

# Intelligent Virus Detection on Mobile Devices

Deepak Venugopal, Guoning Hu, and Nicoleta Roman

**Abstract**— In this paper, we describe a new solution for detecting mobile phone viruses. The solution is based on Bayesian decision theory using heuristic rules derived from common functionalities among different virus samples. Specifically, we detect viruses according to the DLL usage of a program, which is directly linked to the functionality of this program. Our solution is able to detect unknown viruses, especially the variants of existing ones. We evaluate our solution on the Symbian platform, where most viruses are present in the wild. We constructed a virus detector based on DLL functions from a small set of virus samples and obtained 95% detection on all the viruses in the mobile domain and 0 false positives on nonvirus programs.

**Index Terms**— Mobile, Virus detection, DLL, Common functionality, and Security

## I. INTRODUCTION

Traditionally, virus detection is done by identifying a signature, i.e., a unique sequence of binary code, in a virus. The major disadvantage of this method is that one usually needs a new signature to detect a new virus, even a variant of an existing virus. The problem was aggravated in the PC domain with the introduction of the Dark Avenger's Mutation Engine [1], a polymorphic virus that changes its signature every time it replicates. Accordingly, the cost of detection is a concern for signature-based detection algorithms since adding a new signature requires more memory and processing time and a new signature needs to be added for every virus in the worst case. The importance of this cost is more pronounced in the mobile domain where system resources are constrained.

A lot of interest has been generated in virus detection for mobile phones recently with the outbreak of Cabir and Commwarrior viruses. The mobile phone platforms now are offering a rich set of features for application programming. In particular, new 3G networks offer high bandwidth for data transfer, which increases the services offered to the client devices [4] [5]. This opens the door to more serious virus attacks in the future and makes security against malicious content like spreading-worms a necessity nowadays on these

devices. Most of the previous methods on the virus detection at mobile phones are signature-based. As a result, they are limited by their cost of detection and capabilities in detecting new viruses. In fact, to our knowledge, there is no previous work specific to the detection of unknown mobile phone viruses.

Heuristic scanning is an advanced form of virus detection that detects viruses based on their common characteristics [6]. It is capable of detecting new viruses that have not been analyzed for signatures, especially variants of existing viruses. Tesauro [2] developed a neural-network based solution for detecting boot-sector viruses in PC. Although the proposed method performed well on this task, it did not generalize to detect viruses in the entire PC domain. Schultz et al. [3] proposed various data mining techniques to identify large sets of PC viruses and obtained some success.

In this paper, we propose a heuristic method that detects viruses in the mobile domain. Our aim here is to find simple heuristic rules that require low resources to detect viruses. On the other hand, these rules shall be effective in detecting existing viruses as well as new viruses, especially the variants of existing ones.

The spreading mechanism and targeted exploits of viruses in the mobile domain are different from those in the PC domain. Hence, we found that existing heuristic methods, which were developed for detecting PC domain viruses cannot be directly applied to the mobile domain. Also, the classification-based techniques described in [2] and [3] require a huge amount of training data to offer reliable results. However, the number of virus samples existing in the mobile domain is scarce and far from sufficient for such training. Hence, a simple detection model is required to avoid the problem of over-fitting the training data. We observe that most viruses in the mobile domain demonstrate common functionalities. e.g., deleting system files and sending MMS messages. To implement these functions, a program needs to use certain dynamic link libraries (DLLs). The list of DLL functions used by a virus indicates the behavior of the virus in terms of its functionality. Therefore, we approach the detection of viruses by exploiting the common patterns of DLL usages among viruses. Our approach is able to detect new viruses that have the similar functionalities as existing ones. In addition, since these DLL functions are easy to extract from the executable files, our method is computationally efficient.

Section II gives details of our proposed method. Section III evaluates our method on the Symbian OS platform and the last Section concludes our paper.

---

Manuscript received April 4, 2006. This work was done in SMobile Systems, Columbus Ohio as a part of the Mobile Anti Virus Research effort. Deepak Venugopal is a Researcher with SMobile Systems, Columbus, OH 43212 USA (phone: 614-432-3016; e-mail: deepak@fb-4.com). Guoning Hu, is a Researcher with SMobile Systems, Columbus, OH 43212 USA. (e-mail: hu@cse.ohio-state.edu). Nicoleta Roman is with SMobile Systems, Columbus, OH 43212 USA and a visiting assistant professor at The Ohio State University at Lima(e-mail: roman.45@osu.edu).

