



SILESIAIAN UNIVERSITY OF TECHNOLOGY

**FACULTY OF AUTOMATIC CONTROL, ELECTRONICS
AND COMPUTER SCIENCE**

Engineer thesis

Sensitive data extraction tool

author: Maksym Brzęczek

supervisor: Błażej Adamczyk, DSc PhD

consultant: Michał Kawulok, PhD

Gliwice, December 2019

Oświadczenie

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

Gliwice, dnia 19 grudnia 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Sensitive data extraction tool” spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 19 grudnia 2019

.....
(podpis promotora)

Contents

1	Introduction	1
1.1	Description of the problem	1
1.2	Project scope	2
1.3	Description of chapters	2
2	Problem analysis	5
2.1	Sensitive data definition	5
2.2	Modularity and Customisation	6
2.3	Enviroment independence	7
2.4	Result presentation	7
3	Requirements and tools	9
4	External specification	11
5	Internal specification	13
6	Verification and validation	15
7	Conclusions	17

Chapter 1

Introduction

This chapter presents the problem that the project tries to solve and presents the document structure.

1.1 Description of the problem

The invention and propagation of the internet has boosted the ways in which technology impacts all of us. In this age an increasing amount of our everyday life is digitized and dependent on cybernetic systems hosted and operated by independent corporations and institutions. Significant amount of our tasks has become automated through information technology solutions. The physical world surrounding us also becomes intertwined with technology. We are gradually connecting things of everyday use like cars or house locks to the web through Internet of Things technologies. This process 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 rise of the impact of technology was followed by the rise in the cyber crime and cyber security providers. [3]

Over the years an ecosystem has emerged that constantly competes with malicious hackers to keep us all secure. One of the elements of this structure is penetration testing also known as ethical hacking. In its core, this practice is simply simulating a real attack. There are multiple sources that depict approaches used to perform this process. One of the common denominators between all of them is the

importance of gathering information. [1] The reason for that is because the more information you can uncover and analyze, the bigger the chance of finding vulnerable systems or flaws in them. One of the clusters of information in companies is a communication channel like slack or discord. There are many situations where employees share information connected to projects and their workplace environment. If an attacker was to access such a platform he could potentially analyse the conversation history in search of sensitive information like ip addresses, logins, passwords, emails, phone numbers, etc.

Such information is especially important from a legal point of view. Introduction of General Data Protection Regulation in 2016 has put a pressure on many legal bodies to responsibly handle people's personal data under a threat of heavy financial penalties. [4] Monitoring of the data located in the private entity's internal communication channel might prove very useful in fulfilling the legislative requirements of private information processing.

Unfortunately such a task may be very time and resource consuming. A tool capable of scanning the history of communication channel in search of data that would meet some established criteria could however fulfill this job or at least increase efficiency of the person responsible for it.

1.2 Project scope

This thesis focuses on the research into proper implementation of a cyber security tool with the purpose of processing a large amount of natural language messages in search of data that can be classified as sensitive. Special focus is put on possibility of calibration and customisation of the search engine and its reusability, regardless of circumstances and environment.

This thesis however does not elaborate the process of defining a proper sensitive data structure outside of the scope required for testing.

1.3 Description of chapters

This document is divided into seven chapters with following focus:

- Chapter 1 Introduction - Presentation of the problem domain and scope, introduction of the document structure.
- Chapter 2 Problem analysis - Research of the topic and design of the program.
- Chapter 3 Requirements and tools - Description of the used technologies and required prerequisites.
- Chapter 4 External specification - Instruction of the program usage.
- Chapter 5 Internal specification - Elaboration on the programs internal structure.
- Chapter 6 Verification and validation - Presentation of the testing methodology and results.
- Chapter 7 Conclusions - Final remarks and conclusions for the future of the project.

Chapter 2

Problem analysis

There are multiple important project assumptions that have to be thought through before the design and implementation of the project.

2.1 Sensitive data definition

The project will focus on finding nine categories of information:

- IP addresses
- Social security numbers
- ID card numbers
- MAC addresses
- Domains
- Email addresses
- Passwords
- Logins
- Phone numbers

Due to the random nature of those categories generating a dictionary for search purposes can prove difficult or in some cases even impossible. The complexity of this task would be further increased because of the different ways that some of those data types can be presented. In example an ip address can be stated either as "127.0.0.1" or "127 0 0 1" with the same meaning for the recipient of the message.

There is exists however a tool called "regural expresions" that is frequently used for parsing users input and capturing parts of strings. [2] This natural language processing tool is capable of taking under consideration fluctuations in the message formulation.

A dictionary search could however prove useful in searching for specific keywords that could indicate a presence of searched information. Such words like "login" or "password".

It might be advantageous for the purposes of this project to take under consideration additional properties of the considered data types. It is common for the personal information issued by some entities to implement a check sum algorithm used for validation of authenticity. Implementing a adjustable additional check of the data discovered by regular expressions could possibly reduce the amount of false positives. Such feature could be utilised by individual user to further enhance accuracy in any case where identifing additional patterns, that escape regular expression domain, is possible.

2.2 Modularity and Customisation

Each usecase of the designed system might differ depending on the user, environment and multitude of other conditions. Some clients of the program might be interested in searching for a subset of data types implemented in the program. It is also possible that an unanticipated in the design information type might be of a particular interest for a user.

