PV286+PA193 - Secure coding principles and practices

Overview of the subject(s)

Łukasz Chmielewski *chmiel@fi.muni.cz (email me with your questions/feedback)*
Centre for Research on Cryptography and Security, Masaryk University
Consultation hours: Friday 9.30-11.00 in A406 (but email me before).

CRoCS
Centre for Research on Cryptography and Security

PV286+PA193: Secure coding principles and practices

- Main goal: secure coding
  - How to write code in a more secure way
  - So that the program is harder to be attacked/exploited
  - Selected basic building blocks of security applications
- PV286: > 80 students
  - Lecture: 2 hours weekly on Wednesdays
  - Project: about 30-40 hours/person
- PA193: < 40 students
  - Seminar: 2 hours weekly, usually corresponding to the lecture, on Thursdays
  - Homework: about 6-? hours/each
- In case of questions: please email me!
  - I will address all questions at the beginning of next lecture

PV286+PA193: Secure coding principles and practices

- PV286 project – more in the presentation by Jan Kvapil
- For everyone following PA193: you have to also follow PV286!
- PA193 is more practical with hands-on exercises and homeworks.
  - There are still some places to register for that course.

People

- Main contact: Łukasz Chmielewski (CRoCS@FI MU)
  - Office hours: Friday 9:30-11:00, A406
  - chmiel@fi.muni.cz,
  - https://keybase.io/grasshoppper
  - @chmiel:fi.muni.cz
- Other lectures, seminars, and the project
  - Kamil Dudka (Red Hat), Václav Lorenc (HERE Technologies), Marek Sýs (FI), Lukas Rucka (FI), Martin Čarnogurský (RootLUG), Lumir Honus (AT&T).
  - Project: Jan Kvapil, Milan Šorf, Roman Lacko, Štěpánka Trnková, Jiří Gavenda, Tomáš Jaroš, and Antonín Dufka.

PV286: planned lectures (+ HW only for PA193) tentative

21.2.	Language level vulnerabilities: Buffer overflow, type overflow, strings (Łukasz Chmielewski) +**HW**
28.2.	Security testing: static analysis (Łukasz Chmielewski)
6.3.	Security testing: dynamic analysis (Łukasz Chmielewski) +**HW**
13.3.	Static and dynamic analysis @ RedHat (Kamil Dudka) and Legal Implications (Pavel Loutocký)
20.3.	Integrity of modules, parameters, and temporary files (Lukas Rucka) +**HW**
27.3.	Programming in the presence of side channels / faults (Łukasz Chmielewski)
3.4.	Programming with trusted hardware, Securing API, automata-based programming (Ł. Ch.) +**HW**
10.4.	Defense in depth (Lukas Rucka)
17.4.	Supply-chain attacks, 3rd party libs security, patch management (Martin Čarnogurský)
24.4.	Cloud programming security (AWS, Azure..) (Lumir Honus)
1.5. (**V**)	(Pseudo) Random Data (Marek Sýs)
8.5. (**V**)	Code review (Łukasz Chmielewski) +**HW**
15.5.	Threat Modelling (Václav Lorenc)

+ Project Presentations (contact person: Jan Kvapil)

Aims of the subject

- To learn how to program in a way that the resulting application is more secure
  - Decrease number of security related bugs
  - Increase difficulty of exploitation
- To understand security consequences of decisions made by programmer
- Most issues are independent on particular programming language
  - examples will be mostly based on C/C++ and Java

Previous knowledge requirements

- Basic knowledge of (applied) cryptography and IT security
  - symmetric vs. asymmetric cryptography, PKI
  - block vs. stream ciphers and usage modes
  - hash functions
  - random vs. pseudorandom numbers
  - basic cryptographic algorithms (AES, DES, RSA, EC, DH)
  - risk analysis
- Basic knowledge in formal languages and compilers
- User-level experience with Windows and Linux OS
- **Practical experience with C/C++/Java language**
- More is required for seminars (PA193) but the exam and the project will require that too!

