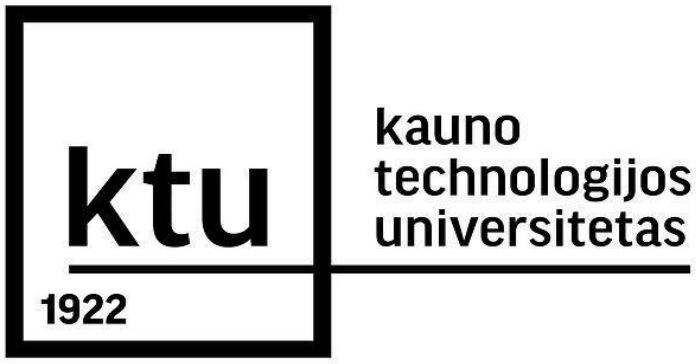


INFORMATIKOS FAKULTETAS

Programų sauga

(T120M153)

Laboratorinis darbas Nr.2

Atliko:
IFM-1/3 gr.
Stud. Eligijus Kiudys

Priemė:
Prof. Jevgenijus Toldinas
Doc. Jonas Čeponis

KAUNAS 2021

Turinys

Paveikslų sąrašas

1. Įvadas

Laboratorinis darbas: pasirinktos spragos analizė ir statinės kodo analizės įrankio panaudojimas.

Darbo tikslas: susipažinti su statinės kodo analizės įrankiu ir šio įrankio pagalba aptikti esamas kodo spragas. Atlikus laboratorinį darba turėtų buti sėkmingai aptiktos ir ištaisytos kodo sprągos, kurios nėra aptinkamos kompiliavimo metu.

1.1. Laboratorinio darbo užduotis

1.1.1. Parašyti programą kurioje būtų pasirinkta kodo spraga.
1.1.2. Naudojant statinės kodo analizės įrankį, aptikti kodo klaidą ir ją ištaisyti

1.2. Įrankių pasirinkimas

1.2.1. Programavimo aplinkos programa: Clion 2021.2.3
1.2.2. Programavimo kalba: C++
1.2.3. Pasirinktas kodo analizės įrankis: SonarLint
1.2.4. Užduotis: 2. Buferio kopijavimas netikrinant ilgio (Buffer Copy without Checking Size of Input).

2. Pasirinktos užduoties aprašymas

Buferio kopijavimas netikrinant ilgio (angl. Buffer Copy without Checking Size of Input) – įvyksta kai yra kopijuojama vieno masyvo reikšmes į kitą masyva naudojant pointerius (angl. pointer).

Dažniausios situcijos:
1. Įrašant žodius į masyva naudojant konsolės įvestį.
2. Kopijuojant iš vieno masyvo reikšmes į kitą.

Bufferio kopijavimas netikrinant ilgio gali paveikti programą tokiais būdais:
1. Programa gali netikėtai sustoti.
2. Gali būti panaudotas piktavališkas kodas programai nežinant.
3. Sugadinti programos naudojamą atminį.
4. Ataka kuri programą priveda prie begalinio ciklo.

Problemos stadymas: kopijuojant iš buferio į kita buferį naudoti funkciją kuri leidžia limituoti kopijuojamo bufferio dydį arba naudoti kitokį kintamajį kuris dinamiškai plečiasi.

3. Demonstracinė programa

Pasirinkos užduoties sprendimui buvo sukurta programa (pav. 1, pav. 2). Programa turi buferi pavadinimu „Last_name“, kurio talpa yra 9, tada yra sukuriami dar du kintamieji „Answer“ ir „Last_name_Without_Overflow“ į kuriuos yra nukopijuojamos konsolės įvestis. Įrašius įvestis yra nuskaitoma informacija iš sukurto failo ir įrašoma į buferį, kurio ilgis yra lygus failo dydžiui. Nuskaityto failo buferis yra nukopijuojamas su nustatytu ilgiu ir be nustatyto ilgio. Nukopijavus buferius rezultatai yra atspauzdinami.

```cpp
// LD 2. Buferio kopijavimas netikrinant ilgio (Buffer Copy without Checking Size of Input).

#include <iostream>
#include <fstream>
#include <string>

void ReadFromFile(char fileName[]);

const int value = 196; //file char count

int main() {

    char Last_name[9];
    std::string Answer;
    std::string Last_name_Without_Overflow;

        std::cout << "Enter your name first time: " << std::endl;
        std::cin >> Last_name_Without_Overflow;
        std::cout << "Enter your name first time: " << std::endl;
        std::cin >> Last_name;
        std::cout << "First read result: " << Last_name_Without_Overflow << " Second read result: " << Last_name << std::endl;
        std::cout << "size comparison: " << Last_name_Without_Overflow.size() << " vs " << sizeof(Last_name) << std::endl;

    ReadFromFile( fileName: "FileToRead.txt");


    return 0;
}
```

