with three parameters. The last line calls the function. The lines above set up the parameters.

Question 5. Exploitation

Consider the following C code:

```
void foo (char *str)
{
  char buffer[16];
  strcpy(buffer, str);
}
```

Figure 1 is a screenshot of Immunity Debugger after breaking inside the function *foo*, just before the call to *strcpy*.

a) Briefly explain what kind of information each window displays (4p).

b) Use the information provided by the screenshot to draw a diagram of the stack frame for *foo*. Annotate the diagram with the address and value of each element of the stack frame. (4p)

c) What is the value of EIP after the function *foo* returns? (1p)

d) Describe what a *stack cookie/canary* is, and how using a stack cookie/canary would affect what happens in this example. (1p)

Answer
a) **1p** for each of:

- Disassembler window (showing the code)
- Register window (showing the current state of the registers)
- Stack window (showing the current state of the stack)
- Memory/data dump window (shows a dump of memory)

b) The diagram should look something like this :

	Address	Value	
	0012FD20	00401F80	the local variable *buffer*
	0012FD24	4825524E	
	0012FD28	FFFFFFFF	
	0012FD2C	004014B1	
EBP →	0012FD30	0012FF40	saved frame pointer (saved EBP)
	0012FD34	00401100	return address (saved EIP)
	0012FD38	0012FD3C	function argument

(Continued on page 5.)

Having all the elements in the correct order (buffer, saved frame pointer, return address, argument), should be sufficient for **3p**. Also having the correct addresses and values gives **4p**.
c) **1p** for: EIP will have the value 46464646 (ascii FFFF).
d) **1p** for something like: A stack cookie/canary is a random value placed on the stack between the local variables and the saved frame pointer. Before the function returns, the system checks that the cookie/canary has the correct value, and terminates the application if it has been overwritten. The use of stack cookies/canaries does not prevent an overflow from occuring, but detects it (thus preventing the attacker gaining control over execution).

(Continued on page 6.)

