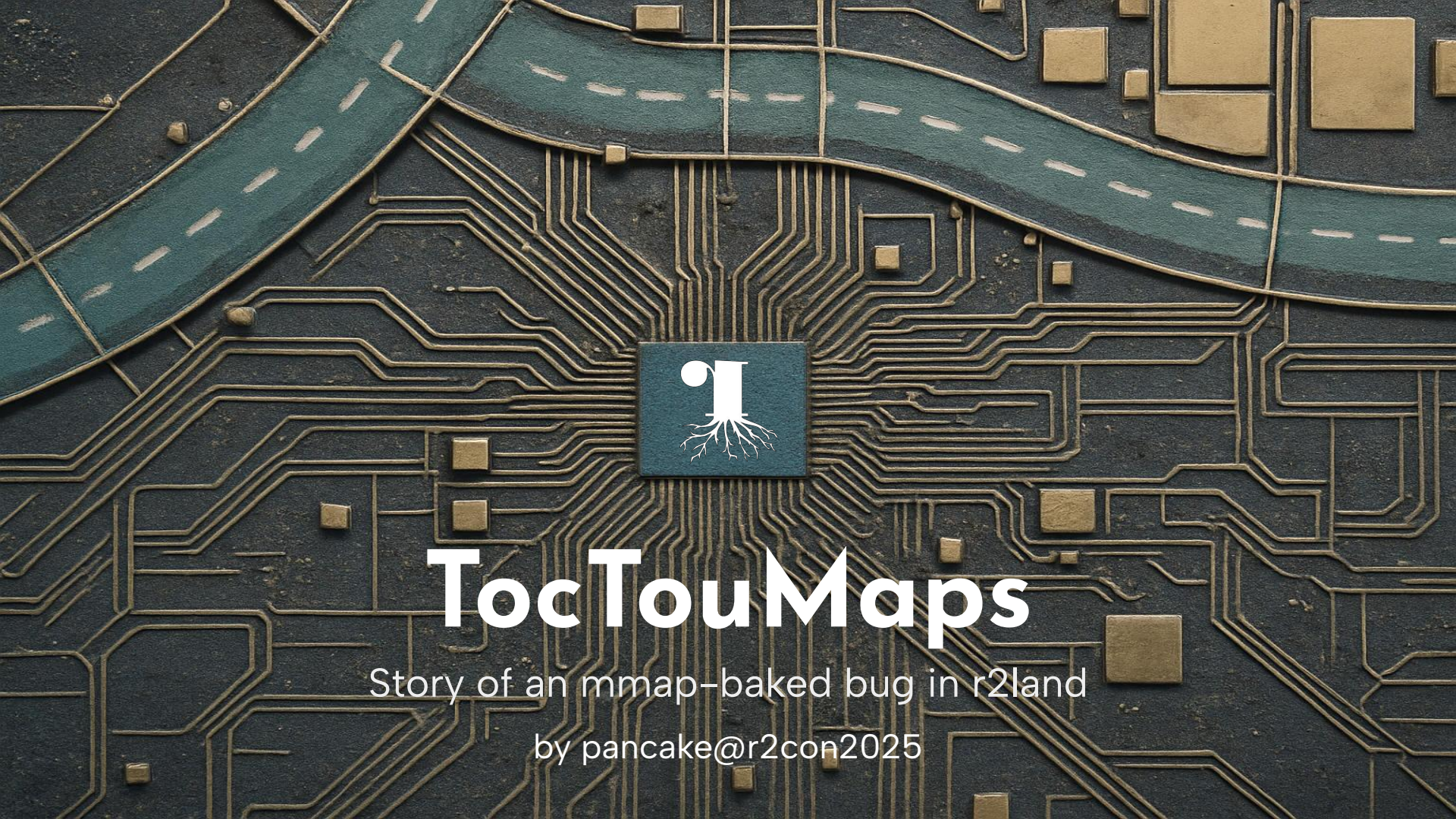

TocTouMaps

Story of an mmap-baked bug in r2land

by pancake@r2con2025

Who Am I?

Sergi Àlvarez aka **pancake**

- Mobile Security Research Engineer at NowSecure
- Author and main contributor of Radare
- Free Software enthusiast and developer

Introduction

Once upon a time... we decided to use mmap for File IO

- Performance motivations
- Kernels back then didn't cache read-ahead filedescriptors
- Sharing file changes in realtime across multiple r2 instances

But:

- Cannot determine when the underlying data changes
  - **REALLY?**

The Bug

MMaped memory is tied to physical files in disk.

