Race Conditions

Secure Programming
Lecture 15

Announcement

- Friday 7, Exploitation workshop with MWR Labs
  - 10am – noon: Learning Centre, UG06
  - 1pm – 5pm: Learning Centre, UG05
  - Free food :-)
- Monday 10 March **no** lecture

Overview

- Parallel execution of tasks
  - multi-process or multi-threaded environment
  - tasks can interact with each other
- Interaction
  - shared memory (or address space)
  - file system
  - signals
- Results of tasks depends on relative timing of events

→Non-deterministic behavior

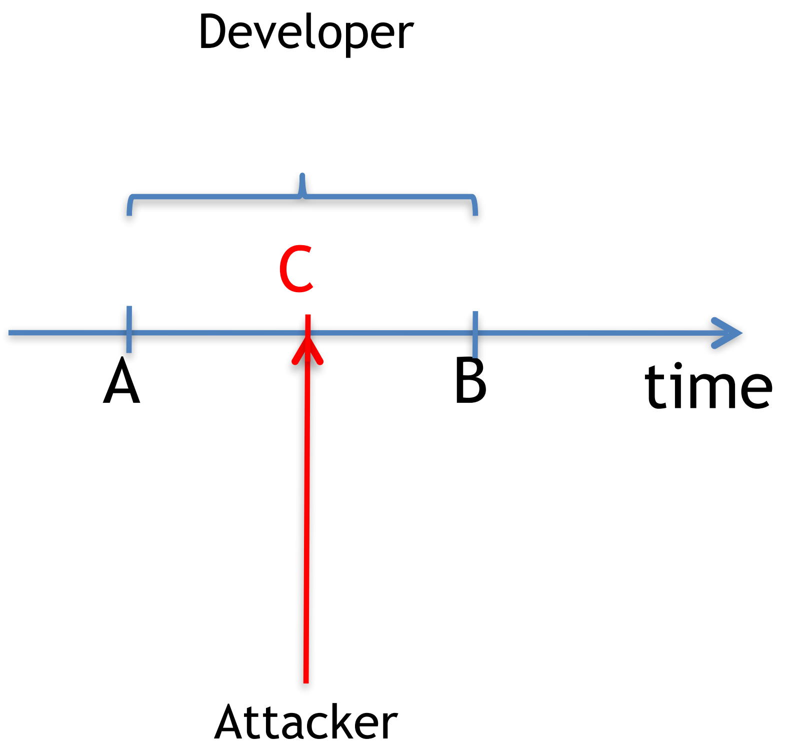
Race conditions

- Race conditions
  - alternative term for non-deterministic behavior
  - often a robustness issue
- Also many important security implications
  - Assumption needs to hold for some time for correct behavior, but assumption can be violated
  - Time window when assumption can be violated

  →window of vulnerability

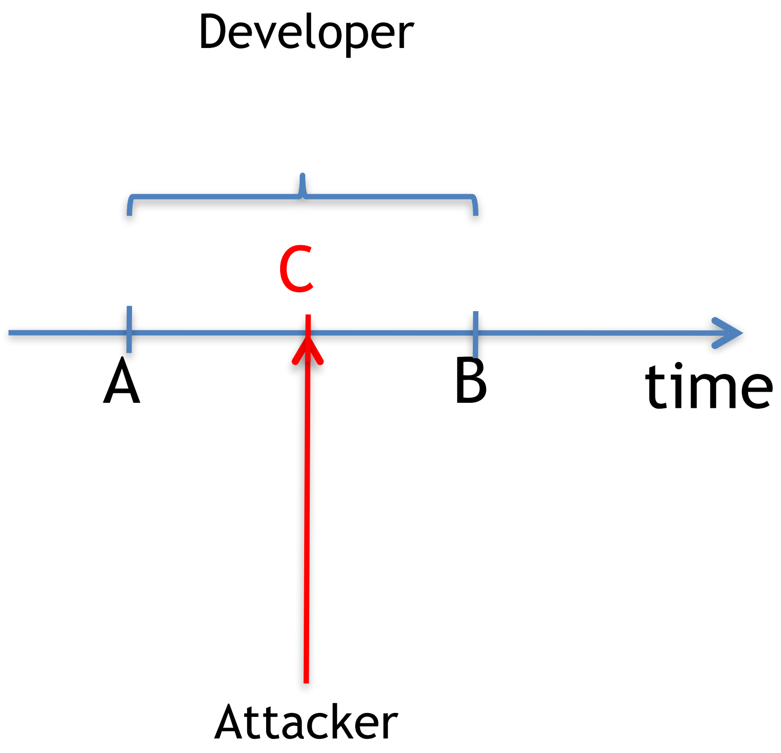
Race conditions

- Programmer views a sequence of operation as atomic
- In reality, atomicity is not enforced
- Attacker can take advantage of this discrepancy



