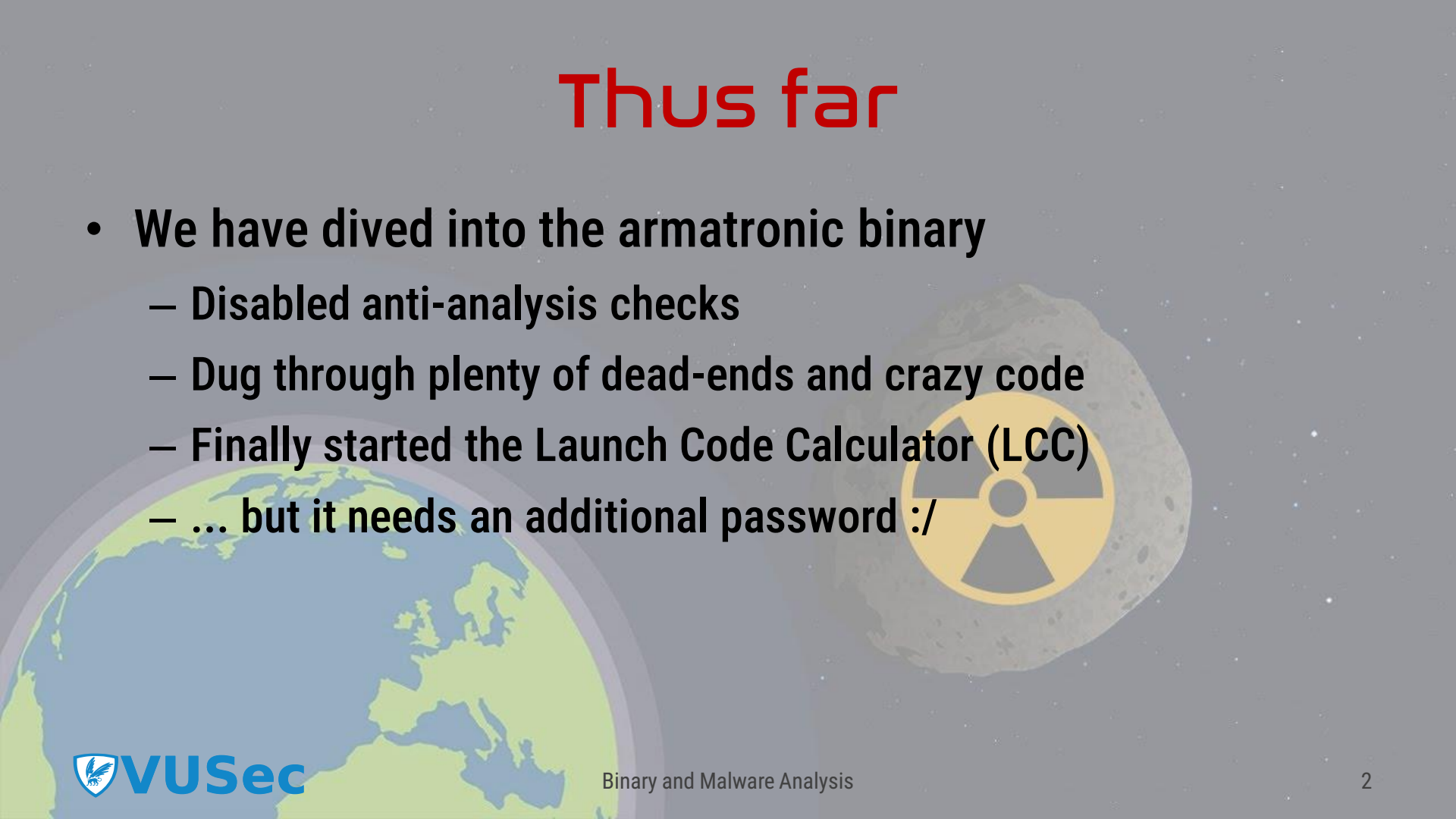
Challenge 2

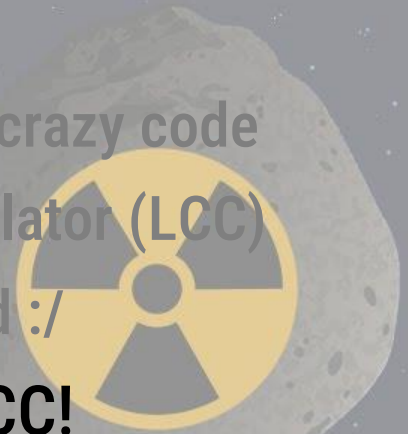
Thus far

- We have dived into the armatronic binary
  - Disabled anti-analysis checks
  - Dug through plenty of dead-ends and crazy code
  - Finally started the Launch Code Calculator (LCC)
  - … but it needs an additional password :/