pav. 1 Pirma dalis demonstracinio kodo

```cpp
void ReadFromFile(char* fileName)
{
    std::fstream newfile;

    newfile.open(fileName);
    if (!newfile.is_open())
    {
        std::cout << "error opening file" << std::endl;
    }
    const int size = 171;
    char achData[value];
    newfile.get(achData, _Count: value-1);
    char dataCopied[size];
    strncpy_s( &: dataCopied, achData, _Count: size-1);
    char achDataCopied[size];
    strcpy_s( &: achDataCopied, achData);
    std::cout << "size comparison: " << sizeof(achData) << " vs " << sizeof(achDataCopied) << " vs " <<  sizeof(dataCopied) << std::endl;
    std::cout << "text comparison: " << std::endl;
    std::cout << achDataCopied << std::endl;
    std::cout << dataCopied << std::endl;
}
```

pav. 2 Antra dalis demonstracinio kodo

Programa susikompiliuoja ir pasileidžia be problemų (pav. 3). Priklausomai nuo įrašytu žodžių programa veiks kaip turėtų veikti arba programa veiks nenuspėjamai. Jei antras įvestas žodis yra ilgesnis negu 8 raidės tada programos atmintis susigadins (pav. 4). Kitas atvėjis kai programa susigadins yra kopijuojant iš failo buferio į pasirinktą buferį kurio ilgis yra mažesnis ir kopijavimo ilgis nėra nustatytas (pav. 5).

```
"C:\Users\Eligijus\Desktop\KTU\Saugos Magistro studijos\ProgramuSauga\cmake-build-debug\ProgramuSauga.exe"
Enter your name first time:
Eligijus
Enter your name first time:
Eligijus
First read result: Eligijus Second read result: Eligijus
size comparison: 8 vs 9
size comparison: 196 vs 196 vs 171
text comparison:
Sveiki, cia yra laboratonis darbas bufferio perpildymas. bufferio perpildymas ivyksta kai yra aprasytas limituoto kiekio
 masyvas i kuri yra nuskaitoma nelimituoto kiekio tekstas. gggggggggaaaaa
Sveiki, cia yra laboratonis darbas bufferio perpildymas. bufferio perpildymas ivyksta kai yra aprasytas limituoto kiekio
 masyvas i kuri yra nuskaitoma nelimituoto kiekio

Process finished with exit code 0
```

pav. 3 Programos veikimas su klaida

```
"C:\Users\Eligijus\Desktop\KTU\Saugos Magistro studijos\ProgramuSauga\cmake-build-debug\ProgramuSauga.exe"
Enter your name first time:
EligijusEligijus
Enter your name first time:
EligijusEligijus
First read result: EligijusEligijus Second read result: EligijusEligijus
size comparison: 16 vs 9
size comparison: 196 vs 196 vs 171
text comparison:
Sveiki, cia yra laboratonis darbas bufferio perpildymas. bufferio perpildymas ivyksta kai yra aprasytas limituoto kiekio
 masyvas i kuri yra nuskaitoma nelimituoto kiekio tekstas. gggggggggaaaaa
Sveiki, cia yra laboratonis darbas bufferio perpildymas. bufferio perpildymas ivyksta kai yra aprasytas limituoto kiekio
 masyvas i kuri yra nuskaitoma nelimituoto kiekio

Process finished with exit code -1073740791 (0xC0000409)
```

