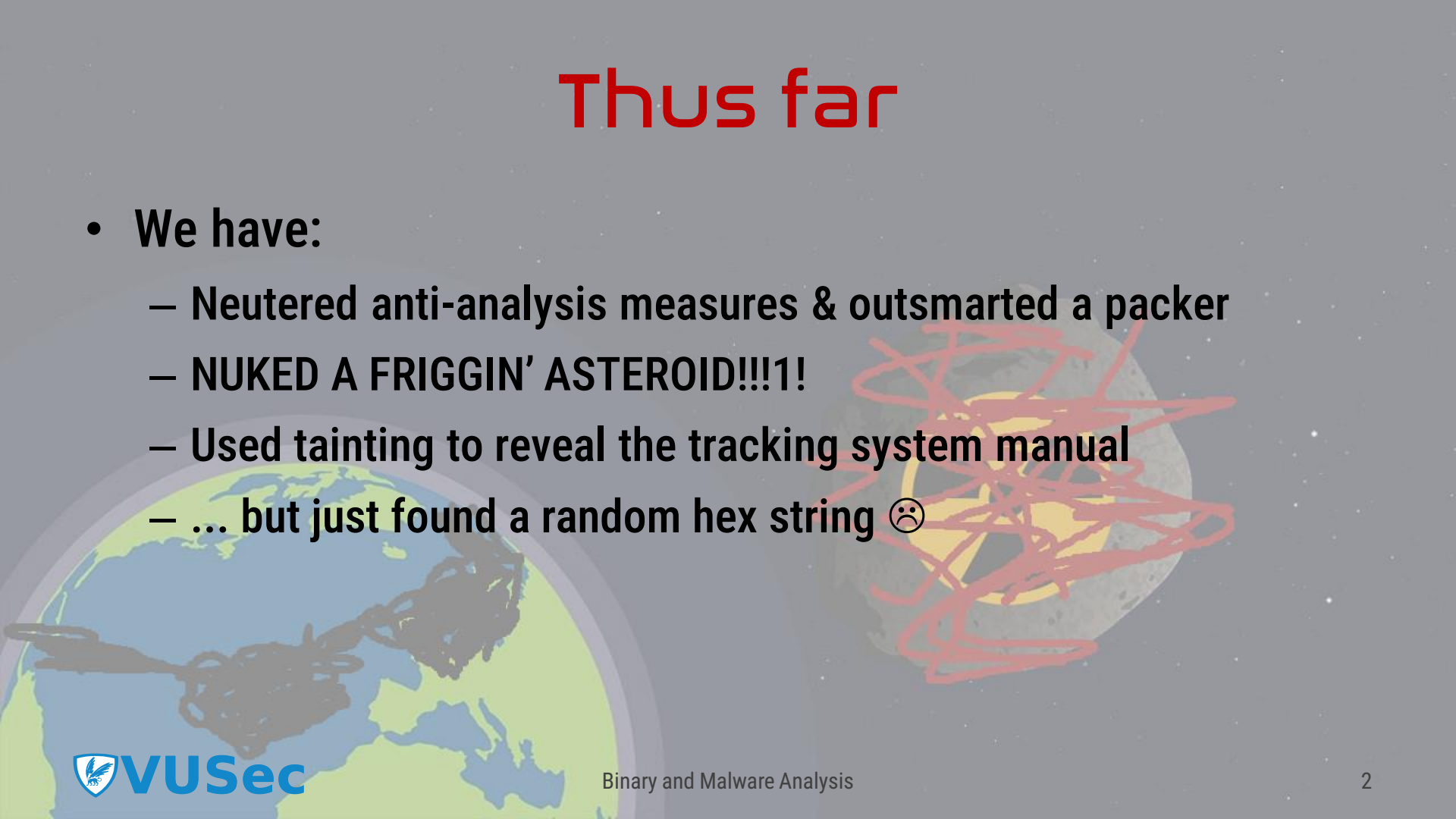
Challenge 5

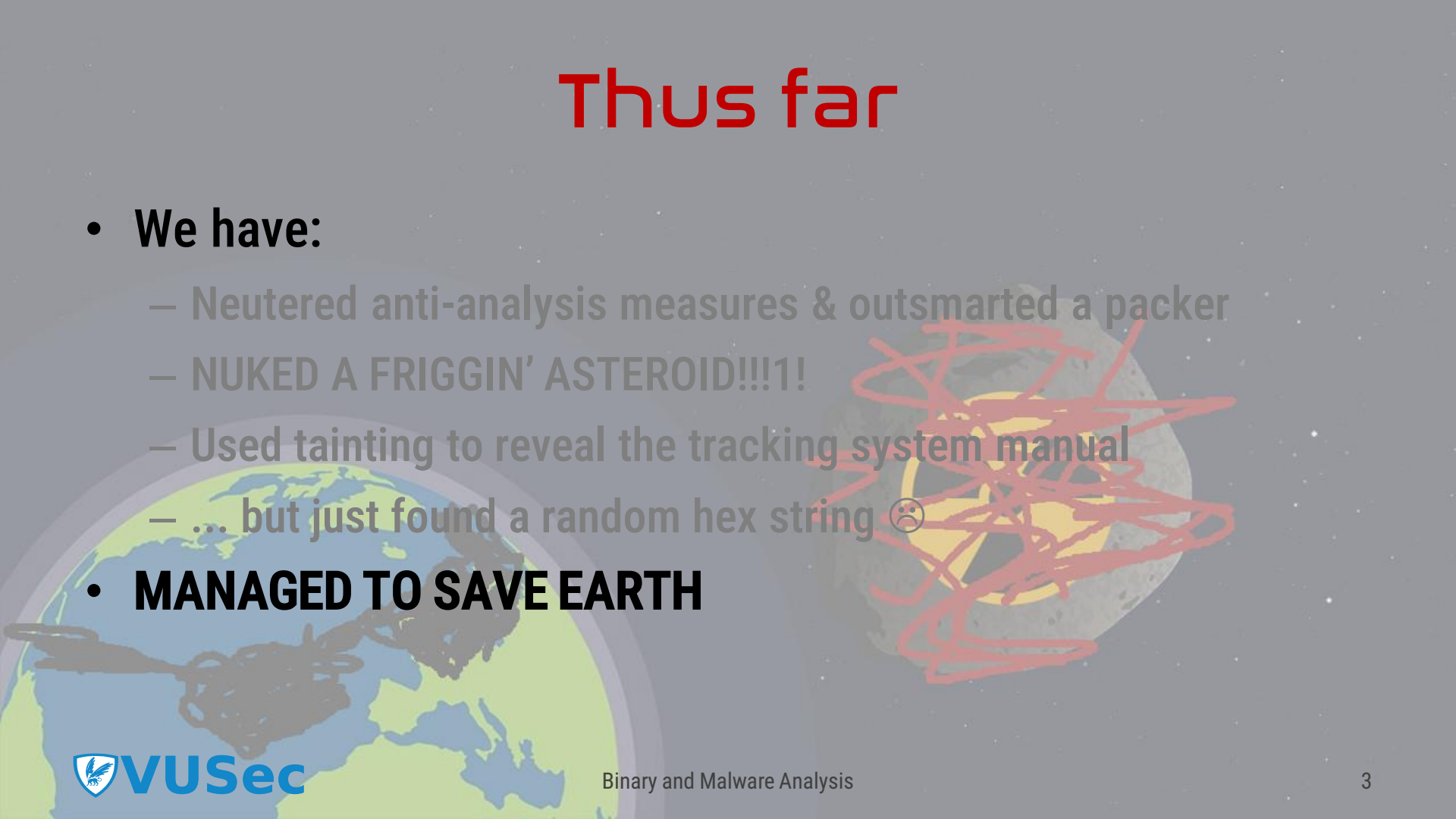
Thus far

- We have:
  - Neutered anti-analysis measures & outsmarted a packer
  - NUKED A FRIGGIN' ASTEROID!!!1!
  - Used tainting to reveal the tracking system manual
  - ... but just found a random hex string ☹



Thus far

- **We have:**
  - Neutered anti-analysis measures & outsmarted a packer
  - NUKED A FRIGGIN' ASTEROID!!!1!
  - Used tainting to reveal the tracking system manual
  - … but just found a random hex string ☹
- **MANAGED TO SAVE EARTH**



