Finding Bugs:
42 Tips from 4 Security Researchers

WHO?

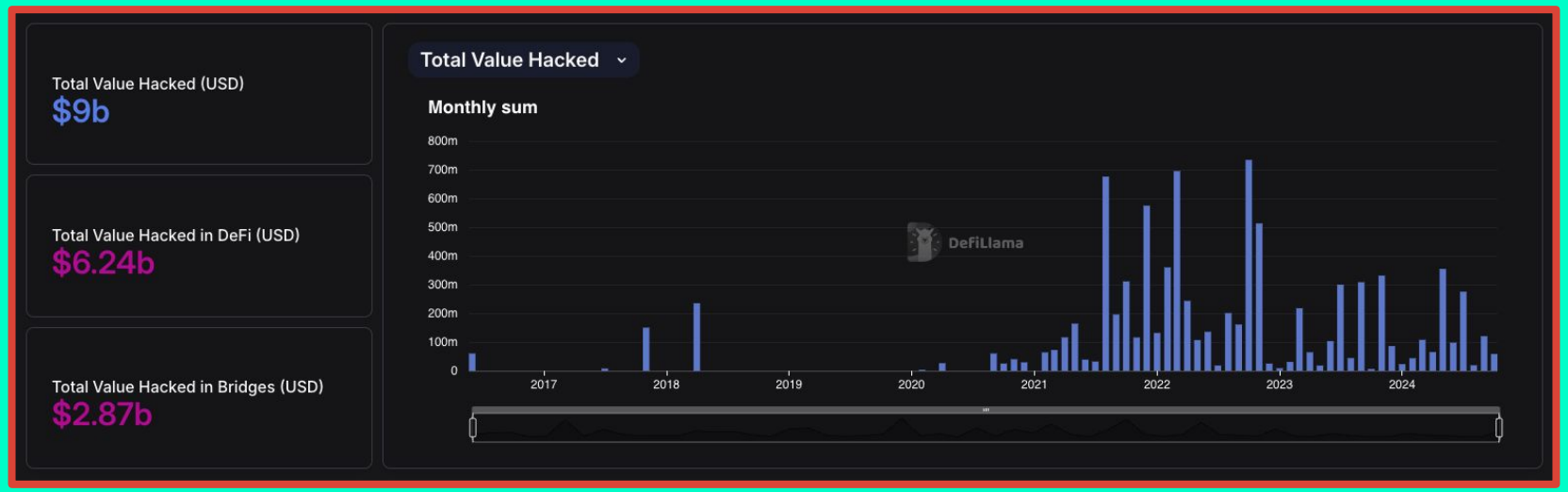
Joran

Nat

tincho

0xRajeev

WHY?


Total Value Hacked (USD)
$9b
Total Value Hacked in DeFi (USD)
$6.24b
Total Value Hacked in Bridges (USD)
$2.87b
Total Value Hacked
Monthly sum
800m
700m
600m
500m
400m
300m
200m
100m
0
2017
2018
2019
2020
2021
2022
2023
2024
DeFiLlama

Tip #1 - Pacing

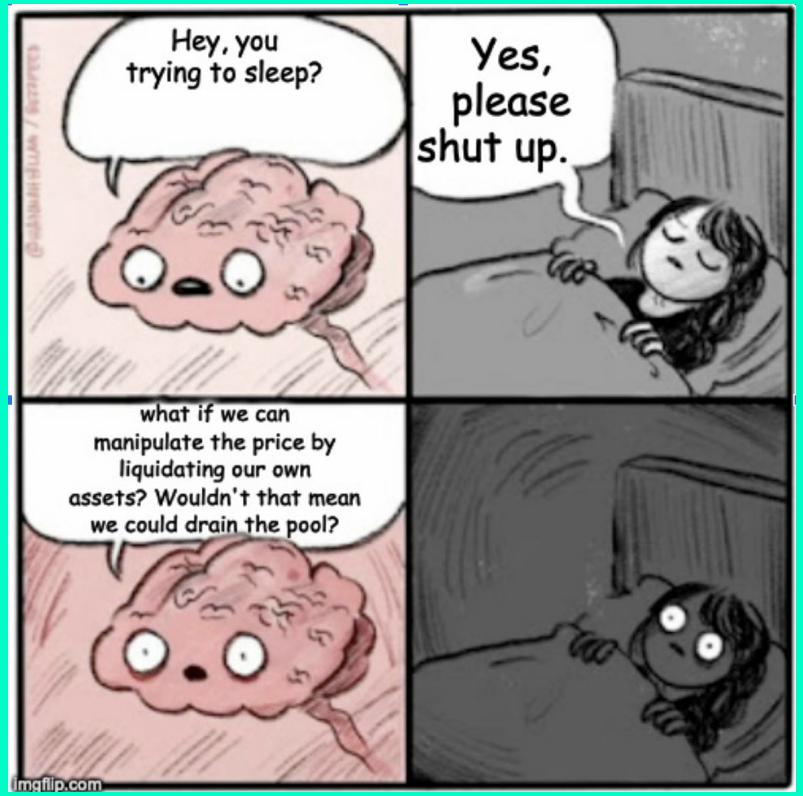
the grindset is a lie

Tip #2 - Scoping out targets

- picking the right target is important

- type
- amounts (espec. medium)
- age of program
- size

Tip #3 – try to sleep (sort of)



Tip #4 - BB discussions -> prepare for fallacies

- every bug bounty submission is different
- everyone responds differently to having a bug submitted
- sometimes there is disagreement

- not intentional
- preparation -> productive conversation

Tip #5 - BB Discussions -> See devs as a business partner

- style of communication
- productive

Tip #6 - diversify your skills

- competitive advantage

- Become a quant nerd
- learn about economics
- learn cryptography
- learn how provers work
- learn how compilers work

