Graphs for Malware Detection: The Next Frontier

Abhishek B. Sharma° B. Aditya Prakash*
°Fidelis Cybersecurity
*Department of Computer Science, Virginia Tech
abhishek.sharma@fidelissecurity.com, badityap@cs.vt.edu

ABSTRACT

Machine Learning based approaches for malware detection have achieved a certain level of maturity as product offerings by Cybersecurity companies. VirusTotal recently included X by Invincea [21] and MalwareScore by Endgame as *signature-less* anti-virus scanners. In this position paper, we argue that the significant challenges related to information heterogeneity, noisy and uncertain inputs, and the demand (from cybersecurity professionals) to generalize beyond per-sample prediction are impeding further advancement in this field. These challenges cannot be addressed by standard supervised machine learning approaches. We highlight the fact that the current applications of machine learning for cybersecurity have focused on *feature based learning* and largely ignored *identifying and learning from the underlying relationships* between malware samples. **Malware graphs** are the obvious abstractions for representing such relationships. We briefly discuss potential approaches and outline further research that is needed to build high-impact deployable cybersecurity solutions based on malware graphs.

1 INTRODUCTION

Machine Learning for Cybersecurity is an active area of research and innovation within academia as well as the industry. Recently, the cybersecurity industry has started offering malware detection solutions based on supervised learning algorithms like Random Forests [1], Support Vector Machines, and Deep Neural Networks [2] [21]. However, we believe that the era of plucking low hanging fruits is over.

Current approaches for malware detection require three important components: (1) large labeled datasets that contain a variety of malware samples, (2) feature engineering to convert information contained in file properties, raw binary, and static analysis into representations that can be used by machine learning algorithms, and (3) use of either supervised or unsupervised learning algorithms [2, 5, 7, 12, 21].

In this paper, we first highlight the challenges related to these three components that are impeding further advancement in this area. We propose using *graphs that capture relationships between different malware samples* with semi-supervised learning to address these challenges. However, building malware graphs involves its own challenges arising from both information heterogeneity as well as the need to *explain* an algorithm's decision to security analysts. We outline these challenges and discuss promising ideas to address them.

Challenge 1: Incomplete and Noisy labels. Security practitioners rely on public and propriety malware repositories to collect obtain samples to train machine learning models [3] [22]. However, all such efforts suffer from lack to reliable labels. We have datasets where only a subset of samples are labeled (either as malware or benignware). Furthermore, these labels can be incorrect, either a malware sample is incorrectly labeled as benignware (or vice-versa) or a malware sample is identified as belonging to an incorrect class (e.g. a highly dangerous remote access trojan used to take over a computer system incorrectly labeled as the least dangerous Adware that simply displays advertisements on a computer). To address this issue in the context of large datasets, we currently rely on services like VirusTotal that allow you to submit samples for analysis by multiple antivirus engines. A typical *rule-of-thumb* has been to label any sample with no antivirus engine hits as benignware and samples with 30% or more antivirus engine hits as malware [22]. However, this approach has four major issues: (1) It is no guaranteed to eliminated all incorrect labels, (2) It is not consistent because antivirus engines constantly update their signatures, (3) We are forced to throw away potentially useful samples, i.e. the ones with less than 30% antivirus engine hits, and (4) It is not sufficient in a multi-class setting, i.e. we want to distinguish between different malware types (in addition to malware vs. benignware) because different antivirus engines are known to disagree on the type of a malware [15].

Challenge 2: Information heterogeneity. The cybersecurity community has built several tools for both static and dynamic analysis of malware samples. Today, we can generate multiple *views* of a sample by extracting information from raw sample binary, static analysis (referenced libraries and functions , PE (Portable Execution format) properties, digital signatures, etc.), and dynamic or execution based analysis (file system accesses, network behavior, API calls, processes created, etc.). As malware reverse engineers and threat researchers improve at their craft, these analysis techniques will provide more and more data. However, this creates a data heterogeneity as well as curse of dimensionality problem for machine learning based approaches. Recent work has used approaches like *hashing trick*, *random projections*, and *multiple kernel learning* [2, 22], to address this challenge. However, these approaches sacrifice *explainability* by creating features that security analysts cannot interpret.

Challenge 3: Generalize beyond per-sample detections. A vanilla malware classifier simply flags a sample as malware or benignware with a certain score. However, often a malware sample is a variant of a known type (produced by using obfuscation techniques to evade antivirus engines) [27]. Hence, we need to get better at discovering underlying relationships between different

[1]https://www.endgame.com/blog/world-meet-malwarescore
[2]http://blog.virustotal.com/2016/08/virustotal-invincea.html
[3]www.virustotal.com

malware samples and move beyond predicting individual samples as malware or benignware. This will enable us to identify interesting/anomalous samples to prioritize the efforts to malware reverse engineers (a precious resource at any organization and the community in general).

