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)

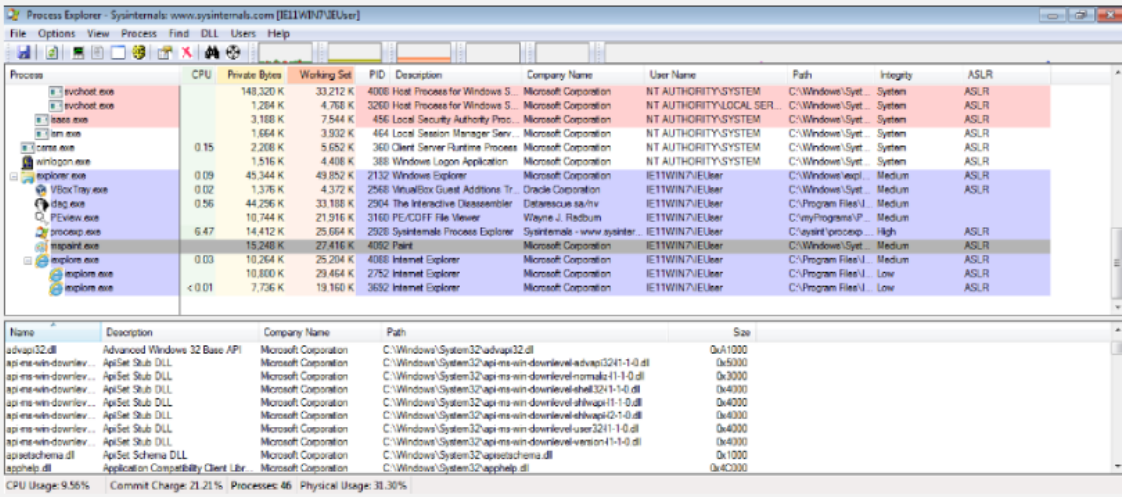
Answer:

- 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.

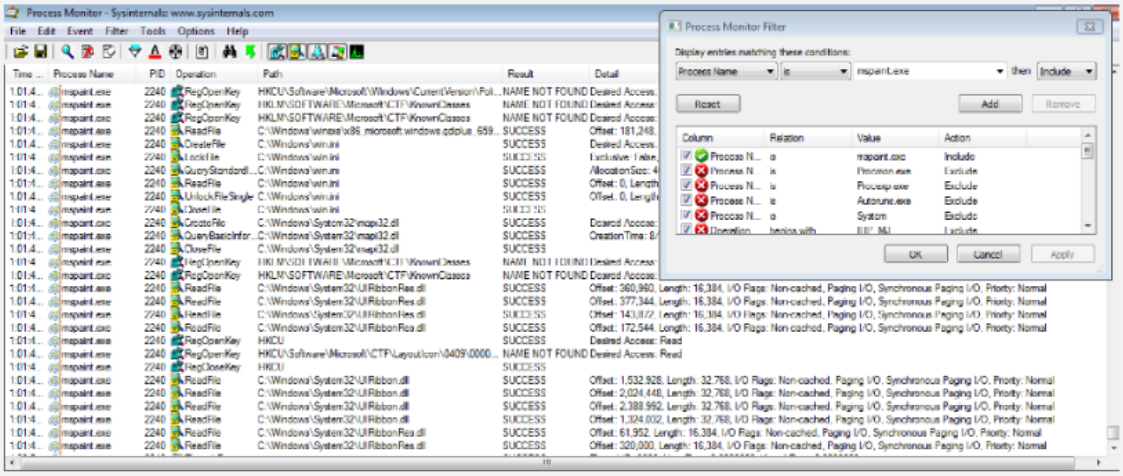
(5 points)

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.

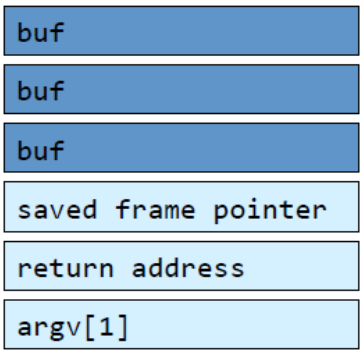
1 p for each of these. Similar or other reasonable answers should be accepted, but maximum 5 p total. Filters must be mentioned to achieve full score.

