Non-atomic check and use
aka
TOCTOU (Time of Check, Time of Use)
or
Race conditions

Erik Poll
Digital Security group
Radboud University Nijmegen

TU/e Master in Cyber Security

Race condition

- Two concurrent execution threads both execute the statement

  ```
  x = x+1;
  ```

  where x initially has the value 0.
- *What is the value of* `x` *in the end?*

  Answer: `x` can have the value 2 or 1
  - Worse still, in some languages, eg. Java, it can have an arbitrary value

- The root cause of the problem is that `x = x+1` is not an atomic operation, but happens in two steps, reading x and assigning the new value, which may be interleaved in unexpected ways

- Why can this lead to security problems?
- Think of internet banking, and running two simultaneous sessions with the same bank account… *Do try this at home!* ☺

A classic source of (security) problems

- Race condition aka data race is a common type of bug in concurrent programs
  - Basically: two execution threads mess with the same data or object (program variable, file, …) at the same time
  - Not necessarily a *security* bug, but it can be…

- Non-atomic check and use

  aka TOCTOU (Time Of Check, Time of Use)

  is a closely related type of security flaw

  Problem: some precondition required for an action is invalidated between the time it is checked and the time the action is performed
  - Typically, this precondition is access control condition
  - Typically, it involves some concurrency

Classic UNIX race condition

- `lpr -r`

  Print utility with `-r` option to remove file after printing

  Could be used to delete arbitrary files

  How?

  1. user executes `lpr -r symlink`
     where `symlink` is a symbolic link
  2. OS checks that user has permission to read & delete this file
  3. while the file is printing move the link is moved, eg to `/etc/passwd`
  4. after printing `lpr`, which has *root permission*, deletes `/etc/passwd`

  Root of the problem: time between check (2) and use (4)

Learning from past mistakes?

`lpr -r` is a classic security flaw from the 1970s, but similar flaws happen decades later

CVE-2003-1073
A race condition in the `at` command for Solaris 2.6 through 9 allows local users to delete arbitrary files via the -r argument with .. sequences in the job name, then modifying the directory structure after at checks permissions to delete the file and before the deletion actually takes place

Combination of race condition with failure to check that file names do not contain ..

Another classic: `mkdir` on Unix

- `mkdir` creates a new directory/folder
- this program is `setuid root`, ie. executes as root

- It creates new directory *non-atomically*, in several steps:
  1. enter super-user mode
  2. creates the directory, with owner is root
  3. sets the owner, to whoever invoked `mkdir`
  4. exit super-user mode

- Attack: by creating a symbolic link between steps 2 and 3, attacker can own any file

Example race condition

```
const char *filename="/tmp/erik";
if (access(filename, R_OK)!=0) {
    ... // handle error and exit;
}
// file exists and we have access
int fd open (filename, O_RDONLY);
...
```

Between calls to `access` and `open` the file might be removed, or a symbolic link in the path might be reset!

Race condition & file systems

Signs of trouble:

- Access to files using filenames rather than file handles or file descriptors
  - filenames may point to different files at different moments in time

- Creating files or directories in publicly accessible places, for instance `/tmp`
  - especially if these have predictable file names

Spot the race condition!

```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                  throws ServletException, IOException {
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
          query = "INSERT INTO roles" + "(userId, userRole)" +    "VALUES "  + "('"  +
                  request.getParameter("userId") + "'," +
                  "'standard')";
          Statement stmt = conn.createStatement();
          stmt.executeUpdate(query);
          } catch ...
  }
```

Spot the race condition!

```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                throws ServletException, IOException {
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
          query = "INSERT INTO roles" + "(userId, userRole)" +   "VALUES "  + "('"  +
                  request.getParameter("userId") + "'," +
                  "'standard')";
          Statement stmt = conn.createStatement();
          stmt.executeUpdate(query);
          } catch ...
  }
```

Concurrent calls of doGet will be on the *same* HttpServlet object and hence use the *same* instance field `query`

How could you know this?

Spot the race condition!

```
public class SimpleServlet extends HttpServlet {
  private String query;
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                throws ServletException, IOException {
   String query;
    try { Connection conn =
            DriverManager.getConnection("jdbc:odbc ... ");
        query = "INSERT INTO roles" + "(userId, userRole)" +   "VALUES "  + "('"  +
                request.getParameter("userId") + "'," +
                "'standard')";
       Statement stmt = conn.createStatement();
       stmt.executeUpdate(query);
       } catch ...
  }
```