Organization

- **PV286** = Lectures + project + exam
- Project
  - **Team work** (2-3 members)
  - Details by Jan Kvapil later
- Exam
  - Written exam, open questions, pencil-only
- **PA193** = corresponding seminars + assignments
  - 6 homework assignments
  - **Individual work of each student**

Grading PV286

- Points [Notice minimal number of points required!]
  - Questionnaire from lectures (10) [no minimum limit]
  - Project (45) – [minimum 23 required]
  - Exam (45) – [must known basics] + 95% correct from drill questions
  - Occasional bonuses ☺
- Grading 100 (max)
  - A ≥ 90
  - B ≥ 80
  - C ≥ 70
  - D ≥ 60
  - E ≥ 50
  - F < 50
  - Z ≥ 50 (including minimum numbers from the Project)
- About PA193:
  - 60% points from the assignments
  - More at the first seminar

Attendance

- Lectures (**PV286**)
  - Attendance not obligatory, but highly recommended
  - I will try to record giving the lectures but that is not guaranteed and depend on the teacher
  - 2 lectures will be only available in video form due to public holidays
  - For some lectures, old pre-recorded lecture videos are in IS
  - 1-2 hour lecture on selected topics + Q&A (depends on the teacher)
- Assignments and projects (**PV286**)
  - Done during student free time (e.g. at a dormitory)
  - Access to network lab and CRoCS lab possible
- Seminars (**PA193**)
  - Attendance **obligatory**
  - Absences must be excused at the department of study affairs
  - 2 absences are OK (even without excuse)

Discussion forum in Information System

- Discussion forum in Information System (IS)
  - https://is.muni.cz/auth/discussion/predmetove/fi/jaro2024/PV286/
  - https://is.muni.cz/auth/discussion/predmetove/fi/jaro2024/PA193/
- Mainly for discussion among the students
  - Not observed by stuff all the time!
  - Write us an email if necessary please
- What to ask?
  - OK to ask about ambiguities in assignment
  - NOT OK to ask for the solution
  - NOT OK to post your own code and ask what is wrong

