

PA193 - Secure coding principles and practices

LABS: Static analysis of source code



Petr Švenda svenda@fi.muni.cz

CRCS

Centre for Research on
Cryptography and Security

Overview - Lab

- Goal: Learn how to use basic tools
- Discuss false positives / false negatives

- Check C/C++ code with compiler warnings
- Check C/C++ code with VS PReFast
- Check C/C++ code with CppCheck
- Check Java code with FindBugs

MS Visual Studio: Warnings and PRefast

- Set project warning level to /W4 (or /Wall)
 - Run and compile `bufferOverflowDemo.cpp`
 - (don't forget: new project must be created)
 - Fix all warnings for clean compilation in VS /W4
- Run Code analysis on `bufferOverflowDemo.cpp`
 - Analyze→Run code analysis on ...
 - You need have Project selected inside Project explorer (otherwise Run code analysis... option will not appear)
 - Try difference between 'minimum' and 'all rules'
- Try at home: `gcc -Wall -Wextra`

BufferOverflow - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug x86 Local Windows Debugger

BufferOverflow.cpp

```

#include "stdio.h"
#include "string.h"
#include "stdlib.h"
#include "memory.h"
// Note: GCC and MSVC use different definitions for these macros
// Try "12345678DevilEve"
// Try "1234567812345678"

void demoBufferOverflowData()
{
    int unused;

#define NORMAL_USER
#define ADMIN_USER
    int user;
#define USER_INPUT_MAX_LENGTH 100
    char userName[USER_INPUT_MAX_LENGTH];
    char passwd[USER_INPUT_MAX_LENGTH];

    // print some info about user
    printf("%-20s: %p\n", "userName", userName);
    printf("%-20s: %p\n", "passwd", passwd);
    printf("%-20s: %p\n", "user", &user);
    printf("%-20s: %p\n", "unused", &unused);
}
    
```

BufferOverflow Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

- Configuration Properties
 - General
 - Debugging
 - VC++ Directories
 - C/C++
 - General
 - Optimization
 - Preprocessor
 - Code Generation
 - Language
 - Precompiled Headers
 - Output Files
 - Browse Information
 - Advanced
 - All Options
 - Command Line
 - Linker

Additional Include Directories	
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	EnableAllWarnings (/Wall)
Treat Warnings As Errors	Turn Off All Warnings (/W0)
SDL checks	Level1 (/W1)
Multi-processor Compilation	Level2 (/W2)
	Level3 (/W3)
	Level4 (/W4)
	EnableAllWarnings (/Wall)
	<inherit from parent or project defaults>

BufferOverflow.cpp

BufferOverflow (Global Scope) demoBufferOverflowData()

BufferOverflow Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

- Configuration Properties
 - General
 - Debugging
 - VC++ Directories
 - C/C++
 - Linker
 - Manifest Tool
 - XML Document Generator
 - Browse Information
 - Build Events
 - Custom Build Step
 - Code Analysis
 - General

Enable Code Analysis on Build

Rule Set

Run this rule set:

- Microsoft Native Recommended Rules
- Microsoft All Rules
- Microsoft Mixed (C++ /CLR) Minimum Rules
- Microsoft Mixed (C++ /CLR) Recommended Rules
- Microsoft Native Minimum Rules
- Microsoft Native Recommended Rules
- <Browse...>
- <Choose multiple rule sets...>

Path:
C:\Program Files\Microsoft Visual Studio 14.0\Team Tools\Static Analysis Tools\Rule Sets\NativeRecommendedRules.ruleset