Fix: now every (possibly concurrent) call of doGet has its own `query` field

MIDP Java feature phone security bug

Malicious game on Siemens S55 feature phone

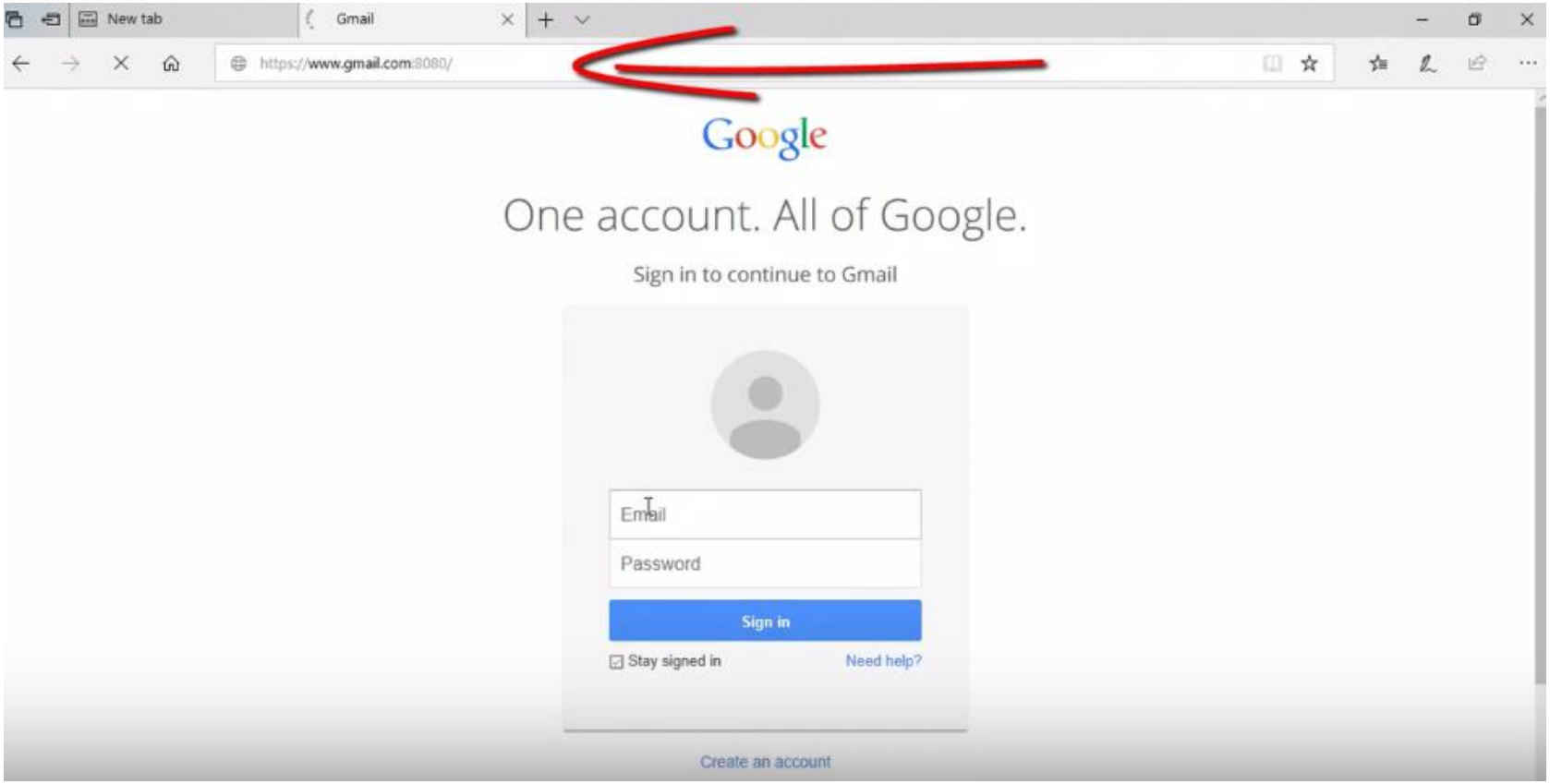
exploited race condition in GUI

to let user unwittingly authorise an SMS

Do you want to play the game?

If user presses ok he agrees to the underlying pop-up

OK to send SMS to 6492?


New tab
Gmail
https://www.gmail.com:8080/
Google
One account. All of Google.
Sign in to continue to Gmail
Email
Password
Sign in
Stay signed in
Need help?
Create an account

Edge & Safari GUI bug [CVE-2018-8383]

Security

Safari, Edge fans: Is that really the website you think you're visiting? URL spoof bug blabbed

Egghead says Apple has yet to patch spoofing vulnerability

By Shaun Nichols in San Francisco 11 Sep 2018 at 05:01 13 SHARE ▼

URL in address bar can be spoofed with a race condition

- **Script loads legitimate page, changing address bar, but over non-existent port, and then quickly loads another page**

https://www.theregister.co.uk/2018/09/11/safari_edge_spoofing/

https://youtu.be/Ni2XzF5-ixY

https://youtu.be/dGJSsK55nfQ

Promising future for data races?

- Trend: more multi-CPU machines
  - to keep improving computer power in accordance with Moore's Law, despite physical contraints
- Hence: more multi-threaded software
  - to take advantage of max. available computing power
- Hence: more problems with data races in code
  - as programmers cannot cope with concurrency
  - writing correct concurrent programs is hard

[Interesting article to read: NOT EXAM MATERIAL

"The free lunch is over :a fundamental turn toward concurrency in software", by Herb Sutter]