Race and Spray

Mustakimur R. Khandaker

Race Condition

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.

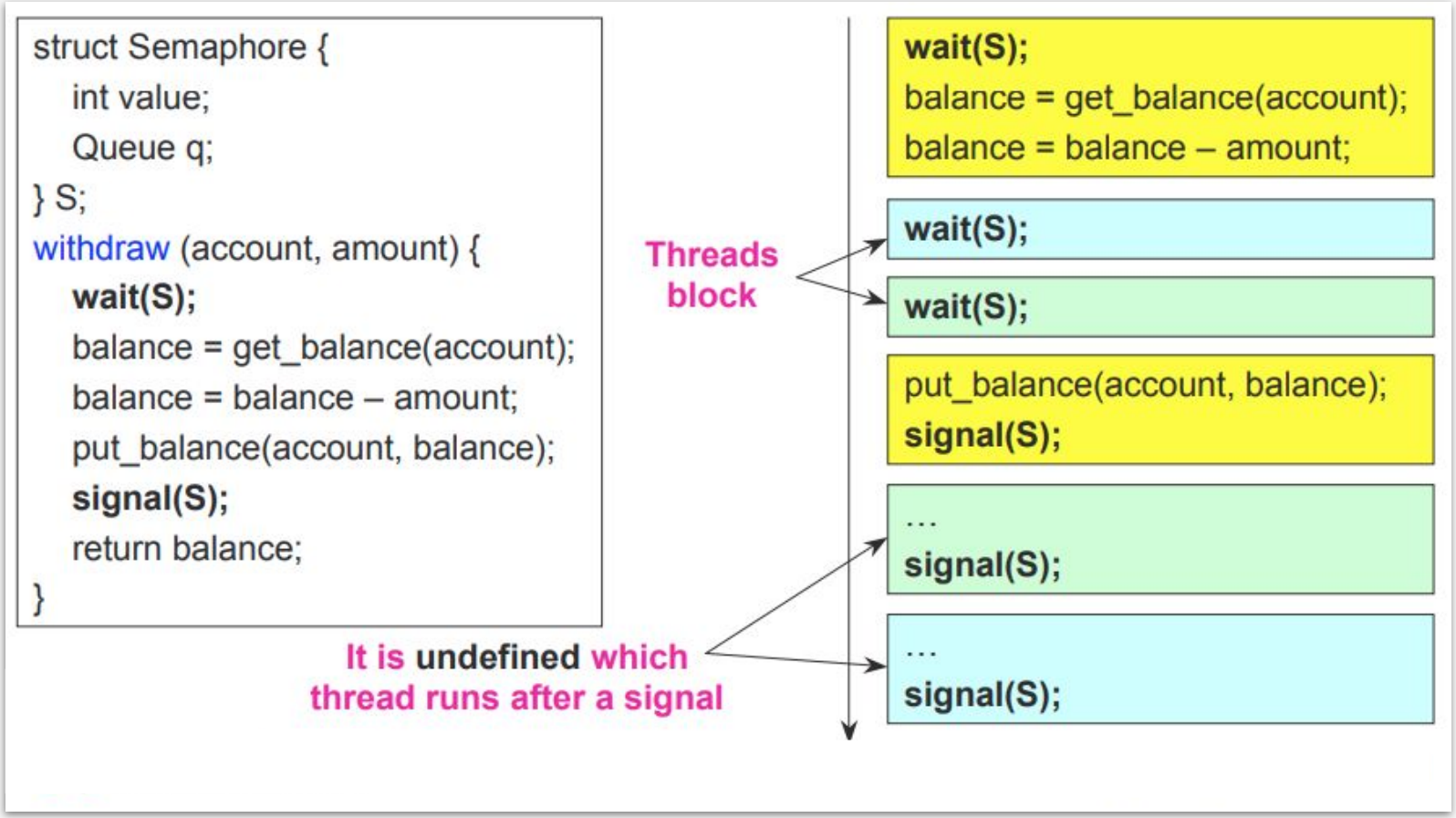
When wait() is called by a thread:

- 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;
}
Threads block
wait(S);
balance = get_balance(account);
balance = balance – amount;
wait(S);
wait(S);
