Malware Analysis using GPGPU

Mayank Chaudhari,and Jane Doe,

Department of Computer Science and Information System,

BITS Pilani, K K Birla Goa Campus,

Zuari Nagar, 403726, Goa, India

Abstract—The abstract goes here.

Index Terms—Malware Detection, GPGPU, Machine Learning, Computer Security.

1 Introduction

Malware refers to a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of the victims data, applications, or operating system or of otherwise annoying or disrupting the victim [1]. There has been a significant development in the field of malware creation in terms of writing highly complex and undetectable malware and based on the complexity of malware they can be categorized as first generation and second-generation malware. In first generation, structure if malware does not change, while in the second generation, structure changes to generate new variant, keeping the action same [2]. On the basis of how variances are created in malware, second generation malware are further classified into Encrypted, Oligomorphic, Polymorphic and Metamorphic Malware [3]. According to 2017 Threat analysis report they has now more than 780 million malware samples in their database and the new malware increased by 10% in third quarter to 57.6 million. In addition to it 60% increase in new mobile malware is also observed in the third quarter of 2017 due to large increase in Android screen locking ransomware [4]. The Symantec 2017 Internet security Threat report indicates that there were 357 million new malware variants, 3.6 thousand new mobile malware variants were detected and every two minutes an IoT device is being attacked [5]. The recent Internet security Threat Report [6] from Symentec shows an increase of 88% in overall malware variants. The increase in malware variants is linked to the interest of malware writers in ICS, cryto currency mining and banking sector. The report states an increase of 92% in script and macro down-loaders, 46 % increase in ransomware variants.According to the threat report coin mining was the biggest area in cyber crime with an increase of detection figure up to 8,500%. This increase in malware threat can be correlated to the increasing use of worldwide web due to the exponentially increasing mobile and IoT devices resulting increased attack surface. The malware attack/threat are not only

- *M. Shell is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

limited to individual boundaries, but they are highly skilled state funded hackers writing customized malicious payloads to disrupt political, industrial working and military espionage [5] [7] [8]. The most high-profile, subversive incident of the year was a series of intrusions against the Democratic Party, which occurred in the run-up to the 2016 US presidential election [5].

It is indubitable fact that various traditional (signature based) approaches are ineffective to combat the dynamic and complex behavior of second generation viruses. If adequate advancement in anti-malware techniques are not achieved, consequences at this scale (more than 56.6 million new viruses are reported in one quarter year [4] in 2016) at which new malware are being developed can create fatal affects and the results will we more severe then past as due to more reliance on digital world. The second-generation malware contain very complex structure and advanced obfuscation techniques to make the detection process harder and counter the treat. Recently in 2107 the WannaCry ransomware drew the attention of researchers from all over world causing a lot of trouble. Another major threat I CCleaner was disarmed before it would have caused any widespread harm, despite an estimated 2.27 million infections. Therefore, there is a need that researchers and anti-malware developers should work side by side to counter the threat/attack from the new malware. The most widely used malware detection engines are based on signature-based detection, malware normalization, heuristic based detection, machine learning, etc. [3].

In recent years, various machine learning techniques has been proposed by authors [9] [10] [11] [12], which can enhance the capabilities of traditional malware detection engines(signature based detection) but, with the use of a complex machine learning based anti-malware engine the detection time increases. This can result in inefficient resource utilization and less throughput hampering the performance of whole system. Hence, in this paper we discuss an approach to detect malware with high throughput and accuracy. We present a static malware analysis technique using GPGPU for faster detection of malware based on the approach proposed by authors in [11]. We were able to achieve a maximum speedup of 120 over the actual implementation [11] and achieving the same accuracy mentioned by authors. The remaining work is organized as follows. Section 2 discuss about the related work, in section 3 we discuss the dataset description, data preprocessing and feature selection. Section 4 contains the brief description of the Nave Byes and detection technique. Section 5 describes proposed approach for parallel implementation of Nave Bayes using CUDA architecture. Finally, in section 6 contains conclusion and future scope of work.

