Race Conditions

TDT60
20.09.2005
Why race conditions?

- Concurrency issue

- May occur in multithreaded/multiprocess environments
  - Where threads/processes occurring at once may interact
  - Also: multiple processes on a single machine can have race conditions when they operate on data that may be shared

- A major problem because:
  - A program that seems to work fine may still contain race conditions
  - Hard to detect – especially if you’re not looking for them
  - Hard to fix – even when you are aware of their existence

- A seemingly deterministic program can behave in a seriously nondeterministic way
The elevator example
Race condition

- Occurs when an assumption needs to hold true for a period of time, but actually may not.
  - Whether it does is a matter of exact timing.
- In every race condition there is a *window of vulnerability*.
- *Window of vulnerability* is often very small – meaning that bugs may show up only very infrequently – if ever.
A Java example

```java
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;
public class Counter extends HttpServlet{
    int count = 0;
    public void doGet(HttpServletRequest in, HttpServletResponse out) throws ServletException, IOException{
        out.setContentType("text/plain");
        PrintWriter p = out.getWriter();
        count++;
        p.println(count + " hits so far!");
    }
}
```
Exploiting race conditions

• Testing for race conditions and/or reproducing the problem may be almost impossible.
• An attacker can increase his odds by slowing down the machine
• Race conditions with security implications generally only need to be exploited once
  – An attacker may automate code that repeatedly tries to exploit a race condition
Why a security issue?

• File access

• Time of check – time of use (TOCTOU)
  – A check precedes an action
  – The result of the check needs to be valid at the time of action
  – However, in the time between operations the result of the check may change

• Example: passwd
The fix

• Reduce the window of vulnerability to zero
  – Make sure all assumptions hold for as long as they need to hold
  – Atomic code –
    • Critical section executes as one single unit
    • Should be kept as small as possible – to avoid efficiency issues

• File locking