put_balance(account, balance);
signal(S);
…
signal(S);
…
signal(S);
It is undefined which thread runs after a signal

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();
}
```

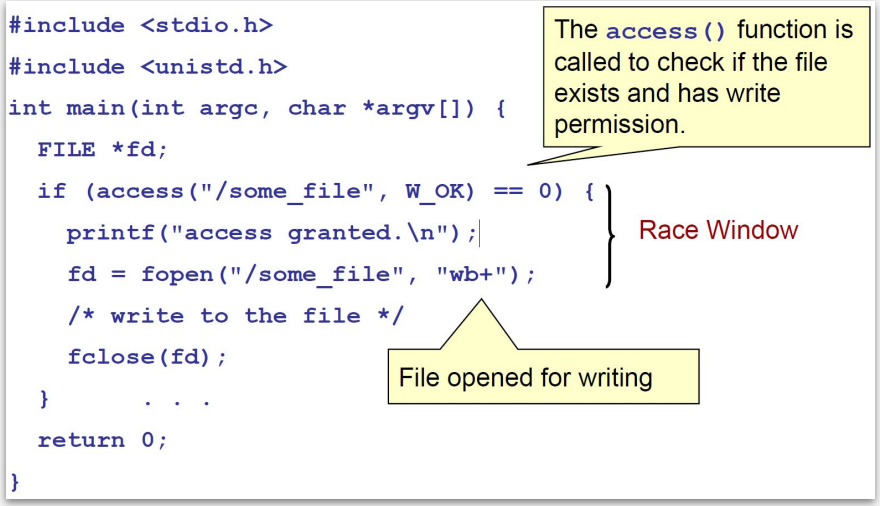
Time of Check, Time of Use (ToCToU)

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.



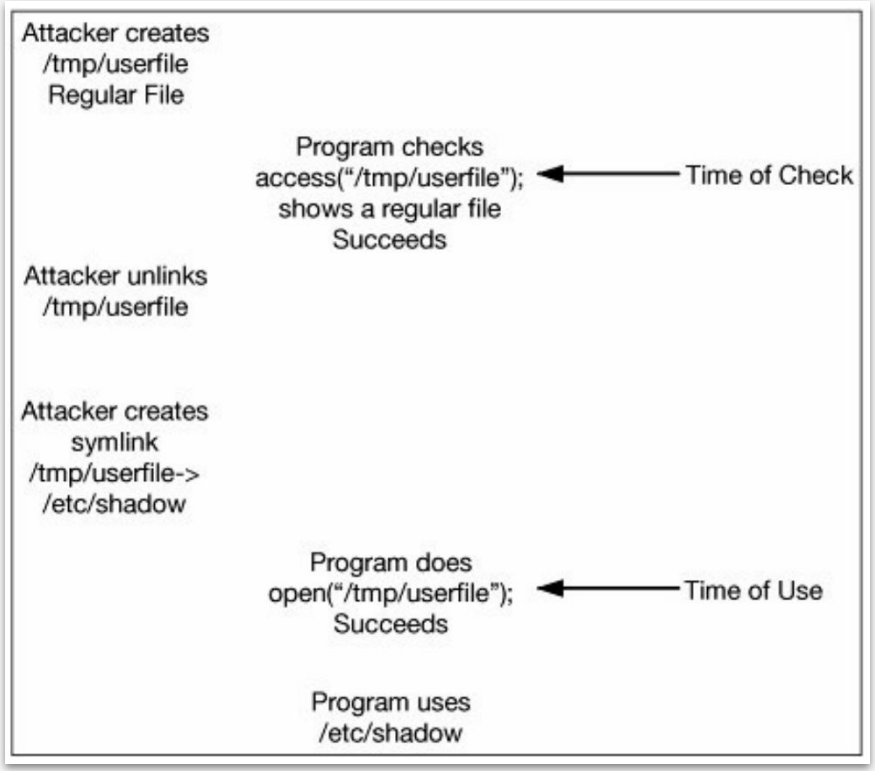
Continue ...