2 Related Work

To combat the threats/attacks from the second generation malware, Schultz et al. (2001) was the first to introduce the concept of data mining to classify the malware [13]. In 2005 Karim et al. [14] addressed the tracking of malware evolution based on opcode sequences and permutations. In 2006, O. Henchiri et al. [15] reported 37.17% detection accuracy from a hierarchical feature extraction algorithm by using NB (Naive Bayes) classifier. Bilar (2007) uses small dataset to examine the opcode frequency distribution difference between malicious and benign code [16] and observed that some opcodes seen to be a stronger predictor of frequency variation. In 2008, Yanfang Ye et. al. [17] applied association rules on API execution sequences and reported an accuracy of 83.86% with NB classifier. In 2008, Tian et al. [18] classified the

Trojan using function length frequency and shown that the function length along with its frequency is significant in identifying malware family and can be combined with other features for fast and scalable malware classification. Moskovitch et al. (2008) studied many classifier viz. NB, BNB, SVM, BDT, DT and ANN by byte-sequence n-grams (3, 4, 5 or 6) and find that NB classifier detect the malware with 84.53% accuracy [19]. In 2009 S. Momina Tabish [20] used 13 different statistical features computed on 1, 2, 3 and 4 gram by analyzing byte-level file content for classification of malware. In 2009 Syed Bilal Mehdi et al. [21] obtained 64% and 58% accuracy by NB classifier with good evaluator feature selection scheme on 4 gram and 6 gram features from the executables. Chandrasekar Ravi et al. in year 2012 reported 48.69% accuracy with NB classifier by using 4-grams Windows API calls features [22]. Chatchai Liangboonprakong et al. (2013) proposed a classification of malware families based on N-grams sequential pattern features [23]. They used DT, ANN and SVM classifier and obtained good accuracy. In 2013 Santos et al. [24] used Term Frequency for modelling different classifiers and among the studied classifier, SVM outperform for one opcode and two opcode sequence length respectively. Recently (2014) Zahra Salehi et al. construct feature set by extracting API calls used in the executable for the classification of malware [25] .

In addition to the traditional work on increasing malware classification accuracy alone researchers has now also identified the need to improve the execution time of such algorithms. In 2012 Ciprian Pungila and Viorel Negru [26] has proposed a highly-efficient memory compression model for implementing GPU-accelerated virus signature matching. In their work [they were able to achieve 22 less memory utilization and 38 times higher bandwidth compared to then existing single core implementation. Che-Lun Hung, Hsiao-Hsi Wang (2014) [27] has proposed a GPU based botnet detection technique in their work. They implemented the network traffic reduction stage on GPU and were able to achieve 8x times performance over CPU based traffic reduction. Manel Abdellatif et al. [28] (2015) has designed and implemented a host-based parallel anti-malware based on mobile GPU for Android devices. Their implementation was three times faster than the serial implementation on CPU. In 2016, Igor Korkin, Iwan Nesterow has proposed a technique for ”Acceleration of Statistical Detection of Zero-day Malware in the Memory Dump Using CUDA-enabled GPU Hardware” [29]. In their work they used GPU mainly for achieving speedup in memory forensic. task. Recently in 2017, Radu velea and Stefan Dragan [30] has proposed a CPU/GPU based hybrid approach to accelerate pattern matching for malware signatures. In their work they found that the hybrid approach is twice as fast as CPU only approach and consumes 25% less power.

3 Data Preprocessing and Feature Selection

3.1 Dataset

For this experiment we downloaded 11355 malware samples from malacia-project and collected 2967 benign programs (also verified from virustotal .com) from different systems. In the collected dataset it was found that majority of malware are below 500 KB hence for our study we focused on malware and benign files below 500 KB. After applying the size limit of 500 KB on samples we are left with 2363 benign samples and 11305 malware samples for our work.

3.2 Preprocessing

The malware and benign samples were processed using objdump utility to get opcodes. A unique opcode list was prepared from the processed samples which was then used to make the feature matrix for each malware or benign sample. For each sample the feature matrix consists of the frequency of a particular opcode in a malware and benign sample. The data was normalized by dividing each malware and benign opcode frequency by corresponding maximum opcode frequency.

