Prevention of Spyware by Runtime Classification of End User License Agreements

Muhammad Usman Rashid
Balakrishna Garapati
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Author(s):
Muhammad Usman Rashid
E-mail: m.usmanrashid@hotmail.com

Garapati BalaKrishna
E-mail: balucse04@hotmail.com

University advisor(s):
Niklas Lavesson
School of Computing

School of Computing
Blekinge Institute of Technology
Soft Center
SE – 372 25 Ronneby
SWEDEN

Internet : www.bth.se/com
Phone : +46 457 38 50 00
Fax : + 46 457 102 45
Abstract

Spyware is a threat to Internet users because it may obtain valuable information from the users’ machines without their consent. The existing anti-spyware techniques are not found to be accurate enough in the prevention or detection of spyware. According to the law in many countries, vendors are bound to mention any inclusion of spyware in the End User License Agreement (EULA) of the associated software. Moreover, this agreement must be accepted by the user to have the software installed on the user machine. Thus, if the user accepts the agreement without reading it, he or she will unknowingly accept all the regulations mentioned in the EULA. Consequently, this study emphasizes that the EULA can be used to classify the software as spyware or legitimate by using data mining algorithms. We validate our approach by implementing an application and compare it with existing EULA analysis tools.

Keywords: Spyware, Classification, EULA
ACKNOWLEDGEMENTS

We are very thankful to God Almighty for the completion of our thesis. This thesis was completed with the support of some people whom we would like to express our gratefulness. Firstly, we are thankful to our parents who always kept us in their minds and prayers. Secondly and most importantly we would like to thank our supervisor, Dr. Niklas Lavesson, for his endurance and for guiding us in this work through interactive meetings, by the exchange of his knowledge and criticism on the contents of the thesis a number of times. Thus we improved its quality. Finally, we would like to say thanks to all of those whom we met and worked with in Blekinge Institute of Technology and specifically our close friends, who compassionately exchanged their skills and knowledge in regard to the completion of this thesis.
Table of Contents

ABSTRACT .................................................................................................................. 1

ACKNOWLEDGEMENTS .......................................................................................... 2

1 CHAPTER 1: INTRODUCTION ................................................................................. 5
  1.1 AIM AND SCOPE .......................................................................................... 6
  1.2 PROBLEM DEFINITION/GOAL .................................................................... 6
  1.3 THESIS OUTLINE ....................................................................................... 7

2 CHAPTER 2: BACKGROUND .................................................................................... 8
  2.1 OBJECTIVE .................................................................................................. 8
  2.2 RESEARCH QUESTIONS .............................................................................. 9
  2.3 EXPECTED OUTCOME ............................................................................... 9
  2.4 RESEARCH METHODOLOGY OUTLINE .................................................... 9
  2.5 RELATED WORK ........................................................................................ 10

3 CHAPTER 3: METHODOLOGY ............................................................................. 11
  3.1 EVALUATION METRICS ............................................................................. 13

4 CHAPTER 4: ANTI-SPYWARE APPROACHES ..................................................... 14
  4.1 STATIC IDENTIFICATION ............................................................................ 14
  4.1.1 Signature Based Identification ................................................................ 15
  4.1.2 Signature Heuristics ............................................................................. 16
  4.2 DYNAMIC IDENTIFICATION ..................................................................... 18
  4.2.1 Behavioral Heuristic ............................................................................. 18
  4.3 TAXONOMY OF ANTI-SPYWARE APPROACHES ....................................... 19

5 CHAPTER 5: EULA CLASSIFICATION ................................................................. 20
  5.1 THE EULA CLASSIFICATION TASK ......................................................... 20
  5.2 SUPERVISED CONCEPT LEARNING ......................................................... 21
  5.3 FEATURE VECTOR .................................................................................... 21

6 CHAPTER 6: SIMULATION OF EULA CLASSIFICATION ................................. 23
  6.1 GENERATION OF CLASSIFIER .................................................................... 23
  6.1.1 Word vector .......................................................................................... 24
  6.1.2 TF IDF Document ............................................................................... 25
  6.2 EULA CLASSIFICATION .............................................................................. 25
  6.2.1 Comparison using Multinomial Naive Bayes ........................................... 26

7 CHAPTER 7: EXPERIMENTAL RESULTS ............................................................ 27
  7.1 EULA ANALYZER ..................................................................................... 27
  7.2 EULALYZER ............................................................................................. 28
  7.3 EULA CLASSIFICATION BASED APPROACH ......................................... 31

8 CHAPTER 8: CONCLUSIONS .............................................................................. 33

REFERENCES: .......................................................................................................... 35
CHAPTER 1: INTRODUCTION

Malicious Software (Malware) is a well-known term that denotes all kinds of software particularly designed for malicious intent. There are different types of malware spreading on the internet (e.g., viruses, spyware, worms or Trojan horses) [1]. A specific type of malware, denoted spyware, tries to get valuable data from users without their knowledge or consent with the intent of giving the personal information to unauthorized companies or individuals [2]. Spyware exists because information has value [3]. The main purpose behind the design of a spyware is to gather valuable information from the victim’s machine and sell it to third parties. Once the machine has been infected by spyware there is a strong probability for losing private information. Therefore, the best approach to get rid of spyware is to prevent it from being installed into the system rather than detecting and removing it after installation.

Spyware is increasing day by day and it creates problems for novice users, as they may be unaware of its harms. These users may inadvertently install malware masquerading as legitimate software which contains spyware or they may just install spyware. The presence of spyware on a computer makes it vulnerable to new security threats as well. In general, spyware is referred to as software that gathers information about the computer usage and sends this information to a third party. This may also affects a computer’s performance and its stability [2]. Spyware is usually hidden in the software that is trusted. Many anti-virus software tools are available to protect the system from malicious software but to our knowledge no anti-spyware tool has been proven to be completely effective in the detection of spyware. This is because spyware has many different forms, which makes it difficult to distinguish using the anti-spyware tools [28]. Thus, it is good to find other possible ways of detecting spyware. There are many techniques that detect spyware on the basis of its signature or behavior but there should be some way to better prevent them from being installed rather than detecting them after the installation. At the moment, signature-based detection is a common technique to detect spyware before installation. There is, however, one more technique called behavioral- or heuristic-based detection but this technique is time-consuming and also costlier when compared to signature-based detection [22]. An existing hypothesis states that; whether a software hosts spyware or not can be found out before installation by analyzing its End User License Agreement (EULA) [3]. According to the law in California, the EULA must explicitly mention what the software actually does [28]. Some users generally skip reading the EULA because of the tiny size of the font and most often the difficult language used that makes it hard to read and understand [28]. In [3] the authors concluded that it is possible to classify software as malicious or non-malicious by using patterns found in the EULA. In their analysis of the EULAs, software was classified as good or bad, where bad denoted the elevated probability of it hosting spyware. The automatic classification was carried out using machine learning algorithms [3].
1.1 Aim and Scope

This paper attempts to investigate an approach to detect the possibility of the software in question installing spyware (without the users consent) by analyzing patterns found in its End User License Agreement (EULA). The aim of this paper is to make a simulation of an actual application that will classify any software as either good or bad by using its EULA and this approach will be evaluated by comparing with other existing EULA analysis tools, based on the results obtained from the simulation. If this technique indeed yields good results then it may prove to be a better approach to protect against spyware in comparison to the state-of-the-art.