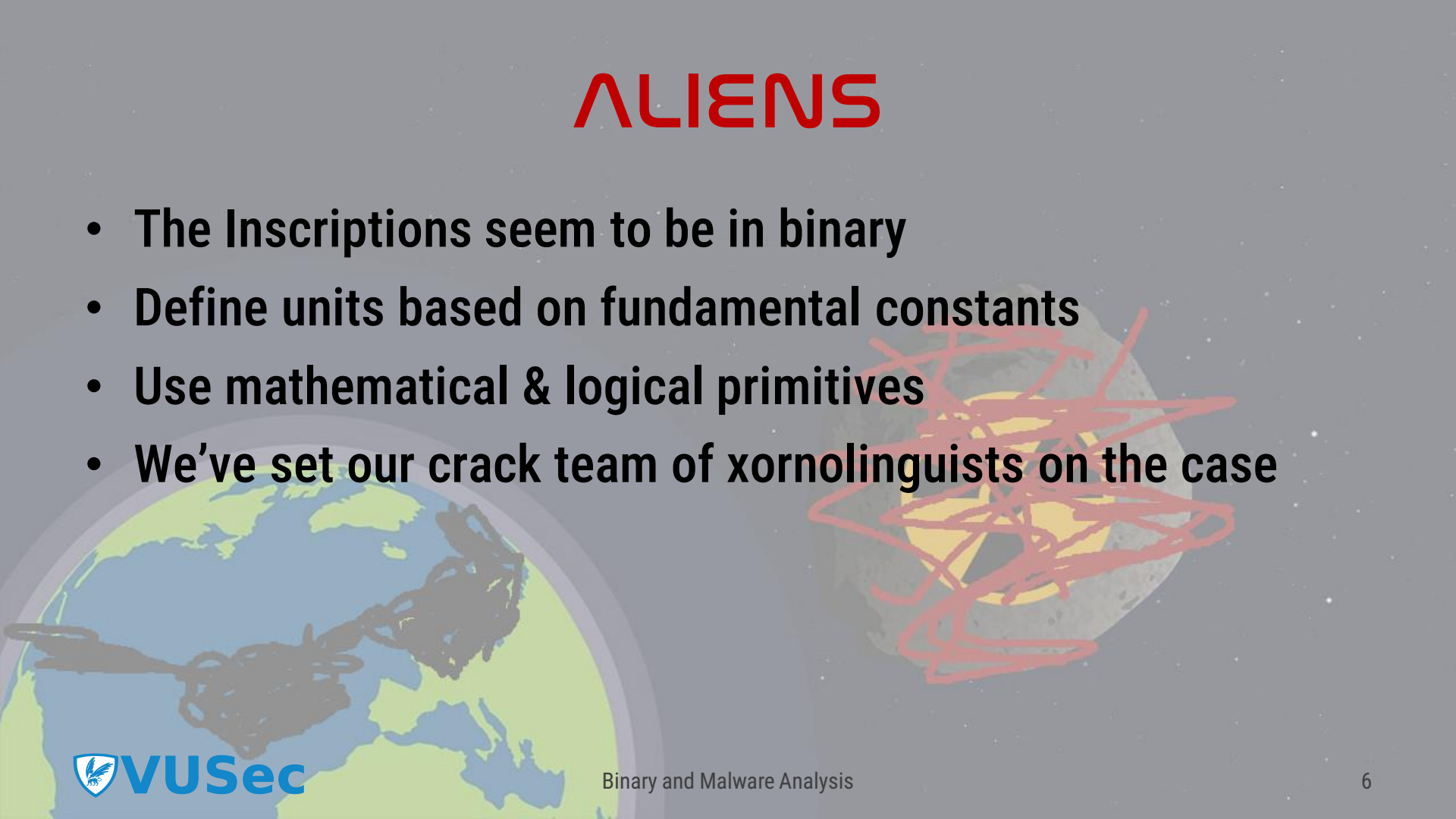
Now

- One of our agents has recovered what he claims is one of the asteroid pieces that have fallen to Earth
- He's provided:
  - Photographs
  - Chemical analysis
  - Transcript of peculiar inscriptions found on the meteorite

ALIENS

ALIENS

- The Inscriptions seem to be in binary
- Define units based on fundamental constants
- Use mathematical & logical primitives
- We've set our crack team of xornolinguists on the case



