Race Condition

Mustakimur R. Khandaker

Concurrency & Race Condition

Concurrency:

- Execution of Multiple flows (threads, processes, tasks, etc).
- If not controlled can lead to non-deterministic behavior.

Race Conditions:

- Software defect/vulnerability resulting from unanticipated execution ordering of concurrent flows.
  - e.g., two people simultaneously try to modify the same account (withdrawing money).

Concurrency Humor

Knock knock.

– "Race condition."

– "Who's there?"

Properties

Concurrency property.

- At least two control flows executing concurrently.

Shared object property.

- The concurrent flows must access a common shared race object.

Change state property.

- At least one control flow must alter the state of the race object.

Race Window

A code segment that accesses the race object in a way that opens a window of opportunity for race condition.

- Also referred as ***critical section***.

Traditional approach to avoid race condition:

- Ensure race windows do not overlap.
  - Make them **mutually exclusive**.
  - Language facilities - *synchronization primitives (SP)*.
    - Semaphores, mutex, locks, etc.
- *Deadlock* is a risk related to SP.
  - Denial of service.

Semaphore

Semaphores are data structure that provides mutual exclusion to critical sections.

- Block waiters, interrupts enabled within CS.
- Described by Dijkstra in THE system in 1968.

Semaphores support two operations:

- **wait(semaphore):** decrement, block until semaphore is open.
  - Also P(), after the Dutch word for test, or down().
- **signal(semaphore):** increment, allow another thread to enter.
  - Also V() after the Dutch word for increment, or up().

Continue …

Associated with each semaphore is a queue of waiting processes.

When wait() is called by a thread:

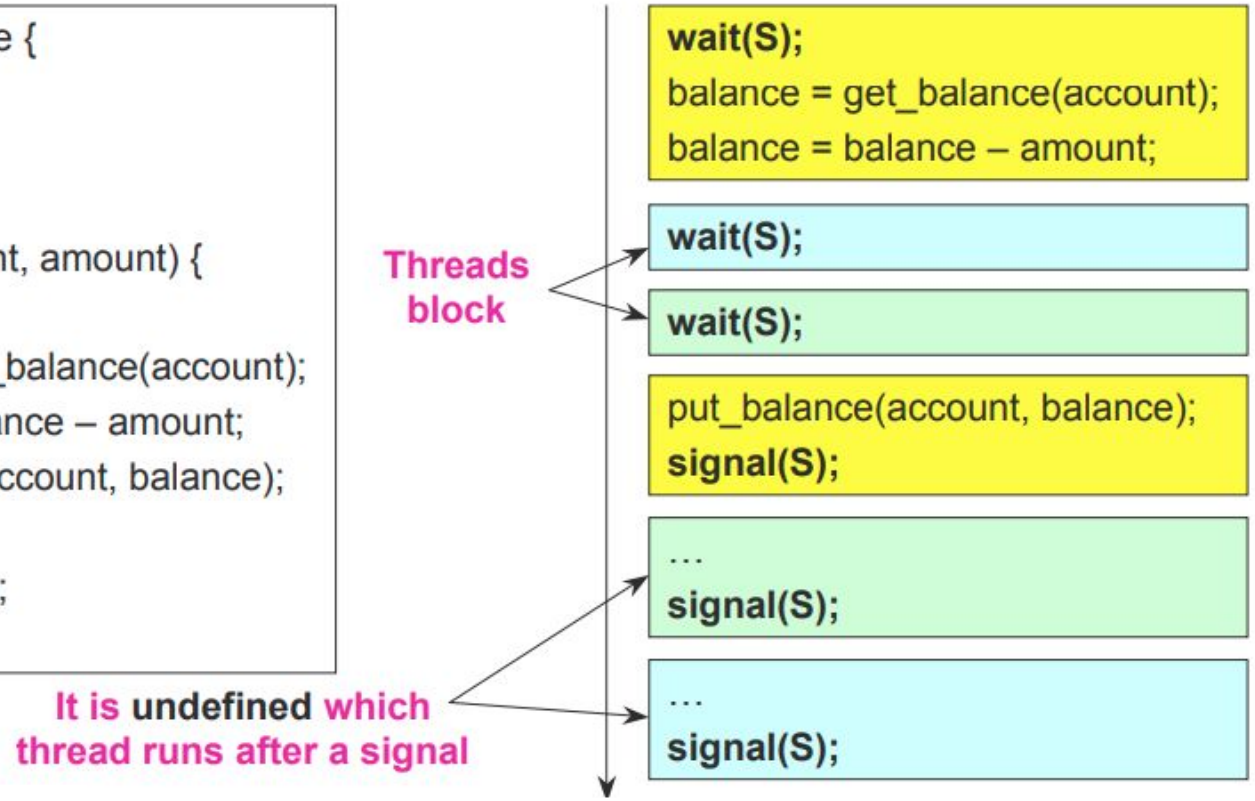
- If semaphore is open, thread continues.
- If semaphore is closed, thread blocks on queue.