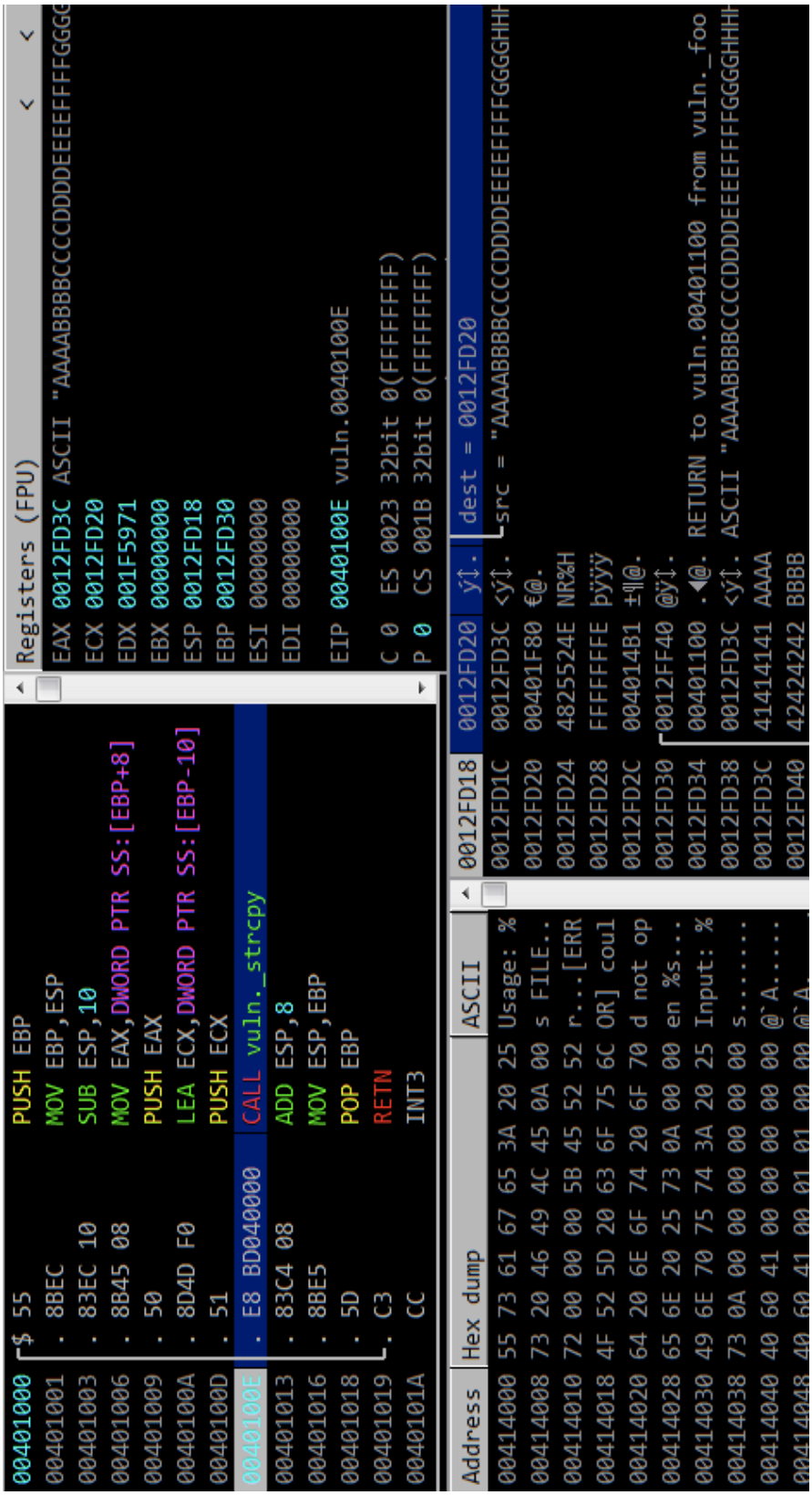


Figure 1: Breaking inside foo

(Continued on page 7.)

Question 6. Linux

a) What is stored in the */etc/passwd* file, and who has read access to this file? (2p)

b) What is stored in the */etc/shadow* file, and who has read access to this file? (2p)

c) What is the difference between *deleting* and *wiping* a file? (2p)

Consider the following directory listing:

```
-r--rw-r-- 3  sam staff 16289 Nov 11 19:23  notes.tex
-rws--x--x 3  root bin  16384 Nov 07 10:20  passwd
```

The group staff consists of Sam and Jenna. Bob and Bill are the other users on the system.

d) Who can read *notes.tex*? (1p)

e) Who can write to *notes.tex*? (1p)

f) Who can change the permissions of *notes.tex*? (1p)

g) For the file *passwd*, the execute permission of the owner is given as an **s** instead of an **x**. What does that mean? (2p)

Answer

a) **1p** for: User accounts are stored in the /etc/passwd file. No details needed. **1p** for: World readable.
b) **1p** for: Password hashes are stored in /etc/shadow. No details needed. **1p** for: Only readable by root.
c) **2p** for something like: Deleting a file means that the disk blocks allocated to the file becomes available again, but the blocks may still contain the file content until the disk blocks are written to again. Wiping a file means overwriting the content of the file.
d) **1p** for: All (Sam, Jenna, Bob and Bill)
e) **1p** for: Jenna
f) **1p** for: Sam
In addition, *root* can read, write and change permission of every file, but *root* does not need to be included for full points to be awarded.
f) **1p** for: It means that the program has the SUID bit set. **1p** for: This means the program will run with the effective user ID of its owner, which is *root* in our case.

Question 7. Windows

a) Which processes are involved in the logon process and which role do they have? Identify these processes in the screenshot in figure 2, for example by finding their PIDs. (6p)

b) What does the screenshot in figure 2 tell us about how the two calc.exe processes were started? (2p)

Answer

a) **1p** for winlogon.exe with PID 408
1p for lsass.exe with PID 460
1p for explorer.exe with PID 904

(Continued on page 8.)

