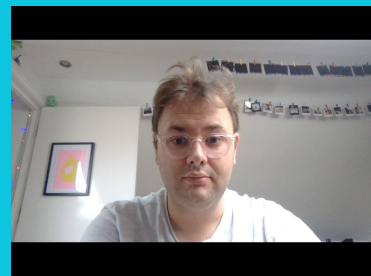
University of BRISTOL

The Dirty CoW

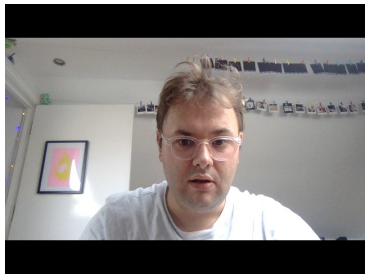
Joseph Hallett



What is it?

- Dirty *Copy on Write* vulnerability
- Linux vulnerability that existed between ~2007 to 2016
- Exploited in the wild (for Android (<7) rooting and worse)
- (Re) discovered by Phil Oester
- CVE-2016-5195

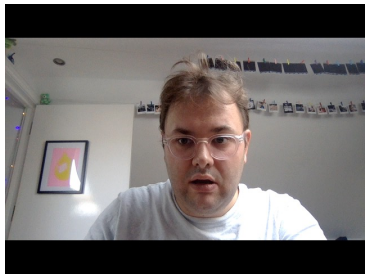
- Allowed writes to read-only root files (e.g. system binaries and password files)
- A race condition!



Mmap'd files

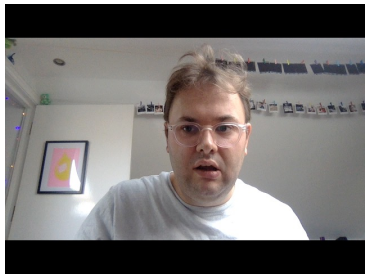
On Linux *(almost)* everything is a file
Sometimes you'd like to access a file as just a big region of memory though
Handy for writing device drivers
Avoids mucking about with read and write

```
NAME
     mmap -- allocate memory, or map files or devices into memory

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <sys/mman.h>

     void *
     mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset);
```

So what does this look like…

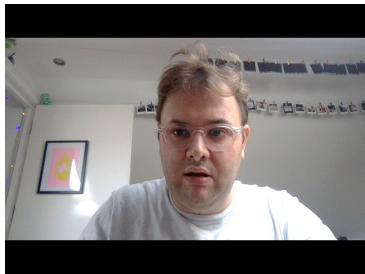
```
int fd;
size_t size;
fd = open("file.txt", O_RDONLY);
/* get file size with stat */
mapped = mmap(0, size,
              PROT_READ,   // Read-only
              MAP_PRIVATE, // Create a copy if modified...
              fd, 0);
mmaped[10]; // Access the file directly!
```



MAP_PRIVATE

Map private allow marks the memory as being *copy on write* (COW)
Sometimes you don't know whether you're going to need to modify memory…
Copying whole pages of memory is slow… so only do it when you write!

Mmap'd file is the original RO file… but as soon as you write to it the memory is marked dirty and the file gets re-written out to a different private region of memory…
Write happens by locating the real location of the file in memory,
then doing the real write

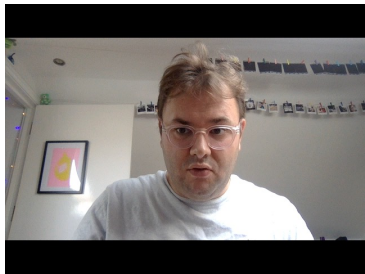


madvise

```
NAME
     madvise, posix_madvise -- give advice about use of memory

SYNOPSIS
     #include <sys/mman.h>

     int
     madvise(void *addr, size_t len, int advice);
```

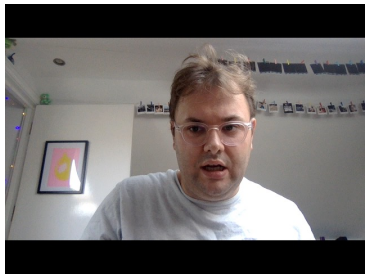
```
MADV_DONTNEED    Indicates that the application is not expecting to
                 access this address range soon.  This is used with
                 madvise() system call.
```

in other words, throw away any dirty pages and re-read from the disk if when you get asked next time



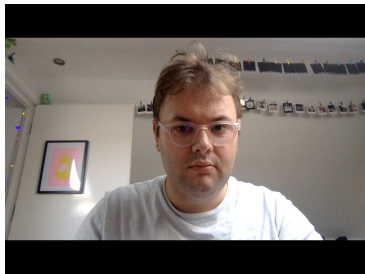
Race Condition

- Mmap the read-only file with MAP_PRIVATE
- Try and write to it (via a direct pointer into memory)



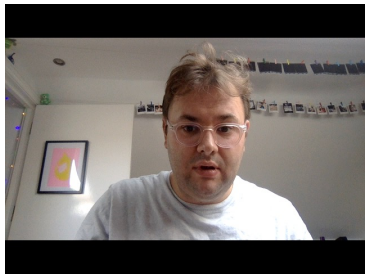
Race Condition

- Mmap the read-only file with MAP_PRIVATE
- Try and write to it (via a direct pointer into memory)
- Write locates what memory to write



Race Condition

- Mmap the read-only file with MAP_PRIVATE
- Try and write to it (via a direct pointer into memory)
- Write locates what memory to write
- Marks as dirty
- Does the write

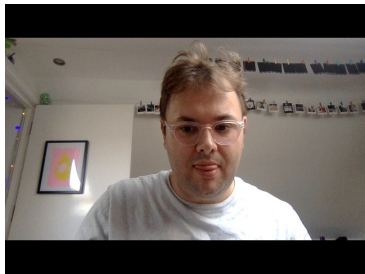


Race Condition

- Mmap the read-only file with MAP_PRIVATE
- Try and write to it (via a direct pointer into memory)
- Write locates what memory to write
- Marks as dirty

- Does the write
- Write gets applied to underlying file
- No CoW

- Advises you don’t need the page
- Throws away dirty flag



What went wrong

- Linux assumed the race would be too hard to win
- Which was true-ish in 2007… but not in 2016

- Security fixes got reverted and forgotten…