Some strategies

Tip #7 - find a paradigm and specialize

- defi
  - leverage
  - concentrated liquidity
  - …
- zk
- go broad

Tip #8 - go deep

- optimal for time constrained bb hunters

- explore
- identify
- exploit

Tip #9 - Read the documentation

- design mistakes can be critical
- quick
- setup for success

Tip #10 - the toolbox mental model

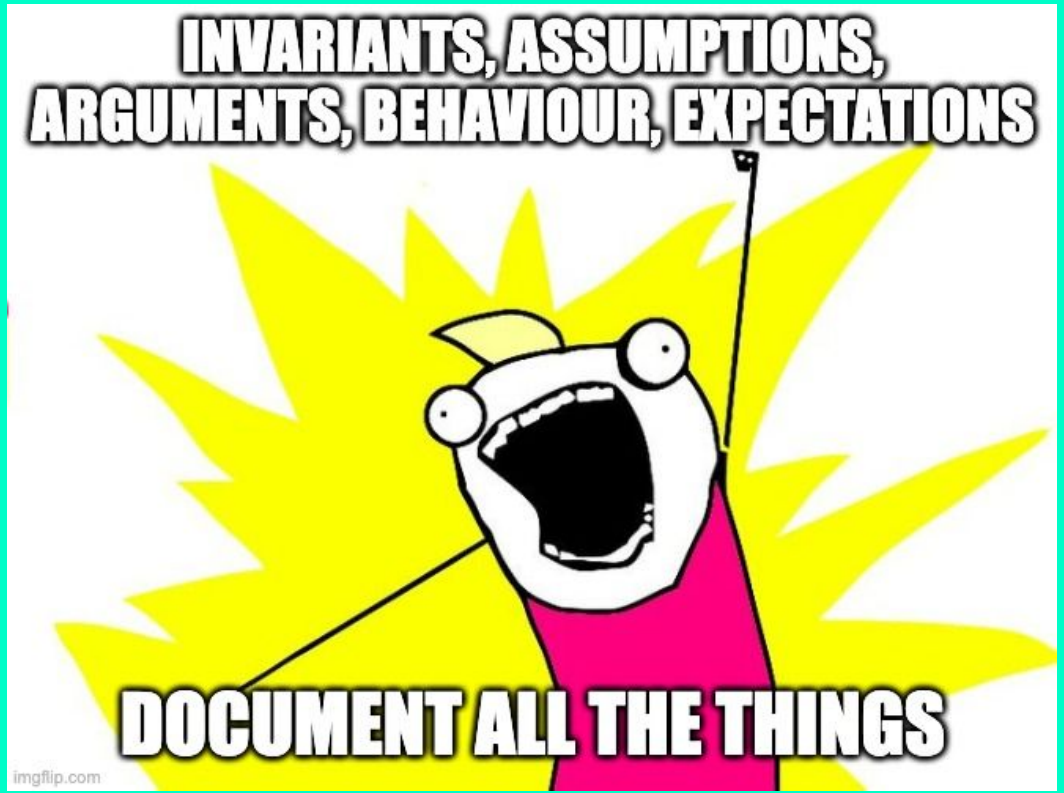
- killchain
- tricks / gadgets

For devs

Tip #11: Architect first, code later

Me trying to fix bug in my code
avoiding the other bugs to happen.

