IT HAS A EULA, IT MUST BE LEGIT

Stefan Catalin Hanu, Stefan Mosoi & Marius Lucaci
Bitdefender, Romania

Email {shanu, smosoi, mlucaci}@bitdefender.com

ABSTRACT

Over the last few years, a certain category of software has become more and more of a nuisance to AV labs and computer users alike: adware and potentially unwanted applications (PUAs). Walking the thin greyware line, these applications try (and sometimes succeed) to persuade even the AV labs that they are honest and trustworthy. Their motivation is monetary gain, so getting installed on as many computer systems as possible is a way to increase their earnings. This is often achieved by using dubious methods of distribution or using social engineering to trick the user into willingly accepting the software's installation.

Analysing greyware applications and taking a definitive decision on whether or not to block them is more than often a tedious job, combining research and both dynamic and static analysis. This paper explores the possibility of streamlining the analysis of PUAs by using some of the resources the developers of these applications utilize to justify their behaviour. The End-User License Agreement (EULA) and privacy policy can provide meaningful information about what an application might do. Using natural language processing (NLP) and other techniques, one can begin to distinguish some new patterns. By analysing more than 15 known adware families and their EULAs, we found this to be an effective method to discover new PUAs, even when using automated systems.

1. INTRODUCTION

We define Potentially Unwanted Applications (PUAs)[1] as computer programs that in some circumstances employ techniques that circumvent security measures or have a negative effect on the user's interaction with specific applications or actions. This is a rather broad definition, and includes different types of computer programs such as jokes, diallers, browser hijackers, spyware and adware. The latter has seen its ups and downs but the statistics for the last year are showing a disturbing trend, as shown in Figure 1. To better understand the recent evolution of adware, we considered the data from March 2013 as the baseline and determined the monthly growth. This baseline is shown as the 0 tick on the Y axis.

Advertising has been around for a long time, in different forms, always finding ways to interact with people and influence them. The Internet offers access to millions of potential 'victims' and this source of revenue is being exploited by legitimate companies and adware creators alike. Having as many users installing their software as possible is a way to increase their earnings.

This is often achieved by using dubious methods of distribution or social engineering to trick the users into willingly accepting the software's installation. Since the early days of adware

[1] Sometimes also referred to as Potentially Unwanted Programs (PUPs).

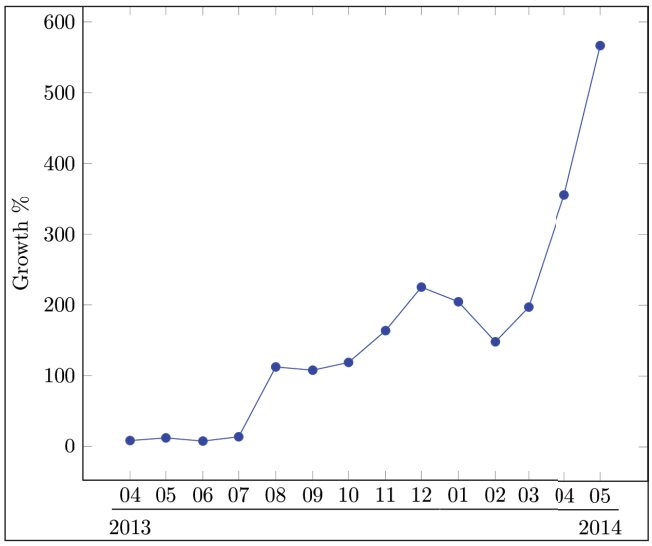

Figure 1: Monthly procentual evolution with 2013-03 as a baseline. (Data provided courtesy of Bitdefender.)

infections, we have noticed that most of the companies that produce this kind of software have changed their approaches. They are trying to benefit from the lack of a commonly accepted standard in the anti-virus industry regarding PUAs, sometimes caused by the risks imposed by legal actions. One way to achieve this is by enforcing the legal implications of the user accepting the End-User License Agreement (EULA) and privacy policy. This can also be used by adware companies to avoid judicial actions against them. Fortunately for us, this means that adware programs effectively come with a written testimony of how they are going to behave.