Race conditions

- Scheduler can interrupt process at any time
  - Can happen between A and B
  - Can be made more likely to happened between A and B by an attacker



ATM withdrawal

```
void widthdraw(int amount) {
  balance = this->getBalance();

  if (amount > balance)
    throw new Overdraft(“Not enough funds!”);
  dispense(amount);

}
```

How can we fix this?

Race conditions

- Window of vulnerability can be very short
  - race condition problems are difficult to find with regular testing and difficult to reproduce, but…
  - attacker can slow down victim process/machine to extend window and can often launch many attempts
- Some myths about race conditions
  - “Races are hard to exploit”
  - “Only 1 chance in 10,000 that attack succeeds”
- Attackers can often find ways to beat the odds

Beating the odds

- Can the attacker try exploit 1 million times?
  - If so and odds are 1 in 10,000; then attackers has a reliable exploit
- Can the attacker slow down the victim machine or process?
  - E.g., high load/computational complexity attacks
  - This may extend the window of opportunity!

Time of Check to Time of Use (TOCTOU)

- Common race condition problem
- Problem
  - Time of check (t_A): validity of assumption X on entity E is checked
  - Time of use (t_B): E is used, *assuming that X is still valid*
  - Time of attack (t_C): attacker invalidates X
  - $t_A < t_C < t_B$
- Of course, attacker targets a process executing with higher privileges

Pattern of vulnerability

- Steps to access a resource
  1. obtain reference to resource
  2. query resource to obtain characteristics
  3. analyze query results
  4. if resource is fit, access it
- Often occurs in Unix file system accesses
  – check permissions for a certain file name, e.g., using **`access`**
  – open the file, using the file name, e.g., using **`fopen`**

Access/open race

- Case study: setuid program

  man access: “The check is done using the calling process's real UID and GID, rather than the effective IDs as is done when actually attempting an operation (e.g., open(2)) on the file. This allows set-user-ID programs to easily determine the invoking user's authority.”

```
/* access returns 0 on success */
if(!access(file, W_OK)) {
  f = fopen(file, "wb+");
  write_to_file(f);
} else {
  fprintf(stderr, "Permission denied when trying to open %s.\n", file);
}
```

Note: access dereferences symbolic links

- Attack

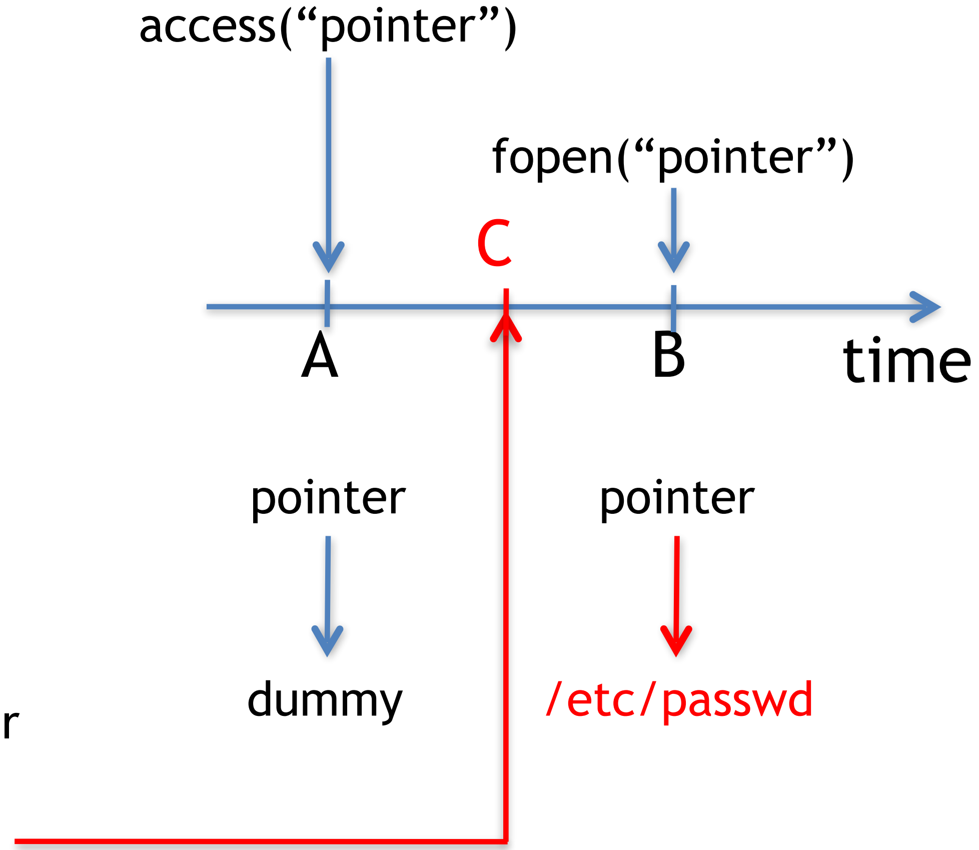
```
$ touch dummy; ln –s dummy pointer
$ rm pointer; ln –s /etc/passwd pointer
```