Tip #12: Document EVERYTHING



Tip #13: Simultaneously test and code

Tip #14: Automate as much as you can; focus on real things

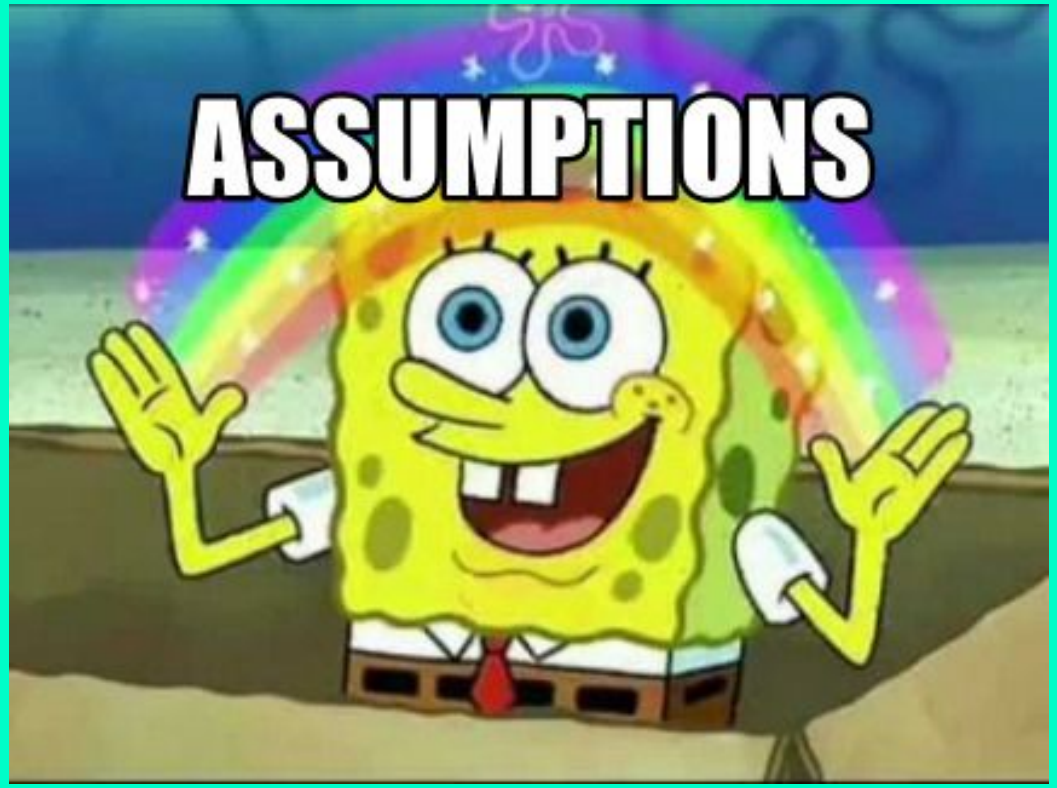
- 

Tip #15: Proactively care

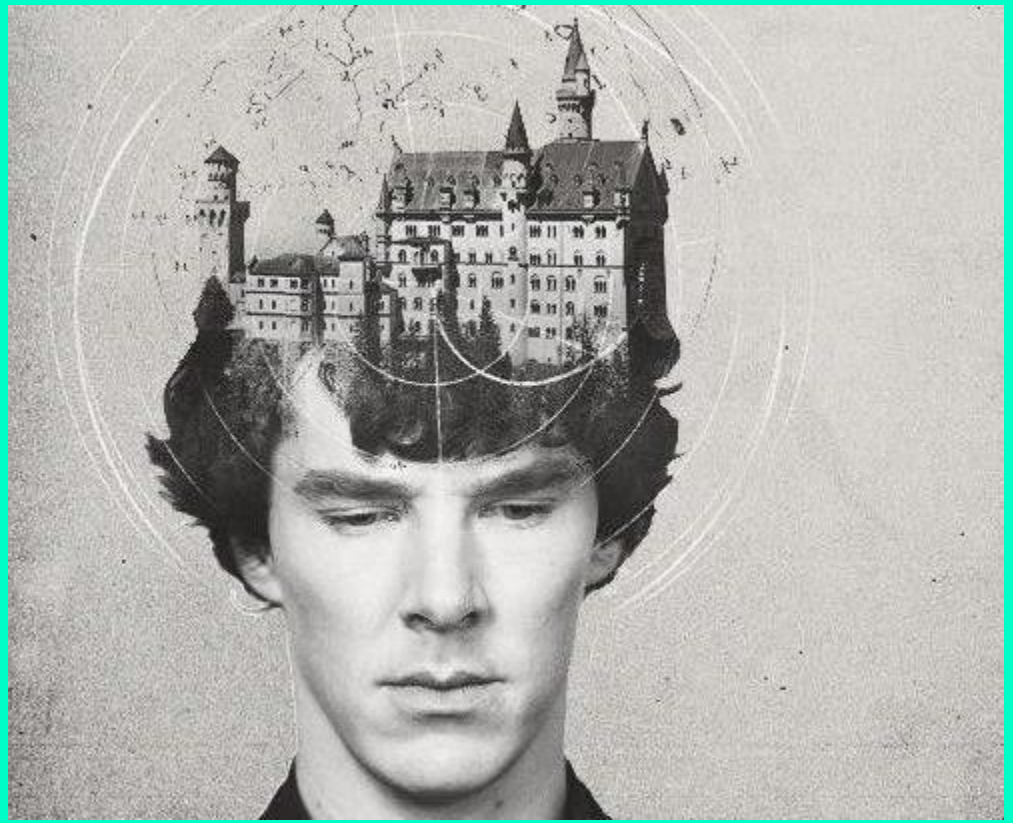
- Prevents bugs in the future
- Smaller code = bugs with less impact