We investigate ways in which we can use the information willingly offered by means of the EULA and privacy policy in order to distinguish legitimate applications from adware or ad-supported software. In order to reach our target we took into consideration different technologies such as Natural Language Processing (NLP) and supervised learning.

1.1 Previous work

Our scope is to build a system that can automatically decide if a given executable file is adware or ad-supported, based on the EULA or privacy policy. The feasibility of this task has been proven in a pilot study [1] and a later paper [2]. We also found other applications that offer the possibility of classifying EULAs, using a different approach. One of them is EULAlyzer [3], a piece of commercial software available in both free and paid versions. The user must load the content of the EULA and start the analysis in order to receive the result, consisting of a list of words or sentences each given a score. The paid version also allows for dynamic recognizing and intercepting of EULAs from running installers.

The other application is similar to the free version of EULAlyzer, the main difference being that it's an online web page and 'uses only automated keyword recognition systems' [4]. Both EULAlyzer and EULA Analyzer rely on the user's

ultimate decision, making both of them unusable for large-scale automated analysis.

1.2 Outline

The remainder of this article is organized as follows. In Section 2, we describe our methods of collecting and interpreting data. The experimental results are presented in Section 3, followed by conclusions and future work in the last section.

2. EXPERIMENTS

Our goal is to create a system that can automatically classify a binary executable as being either clean or adware while providing more relevant information for the researcher. This means that it must first obtain a EULA from the input file and then compute a score that will help classify it. Even at this level of analysis, two problems emerge: obtaining the EULA, and understanding it. Our trials to overcome the latter are further discussed in Section 2.1, while data collection and processing are detailed in Sections 2.2 and 2.3.

2.1 Interpreting the EULA and privacy policy

Reading and understanding the contents of the End-User License Agreement is one of the recommendations computer users generally tend to follow the least. From a study proposed by Jeff Sauro [5] and confirmed by a later paper [6], we find that 'more than 50% of the users take less than eight seconds, which is clearly too short, to read the entire notice' [6]. So why is this happening, knowing that a 'EULA is a legal contract between you and the software publisher' [7]? For once, their presence in every piece of software or website, makes implicit approval an accepted part of installing an application. Furthermore, the legal terminology associated with them, combined with unnecessary and sometimes voluntary language obfuscation makes reading this contract a daunting task for the average user.

Most EULAs and privacy policies adhere to a common writing standard, and in our research, we found that the majority of the documents follow the same patterns. And patterns are great for creating algorithms that can tap into their predictability. So let us further investigate them from a human perspective. Starting from a macro-level, a EULA is nothing more than a document providing information about the relationship between a consumer and the software publisher, regarding a specific service or application. If we go deeper, we can distinguish a specific format in which the data is presented: sections and paragraphs related to a specific subject tend to be located in the same part of the document. This opens new possibilities for searching relevant data in a subset of the document to improve performance. Going even deeper, at the word-level, we can see that some adware-related behaviour is described similarly by using a relatively small subset of words. Making sense of all this might seem trivial for us, but constructing efficient algorithms that can correctly understand and flag parts of the EULA is more difficult than one might anticipate.

This is a text classification problem that we think is most suited for NLP processing and supervised learning. We will also research whether and how it is feasible to classify EULAs by using deterministic algorithms based on a database of features constantly maintained and improved by the researcher.

2.2 Data collection

Because the main target for adware producers is the *Windows* operating system, we will concentrate our efforts on processing binary executable files for this platform. The PUA collection was randomly chosen from a batch of files detected by at least three known anti-virus products or that were part of known adware families. The clean binaries were taken from different download websites known to have strict validation rules for the hosted products and tested by scanning with different AV products to ensure a 0% detection rate. All the files were processed using the same steps, as described in Figure 2, with the first operation requiring the filtering of Nullsoft Scriptable Install System (NSIS) [8] installers. We chose to handle NSIS files in a different way as they offer a simple manner of extracting the EULA from its decompiled script, making it possible to correlate the document to the installer itself and not a bundled file. The next step is to send all the binary files, including the NSIS installer (if applicable), to a generic unpacker so that we can extract Internet addresses from any file contained within. This should cover the cases where an installer bundles adware components.

