PB173 - Tématický vývoj aplikací v C/C++ (podzim 2013)

Skupina: Aplikovaná kryptografie a bezpečné programování

https://is.muni.cz/auth/el/1433/podzim2013/PB173/index.qwarp?fakulta=1433;obdobi=5983;predmet=734514;prejit=2957738;

Petr Švenda svenda@fi.muni.cz

Konzultace: A.406, Úterý 15-15:50

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

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)
  - 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/
  - ...

Cppcheck

- A tool for static C/C++ code analysis
  - Open-source freeware, http://cppcheck.sourceforge.net/
- Last version 1.61 (2013-08-03)
- Used to find bugs in open-source projects (Linux kernel...)
- Command line & GUI version
- Standalone version, plugin into IDEs, version control...
  - Code::Blocks, Codelite, Eclipse, Jenkins...
  - Tortoise SVN
  - not Visual Studio ☹
- Cross platform (Windows, Linux)
  - **`sudo apt-get install cppcheck`**

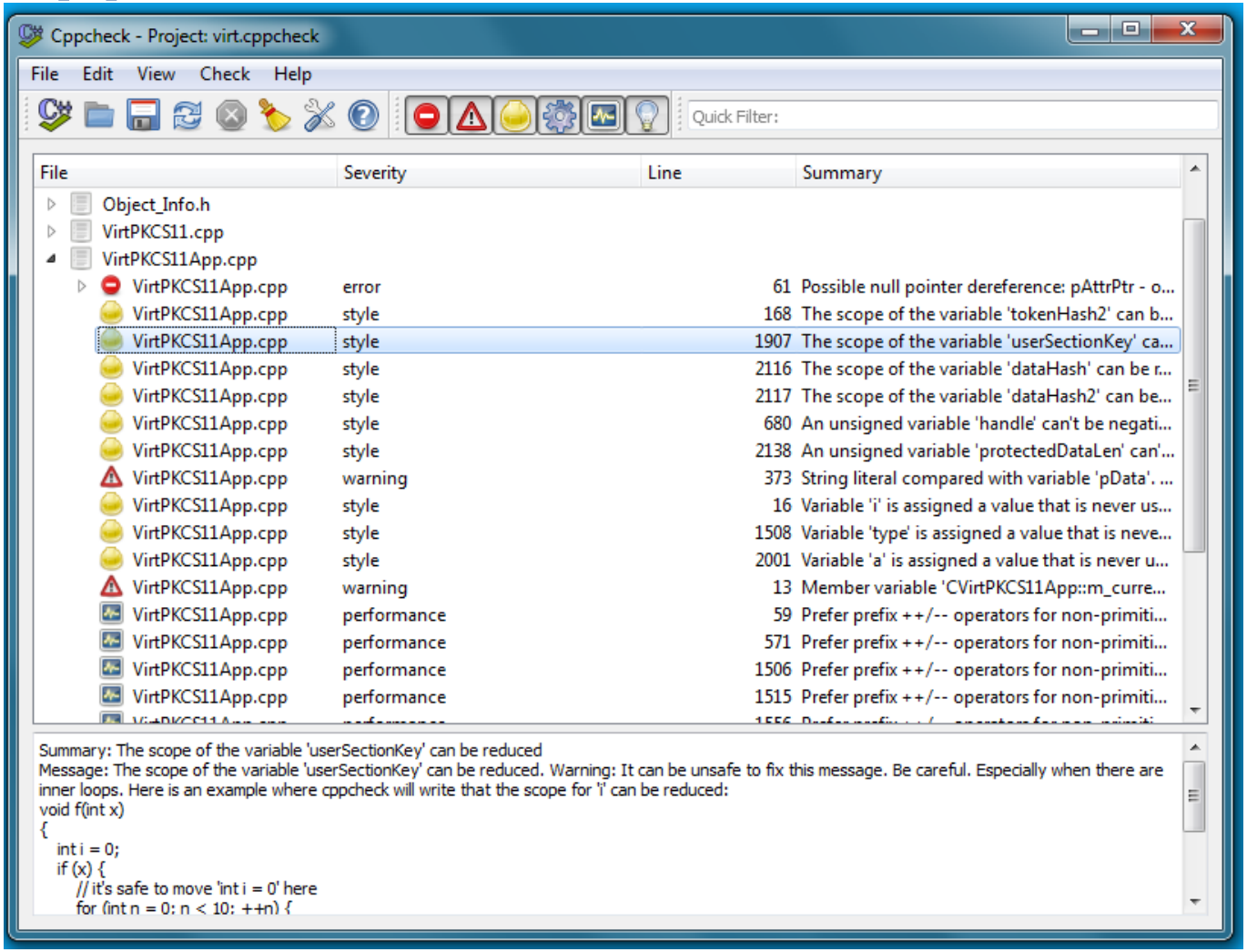
Cppcheck – what is checked?