3.3 Feature Selection

To find distinctive features in a group we first divide the normalized frequencies of malware and benign per group by the total number of malware and benign in that group and then perform column wise sum for that group where each column denotes an opcode. By performing the above procedure, we get two different vectors corresponding to malware and benign. Now we find absolute difference between the malware and benign vector where each column entry corresponds to an opcode. The absolute difference is then sorted in descending order. Finally, top K features (opcodes) are chosen based on the top K absolute difference. The whole process is done to find those opcodes which are able to separate malware and benign clearly in group. This process is then repeated for each group. It was observed that some of the groups were not having sufficient malware or benign samples for training and testing so we neglected them from our course of study (group 5, 8, 61, 65, 97 contain less than 6 samples in either malware or benign classes while group 98 and 100 has 0 malware samples). However, with more malware and benign samples in hand it they can be included. The pre-processed data is the divided into testing and training sets. We used 67% data for training and reaming 33% data for testing.

4 Naive Bayes Classifier

Naive Bayes is a linear classifier based on generative model. It follows the Byes theorem and assumes strong class independence between different attributes under consideration. Given set of an attributes (Opcodes in our case), A =a_1, a_2, a_3,..., a_n, the Naive Bayes model computes posterior probability for target class C(malware/benign) and can be represented as follows.

$$P(C|a_1, a_2, a_3, ..., a_n) = \frac{P(a_1,a_2,a_3,...,a_n|C)}{P(a_1,a_2,a_3,...,a_n)}$$

where, $P(C|a_1, a_2, a_3, ..., a_n)$ is the posterior probability of an executable sample of belonging to class C. Due to the class conditional independence assumption of Naive Bayes the likelihood can be represented as follows.

$$P(a_1, a_2, a_3, ..., a_n|C) = \prod_{i=1}^{n} P(a_i|C)$$

Here $P(a_i|C)$ is the likelihood of attribute a_i, and can be computed with the help of Gaussian Naive Bayes as follows.

$$P(a_i|C) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(a-\mu_C)^2}{2\sigma_C^2}}$$

where a denotes the attribute a_i (opcode of the executable under test) , μ_C and σ_C represent the mean and variance of class C respectively.

By using above equations we can calculate the posterior probabilities for the test executable. If the malware class probability is higher then it classified as malware otherwise it is labelled as benign.

5 Proposed approach and Results

5.1 Hardware configuration

The study is performed using two different hardware configurations. The preliminary tests were performed on Intel Pentium processor with 3.0 Ghz clock speed and 32 GB RAM. The parallel algorithm was tested using Nvidia Quadro K2000, a Kepler architecture based graphic processor unit consisting of 384 CUDA cores divided into set of 2 streaming processors of 192 cores each and GPU clock of 954MHz. It also has 64KB of on chip configurable memory which is divided into shared memory and L1 cache which is equivalent to CPU cache memory and can be programmed as per the programmers use which is a unique feature associated with this GPU architecture. The available configurations for this series are 32KB shares and 32 KB L1 cache, 48 KB shared and 16 KB L1 cache, 16 KB shared and 48 KB L1 cache. In addition to it the Keplar GPU has a DRAM of 2 GB.

The second experiment was performed on a system with Intel i7-7700HQ Quad core processor with base frequency of 2.8Ghz, 8 GB RAM, Pascal architecture (GP107) based Nvidia 1050Ti GPU with 768 CUDA cores distributed across 6 SMP and 4GB GPU DRAM operating on a base speed of 1291 Mhz.

5.2 Proposed approach

In our work first, we divided the dataset containing both malware and benign opcode frequency into 100 groups of 5 KB sizes as discussed in [10] for improving the malware detection. This step was followed by per group feature selection and training of Naive Bayes classifier. All these steps were performed on CPU and we did not took readings for their parallel implementation due to mainly two reasons. Firstly, our main aim was to improve the detection process so we did not focus on model construction process as it is a one-time process. Secondly, there are few memory limitations applied by the GPUs.