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)

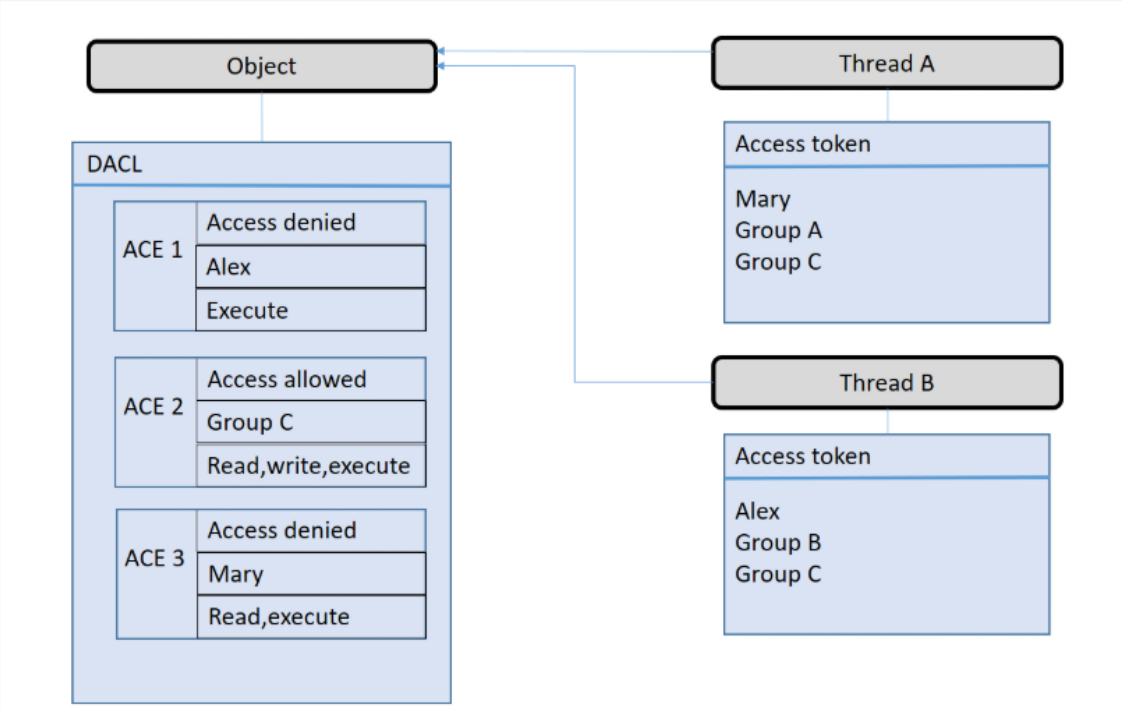
Answer:

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)

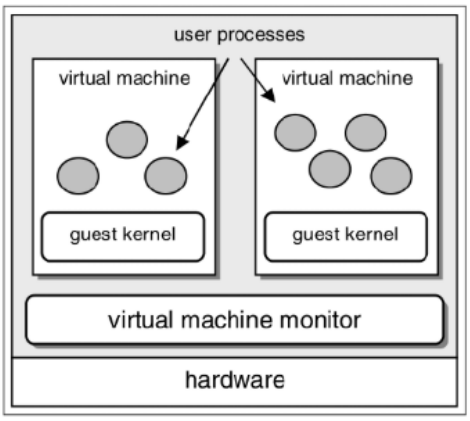
Answer:

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

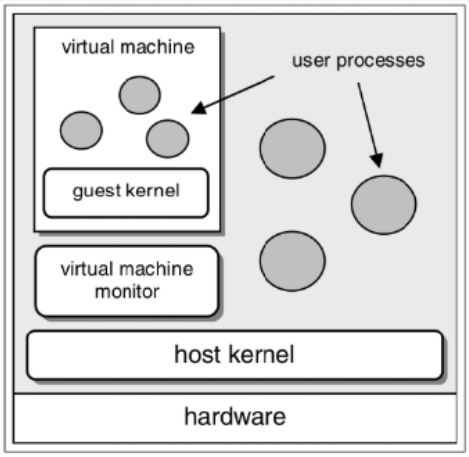
The attacks want to gain access to a secret. The attacker makes the secret be used indirectly in a **speculatively executed instruction (also accepted: out of order execution)**, for example as an index of an array. This instruction stores an array element into an internal cache on the processor. The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