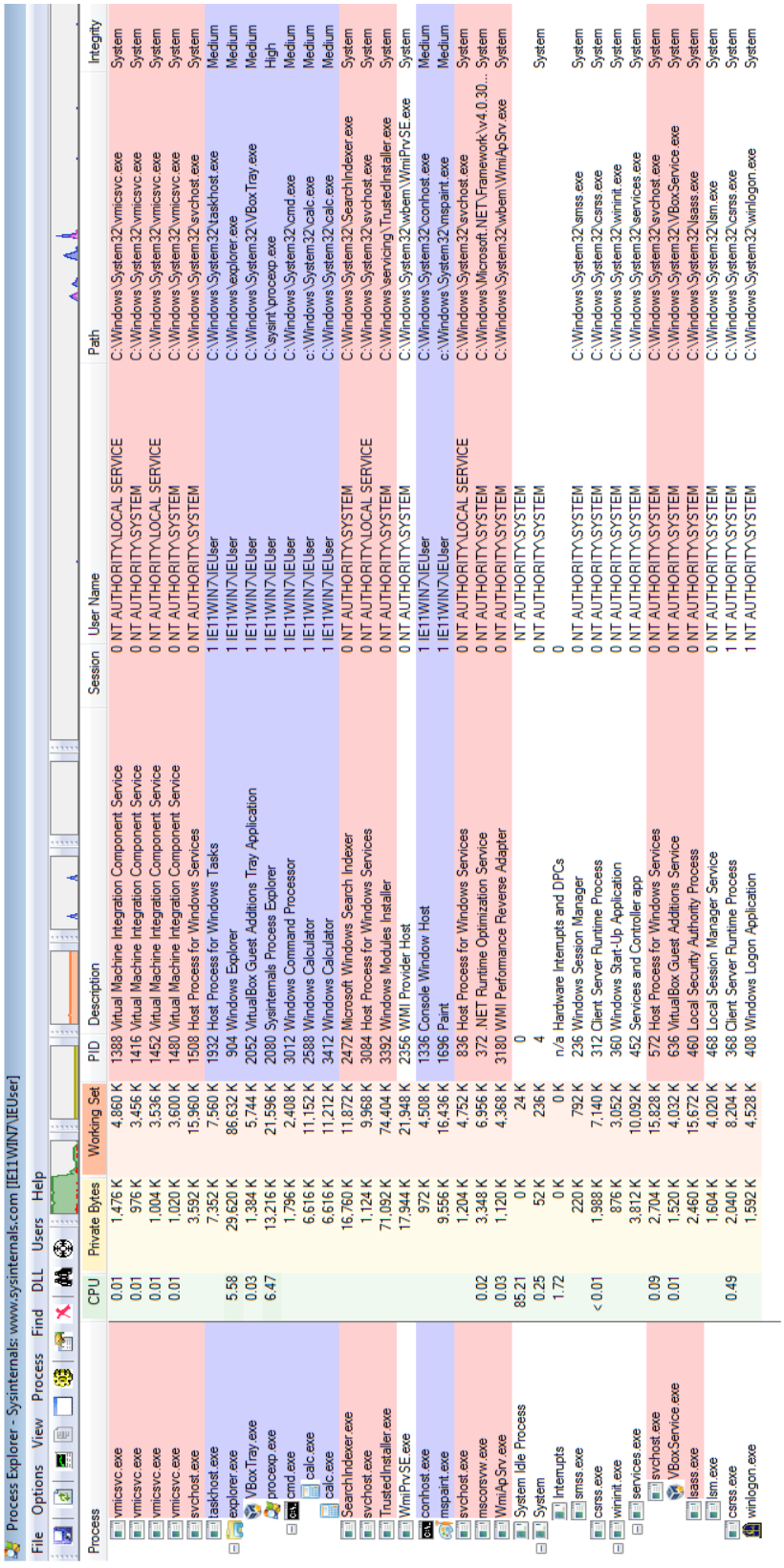


Figure 2: Processes in Windows

(Continued on page 9.)

1p for: The winlogon.exe process receives the user credentials and sends them to the Local Security Authority (LSA)
1p for: LSA (lsass.exe) verifies the credentials, using the users' passwords that (for local users) are stored in the Security Account Manager (SAM).
1p for: The logon process starts a shell (explorer.exe) in a new logon session for the user. This shell spawns the processes started by the user.
b) **1p** for: The calc process with PID 2588 was started from the command line in cmd.exe, since it is the child of this process.
1p for: The calc process with PID 3412 was started directly, for example from the start menu or by clicking it in the folder it is located.

Question 8. Trusted computing, BIOS/UEFI and Android

a) Briefly explain what remote attestation is. (2p)

The BIOS/UEFI has traditionally been considered the root of trust for the OS and applications that execute on the platform.

b) Describe what makes the BIOS/UEFI important for the security of the system. (3p)

Android's access control model is based on Linux.

c) What is the principle for isolating, or sandboxing, apps from each other? (2p)

d) What are the differences between *Normal*, *Dangerous* and *Signature* permissions? (3p)

Answer
a) **2p** for something like: Remote attestation is a method to prove to a remote party that the local PC is a trusted platform, by showing its current configuration to a remote party. Basically, the local PC takes measurements (trust anchored in the TPM) of its configuration, and sends them (signed) to a remote party.
b) **3p** for something like: BIOS/UEFI is the first code to run on the processor, so it can maliciously alter the OS image it's supposed to load after. The BIOS/UEFI has fully privileged access to all hardware, can talk to and reprogram deveices at will. BIOS/UEFI provides code that runs in SMM during the whole lifespan of the platform, can easily inject rootkits there.
c) **2p** for something like: Each app is a separate user with its own UID (user identifier).
d) **3p** for something like: *Normal* are given to any app requesting them without prompting the user, *Dangerous* must be approved by the user, *Signature* can only be given to an app signed with the same certificate that defined the permission.