## II. PROPOSED MODEL

The intelligent virus detection method described here aims to identify a virus based on its functionality or behavior. Mobile viruses are classified into different families or classes based on their functionalities. A completely new virus which is not a variant of any existing type starts a family of its own. All virus variants in a family share a common malicious core behavior. For example, “Skulls” viruses disable device applications and “Commwarrior” is a worm that spreads using MMS messages. Here, we build a classifier for each family to detect all the viruses in that family. Alternatively, one may consider building a universal classifier for all the viruses. However, there are two problems of building such a classifier. First, it is unlikely that there is any common core among all the viruses. Second, such a classifier tends to be biased by dominant virus families, i.e. the family with most samples, in the training set.

As we have discussed in Section I, the DLL functions used by a virus give us a good insight into the functionality of this virus. Therefore, we use the imported DLL functions as features for our virus detection. The total number of DLL imports obtained from an executable file is usually very large (from hundreds to thousands). We therefore need to extract DLL functions that are related to the core functionality, which is the key for us to build a reliable classifier and ensures that it is not over-trained on the existing data. We first use knowledge based feature reduction where we eliminate imports known to occur commonly in all executables including nonvirus programs, e.g., functions that are used by the platform to load an application. Since each variant in a family can have its own specific functionality such as a graphical interface, we then eliminate these DLL functions that generally do not contribute to the malicious behaviors. By eliminating such DLL functions, we obtain the core DLL function set for each virus. All the core functions in a particular family compose a feature set for this family. These features are represented as binary features and used as input to the associated classifier described below. Specifically, for an executable file, the input to the classifier is a binary vector that completely specifies the occurrence of each DLL function in the feature set, i.e., 1 when it occurs and 0 otherwise.

We adopt a Bayesian classifier to detect the viruses in each family [7] [8]. Let  $H_0$  be the hypothesis that an executable file contains malicious code and  $H_1$  otherwise. Let  $C_{00}$  be the cost of correct virus detection,  $C_{11}$  that of correct rejection,  $C_{10}$  that of missing or false rejection, and  $C_{01}$  that of false alarm. To minimize the total cost we use the following decision rule:

$$\frac{P(H_0 | A)}{P(H_1 | A)} > \frac{C_{01} - C_{11}}{C_{10} - C_{00}}, \quad (1)$$

where  $A = \{a_n\}$  is the input binary vector representing the DLL feature set described above,  $a_n$  is the binary value of a particular feature  $n$ , and  $P(H_i | A)$  is the posterior probability of the hypothesis  $H_i$  given the input.

Applying the Bayesian rule to (1), we write the posterior probabilities in terms of likelihood as:

$$\frac{P(A | H_0)}{P(A | H_1)} > \frac{C_{01} - C_{11}}{C_{10} - C_{00}} \frac{P(H_1)}{P(H_0)} = \alpha, \quad (2)$$

where  $\alpha$  is a parameter that depends on the prior probabilities as well as the cost functions.  $P(A | H_i)$  is the corresponding likelihood.

At this moment, we do not have the accurate values of the cost and the prior probabilities, except that  $C_{10}$  and  $C_{01}$  are much larger than  $C_{00}$  and  $C_{11}$ . From user’s feedback, we know that false alarm is really undesirable and therefore we assume  $C_{01}$  is much larger than  $C_{10}$ . With this assumption,  $\alpha$  is a large number. As suggested by the feedback from users, the value  $\alpha$  is chosen to be the minimum value that yields less than 1% false alarm rate.

