Projects

CS6301-001: Language-based Security
Fall 2021
Logistics

• Teams
  • 3-4 students per team (self-selected)
  • Each team should send one email to me and the TA by Mon, Nov. 1:
    • Cc all team members
    • Include: Team Name (for eLearning), general project idea (need not be final)
  • TA will be main contact person for project help

• Deliverables
  • Implementation (Coq code plus any other tools you might develop)
  • Written report (pdf or Word)
  • Presentation slides (in-class presentations on last 2 days of class)
  • Due: Wednesday, December 15, 9am
Project Idea #1: Verify a **simple** correctness property of some **very simple** binary code

- Feasible properties:
  - Standard C string functions return correct answers (strcpy, strchr, etc.)
  - Windows “safe” string functions are really safe (e.g., strcpy_s, strcat_s)
  - Choose your own!

- Sample proof of full correctness of strcmp in strcmp_proofs.v
  - The correctness proof of strcmp turns out to be reasonably easy, making it a good sample proof. But many proofs turn out to be unexpectedly harder.
  - If the property you attempt turns out to be too hard and you can’t finish it, that’s okay. Making solid progress by proving important lemmas is still a successful project. (could contribute to Picinae’s library of theorems)
Project Idea #2: Improve Picinae

• Improve one of the architectures (x86, ARM7, RISC-V)
  • Example: Extend RISC-V with more instructions

• Improve expression auto-optimization tactic library
  • Example: Auto-cancel all identical terms with opposite signs in a nest of +/-

• Make induction easier / more obvious to use
  • I wish Picinae’s induction worked more like Coq’s induction

• Evaluation
  • A successful project should be demonstrable in some way.

• Warning: This is a highly open-ended project idea! I have no idea how hard this might be to do. But usually hard=interesting. 😊
Project Idea #3: Implement a new Lifter

• Currently Picinae uses a small plug-in (implemented by a former student of this class!) to auto-convert BAP IL to Picinae IL.
• BAP (Binary Analysis Platform) shouldn’t be the only option:
  • Angr (UC Santa Barbara and Arizona State)
    • Binary analysis framework (open source)
    • Widely used, extensive GUI, many years of development
    • Uses an IL called VEX to represent the semantics of machine instructions
    • API available for Python (so if you like Python better than Coq...)
• Project goal: Implement a Python plug-in for Angr that converts VEX to Picinae IL!
  • Use BAP-to-IL plug-in implementation and output as a guide
Project Idea #4: **Automate** proofs of “simple” binary code properties

- Proofs of architectural calling convention adherence
  - Example: callee-save registers are preserved (sample proofs in `strncpy_proofs.v`)
- Automation implementation
  - Learn Coq’s Ltac language and write tactics that can (usually) automatically prove it for common functions
  - Implement scripts (in any language) that auto-apply the “lifter” tool to all functions in a binary library (e.g., glibc) to get `.v` files
  - Apply your Ltacs to lifted `.v` files to auto-prove calling convention adherence
- Evaluation
  - Try it out on various binary libraries (e.g., glibc)
  - Try to refine it so that if it fails on some functions, it does so by leaving the harder subgoals for the user to manually prove
Presentations

• Last two days of class reserved for team presentations
• Create slides and present:
  • Your project idea/goals
  • Progress to date (show some code samples)
  • Any open challenges (solicit suggestions from class)
  • Either select one team member to present, or present as a team
  • Allocate ~15 minutes of presentation time + 5 min. Q&A / transition
• All team members receive same project grade (which includes presentation) irrespective of who presents.
Project Reports

• **Length:** Anywhere from 5-30 pages, but length doesn’t really concern me. I grade based on content and what you accomplished.

• **Sections:**
  • Intro/motivation: Explain the goal of the project and why it’s useful, important, and novel with respect to prior research.
  • Technical approach: Outline your approach to achieving the goal (e.g., major theorems proved, tactics implemented, etc.)
  • Team organization: Who did what?
  • Evaluation: Evaluate the success of your project. What did you achieve and how well did it work? What didn’t work and why?
  • Future Work: Recommend how future efforts should build upon your work to address any limitations.
  • Related work: Describe and cite any prior research that is significantly related to what you did.

• **Format:** LaTeX PDF is ideal, but any standard document format is fine (e.g., Word)