ALIENS

- They found two things:
  - A blob of binary they couldn't make sense of (yet)
  - Mathematical Rosetta Stone
  - ... that they kindly split up into more manageable chunks
  - ... and translated it into a program for us
  - ... but it looks quite convoluted even so

ALIENS

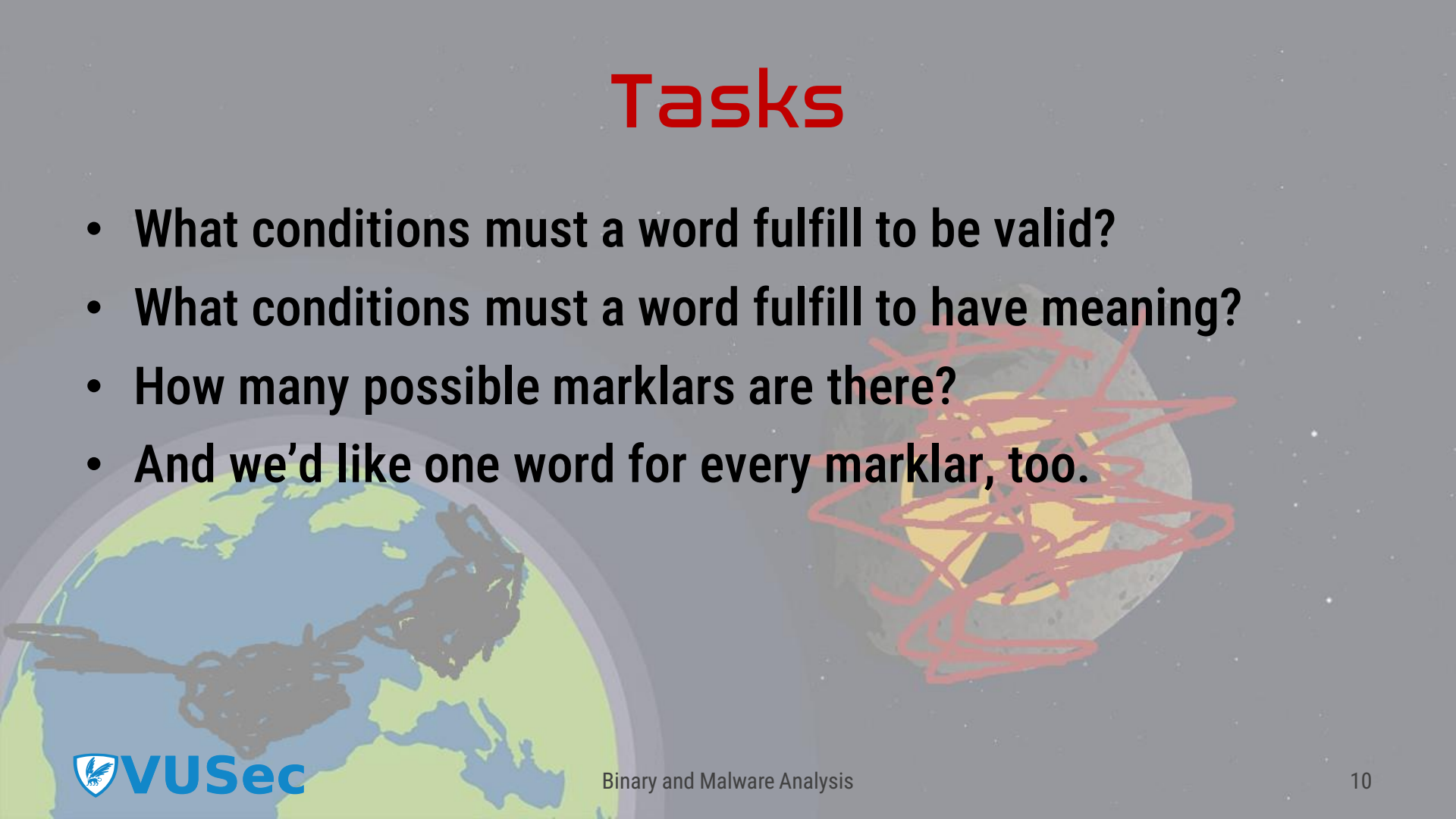
- They found two things:
  - A blob of binary they couldn't make sense of (yet)
  - Mathematical Rosetta Stone
  - ... that they kindly split up into more manageable chunks
  - ... and translated it into a program for us
  - ... but it looks quite convoluted even so
- **Symbex/SMT to the rescue!**