To compute the likelihood  $P(A | H_i)$ , we assume that the individual features, i.e., the DLL functions in the feature set, are independent. Therefore, we have,

$$P(A | H_i) = \prod_n P(a_n | H_i), \quad i = 0, 1 \quad (3)$$

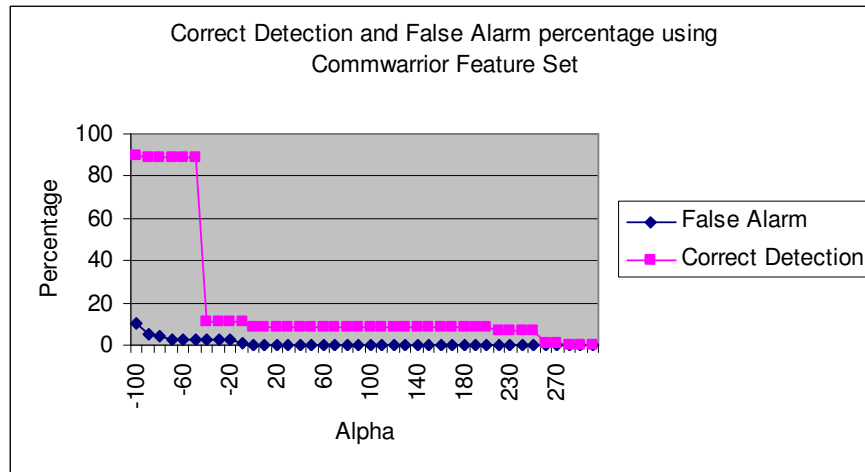
One may consider the dependency among features in addition. However, to build such a model requires much more training data, which we do not have at this moment. The likelihood of a feature,  $P(a_n | H_i)$ , corresponds to the occurrence frequency of the corresponding DLL function in the training samples. Let  $M$  be the total number of training samples of a particular virus family or nonvirus programs and  $m$  that of samples where feature  $n$  is 1. We have

$$\begin{cases} P(a_n = 1 | H_i) = \frac{m}{M} \\ P(a_n = 0 | H_i) = 1 - \frac{m}{M} \end{cases}, \quad i = 0, 1 \quad (4)$$

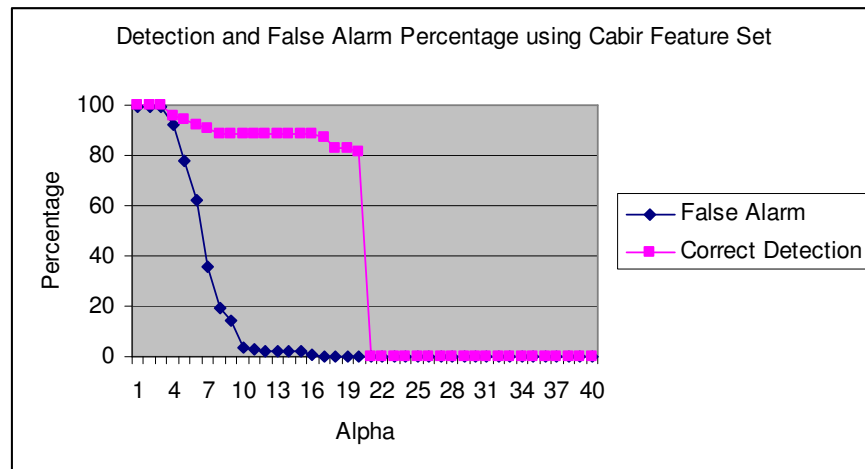
## III. RESULTS

Symbian OS is the most widely used platform for mobile devices. It is also the mobile platform where maximum number of viruses has been discovered. We evaluate our solution using all the Symbian-OS viruses discovered so far. The existing Symbian viruses are put into different families based on their functionalities. Currently, we have altogether 140 virus samples and 260 nonvirus samples. We use 27 virus samples and 95 nonvirus samples for training and the remaining for test.

