

PB173 - Tématický vývoj aplikací v C/C++ (podzim 2012)

Skupina: Aplikovaná kryptografie a bezpečné programování

https://minotaur.fi.muni.cz:8443/pb173_crypto

Petr Švenda, svenda@fi.muni.cz

Konzultace: G201, Pondělí 16-16:50

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

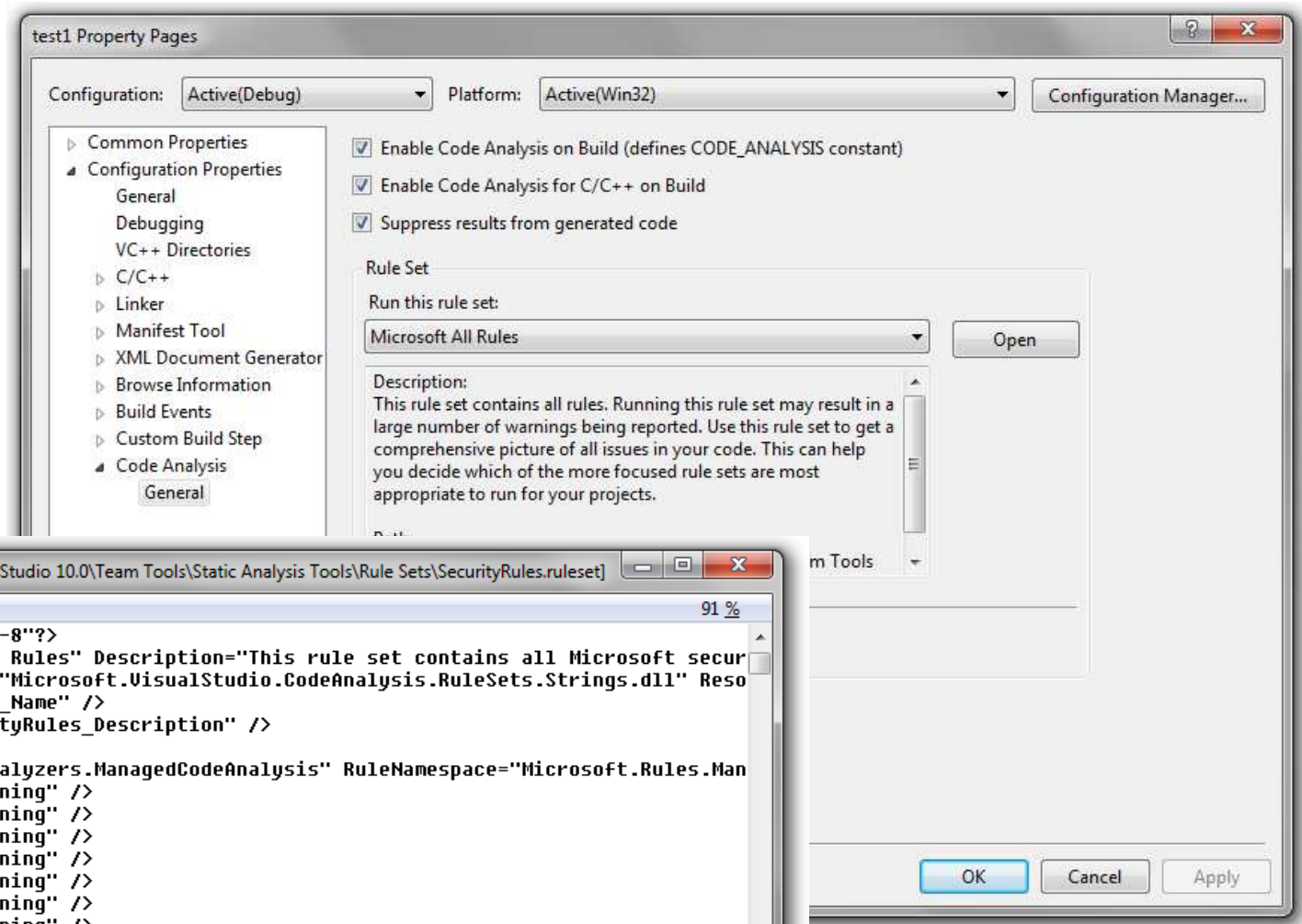
- Process of security code review
 - <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01668009>
- Why cryptosystems fail, R. Anderson
 - <http://www.cl.cam.ac.uk/~rja14/Papers/wcf.pdf>
- Software Security Code Review
 - <http://www.softwaremag.com/l.cfm?doc=2005-07/2005-07code>
- Static code analysis 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

- Many tools, free&commercial
 - gcc -Wall -Wextra
 - MSVS:Project→C/C++ →General →Warning level (/W4 /Wall)
 - MSVS2010: Analyze →Run code analysis (Code analysis settings)
 - Splint (C, Linux) <http://www.splint.org/>
 - Flawfinder (C/C++, Linux) <http://www.dwheeler.com/flawfinder/>
 - Cppcheck (C/C++, Windows) <http://cppcheck.sourceforge.net/>
- Call graphs Doxygen, <http://cecko.eu/public/doxygen>
- Problem of false positives
 - Problems that are not problems
- Problem of false negatives
 - Missed problems

MSVS2010 Code analysis

- <http://msdn.microsoft.com/en-us/library/ms182025.aspx>



Flawfinder

- Download at <http://www.dwheeler.com/flawfinder/>
- Build by `setup.py build`
- Install by `setup.py install`
- `/build/scripts***/flawfinder.py`
- `flawfinder.py --context --html source_dir`