After the implementation of this technique, it will process a data set of EULAs and we will obtain the results for each EULA for a further step of evaluation. Subsequently, our aim is to implement and then evaluate this technique. The evaluation can be done by analyzing and comparing existing EULA analysis tools, such as EULA Analyzer [4] and EULAlyzer [5] with our approach. EULA Analyzer is a simple web-based application, which has a text field in which the user needs to paste the EULA content. On the other hand, EULAlyzer is an installable application that can extract the EULA automatically from any currently installing software in order to analyze it. If we find that our approach is more feasible than the mentioned EULA analysis tools, then possible improvements can be done to implement it as a full-fledged anti-spyware tool.

1.2 Problem Definition/Goal

Software today can be very deceitful and install spyware without the user’s explicit consent. Since the user unknowingly accepts the EULA due to the EULA's length and incomprehensible language. Most users do not bother reading the EULA. In some countries the law requires the EULA should describe the functionality of the software, which most do but this fact is obscured in the length and the language used. There are some EULA analyzers that look for suspicious word patterns in the EULA of installing software. Yet there is a need for a tool that analyzes the EULA and is well trained by a machine learning approach to predict the type of EULA (legitimate or illegitimate). We intend to analyze the EULA and structure it in such a way that the end user can understand it and be warned of any malicious intent that the software may have. Our tool would implement a machine learning algorithm (Multinomial Naive Bayes) which will analyze the data extracted from EULA. On the basis of the results obtained in the earlier experiment [1] we made a prototype of the tool using C# programming language which will implement the machine learning approach. Hence this approach is expected to be a new and a better way of predicting the tendency of any software to install spyware.

Since, there is no such tool yet been made which uses machine learning to analyze the EULA, this would be a new approach for detecting the spyware. At the end we will compare the results generated from this approach with other existing EULA analysis tools to suggest which approach is better.
1.3 Thesis Outline

This first chapter has introduced the problem. The second chapter includes a deeper background and related work is also reviewed. The third chapter presents the methodology that we use to conduct our study. In the fourth chapter, we have gathered information about the existing anti-spyware approaches by the theoretical study. In the chapter we also introduce a new terminology and in the end we present taxonomy of the main anti-spyware approaches and show where our particular technique resides. The next chapter goes into detail about our technique and describes what EULA classification actually is and what the requirements for the classification of the EULA text are. The sixth chapter presents the simulation that we made to conduct the validation of our approach for classifying the EULA text to find out the tendency of software to install a spyware or not. The seventh chapter reviews the experimental results, obtained from the simulation. The last chapter includes conclusions and pointers to future work.
CHAPTER 2: BACKGROUND

Software today can be very unscrupulous and install spyware without the user’s explicit consent. The security of the computer systems used is heavily dependent upon many things like routers, anti-virus tools, anti-spyware tools, and firewalls. As the time is moving on many new techniques have been introduced, replacing the old ones to detect malware while executing in the systems. There is a need for gradual improvement in those techniques and to introduce new approaches because the malware authors implement the same malware software by using different ways to obfuscate the anti-spyware, anti-viruses, firewalls and so forth.

Many different types of threats exist for the computer systems nowadays, because malwares are attached with different software and they also get installed into the system blindly without the user consent. Now these malwares are of many different types and they attack the systems for different reasons. Our study will focus on one of the types of malwares which is spyware. One type of spyware that installs into the system is just to fetch the data from the victim’s machine and sends back to the main server. Information which is stored on any of the systems has a prime importance for that user of the system and it can be fetched secretly just by the help of the spyware. So such a malware could be a great threat to the victim’s machine.

Different techniques are available and been implemented to detect any software which is acting as a spyware. Most of the anti-spyware software’s which are in use either look for particular behavior of the software or comparing the signatures with the existing signature database and alarm for the spyware. These are actually very basic sub types of anti-spyware. All of these techniques which are in use by the anti-spyware will be explained in detail and will be categorized for better understanding in the later sections of the study. But to see the big picture at the moment: this study focuses on protecting the system before installing a spyware rather than detecting after spyware installed into the system. This will be carried out by using a machine learning approach.

2.1 Objective

The user unwittingly accepts the End User License Agreement (EULA) while installing any application due to the big length of EULA’s and incomprehensible language. Most users do not bother reading the EULA. The law requires the EULA to mention all that the software installs to avoid repercussions [3], which most do but this fact is concealed in the length and the language used. Because of the length of the EULA most users don’t read the EULA before accepting. That’s why some tools were made to analyze the EULA for users.

There are some EULA analyzers which look for word patterns in the EULA to find the probability of spyware in the installing software. Yet there is a need for a tool that analyses the EULA and is well trained by a machine learning approach to predict
the type of EULA, specifying that this software can install the spyware. So, we will implement a tool using machine learning approach.

2.2 Research Questions

The research questions which we will address our study are formulated as follows:

1. Which anti-spyware approaches exist and how can these be categorized in a suitable way?
2. Which of the existing approaches are based on EULA analysis and how can these approaches be distinguished from each other?
3. How feasible is the machine learning approach in comparison to other EULA analysis approaches?

To answer the first question we will define the different categories of anti-spyware approaches and present taxonomy. This taxonomy will be used to explain the categories in detail and show where our approach resides.

The second question will be addressed by more specifically looking for other approaches for analyzing EULAs and we will try to find out how to distinguish between these approaches.

We will compare our approach of classifying EULAs on the basis of a classifier generated by data mining algorithms. By experimenting with a set of EULAs and this technique we will gather results. On the basis of these results we will conclude whether this technique is more suitable for catching any application having the tendency to install spyware (just by analyzing its EULA).

2.3 Expected Outcome

The main outcome of this study is a validation of the approach of classifying EULAs to classify new software applications as either spyware or legitimate. On the basis of the classifier which we generated from a large data set of good and bad EULAs. Since, there is no such tool yet been made which uses machine learning to analyze the EULA, this would be a new approach for detecting the spyware. The results generated by this application will be compared with the other EULA analyzing approaches to validate this technique.

The reason behind implementing such a product is to prevent the installation of spyware before installation into the system rather to detect it after it is installed and working maliciously in the system.

2.4 Research Methodology Outline
As the aim is to validate the EULA classification approach against existing EULA analysis tools, we will implement the technique to extract information from a set of EULAs and then on the basis of that information we will use any machine learning algorithm to make decisions. After this we will evaluate this application on different EULAs and validate that approach on the basis of it and we will also compare the results with the other Eula Analysis tools to find out how feasible is that approach. So our approach to validate the study is quantitative. As this is a new approach and not in use we want to make a simulation very close to the real application, in which we will implement the data mining technique to gather data from the set of EULAs and we’ll compare new EULAs with the data we just obtained to generate the results.

