PA193 - Secure coding principles and practices

Designing good and secure API
Automata-based programming
Seminar

Łukasz Chmielewski *chmielewski@fi.muni.cz*
Centre for Research on Cryptography and Security, Masaryk University

CROCS
Centre for Research on Cryptography and Security

Outline

- Discussion on mindmap (miro) from seminar 1
- Check-in
- Mindmap from slides
- Side-Channel Secure API
- Small FSM exercise

Note: password for the various activities

- If you are asked for password (Miro boards…) use ‘fimunicz’

Task 1: seminar 1 mindmap

5’

- Have a look here:
  - https://miro.com/app/board/uXjVMUuprJw=/
  - Password: fimunicz
- What do you think about it?

Task 2: check-in

10’

- Why “I know Kung-fu”?
- One single interesting point in video talk (every person) + speedup factor

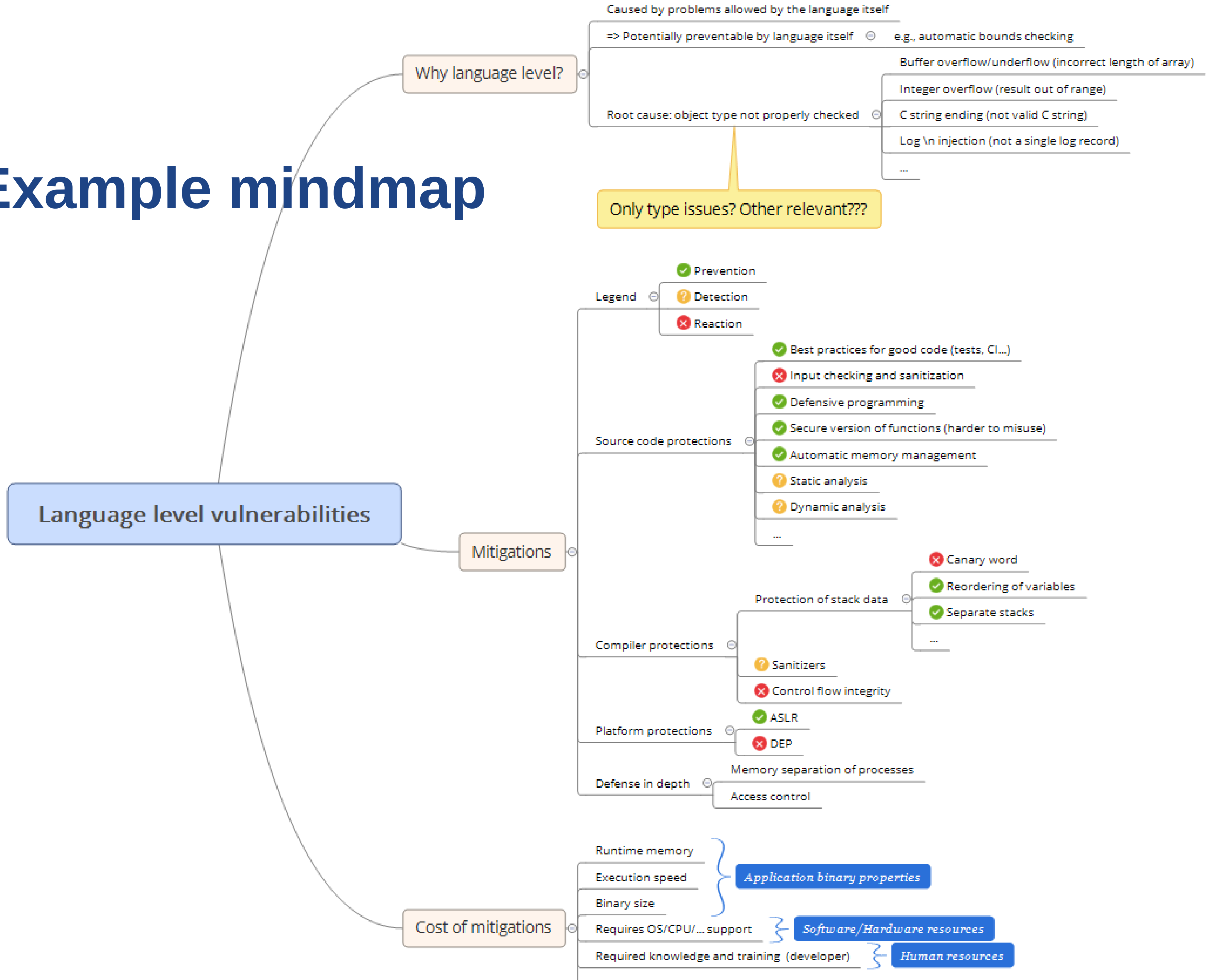
Task 3: Draw Mindmap of this lecture

15’

