Introduction to IT Security - Exercice sheet 1

Vianney HERVY

1. User Authentication: Passwords

There are 26 possible characters for each one of the 14 symbols of the password. This gives $14^{26} \approx 6.45 \cdot 10^{19}$ different passwords.

1.1. Base case

To get a specific password, it would take the attacker $6.45 \cdot 10^{19}$ attempts (number of passwords).

He can make $40000_{\text{tries/s}} \times 86400_{\text{s/day}} \approx 3.46 \cdot 10^{9}$ tried per day.

It would then take him $\frac{6.45 \cdot 10^{19}{}_{\text{tries}}}{3.46 \cdot 10^{9}{}_{\text{tries/day}}} \approx 1.87 \cdot 10^{10}$.

This result is the worse case, if the targeted password is tried last. On average, it would take him half that time (same order of magnitude)

1.2. Irregular distribution

The answer here depends on the attacker's startegy. The optimal one is to only test the 1,000 popular passwords and get 90% of the passwords in a fifth of a second. But having a correct password isn't enough; the user has to match as well ("*The right key is nothing without the right lock*").

There are $5000 \times 90\% = 4500$ vulnerable users.

Each attempt is a user-password pair. The probability that that specific user has that specific password from the popular list is:

$$P(\text{User is vulnerable}) \times P(\text{User chose this password}) = 90\% \times \frac{1}{1000} = 0.0009$$

To get 90 passwords, the attacker would have to do $\frac{90}{0.0009} = 1 \cdot 10^{5}$ attempts

Given a 5000 rate, it would take him $\frac{100000_{\text{tries}}}{5000_{\text{tries/s}}} = 20$ seconds to find a matching pair.

1.3. Defense

The probability of success is:

$$P_{\text{success}} = \left(P_{\text{secure}} \times P_{\text{success|secure}}\right) + \left(P_{\text{vulnerable}} \times P_{\text{success|vulnerable}}\right)$$

$$P_{\text{success}} = ((1 - 90\%) \times 0) + \left(90\% \times \frac{N}{1000}\right)$$

2. Access Control

2.1. Mobile Apps

Pinterest is a social media where users can view, share, comment and save "pins". Pinterest asks access to the contacts (read), location (read), camera (read) and storage (read/write). Too Good To Go connects customers to restaurants and stores to save unsold food surplus. The app requires the user's location (read) and their agenda (write). Bonjour RATP is the official app of the parisian public transportation company. It requires access to the user's location (read) and to their physical activity data (read)

All these permissions are grouped from the app's perpective, the access rights are associated with the subject (the app). This is a capability list. Mobile systems use capability lists because each app runs in its own sandbox and needs only small, explicit permissions instead of complex resource-based rules. Capabilities are simpler for users and developers, enforce least privilege cleanly, and avoid the overhead of maintaining global access control lists. This model scales better, is easier to secure, and matches how mobile OS APIs are designed.

2.2. Role-based Access Control

Roles allow permission grouping to avoid granting the same permissions again and again. Inheritance is also permitted, enabling simpler permission distribution and management. Roles may also be parameterized or hierarchized.

In an application like Moodle, the users can be grouped by role with parameters: "student of [program]", "lecturer of [subject]", "head of [department]". The permissions can be direct functions of these parameters: student can only read a course he's taking, lecturer can only write a course he's giving etc.

3. Application Whitelisting

1. Explain how application whitelisting works with WDAC (Windows Defender Application Control)

   Windows Defender Application Control enforces application whitelisting by using a policy that defines trusted apps (via the publisher, package, etc). Every executable, script or driver is checked against this policy, allowing only verified (approved) apps to run and blocking all others. The philosophy is that an unknown app is considered potentially undafe until proven otherwise.

2. Does WDAC restrict users/processes with respect to executing code ? How ?

   Yes, WDAC restricts code execution by enforcing its policy on all user accounts and processes. When a user or a process attempts to run an app, WDAC checks if the whitelist allows the application. If it is not approved, the execution is denied.

3. Does WDAC restrict users/processes with respect to reading and writing files ? How ?

   No, WDAC does not control read/write operations. Its scope is limited to the "execute" operation. On the other hand, it can block an app opened to read/write to a file.

4. What are the different file rules hat you can use with WDAC ?

   WDAC supports publisher rules (verified via digital signatures), file hash rules (ensuring installed/downloaded content is identical to expected) and path rules (based on location in the file system).

5. What is the purpose of the audit mode in WDAC ?

   WDAC Audit Mode is a mode where the policy is enforced only for logging. It does not block any application from running but it records events about what would have been blocked if the policy were enforced. It is like a "test playground" where to test rules and policies before applying them.

6. How do you debug WDAC policies, i.e., how do you find out that they are effective and where are WDAC events logged ?

   Debugging WDAC policies relies on the Audit Mode to check effectiveness and identify missing policy rules. The WDAC events are systematically logged into the **Application and Services Logs\Microsoft\Windows\CodeIntegrity\Operational** event log.