Access/open race

```
/* access returns 0
   on success */
if(!access(file, W_OK))
{
  f = fopen(file, "wb+");
   write_to_file(f);
} else {
   …
}
```

```
$ touch dummy; ln -s dummy pointer

$ rm pointer; ln –s /etc/passwd
pointer
```



MORE TOCTOU EXAMPLES

Script execve race

- Filename Redirection
  - Soft links again
- Setuid Scripts
  1. exec() system call invokes seteuid() call prior to executing program
  2. program is a script, so command interpreter is loaded first
  3. program interpreter (with root privileges) is invoked on script name
  4. attacker can replace script content between steps 2 and 3
- Setuid non allowed on scripts on most platforms

Directory operations

```
rm –r race
```

- `rm` can remove directories recursively, traversing the directory tree depth-first
- It issues chdir(“..”) to go up one level after removing a branch
- By relocating the subdirectory to another location in the filesystem, arbitrary files can be deleted

Races on temporary files

- Similar issue with temporary files
  - Typically created in /tmp
  - No special privileges needed to create files in there
  - What if victim application uses guessable file names?
- Attack
  - Application writes into /tmp/tmp0001
  - `ln -s /etc/target /tmp/tmp0001`
  - Vulnerable program creates /etc/target for attacker…

Secure temp file creation

- Pick hard to guess filename (random)
- Set umask appropriately (e.g., 0066)
  - Group, others cannot read or write
- Atomically test for existence and create file
  - open with `O_CREAT|O_EXCL` flag
  - If file exists, open will fail; then try again with different filename
- Delete file immediately with `unlink`
- Read/write as needed
- Close the file
  - It will be automatically deleted

Secure temp file creation

- Not exactly trivial, right?
- Don’t roll your own
  - Rely on (well-tested) libraries
- Libraries may get it wrong
  - `mktemp` is not secure
  - Use `mkstemp`

Fixing filesystem races

- Don’t operate on file names
  - Mapping between names and underlying objects may change
  - E.g., access and open
- We need “immutable bindings”
  - Binding between object identifier and referenced object that cannot be changed
  - All operations performed using this binding
- Open a file and operate on its file descriptor
  - chown vs fchown
  - chmod vs fchmod

Not just filesystem operations

- SQL: select before insert (bad)
  - SELECT to check if a given element already exists
  - When SELECT returns no element, insert a (unique) element
- Race condition
  - Two processes may do this check at the same time, leading to two insertions
- Solution:
  - Locking
  - Let the database guarantee the unique constraint (PRIMARY KEYs and UNIQUE constraints)

Signal handlers

- Signals
  - used for asynchronous communication between processes
  - signal handler can be called in response to multiple signals
  - signal handler must be written re-entrant or must block other signals
- Example (CVE-2006-0058)
  - sendmail up to 8.11.3 and 8.12.0.Beta7
  - syslog(3) is called inside the signal handler
  - race condition can cause heap corruption because of double free vulnerability

ptrace/setuid execve vulnerability

- ptrace
  - Debugging facility
  - Used to access other process’ registers and memory address space
  - Can only attach to processes of same UID, except when run by root
- execve
  - execute program image
  - setuid functionality (modifying the process EUID)
  - not invoked when process is marked as being traced

ptrace/setuid execve vulnerability

- ptrace/execve race (CVE-2001-0317)
- Execve operations:
  1. Check whether the process is being traced
  2. Open image (may block)
  3. Allocate memory (may block)
  4. Set process EUID according to setuid flags
- Window of opportunity between 1 and 4
  - Attacker can attach via ptrace
  - Blocking kernel operations (step 2 and 3) may allow other processes to run

