PB173 - Tématický vývoj aplikací v C/C++ (jaro 2017)

Skupina: Aplikovaná kryptografie a bezpečné programování

Petr Švenda svenda@fi.muni.cz

CROCS
Centre for Research on Cryptography and Security

Security code review

- Architecture overview
  - Design choices and possible design flaws
- Code review
  - How well is architecture actually implemented
- Whitebox, greybox & blackbox testing
  - different level of access to code and documentation
- Available tools
  - mainly for code review

Security code review (2)

- You will always have a limited time
  - try to rapidly build overall picture
  - use tools to find low hanging fruit
- Focus on most sensitive and problematic areas
  - use tools to focus your analysis scope
- More eyes can spot more problems
  - experts on different areas

Architecture overview

Architecture overview

- Get all information you can quickly
- Assets
  - What has the value in the system?
  - What damage is caused when successfully attacked?
  - What mechanisms are used to protect assets?
- Roles
  - Who has access to what?
  - What credentials needs to be presented?
- Thread model
  - What is expected to do harm?
  - What are you defending against?

Architecture overview (2)

- Usage of well established techniques and standards
- Comparison with existing schemes
  - What is the advantage of new scheme?
  - Why changes were made?
- Security tradeoffs documented
  - Possible threat, but unmitigated?
  - Is documented or overlooked?

Sensitive data flow mapping

- Identify sensitive data
  - password, key, protected data...
- Find all processing functions
  - and focus on them
- Create data flow between functions
  - e.g. Doxygen call graph
- Inspect when functions can be called
  - Is key schedule validity checked?
  - Can be function called without previous function calls?
- Where are sensitive data stored between calls?

Protocol design (and implementation)

- Packet confidentiality, integrity and authenticity
- Packet removal/insertion detection
- Replay attack
- Reflection attack
- Man in the middle

Code overview

Cryptography usage

- CIA (Confidentiality, Integrity, Availability)
  - Plaintext data over insecure channel? Encrypted only?
  - Can be packet send twice (replay)?
  - What is the application response on data modification?
- What algorithms are used
  - Broken/insecure algorithms? MD5? simple DES?
- What key lengths are used?
  - < 90 bits symmetric crypto?
  - < 1024 bits asymmetric crypto?
- Random number generation
  - Where the key comes from?
  - Is source entropic enough?
  - srand() & rand()?

Cryptography usage (2)

- Key creation
  - Where the keys originate? Enough entropy?
  - Who has access?
- Key storage
  - Hard-coded keys
  - Keys in files in plaintext
  - Keys over insecure channels
  - Keys protected by less secure keys
- Key destruction
  - How are keys erased from memory?
  - Can exception prevent key erase?

Cryptography implementation

- Implementation from well known libraries?
- Own algorithms?
  - security by obscurity?
  - usually not secure enough
- Own modifications?
  - Why?
  - sometimes used to prevent compatible programs
  - decreased number of rounds?
  - Performance optimization with security impact?

Code inspection

- Overall code logic
- Memory management - allocation, input validation
- String operations – copy, concatenate, string termination
- Data flow – conditional jumps, test of return values
- Race conditions (TOCTOU)

Input validation

- Hard (and expensive) to do right
- Always use white-listing (what is allowed), not black listing (what is banned)
- Check for buffer overruns
  - functions called with attacker's input
  - dangerous functions (strcpy…)
  - arrays with fixed lengths
- Large inputs in general
  - try to insert 1KB of text instead of user name
- Fuzzing
  - large amount of automated inputs with different length

Recommended reading

- Process of security code review
  - http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01668009
- Why cryptosystems fail, R. Anderson
  - http://www.cl.cam.ac.uk/~rja14/Papers/wcf.pdf
- Software Security Code Review
  - http://www.softwaremag.com/l.cfm?doc=2005-07/2005-07code
- Static code analysis tools
  - http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
- Security in web applications (OWASP)
  - http://www.owasp.org/index.php/Code_Review_Introduction