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.

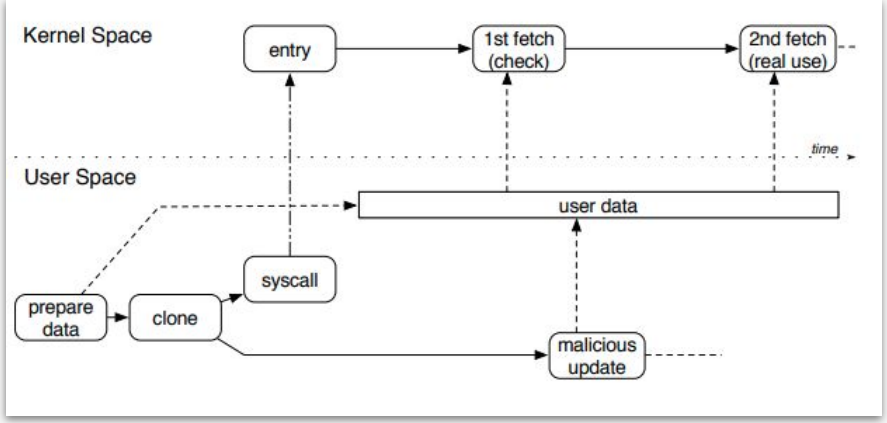


CVE Reports

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)

CVE Reports

Name	Description
CVE-2020-1641	A Race Condition vulnerability in Juniper Networks Junos OS LLDP implementation allows an attacker to cause LLDP to crash leading to a Denial of Service (DoS). This issue occurs when crafted LLDP packets are received by the device from an adjacent device. Multiple LACP flaps will occur after LLDP crashes. An indicator of compromise is to evaluate log file details for lldp with RLIMIT. Intervention should occur before 85% threshold of used KB versus maximum available KB memory is reached. show log messages | match RLIMIT | match lldp | last 20 Matching statement is " /kernel: %KERNEL-[number]: Process ([pid #],lldpd) has exceeded 85% of RLIMIT_DATA: " with [] as variable data to evaluate for. This issue affects: Juniper Networks Junos OS: 12.3 versions prior to 12.3R12-S15; 12.3X48 versions prior to 12.3X48-D95; 15.1 versions prior to 15.1R7-S6; 15.1X49 versions prior to 15.1X49-D200; 15.1X53 versions prior to 15.1X53-D593; 16.1 versions prior to 16.1R7-S7; 17.1 versions prior to 17.1R2-S11, 17.1R3-S2; 17.2 versions prior to 17.2R1-S9, 17.2R3-S3; 17.3 versions prior to 17.3R2-S5, 17.3R3-S6; 17.4 versions prior to 17.4R2-S4, 17.4R3; 18.1 versions prior to 18.1R3-S5; 18.2 versions prior to 18.2R2-S7, 18.2R3; 18.2X75 versions prior to 18.2X75-D12, 18.2X75-D33, 18.2X75-D50, 18.2X75-D420; 18.3 versions prior to 18.3R1-S7, 18.3R2-S3, 18.3R3; 18.4 versions prior to 18.4R1-S5, 18.4R2; 19.1 versions prior to 19.1R1-S4, 19.1R2.
CVE-2020-15586	Go before 1.13.13 and 1.14.x before 1.14.5 has a data race in some net/http servers, as demonstrated by the httputil.ReverseProxy Handler, because it reads a request body and writes a response at the same time.