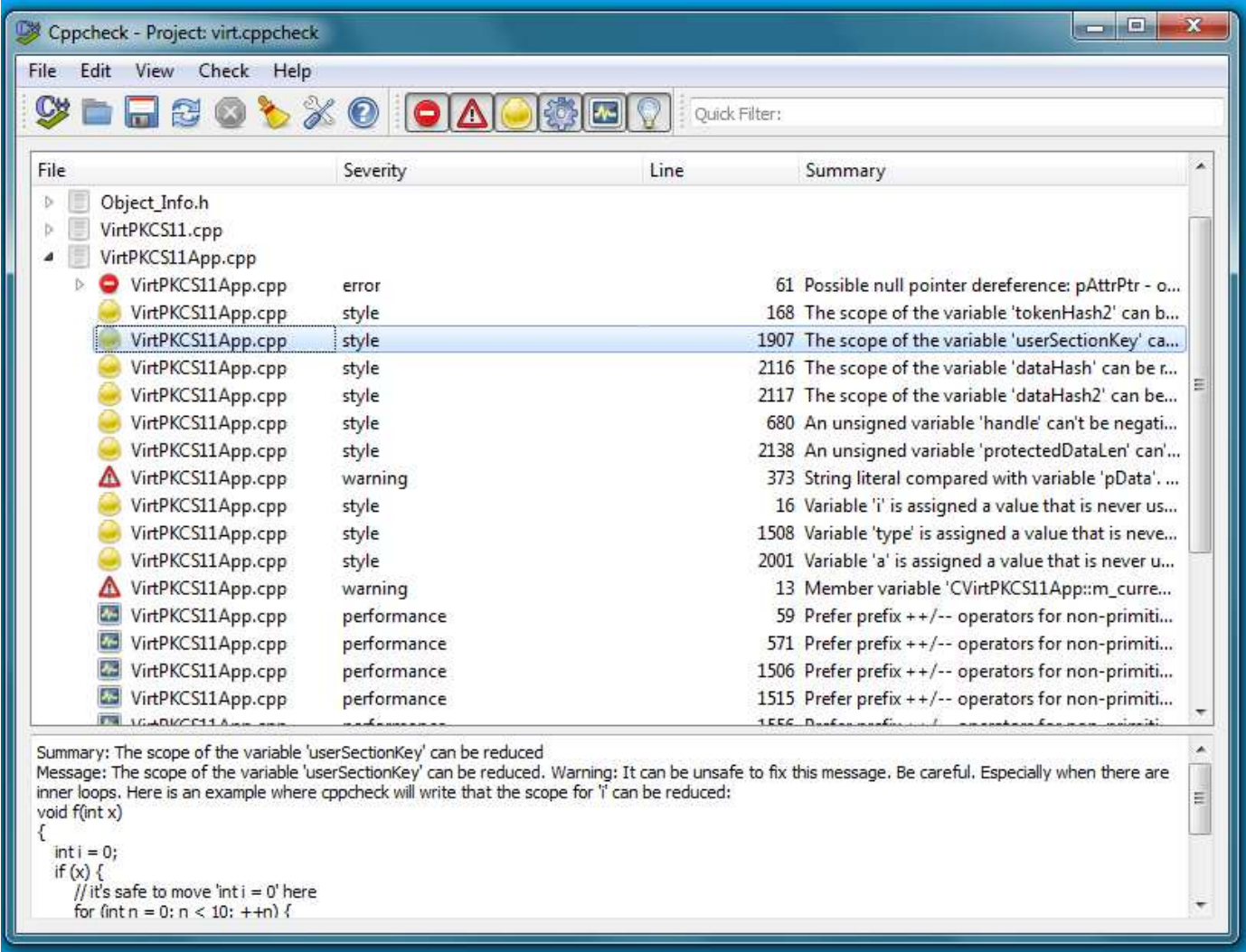
Flawfinder - example

```
strncat(d,s,10);
source\test.c:58: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated (it could cause a
crash if unprotected).
n = strlen(d);
source\test.c:64: [1] (buffer) MultiByteToWideChar:
Requires maximum length in CHARACTERS, not bytes. Risk is very low,
the length appears to be in characters not bytes.
MultiByteToWideChar(CP_ACP,0,szName,-1,wszUserName,sizeof(wszUserName)/sizeof(
wszUserName[0]));
source\test.c:66: [1] (buffer) MultiByteToWideChar:
Requires maximum length in CHARACTERS, not bytes. Risk is very low,
the length appears to be in characters not bytes.
MultiByteToWideChar(CP_ACP,0,szName,-1,wszUserName,sizeof wszUserName /sizeof(
wszUserName[0]));

Hits = 36
Lines analyzed = 117 in 0.93 seconds (273 lines/second)
Physical Source Lines of Code (SLOC) = 80
Hits@level = [0] 0 [1] 9 [2] 7 [3] 3 [4] 10 [5] 7
Hits@level+ = [0+] 36 [1+] 36 [2+] 27 [3+] 20 [4+] 17 [5+] 7
Hits/KSLOC@level+ = [0+] 450 [1+] 450 [2+] 337.5 [3+] 250 [4+] 212.5 [5+] 87.5
Suppressed hits = 2 (use --neverignore to show them)
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!

C:\Program Files\Flawfinder\build\scripts-2.5>flawfinder.py --context source
```

Cppcheck



Practical assignment

- Every team will make its own documentation & code available online
 - send link to me - now
- Other teams will make security analysis of the architecture and code (2 projects)
- Points will be awarded according to:
 - number and severity of the problems found in reviewed projects
 - quality of own 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>)
 - Splint (C, Linux) <http://www.splint.org/>
 - Flawfinder (C/C++, Linux) <http://www.dwheeler.com/flawfinder/>
 - Cppcheck (C/C++, Windows) <http://cppcheck.sourceforge.net/>

Practical assignment (2)

- Summarize your findings
 - problem identification + severity + applicability + short description
 - 2 pages enough

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)