Windows DCOM/RPC

- Windows DCOM/RPC vulnerability (CVE-2003-0813)
  - RPCSS service
  - multiple threads process single packet
- Race:
  - one thread frees memory
  - while other process still works on it
  - can result in memory corruption
  - and thus denial of service

BEATING THE ODDS

Beating the odds

- Window of vulnerability can be very short
- BUT Attackers can try to make it longer
  - Make program run slower
- Example: filename lookups
  - Deeply nested directory structure
  - Chain of symbolic links
- Demo courtesy of Paolo Milani, TU Vienna, who implemented attack of "Fixing Races for Fun and Profit: How to abuse atime"

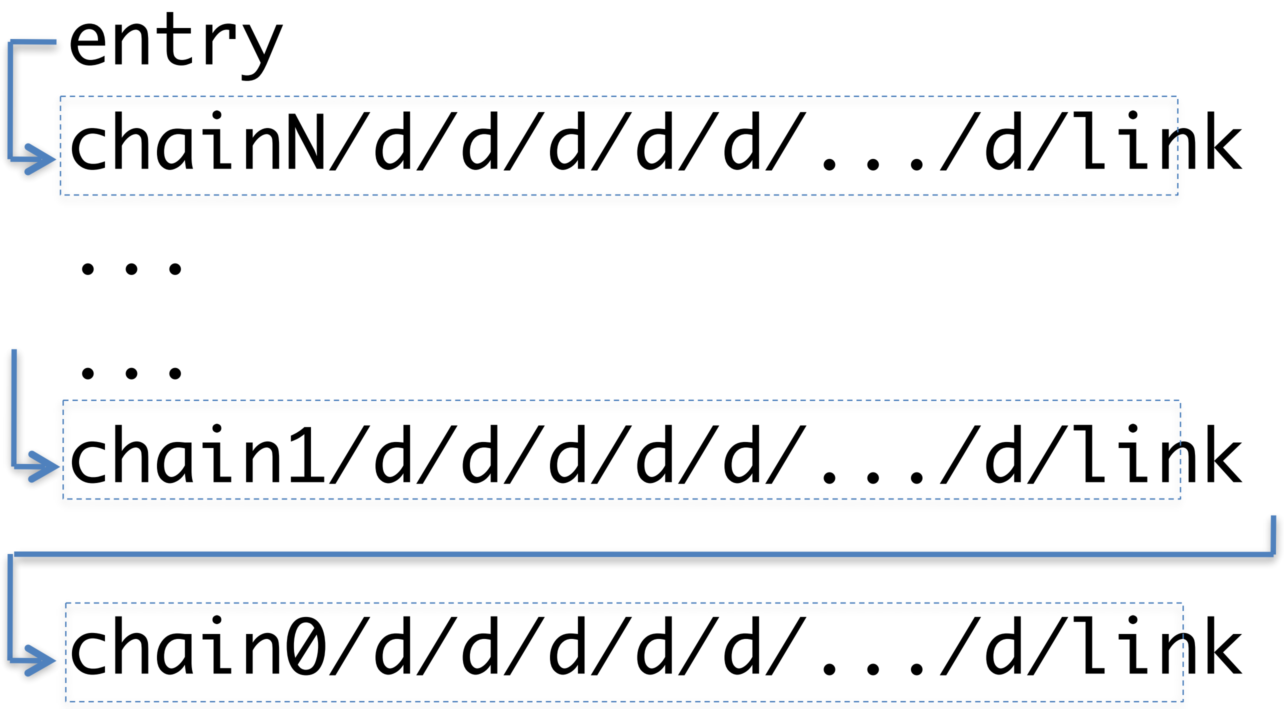
Slow file lookups

- Deeply nested directory structure `d/d/d/d/d/d/d/……/d/file.txt`
- To resolve this file name, the OS must
  - Lookup the directory d in the current directory
  - Lookup the directory d in that directory
  - …
- OS limits the length of a file name
  - MAXPATHLENGTH (4096 on my test Linux box)
  - Max depth ~2000

