CS 301 — Final Project

By: Solomon Himelbloom (Fall 2021)

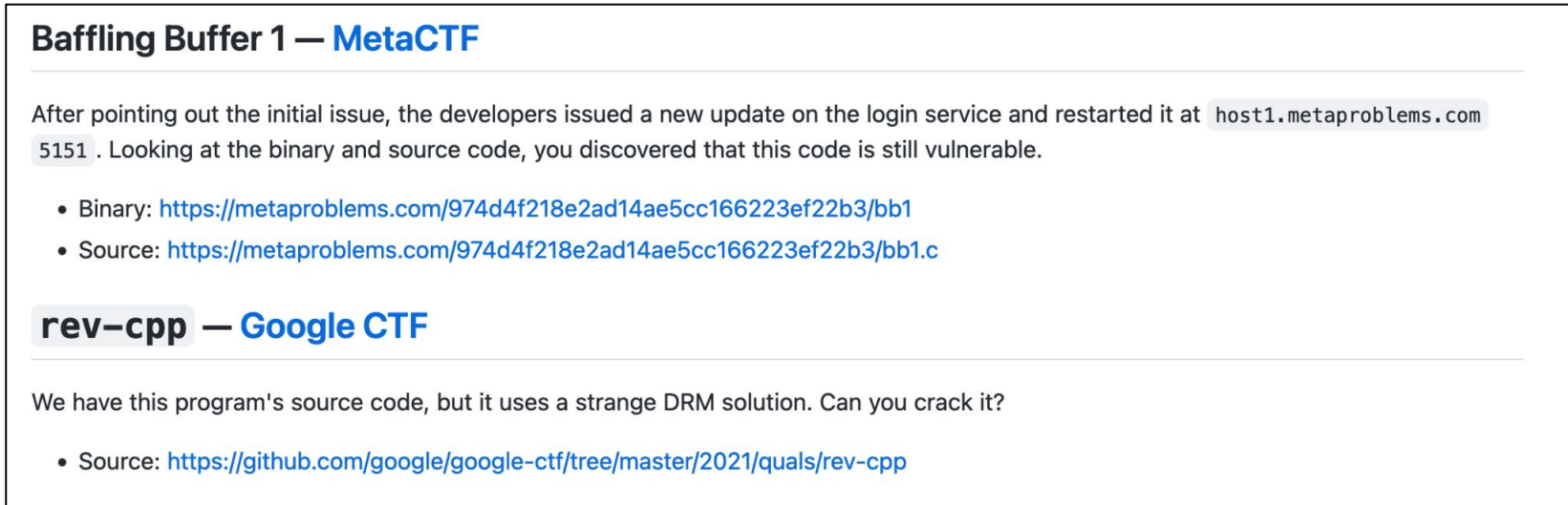
UNIVERSITY OF ALASKA FAIRBANKS *Naturally Inspiring.*

Reverse Engineering Prior CTF

🔗 https://github.com/TechSolomon/cs301/

Initial Idea

- Disassemble a binary and add comments under each line
- **Reverse Engineering** → function analysis; understanding behavior; malware & cyberdefense
- Originally practiced on **Capture The Flag** examples (below)
- Shifted towards recommendation of binary bomb challenge



Baffling Buffer 1 — MetaCTF

After pointing out the initial issue, the developers issued a new update on the login service and restarted it at `host1.metaproblems.com` `5151`. Looking at the binary and source code, you discovered that this code is still vulnerable.

- Binary: https://metaproblems.com/974d4f218e2ad14ae5cc166223ef22b3/bb1
- Source: https://metaproblems.com/974d4f218e2ad14ae5cc166223ef22b3/bb1.c