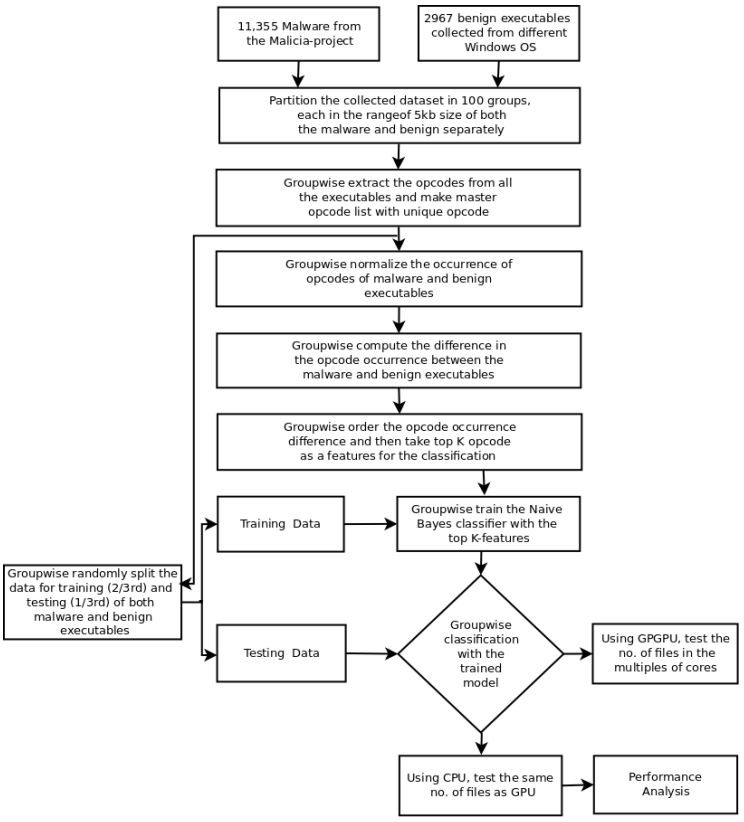

Fig. 1. Flow chart for malware detection system

Training is performed on CPU to generate the trained model by exploiting the term independence principle of Naive Bayes and malware mean, malware variance, benign mean, benign variance per group are calculated. After the completion of training we are left with the trained model and a Feature matrix representing the top K features from each group where K is given by user at the time of training. These matrices are dumped to files after the completion of training and used again testing step.

During testing step the trained model and feature matrix are loaded in memory and represented in the form of matrices and the testing data set is also read

simultaneously. The test dataset is again divided into 5KB size similar to training step [10]. Finally, all the data is loaded in GPU DRAM and Naive Bayes algorithm is applied to calculate probabilities of testing samples for assigning class label to it. The testing sample is only tested against the group corresponding to its size. This is done in order to increase the accuracy as discussed by authors in their previous findings [10]. In this whole process we used a task parallel approach for parallel implementation of detection algorithm on GPU.

5.3 Results

It was observed that the parallel implementation on GPU of Naive Bayes algorithm was able to achieve an execution time performance of 45x to 52x in case of first configuration and an execution time performance boost of 100x to 125x over sequential implementation on CPU with second configuration. It was also observed that the performance gain is dependent on no of features (K) taken and the no of cores in GPU. We observed that initially the CPU execution time dominates the GPU execution time in terms of execution time performance but as soon as the no of files increases and reach a threshold the GPU dominates the CPU in terms of performance. This was because of underutilization of GPU cores due to lack of task to perform as it processes tasks in blocks and some cores were idle while CPU executes tasks one by one basis.

In our experiment we found that the dataset size was not sufficient to observe the GPU execution time as after taking out training data we were left with 4500 samples only for testing. So, for measuring the relation between CPU and GPU execution time we considered whole dataset. Although accuracy was calculated using a split of 67% for training and 33% for testing. We also observed that there is always a trade-of between accuracy and execution time. The accuracy significantly decreases when the no of features (K) reduces below 90 in our dataset and after increasing K beyond 200 the accuracy does not change, best case being above 87%. Although for smaller data set the shared memory can also be utilized.

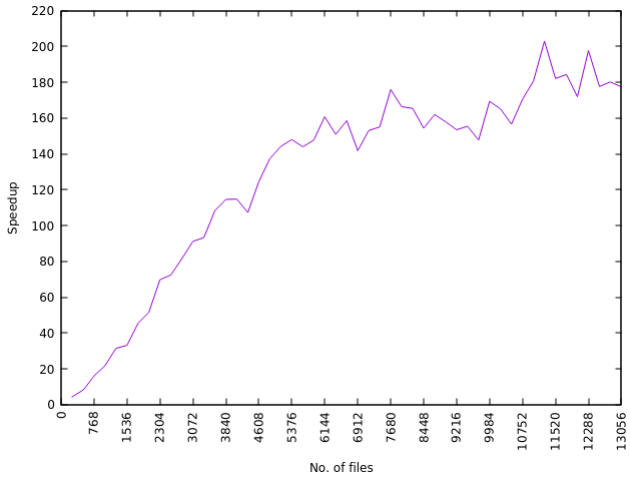

Fig. 2. CPU v/s GPU execution time comparison using configuration 1 using all files

It was also observed that the speedup is also dependent on he number of features used for malware detection. The speedup decreases with increasing number of features for given number of cores. In our experiment we experimented with various sets of features and found that the accuracy increases with increasing the value of K (top K features) till 200, after which there is no significant change in accuracy. This situation creates a new challenge of coming up with a new and efficient feature selection technique so that the time and accuracy trade-of can be minimized.