- Shrinking the file results in invalid memory accesses

```
r2 -qwc'wtf a $$ @ 5;px 32' a
```

Stupid but fatal bug spotted by **@astralia**@infosec.exchange

Fixed by me in: **987acbb** (“Fix oobread when using mmap baked files”)

Safer approach: **c25aca4** (“Use the safe and atomic mmap read api”)

Proof Of Concept

mmap-based linear buffer access caused OOB/SIGBUS crashes when files were truncated or rewritten mid-call

- Program **A** maps the file and reads in loop
- Program **B** truncates the file and writes back
- Program **A** receives a SIGBUS

```
Use of a mapped region can result in these signals:

SIGSEGV
       Attempted write into a region mapped as read-only.

SIGBUS Attempted access to a page of the buffer that lies beyond
       the end of the mapped file.  For an explanation of the
       treatment of the bytes in the page that corresponds to the
       end of a mapped file that is not a multiple of the page
       size, see NOTES.
```

Wait, a BUS?

```
$ man 7 signal | grep BUS

SIGBUS   Core    Bus error (bad memory access)
```

We can capture the SIGBUS signal, but that's clearly not a fix.

- Can we have atomic operations with mmap? **NO**
- A notification at least? **NO**
- File Locks (kind of filesystem-level mutexes) **NO** (not portable)
- **Other software is affected? YES**

How it works

Before going into details. Let's learn about this UNIX syscall:

```
void *mmap(void addr[.length], size_t length, int prot, int flags, int fd, off_t offset);

int msync(void addr[.length], size_t length, int flags);

int munmap(void addr[.length], size_t length);
```

Remember **sbrk**? Well, now mmap is the recommended way to implement memory allocators on UNIX systems

Fifty Shades of MMap Flags (macOS)

MAP_SHARED, MAP_PRIVATE, MAP_FIXED, MAP_ANON, MAP_FILE, MAP_FIXED, MAP_ANON, MAP_FILE, MAP_HASSEMAPHORE, MAP_NOCACHE, MAP_JIT, MAP_32BIT

- MAP_SHARED: all processes see the changes
- MAP_PRIVATE: aka CoW with UB (unspecified by POSIX)
- MAP_FIXED: address specified is mandatory, not a hint

Fifty Shades of MMap Flags (Linux)

MAP_SHARED, MAP_SHARED_VALIDATE, MAP_PRIVATE, MAP_32BIT, MAP_ANON. MAP_ANONYMOUS, MAP_DENYWRITE, MAP_LOCKED, MAP_NONBLOCK, MAP_NORESERVE, MAP_POPULATE, MAP_STACK, MAP_SYNC, MAP_UNINITIALIZED, MAP_EXECUTABLE, MAP_FILE, MAP_FIXED, MAP_FIXED_NOREPLACE, MAP_GROWSDOWN, MAP_HUGETLB, MAP_HUGE_2M, MAPHUGE_1GB

- There are some discussions about in Linux to introduce MAP_NOSIGBUS to reduce the dramatism, but the root problem persists.

The DLOPEN case

Linux's dlopen is affected by this problem (macOS not much)

- Deleting the file or renaming it does not invalidates the map
- Unix use "install" or "rm;cp" to replace libraries and avoid segfaults

DEMO

- Modify dlopened library in-place
  - Hot-reload code at runtime

Locks and Atomicity

Filesystem-level locks can emulate atomicity and reduce the TOCTOU window, but all the programs accessing it MUST use it. (also NFS..)

- flock(2): locks **advisory** (LOCK_EX/LOCK_SH) - not portable, not respected by NFS/SMB;
- fcntl(F_SETLK) locks record locks
- Use pread/pwrite to avoid seek+read races
- Explain ETXTBSY <- MAP_DENYWRITE - avoid writing the file if the program is being executed

https://stackoverflow.com/questions/1573732/using-dlopen-how-can-i-cope-with-changes-to-the-library-file-i-have-loaded

Portability

We explored GNU/Linux and macOS provide weak and overengineered solutions that just don't solve the problem we are facing.

- Lots of non-portable and non-posix flags
- POSIX lefts the door open to undefined behaviour

Windows is not affected because CreateFileMapping/MapViewOfFile does not allow to truncate or modify a file that is mapped with a sharing violation exception. More limited, but at least its safe.

Advise with madvise(2)

Hint the kernel about how to use a specific memory range:

- read-ahead optimizations, access types, disable swap
- MAP_PRIVATE (CoW)
- MADV_WILLNEED/DONTNEED - prefetch range into cache
  - Same as mmap with MAP_POPULATE

Portability? **Not today**

Actions Taken

Short solution

- Reduce toctou window:
  - Use flocks with fstat to check size
  - Use atomic pread/pwrite
  - Use madvise hints or page guards

Long term solution

- Keep the mmap:// implementation as is, but not as the default
- Default file IO using plain fds (now it's 2x faster than mmap)
- Test with stdio://, but not yet enabled by default

Questions and Follow Ups

I'll publish the source code of all these demos in the r2con2025 repository after the conference.

If you have questions or ideas drop them in the chat!

- Or reach me out in the fediverse! –> @pancake@infosec.exchange


Thanks for Watching
—pancake