

Assignment 2

Joachim Hansen 120483

16.09.14

Disclaimer

There is no sarcasm in this text.

Abstract

With most nations voter turnout is rather lacking, the exceptions would be great nations like that of North Korea, where even the dead are casting their ballots. One college student Mr Stoltenberg residing on the right side of Mjøsa wants to mimic this success by implementing an E-voting system in the kingdom of Norway. Even though this project is a strictly mandatory task given by the school to all students in the same course, Mr Stoltenbergs will not change his mind: that he will be the one to save the democratic process in our great nation. While only scoring a bit above average in his courses and with no relevant job experience, he estimates that creating a E-voting system without any security flaws will only take him a couple of hours. Because how hard can it really be?

Definition off time of check to time of use

A process will often validate that it is working with the right set of resources (check). If a context switch happened now, there would be a window of opportunity between the time of check and when the process will use this resource. In this window the attacker can change the state off this resource, to make the suspended process do his bidding, when it is changed back to running state.[1, page 526-527]

While this window of opportunity is small, and hence this exploit is difficult, it is not impossible. A wise man once said "never say, never" - Justin Bieber (Song never say never)

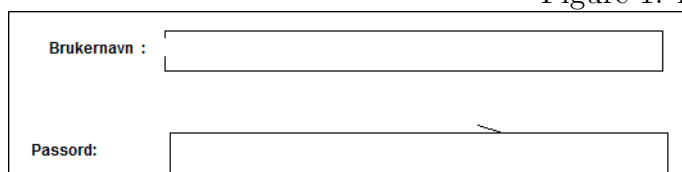
Methodology

The methodology used in this report is to present the reader with an abstract piece of software, written in pseudo code that represent a E-voting system. I will highlight potential race condition in the code and their impact on the security, as well as providing possible solutions to alleviate this threat.

Login screen

Mr Stoltenberg have used his expertise with GUI to create a prototype of the login screen, he believes that this GUI is something that all Norwegians can relate to. The prototype was created in the recognized development tool known as Microsoft paint. You can see this masterpiece in figure 1

Figure 1: login screen



This login screen allows people to vote from the comforts of their own living room. This scheme makes a dispatcher thread force his minion army of slave threads, to satisfy the request coming from inpatient clients all over Norway to our webserver. (kilde) On our made up scenario, the application is running on a server has a Linux OS.

Created equal

"All animals are equal, but some animals are more equal than others" - George Orwell (Animal farm)[2, page 52]. To make the god forsaken people of the north of Norway believe that they still have some political influence, their vote was decided to be counted for twice that of a normal citizen. To account for this most reasonable rule, Mr Stoltenberg decided to make a text file that contains all the cities and their ballot modifier for which this rule apply. Parts of the file is shown in figure 2

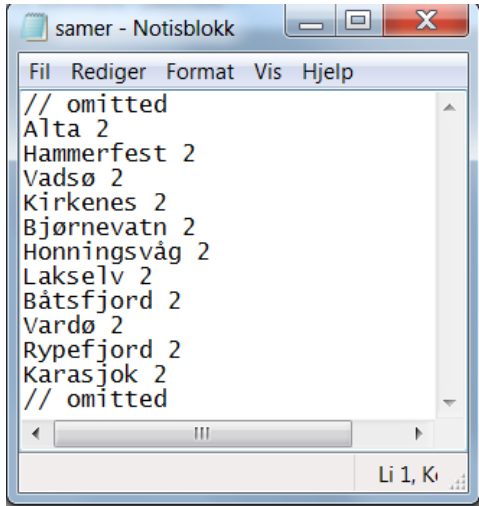


Figure 2:

Written in stone:

Or rather stored in a MySQL Database. This is the technology Mr Stoltenberg uses to hold the relevant data for each vote.

Psauco code of the application

Figure 3: Class Authenticate

```
1 // omitted
2
3 public class Authenticate {
4
5     if( authenticated() )
6         new RegisterVote( new Credential( nationalID ,... ) );
7 }
```

The class Authenticate is shown in figure 3 which has a check in line 5 whether or not the user was able to authenticate, using a similar login screen as that showed in figure 1. If access is granted, then a new instance of RegisterVote will be created with the users credentials.

