Achieve Pareto Principle in secure code review, or die trying

Sandro “guly” Zaccarini
End Summer Camp 2K20

rushing a secure code review

- on last day of a pentest and you've just found a git repo with Intranet source code?
- sales sold 2days when you asked for 8+?
- a coworker/friend asked you to audit a project offering beer+pizza?
- got some spare hours and want to hunt for bugs in an opensource app?

ANTANI METHOD
100% SUCCESS GUARANTEED

DISCLAIMER:
this is NOT a professional method, read this as a method for hobbist/enthusiast.
provided mindmap will hoply grow enough to become a pro reference, with the help of community

whoami

- Sandro "guly" Zaccarini
- read, write, broke, fix software
- haven't had a drink^W^W^W found a bug for 14 weeks

good old ~5 W

- the WHAT you are looking for
- the WHERE you are going to search for it
- the WHY you need it
- the HOW you'll going to find it

before to start, get in the mood

- EVERY software has bugs
  - some are exploitable, some will become
- NO software has been reviewed enough
  - new features means new interaction, refactory, legacy support, often overlooked simple stuff
- NO vendor is big enough to have everything properly reviewed
  - the bigger is, the wider is the codebase to audit

think about Google P0: does they make google bug free?
also Checkpoint has a fantastic research team, but they still suffer for bugs
Let alone other big vendors like Cisco, Fortinet, just to name two...

before to start, get in the mood

it's surely true that more skilled people already worked on that code, but they're human and human fail.
it's also true that maybe they rushed too, so there is surely plenty of bugs waiting for you.

and don't forget that everyone has his own experience, that means a different approach to a problem, which makes huge difference when in a rush

the WORST thing you can do is to start auditing thinking that more talented/skilled people already did it and there is nothing left

before to start, get in the mood

you're not reading a book!!!
fit in your head that code audit != code read

the BEST thing you can do is to start knowing that you are not reading code, you are auditing it

the WHAT

AKA LOW HANGING FRUIT

- auth bypass
- lateral movement
- privilege escalation
- code execution

the WHERE

WHERE THERE ARE MORE PROBABILITIES OF LOOT

- ideally: everywhere there is user input
- in a rush: login, registration, profile, messages, ...

STAY OPEN, DON'T FOCUS TOO MUCH ON A SINGLE GOAL

the WHY

- you are going to look for bugs for a reason:
    - impersonate admin or another user
    - bypass login at all
    - modify a configuration
    - dump db and decrypt data
    - upload a webshell
    - straight RCE

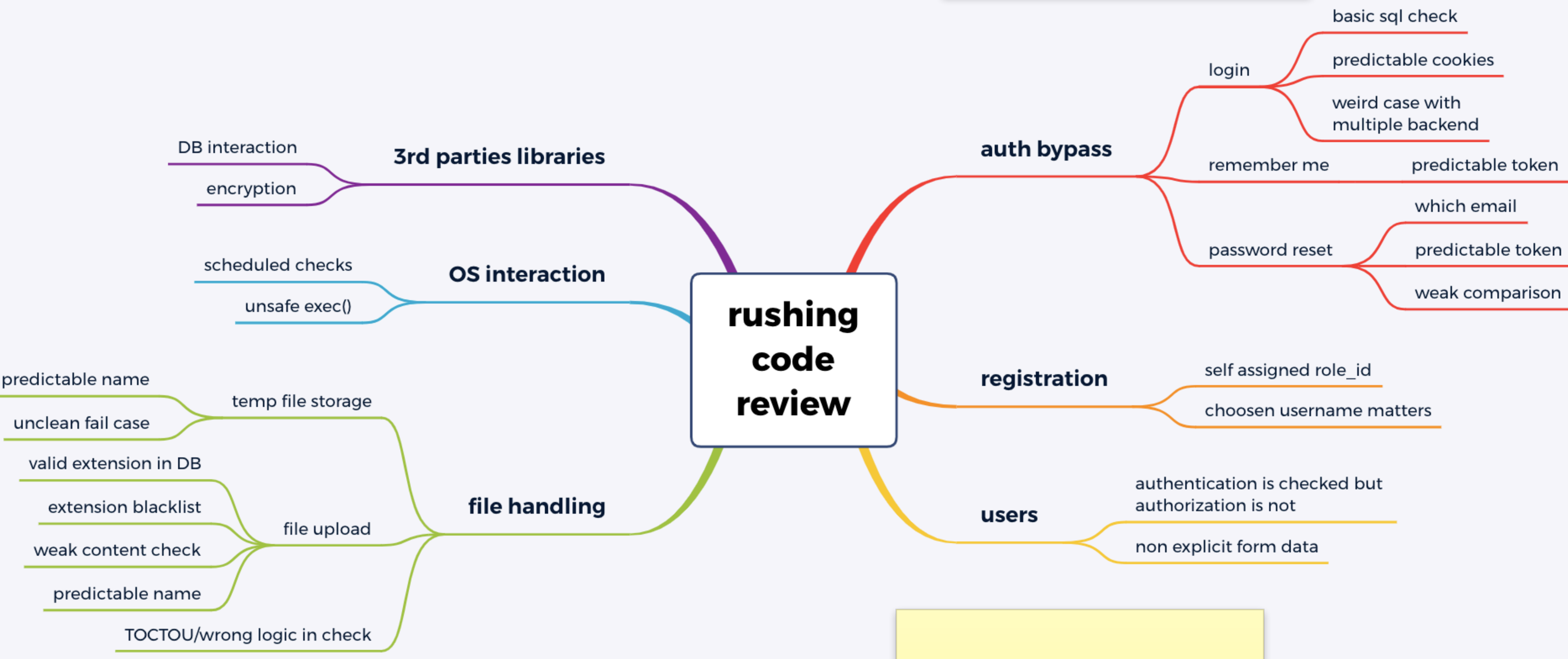
actually: start with a goal, but don't fall for a tunnel vision