To begin with, we will feed the mining algorithm with EULAs from different applications so that it can learn and then it would generate a classifier. The classifier will be used to classify the new EULA and to generate the results. We will evaluate this on different software EULAs and on the basis of it we will conclude whether this approach is good or not. We will test this approach and then we will compare the results with the other EULA analysis tools to conclude which one is seemingly better. More on research methodology is explained in the next chapter.

2.5 Related Work

The area of our study is to classify any new EULA on the basis of classifiers which are generated by doing data mining on the previously known good and bad EULAs. In the past an experiment was performed but with only a data set of 100 EULAs and classifying whether the associated software has the likelihood of installing spyware using 13 default-configured machine learning algorithms and it was concluded that two algorithms of the 13 were significantly better than a random guesser [3]. We intend to implement a tool that uses one of these two efficient machine learning algorithms (Multinomial Naive Bayes) with larger data sets.

In the area of spam classification a lot of work is done by using different machine learning algorithms like decision trees, stacking, support vector machines and so forth [15, 16, 17, 18]. In the spam filtering area we deal with the E-mails and as compared to the EULA of any software they have shorter text size. The spam filtering is somewhat similar in technology as there is also a need to apply the techniques to obtain the important features belonging to any particular class of spam and then comparing it with the newer mail.

There was a research work done for malware detection by applying machine learning algorithms on the executables obtained from the associated software [22]. But the detection of spyware by analyzing the text patterns of EULA using machine learning algorithms is a newly proposed research approach.
CHAPTER 3: METHODOLOGY

The goal of this study is to validate, the newly proposed approach of discriminating the good and bad software by classifying the EULA using machine learning algorithms [3] with existing approaches. Classification is done by labeling the given EULA as either a good or bad instance. For this, a data set of 996 EULAs including 900 good EULA instances and 96 bad EULA instances is studied. All the EULAs are collected from the known website [29]. An application is implemented with a Multinomial Naive Bayes algorithm to train the classifier on the data set in order to classify a new EULA.

Our study can be divided into a theoretical and an empirical part. The literature survey is an important part for any research work. In this part we will look for the work done in our area and the theoretical analysis was done to make the taxonomy of the anti-spyware approaches. In the simulation part we will look for the other related work done and will try to get help from that to get ease while implementation. Then quantitative analysis would be done on the results.

In the next step of validating this approach with existing approaches, two EULA analysis tools EULA Analyzer [4] and EULAlyzer [5] were considered. EULA Analyzer is a web based application where user has to copy the EULA text into the text field of the tool on the other hand EULAlyzer is an installable application. Initially a set of EULAs were taken from the data set which was mentioned before. Then each EULA was given as an input manually to both the tools and observed the EULA classification process of these tools. The results obtained for each EULA from both the tools were scrutinized and tried to understand how they are generating results for the EULAs and on what basis the process of discriminating each EULA is carried out. The results of both EULA Analysis tools are compared with the proposed classification approach and then we found the differences between new approach and existing approaches at classifying the EULA and conclude which approach is seemingly better along with the discussion of positives and negatives of each approach.
The figure 3.1 below demonstrates different phases we have employed in this research.

**Figure 3.1: Overview of Research Methodology**

To draw conclusions we have actually four different types of threats to the validity of our results according to Wohlin [30]. These threats are as follows:

**Conclusion Validity:**

While conducting an experiment we have a sample set and we do some treatment with it then after the experiment we get some output, such experimental study comes under the conclusion validity threat because the size of the sample set and the treatment done can affect the results.

**Construct Validity:**

This type of validation has the relationship between the theory and the observation. It is based on judgment on different studies.

**Internal Validity:**

In this kind of validity threat the relationship between both the treatment and the outcome is observed on the basis of the internal factors which can affect. Those factors which can disturb the experiment and can result into wrong results like any
special event occurs during the experiment or subject are treated or compensated during the experiment.

**External Validity:**

These are the threats to the experimental results which can affect externally. There could be three types of threats like the wrong subjects chosen for the experiment, the experiment is conducted in unsuitable environment and the time when the experiment was performed can affect the results.

In our study we will do some treatment with the sample set which will be the input for the experiment. The results will help us to conclude so there would be a statistical relationship between the sample sets, treatment done and then the outcome of experiment. The sample set is the set of EULAs we will use for the experiment. Treatment is actually the data mining algorithms we are using for the feature selection and then classification and the outcome is the result produced at the end of the experiment. Considering this our study will come under the Conclusion validity threat.

### 3.1 Evaluation Metrics

The Performance of implemented application is estimated quantitatively using a table presenting counts of true positive (TP), false positive (FP), true negative (TN), false negative (FN), true positive rate (TPR) and false positive rate (FPR). A true positive is a good EULA that correctly identified as a good one, false positive is a bad EULA that is classified as good one, true negative is a bad EULA that correctly identifies as bad one and false negative is a good EULA that is classified as bad. A table below showed all the metrics along with their definitions [29].

**Evaluation Metric:**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>TP</td>
<td>Good EULAs classified as good</td>
</tr>
<tr>
<td>True Negative</td>
<td>TN</td>
<td>Bad EULAs classified as bad</td>
</tr>
<tr>
<td>False Positive</td>
<td>FP</td>
<td>Bad EULAs classified as good</td>
</tr>
<tr>
<td>False Negative</td>
<td>FN</td>
<td>Good EULAs classified as good</td>
</tr>
<tr>
<td>True Positive Rate</td>
<td>TPR</td>
<td>TP/TP+FN</td>
</tr>
<tr>
<td>False Positive Rate</td>
<td>FPR</td>
<td>FP/FP+TN</td>
</tr>
</tbody>
</table>

Table 3.1: Evaluation Metrics [29]
CHAPTER 4: ANTI-SPYWARE APPROACHES

It is already discussed that spyware is a malicious form of software which resides on a user’s system to pass on valuable information to third parties without the permission of the user. There are many techniques already existing for the detection of the spyware. Most of the anti-spyware software use techniques to detect the spyware after the installation but there are not a lot of techniques that prevents the spyware from being installed on a system. There are various ways of categorizing the different approaches of detecting and preventing spyware.

Spyware mainly can be divided into two major categories; advertising spyware and surveillance spyware [6]. Advertising spyware are the ones that maintain logs of activities of the user on the system. These activities could be like browsing different websites like banking sites and others. The example of such spyware is advertisements appearing on the user’s machine. They can also log email addresses. On the other hand surveillance spyware are more detrimental to the system. They are used by the hackers to silently access the user’s system to launch an attack. Key loggers, Trojan horses and system monitors are examples of such spyware. These spyware runs at the backend to collect and convey useful and valuable information such as the bank accounts, credit card numbers and passwords to third parties.

Anti-spyware was thus introduced to combat the threat of spyware. When anti-spyware was first introduced it just looked for known threats in the systems where they were installed. This proved to be satisfactory until spyware writers started to write spyware that deceived antispyware. They used the same spyware with different properties and different look to keep them undetected by the anti spyware. This way the spyware makers are actually ahead of antispyware vendors, they always come up with a newer thing in spyware and then antispyware vendors have to counter that. And thus the battle rages on. With time newer antispyware approaches are being introduced. Major Antispyware approaches with all the relevant details are explained in the portion below.