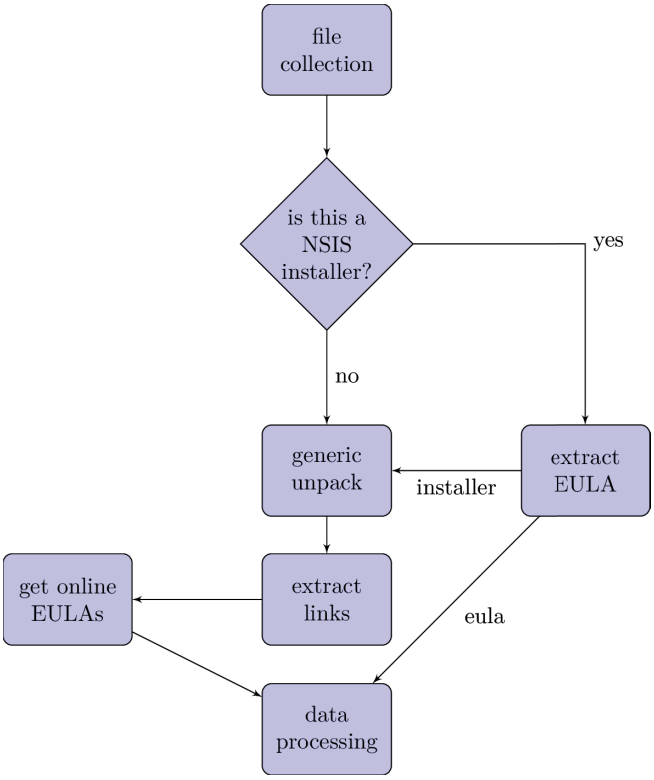

Figure 2: Data collection flow chart.

When analysing a possible piece of adware, the extraction of web links from the executable can give clues as to its behaviour and the source website. This usually allows the researcher to find more information such as a download link for the parent installer and the EULA or privacy policy. In addition to using

different regular expressions for Internet address extraction, we also searched for words and word combinations that might be part of a website link or IP address. An example of the flow that our script would follow is shown in Figure 3. This represents a binary view of the .rdata section of a 32-bit Portable Executable (PE) binary [9] member of the 'Gamevance' adware family. The yellow box highlights a domain that is extracted by our script.

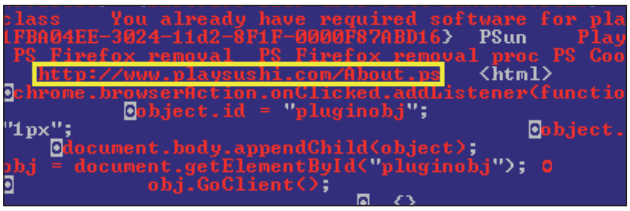

Figure 3: Sample slice of an adware binary view.

If the extracted link does not point to a valid EULA or privacy policy, we parse the domain main page and search for keywords in the links shown on the page that might give away the presence of a EULA.

Using this flow, we processed 1,028,146 PE binaries, obtaining 52.49% clean and 47.51% adware EULAs. This distribution was obtained by correlation with the verdict of the document's parent executable and by filtering manually. After obtaining this new collection of ASCII files we went further with processing them, as described in the next section.

2.3 Data processing

At this stage, our goal is to obtain interpretable data from the collected text files. In order to achieve this, we went with what NLP has to offer in the form of the Natural Language Toolkit (NLTK) [10] library for Python. This gives access to several functions that streamline interaction with some of the most commonly used tasks in NLP, like part-of-speech tagging and word sense disambiguation.

Every language has its own semantic and grammar rules, and most of the functions and algorithms backing NLTK have been trained on English documents. This makes it only natural that we should ignore any sample written in a different language. For this task we chose the langid Python library, which uses 'cross-domain feature selection for language identification' [11].

One challenge was representing the content of EULAs and privacy policies in a meaningful manner. The first thing that comes to mind is getting all the words and using them as features for a supervised learning algorithm, but that wouldn't be very effective because of language obfuscation. In order to overcome this we started by splitting the text into sentences and individually tokenizing them, as summarized in Figure 4. Using the extracted data, we obtain the part-of-speech tags in the form of two-element tuples that we have stored in order to use in a later step. Next, we want to strip any stop words because at this point they are nothing more than noise that will ultimately have a negative effect on our statistics.