Then signal() opens the semaphore:

- If a thread is waiting on the queue, the thread is unblocked.
- If no threads are waiting on the queue, the signal is remembered for the next thread.
    - In other words, signal() has “history” (c.f. condition vars later).
    - This “history” is a counter.

Continue …

```
struct Semaphore {
    int value;
    Queue q;
} S;
withdraw (account, amount) {
    wait(S);
    balance = get_balance(account);
    balance = balance – amount;
    put_balance(account, balance);
    signal(S);
    return balance;
}
```



Deadlock

A deadlock is the situation where a group of threads wait forever because each of them is waiting for resources that are held by another thread in the group (circular waiting).

```
Semaphore s=1, q=1

process p0 {
   s.acquire();
   q.acquire();
   ...
   s.release();
   q.release();
}

process p1 {
   q.acquire();
   s.acquire();
   ...
   q.release();
   s.release();
}
```

```
Semaphore s=1, q=1;

process p0 {
   // order matters a great deal on the waits
   q.acquire();
   s.acquire();
   ...
   // order does not matter that much on the signals
   s.release();
   q.release();
}

process p1 {
   // order matters a great deal on the waits
   q.acquire();
   s.acquire();
   ...
   // order does not matter that much on the signals
   q.release();
   s.release();
}
```

Time of Check, Time of Use (ToCToU)

Source of race conditions:

- Trusted (tightly coupled threads of execution) or untrusted control flows (separate application of process).

ToCToU race conditions:

- Can occur during file I/O.
- Forms a race window by first checking some race object and then using it.

```
#include <stdio.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
  FILE *fd;
  if (access("/some_file", W_OK) == 0) {
    printf("access granted.\n");
    fd = fopen("/some_file", "wb+");
    /* write to the file */
    fclose(fd);
  }      . . .
  return 0;
}
```

The `access()` function is called to check if the file exists and has write permission.

