Why should you learn about the filesystem in 2026?

Gergely Kalman
OFTW, 2026

whoami

- my name is **Gergely** (**call me Greg**)
  - twitter: **@gergely_kalman**
  - I do bug bounties
  - full time in the **Apple Security Bounty** (**ASB**) program

My assumptions

- you are either a **student**
- or looking for a **career change**

Why should you listen

- Because I started bug bounties **4 years ago**
  - as a **burned-out programmer**
- managed to find a few **0days**
- was enough to **live comfortably**

You might have heard these

“Build your career”

“Climb the corporate ladder”

“Firm handshakes”

“Other boomer sh*t”

None of that works

- getting a job is **insanely difficult**
- especially:
  - getting a **good** one
  - if you have little to no **experience**
- to make matters worse...

The state of the industry

- the industry is going through a **“transformation”**
- **Nobody knows what will happen**
- but it doesn’t look great
  - We’re clearly in a bubble
  - AI might take our jobs
- feels like when fuzzing became mainstream

It’s a scary time

Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

What can you do?

- learn **objectively valuable** skills (research what these are)
- **demonstrate the skills** you already have (publish code, past research)
- **research opportunities** (what is valuable and **why**)
- **talk to people in your field** (they can become mentors or future coworkers)

A concrete example

- **hacking** was my choice of **objectively valuable skill**
  - might as well do something that I always wanted to...
- I researched this career
- I read a lot
- I practiced
- I eventually found Apple’s bug bounty program
- Let me explain bug bounties quickly...

Bugs and vulnerabilities

- all software has **bugs**
- some bugs are **vulnerabilities** (security bugs)
- **bug → vulnerability** if it's **exploitable**

Vulnerabilities are exploitable bugs

What are 0days?

- **0days** are vulnerabilities that are **not public**
  - unknown to the vendor and users
  - unpatched (obviously)
- Good ones are **rare** and **powerful** → **valuable** and **interesting**
- Finding them is **impressive:**
  - **You have something nobody else has**

Basically

bugs

vulnerabilities

0-days

Each 0day is unique

serious bug in niche software

useless bug in popular software

VS

serious bug in popular software

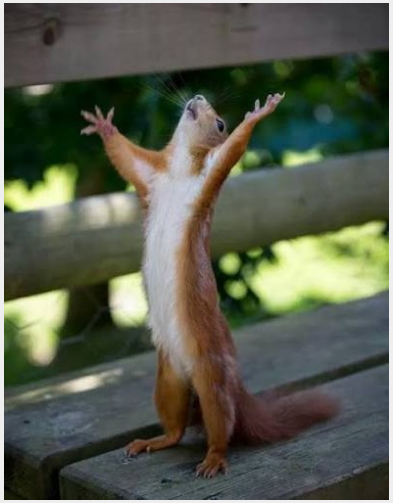
Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

You mean they pay money?

Bug bounties

YES!

this is what a bug bounty is



Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

Unsolicited career advice

- **Let me suggest that you try bug bounties**
- Specifically the **ASB** (Apple Security Bounty)
  - everyone knows **Apple**
  - they **pay well***
  - program scope is **huge**

*: except when they unilaterally decide to nerf full TCC bypasses by ~84%

Being a bug hunter is cool

- **Be your own boss: pick when/how/what to attack**
- collect **bounties** and **CVEs**
- improve the life of **millions** if not **billions of people**
  - **macOS** is used by **~100 million people** (~1.2% of the world)
  - **iOS** is used by **1 billion people** (12% of the world)
- go to conferences (travel the world)

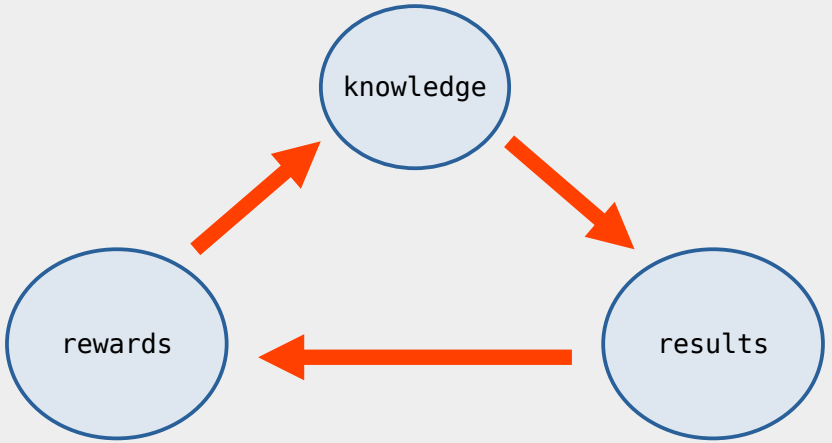
Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