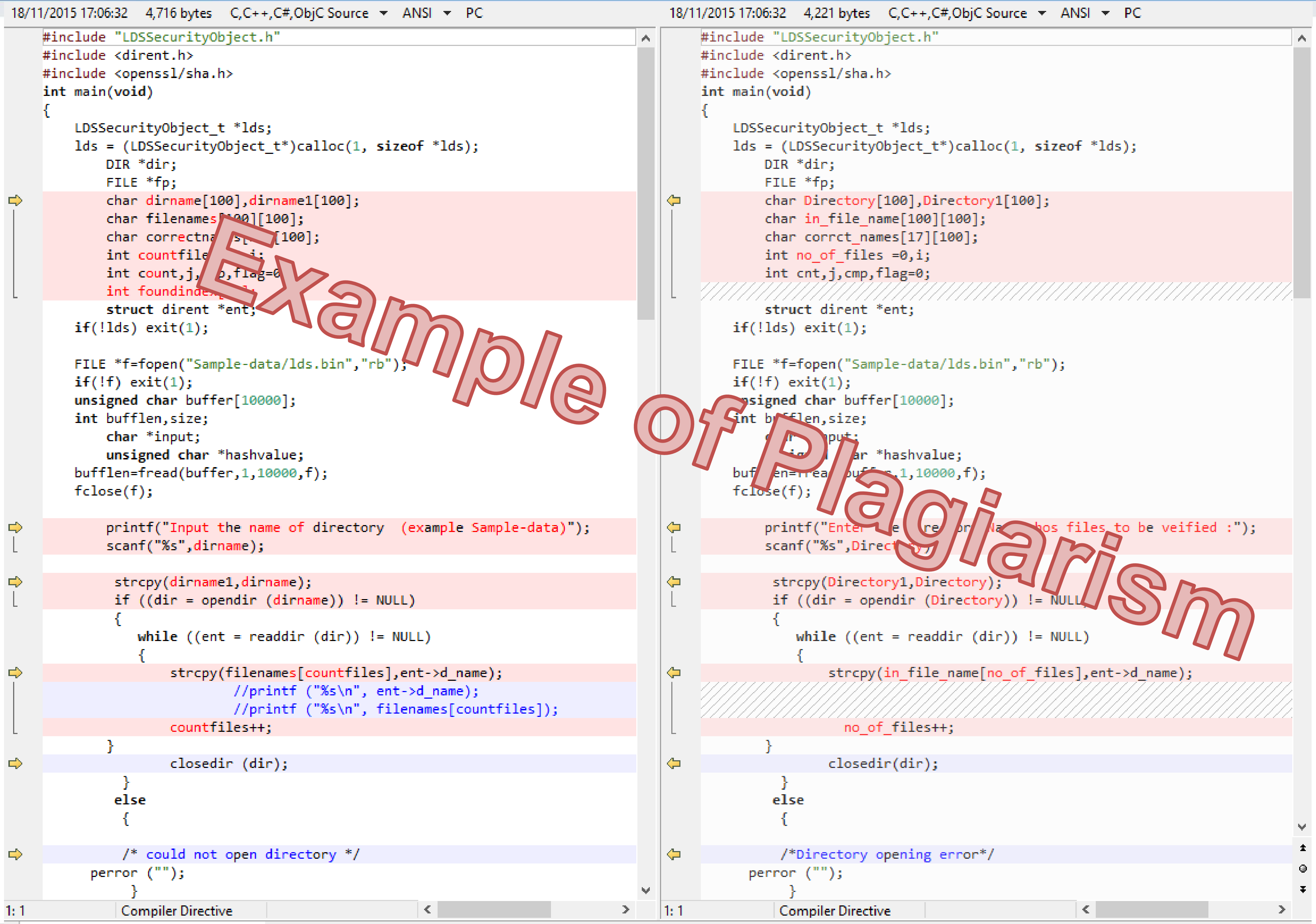
Plagiarism

- Homework assignments
  - Must be worked out independently by each student
- Projects
  - Must be worked out by a team of 3 students
  - Every team member must show his/her contribution
- Plagiarism, cut&paste, etc. is not tolerated
  - Plagiarism is use of somebody else words/programs or ideas without proper citation
  - Automatic tools used to recognize plagiarism
  - If plagiarism is detected student is assigned -7 points
  - More serious cases handled by the Disciplinary committee

Reuse of existing code

- Code reuse is generally great thing, but..
- NOT in homework or assignments!
- It is NOTOK:
  - Take any code from web when you should create code completely on your own (project - parser)
  - Share code of your solution with others (homework)