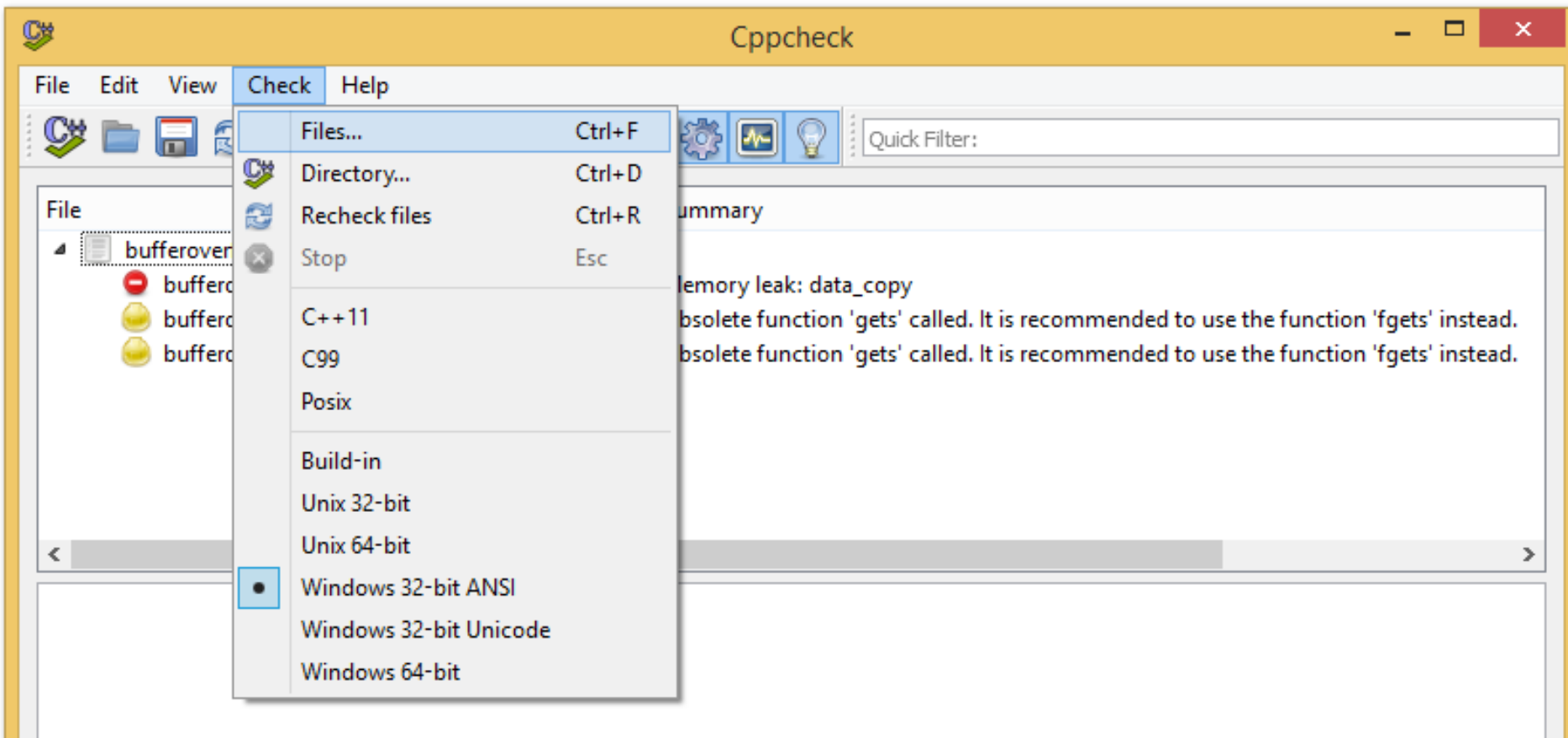
Open [Learn more about rule sets](#)

Questions (inspect whole BODemo.cpp)

- What is the difference between /W3 and PREFast analysis?
- Why you should compile without warning?
- Are all bugs caught by static analysis?
- Which bugs are not caught?