CVE-2020-15567	An issue was discovered in Xen through 4.13.x, allowing Intel guest OS users to gain privileges or cause a denial of service because of non-atomic modification of a live EPT PTE. When mapping guest EPT (nested paging) tables, Xen would in some circumstances use a series of non-atomic bitfield writes. Depending on the compiler version and optimisation flags, Xen might expose a dangerous partially written PTE to the hardware, which an attacker might be able to race to exploit. A guest administrator or perhaps even an unprivileged guest user might be able to cause denial of service, data corruption, or privilege escalation. Only systems using Intel CPUs are vulnerable. Systems using AMD CPUs, and Arm systems, are not vulnerable. Only systems using nested paging (hap, aka nested paging, aka in this case Intel EPT) are vulnerable. Only HVM and PVH guests can exploit the vulnerability. The presence and scope of the vulnerability depends on the precise optimisations performed by the compiler used to build Xen. If the compiler generates (a) a single 64-bit write, or (b) a series of read-modify-write operations in the same order as the source code, the hypervisor is not vulnerable. For example, in one test build using GCC 8.3 with normal settings, the compiler generated multiple (unlocked) read-modify-write operations in source-code order, which did not constitute a vulnerability. We have not been able to survey compilers; consequently we cannot say which compiler(s) might produce vulnerable code (with which code-generation options). The source code clearly violates the C rules, and thus should be considered vulnerable.
CVE-2020-14416	In the Linux kernel before 5.4.16, a race condition in tty->disc_data handling in the slip and slcan line discipline could lead to a use-after-free, aka CID-0ace17d56824. This affects drivers/net/slip/slip.c and drivers/net/can/slcan.c.
CVE-2020-10602	In OSIsoft PI System multiple products and versions, an authenticated remote attacker could crash PI Network Manager due to a race condition. This can result in blocking connections and queries to PI Data Archive.
CVE-2019-9818	A race condition is present in the crash generation server used to generate data for the crash reporter. This issue can lead to a use-after-free in the main process, resulting in a potentially exploitable crash and a sandbox escape. *Note: this vulnerability only affects Windows. Other operating systems are unaffected.*. This vulnerability affects Thunderbird < 60.7, Firefox < 67, and Firefox ESR < 60.7.
CVE-2019-7614	A race condition flaw was found in the response headers Elasticsearch versions before 7.2.1 and 6.8.2 returns to a request. On a system with multiple users submitting requests, it could be possible for an attacker to gain access to response header containing sensitive data from another user.
CVE-2019-5796	Data race in extensions guest view in Google Chrome prior to 73.0.3683.75 allowed a remote attacker to potentially exploit heap corruption via a crafted HTML page.
CVE-2019-3901	A race condition in perf_event_open() allows local attackers to leak sensitive data from setuid programs. As no relevant locks (in particular the cred_guard_mutex) are held during the ptrace_may_access() call, it is possible for the specified target task to perform an execve() syscall with setuid execution before perf_event_alloc() actually attaches to it, allowing an attacker to bypass the ptrace_may_access() check and the perf_event_exit_task(current) call that is performed in install_exec_creds() during privileged execve() calls. This issue affects kernel versions before 4.8.
CVE-2019-2219	In System UI, there is a possible bypass of user's consent for access to sensor data due to a race condition. This could lead to local information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.Product: AndroidVersions: Android-8.0 Android-8.1 Android-9 Android-10Android ID: A-119041698
CVE-2019-16779	In RubyGem excon before 0.71.0, there was a race condition around persistent connections, where a connection which is interrupted (such as by a timeout) would leave data on the socket. Subsequent requests would then read this data, returning content from the previous response. The race condition window appears to be short, and it would be difficult to purposefully exploit this.
CVE-2019-15879	In FreeBSD 12.1-STABLE before r356908, 12.1-RELEASE before p5, 11.3-STABLE before r356908, and 11.3-RELEASE before p9, a race condition in the cryptodev module permitted a data structure in the kernel to be used after it was freed, allowing an unprivileged process can overwrite arbitrary kernel memory.
