Høgskolen i Gjøvik

Software Security

Obligatory exercise #2

Race conditions

Author:
120915 - Halvor Mydske Thoresen

September 17, 2014

1 Task

- Imagine an electronic id solution (e.g. login to e-voting).
- Think about two (or more) possible race conditions and their effects.
- Describe countermeasures for the race conditions you observe

2 Scenario

This assignment will be taking a look at a fictional E-voting system with little to no security measures in place, how it can be affected by possible race conditions, and describe countermeasures to prevent or mitigate the conditions.

3 Condition #1

For a voting system to be effective, it has to store the votes somehow. In this fictional system the votes are stored in a variable X.
Lets say two voters vote for the same candidate at the same time. This will create two threads that are both editing the same variable and we have a race condition.
The threads edit the variable with the fomula: X = X + 1. To do this the thread have to execute multiple steps.

1. The thread has to read the current value of X.
2. The thread has to temporarily store this value in memory.
3. Add 1 to the temp value.
4. Write the temp value back to X.

This is all fine until we have two threads working at the same time.

Thread 1: Reads X = 0
Thread 2: Reads X = 0
Thread 1: Adds 1 to the temp value of 0. And gets 1.
Thread 2: Adds 1 to the temp value of 0. And gets 1.
Thread 1: Writes 1 back to X.
Thread 2: Writes 1 back to X.

After two votes are submitted, the number of votes is 1 instead of 2.

4 Solution

To prevent threads from modifying the same variable at the same time, locks can be implemented.

- obtain lock
- Read X
- Add 1
- Write X
- release lock

If the a thread can't obtain the lock, it has to wait for the other thread to release it before executing the code. This ensures that the correct value is read and modified. [1]

5 Condition #2

With an E-voting system, a lot of people will most likely log into the system at the same time. This fictional system temporarily stores the login credentials in a variable while checking the database and verifying the user. It then creates a login session and gives it to the user in the variable. A big problem arises when the user in the variable has been changed by another process/thread while the database verification was going on. The new user will now get the wrong login session.

6 Solution

This might be a very unlikely case, but a solution to prevent this would be to lock down the variables while the verification process is working, to ensure that the user who is getting verified is the one getting the login session. [2]

References

[1] Microsoft. Description of race conditions and deadlock. https://support.microsoft.com/kb/317723, 2012. [Online; accessed 17-September-2014].

[2] mitre.org. Time-of-check time-of-use (toctou) race condition. http://cwe.mitre.org/data/definitions/367.html, 2014. [Online; accessed 17-September-2014].