Cppcheck

1. Download Cppcheck and unpack (or install)
2. Use Cppchcek against bufferOverflow.cpp
 - run command line, `cppcheck bufferOverflow.cpp`
 - `cppcheck --enable=all bufferOverflow.cpp`
3. Setup Cppcheck GUI viewer for Cppcheck
 - (Notepad++ is already predistributed on lab computers or download at <http://sourceforge.net/projects/notepadpp-usb/>)
 - Edit → Preferences → Applications → Add
 - Executable: "C:\Program Files\Notepad++\notepad++.exe"
 - Parameters: -n(line) (file)
4. Run Cppcheck GUI and analyze files or directories



FindBugs/FindSecurityBugs - Java

- Download FindBugs <http://findbugs.sourceforge.net/>
- Download FindSecurityBugs (plugin)
 - <https://find-sec-bugs.github.io/download.htm>
 - copy findsecbugs-plugin-1.5.0.jar into FindBugs\plugin\ directory
 - List of patterns: <https://h3xstream.github.io/find-sec-bugs/bugs.htm>
- Run FindBugs\bin\findbugs.bat (on Windows)
 - Or directly FindBugs\lib\findbugs.jar
 - Enable plugin in Edit → Preferences → Plugins

FindBugs/FindSecurityBugs - Java

- Note: you need compiled *.jar for analysis
 - And source code for quick display of problems 😊
- Extract content of [crypto-java.zip](#)
- Run FindBugs
- Start analysis
 - **File** → **New project**
 - **Classpath for analysis**: select target *.jar file (crypto_java.jar)
 - **Source directories**: select parent dir of target package
 - crypto-java\src\main\java\ in our case

Reconfigure

Project name
java-crypto

Classpath for analysis (jar, ear, war, zip, or directory) [Help](#)
C:\temp\Java Security-master\crypto-java\out\artifacts\crypto_java\crypto_java.jar

Auxillary classpath (optional; classes referenced by analysis classpath) [Help](#)

Source directories (optional; used when browsing found bugs) [Help](#)
C:\temp\Java Security-master\crypto-java\src\main\java

Store bug reviews in:
<default>

File Edit View Navigation Designation Help

Class name filter: Filter

Group bugs by: Bug Pattern Category Bug Kind ↔ Bug Rank Designation

- ECB Mode Unsafe (2)
 - Security (2)
 - ECB Mode (2)
 - The cipher chosen uses ECB mode, which provides poor confidentiality f
 - The cipher chosen uses ECB mode, which provides poor confidentiality f
- Hard Coded Key (1)
- Method may fail to clean up stream or resource (3)
- Method may fail to close stream (1)

No cloud selected Enable cloud plugin...

[Click to add review...](#)

unclassified

Saved Cancel

```

AES.java in de.dominikschadow.javasecurity.symmetric
67     Key key = ses.loadKey(ks, keyAlias, keyPa
68     SecretKeySpec secretKeySpec = new Secret
69     byte[] ciphertext = ses.encrypt(secretKey
70     byte[] plaintext = ses.decrypt(secretKey
71
72     ses.printReadableMessages(initialText, c
73
74     byte[] secretKey = {1, 2, 3, 4, 5, 6, 7,
75     SecretKeySpec spec = new SecretKeySpec(s
76     Cipher aes = Cipher.getInstance("AES");
77     aes.init(Cipher.ENCRYPT_MODE, spec);
78     byte[] encrypted = aes.doFinal(initialTe
79
80     } catch (NoSuchPaddingException | NoSuchAlgo
81             KeyStoreException | CertificateExcep
82             InvalidAlgorithmParameterException |
83             InvalidKeyException | IOException ex
84             LOGGER.error(ex.getMessage(), ex);
    
```

The cipher chosen uses ECB mode, which provides poor confidentiality
 At AES.java:[line 76]
 In method de.dominikschadow.javasecurity.symmetric.AES.main(String[])

ECB Mode Unsafe

An authentication cipher mode which provides better confidentiality of the encrypted data should be used instead of ECB mode, which does not provide good confidentiality. Specifically, ECB mode produces the same output for the same input. For example, if you are sending a password, the encrypted value is the same each time. This allows an attacker to intercept and compare encrypted values. To fix this, something like Galois/Counter Mode (GCM) should be used instead.