CVE-2019-14070	Possible use after free issue in pcm volume controls due to race condition exist in private data used in mixer controls in Snapdragon Auto, Snapdragon Compute, Snapdragon Consumer IOT, Snapdragon Industrial IOT, Snapdragon Mobile, Snapdragon Voice & Music, Snapdragon Wearables, Snapdragon Wired Infrastructure and Networking in APQ8009, APQ8017, APQ8053, APQ8064, APQ8096AU, APQ8098, IPQ4019, IPQ6018, IPQ8064, IPQ8074, MDM9206, MDM9207C, MDM9607, MDM9615, MDM9640, MDM9650, MSM8905, MSM8909W, MSM8917, MSM8920, MSM8937, MSM8940, MSM8953, MSM8996AU, MSM8998, Nicobar, QCS605, QM215, Rennell, SA6155P, Saipan, SC8180X, SDA660, SDA845, SDM429, SDM429W, SDM439, SDM450, SDM630, SDM632, SDM636, SDM660, SDM670, SDM710, SDM845, SDX20, SDX24, SDX55, SM6150, SM7150, SM8150, SM8250, SXR1130, SXR2130
CVE-2018-5905	In all android releases (Android for MSM, Firefox OS for MSM, QRD Android) from CAF using the linux kernel, a race condition while accessing num of clients in DIAG services can lead to out of boundary access.
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-2018-12633	An issue was discovered in the Linux kernel through 4.17.2. vbg_misc_device_ioctl() in drivers/virt/vboxguest/vboxguest_linux.c reads the same user data twice with copy_from_user. The header part of the user data is double-fetched, and a malicious user thread can tamper with the critical variables (hdr.size_in and hdr.size_out) in the header between the two fetches because of a race condition, leading to severe kernel errors, such as buffer over-accesses. This bug can cause a local denial of service and information leakage.

Data Races vs Race Conditions

Not every race condition is a data race.

- race conditions can occur even when there is no shared memory access.
- for example in file systems or network access.

Not every data race is a race condition.

- the data race may not affect the result.
- for example if two threads write the same value to shared memory.

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

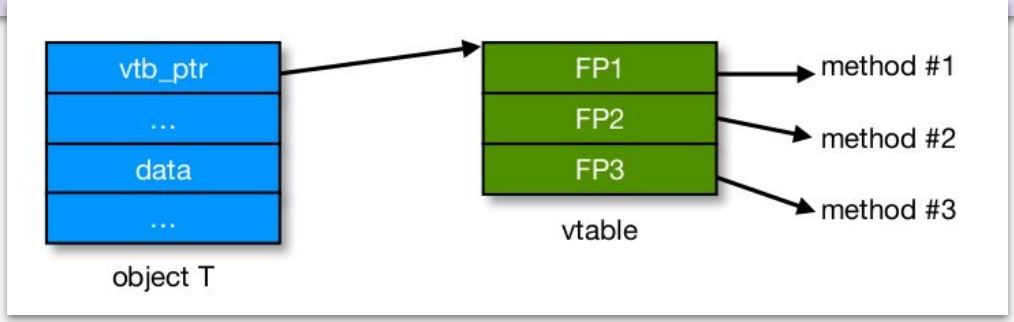
Race condition detection is NP complete.

- 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.

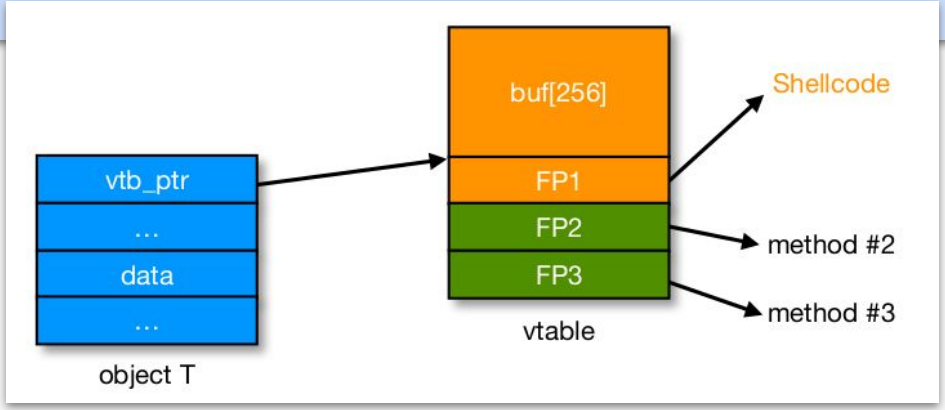
Heap Spray

Heap-based Control Flow Hijacking

C++ uses vtable to implement virtual functions.