pav. 4 Programos veikimas su klaida

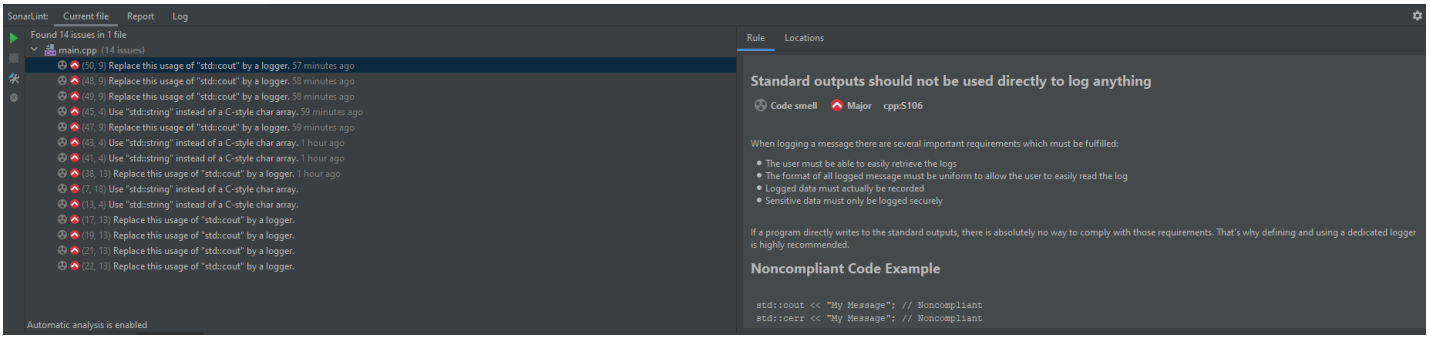
```
"C:\Users\Eligijus\Desktop\KTU\Saugos Magistro studijos\ProgramuSauga\cmake-build-debug\ProgramuSauga.exe"
Enter your name first time:
Eligijus
Enter your name first time:
Eligijus
First read result: Eligijus Second read result: Eligijus
size comparison: 8 vs 9

Process finished with exit code -1073740791 (0xC0000409)
```

pav. 5 Programos veikimas su klaida

4. Programos tvarkymas įrankių pagalba

Programos tvarkymui buvo pasirinktas SonarLint kodo statinės analizės įrankis, kurio pagalba galima sumažinti kodo spargų kiekį, kurių gali nematyti kompuliatorius. Pasirinktame įrankyje galima naudoti ir savo sukurtas taisykles, kurios padės papildomai palaikyti kodo tvarkygumą ir aptikti kodo spragas, kurių neaptinka įrankis. Įsirašome įrankį naudojant Clion įskiepių parduotuvę. Įsirašius įskiepį jį pasileidžiame ir analizuojame aptiktas klaidas (pav. 6).



pav. 6 SonarLint įrankio rezultatai

Yra pateikta keturioliką klaidų. Išanalizavus rezultatus matome tik dviejų skirtingų kodų S106, S5945 problemas. Visos klaidos priklauso vienam arba kitam tipui. Viena iš rodomų problemų yra char tipo masyvo naudojimas, šia problemą statinės kodo analziės įrankis siūlo išspręsti naudojant string tipo kintamjį. Įrankis siūlo pakeisti char masyvo tipo kintamajį į string tipo kintamajį. Taip būtų galma apsisauguoti nuo buferio kopijavimo be ilgio problemos. Tada programuotojas yra priverstas keisti kopijavimo finkciją jei naudoja char masyvą. Statinės kodo analizės įrankis taip pat surado kitą problemą kurios nepastebėjau. Įrankis siūlo pakeisti tiesioginį spauzdinimą į netiesioginį kuris yra saugesnis ir kuris padeda išlaikyti vienodą formatavimą.