For security researchers

Tip #16: Pre-emptively ask



Tip #17: Build a mental model of the codebase



Tip #18: learn your tools

kontrol

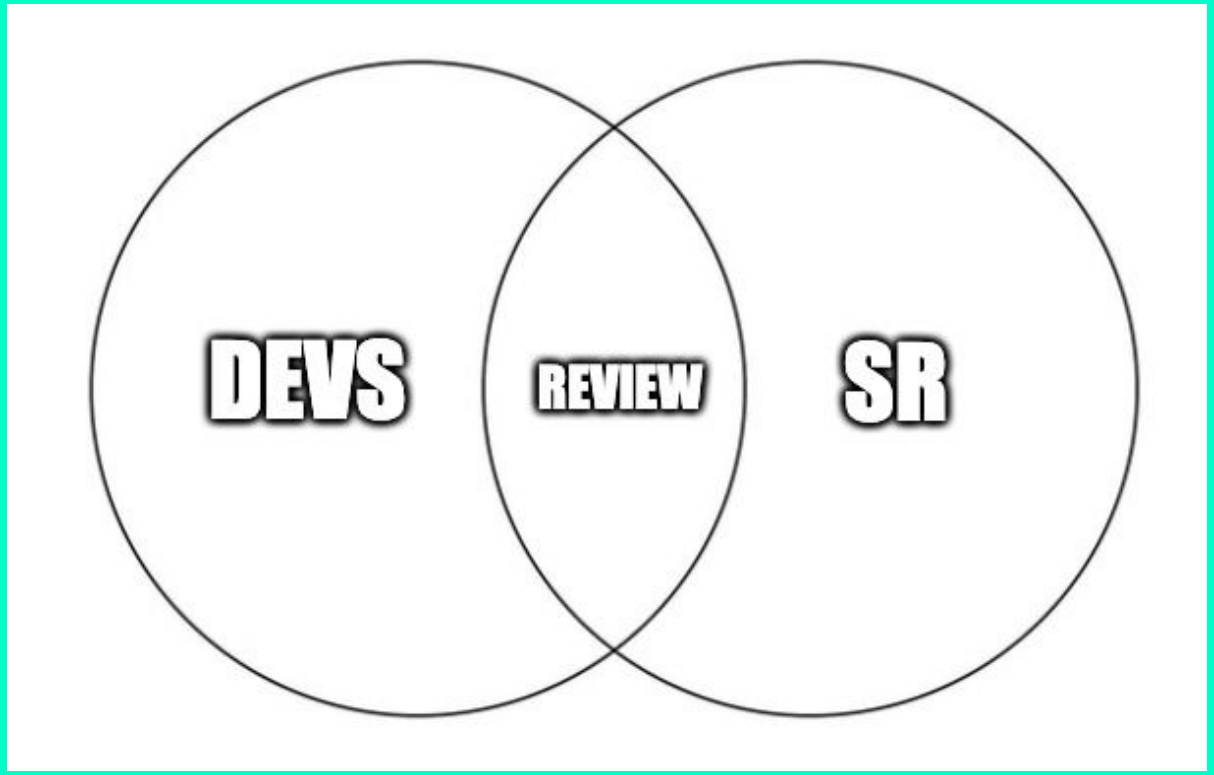
Tip #19: Learn how users use the system

How developers want users to use their system

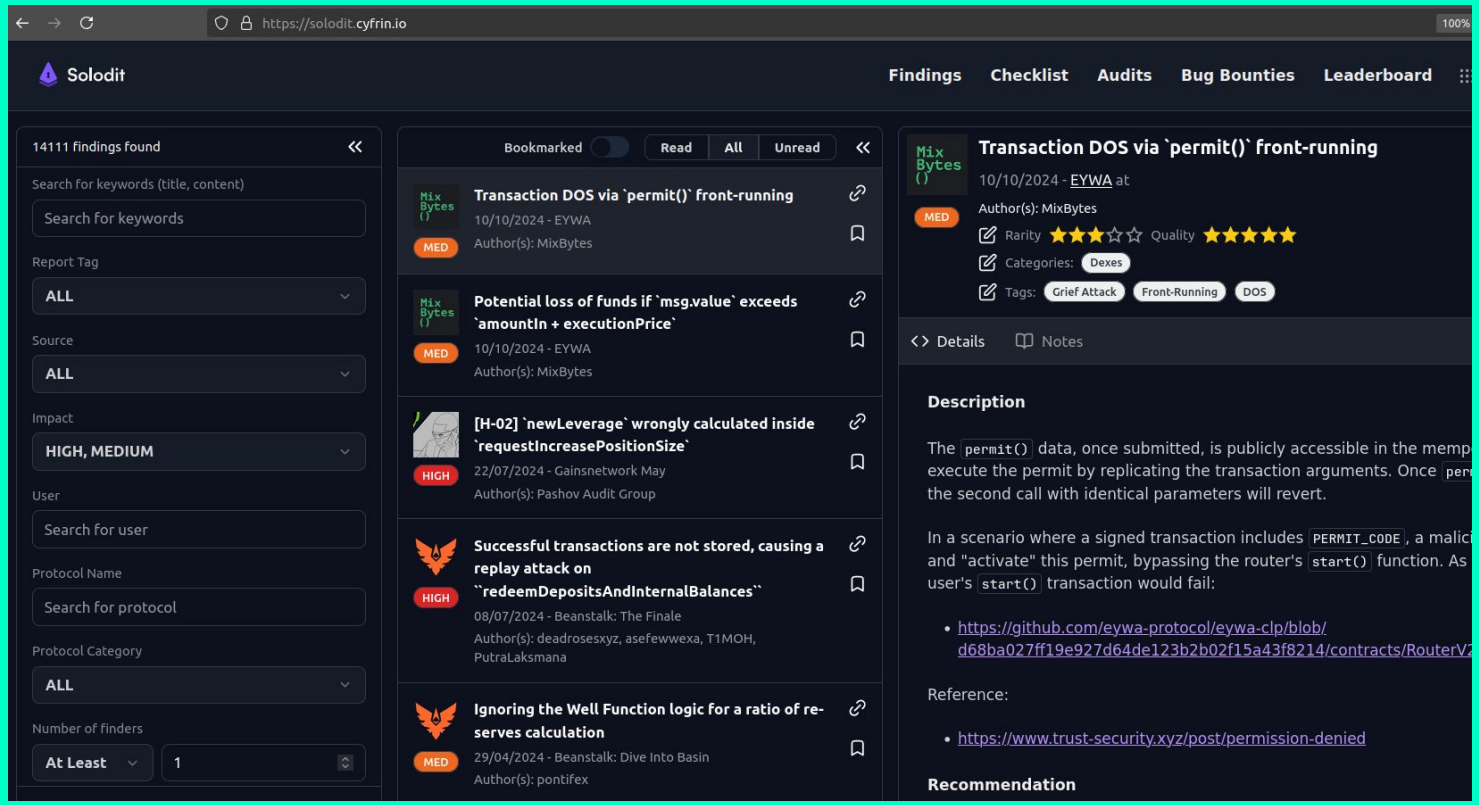
!=

How users are intended to use the system

Tip #20: lean on each others strengths



Tip #21 - Learn from the past



solodit.cyfrin.io

Tip #22 - Depth or Width?

In **depth** you can find special / rare issues → Pointless without **context**

MUST **zoom out!**

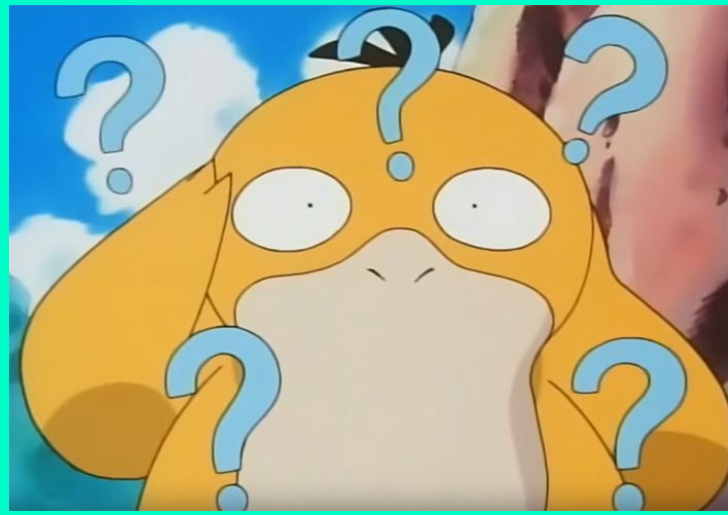
Understand expected use cases, relationships, cross-contract dynamics