After overflow of buf to overwrite vtable.

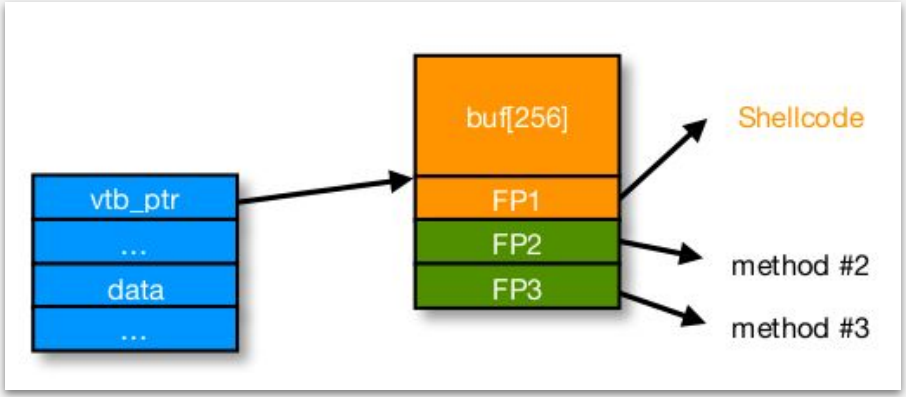


Heap Overflows in Browser

```
<SCRIPT language="text/javascript">
  shellcode = unescape("%u4343%u4343%...");
  overflow-string = unescape("%u2332%u4276%...");
  cause-overflow( overflow-string );    // overflow  buf[ ]
</SCRIPT>
```

Problem:

- Browser places shellcode on the heap at unknown location.
- How to reliably hijack control flow.

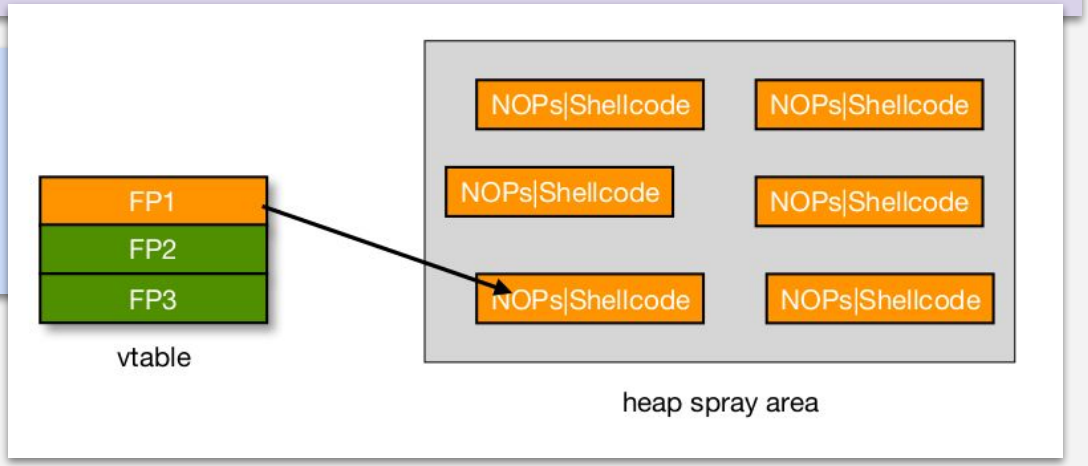


Heap Spraying

A technique used to increase exploit reliability, by filling the heap with large chunks of data relevant to the exploit you're trying to land.

- It can assist with bypassing ASLR.
- A heap spray is not a vulnerability or security flaw.

