Answers to Exercise 1

The fundamental information security principles are often called the CIA triad. Which three principles form this triad?

- ☐ Authentication
- ☐ **Confidentiality**
- ☐ Accessibility
- ☐ Identity
- ☐ **Availability**
- ☐ Classification
- ☐ Intimate
- ☐ **Integrity**
- ☐ Control

The CIA triad of information security comprises the concepts *Confidentiality*, *Integrity*, and *Availability*.

Answers to Exercise 2

1. Let's assume an attacker steals a valid credit card. For misusing the card, the thief needs to circumvent the following mechanism:
    - **Signature (and magnetic strip):** The thief only needs to forge the signature.
    - **Offline PIN verification (and secure chip):** The thief would need to obtain the PIN.
    - **Online PIN verification:** The thief would need to obtain the PIN.

The first signature method is very insecure as the thief only needs to forge the signature. To make it even more easy, a sample of the signature is on the backside of the card and signatures are usually not checked very thoroughly.

The second and third method require a PIN entry. As cards are locked after a certain number of failed PIN entries (usually after three failed attempts) and as PINs are usually at least four digits it is much harder to guess a PIN than to forge a signature. One can argue that the online method is more secure than the chip-based offline variant, as, e.g., a mechanism for enforcing at most three pin entries that is implemented on the chip could be circumvented easier (respectively, the risk of being detected is much higher in the online scenario).

2. Let's have a look on the integrity of the data stored on the magnetic strip or the chip:
    - **Signature (and magnetic strip):** The magnetic strip can be easily read (and written) with a swipe-card reader that costs less than 50 pounds. There is no protection. Thus, it is easy to clone magnetic strip cards as well as read/write their content.
    - **Offline PIN verification (and secure chip):** The secure chip protects the stored data using cryptographic protocols. Thus, only authorized groups are able to modify the data easily.
    - **Online PIN verification:** The secure chip protects the stored data using cryptographic protocols. Thus, only authorized groups are able to modify the data easily. Similarly, we assume that modifying PIN data stored in the credit card network is much harder to change than an easily modifiable magnetic strip.

   Again, the chip-based methods are much better than the magnetic strip.

3. Let's have a look on the confidentiality of the data:
    - The card number and the signature (as a blueprint for practicing a fake signature) are readable by anybody obtaining the card (e.g., any shop clerk or a thief). None of them is confidential.
    - The information on the magnetic strip (e.g., card number, name of holder, expiration date) is not encrypted and can also be read by anybody with a simple card reader (which can easily be obtained for less than GBP 50).
    - The information stored on the chip (e.g., the PIN) is stored in a cryptographically secure way and cannot (or only with high effort) be read by an unauthorized party (we will discuss encryption in more detail in one of the following lectures).
    - Also the PIN stored in network of the card issuer (for the online verification) is stored securely. Still, the attack surface might be larger as more systems (compared to the chip-based verification) are involved.

Overall, the chip and the online verification offer a similar level of confidentiality of the PIN (They differ in various details and attacks are possible on both of them. Still they require quite some effort. The details are out of scope of this lecture). In contrast, the way the information about the card and the card account (name of card holder, card number, expiration date) either stored on the magnetic strip or the card itself does not provide any means of confidentiality (as anybody that can access the card can easily read this information).

4. Let’s have a look at the availability of the different approaches:
    - **Signature (and magnetic strip):** All information necessary for validating the card holder’s identity is on the card and can be checked manually. Thus, the availability is very high.
    - **Offline PIN verification (and secure chip):** All information necessary for validating the card holder’s identity is on the card and can be checked offline by a chip card terminal. Thus, the availability is still very high (but not as high as for the signature-based method, as a card reader and electricity is required which could fail).
    - **Online PIN verification:** The online PIN verification requires a working network connection which could fail easily. Thus, this method has the lowest availability.

The signature-based approach has the highest availability, followed closely by the offline PIN verification method.