Race Window

File opened for writing

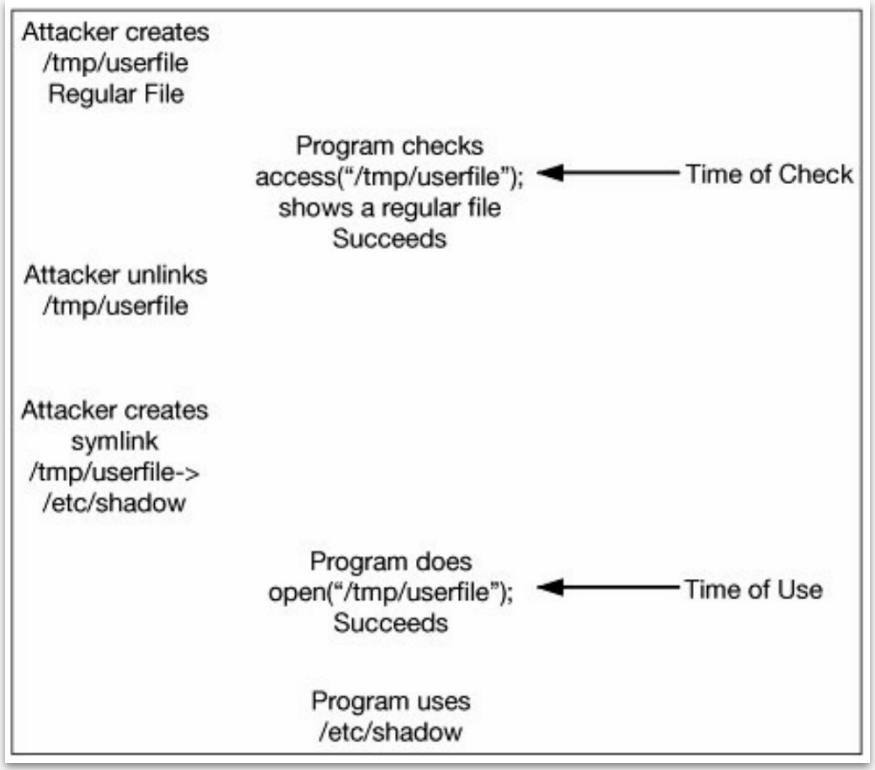
Continue …

Unix runs multiple processes at once.
- Attacker runs a process alongside suid program.
- Must attack at exactly right moment.

Processes are scheduled by the OS.
- maybe on multiple CPUs.

Attacker may be able to influence scheduling.
- slow down system, send job control signals.

Attacker may be able to automatically schedule attack.
- e.g. Linux inotify API for monitoring file system.

