LPS Update Meeting

October 31, 2022

AWS Assemblage dataset

- ~500k binaries in dataset
  - More than 5M:1,142
  - More than 1M: 6,800
  - More than 100K: 115,320
- 62GB total size (all binaries, uncompressed)
- 500G total data w/ detailed JSON (roughly 7-9x binary size)

Subsetting data

- Lots of data
- Lots of smaller (not necessarily trivial) apps
- Solution: build subsets of data? >5MB, >1MB, >100k, full
  - Working to build each of these each time we export dataset
- Perhaps only full metadata for large apps?
- Notes:
  - Store rough metadata of binaries even if not actual binary / JSON
  - Start exporting subsets of data w/o deep JSON for "very small" repos (~<.5MB)

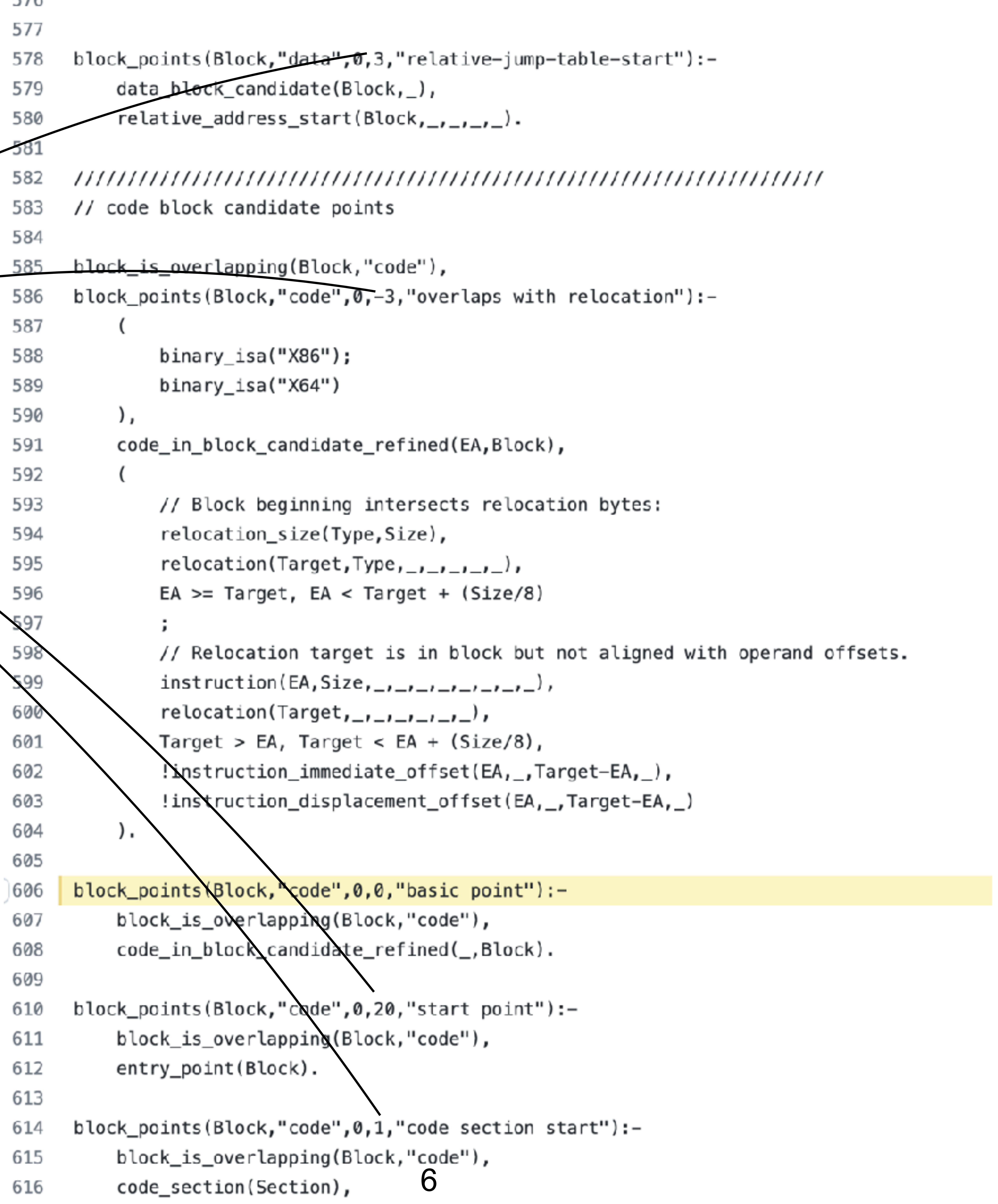
Dataset Reproducibility

- We are 3/4 done with reproducing our AWS dataset now
- Roughly ~5-10% of builds fail upon second build
  - Should be higher—we are looking into this now
- Out of the 407k binaries from second round:
  - 39k binaries are not in the first round (difference)
  - 368k are same as first round (successfully rebuilt)

Ddisasm Tuning

- Looking to use Assemblage data for novel binary analysis directions
- Idea: ddisasm uses hand-set *weights* to tune disassembly
  - Entrypoint identification
  - Symbolization
- Both of these are binary inferences, decided using weighted aggregate queries

Vector of weights



```
block_points(Block,"data",0,3,"relative-jump-table-start"):-
    data_block_candidate(Block,_),
    relative_address_start(Block,_,_,_,_).

////////////////////////////////////////////////////////////////////////
// code block candidate points

block_is_overlapping(Block,"code"),
block_points(Block,"code",0,-3,"overlaps with relocation"):-
    (
        binary_isa("X86");
        binary_isa("X64")
    ),
    code_in_block_candidate_refined(EA,Block),
    (
        // Block beginning intersects relocation bytes:
        relocation_size(Type,Size),
        relocation(Target,Type,_,_,_,_,_),
        EA >= Target, EA < Target + (Size/8)
        ;
        // Relocation target is in block but not aligned with operand offsets.
        instruction(EA,Size,_,_,_,_,_,_,_,_),
        relocation(Target,_,_,_,_,_,_),
        Target > EA, Target < EA + (Size/8),
        !instruction_immediate_offset(EA,_,Target-EA,_),
        !instruction_displacement_offset(EA,_,Target-EA,_)
    ).

block_points(Block,"code",0,0,"basic point"):-
    block_is_overlapping(Block,"code"),
    code_in_block_candidate_refined(_,Block).

block_points(Block,"code",0,20,"start point"):-
    block_is_overlapping(Block,"code"),
    entry_point(Block).

block_points(Block,"code",0,1,"code section start"):-
    block_is_overlapping(Block,"code"),
    code_section(Section),
```

Ddisasm Tuning (Progress)

- Key idea: can get **ground truth** from Assemblage
  - This is the our key strategic advantage in this space
- Running ddisasm to *recompile* binaries has been hard
  - Linker issues, etc…
- May focus on Linux binaries to test feasibility of this idea while we reach out to GT surrounding rebuilding Windows exes