PA193 - Secure coding principles and practices

Overview of the subject

Petr Švenda *svenda@fi.muni.cz* *@rngsec*
Centre for Research on Cryptography and Security, Masaryk University

CROCS
Centre for Research on Cryptography and Security

Top questions (1)

Anonymous 0

Is information disclosure vulnerability relevant for heap and dynamically allocated memory if language has garbage collection?

Join at

slido.com

#pa193_2022

- **Place/upvote questions in slido while listening to lecture video**
- **We will together discuss these during every week lecture Q&A (every Monday)**

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: Friday 8:30-11:00, A403

svenda@fi.muni.cz, @rngsec

https://keybase.io/petrs

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

- Other lectures and seminars
  - Lukasz Chmielewski (FI), Milan Patnaik (DRDO), Marek Sýs (FI), Jan Masarik (Facebook) Kamil Dudka (Red Hat), Mirek Jaroš (Red Hat), Martin Ukrop (FI), Antonín Dufka (FI)

Planned lectures (tentative)

14.2. Language level vulnerabilities: Buffer overflow, type overflow, strings (Petr Svenda)
21.2. Security testing: dynamic analysis, fuzzing (Petr Svenda)
28.2. Security testing: static analysis (Lukasz Chmielewski, Kamil Dudka)
7.3. Programming in the presence of side-channels / faults (Lukasz Chmielewski)
14.3. Securing API, automata-based programming (Petr Svenda)
21.4. Code review (Lukasz Chmielewski)
28.4. (Pseudo)Random Data (Marek Sys)
4.4. Integrity of modules, parameters, temp files (Lukas Rucka)
11.4. Defense in depth (Lukas Rucka)
18.4. Web security, 3rd party libs security, patch management (Jan Masarik)
25.4. Return Oriented Programming (Milan Patnaik)
2.5. Cloud programming security (AWS, Azure..) (Lumir Honus)
9.5. Project presentation (Antonin Dufka)

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**

Organization

- Lectures + seminars + assignments + project + exam
- Assignments
  - 6 homework assignments
  - **Individual work of each student**
  - Lab A403 available to students (except teaching hours)
- Project
  - **Team work** (2-3 members)
  - Details by Antonin Dufka later (bech32m parser…)
- Exam
  - Written exam, open questions, pencil-only

Grading

- Credits: 2+2+2 credits, plus 2 if exam
- Points [Notice minimal number of points required!]
  - Questionnaire from lectures (10) [no minimum limit]
  - Assignments (30) – [minimum 15 required]
  - Project (30) – [minimum 15 required]
  - Exam (30) – [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 Assignments and Project)

Attendance

- Lectures
  - Attendance not obligatory, but highly recommended
  - For some lectures, pre-recorded lecture video in IS (from Friday)
  - 1-2 hour lecture on selected topics + Q&A (depends on the teacher)
- 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 a dormitory)
  - Access to network lab and CRoCS lab possible

Discussion forum in Information System

- Discussion forum in Information System (IS)
  - https://is.muni.cz/auth/cd/1433/jaro2022/PA193/
- Mainly for discussion among the students
  - Not observed by stuff all the time!
  - Write us 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)

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,
        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 ("");
           }
```

1: 1 Compiler Directive

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 directory name of the files to be veified :");
        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

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
```

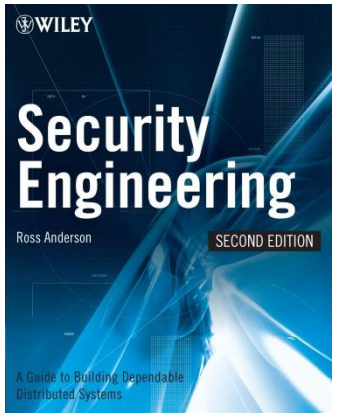
11/20 6:40:53 11,086 bytes C,C++,C#,ObjC Source ANSI UNIX

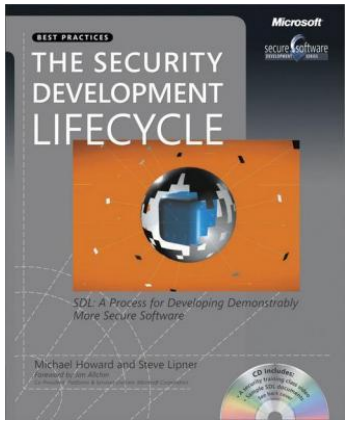
```
t Read 3He er ILE *f, MP3HEADER *h, unsigned int StartFlag, uint16_t frar
    uns  ha head[4] ;
    int cont ;
    MP3ID3TAG2 ag
    int lc = 0;

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

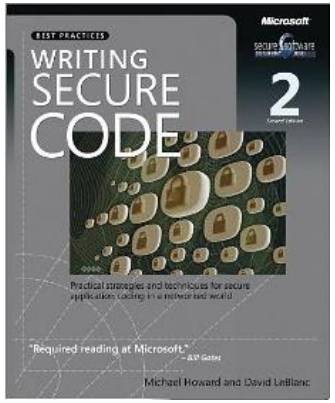
- Lectures (video, PDF) available in IS
  - IS = Information System of the Masaryk University
  - Lecture questionares in IS opened till end of Monday
- Assignments (what to do) available in IS
  - Submissions done also via IS (homework Vault)
- Additional tutorials/papers/materials from time to time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

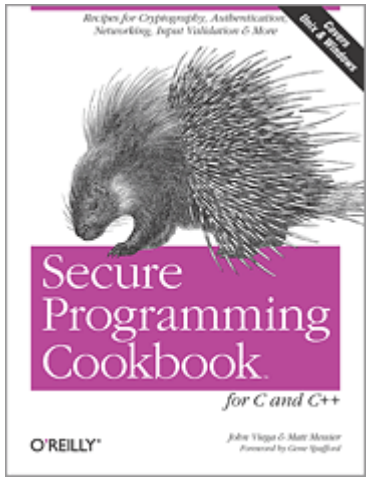
Recommended literature





- 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