Išanalizavus statinės kodo analizės įrankio pranešimus žinome ką reikia pakeisti, kad būtų sumažintas kodo spragų kiekis ir pagerintas kodo saugumas. Visi char masyvai yra pakeičiami į string tipo kintamuosius. Jei yra poreikis naudoti char masyvus ir reikia kopijuoti is vieno masyvo į kitą masyvą reikšmes galima limituoti kopijuojamo masyvo ilgį kopijavimo metu. Kita problema yra spendžiama saugiai kviečiant konsolės išspauzdinimą. Šia problemai reikėtų naudoti papildomą biblioteką kuri suteikia sugesnį būdą spauzdinti duomenis į konsolę.

```cpp
// LD 2. Buferio kopijavimas netikrinant ilgio (Buffer Copy without Checking Size of Input).

#include <iostream>
#include <fstream>
#include <string>

void ReadFromFile(const char* fileName, std::ostream& log_stream);
void ReadFromFileUsingStrings(const char* fileName, std::ostream& log_stream);

const int value = 196; //file char count

int main() {
    std::ostream& log_stream = std::cout;
    std::string Last_name;
    std::string Last_name_Without_Overflow;

    log_stream << "Enter your name first time: " << std::endl;
    std::cin >> Last_name_Without_Overflow;
    log_stream << "Enter your name first time: " << std::endl;
    std::cin >> Last_name;
    log_stream << "First read result: " << Last_name_Without_Overflow << " Second read result: " << Last_name << std::endl;
    log_stream << "size comparison: " << Last_name_Without_Overflow.size() << " vs " << sizeof(Last_name) << std::endl;

    ReadFromFile( fileName: "FileToRead.txt",  &: log_stream);
    log_stream << std::endl;
    ReadFromFileUsingStrings( fileName: "FileToRead.txt",  &: log_stream);

    return 0;
}
```

pav. 7 Programos kodas po pataisymų

```cpp
void ReadFromFile(const char* fileName, std::ostream& log_stream)
{
    std::fstream newfile;

    newfile.open(fileName);
    if (!newfile.is_open())
    {
        log_stream << "error opening file" << std::endl;
    }

    const int size = 171;
    char achData[value];
    newfile.get(achData,  _Count: value-1);
    char dataCopied[size];
    strncpy_s( &: dataCopied, achData,  _Count: size-1);
    log_stream << "text comparison: " << std::endl;

    log_stream << achData << std::endl;
    log_stream << dataCopied << std::endl;
}
```

pav. 8 Programos kodas po pataisymų

```
void ReadFromFileUsingStrings(const char* fileName, std::ostream& log_stream)
{
    std::ifstream in(fileName);
    std::string contents((std::istreambuf_iterator<char>( &: in)),
                         _Last: std::istreambuf_iterator<char>());

    std::string achData;
    achData = contents;
    contents[0] = 'a';

    log_stream << "text comparison: " << std::endl;
    log_stream << achData << std::endl;
    log_stream << contents << std::endl;
}
```

pav. 9 Programos kodas po pataisymų

Pataisius kodą ir paleidus naudojamą kodo analizės įrankį yra gaunami kurie yra pateikti paveikslėlyje (pav. 10)

pav. 10 SonarLint įrankio rezultatai po kodo keitimo

Iš pateiktų įrankio rezultatų matome, kad liko trys klaidos. Dvi viršutinės klaidos vis dar yra, kadangi nepakeičiau char masyvo tipo kintamąjį į string tipo kintamąjį, nes norėjau pataisytame kode parodyti, kad galima saugiai kopijuoti ir char masyvo tipo kintamuosius vieną į kitą. Kita klaida liko, nes reikia panaudoti biblioteką skirtą spauzdinimui į konsolę.

5. Išvados

Laboratorinio darb metu susipažinau su C++ statinės kodo analizės įrankiu „SonarLint“. Taip pat išmokau, kaip reikia pradėti naudotis statinės kodo analizės įrankiais ir kaip galima konfiguruoti įrankį padedant išlaikyti kodo vientysumą. Naudojamas statinės kodo analizės įrankis buvo sėkmingai panaudotas, pamatyti esamas klaidas parašytoje programoje ir jas dalinai išspręsti. Išsiaiškinau ir įsigilinau į buferio kopijavimo be ilgio problemą. Ši problema padėjo suprasti, kad yra svarbu pagerinti kodo saugumą. Taip pat pastebėjau, kad lengvai galima parašyti nesaugią programą kurią, galima lengvai pasinaudoti piktavališkiems tikslams, programuotojui ar naudotojui nežinant. Manau, kad Statinės kodo analizės įrankis yra labai naudingas, tiek pradedančiajam programuotojui, tiek patyrusiam. Pakoregavus kodą pagal statinės kodo analizės įrankio pasiūlymus galima labai lengvai pastebėti ir išspręsti esamas klaidas ir taip pagerinti programos saugumą.