Semestral Project

PV286 – Secure Coding Principles and Practices

Spring 2024

CRoCS

Centre for Research on
Cryptography and Security

Project introduction

- Implementation of a project in a team of three
- Programming language from the following set
  - C, C++, Rust, Go, C#, Java (ask about other low-level languages)
- Four phases
- Up to 45 points awarded (**23 minimum**)
  - Bonus points possible for exceptional contribution (TBA)
- Questions
  - Put general questions in the discussion forum
  - Email other to kvapil@mail.muni.cz or your to-be assigned project reviewer
- Phase deadlines are strict
  - One day extension possible for 20% point loss
- All team members are expected to contribute equally
  - Learning to cooperate is part of the project goals
  - E.g., commits, textual description, presentation, etc.

Project idea

- Project topic assigned in two weeks
  - Attempts to put you in the shoes of an open-source project maintainers
- Implementation of an command line application that includes
  - Input parsing (CLI arguments, files)
  - Data manipulation and data structures, state preservation
  - Output formatting
- Allowed to use only the **standard** library
- Focus on the security of the implementation
  - Use defensive programming (strict user input checking, return values checking,...)
  - Write unit and integration tests, write **your own** tests (Test-driven development)
  - Use code analysis tools (fuzzing, static and dynamic analysis, symbolic analysis,...)

Project phase outputs

Loosely based on Build it Break it Fix it

1. Teams formation – deadline 3rd week
   - 3 students, chosen programming language, GitHub repository
2. Build It – deadline in 9th week
   - Major part of the specification implemented, video presentation and release provided
3. Break It – deadline in 11th week
   - Breaking another team's implementation, disclosing the issues, writing PR fixes
4. Fix It – deadline in 13th week
   - Found issues fixed and a final PDF report submitted

Each phase is graded separately.

Teams Formation

1 Teams Formation

- Form a team of 3 students
- Agree on your programming language
  - C, C++, Rust, Go, C#, Java
- Create a **private** repository on GitHub
- Write an email to kvapil@mail.muni.cz containing
  - [PV286] in the subject header
  - Team member names + UČOs + GitHub usernames
  - Selected programming language
  - Link to your GitHub repository
- You will be assigned a reviewer
- Deadline: Monday 4th March 2024
  - Don't wait for the deadline
  - No points assigned for this phase

1 Teams Formation

- No penalization for late submissions
  - Other phases have **strict** deadlines!
- Teams of 2 and unassigned students
  - Finalized this week
- The tutor reviewers will be assigned in a week
  - Add them to your GitHub repository

Build It

2 Build It

- Implement the project specifications except the `settle` subcommand
  - Include all test vectors in GitHub Actions
  - Test it for correctness and potential security issues
  - Setup commit signing
  - Release the final binary/jar build on GitHub
- Prepare and record a 10-minutes long presentation of your project
  - Code and design overview
  - Description of your testing and analysis
  - Application demonstration (building and running)
- Deadline: 23:59 Wednesday 17th April 2024
  - Submit the presentation slides, recording and project release to IS
    - Submission from this phase will be made available to the Break It team
  - Setup your code for a review
    - Create a private clone of your repository, with `-break-it` suffix
    - You will be requested to add access to your repository to the Break It team later on

Project Description

- Available on GitLab
- Intentionally a bit underspecified
- Feel free to comment on it
  - Use issues or even merge requests
  - Expect a few updates in the upcoming week
- Recommendations
  - Read the description several times, make notes
  - Meet regularly with your team in person and discuss your design on a whiteboard
  - Split the work (testing, CLI parsing, arithmetics calculation, transaction validation,...)
  - Don't Repeat Yourself (DRY) and Keep It Simple Stupid (KISS)
  - See previous suggestions about testing

Project Description

- The subcommands alter the *state*
  - Either all changes are committed or they are rolled back
- Respect the `--path` to limit the files your application creates
- Attacker model
  - Can run the application sequentially any number of times
  - Can modify any inputs (CLI options, stored files) and the execution environment
  - Cannot modify the binary
- The application must work on valid inputs and end gracefully on malicious ones
- Learn to think like an attacker
  - *Developer hopes that things work*
  - *Attacker tries things*
  - Verify all assumptions: *"surely, this shouldn't be possible"* — test it!
- Have fun!

Note

- This (9th) week there is no ROPOT

Break It

3 Break It

- You will be given access to another team's repository (`break-it` suffix)
  - Watch their presentation
  - It's recommended that you run it inside some kind of a sandbox environment
- Create your own fork(s) of the `*-break-it` repository
- Test the code both manually and automatically
  - Perform manual security analysis
  - Static analysis (at least 1), dynamic analysis (at least 1), fuzzing (at least 1)
  - Altogether use at least 4 tools (writing **multiple** unit tests counts as a single tool)
- Create issues in the `*-break-it` repository
  - Documenting your testing and results (single or multiple issues, depends)
  - Individual issues (can be PRs) for the (security) bugs found
    - Should contain description, proof of concept, and deemed severity (Low, High), etc.

3 Break It

- Create pull requests (from the fork) attempting fixing the issues
  - 1–3 PRs are enough
  - Can also be new tests, improved GA
  - Might not always be simple or possible, but three “whitespace” ones *won’t do*
  - We will value the efforts put into it (your recommendations, clarity, etc.)
- Engage in discussions (both the maintainers and reviewers)
  - Consider using Conventional Comments
- Again, individual contributions matter!
  - Comments in issues, PRs
  - Commits in PRs (bug fixes, tests, etc.)
- Deadline: 23:59 Wednesday 1st May 2024

Fix It

4 Fix It

- Full specifications shall be implemented and tested
- All tests shall be in GitHub Actions
  - If possible, some tools might need to be run locally or elsewhere (online)
- Static analysis (at least 1), dynamic analysis (at least 1), fuzzing (at least 1)
  - Can reuse/update the setups from the Break It phase
- Create regression tests for reported issues
  - Either by reviewer or the Break It teams
  - “Merge” the pull requests (if reasonable), or update them
- Write PDF report documenting your project, experience
  - Spare us any ChatGPT nonsense—can’t stop you from using it, but revise, shorten, etc.!
  - To be fully specified later