Tip #23 - Beware of diminishing returns

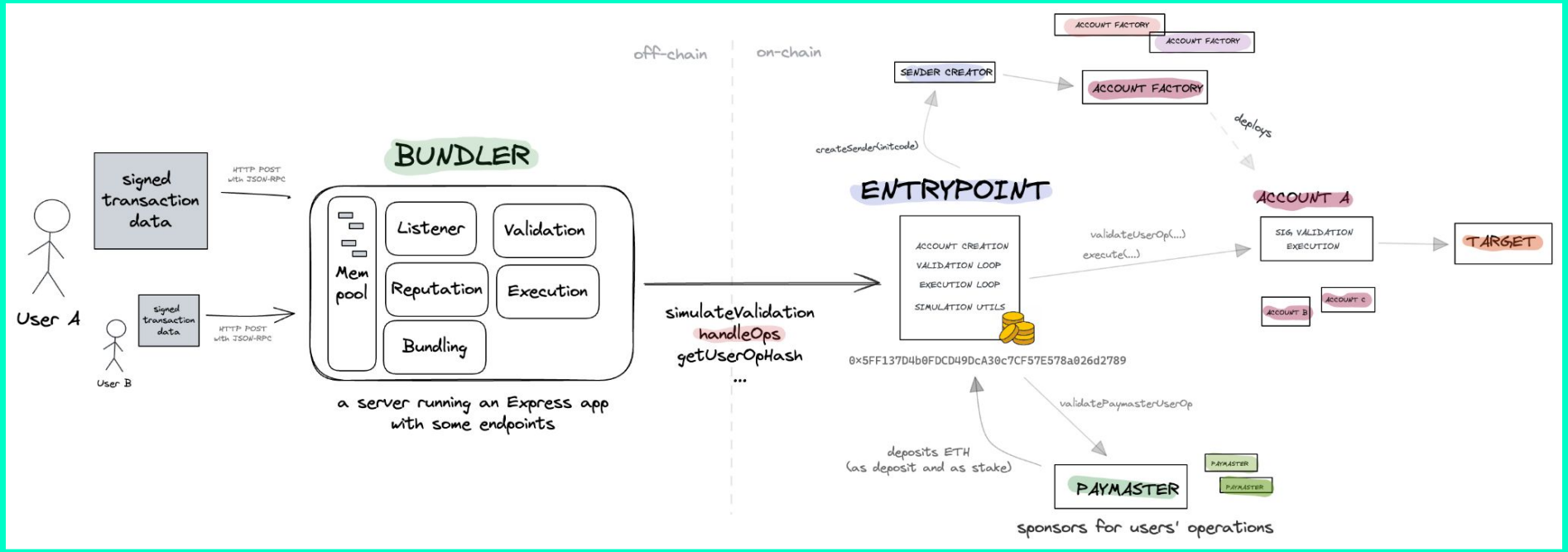
How long should I chase down this bug?

How much detail should I include in this report with PoC?

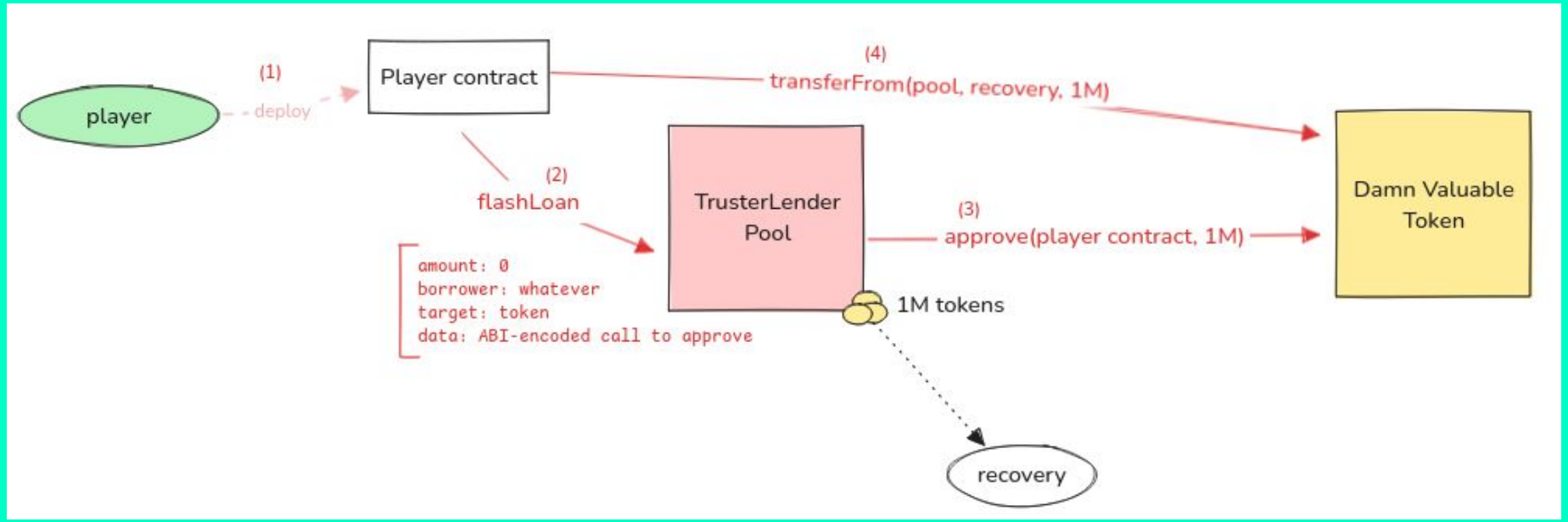
How much should I fight back in this endless escalation thread?