- Group of 3 (not seminar team)
- Draw a visual mindmap here: https://miro.com/app/board/uXjVMU01_YU=/

1. Addressed problems (root cause)
2. Solution concepts
3. Practical techniques & tools

Example mindmap



Language level vulnerabilities
Why language level?
Caused by problems allowed by the language itself
=> Potentially preventable by language itself
e.g., automatic bounds checking
Root cause: object type not properly checked
Buffer overflow/underflow (incorrect length of array)
Integer overflow (result out of range)
C string ending (not valid C string)
Log \n injection (not a single log record)
...
Only type issues? Other relevant???
Mitigations
Legend
✓ Prevention
? Detection
✗ Reaction
Source code protections
✓ Best practices for good code (tests, CI...)
✗ Input checking and sanitization
✓ Defensive programming
✓ Secure version of functions (harder to misuse)
✓ Automatic memory management
? Static analysis
? Dynamic analysis
...
Compiler protections
Protection of stack data
✗ Canary word
✓ Reordering of variables
✓ Separate stacks
...
? Sanitizers
✗ Control flow integrity
Platform protections
✓ ASLR
✗ DEP
Defense in depth
Memory separation of processes
Access control
Cost of mitigations
Runtime memory
Execution speed
Binary size
Application binary properties
Requires OS/CPU/... support
Software/Hardware resources
Required knowledge and training (developer)
Human resources

Task 4: Crypto Library API
(real issues…)

15’

- Check out the code from:
  - https://github.com/sca-secure-library-sca25519/sca25519/blob/main/common/crypto/include/crypto_scalarmult.h
- Read README and we will look at the difference between:
  - STM32F407-unprotected
  - STM32F407-static
- Locate the main API files.
- After you locate them – check what is really wrong (hint …unprotected)
- Check for the real differences in the API.
  - Why they are there?
  - What more changes could be there?

Task 5: JavaCard applet FSM

15’

- Have a look at https://github.com/petrs/state_enforcer/
- Install the tool (includes the GraphViz package)
- Have a look at the example – what does it do?
- Extend the example by adding functionality to set the PIN
  - Pin should be linked to the generate key pair

Task 6: Question 1

5’

- Explain how is automata-based programming (FSM) improving the security of an application.
- Discuss what differences in resulting API are likely to be seen in case FSM 1) is used and 2) is not used.

Task 6: Question 2

5’

- Explain how ‘Source-code annotation language (SAL)’ is making the code better .
- Describe what fundamental problems are addressed, where and how applied, what obstacles are expected in deployment…

Task 6: Question 3

5’

- Describe how Application Programming Interface (API) relates to Application Binary Interface (ABI) and provide example of relevant security problem.

Optional Task 7: Kahoot

- Deeper and more difficult questions
- More time per question, think, don’t just guess
- Discussion of answers

Task 4: Feedback to this lecture

5’

- Activities performed
  - Lecture discussion
  - Mindmap exercises
  - Crypto Library API
  - JavaCard Applet FSM
  - Questions

HOMEWORK ASSIGNMENT 5

Security module – informal description

“Hardware token (module) allows to import two master symmetric cryptography keys and set initial user PIN value in the trusted environment during the personalization. Once token is personalized and issued to a user, a user can submit its own data, that are then encrypted (or decrypted) and integrity protected (or integrity verified, MAC). These operations are available only after successful verification of the user PIN. User can change his/her PIN to new value after successful verification of the current PIN. Cryptographic algorithms used are set and fixed during personalization, but should be easily modifiable for the newer tokens, if required.”

Design security API, make FSM model

- Design security API for security module adhering to the best practices described during the lecture
  - – gather requirements, make use cases, create API, document...
  - – provide declaration of API in C/C++
  - – add SAL annotations
- Create FSM states model with transitions
  - – use Graphviz's .dot format
  - – visualize, add to documentation

Homework Assignment 5

- Deadline: 24.04.2023 23:59
- Finish API design and FSM
- Submit (all below totals to 10 points):
  - use cases (plaintext or UML)
  - API written in source code (C/C++)
  - JavaDoc-like documentation inside source code
  - SAL annotations inside source code
  - generate html from documentation (Doxygen)
  - include FSM visualization (Graphviz)
  - (Pseudo)code for client using API

Questions