Continues on the next page

Figure 4: Class: Register Vote

```

1 // omitted
2
3 public class RegisterVote {
4
5     public static MySQLConnection mSQLC = new MySQLConnection(database);
6
7
8
9     public RegisterVote( Credential c ) {
10         bool voteCountForTwice = false;
11         int voteNumber = 0;
12
13         if( !unique( c ) ) throw exception;
14
15         open(/usr/home/myFolder/Samer,OREADONLY)
16         if (lstat(/usr/home/myFolder/Samer) < 0) throw exception;
17
18         if( toString( file ).contains( c.resident ) )
19             VoteCountForTwice = true;
20
21         voteNumber = SelectionBox();
22
23         store( mSQLC, c, VoteNumber );
24
25     }
26 }
27 }
28
29 // omitted

```

In line 5 we have our global database connection. From line 9 starts the constructor of the RegisterVote class. Line 10 has a voteCountForTwice that is of Boolean type, and if true: then we will set the value of the vote accordingly. The voteNumber variable in line 11 will be used to hold the return value of which party the user voted for. In line 13 we have a test to see if the user has voted before in the same elections, and if so then we throw an exception. In lines 14-15 we open the file and test if the file residing on a given file path is a symbolic link, If it's then we throw an exception [1, page 528-529]. In line 18 we check if the user have residency in one of the "special treatment cities" and then set the VoteCountForTwice if he is special. And finally in line 23 we store the ballot in a database using our database connection.

Problems and solutions

Automated attacks

I will introduce this subject by a quote from Gary McGraw and John Viega: "an attacker can automate code that repeatedly tries to exploit the race condition, and just wait for it to succeed. If the odds are one in a million that the attacker will be able to exploit the race condition, then it may not take too long to do so with an automated tool." [3]. In our authenticate class in figure 3 in line 5, the user is prompted to provide his username and password. It is easy for a attacker to write a piece of code that executes in parallel and tries multiple username and password combination, and if successful he will have a far greater chance to exploit the possible race conditions in RegisterVote (as shown in figure 4).

We can try to minimize the probability that an automation tool will successfully be granted access by our system, by introducing a CAPTCHA. In Short a CAPTCHA is a distorted image containing a string, or an audio file containing a secret. Humans are far better than a machine to extract this string/secret, and therefore CAPTCHA is a way to differentiate an automated tool and real users [4]. An example of this solution is shown in figure 5

Figure 5: CAPTCHA example - **Source(s):** [5]



File path and lstat()

In the class RegisterVote (see figure 4) and in line 16, we make sure that we are still working on a regular file and not a symbolic link. The problem here is not that lstat will not tell the truth about the file at the time of check, but that if context switches where to happen now: a attacker can just use a symbolic link and we will be none the wiser[1, page 526,528-529]. By doing so the attacker can decide the outcome of the elections and this outcome is unacceptable for Mr Stoltenberg. A better alternative is depicted in figure 6

Figure 6: Changes to the code in figure 4

```
12  int fd;  
13  if( !unique( c ) ) throw exception;  
14  
15  fd = open( /usr/home/myFolder/Samer, O_RDONLY)  
16  if (fstat(fd) < 0) throw exception;
```

In a whitepaper written by the “Sans institute” it is recommended that we should only use references that do not change (immutable). One reference in a UNIX system that has this property is the file descriptor (fd) as this will be guaranteed to not be changed after its creation. I have replaced the call to lstat() and uses fstat() instead that takes an fd as an argument. This is far safer than using a file path. [6], [1, page 528-533]

Only one vote?

In the class RegisterVote (see figure 4) line 13, we check that we are the only user with this credentials in the system. If the process is preempted before he is able to store his credentials and ballot, multiple threads might get the opportunity to allow one user to vote multiple times. If the system where to allow multiple votes for i.e. ”Sosialistisk Venstreparti” (SV) then that would most likely have no effect on the elections, but multiple votes for any other party might change the outcome of the election.

We have already made automated attacks less of an factor. With adding more test and closer test before use, we would make the window off opportunity smaller. All off these measures will only provide more race conditions, but the attacker needs to win all off these and perhaps the probability of exploit is so small that it is within acceptable risk.[6]

Bibliography

- [1] M. Dowd, J. McDonald, and J. SCHUH, *The art of software security assesment identifying and preventing software vulnerabilities*.
- [2] G. Orwell. (1945). Animal farm, [Online]. Available: http://msxnet.org/orwell/print/animal_farm.pdf.
- [3] G. McGraw and J. Viega. (2001). Building secure software: race conditions, [Online]. Available: <http://www.informit.com/articles/article.aspx?p=23947>.
- [4] A. Fruz. (2014). Limiting automated access, [Online]. Available: <http://resources.infosecinstitute.com/limiting-automated-access/>.
- [5] C. M. University. (2010). Captcha: telling humans and computers apart automatically, [Online]. Available: <http://www.captcha.net/>.
- [6] J. C. Lowery. (2002). A tour of tocttous, [Online]. Available: <http://www.sans.org/reading-room/whitepapers/securecode/tour-tocttous-1049>.