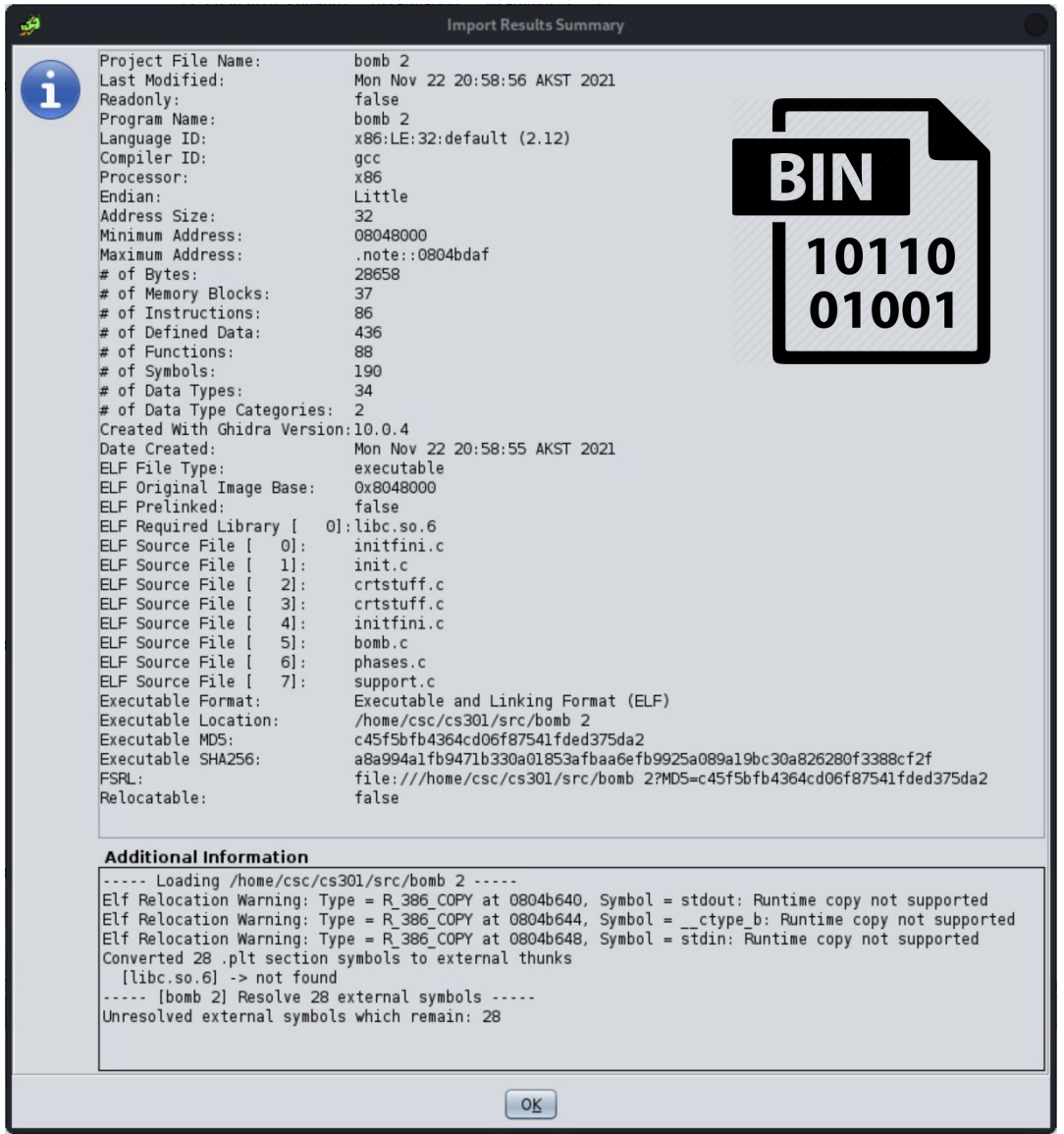
`rev-cpp` — Google CTF

We have this program's source code, but it uses a strange DRM solution. Can you crack it?