- Use Javascript to spray heap with shellcode (& NOP sleds).
- Then, point vtable pointer anyway in the spray area.



Javascript Heap Spraying

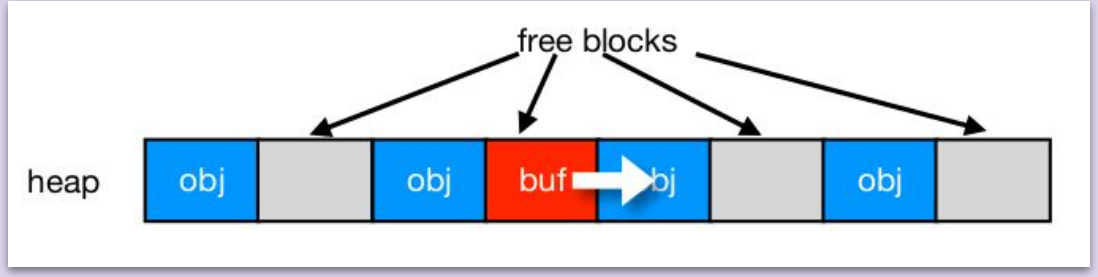
Pointing function pointer almost anywhere in heap will cause shellcode to execute.

```
var  nop = unescape("%u9090%u9090")
while (nop.length < 0x100000)  nop += nop

var shellcode = unescape("%u4343%u4343%...");

var x = new Array ()
for (i=0;  i<1000;  i++) {
    x[i] = nop + shellcode;
}
```

Vulnerable Buffer Placement

Placing vulnerable **buf[256]** next to object O:

- By sequence of Javascript allocations and frees make heap look as follows:



- Allocate vuln. buffer in Javascript and cause overflow.
- Successfully used against a Safari PCRE overflow.

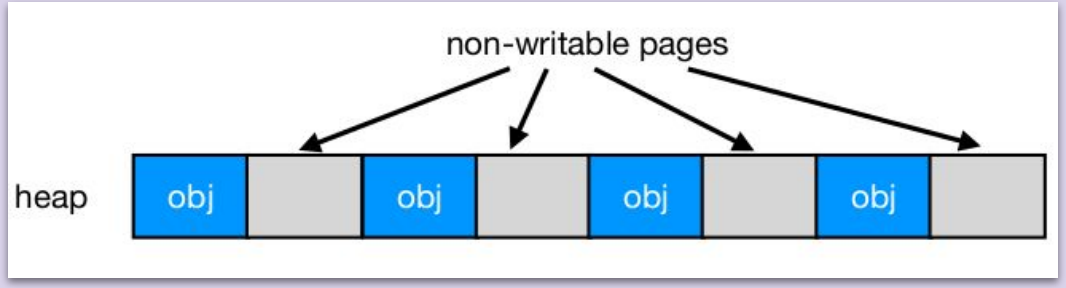
Heap Feng Shui

- Reliable heap exploits on IE without spraying.
- Gives attacker full control of IE heap from Javascript.

Date	Browser	Description
11/2004	IE	IFRAME Tag BO
04/2005	IE	DHTML Objects Corruption
01/2005	IE	.ANI Remote Stack BO
07/2005	IE	`javaprxy.dll` COM Object
03/2006	IE	`createTextRang` RE
09/2006	IE	VML Remote BO
03/2007	IE	ADODB Double Free
09/2006	IE	WebViewFolderIcon `setSlice`
09/2005	FF	0xAD Remote Heap BO
12/2005	FF	`compareTo()` RE
07/2006	FF	`Navigator` Object RE
07/2008	Safari	Quicktime Content-Type BO

Defenses

- Protect heap function pointers (e.g. PointGuard).
- Better browser architecture:
  - Store JavaScript strings in a separate heap from browser heap.
- OpenBSD heap overflow protection:
  - Prevents cross-page overflows.



- Nozzle: detect sprays by prevalence of code on heap.

< Software Attack />