Poster: Static Malware Modeling and Detection using Topic Models

Shamik Bose
Department of Computer Science
Florida State University
Tallahassee, Florida 32306
Email: bose@cs.fsu.edu

Angela Gibbens
Department of Computer Science
Florida State University
Tallahassee, Florida 32306
Email: ag11z@my.fsu.edu

Xiuwen Liu
Department of Computer Science
Florida State University
Tallahassee, Florida 32306
Email: liux@cs.fsu.edu

***Abstract*—Malware detection has been a problem for security researchers for a very long time. Signatures are very widely used, but they have inherent limitations when it comes to detecting new malware since signatures use sequences of instructions to detect malware. In this poster, we present a method to look at the underlying topic structure of binaries and classify files as malicious or benign based on its structure.**

***Keywords*—Malware detection, Topic models**

I. Introduction

The first malware, called the Elk Cloner, was found in 1982, and malware has been on the rise ever since. Malware has been defined as "software that deliberately fulfills the harmful intent of an attacker" [1]. Current malware detection techniques primarily use signatures where malware samples can be identified by the use of sequences of instructions contained in them. These are effective at detecting malware that has already been analysed, but has a shortcoming when faced with new malware. Since there are practically unlimited ways to achieve functional equivalence using instruction sequences, malware is often mutated to evade detection. In this poster, we propose a novel method of detection by examining the underlying 'themes' of malware. Our preliminary results show that the proposed method could lead to significant improvements.

II. Proposed Model

The basis underlying our proposed system is that there are styles in all software, malicious or otherwise, and that these styles differ significantly between malicious and benign code. This is supported by the findings in [7], where Linstead *et al.* found that certain commands are used heavily for certain "concerns" or aspects of code as part of their experiment on a repository of Java source code. As an example, the aspect of $ExceptionHandling$ had the words **exception**, **illegal**, **argument**, **runtime** and **pointer** as the top words whereas the aspect of $Logging$ had the words **error**, **log**, **debug**, **string** and **throwable** as the top words. The concerns or aspects effectively represent each topic in the source code. The 'style' refers to a distribution of themes or topics over the code. In our system, we discover the underlying themes using Topic Models and then classify the binaries according to this structure.

Topic models are a suite of algorithms which are used to compute hidden topic structures from a set of documents [4](in our case, executable binaries). A *topic* is defined as a distribution over a set of words, or *vocabulary*. So, a topic like *computer science* would have words relating to computer science with high probability. The only observation is the words in the documents; everything else like the topic-per-document, words-per-topic and the topics themselves are *hidden*. This takes place in two stages:

- **Estimation**: In this stage, the algorithm is provided with a set of documents, an α value which determines the nature of the peak of the underlying Dirichlet distribution and the number of topics in the distribution. The algorithm then calculates per-document and per-topic word distributions and assigns words to each topic. This is then saved in a model file
- **Inference**: This stage takes as input the documents for which the underlying topic structure is to be inferred, according to the model generated. It generates a file with the gamma values for each topic for each document, which indicates the presence of a particular topic based on the word correlations found in the model

III. Implementation

For preliminary results, we used a small sample of binaries containing both malicious and benign files. The malicious sample was taken from the MALICIA dataset [5], a dataset of 11,000+ malware samples collected from drive-by downloads. The benign set was taken from the System32 folder in Windows 7. The benign and malicious set of binaries each contained a comparable number of samples to ensure that there is no bias towards any particular family of binaries. The general workflow is shown in Fig. 1. The steps of our experiments are as follows:

- The first step is to generate the topics
  - Disassemble binaries to x86 assembly code
  - Strip binaries of operands like labels, memory locations and registers
  - Convert disassembled file to format required by the LDA program [2]
  - Run LDA estimation on converted files
- The next step is to infer the topics in each file.
  - Run LDA inference on all the disassembled, converted binaries