Challenge 4: Track evolution of malware. We know that malware classifiers are effective at detecting variants of known malicious platforms and toolkits [22]. However, Saxe et al. show that the performance of a classifier can degrade significantly over time by performing a *time split experiment*. They trained a Deep Neural Network classifier using samples with compilation timestamp before July 31st, 2014 in their dataset and used it to classify samples compiled after July 31st, 2014. It detect 67.7% of malware at 0.1% false positive rate. In contrast, the same algorithm when trained with samples with compilation time similar to the samples in their test dataset detected 95.2% of malware at 0.1%. This is not surprising because malware evolves and hence, we need to keep the classifiers up-to-date by re-training it on new samples. However, this fact underscores the importance of tracking malware genealogy; something that our current focus on building malware classifiers ignores.

2 GRAPH MINING TO OUR RESCUE

Our position is that current applications of machine learning for cybersecurity have focused on *feature based learning* and largely ignored *identifying and learning from the underlying relationships* between malware samples. **Malware graphs** are the obvious abstractions for representing such relationships.

Graph based approaches for detecting malware is an active area of research for malware targeted at PCs [3, 4, 8, 9, 14, 18–20] as well as mobiles [6, 17, 26]. A popular approach is to build *Program Representation Graphs* [17], i.e. graphs that capture semantic representations derived from *static analysis*, e.g. function call graphs, control and data flow graphs, and data and program dependency graphs [3, 6, 8, 11, 18, 26].

Polonium [4] uses a *reputation-based approach* by constructing a bi-partite graph between files and the users' computers on which they are found. Kwon et al. [14] propose a *downloader-graph abstraction* to capture the download activity on end hosts, and exploit the differences between the growth patterns of benignware downloader-graphs and malware downloader-graphs to detect the *downloader trojans* or *droppers* malware. However, these approaches build graphs using only one *view* of malware. In contrast, the feature-based approaches discussed in Section 1 exploit *multiple views* by combining information from static as well as dynamic analysis.

There are benefits to capturing multiple views or aspects of a malware using graphs [16]. Our claim is based on how entities behave and interact in many real-world applications. E.g., humans are associated with others based on different types of relationships that can only be inferred by combining information from multiple sources. Even in our specialized roles as researchers, we are part of multiple types of relationships, e.g., co-authors on a paper, publishing at the same/similar conferences or journals, citing the same/similar papers. The similarity between web pages based on hyperlinks or words occurring on a page is another example. Researchers have used multiple graphs [24], heterogeneous networks [23], and multi-view spectral clustering [28] to study such networks.

Multiple kernel learning offers an interesting possibility [2] for building networks to capture multiple malware views. Anderson et al. define a similarity metric on each distinct view of a malware using kernels. They use multiple kernel learning to find a weighted combination of the different kernel which yields the best classification accuracy in a support vector machine classifier. Another approach can be based on metric learning to automatically learn the importance of each view for the problem [13]. Multiple kernels can also be used for semi-supervised learning [25], which might be more suited for malware detection due to the problem of incomplete and noisy labels (see Section 1).

An alternate approach can use graphs from different views of malware to detect different malware families (instead of fusing them together as done by multiple kernel learning based approaches). Kwon et al. [14] have shown that network activity graphs are effective at capturing the behavior of *trojans* or *droppers* malware whereas file system activity graph can be effective at detecting ransomware [10]. More research is needed to understand the relationship between different malware views and the frameworks and toolkits used by different malware families.

Graphs can help in tracking malware genealogy as well. Genealogies are naturally represented as relations which evolve with time. Hence temporal network analysis can help in pinpointing different malware families and how they evolve. Dynamic graph mining is a growing field of research [1], and we believe this can generate interesting novel problems.

Graph mining also raises various interesting computational issues. Storing the entire graph centrally sometimes may not be an option for real-time detection. Hence in such cases, we may want to do the computation in a decentralized fashion among the user facing sensors. This is a challenging problem, as sensors typically have severe resource constrains. We envisage this to be fruitful area of research, which can aid in deployment in different settings.

Where are the datasets? We know of three malware datasets that are publicly available: WINE from Symantec[4], Sherlock-a mobile malware dataset[5], and the Microsoft Malware Classification Challenge dataset hosted by Kaggle[6]. Additionally, one can harvest samples from online resources shared by the cyber security professionals[7]. Representative datasets on real-world malware is important for engaging researchers. We should continue to explore opportunities for creating more such datasets.

3 CONCLUSIONS

To conclude, we have discussed new challenges in data-driven malware detection, which can not be handled using standard approaches used so far. Our position is that sophisticated approaches leveraging graph mining can play an important role in addressing such

[4]http://securityresponse.symantec.com/about/profile/universityresearch/sharing.jsp
[5]http://bigdata.ise.bgu.ac.il/sherlock/
[6]https://www.kaggle.com/c/malware-classification
[7]http://thezoo.morirt.com/
https://zeltser.com/malware-sample-sources/

challenges. In summary, graphs can help in dealing with information heterogeneity, noisy and uncertain inputs, help generalize detection and track malware families instead of just samples. Much work needs to be done, but addressing these crucial issues are necessary to craft high-impact deployable security solutions.

ACKNOWLEDGEMENTS

This paper is based on work partially supported by the NSF (IIS-1353346), the NEH (HG-229283- 15), ORNL (Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift.