From the root we can divide the antispyware approaches into either static identification or dynamic identification.

4.1 Static Identification

Static identification mainly works based on the signatures found from the programs. In static identification a large set of signature database is maintained. This database can be further used in different ways depending on the identification process that we follow for detecting spyware. Identification process can be divided into two categories called “simple signature based identification” and “signature heuristics”. We have discussed them both in detail in the following sections. In other words, static identification of spyware detection is done without actually running the malicious program [10].
4.1.1 Signature Based Identification

Signature based identification is one of the simplest ways and is used by all of the antispyware or other antimalware software to perform spyware detection. This approach works by simply looking for any matched signature in the database. In this approach a database is maintained with all known signature variants of the spyware. Though this technique is faster and is a basic step in detecting spyware, it has its own drawbacks. So, this technique is always being used in conjunction with other techniques. After a scan by this technique other approaches are applied to capture those spyware which were left undetected by the signature based approach. This technique works as a first line of defense by the antispyware.

Signature based identification can be further divided into the two sub categories: *File Properties Based* and *File Content based*.

4.1.1.1 File Properties Based

This approach is perhaps the simplest and the oldest of all the anti-spyware approaches. As the name suggests this approach looks for the file properties and on the basis of these properties it detects whether a particular file is a legitimate one or a corrupted one. A database is maintained with the file properties of known spyware. The antispyware vendors have to keep this database up-to-date with the newly found properties of all the spyware.

The filename is one of the basic properties used to detect known spyware. If any filename in the application folder matches with the name database then the application will be flagged as malicious. A problem arises if any of the legitimate files have the same name as the spyware files or any corrupted files then it will be deleted or the antispyware will raise an alarm for the spyware found. There are databases for the other properties of the file like the author name, publisher, version and the size of the file. On the bases of these properties we can detect the spyware just like the file name property is used. So both file name and the other properties are mixed together to make a signature based authentication very robust but this technique can be fooled by changing the names and other attributes of the spyware. So that’s why this technique is not reliable always.

4.1.1.2 File Content Based

This is another technique which actually looks for the contents of the file instead of the file properties. By using this technique the antispyware looks for particular malicious strings (signatures) inside the file and if any matched signatures are found then the file is flagged for removal. In general it is quite difficult to change the program structure and thus this technique is more reliable and more often used than the technique mentioned in the previous section. Simple obfuscation techniques can still defeat this approach. [21], this technique is mainly designed for detecting existing spyware.
4.1.1.2.1 EULA String Analysis

Mostly software has the EULA or the privacy statement at the start because it is actually a contract with the user that if you will install it the application will perform these activities no other than what it is stated in it. Because the company can be sued in case, if any of the activity which the software was not expected to perform and it is not mentioned in the EULA text then it’s against the law. So that is why the EULA of the application has a lot of importance and it should be read by the users while installing any software so that the software will not perform any illegal activity.

In the EULA string analysis there is a database of flagged strings that are maintained in the anti-spyware tool using such a technique. Whenever a new EULA text is given to that software then it looks for the flagged sentences and on the basis of these sentences it generates statistics but doesn’t give any decision. It is up to the user what he or she will infer from it.

4.1.1.2.2 Executable Byte Sequences

This technique is similar to the above one as it also comes under the signature based identification which is a static approach of identification in this one the malicious executable byte strings are maintained in the database by the anti-spyware. Then after the disassembling of any file the signatures are compared and if any flagged one found the anti-spyware alarms for the malicious software found.

4.1.2 Signature Heuristics

Static Signature heuristic is advanced approach than the simple signature. As we discussed earlier simple signature identification has many shortcomings because it looks for specific strains in the program logic to detect the spyware. If the unknown spyware sample entered into the wild, simple signature approach has difficult to find it. It needs continuous updating of signature database. This approach can also be easily subverted by changing the program logic. Signature heuristic approach is similar to the signature based identification except that it looks for the behavior of the executable program. In the static signature heuristics a database is maintained with the behavioral signatures (signatures along with their functional behavior). The programming logic is analyzed using some handful of methods to find the new or unseen malicious examples. We have seen that this approach can be applied in two different ways. We have discussed about these in the following sections.

4.1.2.1 String Classification

The name suggests that detecting spyware by applying some classification methods to strings found from the application. These string patterns can be either text features or features extracted from the executables. And the classification methods could be any machine learning algorithms. So this technique is further subdivided into two techniques which will be explained in detail to know what this string classification is.
4.1.2.1 EULA Classification

EULA or the privacy statement must be accepted by the user before he/she wants to install most of the software. As we discussed earlier it could be possible to discriminate the software based on the text patterns in EULA document. This analysis is carried out by applying some machine learning algorithms. Hence, this approach comes as one of the subcategories of string classification. In this approach the classifier is trained over good and bad EULAs and generates the rule set to find whether the new software is hosting spyware or not.

4.1.2.1.2 Classification of Executable Strings

The classification process of this approach is similar to the above approach except that the strings of text patterns from the text document are replaced with strings of byte code sequences extracted from the executable programs. Initially a set of benign and malicious programs are selected to collect the benign and malicious byte code patterns in a separate dataset. And these datasets will be trained by the machine learning algorithms to detect the new or unseen malicious programs [22]. Schultz [22] applied three different machine learning algorithms each having its own feature extraction techniques. They are;

- **Portable Executable Header (PE) mining**: In this method the Ripper rule learning algorithm is applied on the features found in the plain text headers of PE-executables.
- **Signature-based Mining**: in this method the Naive Bayes algorithm is applied on the features extracted as binary strings.
- **Mining of Byte Sequences**: In this method the Multinomial Naive Bayes algorithm is applied on byte sequences as features.

4.1.2.2 Heuristic-based Byte code analysis

This approach can recognize the possible behaviors of the program by using some handful of methods. As we said before a large database is maintained with the all known signatures along with their functional behaviors. Whenever we need to analyze the program logic, it is undergone for the examination. And if any suspicious behavior found in the program the associated signature will be compared to the database of behavioral signatures. This process will continue for the entire program and then recognize the possible behavior of program. And on the basis of this behavior the associated application will be flagged. The main difference of this approach from the simple signature identification is finding what actually the program does instead of flagging the application based on the specific signature. This way we can overcome the obfuscation techniques and reduce the false positive rate. This approach works fine only when the low number of executables needs to be analyzed. If we need to analyze the large number of executables manually, this technique becomes costly and also time consuming process. Below we briefly explained one of the techniques proposed similar to this approach called static analysis of malicious executables [27].
Static Analysis of Malicious Executables:

In this technique the authors analyzed the API calling behaviors from the malicious executables [27]. Once the PE-executable is undergone for the examining, the API calls will be extracted and compared with the API-sequence database. Then the API analysis report will be generated and on the basis of this report the intension of the program can be expected.

