PA193 Secure coding principles and practices

Overview of the subject

Petr Švenda, Petr Ročkai, Milan Patnaik, Marek Sýs,
Kamil Dudka, Mirek Jaroš, Martin Ukrop, Jan Masarik

CROCS
Centre for Research on Cryptography and Security

PA193 Secure coding principles and practices

- 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
- 2/2/2
  - Lecture: 2 hours weekly
  - Seminar: 2 hours weekly
  - Homework: about 6-? hours/each
  - Project: about 30-40 hours/person

People

- Main contact: Petr Švenda (CRoCS@FI MU)
  - Office hours: Tuesday 13:00-13:50, A406


svenda@fi.muni.cz,  @rngsec


https://keybase.io/petrs


https://crocs.fi.muni.cz/people/svenda

- Other lectures and seminars
  - Milan Patnaik (DRDO) Marek Sýs (FI), Jan Masrik (Kiwi) Kamil Dudka (RedHat), Mirek Jaroš (RedHat), Martin Ukrop (FI)

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**

Organization

- Lectures + seminars + assignments + project + exam
- Assignments
  - 10 homework assignments
  - **Individual work of each student**
  - Lab A403 available to students (except teaching hours)
- Project
  - **Team work** (2-3 members)
  - Details next week (cryptowallet derivation, CI, fuzzing…)
- Exam
  - Written exam, open questions, pencil-only

Grading

- Credits
  - 2+2+2 credits, plus 2 for the final exams
- Points [Notice minimal number of points required!]
  - Homework (50) – [minimum 25 required]
  - Project (20) – [minimum 10 required]
  - Written exam (50) – [minimum 25 limit]
  - Occasional bonuses ☺
  - TLDR: must get at least half points from each area
- Grading 120 (max)
  - A ≥ 110
  - B ≥ 100
  - C ≥ 90
  - D ≥ 80
  - E ≥ 65
  - F < 65

Attendance

- Lectures
  - Attendance not obligatory, but highly recommended
- Seminars
  - Attendance **obligatory**
  - Absences must be excused at the department of study affairs
  - 2 absences are OK (even without excuse)
- Assignments and projects
  - Done during student free time (e.g. at the dormitory)
  - Access to network lab and CRoCS lab is possible

Discussion forum in Information System

- Discussion forum in Information System (IS)
  - https://is.muni.cz/auth/cd/1433/podzim2019/PA193/
- Mainly for discussion among the students
  - Not observed by stuff all the time!
  - Write us email if necessary
- 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

- Homeworks
  - 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)

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

```c
#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[100][100];
        char correctna[100];
        int countfile;
        int count,j,i,flag=0
        int foundinde
        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 ("");
          }
```

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

```c
#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);
    unsigned char buffer[10000];
    int bufflen,size;
        char *input;
        unsigned char *hashvalue;
    bufflen=fread(buffer,1,10000,f);
    fclose(f);

        printf("Enter the name of directory  (example Sample-data) to be verified :");
        scanf("%s",Directory);

         strcpy(Directory1,Directory);
         if ((dir = opendir (Directory)) != NULL)
         {
            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

1: 1 Compiler Directive

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,  BITRATEBAD, BITRATEBAD, 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
```

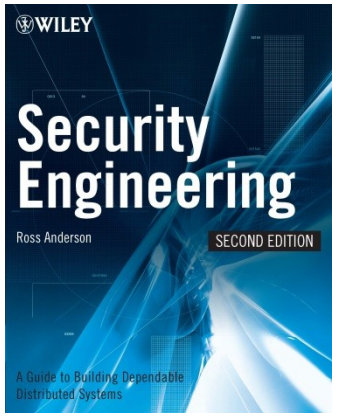
11/20 16:40:53 11,086 bytes C,C++,C#,ObjC Source ANSI UNIX

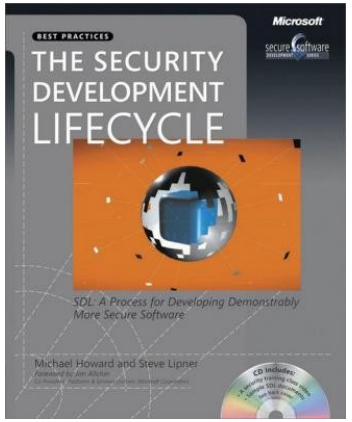
```
t Read 3He er ILE *f, MP3HEADER *h, unsigned int StartFlag, uint16_t fra
    uns  head[4] ;
    int cont ;
    MP3ID3TAG2 ag
    int lc = 0;

     if ( StartFlag == 1 )
      {
        rewind(f);      /////  set file pointe to be, ni  o  ile
        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

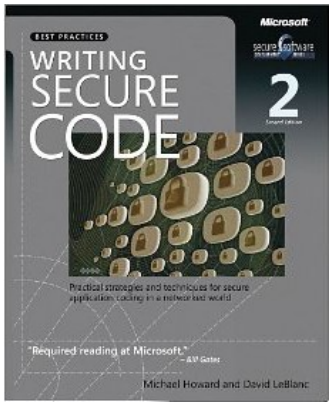
- Lectures (PDF) available in IS
  - IS = Information System of the Masaryk University
  - https://is.muni.cz/auth/el/1433/podzim2019/PA193/
- Homeworks/assignments available in IS
  - Submissions also done via IS (Homework vaults)
- Additional tutorials/papers/materials from time to time will also be provided in IS
  - To better understand the issues discussed
- Recommended literatures
  - To learn more …

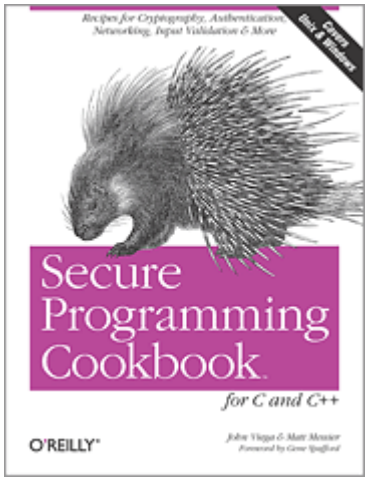
Recommended literature





- 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





Lectures and content

- 16. 9. Intro, Language level vulnerabilities: Buffer overflow, type overflow, strings (PS)
- 23. 9. Security testing: blackbox vs. whitebox testing, static analysis (PS)
- 30. 9. Security testing: dynamic analysis, fuzzing (PS)
- 7. 10. Security code review, automata-based programming, securing API (PS, KD)
- 14. 10. Exploits writing (MP)
- 21. 10. Return-oriented Programming (MP)

PS – Petr Švenda, KD – Kamil Dudka, MP – Milan Patnaik

Lectures and content

National holidays, no lecture

- 28. 10. Security primitives: secure channel, secure storage, key management (PS)
- 4. 11. Web programming security, 3rd party libs security, patch management (JM)
- 11. 11. Integrity of modules, parameters, temp files (PR)
- 18. 11. Proper use of (pseudo)-random data (MS)
- 25. 11. Defense in depth (PR)
- 2. 12. Concurrent issues: IPC, race conditions (PR)
- 9. 12. Access control, privilege separation (PR)

PS – Petr Švenda, JM – Jan Masarik, PR – Petr Ročkai, MS – Marek Sýs

Questions