- Bound checking for array overruns
- Suspicious patterns for class
- Exceptions safety
- Memory leaks
- Obsolete functions
- sizeof() related problems
- String format problems...
- See full list http://sourceforge.net/apps/mediawiki/cppcheck/index.php?title=Main_Page#Checks

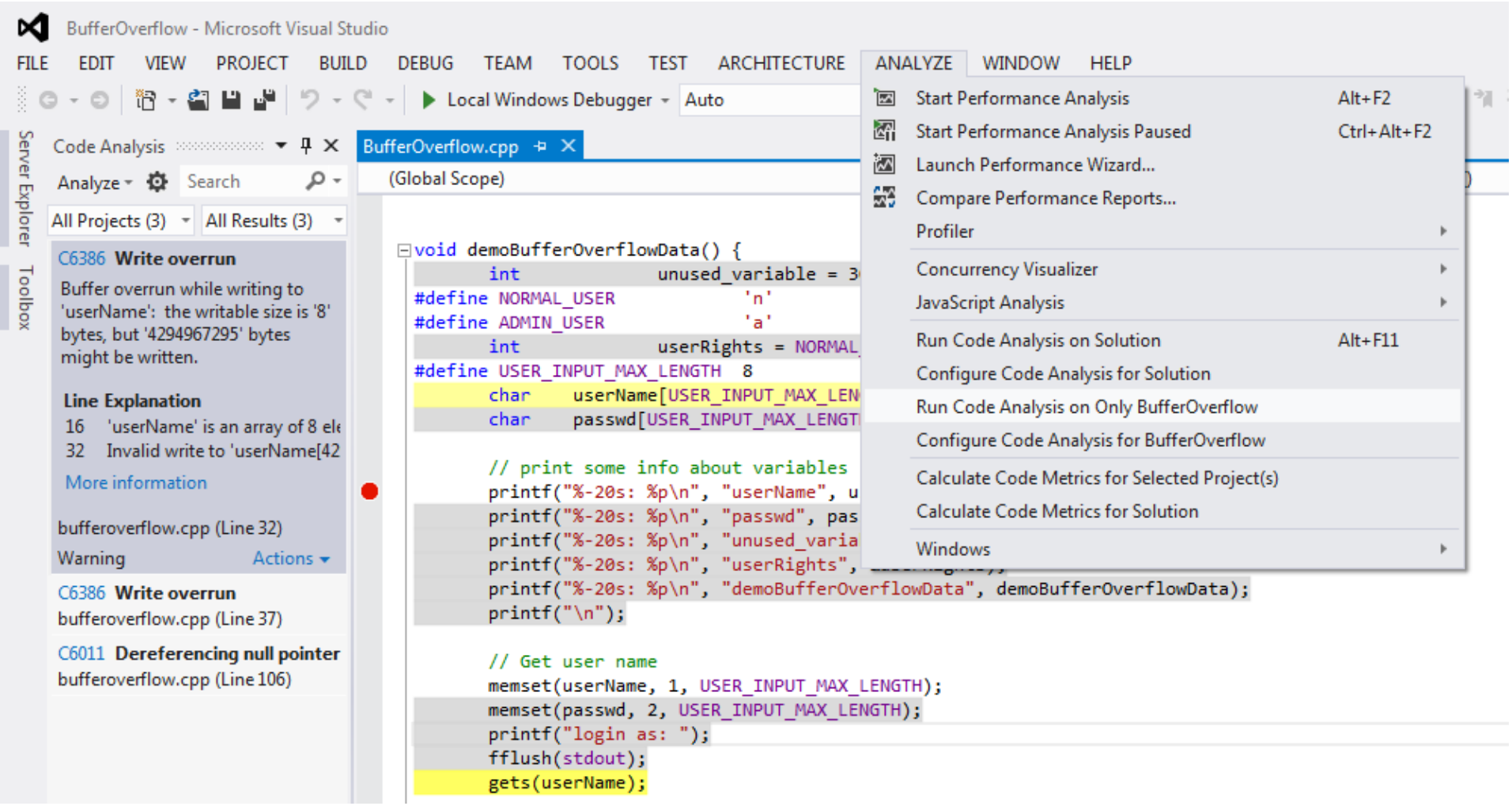
Cppcheck – categories of problems

- error – when bugs are found
- warning - suggestions about defensive programming to prevent bugs
- style - stylistic issues related to code cleanup (unused functions, redundant code, constness...)
- performance - suggestions for making the code faster.
- portability - portability warnings. 64-bit portability. code might work different on different compilers. etc.
- information - Informational messages about checking problems

Cppcheck


Cppcheck - Project: virt.cppcheck
File Edit View Check Help
Quick Filter:
File Severity Line Summary
Object_Info.h
VirtPKCS11.cpp
VirtPKCS11App.cpp
VirtPKCS11App.cpp error 61 Possible null pointer dereference: pAttrPtr - o...
VirtPKCS11App.cpp style 168 The scope of the variable 'tokenHash2' can b...
VirtPKCS11App.cpp style 1907 The scope of the variable 'userSectionKey' ca...
VirtPKCS11App.cpp style 2116 The scope of the variable 'dataHash' can be r...
VirtPKCS11App.cpp style 2117 The scope of the variable 'dataHash2' can be...