4.2 Dynamic Identification

In the dynamic identification the programs are being executed in a virtual environment and the whole working process is monitored for any malicious behavior if exists. This virtual environment is close to the real environment but the critical resources are under control and all the activities performed by the software are monitored. This concept is in use, just to make the programs risk-free at the runtime [7]. It doesn’t mean that program won’t have threats but the program can’t attack the resources because it is running in a protected mode or in a very secure environment and all the resources are monitored. This way the existed vulnerabilities in any system can’t be exploited [7].

Dynamic intrusion prevention is actually in use to build a system, which should let any program work if it is working normally and crash the program working if it tries to do some illegal operation. Also it can crash the program if it is performing any abnormal behavior which is not expected. That operation can be a threat to the targets which are protected. This type of prevention ends up in a form of a complete intrusion prevention system. The further in depth discussion on dynamic identification is as follows with the drawbacks in this approach.

4.2.1 Behavioral Heuristic

The behavioral heuristic identification is done by the anti-spyware in mainly two phases. In the first phase the scanning software identifies what are the expected behaviors that any program is capable of exhibiting. In this approach the executable binary code is scanned for the location where the malicious part can be attached or can be found. This is a very important step in the behavioral detection technique because if the executable is just hundred lines of code then it works fine but if the executables are of thousands of lines of code then the behavioral heuristic will take a very long time in detecting and will make the technique very slow. Actually that technique emerged when there was a DOS operating system and at that time the malicious program size was usually very small but now there is many large-sized software applications and if any of the malicious programs will infect them then this technique will be ineffective with regard to time consumption.

4.2.1.1 Sandboxing

Sandboxing is a technique under the concept of dynamic identification and it’s a practical example of behavioral heuristic. It is a technique to create an environment
which is restricted as what the concept of behavioral heuristic approach [9]. This concept is in use in different situations. One very good example is as follows:

The Java virtual machine provides a sandboxed environment for the Java program. This also raises the confidence level for the user of the system for such a platform. The Java virtual machine is a very good example of a sandboxing technique but it is not a general thing since it is only used for Java programs.

Sandboxing is a strong technique and if it could be implemented at the operating system kernel level then that could provide a lot of help against malware. Because if any layer which can reside in between the programs and there system call then all the software will work under a confined environment.

4.3 Taxonomy of Anti-Spyware Approaches

There are many antispyware approaches existing and we tried to make a hierarchy with these antispyware approaches. The different techniques are already described above and the hierarchy which we made is shown below in Figure 4.1:

![Figure 4.1: Taxonomy of Anti-Spyware Approaches.](image)
CHAPTER 5: EULA CLASSIFICATION

Spyware is spreading and it exists because many are interested in obtaining important information. Spyware is often a hidden component of software packages that are freely available [13]. Spyware is a large threat nowadays since approximately 80% of the personal computers are infected [3]. There are many techniques existing as we have seen for the detection and the prevention of spyware. EULA analysis and classification are also two of those techniques to prevent the installation of any software application which can install spyware. There are two tools for the EULA analysis which will be discussed in the later portion of this study but the EULA classification hasn’t been done yet. The idea is proposed yet and that idea is being implemented by us to validate that approach on the basis of the results generated by our tool and the other analysis tools.

So, we want to investigate that whether or not it is possible to classify the software application as legitimate which is a good category or it have tendency to install a spyware then it’s a bad based on their EULA. As we have mentioned above there are two approaches which for the Text classification task; knowledge engineering and machine learning.

The knowledge engineering systems have been known to do better than the machine learning systems on the text classification task, although the gap in the performance steadily shrinks [14]. The major drawback of this approach is the amount of the skilled labor and the expert knowledge required to generate and maintain the knowledge encoding rules.

In comparison, the machine learning approach only requires a set of classified instances which are not as much difficult to produce. As a result, most of the recent work on the categorization is concentrated on the machine learning approach and our study is no exception. So now we will present how to classify text, especially EULAs, more properly.

5.1 The EULA Classification Task

In the EULA Classification task we have, let us suppose a set of EULAs, which we collected to generate the classifier. The task while learning is to generate a classifier that produces results. There exists two classes for the classification of EULAs either good or bad, good class has some set of good EULAs and bad class has some set of bad EULAs. So, finally the classifier that is trained by machine learning algorithm will decide to which class the new EULA belongs.
5.2 Supervised Concept Learning

Supervised learning is a machine learning approach to automatically build a classifier by learning the properties of the classes from a set of pre-classified training instances. Most of the supervised concept learning algorithms cannot process the text documents in their original form [14], hence they will not be able to process the EULA but the application which we built just requires a folder path which contains the EULA of the specific class. So there would be no need in our case to pre-process the EULA into a manageable representation.

5.3 Feature Vector

The feature vector contains the features of the text and is the representation of different classes of text. So, any new instance could be compared on the basis of these features representing a class to find the similarity. The features can be represented in the text document in the form of feature vector [25]. One of the examples of such a document representation is the bag-of-words model. It is found in many experiments that the more complex document representations do not give any noteworthy results [26]. In this model, each word occurring in any particular document is selected as a feature of that document, all of these features together makes a vector with all the features which are actually distinct words in that document. So, a complete vector contains all the distinct words appeared in all the training instances. In some studies the phrases are also been used in place of the words but the generated results on the basis of the vector made by the phrases were not so convincing [26]. Still many studies are conducted to improve this approach to get more accurate results.

There are different ways to associate weights with features. The first one is to use a binary representation, in which every feature if found once or more in a document will be represented by 1 (true) or otherwise 0 (false). This kind of representation cannot be used to look for how many times a particular word was found in a document, but the most appropriate one is to use the frequency of the word [31]. In our software we also used the frequency of the words. But there are several formulas available that have been used in different studies for the calculation of the frequency of the word or term.

The raw term frequency representation may not do well for most cases [20]. In raw term frequency every word or feature has a equal importance. Some words appear frequently and some words appear occasionally, so they should perhaps be treated differently to increase the precision of classification. We have transformed the raw term frequency to the most commonly used scheme, which is the TF IDF (Term Frequency – Inverse Document Frequency) word weighting scheme [20]. TF is the term frequency of a word in a specific document under consideration. IDF is the inverse document frequency which is a variant of the Document Frequency (DF). Thus, the word weight will be calculated as [20]:

\[ \text{Word Weight} = \text{TF} \times \text{IDF} \]
\[ tf-idf_d = tf_t \times idf_d. \]

According to the above formula the (TF) is the term frequency which tells about how many times a word appeared in any document. The second one is the (IDF) which is the inverse document frequency it shows in how many different documents this term occurred. This one is important because the word occurred the most number of times is a good indicator of any particular class of documents but is that word occurred in almost all of the documents than it’s not a good indicator of that class and using that one with higher frequency will falsify the results.

Once we have transformed all the term frequencies to corresponding TF-IDF weights, this TF-IDF document will be given to the classifier along with the new EULA to generate the result.
CHAPTER 6: SIMULATION OF EULA CLASSIFICATION

The techniques of the text classification discussed in last chapter were used by us for the EULA classification, as these techniques are already been used for the spam filtering but there was an issue in that the E-mails do not contain a lot of text and it is an easy and fast process for the classification of the mail. In comparison, the EULAs contain more text, e.g., usually more than 4,000 words. Thus, that could imply a slower process, considering all of these issues we built an application which will generate the decision by EULA classification and in the end after the completion it could be decided whether this technique should be considered feasible. In fact, we implemented two different applications; the first one for the generation of the classifiers and the second application was used to analyze the new EULA and to compare it with the classifier generated by the first application. The whole process of gathering the required information by the help of the simulated tools which were made is explained in detail in the later sections of this chapter.