CROCS



Example of Plagiarism

18/11/2015 17:06:32   4,716 bytes   C,C++,C#,ObjC Source   ANSI   PC

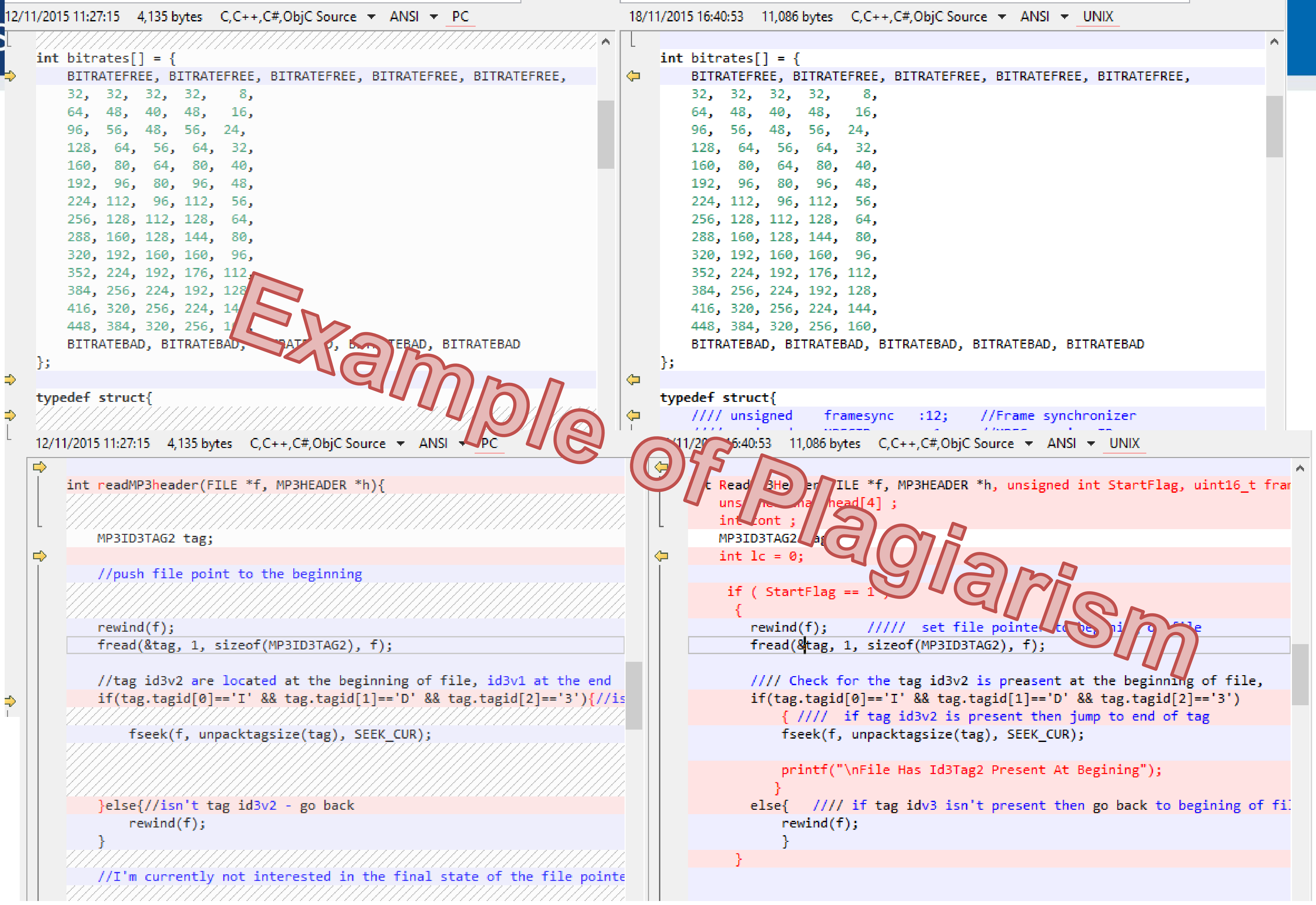
```
#include "LDSSecurityObject.h"
#include <dirent.h>
#include <openssl/sha.h>
int main(void)
{
    LDSSecurityObject_t *lds;
    lds = (LDSSecurityObject_t*)calloc(1, sizeof *lds);
        DIR *dir;
        FILE *fp;
        char dirname[100],dirname1[100];
        char filenames[[illegible]00][100];
        char correctna[illegible]s[[illegible]][100];
        int countfile[illegible]i;
        int count,j,[illegible]p,flag=0[illegible]
        int foundinde[illegible];
        struct dirent *ent;
    if(!lds) exit(1);

    FILE *f=fopen("Sample-data/lds.bin","rb");
    if(!f) exit(1);
    unsigned char buffer[10000];
    int bufflen,size;
        char *input;
        unsigned char *hashvalue;
    bufflen=fread(buffer,1,10000,f);
    fclose(f);

        printf("Input the name of directory  (example Sample-data)");
        scanf("%s",dirname);

         strcpy(dirname1,dirname);
         if ((dir = opendir (dirname)) != NULL)
         {
            while ((ent = readdir (dir)) != NULL)
            {
                strcpy(filenames[countfiles],ent->d_name);
                        //printf ("%s\n", ent->d_name);
                        //printf ("%s\n", filenames[countfiles]);
                countfiles++;
        }
                closedir (dir);
          }
         else
          {

          /* could not open directory */
       perror ("");
           }
```