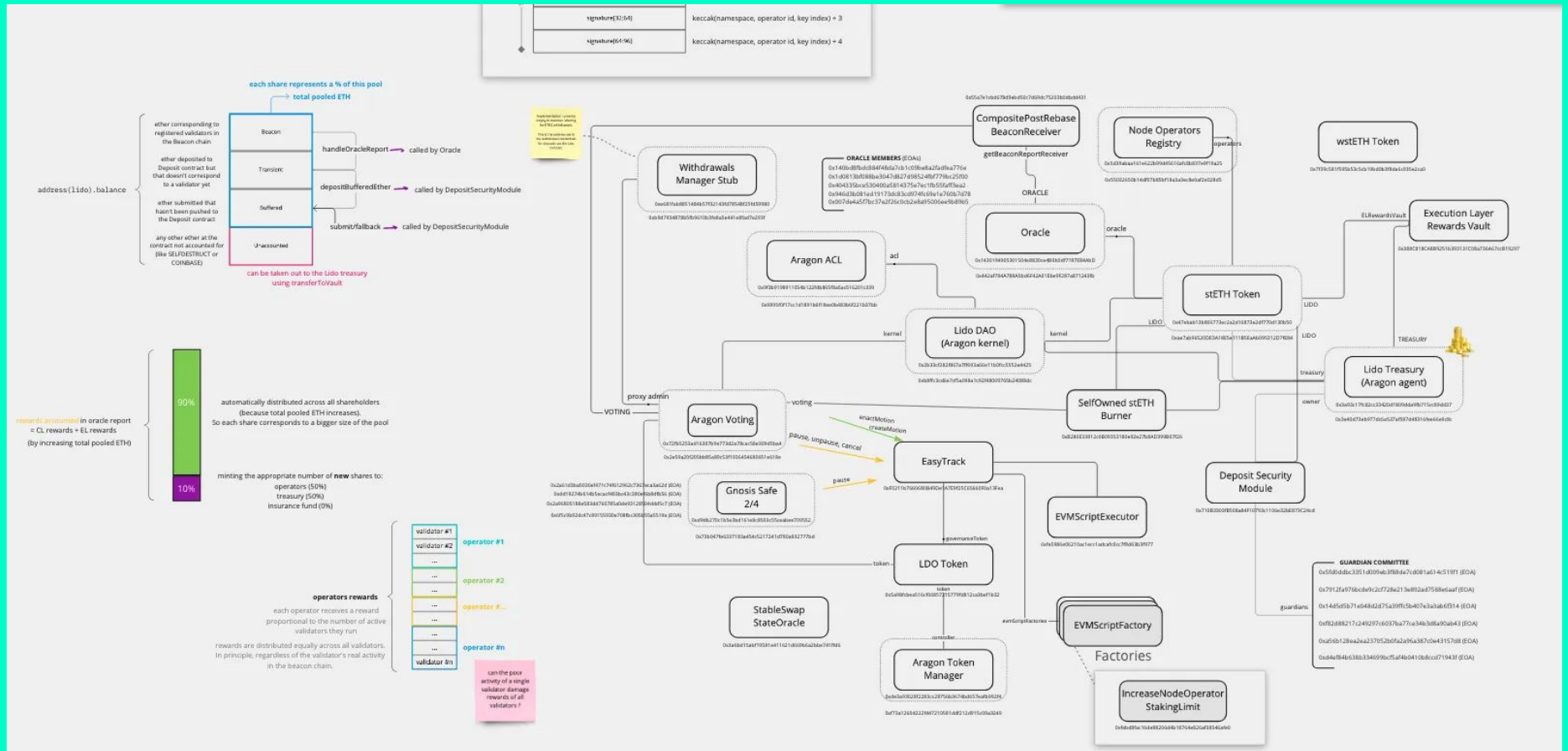
Tip #24 - Diagrams FTW



Tip #24 - Diagrams FTW



Tip #24 - Diagrams FTW



Tip #25 - Out of the comfort zone

What's most people going

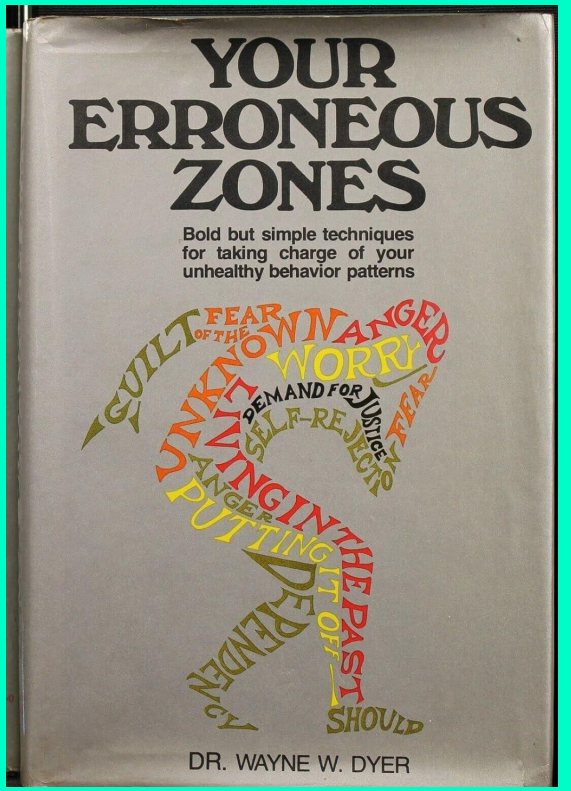
to review?

↓

Can I go somewhere else?

What skills do I need?

Can I learn during the review?



Tip #26 - WRITING FTW

Writing to think ☑

Writing to not forget ☑

Writing to communicate ☑

paulgraham.com/writes.html
paulgraham.com/words.html

Tip #27 - What's not tested?

Less tests, more chances for bugs.

Review, run and rework test suites to find holes.

Tip #28 - Choose your ground

- → **Auditing** is not bug hunting
- → **Bug hunting** is not contests
- → **Contests** are not auditing

Tip #29 - Whatever you do, BE PROfessional

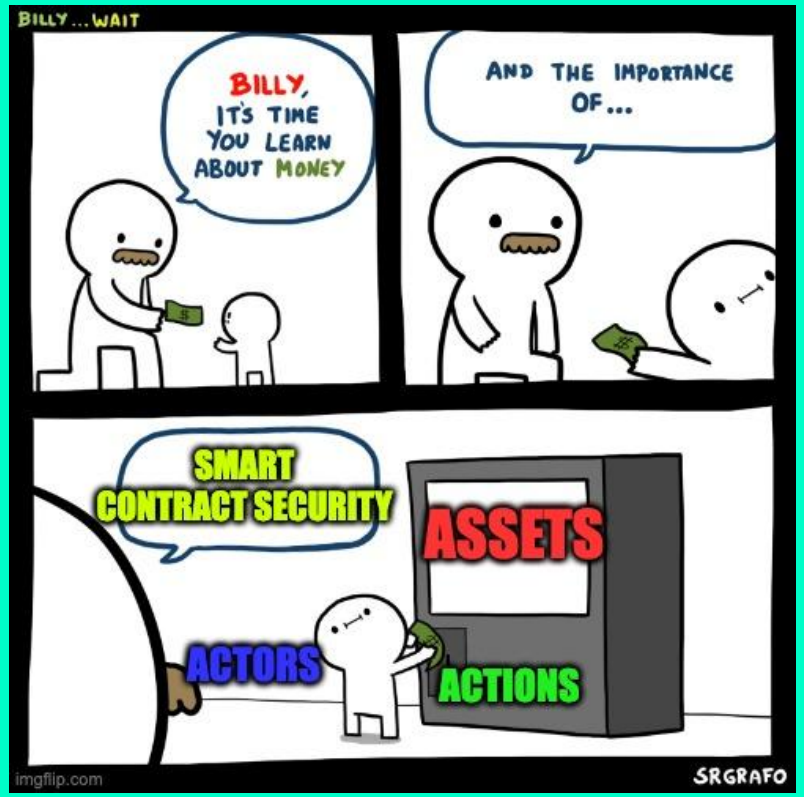
- Assertiveness
- Thoughtfulness
- Attention to detail
- Respect