The last step in preparing our data is to reduce the remaining words to a simpler form. For this task, we have a variety of stemmers, such as Porter [12], Lancaster [13] and Snowball

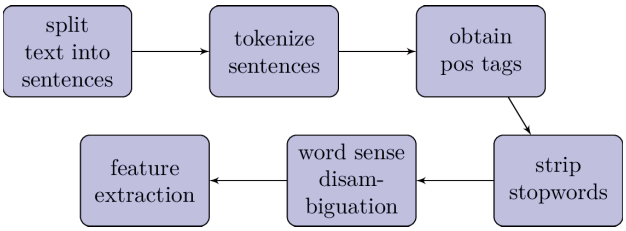

Figure 4: Data representation flow chart.

[14], or we can choose a lemmantizer. Stemming algorithms have been around for quite some time and are still being used, mainly because in some cases they offer the best option in terms of the performance-result ratio. They work by removing or replacing suffixes in order to obtain a common base form of the word, as opposed to lemmatization algorithms. The latter try to get to a common root by first acquiring the part of speech and then using it to choose different normalization rules, depending on the case. After rigorous testing, we ended up using the WordNet synonym ring (synset), which in turn uses the WordNet Database [15] and allows for accessing lemmas in order to get a canonical form of the word. For most of the relevant terms, the WordNet synset approach returns excellent results, such as in the case of the 'ad' word variations as seen in Table 1.

Original word	WordNet	Porter	Lancaster	Snowball
advertisement	ad	advertis	advert	advertis
advertizement	ad	advertiz	advert	advertiz
advertising	ad	advertis	advert	advertis
advertizing	ad	advert	advert	advert
advert	ad	advert	advert	advert
advertise	advertise	advertis	advert	advertis
advertiser	advertiser	advertis	advert	advertis

Table 1: WordNet compared to common stemmers.

In this case, the Lancaster stemmer manages to return an identical form for all the variants, and a comparison of these results with those from the WordNet approach might seem to come out in favour of the first algorithm. But we do not believe that this is the best outcome, since we lose the meaning in a sentence. This is not the case when using synsets – here, all the nouns referring to advertisements reduce to a common 'ad' word, the action term is reduced to itself, and the word defining the entity keeps its form. This is ideal for us because of the approach we used to generate the initial form of the features that will later be fed to our supervised learning algorithm. To achieve this we looked at the rather standard method of computing the word frequencies for the entire EULA lot and selected 10,000 of the most commonly encountered terms. This is usually enough to give clues to the content of a text file, but doing so will also strip any sense markers. So we went a step further and manually constructed a list of keys that are specific to adware and advertising, such as 'ad', 'offer', 'contextual',

'collect', 'relevant' and others. Using the already tokenized data, we created all the combinations of two and three words from a sentence that contains one of the manually selected terms and computed the frequency of such groups. As in the case of the word frequencies, we selected the first 10,000 most common groupings to enrich our feature list, now growing to a quite large 20,000 element array. The reasoning behind creating these word combinations is to extract more contextual sense from a sentence, rather than the whole text, as most of the relevant information in some adware EULAs is contained in no more than a few sentences or phrases. This approach can also make a difference when trying to analyse a EULA that makes reference to advertising on the website, even if the software itself is clean, by associating keywords with, for example, the term 'website'.

3. RESULTS

Having obtained our initial feature list, we continue by using the One Side perceptron [16]. The decision to use this algorithm in favour of others was made bearing in mind the consistent results in similar tasks and its optimizations for lowering the number of false positives. The tests were conducted using different subsets of features from the initial 20,000 element set. These subdivisions were constructed by using four different feature selection algorithms: F1 [17], F2 [17], AbsProcDiff and ProcDiff. For the first three we chose 500 of the most relevant features and for the fourth we used a 200-100-200 split by selecting the first 200 most applicable features for each of the clean and adware classifications and a 100 slice from the middle of this distribution.