VirtPKCS11App.cpp style 680 An unsigned variable 'handle' can't be negati...
VirtPKCS11App.cpp style 2138 An unsigned variable 'protectedDataLen' can'...
VirtPKCS11App.cpp warning 373 String literal compared with variable 'pData'. ...
VirtPKCS11App.cpp style 16 Variable 'i' is assigned a value that is never us...
VirtPKCS11App.cpp style 1508 Variable 'type' is assigned a value that is neve...
VirtPKCS11App.cpp style 2001 Variable 'a' is assigned a value that is never u...
VirtPKCS11App.cpp warning 13 Member variable 'CVirtPKCS11App::m_curre...
VirtPKCS11App.cpp performance 59 Prefer prefix ++/-- operators for non-primiti...
VirtPKCS11App.cpp performance 571 Prefer prefix ++/-- operators for non-primiti...
VirtPKCS11App.cpp performance 1506 Prefer prefix ++/-- operators for non-primiti...
VirtPKCS11App.cpp performance 1515 Prefer prefix ++/-- operators for non-primiti...
Summary: The scope of the variable 'userSectionKey' can be reduced
Message: The scope of the variable 'userSectionKey' can be reduced. Warning: It can be unsafe to fix this message. Be careful. Especially when there are inner loops. Here is an example where cppcheck will write that the scope for 'i' can be reduced:
void f(int x)
{
int i = 0;
if (x) {
// it's safe to move 'int i = 0' here
for (int n = 0; n < 10; ++n) {