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Computing Survey* (2014).

[2] Blake Anderson, Curtie Storlie, and Terran Lane. 2012. Improving Malware Classification: Bridging the Static/Dynamic Gap. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security.*

[3] Blake H. Anderson, Daniel A. Quist, Joshua C. Neil, Curtis B. Storlie, Terran Lane, and Michael E. Fisk. 2011. Graph-based Malware Detection Using Dynamic Analysis. *Journal in Computer Virology* 7-4 (2011), 247–258.

[4] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2010. Polonium: Tera-Scale Graph Mining for Malware Detection. In *Proceedings of the KDD-LDMTA.*

[5] O. E. David and N. S. Netanyahu. 2015. DeepSign: Deep learning for automatic malware signature generation and classification. In *Proceedings of the International Joint Conference on Neural Networks.*

[6] Hugo Gascon, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. 2013. Structural detection of android malware using embedded call graphs. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security.*

[7] Daniel Gibert. 2016. *Convolutional Neural Networks for Malware Classification.* Master's thesis. Universitat Politecnica de Catalunya.

[8] Xin Hu, Tzi cker Chiueh, and Kang G. Shin. 2009. Large-Scale Malware Indexing Using Function-Call Graphs. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS).*

[9] Chanhyun Kang, Noseong Park, B. Aditya Prakash, Edoardo Serra, and V. S. Subrahmanian. 2016. Ensemble Models for Data-driven Prediction of Malware Infections. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16).* ACM, New York, NY, USA, 583–592. DOI: http://dx.doi.org/10.1145/2835776.2835834

[10] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2010. . In *Proceedings of the International Confernece on Broadband, Wireless Computing, Communication And Applications.*

[11] Joris Kinable and Orestis Kostakis. 2011. Malware classification based on call graph clustering. *Journal in Computer Virology* 7(4) (2011).

[12] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. 2016. Deep Learning for Classification of Malware System Call Sequences. In *Proceedings of the Australian Joint Conference on Artificial Intelligence.*

[13] Brian Kulis. 2013. Metric Learning: A Survey. *Foundations and Trends in Machine Learning* 5, 4 (2013), 287–364. DOI: http://dx.doi.org/10.1561/2200000019

[14] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitras. 2015. The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS).*

[15] Federico Maggi, Andrea Bellini, Guido Salvaneschi, and Stefano Zanero. 2011. Finding non-trivial malware naming inconsistencies. In *Proceedings of the International Conference on Information Systems Security.*

[16] Andreas Moser, Christopher Kruegel, and Engin Kirda. 2007. Limits of Static Analysis for Malware Detection . In *Proceedings of the IEEE Annual Computer Security Applications Conference (ACSAC).*

[17] Annamalai Narayanan, Guozhu Meng, Yang Liu, Jinliang Liu, and Lihui Chen. 2016. Contextual Weisfeiler-Lehman Graph Kernel For Malware Detection. *CoRR* abs/1606.06369 (2016). http://arxiv.org/abs/1606.06369

[18] Younghee Park and Douglas Reeves. 2011. Deriving Common Malware Behavior through Graph Clustering. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS).*

[19] B. Aditya Prakash. 2015. *Graph Mining for Cyber Security.* Springer International Publishing, Cham, 287–306. DOI: http://dx.doi.org/10.1007/978-3-319-14039-1_14

[20] B. Aditya Prakash. 2016. Prediction Using Propagation: From Flu Trends to Cybersecurity. *IEEE Intelligent Systems* 31, 1 (Jan 2016), 84–88. DOI: http://dx.doi.org/10.1109/MIS.2016.1

[21] Joshua Saxe. 2015. Why Security Data Science Matters and How It's Different: Pitfalls and Promises of Data Science Breach Detection and Threat Intelligence. In *BlackHat.*

[22] Joshua Saxe and Konstantin Berlin. 2015. Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features. *CoRR* abs/1508.03096 (2015). http://arxiv.org/abs/1508.03096

[23] Yizhou Sun, Rick Barber, Manish Gupta, Charu C. Aggarwal, and Jiawei Han. 2011. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM).*

[24] Wei Tang, Zhengdong Lu, and Inderjit S. Dhillon. 2009. Clustering with Multiple Graphs. In *Proceedings of the International Conference on Data Mining.*

[25] Tao Yang and Dongmei Fu. 2014. Semi-supervised classification with Laplacian multiple kernel learning. *Neurocomputing* 140 (2014), 19–26.

[26] Wei Yang, Xusheng Xiao, Benjamin Andow, Sihan Li, Tao Xie, and William Enck. 2015. AppContext: Differentiating Malicious and Benign Mobile App Behaviors Using Context. In *Proceedings of the IEEE International Conference on Software Engineering.*

[27] Ilsun You and Kangbin Yim. 2010. Malware Obfuscation Techniques: A Brief Survey. In *Proceedings of the International Confernece on Broadband, Wireless Computing, Communication And Applications.*

[28] D. Zhou and C. J. C. Burges. 2007. Spectral clustering and transductive learning with multiple views. In *Proceedings of the International Conference on Machine Learning (ICML).*