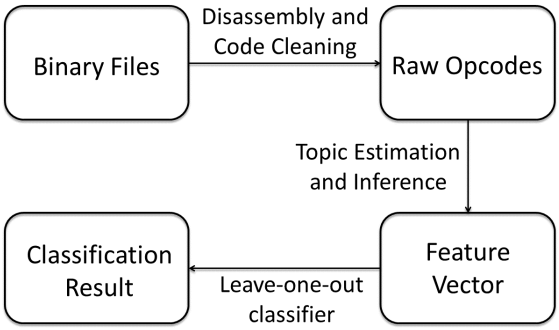

Fig. 1. Implementation Workflow

  - This produces a gamma value for each topic in each document; the higher the gamma value, the higher the probability that the document contains that topic
- The final step is the discriminative process of classifying a given file as malicious or binary
  - We normalize the gamma values to be in the range $[0, 1]$; this gives us a 10-dimensional vector for each file, where each of the values correspond to the normalized gamma distribution
  - We associate a label with each vector, marking it as malicious or benign
  - We then run a leave-one out evaluation as a means of cross-validation on the dataset, using a basic nearest-neighbour classification with the Euclidean distance in the 10-dimensional space. The leave-one-out evaluation enables us to simulate a zero-day attack since the file being classified is not in the feature space and does not exactly match the topic model for another sample

IV. Results

Our experiments show some interesting results. First among these is that with the same vocabulary, the topics vary significantly between malicious and benign files as seen in Figs. 2 and 3 which show the top 10 words from the topics in each type of binary. This confirms our basis that the underlying thematic structures are different between malicious and benign files.

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
int3	push	mov	push	mov	add	add	mov	mov	mov
add	call	push	mov	push	push	or	push	call	push
mov	inc	call	pushl	call	mov	xchg	pop	push	pop
push	add	pushl	je	pop	nop	mov	add	lea	add
call	lea	lea	cmp	cmp	je	sub	dec	add	adc
pop	pop	je	and	jne	inc	dec	inc	pushl	and
jmp	je	test	call	je	pop	inc	and	jmp	cmp
je	mov	cmp	jne	jmp	xor	in	xchg	ret	inc
cmp	jne	xor	jb	test	cmp	xor	cmp	pop	xor
lea	dec	jmp	gs	ret	dec	imul	xor	cmp	sbb

Fig. 2. Benign Topics

Classification accuracy of the leave-one-out evaluation is 94.44% as shown in Table I, using the feature vectors generated from the topic models. According to [6], ESET NOD32 shows the highest accuracy of zero-day detection with 62% while others like AVG, Kaspersky and MalwareBytes can detect less than 50% of zero-days. While our method still has a lot of room for improvement, our results suggest that it could lead to better detection of malware.

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
push	mov	mov	xor	mov	pop	jo	mov	add	add
pop	cmp	push	pushl	xor	push	mov	push	jmp	mov
call	inc	xchg	repz	movl	mov	push	int3	out	push
add	add	pop	jo	or	inc	pop	add	jne	inc
pushl	dec	dec	jmp	add	out	call	jmp	cld	or
nop	ret	sbb	push	sub	xchg	sub	jne	xlat	and
mov	or	sub	call	and	or	xor	call	jbe	adc
ret	xor	adc	lds	imul	jmp	jmp	je	push	sub
sub	cli	xor	ret	cmpl	add	dec	cmp	ljmp	xor
lea	jne	and	test	shl	dec	add	test	inc	cmp

Fig. 3. Malware Topics

TABLE I
Classification Results

	Correctly Classified	Misclassified
Malware	77	4
Benign	76	5
Total	153	9

V. Future Work

We have used the default settings for the LDA program. The settings contain a parameter α, which determines the peakiness of the underlying Dirichlet distribution. Analysis of disassembled files could help us determine what α values to use to get better estimates for the topic models.

The classifier that we are using is a basic nearest neighbor classifier. With more sophisticated classifiers like SVM or neural networks, we anticipate further improvement in classification. We are also going to extend this to other platforms like x64 and ARM, considering that mobile platforms are being increasingly targeted by malware.

We are also going to be using n-grams of opcodes as a 'word' for future experiments.

References

[1] Moser, A. and Kruegel, C. and Kirda, E., Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual, *Limits of Static Analysis for Malware Detection*, *2007*,

[2] David M Blei and John D Lafferty. Topic models, *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *The Journal of Machine Learning research*, 3:993–1022, 2003

[4] David M Blei. Probabilistic Topic Models, *Communications of the ACM*, 2012

[5] Antonio Nappa and M. Zubair Rafique and Juan Caballero. The MALICIA Dataset: Identification and Analysis of Drive-by Download Operations, *International Journal of Information Security*, Springer Berlin Heidelberg, 2014

[6] CRDF Threat Center https://threatcenter.crdf.fr/?Stats

[7] Linstead, Erik and Hughes, Lindsey and Lopes, Cristina and Baldi, Pierre, *Software analysis with unsupervised topic models* 2009