and **please** avoid
bulls**t reports
on Twitter,
thanks

Tip #30 - PATIENCE

- Finding cool bugs **takes time**
- Reporting **takes time**
- Triaging **takes time**
- Judging **takes time**
- Escalations **take time**
- Payments **take time**

Tip #31: Assess - Assets, Actors & actions

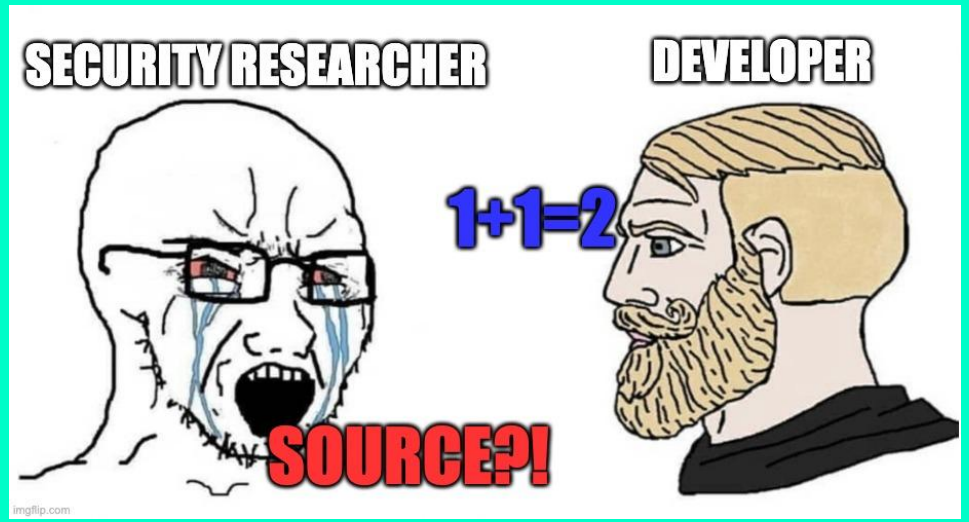


- Assets: What? Where?
- Actors: Who?
- Actions: What? When? How? Why?

Tip #32: Behavior - Expected vs unexpected

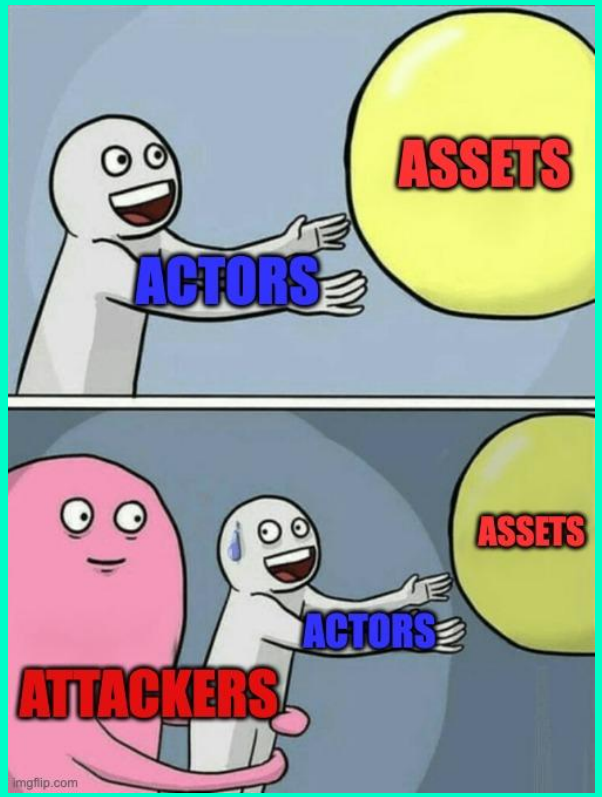
- Normally vs Anomaly
- Bug ⊂ Anomaly
- Bug << Vulnerability

Tip #33 Challenge: Assumptions



- Root of all Bugs!
- Question everything
- 5 W's: Who? What? Why? When? Where?