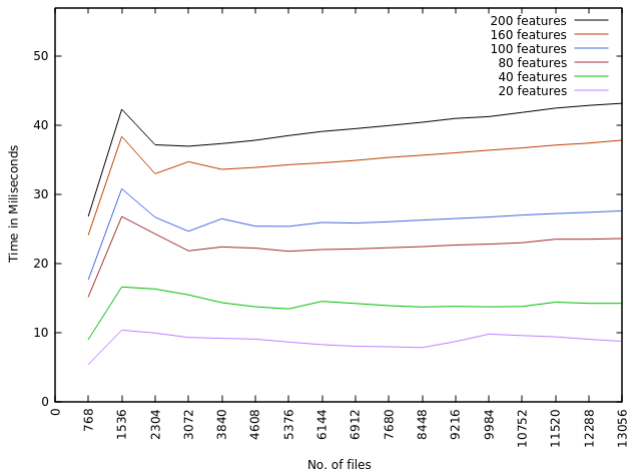

Fig. 3. CPU execution time with different number of features with configuration 2

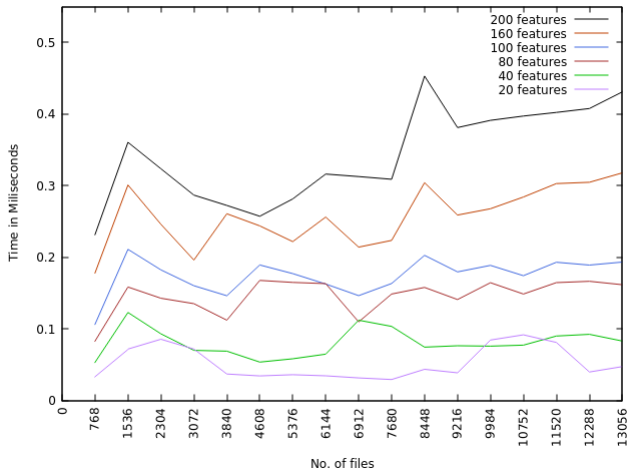

Fig. 4. GPU execution time with different number of features with configuration 2

6 Conclusion

In this work we implement parallel implementation of Naive Bayes algorithm for fast detection of malware on GPU along with size-based partitioning of malware samples for better accuracy. In our approach we discuss a task parallel approach for parallelizing Naive Bayes classifier. We experimentally evaluated the proposed approach against the sequential version of algorithm and got a speedup of 45x to 52x with first configuration and 100x to 125x with second configuration maintaining same accuracy. As our future work we would like to explore suitability of other parallel implementation classification algorithms and compare with our approach.

References

[1] J. N. P. Mell, K. Kent, "Guide to malware incident prevention and handling," *National Institute of Standards and Technology*, 2005.

[2] Govindaraju, "Exhaustive statistical analysis for detection of metamorphic malware," Master's thesis, San Jose State University, 2010.

[3] A. Sharma and S. K. Sahay, "Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey," *International Journal of Computer Applications*, vol. 90, pp. 7–11, March 2014.

[4] "McAfee Labs Threats Report," tech. rep., McAfee, December 2017.

[5] "Internet Security Threat Report (ISTR)," tech. rep., Symantec Corporation, April 2017 (Date last accessed 31-May-2018).

[6] "Internet Security Threat Report (ISTR)," tech. rep., Symantec Corporation, April 2018 (Date last accessed 31-May-2018).

[7] R. Stone, "A call to cyber arms," *Science*, vol. 339, no. 6123, pp. 1026–1027, 2013.

[8] T. C. Robert M. Lee, Michael J. Assante, "Analysis of the Cyber Attack on the Ukrainian Power Grid," tech. rep., E-ISAC group SANS, 2016.