1: 1   Compiler Directive

18/11/2015 17:06:32   4,221 bytes   C,C++,C#,ObjC Source   ANSI   PC

```
#include "LDSSecurityObject.h"
#include <dirent.h>
#include <openssl/sha.h>
int main(void)
{
    LDSSecurityObject_t *lds;
    lds = (LDSSecurityObject_t*)calloc(1, sizeof *lds);
        DIR *dir;
        FILE *fp;
        char Directory[100],Directory1[100];
        char in_file_name[100][100];
        char corrct_names[17][100];
        int no_of_files =0,i;
        int cnt,j,cmp,flag=0;

        struct dirent *ent;
    if(!lds) exit(1);

    FILE *f=fopen("Sample-data/lds.bin","rb");
    if(!f) exit(1);
    [illegible]nsigned char buffer[10000];
    int b[illegible]fflen,size;
        [illegible]r[illegible]nput;
        [illegible]ar *hashvalue;
    buf[illegible]en=frea[illegible]buff[illegible].1,10000,f);
    fclose(f);

        printf("Ente[illegible]e[illegible]re[illegible]or[illegible]Na[illegible]hos files to be veified :");
        scanf("%s",Direct[illegible]);

         strcpy(Directory1,Directory);
         if ((dir = opendir (Directory)) != NULL[illegible]
         {
            while ((ent = readdir (dir)) != NULL)
            {
                strcpy(in_file_name[no_of_files],ent->d_name);


                  no_of_files++;
        }
                closedir(dir);
          }
         else
          {

          /*Directory opening error*/
       perror ("");
           }
```

1: 1   Compiler Directive



Example of Plagiarism

```
12/11/2015 11:27:15    4,135 bytes    C,C++,C#,ObjC Source    ANSI    PC

int bitrates[] = {
    BITRATEFREE, BITRATEFREE, BITRATEFREE, BITRATEFREE, BITRATEFREE,
    32,  32,  32,  32,   8,
    64,  48,  40,  48,  16,
    96,  56,  48,  56,  24,
    128,  64,  56,  64,  32,
    160,  80,  64,  80,  40,
    192,  96,  80,  96,  48,
    224, 112,  96, 112,  56,
    256, 128, 112, 128,  64,
    288, 160, 128, 144,  80,
    320, 192, 160, 160,  96,
    352, 224, 192, 176, 112,
    384, 256, 224, 192, 128
    416, 320, 256, 224, 14
    448, 384, 320, 256, 1
    BITRATEBAD, BITRATEBAD, [illegible]BAT[illegible]), B[illegible]TEBAD, BITRATEBAD
};

typedef struct{
```

```
18/11/2015 16:40:53    11,086 bytes    C,C++,C#,ObjC Source    ANSI    UNIX

int bitrates[] = {
    BITRATEFREE, BITRATEFREE, BITRATEFREE, BITRATEFREE, BITRATEFREE,
    32,  32,  32,  32,   8,
    64,  48,  40,  48,  16,
    96,  56,  48,  56,  24,
    128,  64,  56,  64,  32,
    160,  80,  64,  80,  40,
    192,  96,  80,  96,  48,
    224, 112,  96, 112,  56,
    256, 128, 112, 128,  64,
    288, 160, 128, 144,  80,
    320, 192, 160, 160,  96,
    352, 224, 192, 176, 112,
    384, 256, 224, 192, 128,
    416, 320, 256, 224, 144,
    448, 384, 320, 256, 160,
    BITRATEBAD, BITRATEBAD, BITRATEBAD, BITRATEBAD, BITRATEBAD
};

typedef struct{
    //// unsigned    framesync    :12;    //Frame synchronizer
```

```
12/11/2015 11:27:15    4,135 bytes    C,C++,C#,ObjC Source    ANSI    PC

int readMP3header(FILE *f, MP3HEADER *h){

    MP3ID3TAG2 tag;

    //push file point to the beginning

    rewind(f);
    fread(&tag, 1, sizeof(MP3ID3TAG2), f);

    //tag id3v2 are located at the beginning of file, id3v1 at the end
    if(tag.tagid[0]=='I' && tag.tagid[1]=='D' && tag.tagid[2]=='3'){//is

        fseek(f, unpacktagsize(tag), SEEK_CUR);

    }else{//isn't tag id3v2 - go back
        rewind(f);
    }

    //I'm currently not interested in the final state of the file pointe
```