Question 9. Web security
Consider the web page below, and the code that implements it:

```
<?php
```

(Continued on page 10.)

Figure 3: Web page

```
if (isset($_POST["comment"], $_POST["name"]))
  {
    print($_POST["name"]." said: ".$_POST["comment"]);
  }
?>
```

Listing 1: PHP code

This page is vulnerable to a common type of web attack.

a) What is this type of attack called? (2p)

b) What can the attacker do with this attack? (2p)

c) Briefly describe the difference between normal and blind SQL injection attacks. (3p)

d) Briefly describe what a Local File Inclusion vulnerability is. (2p)

Answer

a) **2p** for Cross Site Scripting (XSS) attack.

b) **2p** answers should include at least two of the follwoing:
Attacker can: provide any html element (including javascript), redirect the page to another site, rewrite the document content (defacing), get the coockie variables (for session hijacking), keylogging (register event listener), phishing (insert fake login form), launch browser exploits.

c) **3p** for something like: Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible. The important parts to understand: With blind sql injections, we don't directly see the output from the query, but are able to make queries that return different result based on a true or false condition.

d) **2p** for something like: The user can provide input to a web page that will make the webserver include a local file in its response.

Good luck!

UNIVERSITY OF OSLO

Faculty of Mathematics and Natural Sciences

Example Exam in: UNIK4270 – Security in Operating Systems and Software

Day of exam: XXX
Exam hours: XXX
This examination paper consists of X page(s).
Appendices: None
Permitted materials: Dictionary

Make sure that your copy of this examination paper is complete before answering.

Answer all questions in this examination paper.

Answers can be written in English or in Norwegian.

Be concise when answering. It is often sufficient to write a single sentence, or at most a few sentences, to describe the concept that each sub-question asks for.

Question 1: Security Concepts

a. Define the concepts of confidentiality, integrity and availability. (3p)
b. Denial of Service is an attack on one of these concepts. Which one? (1p)
c. Assume that you are developing a mobile Bank App which gives customers full access to bank accounts. Explain how you would apply principles for secure software development in this case. (6p)

a. 1p for: Confidentiality means to prevent unauthorized disclosure of information.
   1p for: Integrity means to prevent unauthorized modification of information. In the most general sense we can say that integrity makes sure that everything is as it is supposed to be.
   1p for: Availability is the property of being accessible and usable upon request from an authorized entity
b. 1p for: DoS is an attack on availability.
c. Minimize attack surface, security default settings, defense in depth, fail securely, keep security simple

Question 2: Passwords

a. What does the Ctrl+Alt+Delete key combination do, and why should we press it before entering our password when logging in? (3p)
b. Explain why it is unwise to store passwords in cleartext on the authentication server. (2p)
c. Explain how a password is verified when passwords are stored as unsalted hash values. (1p)
d. Explain two security problems in case passwords are stored as unsalted hash values. (2p)
e. Explain the process of storing passwords as salted hash values. (2p)

a. Pressing CTRL+ALT+DEL ensures communication with the winlogon process. This key combination was registered by winlogon during initialization and cannot have been hooked by other applications.
b. Easily read by attackers
c. Hashing the input and compare with stored hash
d. Pre-computed tables of hashes, user's with same password will have same hash
e. hash Salt + password, store salt and hash

Question 3: Access Control

a. What is Discretionary Access Control (DAC)? (2p)
b. What is the main contents of an Access Control Entry (ACE) in a Discretionary Access Control List (DACL)? (2p)
c. Explain the process of granting or denying access using a DACL. (2p)
d. What is a Security Token? (2)
e. How does Windows ensure that each created SID is unique? (2p)

a. Discretionary Access Control: AC policy based on identity and AC lists/matrices. We define an owner for each resource, and the owner decrees who have access
b. Mask for access rights, entry type (pos/neg), type of object, principal SID ACE applies to
c. Granted if all req perms are obtained, denied if a matching deny entry is found, denied if reached end of dacl
d. Access token – security credentials of a subject. Contains user SID, group SID, privleges
e. Windows uses pseudo-random input (clock value) to construct a SID. This means that you will not get the same SID if you delete an account and then recreate it.