Tip #34: Discover - misuse scenarios

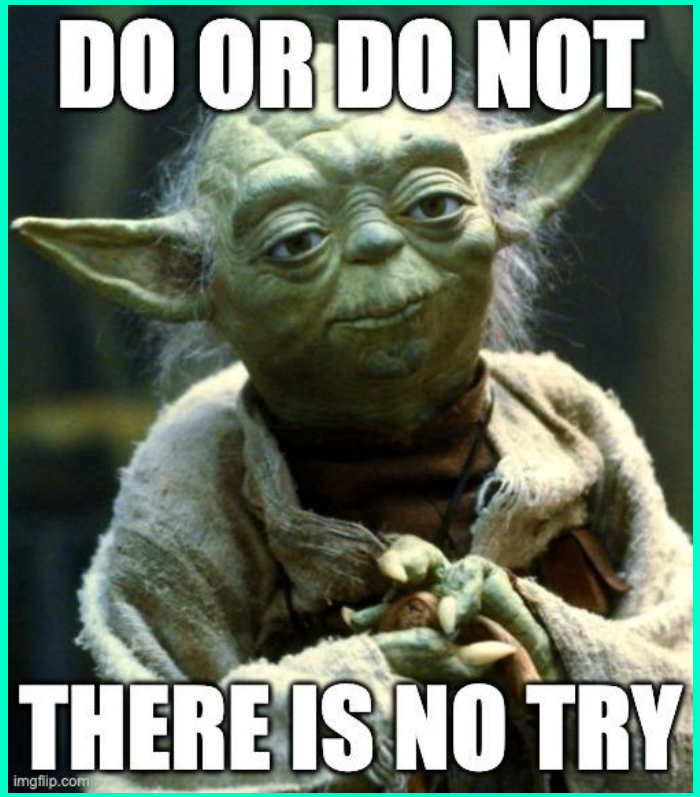


- Use vs Misuse
- Abuse ⊂ Misuse
- Attack surface

Tip #35: Evaluate - Severity

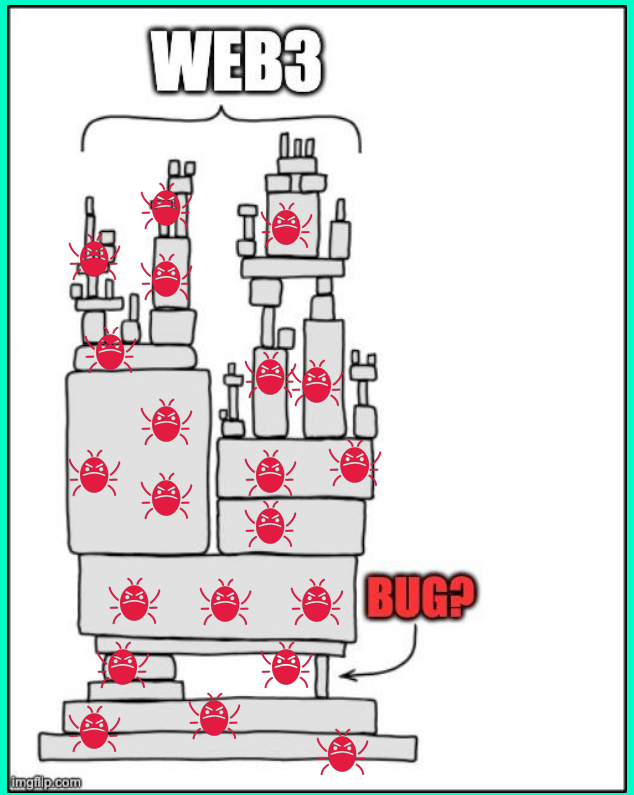
- Impact x Likelihood
- Impact: Loss/Lock, Temp/Perm, All/Few
- Likelihood: Always/Sometimes, Unconditional/Conditional

Tip #36: FinD - inconsistencies



- 🔏 Inconsistent checks
- 🔏 Inconsistent transitions
- 🔏 Inconsistent state

Tip #37: Grok every line



- Every line matters!
- Read between the lines
- Read beyond the lines

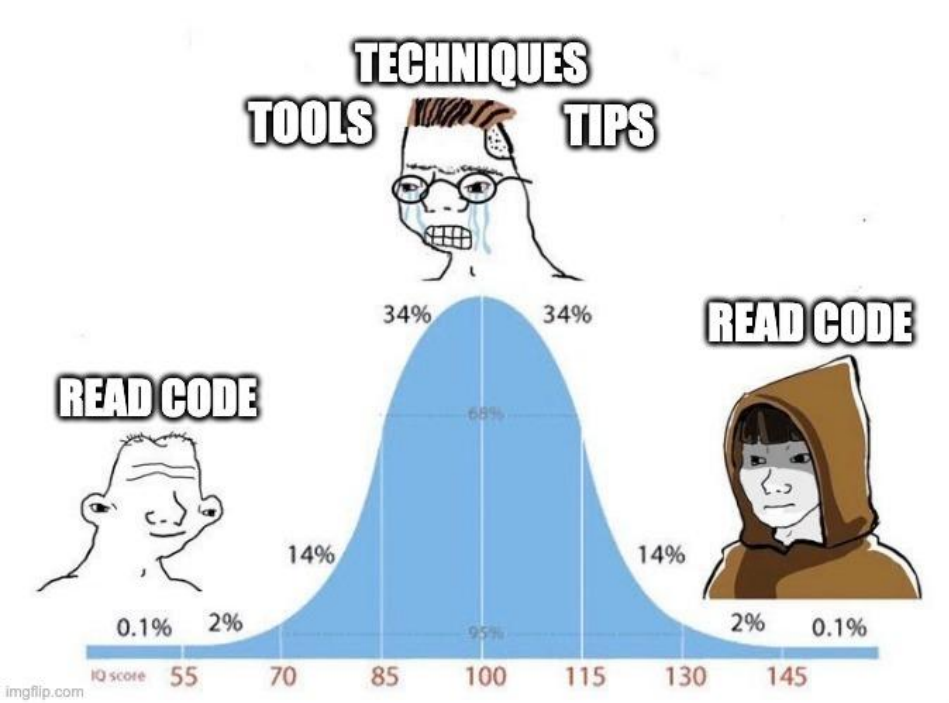
Tip #38: Hacker mentality

- Look: What's there & What's missing
- Think: Attacker
- Act: Whitehat

Tip #39: Inferred invariants

- Properties
- Spec/Doc/Comment/Comms
- Infer: Clues in code

Tip #40: Just ~~do~~ find it ✔



- No bug-free code
- Read code → Find bugs
- Prepare → Practice → Patience

Tip #41 (Devs)

- Specify -> Design -> Code -> Test
- Assets - Actors - Actions
- Security Mindset -> Shift Left
- Testing Properties/Invariants
- Frameworks & Extensions
- Find Your Tools/Techniques
- Use <> Abuse
- Adversarial Environment -> Defensive Programming
- Audits vs Contests vs Bug Bounties
- Code - Coverage - Communications

Tip #42 (SRS)

- Read Code
- Challenge Assumptions
- Documentation & Intuition
- Pre-conditions & Post-conditions
- Coverage & Leverage
- Scoping & Pacing
- Dive-in & Step-out
- Past -> Learn
- Prepare-Practice-Patience
- TLC: Code & Yourself

RACE

- Short (16 minutes) quiz
- 8 MCQs
- Password: TICKER-IS-ETH
- Submit by tomorrow
- Top 2: Win DSS’25 tickets


THANKS!