Code audit doesn't start at page 1 the HOW

don't read this as a step-by-step but as a very high overview of what you can do

- if you can, do a quick crawl on the webapp and find the juicy function
- also have a quick look at the code base: structure, trees, exposed pages
- audit the code starting where you *think* you could find the loot
- grep -A 5 -B 5 / -L is our nursery's best first friend
- take note about everything, you don't know what you will use later

MINDMAP IS FAR FROM COMPLETE, REMEMBER THAT WE ARE NOW 80/20 bear in mind that often we use the short-term "auth" in a wrong way: it can represent both authentication and authorization, CHECK what's in place



mindmap will be published on https://github.com/theguly sooner or later.
this will hoply grow to become more comprehensive and perhaps a detailed reference.
i'd appreciate if you have time to give some feedback

real life 01

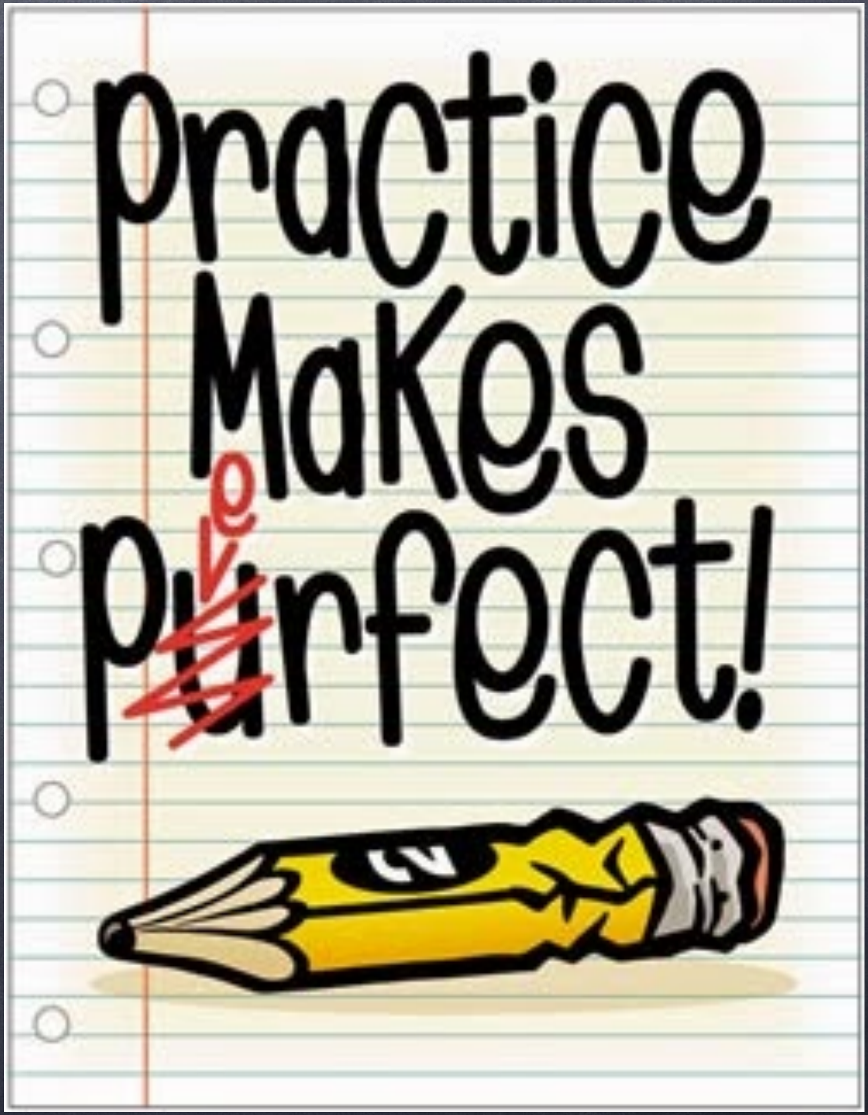
- supposed admin-only page not linked to users but exposed without authorization check
- unauthenticated SQL Injection on that page, won't let you to bypass login but let you to dump the DB
- webapp has a custom encrypting algorithm (suppose they actually need to decrypt information stored to do some "ansible" stuff)
- GOAL: dump the DB, decrypt everything, gain access to more targets

real life 02

- login test all backend for a given user, primary backend is AD, secondary is a local SQL
- unauthenticated SQL Injection, won't let you to bypass login
- the SQL DB driver let you to stack queries, so you can run UPDATE or INSERT
- GOAL: insert a new "admin" user to local SQL and login

closing

- we *should've* achieve the 80% in 20', but lost something trying
- keep in mind that the remaining 20% will take you faaaar more than 80'



Questions?

VOTE AS BEST SHORT-TALK OF THE YEAR!

ASK FOR A LONG TALK OR A WORKSHOP AT ESC 2021

Acta est fabula, plaudite!

MAKE CODE AUDIT GREAT AGAIN

feedback please!
guly@guly.org
@theguly

hold for a future workshop/long talk

- reversing, decompiling, debugging
- grep on steroids
- language specific issue
- home brewed crypto
- OSINT (github issue, commit with partial fix)
- some hand's on based on proposed mindmap

thanks

- ESC staff
- you, the crowd