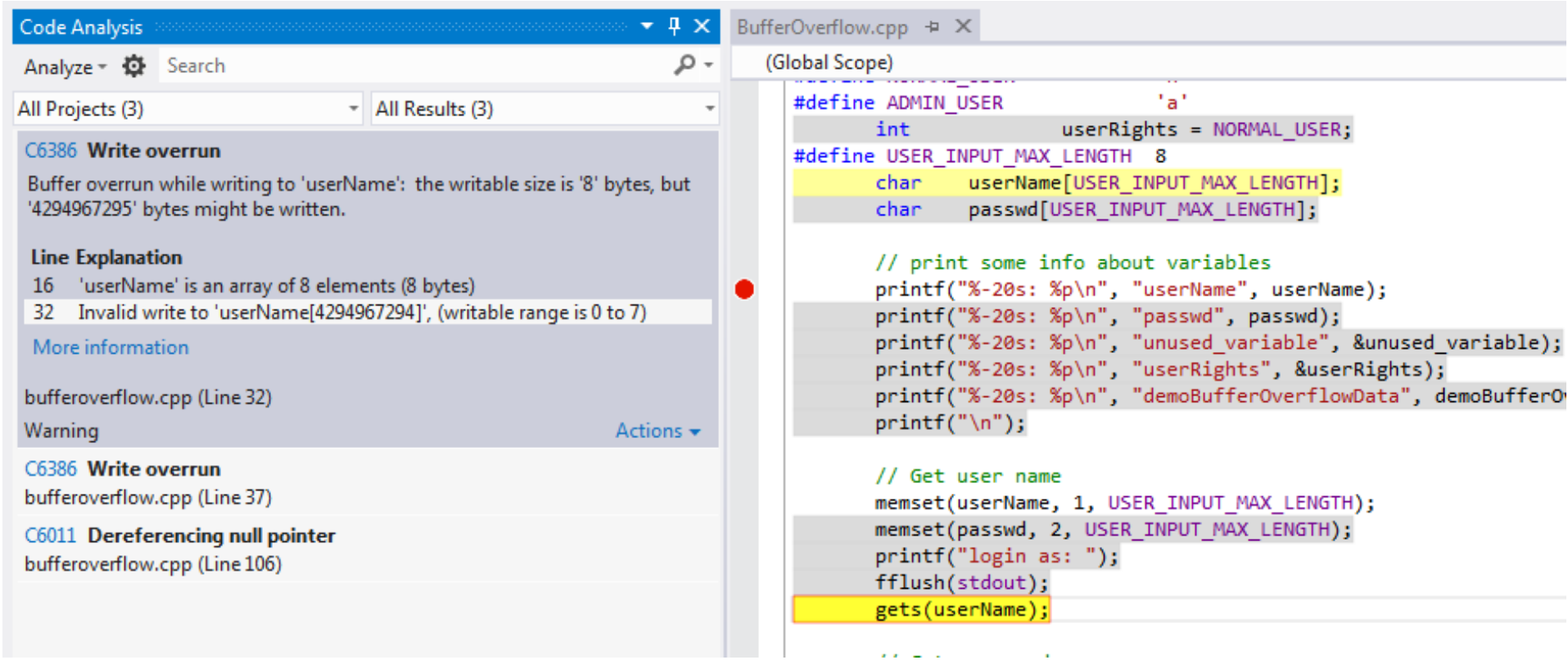
PREfast - Microsoft static analysis tool



PREfast - Microsoft static analysis tool

- Visual Studio Ultimate and Premium Editions
- Documentation for PREfast
  - http://msdn.microsoft.com/en-us/library/windows/hardware/gg487351.aspx
- PREfast tutorial
  - http://www.codeproject.com/Articles/167588/Using-PREfast-for-Static-Code-Analysis
  - http://www.cs.auckland.ac.nz/~pgut001/pubs/sal.html
- Can be enabled on every build
  - not enabled by default, time consuming
- Can be extended by source code annotation (SAL)