Question 4: Software Security and Memory Corruption

If your computer program uses unsigned 8-bit integers, what will the result of the following be?

a. 255 + 2 = ? (1p)
b. 0 – 1 = ? (1p)

If your computer program instead used signed 8-bit integers, what would the result of

c. 0 - 1 be? (1p)
d. Explain what an integer overflow is. (1p)
e. Explain what a buffer overflow is, and how an attacker can exploit it. (4p)
f. Give examples of buffer overflow mitigations. (2p)

b. 1 (1p) b. 255 (1p) c. -1 (1p)

c. An integer overflow is the condition that occurs when the result of an arithmetic operation, such as multiplication or addition, exceeds the maximum size of the integer type used to store it (1p)

d. A buffer overrun (overflow) occurs when the value assigned to a variable exceeds the size of the buffer allocated. Memory locations not allocated to this variable are overwritten.
By overwriting the return address or function pointers or heap chunk header.

e. 1p for each example, maximum 2p:
    a. Cookies/canaries
    b. Non-executable stack
    c. ASLR
    d. Hardware mitigations such as Secure Return Address Stack/separate register for return address
    e. Using safer functions

Question 5:

a. The BIOS is part of a system's trusted computing Base (TCB).
Describe two realistic ways through which the BIOS can become compromised. (2p)
b. Intel boot guard can implement both measured boot and verified boot. Briefly describe each mode of boot. (2p)
c. Explain the concept of SQL injection vulnerabilities on web sites. (2p)
d. What is a ROP chain, and which security mechanism is bypassed with a ROP chain? (4p)
e. You have decompiled and patched the smali code of an Android app. What are the necessary steps for installing the patched app on a device? (6p)

- Being maliciously written by the vendor. OR Due to somebody being able to later modify the original (benign) BIOS with a rogue one OR exploit subtle flaws in the original BIOS and getting code execution before the reflashing or SMM locks are applied.
- Measured boot: the CRTM Passively extends the TPM's PCRs with the hash of the measured boot block.
- Verified boot: the CRTM checks if the boot block is correctly signed with a key which has been encoded in the processor fuses by the OEM.
- SQL: attacker provide input parameter that influences the sql query, attacker tries to execute his own sql query or modify org query to gain info
- Gadgets of small code blocks with a ret type of ending, chained together. DEP/NX.
- - Compile the APK using apktool, - Generate a new certificate
- - Sign the APK, - Install it using e.g, adb

Question 6: Linux/Unix Security

User accounts are stored in the /etc/passwd file. Each entry has the format:
Username:password:UID:GID:ID string:home directory:login shell.

a. Explain the elements of an entry. (2p)

Modern Unix systems also use the file /etc/shadow.

b. Explain the relationship between /etc/passwd and /etc/shadow (2p)

In Linux the access control bits specify access rights to a resource for owner, group and 'world'.

c. State the order of priority of access rights for the owner of a resource who is also member of an applicable group, and who necessarily is also member of 'world'. (2p)

Consider the following directory listing:

```
-rw-r----- 3  diego staff 512 Nov 28 14:23  exam
```

d. What kind of access will a user in group admin have to the file exam? (1p)
e. What kind of acces does diego have to the file exam? (1p)
f. How can diego execute the file exam? (1p)
g. How can we tell from the listing that exam is not a directory? (1p)

a. Elements (2p for more than 5 correct):
    a. User name: up to eight characters long
    b. Password: stored "encrypted" (really a hash)
    c. UID: user identifier for access control (user ID)
    d. GID: user's primary group (group ID)
    e. ID string: user's full name
    f. Home directory: The user's home directory
    g. Login shell: program started after successful log in

b. /etc/passwd is world readable, /etc/shadow is only readable by root. (1p)
   /etc/passwd contains user info, /etc/shadow contains the hashed password (and other password related information, such as salt, password creation time) (1p)
c. 2p for: In Linux there are only positive access rights, and permission bits are applied in the order of priority:
    1. owner
    2. group
    3. world
d. No access (1p)
e. Read and write (1p)
f. He is the file owner, so he can change the file permissions to give himself the execute right (1p)
g. The character d would be in front of the permissions: `drw-r-----` (1p)

Question 7: The registry

a. What is the role of the registry in Windows? (2p)
b. Mention two security related hives. (2p)
c. Some hives are not readable for a normal user starting RegEdit. How can you read the contents of these hives? (2p)
d. Is your password stored in the registry? How? (2p)
e. How can you detect the changes to a registry done by a piece of malware? (2p)

a. Stores low-level settings for Windows and applications
b. HKLM\SAM, HKLM\Security
c. Open as SYSTEM (psexec.exe –s –i regedit.exe
d. Yes, in the SAM, as a hash
e. Using Process Monitor, filtering on registry activity

Question 8: Assembly

Consider the x86 assembly instructions

```
add eax, 11
xor eax, eax
mov eax, 5
add eax, 1
push eax
```