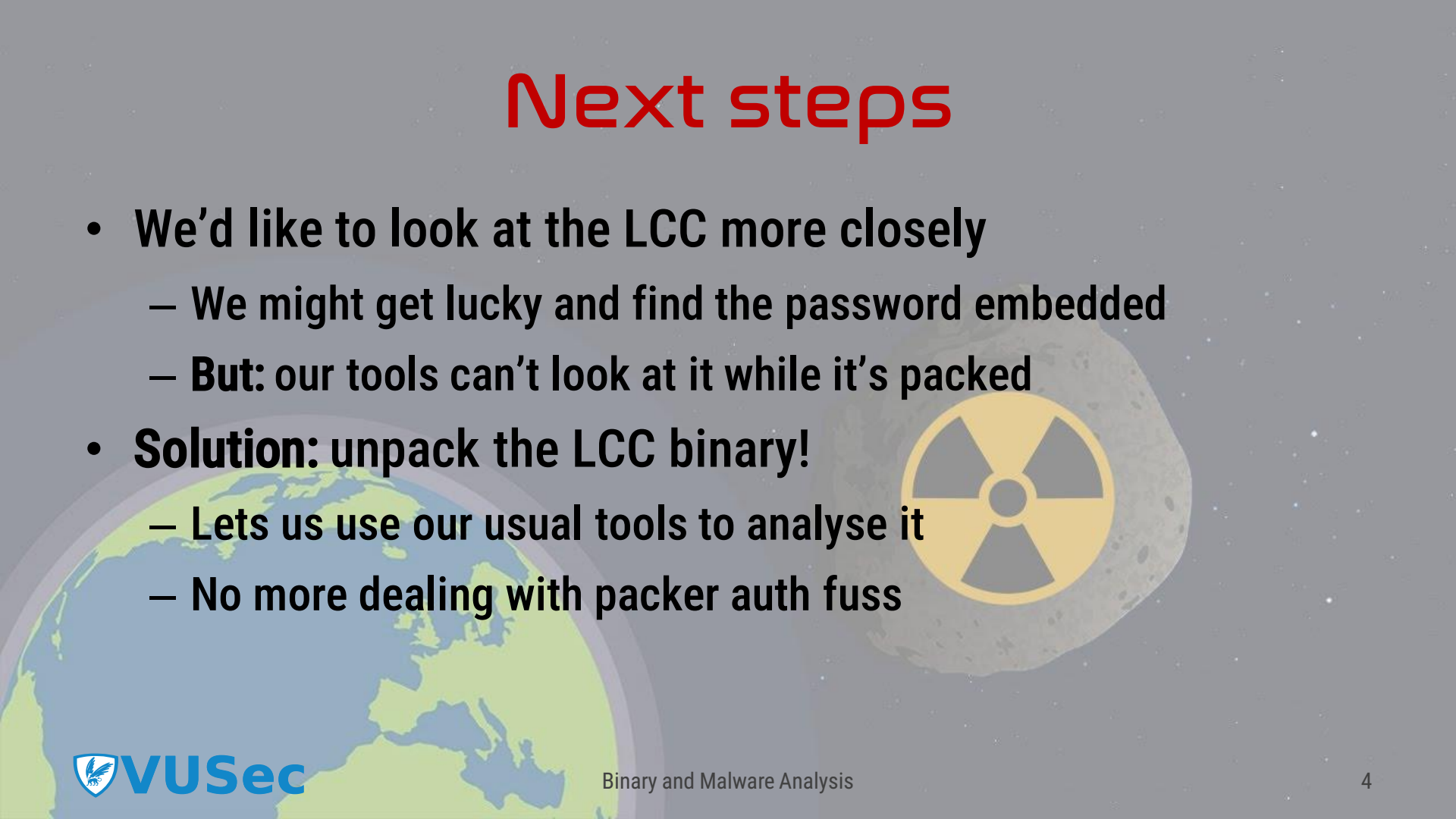
Thus far

- We have dived into the armatronic binary
  - Disabled anti-analysis checks
  - Dug through plenty of dead-ends and crazy code
  - Finally started the Launch Code Calculator (LCC)
  - … but it needs an additional password :/
- **armatronic** is just a **packer** for the LCC!