Challenges

- The aliens have a weird language
  - Not all possible char combinations form a valid word
  - Not all valid words have any real meaning
  - Each word w/ meaning maps to a “class of meaning” called a *marklar*
  - Words that have the same marklar are perfect synonyms
  - There are *many* words for every marklar
- All this is expressed compactly within said Rosetta Stone

Tasks

- What conditions must a word fulfill to be valid?
- What conditions must a word fulfill to have meaning?
- How many possible marklars are there?
- And we'd like one word for every marklar, too.

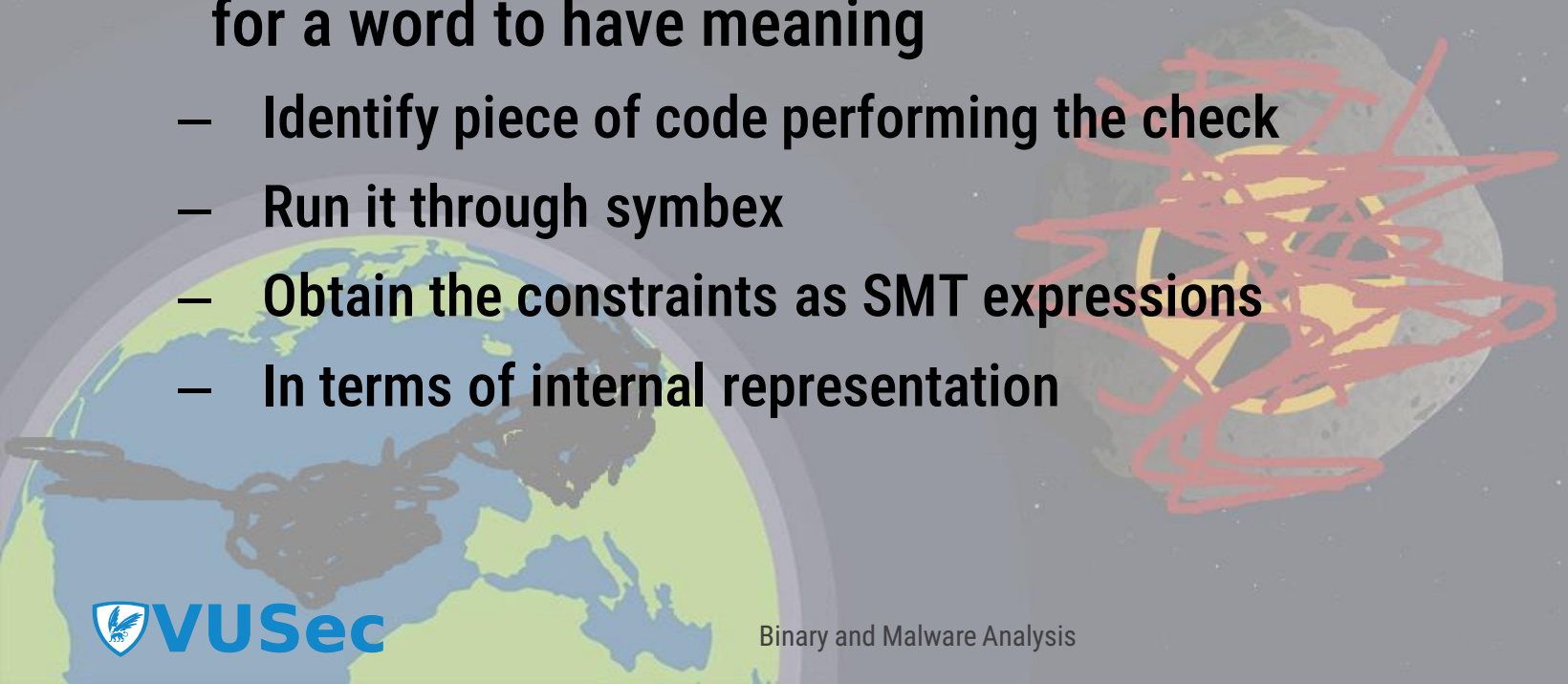


In Detail

1. Determine which words are valid & find one with a marklar
   - Use static analysis / dissassembly / decompilation
- Binary does encoding/decoding to/from an internal representation
   - Running the en-/de-coder through symbex is a pain
   - Figure out what it does and implement it as a program/script
   - Use static analysis / dissassembly / decompilation