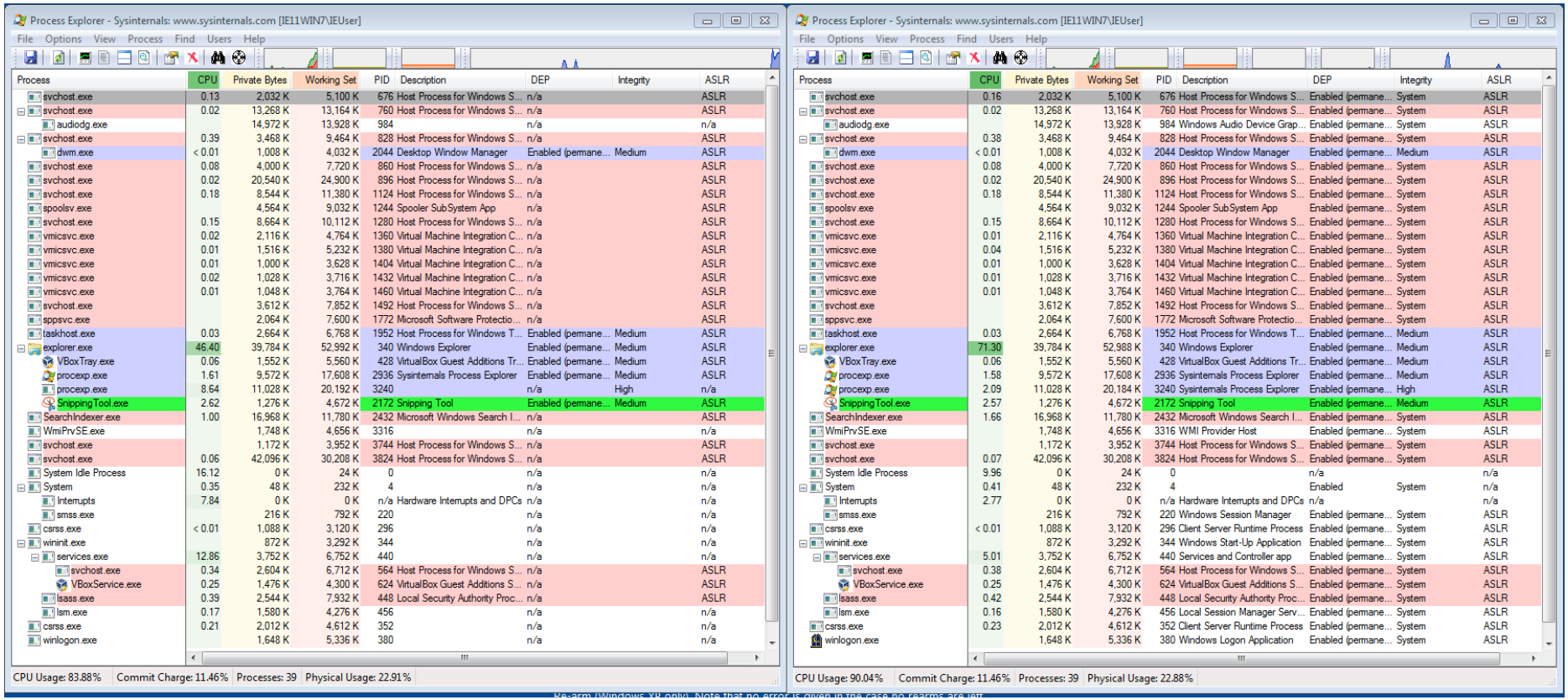
a. Explain what each line does. (6p)
b. What is the value of eax after these instructions are executed? (2p)
c. Explain the role of the stack during process execution. (2p)

Question 8: Process explorer

Two instances of process explorer are running on this system, and the screenshots in the appendix show the information in both of them.

a. What is the difference between these two processes? (2p)
b. What are integrity levels in Windows? (2p)
c. Why are the integrity levels different when they are started by the same user? (2p)
d. What is User Account Control (UAC) (2p)
e. How is UAC related to the tokens of the two process explorer processes? (2p)

Four int levels: Low, Medium, High, System.
Mandatory checks, cannot be turned off. Dominates DACL: principal cannot write to higher int, even if DACL permits.
Low integrity executables receives low integrity. Used to isolate procs handling untrusted data.
A mechanism for letting admins have as low privs as possible, only using admin privs when needed. Standard token used by default, admin token prompted for.
One has standard token, other has admin token.