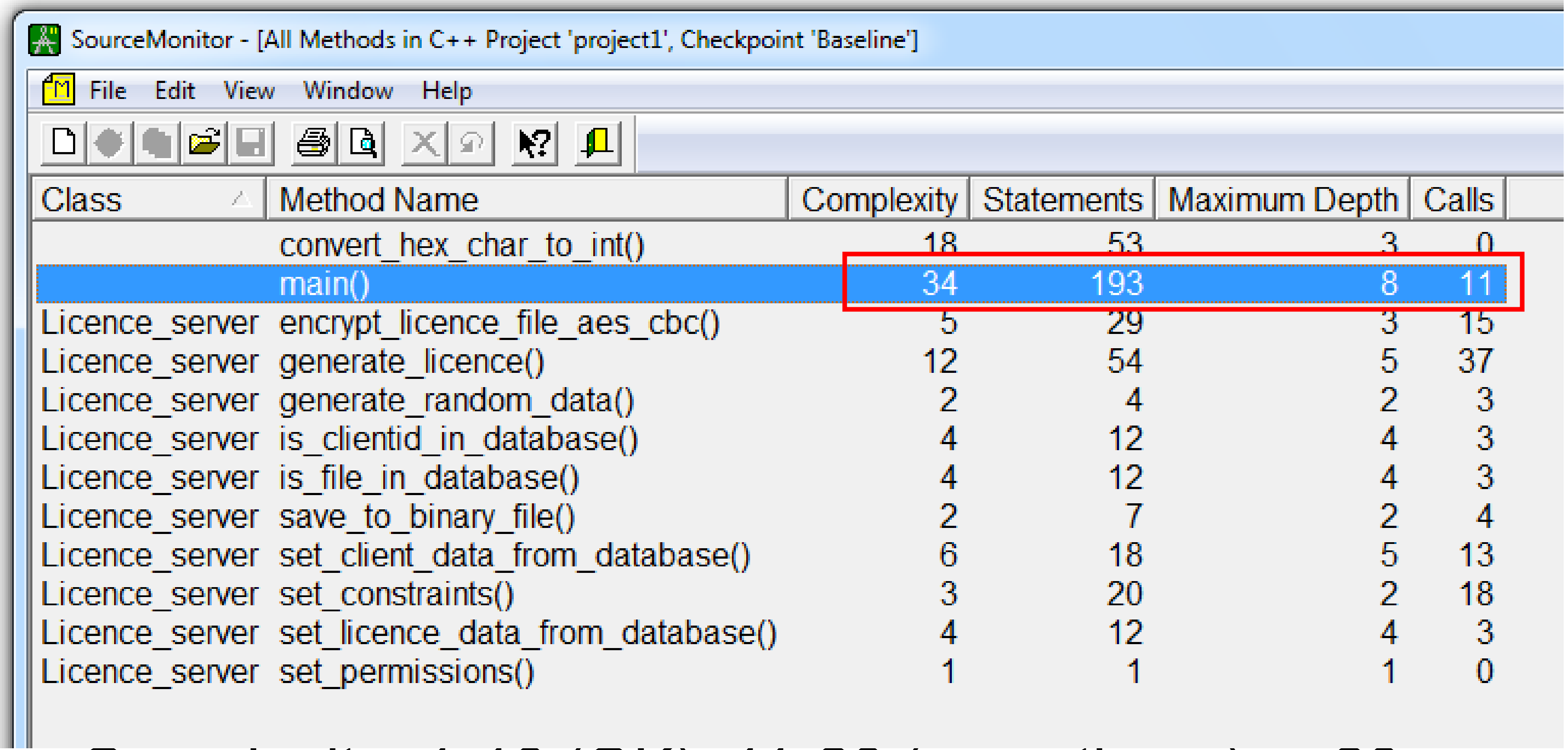
Static analysis tools

- List of static checkers
  - http://spinroot.com/static/
  - http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
  - https://security.web.cern.ch/security/recommendations/en/code_tools.shtml
- We will be interested in C/C++ checkers
  - but tools exists for almost any language