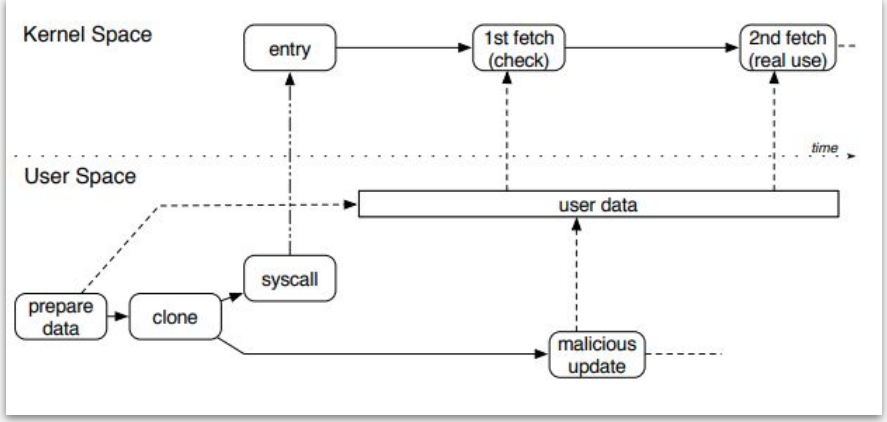


CVE Reports

Name	Description
CVE-2020-3957	VMware Fusion (11.x before 11.5.5), VMware Remote Console for Mac (11.x and prior) and VMware Horizon Client for Mac (5.x and prior) contain a local privilege escalation vulnerability due to a Time-of-check Time-of-use (TOCTOU) issue in the service opener. Successful exploitation of this issue may allow attackers with normal user privileges to escalate their privileges to root on the system where Fusion, VMRC and Horizon Client are installed.
CVE-2020-3808	Creative Cloud Desktop Application versions 5.0 and earlier have a time-of-check to time-of-use (toctou) race condition vulnerability. Successful exploitation could lead to arbitrary file deletion.
CVE-2020-15702	TOCTOU Race Condition vulnerability in apport allows a local attacker to escalate privileges and execute arbitrary code. An attacker may exit the crashed process and exploit PID recycling to spawn a root process with the same PID as the crashed process, which can then be used to escalate privileges. Fixed in 2.20.1-0ubuntu2.24, 2.20.9 versions prior to 2.20.9-0ubuntu7.16 and 2.20.11 versions prior to 2.20.11-0ubuntu27.6. Was ZDI-CAN-11234.
CVE-2020-13882	CISOfy Lynis before 3.0.0 has Incorrect Access Control because of a TOCTOU race condition. The routine to check the log and report file permissions was not working as intended and could be bypassed locally. Because of the race, an unprivileged attacker can set up a log and report file, and control that up to the point where the specific routine is doing its check. After that, the file can be removed, recreated, and used for additional attacks.
CVE-2020-13162	A time-of-check time-of-use vulnerability in PulseSecureService.exe in Pulse Secure Client versions prior to 9.1.6 down to 5.3 R70 for Windows (which runs as NT AUTHORITY/SYSTEM) allows unprivileged users to run a Microsoft Installer executable with elevated privileges.
CVE-2019-7347	A Time-of-check Time-of-use (TOCTOU) Race Condition exists in ZoneMinder through 1.32.3 as a session remains active for an authenticated user even after deletion from the users table. This allows a nonexistent user to access and modify records (add/delete Monitors, Users, etc.).
CVE-2019-5519	VMware ESXi (6.7 before ESXi670-201903001, 6.5 before ESXi650-201903001, 6.0 before ESXi600-201903001), Workstation (15.x before 15.0.4, 14.x before 14.1.7), Fusion (11.x before 11.0.3, 10.x before 10.1.6) contain a Time-of-check Time-of-use (TOCTOU) vulnerability in the virtual USB 1.1 UHCI (Universal Host Controller Interface). Exploitation of this issue requires an attacker to have access to a virtual machine with a virtual USB controller present. This issue may allow a guest to execute code on the host.
CVE-2019-20000	The malware scan function in BullGuard Premium Protection 20.0.371.8 has a TOCTOU issue that enables a symbolic link attack, allowing privileged files to be deleted.
CVE-2019-18644	The malware scan function in Total Defense Anti-virus 11.5.2.28 is vulnerable to a TOCTOU bug; consequently, symbolic link attacks allow privileged files to be deleted.
CVE-2019-1732	A vulnerability in the Remote Package Manager (RPM) subsystem of Cisco NX-OS Software could allow an authenticated, local attacker with administrator credentials to leverage a time-of-check, time-of-use (TOCTOU) race condition to corrupt local variables, which could lead to arbitrary command injection. The vulnerability is due to the lack of a proper locking mechanism on critical variables that need to stay static until used. An attacker could exploit this vulnerability by authenticating to an affected device and issuing a set of RPM-related CLI commands. A successful exploit could allow the attacker to perform arbitrary command injection. The attacker would need administrator credentials for the targeted device.
CVE-2019-15608	The package integrity validation in yarn < 1.19.0 contains a TOCTOU vulnerability where the hash is computed before writing a package to cache. It's not computed again when reading from the cache. This may lead to a cache pollution attack.
CVE-2019-15316	Valve Steam Client for Windows through 2019-08-20 has weak folder permissions, leading to privilege escalation (to NT AUTHORITY\SYSTEM) via crafted use of CreateMountPoint.exe and SetOpLock.exe to leverage a TOCTOU race condition.
CVE-2018-6693	An unprivileged user can delete arbitrary files on a Linux system running ENSLTP 10.5.1, 10.5.0, and 10.2.3 Hotfix 1246778 and earlier. By exploiting a time of check to time of use (TOCTOU) race condition during a specific scanning sequence, the unprivileged user is able to perform a privilege escalation to delete arbitrary files.
CVE-2018-3759	private_address_check ruby gem before 0.5.0 is vulnerable to a time-of-check time-of-use (TOCTOU) race condition due to the address the socket uses not being checked. DNS entries with a TTL of 0 can trigger this case where the initial resolution is a public address but the subsequent resolution is a private address.
CVE-2018-12691	Time-of-check to time-of-use (TOCTOU) race condition in org.onosproject.acl (aka the access control application) in ONOS v1.13 and earlier allows attackers to bypass network access control via data plane packet injection.
CVE-2017-6296	NVIDIA TrustZone Software contains a TOCTOU issue in the DRM application which may lead to the denial of service or possible escalation of privileges. This issue is rated as moderate.
CVE-2017-2619	Samba before versions 4.6.1, 4.5.7 and 4.4.11 are vulnerable to a malicious client using a symlink race to allow access to areas of the server file system not exported under the share definition.
CVE-2017-18869	A TOCTOU issue in the chownr package before 1.1.0 for Node.js 10.10 could allow a local attacker to trick it into descending into unintended directories via symlink attacks.
CVE-2017-12410	It is possible to exploit a Time of Check & Time of Use (TOCTOU) vulnerability by winning a race condition when Kaseya Virtual System Administrator agent 9.3.0.11 and earlier tries to execute its binaries from working and/or temporary folders. Successful exploitation results in the execution of arbitrary programs with "NT AUTHORITY\SYSTEM" privileges.

