Semestral Project

PA193 – Secure Coding Principles and Practices

Spring 2022

CRoCS

Centre for Research on Cryptography and Security

Introduction

- Team of three people
- Programming language of your choice
- Four phases (~3 weeks each)
- Up to 30 points awarded
  - Bonus points possible for exceptional contribution
- Questions
  - Anytime by email: xdufka1@fi.muni.cz
  - Consultation possible upon request

Project phases

- Phase I – deadline 2nd week
  - Form teams of 3 people
  - Setup GitHub repository and prepare a project with test vectors
- Phase II – deadline 5th week
  - Setup automatic testing and commit signing
  - Start implementation
  - Report (3-4 A4)
- Phase III – deadline 9th week
  - Finalize the implementation and release the final build
  - Recording and live presentation of your project (5-7 minutes)
- Phase IV – deadline 13th week
  - Analyze project of another group
  - Final presentation at the last lecture (10 minutes)

Bech32m encoding

- Encoding using only 32 alphanumeric characters with efficient error detection and correction
  - Letters can be in upper or lower case, but lower case is preferred
  - Similar letters avoided (e.g., i and j)
  - Easy to read out loud
  - Easy to implement
  - Alphabet size of prime power is suitable for error-detecting codes
  - Error correction is possible
  - Efficiently encodable in QR codes in “alphanumeric mode”

- Bech32m has been designed as a replacement of base58 encoding, previously used in Bitcoin

Bech32m encoding tool

- Implement tool for bech32m encoding and decoding
  - https://github.com/bitcoin/bips/blob/master/bip-0173.mediawiki
  - https://github.com/bitcoin/bips/blob/master/bip-0350.mediawiki
- Provide command-line interface for:
  - Encoding of arbitrary input to bech32m
  - Decoding of bech32m
  - Choosing of input/output format (base64/hex/binary)
  - Selecting input from cli-argument, file, or stdin (default)
  - Selecting output to file, or stdout (default)
- In case of erroneous decoding suggest the closest valid input
  - (Bech32m supports error correction)
- Focus on security of your implementation
  - Proper error handling
  - Secure handling of user-provided input

Phase I

- Form teams of 3 people
- Agree on your programming language:
  - C, Python, Rust, C++, Haskell, Go, …
  - The language must be unique per seminar group
- Prepare your project on GitHub
  - Create a GitHub repository
  - Agree on a unique name
  - Prepare an empty project with test vectors from bech32m specification
- Write an email to xdufka1@fi.muni.cz containing:
  - Team member names
  - Selected programming language
  - Link to GitHub repository (add dufkan as reader if it is a private repository)
- Deadline Monday 21. 2. 2022 16:00

Phase II

- Configure Github Actions to run tests automatically
- Start digitally signing your commits
- Start the implementation
  - You can use only standard library
  - By the end of this phase, you should have:
    - Basic encoding/decoding functionality passing test vectors.
    - No need to provide user interface yet.
- Prepare 3-4 A4 report
  - Project design
  - Current progress
  - Encountered obstacles
- Deadline Monday 14. 3. 2022 16:00

Phase III

- Finalize the implementation
  - Try to identify any vulnerabilities with analysis tools
  - Release the final binary build with a digital signature (GPG)
- Prepare and record a presentation of your project (5-7 minutes)
  - Structure of the project
  - Encountered obstacles and solutions
  - Used analysis tools
  - How can the tool be used
    - (Quick guide for the other team in Phase IV)
- Discussion of the presentation
- Assignment of other team projects for the next phase
- Deadline Monday 11. 4. 2022 16:00

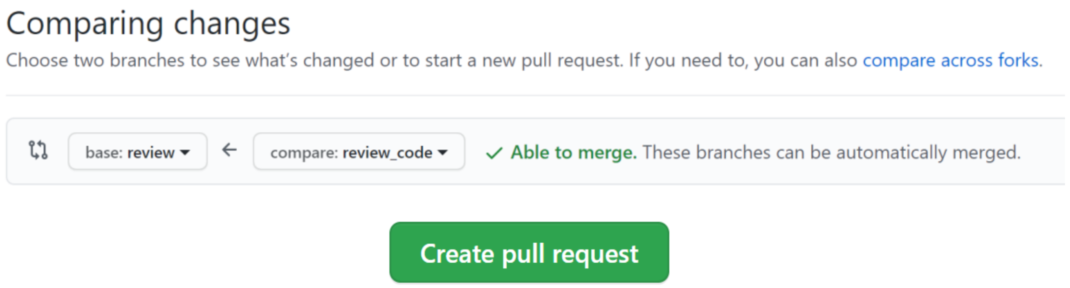
Phase IV – Review setup

```
# 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
```

Phase IV – Review setup

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

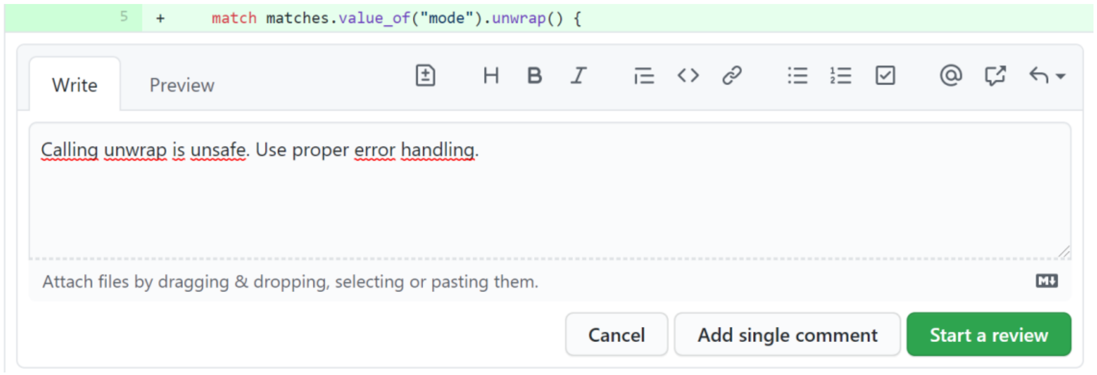


Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: review
compare: review_code
✓ Able to merge. These branches can be automatically merged.
Create pull request

- Review team will comment in the pull request (give them access)



```
+     match matches.value_of("mode").unwrap() {
```

Calling unwrap is unsafe. Use proper error handling.

Write
Preview
Attach files by dragging & dropping, selecting or pasting them.
Cancel
Add single comment
Start a review

Phase IV

- Analyze implementation of other team (assigned on seminars)
  - Static analysis (at least 1 tool)
  - Dynamic analysis (at least 1 tool)
  - Fuzzing (at least 1 tool)
  - Dependency checking, … (use at least 5 tools in total)
- Provide comments on code in GitHub review branch
  - Use conventional comments: https://conventionalcomments.org/
- Prepare pull request with a fix of at least 1 discovered issue
- Prepare presentation for the last lecture (10 minutes)
  - Used analysis tools
  - Discovered issues, code quality
  - Fixes (+ screenshot of pull request)
- Deadline Monday 9. 5. 2022 16:00