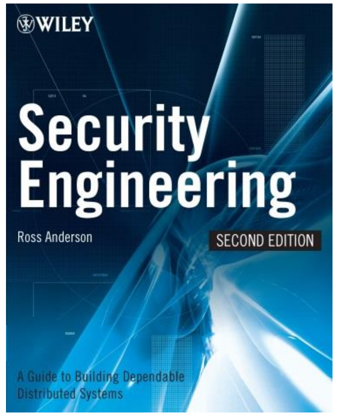
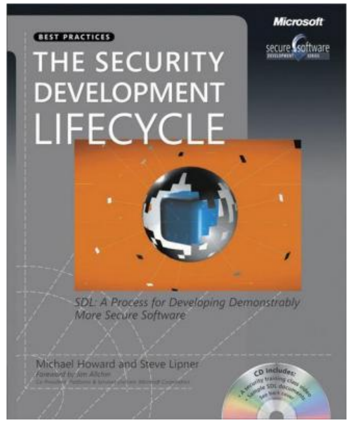
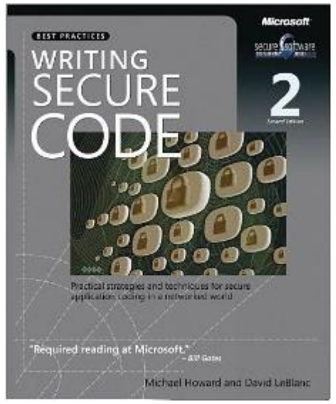
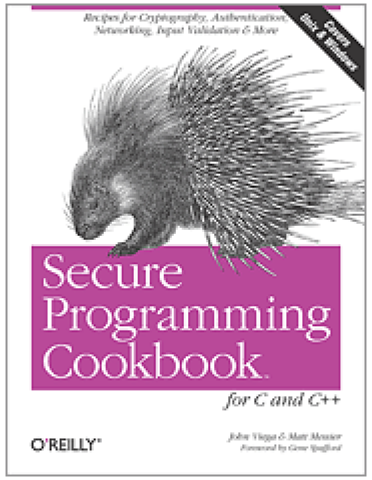
```
[illegible]/11/20[illegible]6:40:53    11,086 bytes    C,C++,C#,ObjC Source    ANSI    UNIX

t Read[illegible]3He[illegible]er[illegible]ILE *f, MP3HEADER *h, unsigned int StartFlag, uint16_t fra
    uns[illegible]head[4] ;
    int [illegible]ont ;
    MP3ID3TAG2 [illegible]ag
    int lc = 0;

     if ( StartFlag == 1 [illegible]
      {
        rewind(f);     /////  set file pointe[illegible] file
        fread(&tag, 1, sizeof(MP3ID3TAG2), f);

        //// Check for the tag id3v2 is preasent at the beginning of file,
        if(tag.tagid[0]=='I' && tag.tagid[1]=='D' && tag.tagid[2]=='3')
            { ////  if tag id3v2 is present then jump to end of tag
            fseek(f, unpacktagsize(tag), SEEK_CUR);

            printf("\nFile Has Id3Tag2 Present At Begining");
           }
        else{   //// if tag idv3 isn't present then go back to begining of fi
            rewind(f);
            }
      }
```

Course resources

- Lectures (video, PDF) available in IS
  - IS = Information System of the Masaryk University
  - Lecture questionares in IS opened till end of Monday
- Assignments (what to do) available in IS
  - Submissions done also via IS (homework Vault)
- Additional tutorials/papers/materials from time to time will also be provided in IS
  - To better understand the issues discussed
- Recommended literature
  - To learn more …

Recommended literature

- Ross Anderson - Security engineering, Wiley
- Michael Howard, Steve Lipner - Secure Development Lifecycle, MS Press
- John Viega, Matt Messier - Secure programming cookbook, O'Reilly
- Michael Howard - Writing secure code, MS Press

Questions