Filesystem maze

- Combine deeply nested directory structure with chain of symbolic links
  - Workaround filename length restrictions by using links
  - Notice that OS limits the length of a link chain
    - 40 on my test Linux
- Total filesystem lookups
  - 40 chains
  - Each with 2000 levels
  - 80,000 lookups

Filesystem maze



```
entry
chainN/d/d/d/d/d/.../d/link
...
...
chain1/d/d/d/d/d/.../d/link
chain0/d/d/d/d/d/.../d/link
```

`entry/link/.../link/link`: malicious link that forces lookup of entire chain

Mini demo

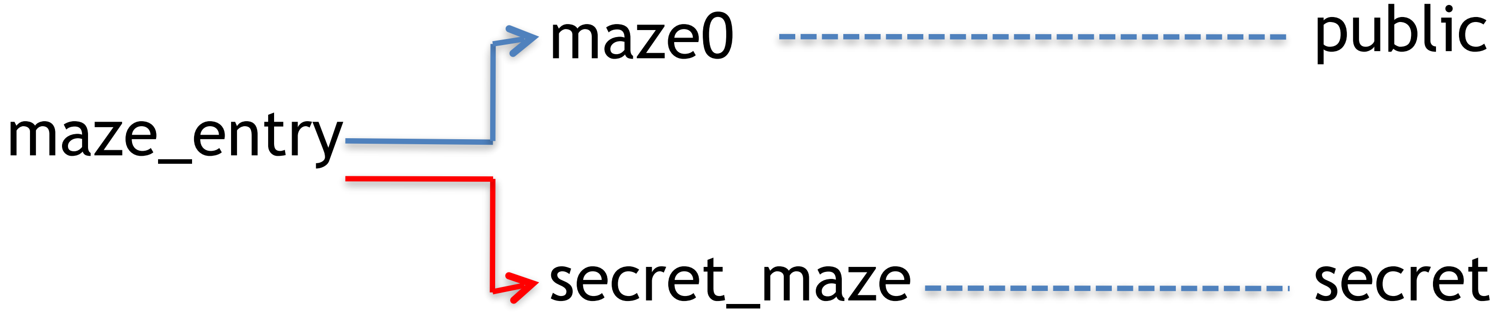
```
$ time cat entry/lnk/
entry/lnk/lnk/lnk/lnk/
lnk/lnk/lnk/lnk/lnk/lnk/
lnk/lnk/lnk/lnk/lnk/lnk/
lnk/lnk/lnk/lnk/lnk/lnk/
lnk/lnk/lnk/lnk/lnk/lnk/
lnk/lnk/lnk/lnk/lnk/lnk/
lnk
CONTENT

real  0m9.269s
user  0m0.000s
Sys   0m4.428s
```

- Worst-case results: even several minutes
  - When testing with the disk busy with other activity
- In all cases, reliably over half a second
- More than enough time to exploit race condition

Access/open race demo

- suid_cat
  - Vulnerable program: setuid version of cat
  - Uses access to check if it can open a file
- Multiple chains of symbolic links



Change maze_entry as suid_cat is running!

From the research dept

- M. Bishop and M. Dilger, Checking for Race Conditions in File Accesses, Computing systems, 1996
  - Statically detects TOCTOU binding flaws
- S. Savage et al., Eraser: A Dynamic Data Race Detector for Multithreaded Programs, Trans. Computer Systems, 1997
  - Dynamically check that access to shared variables are protected by lock
- C. Flanagan and S. Freund, Type-Based Race Detection for Java, PLDI 2000
  - rccjava: type-based tool for finding common synchronization patterns
- D. Engler and K. Ashcraft, RacerX: Effective, Static Detection of Race Conditions and Deadlocks, SOSP 2003
  - Statically compute locksets in Linux, FreeBSD; find bugs in all
- N. Borisov et al., Fixing Races for Fun and Profit: How to abuse atime, USENIX Security 2005
  - One year earlier, others proposed a probabilistic fix for access/open race (k-Race: forces attacker to win k races)
  - Breaks this proposed fix

Take away points

- Race conditions arise from lack of atomicity in sequence of security-sensitive operations
- Time of check to time of use (TOCTOU)
  - Not just for filesystem operations
- Even though time window is small, attackers are often able to make it artificially larger
- Defenses
  - Locking
  - Check at the same time as use