Next steps

- We'd like to look at the LCC more closely
  - We might get lucky and find the password embedded
  - **But:** our tools can't look at it while it's packed
- **Solution:** unpack the LCC binary!
  - Lets us use our usual tools to analyse it
  - No more dealing with packer auth fuss

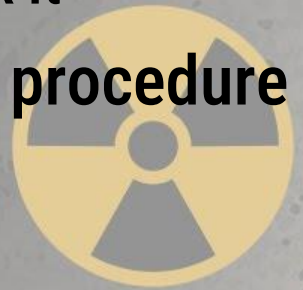


Tasks – urgent

- Dump the unpacked binary from packer memory
  - Determine right time and memory locations to dump
  - Dump binary & assemble ELF file
  - Ensure ELF is readable by standard tools (e.g. readelf)
  - Ensure ELF runs
- Analyse dumped ELF to find crunching password
- Use password to get the launch code & save Earth!

Tasks – nice to have

- Ensure ELF behaves the same as when packed
  - The LCC might notice it's running unpacked
  - Find out how it determines that and fix it
- Script dumping, assembling & fixing procedure



In detail

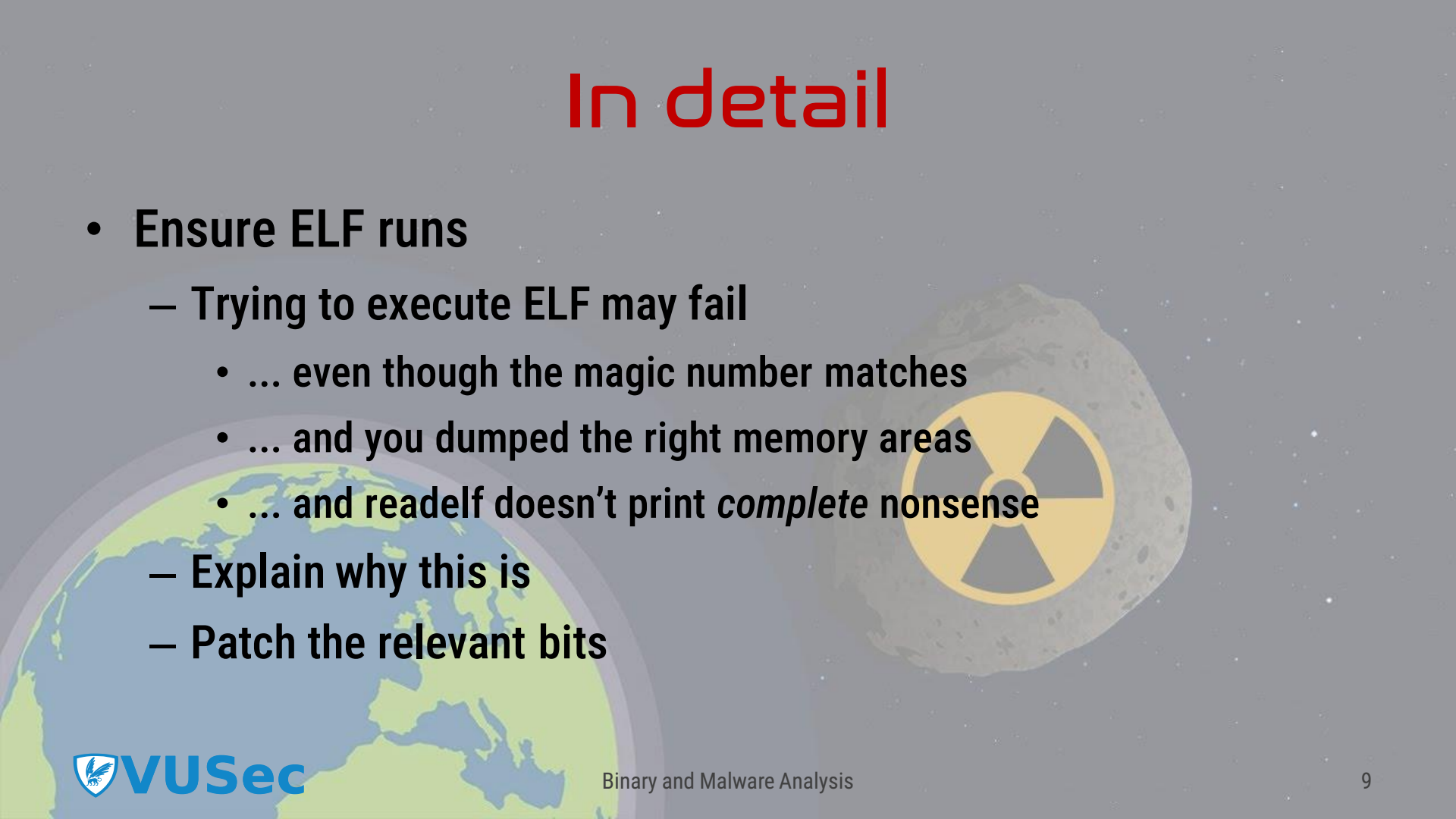
- Determine the right time and memory to dump
  - When is the right time to dump? Argue why precisely then and not some other time.
  - Which memory address(es) should you dump from? How much should you dump? How do you find this out?
- Dump binary into ELF
  - Dumping should be OK once you've figured out the above
  - Describe how you assembled the ELF from dumped memory