Not a sponsored talk

- **Apple did NOT pay me for this BTW**
- I’m telling you, because:
  - this **worked for me**
  - **people ask me about** it all the time
- IMO bug bounties are genuinely useful
  - it’s a 0-risk way to **improve your skills** and **profile**
  - **it’s a ton of fun**
  - **no job interview** or permission is required

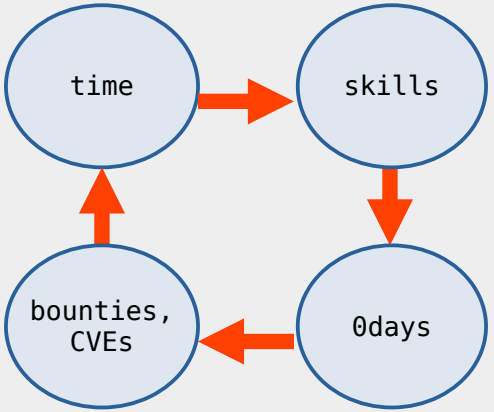
Universal game loop of a career

**Extremely simplified (and wrong),
but bear with me**



Translated to bug bounties

- more time spent → better skills
- better skills → more bugs
- more bugs found → more bounties and reputation
- more bounties and CVEs → more time you can/want to spend



Pitfalls

- time spent but no skills → **you're doing mindless work / relying on LLMs**
- skills but no 0days → **unlucky / underskilled / bad target selection**
- 0days but no bounties/CVEs → **report was bad / program is bad**
- bounties/CVEs but no time spent → **you are burned out / demotivated**

Let’s find 0days!

- my focus
  - **logic bugs**
  - **filesystem / file operation** bugs
- because files / file ops are:
  - **everywhere**
  - **hard to mitigate**
  - **fundamentally hard to get right**

Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

Recommendations for beginners

- also: this is **beginner-friendly**
  - patterns are easy to spot
  - attack paths are well mapped-out
- **Beginner**: Start with **soft targets** and **well-known attacks**
- **Advanced**: Graduate to more **hardened targets**
- **Expert**: Find **novel attack vectors / new bug classes**

How hard are file operations?

- well...

File ops are shockingly hard

no copy() syscall

“Everything is a file”

legacy filesystems

symlinks are hard to prevent

in-band signaling in file operations

. and .. are special

network filesystems

race conditions everywhere

POSIX permissions are incredibly complex

mountpoints can move

CWD is unintuitive

File ops are shockingly hard

no copy() syscall

case-insensitivity

reliance on extended attributes

“Everything is a file”

legacy filesystems

user sets fs options

user mounting

/.file, /.vol/

symlinks are hard to prevent

in-band signaling in file operations

firmlinks

sandbox_exec

union mounts

. and .. are special

user can change mount options at runtime

network filesystems

noowners

race conditions everywhere

applesingle/appledouble

in-band signaling: /..namedfork/rsrc

POSIX permissions are incredibly complex

(force) unmount

hardlinked directories

mountpoints can move

CWD is unintuitive

File ops are shockingly hard

- file operations are racy
- Access Control is complex
- path resolution is unintuitive
- users can mount (alter the FS graph)
- advanced protections are basically regex based
- attackers can deploy a ton of tricks

Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

Mitigation is impossible

- **you can’t:**
  - **break the API**
  - **rewrite everything**
  - **break with POSIX’s 40 year old traditions**
- **best you can do is to silo privileged programs**
  - not always feasible (/tmp/ files, shared files, etc…)
- more on this later...