Double-fetch Bugs

Precise and Scalable Detection of Double-Fetch Bugs in OS Kernels



Common Scenarios:

- ❑ Dependency lookup.
- ❑ Protocol/signature checking.
- ❑ Information guessing.

```
1  void mptctl_simplified(unsigned long arg) {
2    mpt_ioctl_header khdr, __user *uhdr = (void __user *) arg;
3    MPT_ADAPTER *iocp = NULL;
4
5    // first fetch
6    if (copy_from_user(&khdr, uhdr, sizeof(khdr)))
7      return -EFAULT;
8
9    // dependency lookup
10   if (mpt_verify_adapter(khdr.iocnum, &iocp) < 0 || iocp == NULL)
11     return -EFAULT;
12
13   // dependency usage
14   mutex_lock(&iocp->ioctl_cmds.mutex);
15   struct mpt_fw_xfer kfwdl, __user *ufwdl = (void __user *) arg;
16
17   // second fetch
18   if (copy_from_user(&kfwdl, ufwdl, sizeof(struct mpt_fw_xfer)))
19     return -EFAULT;
20
21   // BUG: kfwdl.iocnum might not equal to khdr.iocnum
22   mptctl_do_fw_download(kfwdl.iocnum, ......);
23   mutex_unlock(&iocp->ioctl_cmds.mutex);
24 }
```

Fig. 1: A dependency lookup *double-fetch bug*, adapted from `__mptctl_ioctl` in file `drivers/message/fusion/mptctl.c`

Data Race

A data race is a race condition at the level of atomic memory accesses.

It is the root cause of many subtle programming errors involving multi-threaded programs (also known as *synchronization bugs*).

data race

```
// Shared variable
var count = 0

func incrementCount() {
  if count == 0 {
    count ++
  }
}

func main() {
  // Spawn two "threads"
  go incrementCount()
  go incrementCount()
}
```

!data race

Thread 1	Thread 2
lock(l)	lock(l)
count=1	count=2
unlock(l)	unlock(l)

Mitigation

- Eliminating the race object.
- Checking file properties securely.
- Mutual exclusion.
  - Implement mutually exclusive critical sections with mutex/semaphores.
  - **Avoid sharing objects between signal handler and other program code.**
- Thread safe function.
  - In multithreaded applications, it is not enough to ensure code is RC free.
  - **If non-thread safe function is called, treat it as a critical section.**
- Use of atomic operations.
  - Atomicity implemented by synchronization functions.
- Controlling access to the race object.

Detection

Race condition detection is NP complete.

- Hence approximate detection.
- C/C++ are difficult to analyze statically.
  - Time variant is impossible to input.
- Dynamic analysis.
  - Random time variants.
  - Fails to consider execution path not taken.
  - Runtime overhead.

< Race Condition />