In detail

- Ensure ELF is readable by standard tools
  - readelf, file, etc. may display complete and utter nonsense
    - ... even though the ELF magic number matches
    - ... and you ostensibly dumped the right memory areas
  - Explain why this is
  - Patch the relevant bits

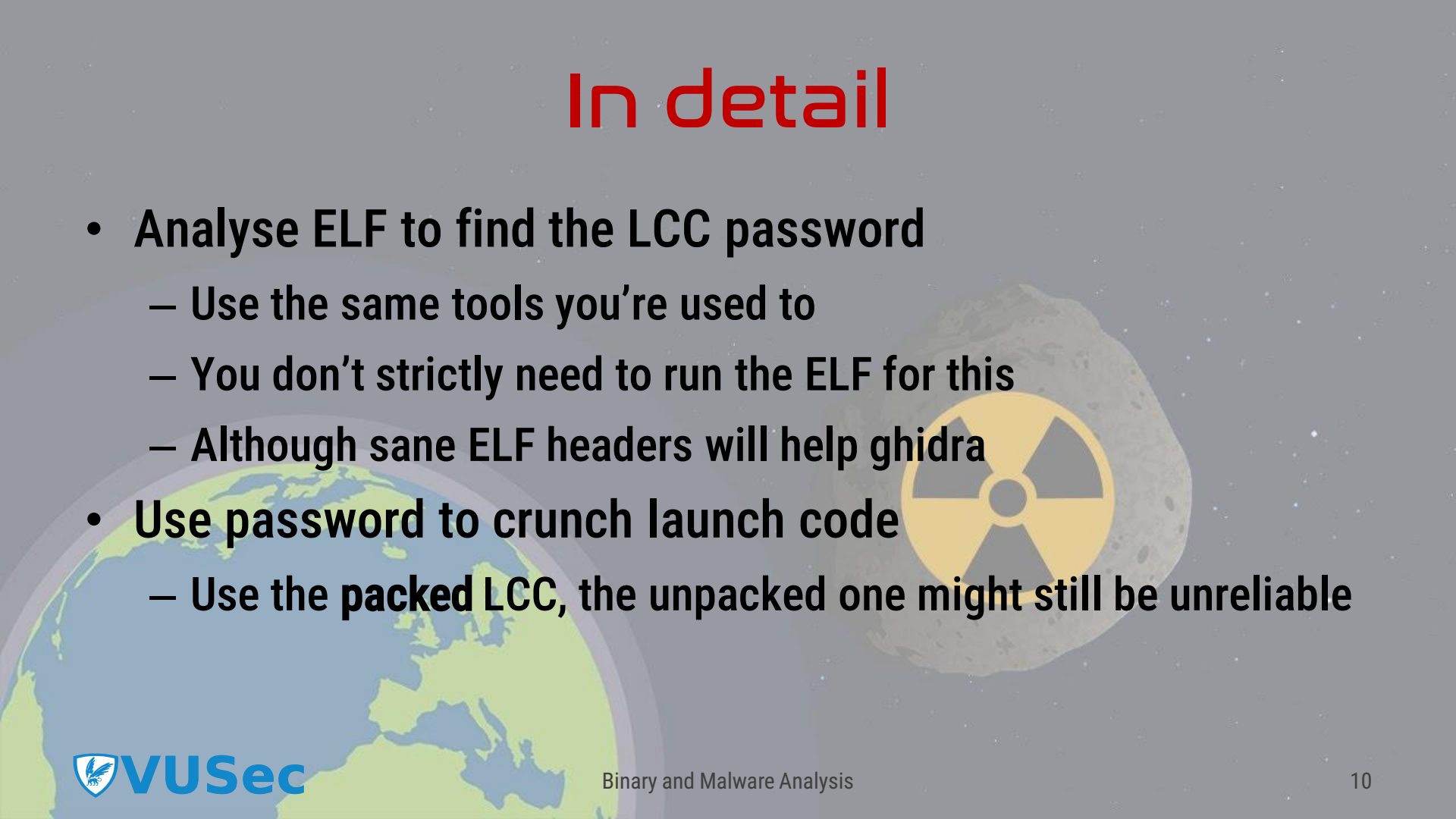
In detail

- Ensure ELF runs
  - Trying to execute ELF may fail
    - ... even though the magic number matches
    - ... and you dumped the right memory areas
    - ... and readelf doesn't print *complete* nonsense
  - Explain why this is
  - Patch the relevant bits



In detail

- Analyse ELF to find the LCC password
  - Use the same tools you're used to
  - You don't strictly need to run the ELF for this
  - Although sane ELF headers will help ghidra
- Use password to crunch launch code
  - Use the **packed** LCC, the unpacked one might still be unreliable



In detail

- Ensure ELF behaves the same as packed
  - Look closer at the dumped LCC, it might not behave the same
    - ... even though you dumped the right memory areas
    - ... and it runs fine, apparently
  - Explain how the binary can tell the difference.
  - And how exactly it behaves differently when unpacked.
  - Give at least two examples of where such a check is made.
  - Patch the code so it thinks it's running in the packer.
    - **IMPORTANT**: do this before main() is called for full marks

In detail