Both free and commercial tools

- Commercial tools
  - PC-Lint (Gimpel Software)
  - Klocwork Insight (Klocwork)
  - **Coverity** Prevent (now under HP, free for open-source)
  - Microsoft **PREfast** (included in Visual Studio)
- Free tools
  - Rough Auditing Tool for Security (RATS) http://code.google.com/p/rough-auditing-tool-for-security/
  - **CppCheck** http://cppcheck.sourceforge.net/
  - Flawfinder http://www.dwheeler.com/flawfinder/
  - Splint http://www.splint.org/
  - **FindBugs** http://findbugs.sourceforge.net (for Java programs)
  - Doxygen's call graphs from source http://www.stack.nl/~dimitri/doxygen/
  - ...

Source monitor – example outputs


SourceMonitor - [All Methods in C++ Project 'project1', Checkpoint 'Baseline']
File Edit View Window Help

Class	Method Name	Complexity	Statements	Maximum Depth	Calls
	convert_hex_char_to_int()	18	53	3	0
	main()	34	193	8	11
Licence_server	encrypt_licence_file_aes_cbc()	5	29	3	15
Licence_server	generate_licence()	12	54	5	37
Licence_server	generate_random_data()	2	4	2	3
Licence_server	is_clientid_in_database()	4	12	4	3
Licence_server	is_file_in_database()	4	12	4	3
Licence_server	save_to_binary_file()	2	7	2	4
Licence_server	set_client_data_from_database()	6	18	5	13
Licence_server	set_constraints()	3	20	2	18
Licence_server	set_licence_data_from_database()	4	12	4	3
Licence_server	set_permissions()	1	1	1	0

- Complexity: 1-10 (OK), 11-20 (sometimes), > 20 (BAD)

Schedule

- 17.4. No seminar
- 24.4. Implementation with TCP socket
- 1.5. No seminar
  - 5.5. Final implementation with TLS socket (QSslSocket) => start of code review
- 8.5. No seminar
- 15.5. Presentation of other project review

Practical assignment – code review

- Make your documentation & code available
  - Create Doc folder in your GitHub repo
  - Upload also to IS repository (available to others)
    - Crypto - Project for review
  - deadline 5.5. 23:59 (Friday)
- Security analysis of design and code of other team
  - Start work after 5.5. 24:00 based on IS and GitHub materials
  - Presentation of findings 15.5. 12:00
- Points (max. 30) will be awarded according to:
  - number&severity of problems found in reviewed projects
  - quality of architecture and code

Practical assignment

- Some tips what to analyze:
  - which functions are manipulating with sensitive information
  - where is random numbers coming from
  - code bugs?
- Use some analysis tools
  - gcc -Wall -Wextra
  - MSVS:Project→C/C++ →General →Warning level (/W4 /Wall)
  - call graphs (e.g., Doxygen, http://cecko.eu/public/doxygen)
  - Cppcheck (C/C++, Windows) http://cppcheck.sourceforge.net/
  - Code complexity
  - ...

Practical assignment (2)

- Summarize your findings
  - problem identification + severity + applicability + short description
  - 2 pages enough (per project)

Identifikace problému: A_x (celková bezpečnostní architektura) / C_x (kód implementace)
Závažnost: nízká / střední / vysoká / není možné rozhodnout
Proveditelnost útoku: snadná (lze přímo externím útočníkem) / v závislosti na dalších součástech systému / není možné rozhodnout (obvykle značí potenciální zranitelnost, kde ale detailní postup pro možné zneužití přímo neznáme)
Popis problému: místo výskytu v kódu ve tvary soubor.c:číslo_řádku:funkce – popis
Navrhované řešení: jednoduchý popis (v případě, že jsme návrh schopni poskytnout)

Total points

- We missed three weeks due to public holiday
  - So far 7 assignments assigned
  - Two more: TLS implementation, project review
  - 30 points for project quality and functionality
- Original requirements:
  - 10x10 (hw) + 30 (project) => max 130, 85 to pass
- Updated requirements:
  - 9x10 (hw) + 30 (project) => **max 120, 75 points to pass**