Semestral Project

PV286 – Secure Coding Principles and Practices

Spring 2025

RNDr. Jan Kvapil

PhD student @ CRoCS, A403

CRoCS

Centre for Research on Cryptography and Security

Project introduction

- Implementation of a specification in a team of three
- Programming language from the following set
  - C, C++, Rust, Go, C#, Java
- Four phases
- Up to 45 points awarded (**23 minimum** for passing)
  - Bonus points possible for exceptional contribution
- Questions
  - Put general questions in this discussion forum
  - Email other to kvapil@mail.muni.cz (Start the subject with `[PV286]`) or your to-be assigned reviewer
- Phase deadlines are strict
  - One day extensions 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

- Implement a command line wrapper around BIP380
  - Put you in the shoes of an open-source project maintainers
- CLI application (TBA) that includes
  - Input parsing (CLI arguments, files)
  - Data manipulation and data structures, no state
  - Output formatting
- Use only the **standard** library
  - Possibly previously agreed upon relevant 3rd party BIPs
  - Cannot use CLI arguments parsing (parse `args` array directly)
- Focus on functional **and secure** implementation
  - Use defensive programming (strict user input and 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 in the 3rd week
   - 3 students, chosen programming language
2. Build It – deadline in the 8th week
   - Major part of the specification implemented, video presentation and release provided
3. Break It – deadline in the 10th week
   - Breaking another team's implementation, disclosing the issues, writing PR fixes
4. Fix It – deadline in the 12th week
   - Found issues fixed and a final PDF report submitted

Points assigned for each phase separately.

Teams Formation

1 Teams Formation

- Three students
- Agree on your programming language
  - C, C++, Rust, Go, C# or Java
- Submit https://forms.gle/QoXaotGmAFsECmXQ8
  - Use <UCO>@mail.muni.cz Google account see here
  - Later, you will be assigned a reviewer
  - And a repository at https://gitlab.fi.muni.cz/pv286/teams
- Check access to https://gitlab.fi.muni.cz
- Deadline: **23:59 on Monday, 3rd March 2025**
  - Don't wait for the deadline (shared calendar)
  - 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
  - To be finalized this week
- Also, the tutor reviewers will be assigned in a week

Build It

2 Build It

- Implement the project specifications
  - Add README with build and run instructions
  - Include all test vectors in GitLab Continuous Integration
  - Test it for correctness and potential security issues
  - Setup commit signing (every team member)
  - Release the final binary/jar build on GitLab
- Record 10-minutes long video presentation of your project including
  - Code design overview
  - Description of your self-testing and analysis
  - Application demonstration, i.e. building and example running
  - Last year's recording as an example
- Deadline: **23:59 on Monday, 7th April 2025**
  - Submit the presentation slides, recording and the final project release to this IS directory
    - Submission from this phase will be then available to the Break It team

Project Description

- TBA on GitLab this week (expect an e-mail)
- Feel free to comment on it
  - Use issues or even merge requests
  - Finalized within a week
- Recommendations
  - Read the description several times, make notes
  - Meet regularly with your team in person and discuss your design on a whiteboard
    - 0,5 point for weekly summary for your reviewer
  - Split the work among the team (testing, CLI parsing,...)
  - Principles Don’t Repeat Yourself (DRY) and Keep It Simple Stupid (KISS)
  - See previous suggestions about testing

Project Description

- Attacker model
  - Can run the application
  - Can provide invalid CLI inputs and *modify 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 FIXME 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

4 Fix It

- Analyze the assigned implementation
  - Overall code quality
  - Static analysis (at least 1), dynamic analysis (at least 1), fuzzing (at least 1)
  - Use at least 5 different tools in total
- Provide comments on the code in GitHub `review` branch
  - Consider using Conventional Comments
- Prepare pull requests fixing at least 1 discovered issue
- Write 3-4 page report of your analysis covering
  - Overall code quality
  - Used analysis tools
  - Discovered issues and fixes
  - Summary and score of the implementation based on your analysis (5 highest, 0 lowest).
- Deadline: FIXME 2023
  - Submit the report to IS

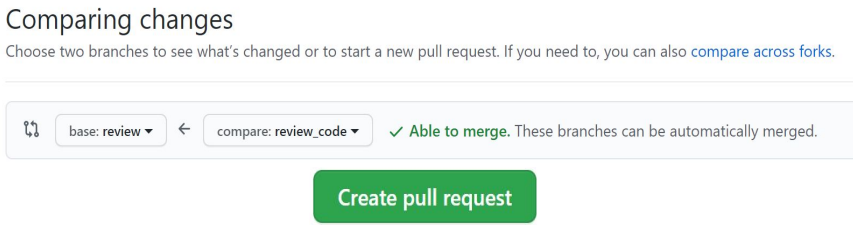
3 Break It

```
# Create review branch without code
git checkout -b review
git rm -r --cached .
git commit -m "Create review branch"
git push --set-upstream origin review

# Create branch for pull request into the review branch
git checkout -b review_code
git add .
git commit -m "Add review code"
git push --set-upstream origin review_code
```

4 Fix It

- Create pull request from `review_code` to `review` branch



- Reviewing team will comment in the pull request