- Script dumping, assembling & fixing up ELF
  - Use whatever tools you like (e.g., raw gdb scripting, python, …)
  - Do this **at runtime** – automate your steps from before

Grading

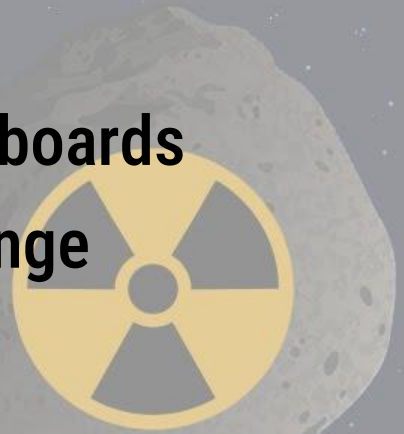
- Dumping & assembling ELF → max. 4 points
  - Determine right time to dump → max. 1 point
  - Determine memory location(s) to dump → max. 1 point
  - Dump memory and assemble ELF file → max. 1 point
  - Ensure ELF runs & readelf output is sane → max. 1 point (0.5 pts each)
- Find password & crunch the launch code! → max. 1 point
- Anti-unpacking measures → max. 3 points
  - Explain anti-unpacking measure and its effects → max. 2 points
  - Patch ELF to fool it into thinking it's packed → max. 1 point (0.5 pts if not before main)
- Script dumping, assembling & fixing up ELF → max. 2 points

Bonus

- Speed bonus for the first 5 students with correct solution.
  - 50, 40, 30, 20 & 10 pts, respectively
- +50 pts for reverse-engineering the packing format
  - Put your detailed explanation in the README
- +20 pts for **statically** extracting the packed LCC
  - Do this scripted and *without* running the packer binary
- +30 pts for **replacing** the LCC with **your own custom binary**
  - Do this *without* replacing packer code

BANA Leaderboard

- Ranked list of our best 1337h4xx0rs
- SCORE = GRADE * 100 + BONUS
- Per-challenge & cummulative leaderboards
- Post & update after grading a challenge

