Gjøvik University College

Software Security

Assignment #2 - Review

Electronic Voting System - Race Conditions

Authors:

Jannis Schaefer - 120914

Reviewed by:

Victor Rudolfsson - 120912

September 26, 2014

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.

Solution

1 Solution description

The solution i choose to inspect for possible race conditions is login to e-voting. The process is as described in table 1, having multiple users on multiple systems to registrate votes and one system to process ballots.

User	Registration of vote	Ballot processing
1. Authenticate		
	2. Verify authentication	
	3. Check whether user has voted already	
	4. Present alternatives of the vote	
5. Vote for desired alternative(s)		
	6. Generate ballot	
	7. Prompt user to affirm vote	
8. Affirm choice(s)		
	9. Send ballot to central processing	
		11. Add vote to polls
	12. Mark user as having voted already	
	Alternate flow:	
	3b. Authentication failed	
	4b. Already voted, reject user	

Table 1: Detailed use case diagram for e-voting

2 Possible race conditions

The first possible race condition in the login process appears when a user logs in twice but to different systems. Since no check is performed (or feasible), the ability to vote multiple times is granted, if the user logs in multiple times before voting. Furthermore might step 3 be performed simultaneously to step 12, making the occurrence of those steps and their order inconsequential.

Another possible race condition lays in the sending and processing of the ballots. Many ballots are send at the same time and ideally one would want as many ballots as possible to be processed in parallel. If two votes for an alternative are counted concurrently, the shared variable might get falsely updated. Furthermore may ballots being written to the same buffer conflict and be lost if multiple are written at the same time. Additionally might a read and removal from the buffer conflict with a write (or another read, in which case a ballot might be multiplied) in a similar manner.

3 Countermeasures

To counter the ability to vote multiple times via multiple simultaneous logins, step 12 should be implemented (attempted) between step 8 and 9, immediately followed by another check to detect whether a user has voted already. But this still doesn't solve the second part of the problem, step 3 and 12 might be executed simultaneously. Here a lock on the variable which stores whether the user has voted solves this problem.

Ballots being written to the shared buffer must be attempted with a lock on the buffer, so that only one system for registration or the system for processing them accesses this buffer at the same time.

Counting votes could be solved by not working on a shared variable, so that each process finishes counting before aggregating the results. This however has the drawback of making live statistics near impossible (without implementing further locks), but live statistics pose a problem since they might enable tactical voting. Should live statistics be desired, a lock on each of the shared poll variables is needed.

1 Review

I found the report to be somewhat jumbled and unstructured - Schaefer begins by presenting the solution and continues to present the problems. This, for me, made it hard to get an overview, especially as countermeasures and solutions were presented as separate things (or maybe I'm just confusing the terms myself?). I believe the first use case diagram isn't actually a solution as the headline states, but a process.

The report goes on to provide two examples of possible race conditions, and they are indeed possible race conditions as far as I can tell, however, they only exist based on the assumption that the ability to vote is determined upon log-in; e.g that the ballot is presented upon logging in based on whether or not the user has previously voted, and no check is performed when the ballot is actually sent. It also seems to assume that the data is stored in a database without constraints, where an existing value is simply incremented and users votes are not inserted as unique rows with the users vote.

I find these assumptions to be valid, as it would be a typical newbie mistake for somebody just learning e.g PHP, and I pity whoever made the decision to assign responsibility of a voting system to that developer.

Appropriate counter-measures are presented, and the presentation is suited for a generic voting system without being too specific, because it's such a typical newbie mistake to make for a developer.

There are, however, no references - but the report makes it point somewhat clear without giving the impression that it must be supported by references since the scenario is rather generic.

I give it 6 out of 10 pandas, or in other words ...

Grade: Passed