6.1 Generation of Classifier

This EULA classification technique was proposed and was tested using the WEKA software for generating the word vector and then by running different algorithms on it for testing [3]. Our approach was to make a separate application which will simulate the real software application for runtime classification of the EULA.

The first task for the generation of the classifier was the gathering of already classified EULAs because these EULAs will help for the machine learning and to train the classifier and on the basis of that we can compare a new EULA. There were 900 good and 96 bad EULAs which we used for the machine learning purpose. Figure 6.1 shows an overview of the text classification process.

![Text Classification Process Diagram](image-url)

Figure 6.1: Text Classification Process [33]
6.1.1 Word vector

A word vector is a simple representation of the EULA text in the form of words along with their frequency (Number of times the word has occurred) after dividing the text into number of individual tokens or words. This way each EULA in the data set is converted to a simple representation of word/frequency, and then placed in a single database. Finally we will end up with a single database that includes all the training EULAs in the simplified form. We refer to this database as a word vector database and it is later used to generate the classifier.

6.1.1.1 Stemming Words

The text in the EULA is first read into a string before removing all the spaces and the digits just considering the alphabets. All the words in that EULA were separated into an array of words. Then, on each word, we applied a stemming algorithm to remove alternative words based on the same word stem. The stemming process was done by using the Porter stemming algorithm, which is widely used in text classification applications [32]. The procedure with which it stems the word can be understood by giving an example. Let suppose we have three words (class, classifier and classification) after applying the porter stemming to all of these words the resultant would be (class, class, and class). At that moment, all the words which occurred again in the array are not removed because we have to count the number of times the word occurred in any particular document.

6.1.1.2 Stopping Words

At this step, we have the list of words which are already stemmed. Now we can remove words like: is, am, are, the, and so forth. The stopping word list for the English language as it is used in the spam filtering approaches contains about 550 words which are normal words that we use a lot in our daily life (in almost every sentence). These stopping words are sometimes referred to as noise words. Thus, by considering them in the learning process, we may actually degrade the accuracy of the generated classifier. To find which words should be removed and which to consider for getting an improved accuracy is a complex research problem on its own.

6.1.1.3 Word Count

Now we have an array of words which is in a refined form. We may now calculate the number of times each word appears in the document under consideration and write this count along with the word into the word vector document.

This way we can make the word vector documents for the good and bad instances while training. After generating it a main step is finished, in the process of generation of feature vector document later on.
6.1.2 TF IDF Document

This document is actually used with the classifier, having the words along with the word weights instead of the raw term frequencies. The word weight is calculated using the combination of term frequency and inverse document frequency. TF-IDF document is generated from the word vector document which contains all the features of each training EULA along with the raw term frequency of the associated feature. For calculating the TF IDF weights for a complete word vector, each word is first read with its count and then it will be searched in the whole word vector document where it appeared again and out of how many document this word found then taking the log of it and multiplying it with the count generated the TF IDF weight for that word for particular document. This TF IDF weight actually describes the frequency of that word in a particular document and also the frequency of that word in the corresponding class. Thus, it is a suitable representation of any word or feature vector in that document.

This step may increase the time consumed while generating the feature vector document. Each selected feature needs to be compared with all the other features of instances in the whole corpus to calculate the TF IDF frequency for that feature. So, the time required for that calculation depends upon the number of the instances we have in whole corpus.

6.1.2.1 Threshold Settings

In the generation of the TF IDF document we look for words that appear in low numbers. Additionally, we look for words that appear more frequently compared to the corpus data set. These words may actually affect the results. Thus, we can set the threshold value while generating this final document which will act as the classifier by removing some of the words on the basis of occurrence and then analyzing the results. The point where we will get the best results will be the threshold for us.

After completing this step the documents that shall be generated are good TF IDF and the bad TF IDF. These documents will be further used with the classifier to classify the new EULA.

6.2 EULA Classification

We have seen in the last step that the feature vector document is generated out of the instances of both good and bad classes and then given to the next application which is actually looking for any EULA on a port because it is a simulation of how the actual application implementing that technique will generate results. In this application there would also be some steps almost identical to those we have just explained but not all of those will be performed here. The steps which will be performed on the new EULA are as follows:

- First of all the stemming technique will be applied on the text of the EULA after separating the different words in the text. Same as we did while in the generation of the feature vector. After applying this technique we will only have the base words found in the whole new EULA.
• Second step would be to apply the stopping words technique. It will remove the noise words from the array of words because they are those words which appear in every kind of text like is, am, the, are, was and so forth.

Later we have given the words to classifier to classify the EULA either good or bad.

6.2.1 Comparison using Multinomial Naive Bayes

We have chosen to work with the Multinomial Naive Bayes supervised concept learning algorithm (McCallum and Nigam (1998)) since it has given many good results for the classification of text document representations in earlier research. We used this algorithm to train the classifier with the TF IDF documents to classify the new EULA. According to the Multinomial Naive Bayes algorithm the probabilities or the frequency of each word from one class, which we got after applying the stemming and stopping word algorithms on the new EULA text are multiplied with each other. Those words which were not there in the classifier of any class were given the probability of negligible size so they would not have a strong impact on the resultant numeral. The resultant numeral which we achieved after the calculation is then multiplied by the prior probability of that class.

The prior probability of the class is actually the frequency of instances of any particular class; it is the probability of any particular class of document in a whole corpus. After multiplying the numeral we just obtained above with the prior probability of that class we will get a value indicating how relevant the new document is in relation to that class. This way, we calculate the relevancy score for both good and bad classes. The relevancy score which is higher among two classes shows that the new document belongs to this particular class.

This way we calculated the results and they will be discussed in the next chapter but what is of main importance is that the document we used for training or to generate the Feature Vector document (TF IDF document) should be a good representation of that class; if the wrong instances were included in one class, e.g., good documents were used for training the bad class, it will obviously degrade the results in the end.
CHAPTER 7: EXPERIMENTAL RESULTS

This section presents the experimental results observed from both, proposed technique and competing techniques. Initially, we discuss the competing techniques EULA Analyzer [3] and EULAlSyzer [4] respectively.

7.1 EULA Analyzer

As it is stated before, the EULA Analyzer is a simple web-based application in which the user has to copy his current EULA text content into the text field provided on the web page to analyze it. The experiment was conducted with a set of EULAs to observe the results and then to understand the EULA analyzing process.

