Obligatory exercise 2: Electronic ID – Race Conditions

Written by, Ole Eivind Parken, StudentID: 121086

September 18, 2014

1 Goals

In this given assignment we will accomplish these given goals:

- *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 What is a race condition?

Lets say that we have voting system with the number of votes stored in variables. So what we want to happens when a user votes the variable get incremented by one, as is the following:

```
1    totalVotes = i;
2    totalVotes = totalVotes + i;
3    i = totalVotes;
```

So now, lets say two users vote on the same option in an asynchronous enviroment, this could happend:

```
1    i = 5;
2    totalVotes = i (totalVotes == 5); //User1
3    totalVotes = totalVotes + 1 (totalVotes = 6) //User1
4    totalVotes = i (totalVotes == 5); //User2
5    totalVotes = totalVotes + (totalVotes = 6) //User2
6    i = totalVotes; //User1: i = 6;
7    i = totalVotes; //User2: i = 6;
```

The value of "i" was initially 5, and after our two user had voted the value were incremented to 6, when we was expecting the value to be 7. But since the way we are incrementing the value of "i" the incrementation will not be successful. This is a typical race condition, where two or more actions are gone change the same variable/data.

2.1 Sessions

So lets say we have a website that are programmed in PHP and uses sessions to remember that a user have logged in. We log into this website and open two windows of the same page. Based on changes we make on the website will affect our session in both of our

windows. If we make changes in both windows simultaneously and our window A uses more time to load then window B, window A will overwrite the session data.

Our session file is normally not locked for writing, but with a appropriate use of a session handler the file will be locked to prevent race condition. The PHP build in session handler system call is to long to write here but here is a small piece of it:[1]

```
open("/var/lib/php/session/sess_XXXXXXXXXXXXXXXXXXXXXXXXXX",O_RDWR|
    O_CREAT, 0600) = 18
flock(18, LOCK_EX)                          = 0
fcntl64(18, F_SETFD, FD_CLOEXEC)            = 0
fstat64(18, {st_mode=S_IFREG|0600, st_size=11, ...}) = 0
pread64(18, "count|i:17;", 11, 0)           = 11
. . . .
pwrite64(18, "count|i:18;", 11, 0)          = 11
close(18)                                   = 0
```

As we can see PHP uses flock() system call, to prevent that other threads from changing the session data. When using a single web server this will work perfectly, but if you intend to scale up multiple web servers, then you have to store the session data on a other way. Because usually does not have access to the same file system. Normally, you will start to store your session data to a database server, but there is no automatic way to prevent this type of things by default, but you can insert or update data to your database while using transaction.

Transaction is a way to make sure that your data are being handled correct, I will explain more in detail.[2]

```
START TRANSACTION;
SELECT balance FROM Account WHERE Account_Number='100';
SELECT balance FROM Account WHERE Account_Number='101';
UPDATE Account set balance=balance-900 WHERE Account_Number='100';
UPDATE Account set balance=balance+900 WHERE Account_Number='101';
COMMIT;
ROLLBACK;
```

Using this SQL on a SQL database using the InnoDB type, we transfer 900 from Account Number 100 to Account number 101. This code will run as a single statement and will only be saved if everything goes successfully. If not, the database will discard inserted data and go back to the state it was before using this code. If you want do lock a whole table for reading or writing you can use this command:[2]

```
LOCK TABLES account WRITE;
INSERT INTO account VALUES .....
UNLOCK TABLES;
```

References

[1] A. Bakun, "Race Conditions with Ajax and PHP Sessions," http://thwartedefforts.org/2006/11/11/race-conditions-with-ajax-and-php-sessions/, 2006, [Online; accessed 17.09.2014].

[2] Javarevisited, "Database Transaction Tutorial in SQL with Example for Beginners," http://javarevisited.blogspot.no/2011/11/database-transaction-tutorial-example.html, 2011, [Online; accessed 17.09.2014].