Code at risk:

```
Cipher c = Cipher.getInstance("AES/ECB/NoPadding");
```

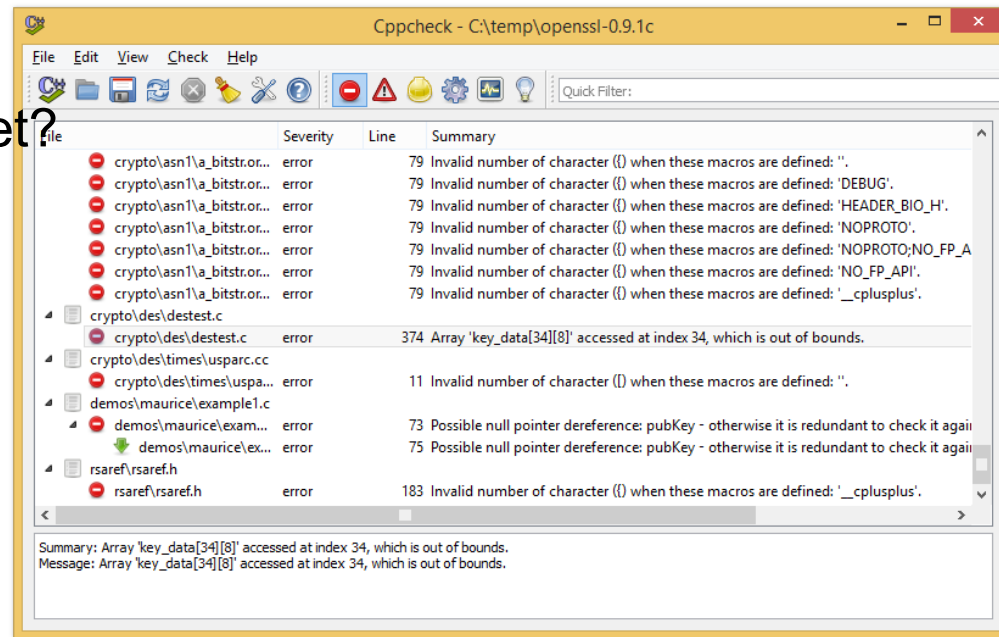
Questions: FindBugs & FindSecurityBugs

- Which issues were found?
- Are all reported issues from project source code?
- How you would rate severity of different issues?
- How can you use FindBugs in team collaboration?

- Is FindBugs working on source code or compiled code? Compare to CppCheck.

CPPCheck + OpenSSL

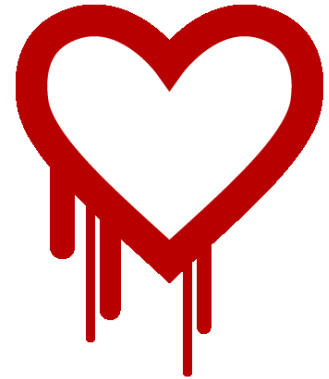
- Run against OpenSSL0.9.1c (1998)
 - <https://www.openssl.org/source/old/0.9.x/>
 - What are the bugs?
- Run against newest OpenSSL
 - <ftp://ftp.openssl.org/source/>
 - Why not completely clean yet?



Questions

- Which bugs are found in `bufferOverflowDemo.cpp`? Compare to PReFast in Visual Studio.
- Which bugs are found in old OpenSSL?
- Are style warnings important?

Hearthbleed bug



- OpenSSL 1.0.1 through 1.0.1f
- Download <https://www.openssl.org/source/openssl-1.0.1e.tar.gz>
- Locate function `dtls1_process_heartbeat(SSL *s)`
 - `Ssl\t1_lib.c`
- Will your static analyzers find anything?
 - Don't be sad, even Coverity didn't before bug was exposed
 - <http://security.coverity.com/blog/2014/Apr/on-detecting-heartbleed-with-static-analysis.html>

Assignment

- No new assignment this week 😊
- But project group, selection, start with static analysis