In the result generating process whenever a new EULA was given to the EULA analyzer, some simple statistical calculation results like the total number of words, total number of characters, total number of sentences, average words per sentence and some advanced calculations such as: Flesch score, Flesch grade, Coleman-Liau Index and so forth, were observed along with a table presenting the number of flagged sentences corresponding to the characteristic found (e.g.: Advertising, tracking, warranty) in a given EULA text. The complete statistics for a given EULA is presented in the table (7.1.1) and (7.1.2) below.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of characters</td>
<td>15201</td>
</tr>
<tr>
<td>Number of words</td>
<td>2414</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>67</td>
</tr>
<tr>
<td>Average words per sentence</td>
<td>36.03</td>
</tr>
<tr>
<td>Flesch Score</td>
<td>9.02</td>
</tr>
<tr>
<td>Flesch Grade</td>
<td>21</td>
</tr>
<tr>
<td>Automated Readability Index</td>
<td>26</td>
</tr>
<tr>
<td>Coleman-Liau Index</td>
<td>23</td>
</tr>
<tr>
<td>Gunning-Fog Index</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 7.1.1: EULA Analyzer (Scoring Metrics)

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Sentences Flagged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference to tracking or monitoring.</td>
<td>1</td>
</tr>
<tr>
<td>Removal: Reference to removal restrictions or procedures.</td>
<td>1</td>
</tr>
<tr>
<td>Reference to removal restrictions or removal rules by third party tools</td>
<td>1</td>
</tr>
<tr>
<td>Discharges any liability against software maker.</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7.1.2: EULA Analyzer (Flagged Sentences)
Flesch Score and Gunning-Fog index rates the readability of given text depending on the length of words and sentences, low scores represent difficult texts (difficult in terms of comprehension). Flesch Grade works like the Flesch score, but determines the number of years of education needed to understand the given text. Automated Readability Index and Coleman-Liau Index are two other readability text metrics like Flesch Grade and Flesch Score, but rates the text based on a factor of characters per word, instead of syllables per word [29].

However, EULA Analyzer does not classify the given EULA as either good or bad. It is up to the user to make the decision. Therefore, the user has to go through all flagged sentences obtained from the tool and decide whether the associated software is malicious or not.

The next step of the analysis is concerned with the basis on which sentences are flagged and assigned to a specific category. For this step, several numbers of flagged sentences were scrutinized and it was observed that flagging is done mainly based on specific keywords found in a sentence. For example, if any keyword like advertising or collectively was found in the sentence then it would be flagged. There are some other keywords like pop-up, No Warranty and so forth. Consequently, from the results it can be said that there is no classification or search for new patterns carried out by the EULA Analyzer. Another issue would be that, if any software author unwittingly or for other reasons uses any keywords mentioned above, the sentence that includes a keyword will simply be flagged and then the user has to go through the flagged sentence even though it is not necessary. However, this tool is proprietary and so one cannot say exactly how the program works or how the database maintained for it is constructed.

### 7.2 EULAlyzer

EULAlyzer is another EULA analysis tool. Unlike EULA Analyzer it is an installable application and also has the facility either to capture the EULA automatically of any currently installing software or to copy the text manually into the provided text field. This tool was also tested with a set of EULAs given manually to observe the results and then to understand its analysis process.

In the analysis process, EULAlyzer is assigns some value called EULA interest ID to each EULA. It then represents its results in the form of two columns each containing the field names Flagged Text and Interest Level respectively. Under the flagged text field the text found in the EULA will be displayed and the interest level field specifies the importance to be accorded to the corresponding text in the Flagged Text field. Subsequently EULAlyzer also presents a brief overview of the given EULA.

It is observed that the value of interest ID first depends on the length of the given EULA and then on the number of flagged sentences found. If the length of a given EULA is fairly short, the value assigned is also low (ex: 11 ID) and then depending
on the flagged sentences found this value is incremented. If it is fairly lengthy, the value assigned is high (ex: 300 ID).

The next step is analyzing the *flagged text* and *interest level* fields. It is observed that each text in the flagged text field is assigned to a specific category (ex: website address, third party, advertising and without notice). And this assignment is done based on the keywords found in the text sentence. For example if a keyword *third party* is found in the text, then a certain range of text including that keyword would be selected and displayed under the category *third party*. It is same for other categories too.

When it comes to *interest level*, it indicates level of importance of reviewing related flagged text. This level is given on a scale that ranges from 1 to 10, higher the number, the greater the need to review the flagged sentence. It can be said that scaling is mostly done based on a range of text selected after finding a specific keyword, this means if any specific keyword is found in the EULA, then that range of text having that keyword will be selected. After the text is selected, it then looks for keywords that can be given some significance or meaning in connection with the main keyword that caused the selection of the text in the first place. For example if the keyword *third party* is found in a EULA, then the scale of selected text would be 5, on the other hand, if the keyword was *third party software* then the scale would be 6 but both come under the category *third party* with different scales. It is also observed that if the word *software* is found independently of any keywords, then it will not be shown as a flagged keyword. This means scaling depends on some predefined combination of words with the base keywords. Therefore, it can be seen that this tool follows 3 stages:

1) Find a specific keyword.
2) Select the text containing that keyword.
3) Analyze that text to scale it between 1 and 10.

As this tool is proprietary it cannot be said exactly how many keywords are used, on what basis a certain range of text is selected and what are the best keyword matches used to scale the text.

The last task that needs to be discussed is about the overview text found after the text field. Overview text is named as *Details*. This section gives an opinion about the given EULA in a short and simple word form. The following example shows an overview text of a sample EULA in which there were no flagged sentences found.

**Example:**

*Details: The license agreement above has a low calculated interest ID. It isn’t too long, and there were no detected ‘interesting’ words and phrases.*

If the value of interest ID is very high and there are in fact a few number of flagged texts found in the EULA, then it would have mentioned that the given EULA has elevated calculated interest ID and some detected interesting words and phrases. If there are many flagged texts, it would mention that many interesting words and phrases were detected.
However, it is important to notice that the basic detection of patterns in a EULA is being done based on a specific and predefined database of keywords and the EULA is also not classified. It is up to the user to make a decision regarding the EULA like in EULA Analyzer. Irrespective of whether the EULA is good or bad, this tool simply gives the results based on the number of detected interesting words in a EULA and the length of a EULA. The user is required to analyze the results and make an installation decision.

An example results from the EULAlyzer tool:

Here are the results which were generated by the EULAlyzer for one of the EULAs which we already know is a bad one. According to the EULAlyzer it has low interest ID 202-1D. It found two flagged sentences:

1. supersedes any other communications or advertising with respect to the Software
2. shall not be disclosed by you to any third party who has not agreed to the term

The Details section shows: The license agreement above has a very low interest ID. It’s a healthy read and there were a few detected ‘interesting’ words and phrases.