In order to mitigate those problems a modular approach to the design could be taken that would allow for

2.3 Enviroment independence

2.4 Result presentation

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)

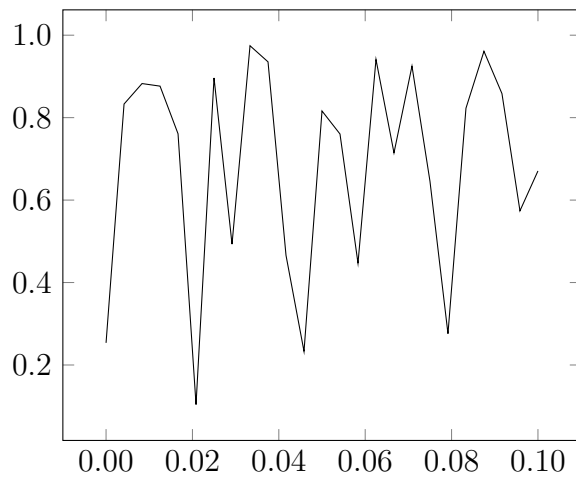


Figure 4.1: A caption of a figure is **below** it.

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.

```
1 class descriptor_gaussian : virtual public descriptor
2 {
3     protected:
4         /** core of the set */
5         double _mean;
6         /** fuzzyfication of the gaussian fuzzy set */
7         double _stddev;
8
9     public:
10        /** @param mean core of the set
11                @param stddev standard deviation */
12        descriptor_gaussian (double mean, double stddev);
13        descriptor_gaussian (const descriptor_gaussian & w
14            );
15        virtual ~descriptor_gaussian();
16        virtual descriptor * clone () const;
17
18        /** The method elaborates membership to the
19                gaussian fuzzy set. */
20        virtual double getMembership (double x) const;
21    };
```

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.

ζ	method						
	alg. 1	alg. 2	alg. 3			alg. 4, $\gamma = 2$	
			$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
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] Rafay Baloch. *Ethical Hacking and Penetration Testing Guide*. Auerbach Publications, 2014.
- [2] Kathryn T. Stolee Carl Chapman. Exploring regular expression usage and context in python. In *25th International Symposium on Software Testing and Analysis*, pages 282–293, 2016.
- [3] Rajesh Kumar Goutam. Importance of cyber security. *International Journal of Computer Applications*, 111(7):4, 2016.
- [4] Axel von dem Bussche Paul Voigt. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer International Publishing, Cham, 2017.

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.)

```
1 partition fcm_possibilistic::doPartition  
2 (const dataset & ds)  
3 {  
4     try  
5     {  
6         if (_nClusters < 1)  
7             throw std::string ("unknown_number_of_clusters"  
8                                 );  
9         if (_nIterations < 1 and _epsilon < 0)  
10            throw std::string ("You_should_set_a_maximal_  
11                               number_of_iteration_or_minimal_difference_--  
12                               _epsilon.");  
13         if (_nIterations > 0 and _epsilon > 0)  
14            throw std::string ("Both_number_of_iterations_  
15                               and_minimal_epsilon_set_--_you_should_set_  
16                               either_number_of_iterations_or_minimal_  
17                               epsilon.");  
  
18         auto mX = ds.getMatrix();  
19         std::size_t nAttr = ds.getNumberofAttributes();  
20         std::size_t nX    = ds.getNumberofData();  
21         std::vector<std::vector<double>> mV;  
22         mU = std::vector<std::vector<double>> (_nClusters)
```

```

    ;
18   for (auto & u : mU)
19       u = std::vector<double> (nX);
20   randomise(mU);
21   normaliseByColumns(mU);
22   calculateEtas(_nClusters, nX, ds);
23   if (_nIterations > 0)
24   {
25       for (int iter = 0; iter < _nIterations; iter++)
26       {
27           mV = calculateClusterCentres(mU, mX);
28           mU = modifyPartitionMatrix (mV, mX);
29       }
30   }
31   else if (_epsilon > 0)
32   {
33       double frob;
34       do
35       {
36           mV = calculateClusterCentres(mU, mX);
37           auto mUnew = modifyPartitionMatrix (mV, mX);
38
39           frob = Frobenius_norm_of_difference (mU,
40                                               mUnew);
41           mU = mUnew;
42       } while (frob > _epsilon);
43   }
44   mV = calculateClusterCentres(mU, mX);
45   std::vector<std::vector<double>> mS =
46       calculateClusterFuzzification(mU, mV, mX);
47
48   partition part;
49   for (int c = 0; c < _nClusters; c++)
```

```
48     {
49         cluster cl;
50         for (std::size_t a = 0; a < nAttr; a++)
51         {
52             descriptor_gaussian d (mV[c][a], mS[c][a]);
53             cl.addDescriptor(d);
54         }
55         part.addCluster(cl);
56     }
57     return part;
58 }
59 catch (my_exception & ex)
60 {
61     throw my_exception (__FILE__, __FUNCTION__,
62                        __LINE__, ex.what());
63 }
64 catch (std::exception & ex)
65 {
66     throw my_exceptionn (__FILE__, __FUNCTION__,
67                        __LINE__, ex.what());
68 }
69 catch (std::string & ex)
70 {
71     throw (__FILE__, __FUNCTION__, __LINE__, ex);
72 }
73 catch (...)
74 {
75     throw my_exception (__FILE__, __FUNCTION__,
76                        __LINE__, "unknown_exception");
77 }
```

Contents of attached CD

The thesis is accompanied by a CD containing:

- thesis (L^AT_EX source files and final pdf file),
- source code of the application,
- test data.

List of Figures

4.1	A caption of a figure is below it.	12
5.1	The descriptor_gaussian class.	14

List of Tables

7.1	A caption of a table is above it.	18
-----	--	----