Answers to Exercise 3

1. We start by defining the users and roles:

$$ROLES = \{\text{lecturer, demonstrator, student}\}$$

$$USERS = \{\text{elif, bilge, kavun, alice, bob}\}$$

Thereafter, we define the permissions:

$$\begin{aligned} PERMISSION = \{&\text{read\_6090\_slides,write\_6090\_slides,} \\ &\text{read\_6090\_exam,write\_6090\_exam,} \\ &\text{read\_6090\_solutions,write\_6090\_solutions}\} \end{aligned}$$

Now we can define the relations UA and PA:

$$\begin{aligned} UA = \{&\text{(elif,lecturer),} \\ &\text{(bilge,lecturer),} \\ &\text{(kavun,demonstrator),} \\ &\text{(alice,student),} \\ &\text{(bob,student)}\} \end{aligned}$$

$$\begin{aligned} PA=\{& \\ &\text{(lecturer, read\_6090\_slides), (lecturer, write\_6090\_slides),} \\ &\text{(lecturer, read\_6090\_exam), (lecturer, write\_6090\_exam)} \\ &\text{(lecturer, read\_6090\_solutions), (lecturer, write\_6090\_solutions),} \\ &\text{(demonstrator, read\_6090\_slides), (demonstrator, write\_6090\_slides),} \\ &\text{(demonstrator, read\_6090\_exam),} \\ &\text{(demonstrator, read\_6090\_solutions),} \\ &\text{(student, read\_6090\_slides),} \\ &\text{(student, read\_6090\_exam)}\} \end{aligned}$$

Our role-based access control configuration is now defined by

$$AC = PA \circ UA$$

and we can, e.g., obtain the permission of **`bob`** by

$$\begin{aligned} AC(\text{bob}) &= (PA \circ UA)(\text{bob}) = PA(UA(\text{bob})) \\ &= PA(\text{student}) \\ &= \{\text{read\_6090\_slides, read\_6090\_exam}\} \end{aligned}$$

2. To define hierarchic RBAC, we introduce a relation $RH \subset ROLES \times ROLES$ that defines the role hierarchy. In our example, we have

   $RH =$ {(lecturer,lecturer),(lecturer, demonstrator),
   (demonstrator, demonstrator), (demonstrator, student),
   (student, student)}

   The transitive closure of the role hierarchy relation (RH^*) allows us to obtain "all effective" roles of a user. The transitive closure contains all transitive applications of RH. For example, it also includes the tuple (lecturer, student) as lecturer have all permissions of demonstrators and demonstrators have all permissions of students.

   Thus, we can simplify the PA relation to

   PA={
   (lecturer, write_6090_exam),
   (lecturer, write_6090_solutions),
   (demonstrator, write_6090_slides),
   (demonstrator, read_6090_solutions),
   (student, read_6090_slides),
   (student, read_6090_exam)}

3. In our simple RBAC model, we can model that students can read lecture slides. This permission gives any student read access to slides of all lectures offered. Our model is not strong enough to express the more realist permission that students grants only read access to slides of lectures they are registered.

   A simple method for introduction constraints is to extend the PA relation:

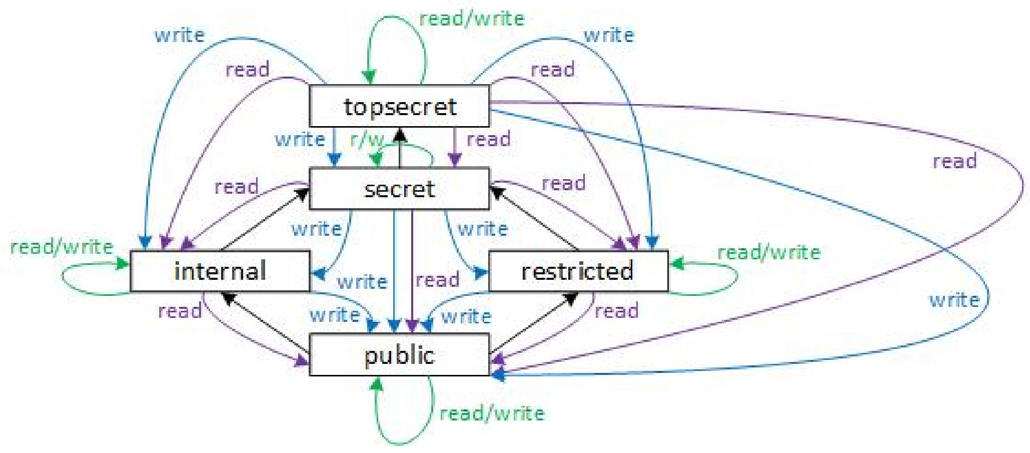
   $$PA_C \subset ROLES \times CONSTRAINT \times PERMISSION$$

   For simplicity, we can assume that a $CONSTRAINT$ is a Boolean predicate. Thus, we could specify our requirement as follows:

   (student, is_registered_for_6090(*user*), read_6090_slides)

   where we assume that user contains the *user* (e.g., **bob**) for whom a permission check (access control request) is evaluated and is_registered_for_6090(*user*) is a Boolean predicate describing if a user attends 6090. For our purposes, you can think of it as a Boolean function that takes a user and returns true if the user is registered for 6090 and false otherwise.