- Source: https://github.com/google/google-ctf/tree/master/2021/quals/rev-cpp

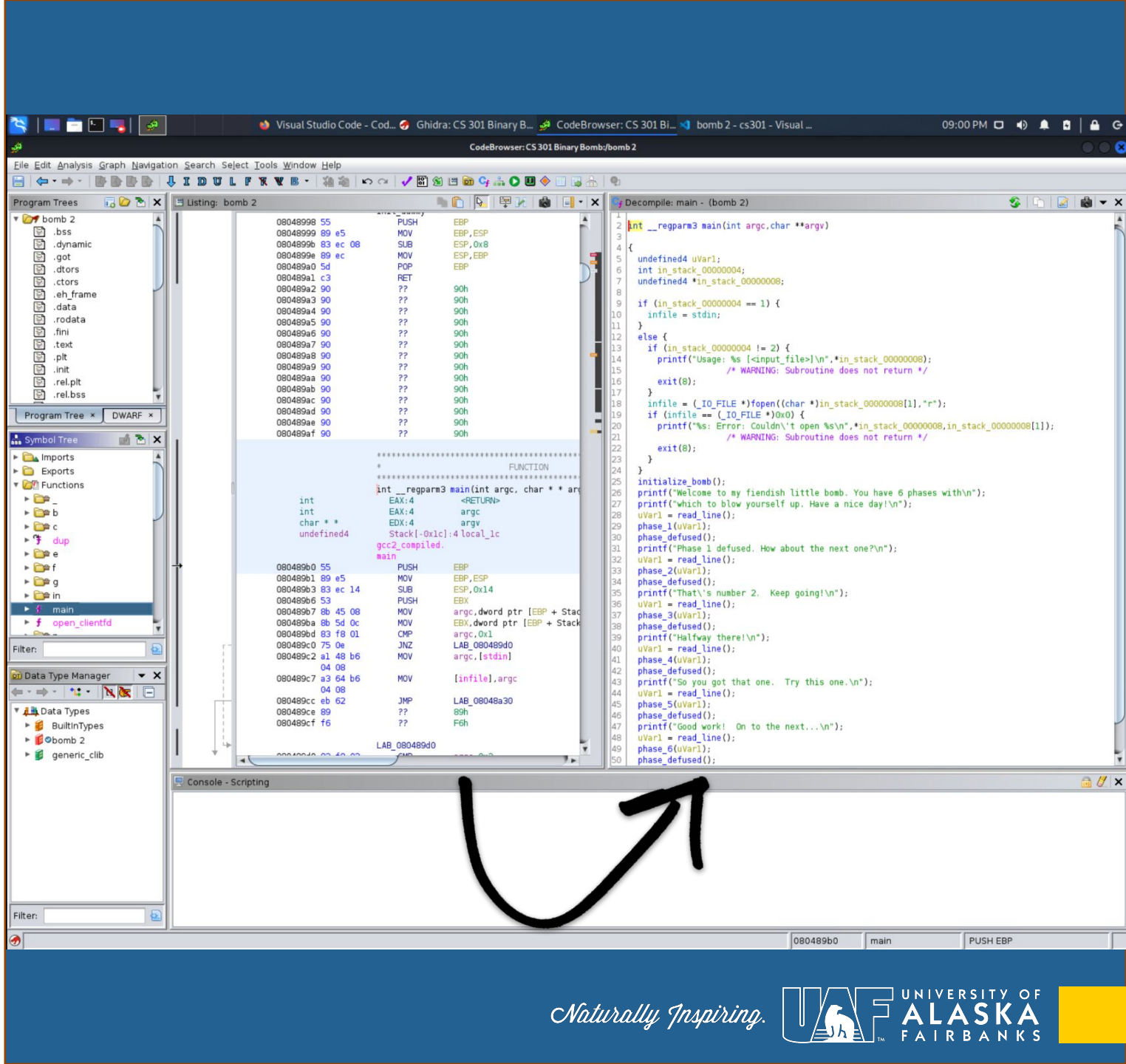
Tools: Ghidra vs. Compiled Static Binaries

- Free + open source tool developed by the NSA.
- Load binary: How was it compiled?
- Auto analysis: Determine flow of program to guess C code based on assembly.



Example Process

- Annotate unknown assembly: do not know the original source code.
- Machine code + labels: decipher program purpose.
- Validate assumptions via debugger.

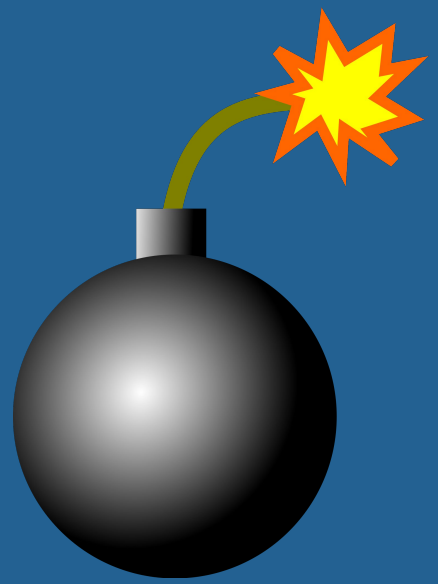


GHIDRA

Naturally Inspiring. UNIVERSITY OF ALASKA FAIRBANKS

Auditing Binaries

- https://trailofbits.github.io/ctf/vulnerabilities/binary.html
- http://csapp.cs.cmu.edu/2e/bomb32.tar



Binary Bomb → Trail of Bits

- Difusion tasks
  - Series of phases
  - Data structures + control flow
- Tracking progress
  - C (or other high level language)
  - IDA Pro (disassembler)
- Lessons learned
  - Organization
  - Comment consistency

Questions?