Mandatory assignment 2
Electronic ID

- AV: Jan Petter Berg Nilsen
- Studnr: 120505

This is the second Mandatory assignment in IMT3501 Software security and this one will revolve around electronic id solution, their race conditions, effects and solutions. In this rapport we will look at at online voting system. Online voting would make it easier to vote for both a president, political parties and other stuff online. This will make more people use their vote, because it will give them easier access to vote from their own pc, tablet or phone. An online voting system will also give some challenges for the developers and drifters, that will develope and maintain the system. One of these challenges is race condition, which we are going to look at in this rapport.

Here is a description of the online voting system. Here you see the working process for a user that is going to vote. The user goes on to the web page, gets access with a user name and password and then the voting process starts, before the user submit the vote and logs out. The race condition we are going to look at inside this system, is if a user can get more than one vote when authenticating to the system and overwriting the vote count. But first we will look at what a race condition is.

Race condition

A race condition is a type of bug in an electronic- or software system. The type of race condition we will look at in this task, is the one for software system. Examples of a race condition in a software system, can be in multi-threaded or distributed programs [2]. So a race condition can be when two threads are executing and have access to a shared resource like a variable or a memory, that will produce different results, depending on the time that the code is executed [1]. That means that when two threads have access to a shared resource, overwriting will happen if there is not implemented any safe mechanisms, to provent it from happening. This type of problems can cause big problems for both the user of the system and the result of what the system was going to be used for.

Possible Race Condition, effects and solutions

Race conditions vote more the once

Before the user can start to vote, it needs to get access to the voting page. The user gets access with a username and a password and then the person can start the voting process. The example of the possible race condition for this operation is: Can a user get access to the system more then once, so that the person gets more than one vote? The person can get access to the voting page from different devices or the person can get access multiple times with the same device.

The possible race condition is that when a user logs on to a device, and before it submits the vote, it will also access with another device, so it can submit its vote two or more times. It can also be that a person can submit the vote more than once from the same device, by only re- submitting the vote or get access to the voting page more than once with the same username and password, when the person has already submitted a vote.

Race conditions vote counts

When the online voting system is open and ready to use, there will be a lot of people that will use the system simultaneously. That means that at some point, some people will vote for the same president or a political party at the same time. Without the right safe mechanisms, the race condition can occur and that means that a overwrite will happen to one or more of the users, that is trying to vote.

An example and a possible race condition is when we have two threads or voters. Voter1 and voter2 both access the online voting system and both are ready to vote. They both choose the same option and submit the vote at the same time.

voter1 votecount++

vote2 votecount++

What will happen is that voter1 will execute the voting process, but voter2 will interrupt voter1 before voter1 can read to the counter variable votecount++. Voter1 will then need to wait until voter2 is done with its execution and have written to the count variable votecount++. Voter1 gets back the control and can pick up where it was stopped by voter2. Voter1 then writes to the counter variable votecount++ and then overwrites the one vote from voter2 on the counter variable votecount++ , so voter2's vote will not count to the final count to see witch president or political party that wins.

Race conditions consequences and effect

The consequences for this type of race condition is that it will effect the voting numbers in the way that not all votes will get counted, because of overwriting or that someone can get more votes than they are supposed to get. This could be for people that try to control the system, and try to control the outcome of

the voting, but it also can happen without the user knowing it. If we are going to look at this in worst case scenario, it can effect the outcome of the voting and then it will be a non legit result. This could result in that a president or a political party, that does not deserve to win, wins. If it doesn't get detected, it could also effect the country in a small or a big way, but also the world. I guess in a situation like this, that have a high priority in both security and justice, they will have a type of solution to detect that there is a problem and then solve it somehow.

Race Condition solutions

The solutions for both of these race conditions, is importent so that the voting process will go as it should, without being tempered with in any way. For the first race condition it would be necessary to have a checking function, so that when a user authenticates, the system check the authentication to see if it's good or not. Then the function checks if the user has voted before. If the person has, it will get an info- message about this, and if not, the person will get access to the submit form. The function also only saves one vote from each user, so that one user only gets one vote, even if the person has gotten the opportunity to vote twice. The session will look something like this:

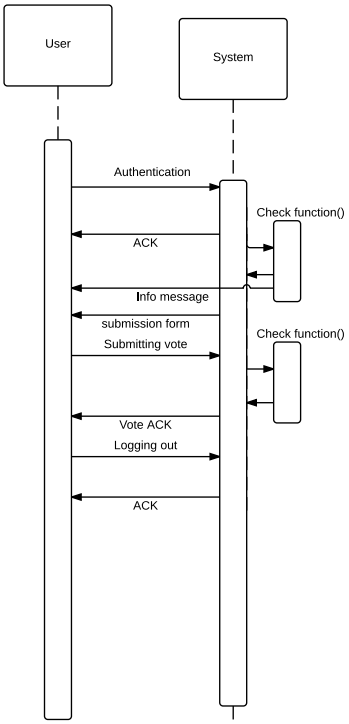

Figure 1: Session diagram

For the second solution it would be the possibility to use a mutex. Mutex is a program object that allows multiple program threads to share the same resource, but not at the sometime. this will help to prevent the possibility for overwriting, because the mutex will always protect the critical areas if used correctly. A mutex dues that with using its two stages where it is locked and

unlocked, so if a thread or a process want to access a resource it can only do so when it is unlocked. If it is locked it will wait until it get unlocked. There is also a possibility to set a priority list for the thread and the processes [3]. Semaphore is also an option to secure the second race condition. Semaphore is used to control access for multiple processes, to a common resource. This will also help to prevent overwriting [4]. Some other solution to this race condition would be to let the database do the work when it come to manage the different votes.

Bibliography

[1] race condition, http://support.microsoft.com/kb/317723

[2] race condition, https://defuse.ca/race-conditions-in-web-applications.htm

[3] mutex, http://searchnetworking.techtarget.com/definition/mutex

[4] semaphore, http://www.techopedia.com/definition/3875/semaphore)