Answers to Exercise 4



1. {topsecret, secret, restricted}
2. {topsecret, secret} (Note that internal and restricted cannot read and write to each other.)

Answers to Exercise 5

1. To simulate RBAC using a DAC model, we first need to implement users and roles. DAC directly supports users. For example, on Linux system we can easily create the necessary users, resulting in the following entries in the **`/etc/passwd`**:

   ```
   elif:x:1001:1001::/home/elif:/bin/bash
   bilge:x:1002:1001::/home/bilge:/bin/bash
   kavun:x:1003:1001::/home/kavun:/bin/bash
   alice:x:1004:1001::/home/alice:/bin/bash
   bob:x:1005:1001::/home/bob:/bin/bash
   ```

   In a next step, we need to simulate roles. DAC only provides groups. As users can be a member of multiple groups, we can easily simulate hierarchical roles by
   - introducing a group for each role
   - assigning users transitively to all groups representing roles the user belongs to

   In our example, this results in the following entries in the **`/etc/groups`** table:

   ```
   lecturer:x:1100: elif, bilge
   demonstrator:x:1101: elif, bilge, kavun
   student:x:1101: elif, bilge, kavun, alice, bob
   ```

   Finally, we need to carefully check the access rights on files:
   - In our model, lecturer can read and write exams, everybody else can read them:
     ```
     --- rw- r-- uni lecturer 4096 Nov 5 12:06 6090_exam
     ```
     where uni is not a member of the groups lecturer, demonstrator, or student.
   - In our model, lecturers can read and write exams, and demonstrators can read exams. In our DAC implementation, we exploit the fact that we only have one demonstrator:
     ```
     r-- rw- --- kavun lecturer 4096 Nov 5 12:06 6090_solutions
     ```
   - In our model, lecturers and demonstrators can read and write slides, everybody else can read them:
     ```
     --- rw- r-- uni demonstrator 4096 Nov 5 12:06 6090_slides
     ```
     where uni is not a member of the groups lecturer, demonstrator, or student.
2. A non-hierarchical RBAC model where the permissions for each role are disjoint can simulated by a DAC model where there is one group for each role. An RBAC configuration where permissions are not disjoint (or with effective role hierarchies), a simple mapping is not possible. We can, though, map many actual RBAC implementations if we are able to constrain the DAC configuration, e.g., to prevent access by users in the DAC system that are not required for simulating the RBAC part.

Answers to Exercise 6

1. The four directories can be easily created with default access rights using **`mkdir`**:
   ```
   mkdir test01 test02 test03 test04
   chgrp -R mygroup test0*
   ```
2. The default access rights can be changed with **`chmod`**:
   ```
   mkdir test01 test02 test03 test04
   chmod 777 test01
   chmod 770 test02
   chmod g+s test02
   chmod 1777 test03
   chmod 1007 test04
   ```
   The gid-bit enforces that new files created in this directory are owned by the group **`mygroup`** and not by any other group the user might be a member of. The file is even owned by the group **`mygroup`** if the file creator is not member of that group.
3. The sticky bit restricts the write permissions on the files that are in the directory that has the sticky bit set such that only the owner of a file can delete it. To understand this, we need to recall a subtlety of the POSIX file permissions: the right to delete a file is not attached to the file itself, it is derived from the permissions of the directory containing that file. Technically, this can be explained by the fact that deleting a file modifies the directory table of the directory and, thus, is (as creating of files) write operation on the directory. Thus, in a directory
   ```
   drwxrwxrwx 2 kavun mygroup 4096 Nov 5 12:05 tmp
   ```
   every user of the system can create and delete files in this directory – regardless if he or she owns that file. Thus, this is not the right set of permissions of a **`tmp`** directory in which every user should be able to store temporary data, as it does not protect the integrity against modification/deletions by other users. In contrast, in a directory
   ```
   drwxrwxrwt 2 kavun mygroup 4096 Nov 5 12:05 tmp
   ```
   every user of the system can create new files but only the owner of a file can delete a file (recall that the permission for reading a file or writing into a file is attached to the file itself). Thus, **`tmp`**-directories need to have the sticky bit set.
4. Only users that are neither the owner of the file nor member of the group owning the file can access this directory. It is noteworthy though that the owner can change the permissions (e.g., using **`chmod`**) and gain again access to the content of the directory.
5. Only users that are neither the owner of the file nor member of the group owning the file can access this directory. Note that the owner can change the permissions (e.g., using **`chmod`**) and gain again access to the content of the directory.