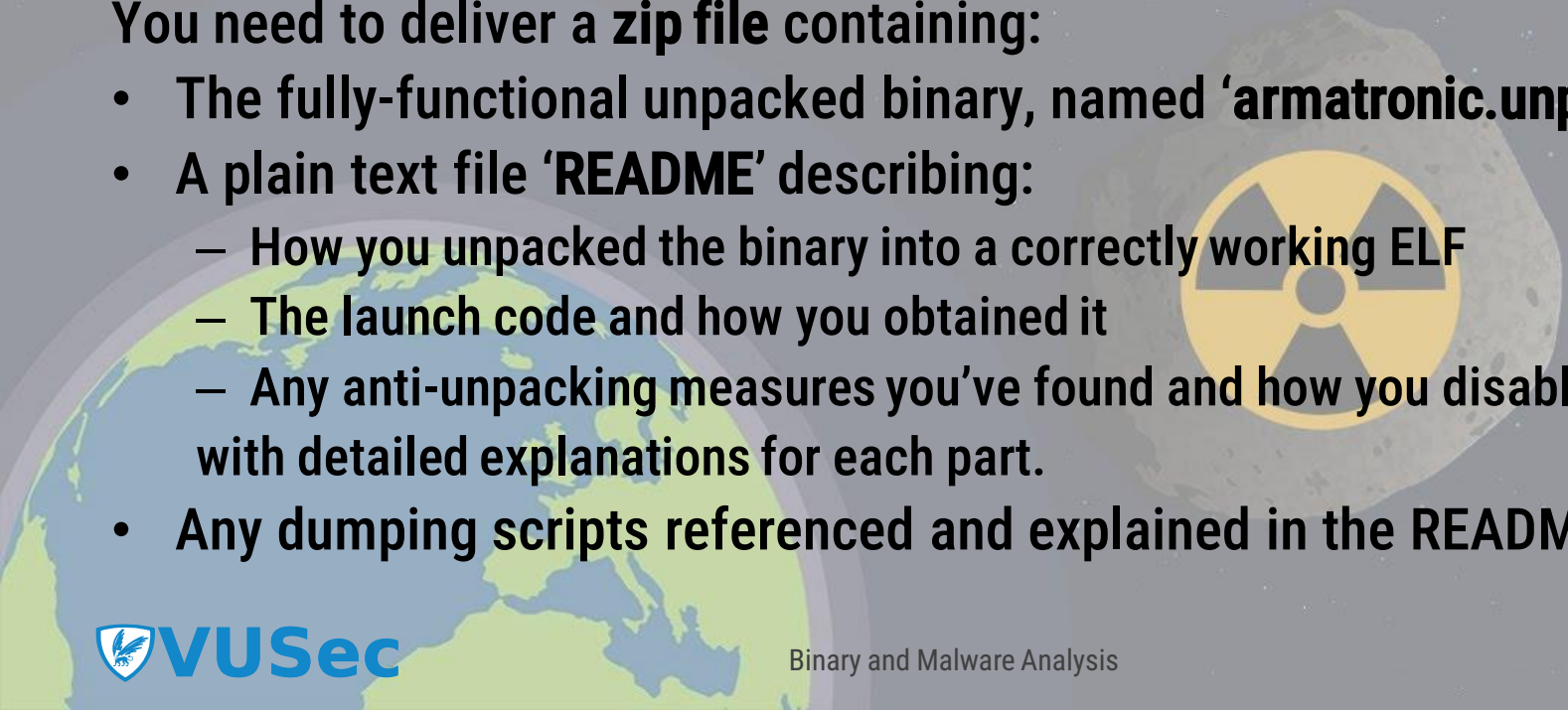


Submission Guidelines

You need to deliver a **zip file** containing:

- The fully-functional unpacked binary, named '**armatronic.unpacked**'.
- A plain text file '**README**' describing:
  - How you unpacked the binary into a correctly working ELF
  - The launch code and how you obtained it
  - Any anti-unpacking measures you've found and how you disabled them

  with detailed explanations for each part.
- Any dumping scripts referenced and explained in the README.



Submission Guidelines

- Submission will be through **Canvas**.
- **Deadline: Tuesday, 21st April 2020, 23:59 CEST**
- Delay penalties: 1pt/24h delayed

Good luck!