In Detail

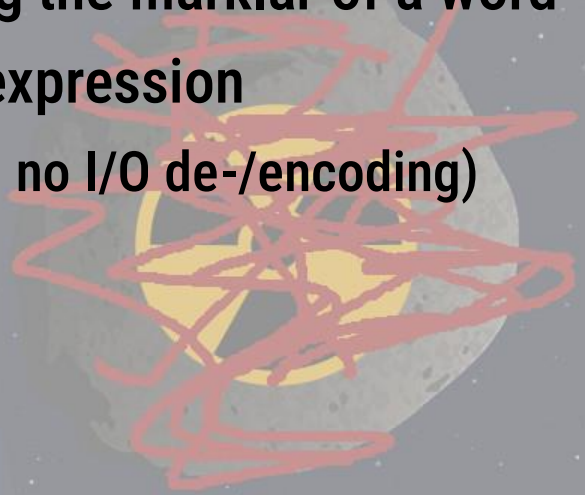
2. Use symbex to gather the constraints that need to apply for a word to have meaning
   - Identify piece of code performing the check
   - Run it through symbex
   - Obtain the constraints as SMT expressions
   - In terms of internal representation



In Detail

3. Find and capture the marklar assignment algorithm
   - Identify piece of code computing the marklar of a word
   - Capture this algorithm as SMT expression

   (in terms of internal representation, no I/O de-/encoding)



In Detail

4. Solve gathered constraints and obtain all possible marklars
   - Generate **marklar.out** – text file containing *all* possible marklars along with one word associated with each, one pair per line, separated by a space (i.e. "MARKLAR␣WORD")
   - Use previously gathered constraints
   - SMT solver will produce solutions in *internal representation*

   => Use your own hacked-together input/output encoders to produce the right marklars & words from internal representation

In Detail

5. Capture constraints for I/O coding
   - Capture constraints for input decoding
     - i.e. internal representation w.r.t. word
     - Integrate into your solving script
   - Capture constraints for output encoding
     - i.e. marklar w.r.t. internal representation
     - Integrate into your solving script
   - Produce marklar.out **without** your hacked I/O encoders

Grading

1. Determine conditions for valid words & find one word/marklar pair – 1p
2. Find and capture algorithm determining whether a word has meaning – 1p
3. Find and capture the algorithm that assigns marklars to words – 1p
4. Solve constraints and obtain all marklars + one word for each – 2p
   - we expect efficient implementations (more on that later)
5. Integrate I/O coding into your symbex script – 2p
   - Input decoding into internal repr – 1p
   - Output encoding from internal repr – 1p

- More to come, stay tuned! – 3p

BONUS

- Submission speed bonus for the fastest 5 students.
  - 50, 40, 30, 20 & 10 pts, respectively
- Execution speed bonus for the fastest 5 scripts.
  - 50, 40, 30, 20 & 10 pts, respectively
  - For fairness: done on a dedicated test binary similar to yours (ergo: provide handy constants for anything you hardcode!)

Submission Guidelines

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

- '**marklar.out**' – `MARKLAR␣WORD` **pairs**
- **angr scripts used to dump SMT and generate marklar.out**
- '**README**' **describing what what you did, how and why**
- **More to come, stay tuned!**

Submission Guidelines

We expect reasonably efficient solutions.

- We'll run with CPython & pypy and take the fastest time
- You get 10min on a 7700K w/ 32GB RAM for parts 2-4
- ...although if you hit this limit you're doing it wrong
- ...good scripts should finish much sooner
- You get 30min on the same system for part 5 integrated

Submission Guidelines

- You will receive your Rosetta Stone binaries by e-mail
- Submission will be through **Canvas**.
- **Deadline: Sunday, 24th May 2020, 23:59 CEST**
- Delay penalties: 1pt/24h delayed

GOOD LUCK!