Meeting the prerequisite

- we need at least **one controllable path segment**
  - **/Users/student/hello/legitimate.txt**
- this is enough because **parents can control the descendants**
  - write access, symlinks, mounts, inheriting permissions

Meeting the prerequisite

Example: **root** daemon writes to **/Users/student/hello/legitimate.txt**

→ **/Users/student/../../../../../../etc/sudoers.d/legitimate.txt**

→ **root write to /etc/sudoers.d/legitimate.txt**

Why should you learn about the filesystem in 2026?
https://gergelykalman.com (@gergely_kalman), 2026

Meeting the prerequisite

- on **macOS** this is often a given
  - particularly with **entitled** apps
- most of the protections are not by-design
- circumvention is easy

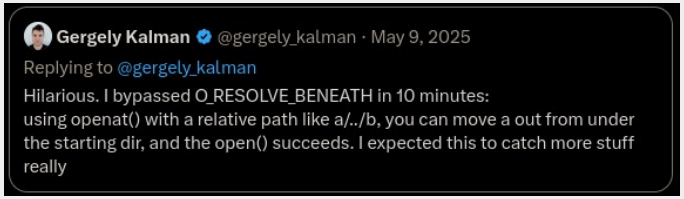
Apple’s mitigations

- Apple introduced **Launch Constraints**
  - You **can no longer directly execute most apps**
    - not really a protection against file operation issues...
- Apple reduced **TCC bypass** payouts by **83.6% (!)** last December
  - this upset **everyone**
    - hardly a mitigation as bad guys don’t care…
- You might have seen these guys:

```
O_NOFOLLOW_ANY                  ----------   1 root  admin     0 Jul 22 23:57 .file
O_RESOLVE_BENEATH               drwxr-xr-x   2 root  wheel    64 Jul 22 23:57 .nofollow
RENAME_NOFOLLOW_ANY             drwxr-xr-x   2 root  wheel    64 Jul 22 23:57 .resolve
```

Apple’s mitigations

- sure, there’s **O_RESOLVE_BENEATH**
  - **which I bypassed in this tweet:**



- and **O_NOFOLLOW_ANY** being put everywhere
  - which only works if you can break **POSIX** compatibility
    - 0% chance of this being merged upstream in portable codebases

Retrofitting security

- we know **retrofitting security is a bad idea**
- but this is what **Apple** is doing (and **AppArmor** and **SELinux**)
- **a commendable effort**
  - but extremely difficult to accomplish
  - mostly a clunky patchwork with slow gradual improvement
- **still better than doing nothing**

my targets in 2026

- ~~**macOS TCC bypass**~~
- **macOS LPE** (user → root)
- **macOS SIP bypass** (root to SIP-exempt root)
- **macOS App SBX** (sandboxed code exec to unsandboxed)
- **iOS: anything** (I can find)

Attack example: LPE

- the classic **POSIX** LPE
- **GOAL:** take control over an important file
  - /etc/sudoers, /etc/sudoers.d/*
  - /etc/cups/*
  - many others...
- **targets**: processes running as **root**
- **examples**: hijacked **chmod()** / **rename()** / **chown()** / **open()**
- there are many things to attack with this...

Attack example: SIP bypass

- bug in an Apple-signed installer package's scripts
- requires **root** access
- **GOAL:** control a process that is a descendant of **system_installd**
- **targets**: scripts in Apple-signed installers
- **examples**: script command injection, execution of a program you control, hijacking privileged file operations
  - make these do something useful

Attack example: App SBX

- Application Sandbox Escape
- **GOAL:** have a directory created without the quarantine flag
- **targets**: **XPC** endpoints reachable from the sandbox
- **examples**: **XPC** endpoint with an arbitrary dir create bug
- a simple dir **whatever.app** dir create is enough to break out
  - https://gergelykalman.com/CVE-2023-32364-a-macOS-sandbox-escape-by-mounting.html

Attack example: iOS

- **I’m out of depth here...**
  - but made partial progress
- **Grateful for any hints**
  - Suppose I have an arbitrary file write as root…
    - talk to me...

Attack surfaces in 2026

- lots of low-hanging fruit is gone
  - but **not nearly all**
- **XPC** (IPC) is still incredibly promising
- **LLM**s are actually decent at reading code
  - credit where credit’s due

Where to go from here?

- lots of good blogs
- lots of good talks
- I published a **lot about this**
- but it’s never enough...

Happy Hunting!

Thank you