SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE

Engineer thesis

Sensitive data extraction from verbal communication leaks

author: Maksym Brzęczek

supervisor: Błażej Adamczyk, DSc PhD

consultant: Michał Kawulok, PhD

Gliwice, November 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 15 listopada 2019

. . . .

(podpis)

. . . .

(poświadczenie wiarygodności podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Sensitive data extraction from verbal communication leaks" spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 15 listopada 2019

. . . .

(podpis promotora)

Contents

Chapter 1

Introduction

The invention and propagation of internet has boosted the ways in which technology impacts all of us. In this age increasing amount of our everyday life is digitized and dependent on cybernetic systems hosted and operated by independent corporations and institutions. Each of us caries in a pocket a computer constantly connected to internet with dozens of applications that constantly communicate with servers which can be located on the other side of the world. Aditionally many of simple items that we have learned to depend on like door locks or even lightbulbs become connected to the web (Internet of Things). This digitization has made our lives simpler and allowed us to achieve amazing things but it has also made us vulnerable to cybernetic attacks. It is only natural that the rize of the impact of technology was followed by the rize in the cyber crime and cyber security providers. Over the years an ecosystem has emerged that constantly compeets with malicious hackers to keep us all secured.

One of the elements of this structure is penetration testing also known as ethical hacking. In it's core, this practice is simply symulating a real attack. There are multiple sources that depict approaches used to petrform this process. One of the common denominators between all of them is the importance of gathering information. The reason for that is because the more information you can uncover and analyse, the bigger the chance of finding vulnerable systems or flaws in them. One of the clusters of information in in companies is a communication channel like slack or discord. There are manny situations where employees share information connected to projects and their workplace enviroment. That is esspecially true with companies that focus on software development. If an attacker was to acces such a platform he could potentially analyse the conversation history in search of sensitive information like ip adresses, logins, passwords etc. Unfortunately such a action may be very time consuming and that is where automating could prove very useful. a

1.1 Contribution

The focus of this thesis is design and implementation of "sensitive data" search tool. Acomplishing this task requires

Chapter 2

[Problem analysis]

- problem analysis
- state of the art, problem statement
- literature research (all sources in the thesis have to be referenced [1, 2, 4, 3])
- description of existing solutions (also scientific ones, if the problem is scientifically researched), algorithms, location of the thesis in the scientific domain

Chapter 3

Requirements and tools

- functional and nonfunctional requirements
- use cases (UML diagrams)
- description of tools
- methodology of design and implementation

Chapter 4

External specification

- hardware and software requirements
- installation procedure
- activation procedure
- types of users
- user manual
- system administration
- security issues
- example of usage
- working scenarios (with screenshots or output files)

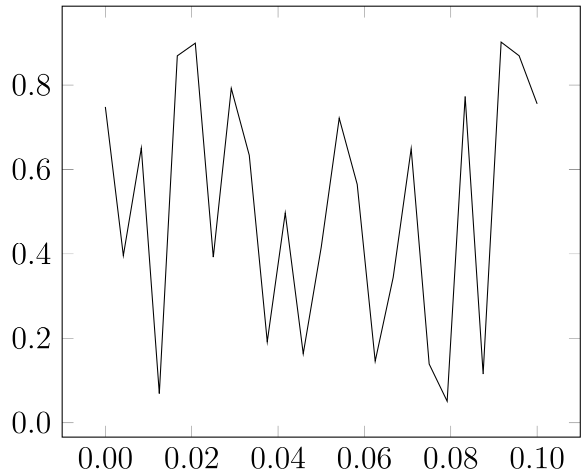

Figure 4.1: A caption of a figure is **below** it.

Chapter 5

Internal specification

- concept of the system
- system architecture
- description of data structures (and data bases)
- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Use special environment for inline code, eg **descriptor** or **descriptor_gaussian**. Longer parts of code put in the figure environment, eg. code in Fig. 5.1. Very long listings–move to an appendix.

```
class descriptor_gaussian : virtual public descriptor
{
   protected:
      /** core of the gaussian fuzzy set */
      double _mean;
      /** fuzzyfication of the gaussian fuzzy set */
      double _stddev;

   public:
      /** @param mean core of the set
          @param stddev standard deviation */
      descriptor_gaussian (double mean, double stddev);
      descriptor_gaussian (const descriptor_gaussian & w
         );
      virtual ~descriptor_gaussian();
      virtual descriptor * clone () const;

      /** The method elaborates membership to the
         gaussian fuzzy set. */
      virtual double getMembership (double x) const;

};
```

Figure 5.1: The **descriptor_gaussian** class.