Process Explorer - Sysinternals: www.sysinternals.com [IE11WIN7\IEUser]
File Options View Process Find Users Help
Process CPU Private Bytes Working Set PID Description DEP Integrity ASLR
svchost.exe 0.13 2,032 K 5,100 K 676 Host Process for Windows S... n/a ASLR
svchost.exe 0.02 13,268 K 13,164 K 760 Host Process for Windows S... n/a ASLR
audiodg.exe 14,972 K 13,928 K 984 n/a n/a
svchost.exe 0.39 3,468 K 9,464 K 828 Host Process for Windows S... n/a ASLR
dwm.exe < 0.01 1,008 K 4,032 K 2044 Desktop Window Manager Enabled (permane... Medium ASLR
svchost.exe 0.08 4,000 K 7,720 K 860 Host Process for Windows S... n/a ASLR
svchost.exe 0.02 20,540 K 24,900 K 896 Host Process for Windows S... n/a ASLR
svchost.exe 0.18 8,544 K 11,380 K 1124 Host Process for Windows S... n/a ASLR
spoolsv.exe 4,564 K 9,032 K 1244 Spooler SubSystem App n/a ASLR
svchost.exe 0.15 8,664 K 10,112 K 1280 Host Process for Windows S... n/a ASLR
vmicsvc.exe 0.02 2,116 K 4,764 K 1360 Virtual Machine Integration C... n/a ASLR
vmicsvc.exe 0.01 1,516 K 5,232 K 1380 Virtual Machine Integration C... n/a ASLR
vmicsvc.exe 0.01 1,000 K 3,628 K 1404 Virtual Machine Integration C... n/a ASLR
vmicsvc.exe 0.02 1,028 K 3,716 K 1432 Virtual Machine Integration C... n/a ASLR
vmicsvc.exe 0.01 1,048 K 3,764 K 1460 Virtual Machine Integration C... n/a ASLR
svchost.exe 3,612 K 7,852 K 1492 Host Process for Windows S... n/a ASLR
sppsvc.exe 2,064 K 7,600 K 1772 Microsoft Software Protectio... n/a ASLR
taskhost.exe 0.03 2,664 K 6,768 K 1952 Host Process for Windows T... Enabled (permane... Medium ASLR
explorer.exe 46.40 39,784 K 52,992 K 340 Windows Explorer Enabled (permane... Medium ASLR
VBoxTray.exe 0.06 1,552 K 5,560 K 428 VirtualBox Guest Additions Tr... Enabled (permane... Medium ASLR
procexp.exe 1.61 9,572 K 17,608 K 2936 Sysinternals Process Explorer Enabled (permane... Medium ASLR
procexp.exe 8.64 11,028 K 20,192 K 3240 n/a High n/a
SnippingTool.exe 2.62 1,276 K 4,672 K 2172 Snipping Tool Enabled (permane... Medium ASLR
SearchIndexer.exe 1.00 16,968 K 11,780 K 2432 Microsoft Windows Search I... n/a ASLR
WmiPrvSE.exe 1,748 K 4,656 K 3316 n/a n/a
svchost.exe 1,172 K 3,952 K 3744 Host Process for Windows S... n/a ASLR
svchost.exe 0.06 42,096 K 30,208 K 3824 Host Process for Windows S... n/a ASLR
System Idle Process 16.12 0 K 24 K 0 n/a n/a
System 0.35 48 K 232 K 4 n/a n/a
Interrupts 7.84 0 K 0 K n/a Hardware Interrupts and DPCs n/a n/a
smss.exe 216 K 792 K 220 n/a n/a
csrss.exe < 0.01 1,088 K 3,120 K 296 n/a n/a
wininit.exe 872 K 3,292 K 344 n/a n/a
services.exe 12.86 3,752 K 6,752 K 440 n/a n/a
svchost.exe 0.34 2,604 K 6,712 K 564 Host Process for Windows S... n/a ASLR
VBoxService.exe 0.25 1,476 K 4,300 K 624 VirtualBox Guest Additions S... n/a ASLR
lsass.exe 0.39 2,544 K 7,932 K 448 Local Security Authority Proc... n/a ASLR
lsm.exe 0.17 1,580 K 4,276 K 456 n/a n/a
csrss.exe 0.21 2,012 K 4,612 K 352 n/a n/a
winlogon.exe 1,648 K 5,336 K 380 n/a n/a
CPU Usage: 83.88% Commit Charge: 11.46% Processes: 39 Physical Usage: 22.91%
Process Explorer - Sysinternals: www.sysinternals.com [IE11WIN7\IEUser]
File Options View Process Find Users Help
Process CPU Private Bytes Working Set PID Description DEP Integrity ASLR
svchost.exe 0.16 2,032 K 5,100 K 676 Host Process for Windows S... Enabled (permane... System ASLR
svchost.exe 0.02 13,268 K 13,164 K 760 Host Process for Windows S... Enabled (permane... System ASLR
audiodg.exe 14,972 K 13,928 K 984 Windows Audio Device Grap... Enabled (permane... System ASLR