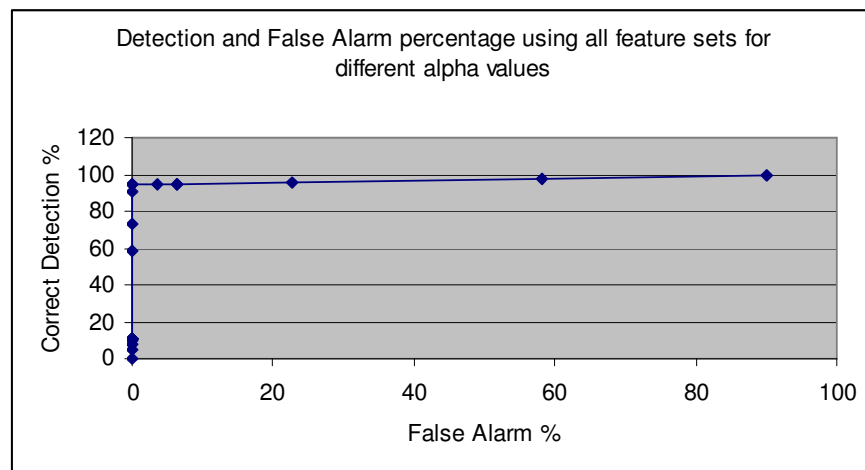
In principle, a separate DLL import set would be required for each of these classes. However, in the course of our experiment, we discovered that there are common functionalities among different families and the classifier of one particular family in fact detects viruses from multiple families. Consequently, the number of individual classifiers is greatly reduced.



**Figure 1. Percentages of detection and false alarm for Commwarrior family**



**Figure 2. Percentages of detection and false alarm for Cabir family**



**Figure 3. Percentages of detection and false alarm for varying alpha**

In particular, we obtained DLL import sets from the Cabir and Commwarrior virus samples. As discussed before, the parameter  $\alpha$  was initially chosen so as to limit the false alarm rate to be no greater than 1.0 %. In practice we obtain 0% false alarm for a comfortable range of thresholds. The percentages of detection and false alarm for the Commwarrior and Cabir families as shown in Figs. 1 and 2, respectively. The percentages of detection shown in these figures are in fact those of all the viruses in the test part.

When we apply only the 2 classifiers derived from these two families to all the families in the test part, we obtain 70% detection and 0% false alarm. This shows that there is much common functionality among viruses in different families. Finally, we add 3 more DLL import sets and obtain 95% detection and 0% false positives as illustrated in figure 3.

As an example of generalization, this method can detect the latest variant of cabir named Cabir.aa and the latest variants of Commwarrior named Commwarrior.C using the classifiers derived from the training set with these two viruses. The signature detection method on the other hand would need a new signature for Cabir.aa and another for Commwarrior.C as an update. Moreover, a signature-based method requires 75 signatures to detect all the samples in the virus dataset whereas our proposed method requires only 5 DLL import sets. Therefore, we found the proposed detection method is computationally more efficient. Our result shows that many virus families have common core functionality and our detection method is capable of determining this functionality and using it to detect virus.

#### IV. CONCLUSION

Protecting Mobile devices against viruses is gaining enormous attention nowadays due to an increase in the range of services that is being offered on these devices. In this paper, we have described a method to detect mobile viruses based on common functionality. Specifically, we used DLL import sets as features to detect viruses. Our evaluation on Symbian-OS platform shows that with a small set of training data, our system detects most existing viruses with a 0% of false positive. Future work will try to explore other representations of behavior that may be used to generalize detection.

#### ACKNOWLEDGMENT

The authors would like to thank SMobile Systems Inc. for their support in this work.

#### REFERENCES

- [1] S. Gordon, "Inside the mind of the Dark Avenger," *Virus News International*, 1993
- [2] G. Tesauro, J.O. Kephart, and G.B. Sorken, "Neural Networks for Computer Virus Recognition," *IEEE Expert*, vol. 11, pp. 5-6, 1996.
- [3] M.G. Schultz, E. Eskin, E. Zadok, and S.J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," *IEEE Symposium on Security and Privacy*, 2001.
- [4] P. Rysavy, "The evolution of Cellular data: On the road to 3G," *Intel Corporation*, 1999.
- [5] B. Bhargava, "Preparing a 3G business case," *RMR PLC conference*, 2001.
- [6] P. Szor, *The art of computer Virus Research and Defense*, Symantec press, pp. 225-295, 2005
- [7] S.J. Russel and P. Norvig, *Artificial Intelligence: A modern approach (2<sup>nd</sup> Edition)*, Prentice Hall, 2002.
- [8] D. Szafron, R. Greiner, P. Lu, D. Wishart, C. Macdonell, J. Anvik, B. Poulin, Z. Lu, and R. Eisner, "Explaining Naïve Bayes Classification," *Technical Report*, Department of Computer Science University of Alberta, 2006.