Chapter 6

Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)

Chapter 7

Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension . . .)
- encountered difficulties and problems

Table 7.1: A caption of a table is **above** it.

	method						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Bibliography

[1] Name Surname and Name Surname. Title of an article in a journal. *Journal Title*, 157(8):1092–1113, 2016.

[2] Name Surname and Name Surname. *Title of a book*. Publisher, Hong Kong, 2017.

[3] Name Surname, Name Surname, and N. Surname. Title of a web page. `http://somewhere/in/internet.html`. [access date: 2018-09-30].

[4] Name Surname, Name Surname, and N. Surname. Title of a conference article. In *Conference title*, pages 5346–5349, 2006.

Appendices

List of abbreviations and symbols

DNA deoxyribonucleic acid

MVC model–view–controller

N cardinality of data set

μ membership function of a fuzzy set

$\mathbb{E}$ set of edges of a graph

$\mathcal{L}$ Laplace transformation

Listings

(Put long listings in the appendix.)

```
partition fcm_possibilistic::doPartition
                                (const dataset & ds)
{
   try
   {
      if (_nClusters < 1)
         throw std::string ("unknown␣number␣of␣clusters");
      if (_nIterations < 1 and _epsilon < 0)
         throw std::string ("You␣should␣set␣a␣maximal␣number␣of␣iteration␣or␣minimal␣difference␣--␣epsilon.");
      if (_nIterations > 0 and _epsilon > 0)
         throw std::string ("Both␣number␣of␣iterations␣and␣minimal␣epsilon␣set␣--␣you␣should␣set␣either␣number␣of␣iterations␣or␣minimal␣epsilon.");

      auto mX = ds.getMatrix();
      std::size_t nAttr = ds.getNumberOfAttributes();
      std::size_t nX    = ds.getNumberOfData();
      std::vector<std::vector<double>> mV;
      mU = std::vector<std::vector<double>> (_nClusters)
```

```
        ;
    for (auto & u : mU)
        u = std::vector<double> (nX);
    randomise(mU);
    normaliseByColumns(mU);
    calculateEtas(_nClusters, nX, ds);
    if (_nIterations > 0)
    {
        for (int iter = 0; iter < _nIterations; iter++)
        {
            mV = calculateClusterCentres(mU, mX);
            mU = modifyPartitionMatrix (mV, mX);
        }
    }
    else if (_epsilon > 0)
    {
        double frob;
        do
        {
            mV = calculateClusterCentres(mU, mX);
            auto mUnew = modifyPartitionMatrix (mV, mX);

            frob = Frobenius_norm_of_difference (mU,
                mUnew);
            mU = mUnew;
        } while (frob > _epsilon);
    }
    mV = calculateClusterCentres(mU, mX);
    std::vector<std::vector<double>> mS =
        calculateClusterFuzzification(mU, mV, mX);

    partition part;
    for (int c = 0; c < _nClusters; c++)
```

```
        {
            cluster cl;
            for (std::size_t a = 0; a < nAttr; a++)
            {
                descriptor_gaussian d (mV[c][a], mS[c][a]);
                cl.addDescriptor(d);
            }
            part.addCluster(cl);
        }
        return part;
    }
    catch (my_exception & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
           __LINE__, ex.what());
    }
    catch (std::exception & ex)
    {
        throw my_exceptionn (__FILE__, __FUNCTION__,
           __LINE__, ex.what());
    }
    catch (std::string & ex)
    {
        throw my_exception (__FILE__, __FUNCTION__,
           __LINE__, ex);
    }
    catch (...)
    {
        throw my_exception (__FILE__, __FUNCTION__,
           __LINE__, "unknown expection");
    }
}
```

Contents of attached CD

The thesis is accompanied by a CD containing:

- thesis (LaTeX source files and final `pdf` file),
- source code of the application,
- test data.

List of Figures

List of Tables