svchost.exe 0.38 3,468 K 9,464 K 828 Host Process for Windows S... Enabled (permane... System ASLR
dwm.exe < 0.01 1,008 K 4,032 K 2044 Desktop Window Manager Enabled (permane... Medium ASLR
svchost.exe 0.08 4,000 K 7,720 K 860 Host Process for Windows S... Enabled (permane... System ASLR
svchost.exe 0.02 20,540 K 24,900 K 896 Host Process for Windows S... Enabled (permane... System ASLR
svchost.exe 0.18 8,544 K 11,380 K 1124 Host Process for Windows S... Enabled (permane... System ASLR
spoolsv.exe 4,564 K 9,032 K 1244 Spooler SubSystem App Enabled (permane... System ASLR
svchost.exe 0.15 8,664 K 10,112 K 1280 Host Process for Windows S... Enabled (permane... System ASLR
vmicsvc.exe 0.01 2,116 K 4,764 K 1360 Virtual Machine Integration C... Enabled (permane... System ASLR
vmicsvc.exe 0.04 1,516 K 5,232 K 1380 Virtual Machine Integration C... Enabled (permane... System ASLR
vmicsvc.exe 0.01 1,000 K 3,628 K 1404 Virtual Machine Integration C... Enabled (permane... System ASLR
vmicsvc.exe 0.01 1,028 K 3,716 K 1432 Virtual Machine Integration C... Enabled (permane... System ASLR
vmicsvc.exe 0.01 1,048 K 3,764 K 1460 Virtual Machine Integration C... Enabled (permane... System ASLR
svchost.exe 3,612 K 7,852 K 1492 Host Process for Windows S... Enabled (permane... System ASLR
sppsvc.exe 2,064 K 7,600 K 1772 Microsoft Software Protectio... Enabled (permane... System ASLR
taskhost.exe 0.03 2,664 K 6,768 K 1952 Host Process for Windows T... Enabled (permane... Medium ASLR
explorer.exe 71.30 39,784 K 52,988 K 340 Windows Explorer Enabled (permane... Medium ASLR
VBoxTray.exe 0.06 1,552 K 5,560 K 428 VirtualBox Guest Additions Tr... Enabled (permane... Medium ASLR
procexp.exe 1.58 9,572 K 17,608 K 2936 Sysinternals Process Explorer Enabled (permane... Medium ASLR
procexp.exe 2.09 11,028 K 20,184 K 3240 Sysinternals Process Explorer Enabled (permane... High ASLR
SnippingTool.exe 2.57 1,276 K 4,672 K 2172 Snipping Tool Enabled (permane... Medium ASLR
SearchIndexer.exe 1.66 16,968 K 11,780 K 2432 Microsoft Windows Search I... Enabled (permane... System ASLR
WmiPrvSE.exe 1,748 K 4,656 K 3316 WMI Provider Host Enabled (permane... System ASLR
svchost.exe 1,172 K 3,952 K 3744 Host Process for Windows S... Enabled (permane... System ASLR
svchost.exe 0.07 42,096 K 30,208 K 3824 Host Process for Windows S... Enabled (permane... System ASLR
System Idle Process 9.96 0 K 24 K 0 n/a n/a
System 0.41 48 K 232 K 4 Enabled System n/a
Interrupts 2.77 0 K 0 K n/a Hardware Interrupts and DPCs n/a n/a
smss.exe 216 K 792 K 220 Windows Session Manager Enabled (permane... System ASLR
csrss.exe < 0.01 1,088 K 3,120 K 296 Client Server Runtime Process Enabled (permane... System ASLR
wininit.exe 872 K 3,292 K 344 Windows Start-Up Application Enabled (permane... System ASLR
services.exe 5.01 3,752 K 6,752 K 440 Services and Controller app Enabled (permane... System ASLR
svchost.exe 0.38 2,604 K 6,712 K 564 Host Process for Windows S... Enabled (permane... System ASLR
VBoxService.exe 0.25 1,476 K 4,300 K 624 VirtualBox Guest Additions S... Enabled (permane... System ASLR
lsass.exe 0.42 2,544 K 7,932 K 448 Local Security Authority Proc... Enabled (permane... System ASLR
lsm.exe 0.16 1,580 K 4,276 K 456 Local Session Manager Serv... Enabled (permane... System ASLR
csrss.exe 0.23 2,012 K 4,612 K 352 Client Server Runtime Process Enabled (permane... System ASLR
winlogon.exe 1,648 K 5,336 K 380 Windows Logon Application Enabled (permane... System ASLR
CPU Usage: 90.04% Commit Charge: 11.46% Processes: 39 Physical Usage: 22.88%