This pick is advantageous for our purpose, knowing how ProcDiff works. It computes the difference between the percentage of clean and adware samples that share the same feature, creating an ordered list where we have representative features for clean samples at one end, and for PUAs at the other. In the middle of the list we should find features that are common to both classifications, which are needed in order for the perceptron to better make feature connections.

AbsProcDiff works in the same way, representing only the absolute values obtained with ProcDiff. The results can be seen in Table 2.

Feature selector	FP	Se	Acc
F1	10	92.86%	96.49%
F2	13	94.65%	97.32%
ProcDiff	12	96.60%	98.24%
AbsProcDiff	15	94.88%	97.44%

Table 2: One Side perceptron test results.

The 'FP' column represents the number of false positives, while 'Se' is the sensitivity and 'Acc' the accuracy. It's obvious that using the ProcDiff feature selector offers the best results in terms of both detection and FP number.

Using the information from the supervised learning experiment we tried to build a deterministic algorithm that could flag a EULA or privacy policy as being clean or adware, based on some simple rules. It's the researcher that must improve the detection by maintaining a list of words and word groups and associate them with a score. As an initial rule set, we hand-picked some of the groupings from the ProcDiff feature selector that were found only in adware documents. The detection rate for this experiment was 82.5% with an FP rate of 0%, proving that this approach is a reliable way of detecting bad EULAs.

4. CONCLUSIONS AND FUTURE WORK

We started by researching new ways of improving the analysis process for adware files with the use of the data provided by the EULA and privacy policy and managed to produce viable and reliable detection algorithms.

Using the One Side perceptron we managed to obtain a 96.60% detection rate with few false positives, and the deterministic approach seems promising. Of course, there are some ideas still not implemented, like company name extraction from EULA and privacy policy documents. We managed to spot a pattern in the way some companies use their name or that of the software with a high frequency. This could be used to exclude legitimate business or increase the likelihood of adding a detection for known adware providers. Unfortunately, at this time, we don't have enough data to prove this to be effective.

At this time, the described algorithms and procedures are being used as a pilot program tapping into the malware stream while trying to detect new adware families. This assures continuous development and the addition of new features to the platform while the EULA collection increases, providing a better training environment.

REFERENCES

[1] Lavesson, N.; Davidsson, P.; Boldt, M.; Jacobsson, A. Spyware prevention by classifying end-user license agreements. In New Challenges in Applied Intelligence Technologies, pp.373–382. 2008.

[2] Lavesson, N.; Boldt, M.; Davidsson, P.; Jacobsson, A. Learning to detect spyware using end user license agreements. Knowl. Inf. Syst., 26(2):285–307. 2011.

[3] BrightFort. EULAlyzer. http://www.brightfort.com/eulalyzer.html.

[4] A. S. Labs. EULA Analyzer. http://www.spywareguide.com/analyze.

[5] J. Sauro. Do Users Read License Agreements? http://www.measuringusability.com/blog/eula.php. 2011.

[6] Böhme, R.; Köpsell, S. Trained to accept? A field experiment on consent dialogs. In CHI, pp.2403–2406. 2010.

[7] Desautels, E. Software License Agreements: Ignore at Your Own Risk. 2005.

[8] NSIS. http://nsis.sourceforge.net/Main_Page.

[9] https://www.virustotal.com/en/file/7d6d5437d1111b6e882934fa0f48552b/analysis/1402225161/.

[10] Natural Language Toolkit. http://www.nltk.org/.

[11] Lui. M.; Baldwin, T. Cross-domain feature selection for language identification. In IJCNLP, pp.553–561. 2011.

[12] Porter. http://tartarus.org/~martin/PorterStemmer/.

[13] Lancaster. http://www.comp.lancs.ac.uk/computing/research/stemming/Links/paice.htm.

[14] Snowball. http://snowball.tartarus.org/.

[15] WordNet Database. http://wordnet.princeton.edu/.

[16] Gavrilut, D.; Benchea, R.; Vatamanu, C. Optimized zero false positives perceptron training for malware detection. In SYNASC, pp.247–253. 2012.

[17] Loong, S. N. K.; Mishra, S. K. De novo svm classification of precursor microrna from genomic pseudo hairpins using global and intrinsic folding measures. Bioinformatics, 23(11):1321–1330. 2007.