[9] K. Allix, T. F. Bissyandé, Q. Jérome, J. Klein, R. State, and Y. Le Traon, "Large-scale machine learning-based malware detection: Confronting the "10-fold cross validation" scheme with reality," in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, CODASPY '14, (New York, NY, USA), pp. 163–166, ACM, 2014.

[10] J. Canto, M. Dacier, E. Kirda, and C. Leita, "Large

scale malware collection : lessons learned," in *SRDS 2008, 27th International Symposium on Reliable Distributed Systems, October 6-8, 2008, Napoli, Italy*, (Napoli, ITALY), 10 2008.

[11] A. Sharma and S. K. Sahay, "Improving the detection accuracy of unknown malware by partitioning the executables in groups," in *Proceedings of the 9th international confrence on advanced computing and comunication technologies*, 2015.

[12] D. Ucci, L. Aniello, and R. Baldoni, "Survey on the usage of machine learning techniques for malware analysis," *CoRR*, vol. abs/1710.08189, 2017.

[13] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, (Washington, DC, USA), pp. 38–, IEEE Computer Society, 2001.

[14] K. M.E., W. A., and L. A. et al., "Malware phylogeny generation using permutations of code. journal in computer virology," *Journal in Computer Virology*, vol. 1, pp. 12–23, 2005.

[15] O. Henchiri and N. Japkowicz, "A feature selection and evaluation scheme for computer virus detection," *Sixth International Conference on Data Mining (ICDM'06)*, pp. 891–895, 2006.

[16] D. Bilar, "Opcodes as predictor for malware," *Int. J. Electron. Secur. Digit. Forensic*, vol. 1, pp. 156–168, Jan. 2007.

[17] Y. Ye, D. Y. Dingding Wang, Tao Li, and Q. Jiang, "An intelligent pe-malware detection system based on association mining. journal in computer virology," *Journal in Computer Virology*, vol. 4, pp. 322–324, 2008.

[18] S. C. Tian, R.Batten nad L. M.Versteeg, "Large scale malware collection : lessons learned," in *Proceedings of the 3rd International Conference on Malicious and Unwanted Software : MALWARE 2008*, (Alexandria, Va.), 10 2008.

[19] M. R., F. C., T. N., E. Berger, M. Gitelman, S. Dolev, and Y. Elovici, "Unknown malcode detection using opcode representation," *Intelligence and Security Informatics*, pp. 204–215, 2008.

[20] S. M. Tabish, M. Z. Shafiq, and M. Farooq, "Malware detection using statistical analysis of byte-level file content," in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, CSI-KDD '09, (New York, NY, USA), pp. 23–31, ACM, 2009.

[21] S. B. Mehdi, A. K. Tanwani, and M. Farooq, "Imad: In-execution malware analysis and detection," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, (New York, NY, USA), pp. 1553–1560, ACM, 2009.

[22] C. Ravi and R. Manoharan, "Malware detection using windows api sequence and machine learning," *International Journal of Computer Applications*, vol. 43, no. 17, pp. 12–16, 2012.

[23] C. Liangboonprakong and O. Sornil, "Classification of malware families based on n-grams sequential pattern features," *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 777–782, 2013.

[24] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013.

[25] M. G. Zahra Salehi, Ashkan Sami, "Using feature generation from api calls for malware detection," *Computer Fraud and Security, ELSEVIER*, vol. 2014,

pp. 9–18, September 2014.

[26] C. Pungila and V. Negru, "A highly-efficient memory-compression approach for gpu-accelerated virus signature matching," *International Conference on Information Security (ISC 2012)*, pp. 354–369, 2012.

[27] C.-L. Hung and H.-H. Wang, "Parallel botnet detection system by using gpu," *2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS)*, pp. 65–70, 2014.

[28] C. T. Manel Abdellatif, A. Hamou-Lhadj, and M. Dagenais, "On the use of mobile gpu for accelerating malware detection using trace analysis," in *2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW)*, (Montreal, QC, Canada), pp. 38–, 2016.

[29] I. Korkin and I. Nesterow, "Acceleration of statistical detection of zero-day malware in the memory dump using cuda-enabled GPU hardware," *CoRR*, vol. abs/1606.04662, 2016.

[30] R. Velea and x015Etefan Drx0103gan, "Cpu/gpu hybrid detection for malware signatures," *2017 International Conference on Computer and Applications (ICCA)*, pp. 85–89, 2017.