PREfast – example bufferOverflow

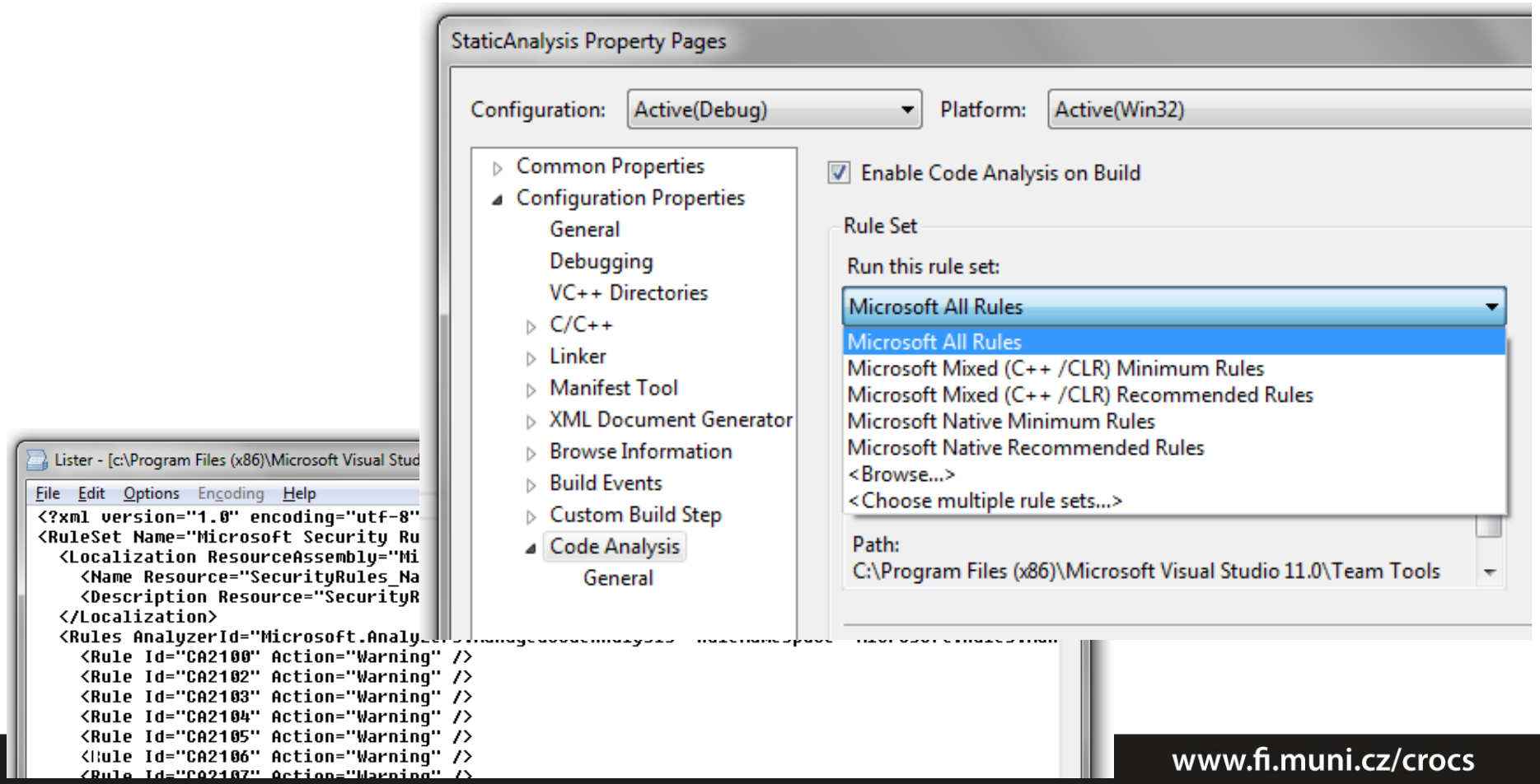


PREfast – what can be detected

- Potential buffer overflows
- Memory leaks, uninitialized variables
- Excessive stack usage
- Resources – release of locks...
- Incorrect usage of selected functions
- List of all code analysis warnings
  http://msdn.microsoft.com/en-us/library/a5b9aa09.aspx

PREfast settings (VS 2012)

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



Practical assignment

- Every team will make its own documentation & code available online
  - upload to IS repository (available to others)
  - deadline 2.12. 20:00
- Other teams will make security analysis of the architecture and code (3 other projects)
  - after 3.12. 12:00
- Points will be awarded according to:
  - number&severity of problems found in reviewed projects
  - quality of own architecture and code

Practical assignment

- Summarize your findings and prepare presentations
  - problem identification + severity + applicability + short description
  - 2 pages enough (per project)
  - Submit before 9.12.2014 12:00
- Present your findings (5-10 minutes presentation)

Problem identification: A_x (security architecture) / C_x (code, implementation)
Severity: low / middle / high / not decidable
Practicability: easy (directly by external attacker) / depends on other parts of the system / cannot decide (potential flaw, but attack unknown yet)
Description of the problem: description
Proposed solution: simple description (in case we know some)

Practical assignment (how to start?)

- 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/
  - ...