The table 7.2 below shows some results by EULAlyzer after analyzing few EULAs:

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Interest ID</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>202-1D</td>
<td>The license agreement above has a low calculated interest ID. It’s a healthy read and there were a few detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>2</td>
<td>1229-1D</td>
<td>The license agreement above has an elevated calculated interest ID. It’s extremely long and there were high number of detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>3</td>
<td>217-1D</td>
<td>The license agreement above has a low calculated interest ID. It’s not too long but there were many detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>4</td>
<td>1160-1D</td>
<td>The license agreement above has an elevated calculated interest ID. It’s extremely long, and there were a high number of detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>5</td>
<td>249-1D</td>
<td>The license agreement above has a low calculated interest ID. It’s a healthy read and there were some detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>6</td>
<td>202-1D</td>
<td>The license agreement above has a low calculated interest ID. It’s a healthy read and there were a few detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>7</td>
<td>360-1D</td>
<td>The license agreement above has a low calculated interest ID. It’s a healthy read and there were a high number of detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>8</td>
<td>481-1D</td>
<td>The license agreement above has an elevated calculated interest ID. It’s a rather long and there were some detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>9</td>
<td>426-1D</td>
<td>The license agreement above has an elevated calculated interest ID. It’s a rather long and there were many detected ‘interesting’ words and phrases.</td>
</tr>
<tr>
<td>10</td>
<td>735-1D</td>
<td>The license agreement above has a high calculated interest ID. It’s extremely long, and there were a high number of detected ‘interesting’ words and phrases.</td>
</tr>
</tbody>
</table>

Table 7.2.2: Results from EULAlyzer
From the results observed from the above table, it can be safely assumed that the user would not be able to comprehend the results completely until he/she gone through all the flagged texts.

### 7.3 EULA Classification Based Approach

The process of classifying the EULA is already explained in chapter 5 and how the simulation of EULA classification application will work is explained in chapter 6. In that process we have given a larger dataset of EULAs for the purpose of training and generated the good word and the bad word vector documents. We then generated the feature vector (TF-IDF documents) from the word vectors and then used these documents with the classifier to classify the different EULAs and noted the results. The initial results were far from satisfactory. This could have been because of the presence of irrelevant words (words which may have been appearing very few numbers of times in the whole data set but affect classification results negatively). The time when the classifiers were generated, we just used each and every word after applying the stemming and stopping technique but we did not apply a threshold value to exclude irrelevant words.

We then applied different threshold values in generating the classifier. This time we kept on generating different feature vectors with different threshold values. At one point we got a better result when we removed words that had appeared 5 times or less in the whole corpus. That means at this threshold value maximum number of noise or irrelevant words could be removed from the document and thus we saw an improvement in results. We then analyzed our results with the newly generated feature vector and then compared it to the other tools, viz. EULA Analyzer and EULAlyzer.

We have used the same data set for the purpose of training and testing and generated the classification results for all 996 EULAs. We used the cross validation technique for the training and testing purpose [33]. We applied two fold cross validation on the whole data set. Means whole dataset was divided into two sub sets each having the 498 instances (good = 450 and bad = 48). While one set was used for training, the other set was used for testing. This way every instance of all 996 instances was used for the testing and the training. The table (7.3.1) below shows the results with TPR (True Positive Rate) and FPR (False Positive Rate), using the TP (True Positive), TN (True Negative), FP (False positive), FN (False Negative),

<table>
<thead>
<tr>
<th>Metric</th>
<th>Abbreviation &amp; Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive Rate</td>
<td>TPR(TP/TP+FN)</td>
<td>0.8077</td>
</tr>
<tr>
<td>False Positive Rate</td>
<td>FPR(FP/FP+TN)</td>
<td>0.3889</td>
</tr>
</tbody>
</table>

Table 7.3.1: Results
From the results, we have achieved 80% accuracy in classifying a good EULA as good and 61% in classifying a bad EULA as bad with our methodology. The other tools do not classify the EULA at all. Our results can be improved in future work by applying different techniques for feature selection. It is important to notice that the classification results were obtained on the basis of analyzing all possible features of the EULA text, not only on the basis of some predefined keywords and phrases.
CHAPTER 8: CONCLUSIONS

EULA Analyzer, which is a web-based application, finds the suspicious sentences based on some predefined keywords and shows characteristics of the text in EULA. For example, it shows the number of words, the number of word per sentence. Additionally, the Gunning Fog Index, Flesch Score and Flesch Grade indicate the complexity of the words that are used in the document. All of these metrics make use of different ranges of readability.

The EUL Alyzer look for the flagged sentences in the documents and on basis of those flagged sentences and the size of the text it assigns an interest ID to that particular text of EULA. It also gives a bit of detail about the analyzed document. But this tool doesn’t classify the document as either good one or bad one as same as EULA Analyzer. We concluded this because of the results. We have found that some of the good documents were also shown with the high Interest ID as they have many flagged sentences which user really don’t bother to read.

In the process of generating classifier, we trained our application with the training set, containing 448 EULAs (400 good and 48 bad) and the results which we achieved after the testing were good but not as accurate as what expected. This could be because of the technique we used for the feature selection (the use of Document Frequency (DF) which is an unsupervised scoring technique as a threshold for feature selection) and another reason might be because of the instances which are wrong representative of their class.

We can conclude with the results obtained that the classification of EULAs is a good approach compared to the existing EULA Analysis approaches because it detects spyware infested software. The results can be improved if we use a supervised scoring technique. Since this study is to make the prototype of a tool using the concept of machine learning for EULA classification and then to evaluate with the existing approaches, the results we obtained were not as accurate as in the previous study. So, to improve the results by using different techniques could be the part of future work. Presenting the results from the classification in a simple and understandable way to convince the user can be one of the future works. Now in the end we’ll like to summarize our work.

Spyware is a threat to the computer users and they are increasing in numbers. Mostly users don’t have a clear knowledge about them and they do install applications without going into the depth of what they are stating in the End User License Agreements, privacy policy or terms and conditions. As there is a threat of spyware so many antispyware also exists and they use different techniques against them. Some techniques are the detective and some are preventive. Always the preventive will be the better but there are not a lot of antispyware using this mechanism. As everything is mentioned in the software EULA and they are not so small so user can easily read them, so EULA classification is an important technique which was proposed [13]. In our study we tried to categorize all the existing EULA
classification techniques to make a flow of whatever things are done and where this EULA classification technique lies. After some discussion of existing EULA analysis tools we explained the term text classification. We implemented that technique to find either this technique is feasible or not. We compared the results with the other EULA analysis tools and then quantitatively we can say that the EULA Classification is a very good approach to prevent the installation of the EULA.
REFERENCES:


[4] EULA Analyzer
   http://www.spywareguide.com/analyze/
   Last checked 2009-05-01.

[5] EUL Alyzer
   http://www.javacoolsoftware.com/eulalyzerpro.html
   Last checked 2009-05-01.


[8] Preventing Malicious Spyware in the Enterprise
   Last checked 2009-05-10.

[9] Sandboxing
   Last checked 2009-04-05.


APPENDIX:

The snapshots of application for EULA classification which we made are as follows with the relevant description.

Snapshot (1)

The above snapshot if of the process while the generation of the Word vector documents it just requires the path of the folder containing the set of EULAs.
This above snapshot takes the word vector documents of both good class of the EULA and the bad class of the EULA to generate the document with TF_IDF frequencies for that class.
This application waits for the EULA at any port and then generates the result of Classification.
EUL Alyzer application which tells about the interesting keywords found in any software EULA.
Snapshot (6) EULA Analyzer

EULA Analyzer a web-based application to analyze the EULA.