To believe or not to believe: Validating explanation fidelity for dynamic malware analysis

Li Chen
li.chen@intel.com
Intel Labs

Carter Yagemann
yagemann@gatech.edu
Georgia Institute of Technology

Evan Downing
edowning3@gatech.edu
Georgia Institute of Technology

Abstract

Converting malware into images followed by vision-based deep learning algorithms has shown superior threat detection efficacy compared with classical machine learning algorithms. When malware are visualized as images, visual-based interpretation schemes can also be applied to extract insights of why individual samples are classified as malicious. In this work, via two case studies of dynamic malware classification, we extend the local interpretable model-agnostic explanation algorithm to explain image-based dynamic malware classification and examine its interpretation fidelity. For both case studies, we first train deep learning models via transfer learning on malware images, demonstrate high classification effectiveness, apply an explanation method on the images, and correlate the results back to the samples to validate whether the algorithmic insights are consistent with security domain expertise. In our first case study, the interpretation framework identifies indirect calls that uniquely characterize the underlying exploit behavior of a malware family. In our second case study, the interpretation framework extracts insightful information such as cryptography-related APIs when applied on images created from API existence, but generate ambiguous interpretation on images created from API sequences and frequencies. Our findings indicate that current image-based interpretation techniques are promising for explaining vision-based malware classification. We continue to develop image-based interpretation schemes specifically for security applications.

1. Introduction

Malware is malicious software created for harming users, computers, and networks. Viruses, trojan horses, worms, spyware, and ransomware are examples of malware. In malware detection, static analysis without executing the application is a quick method to detect malicious patterns in an application. To avoid static detection most malware contain obfuscated code. Dynamic analysis, on the other hand, executes the code and records the malware runtime behavior. Even though dynamic analysis is slower than static analysis, it offers better resiliency and efficacy against malware code obfuscation.

Machine learning has become increasingly popular and important for malware detection because it can generalize to detect new malware families. The manual effort of feature engineering can be costly, especially on unstructured data formats. As the volume of data continues to grow at increasing speed, scalable algorithms for malware detection are in high demand. Computer vision has provided a unique perspective for performing malware classification. First, it enables natural visualization on malware as a whole entity. Second, deep learning has demonstrated state-of-the-art performance for image classification. When malware is represented as images, transfer learning can leverage the superior performance from vision to classify malware with accelerated training speed and maintained classification efficacy. Last but not least, it has superior performance compared with classical machine learning algorithms [16, 15, 22, 7, 6].

For static malware analysis, a binary can be directly mapped to pixel values between 0 and 255 [16, 15, 22, 7, 6]. By visually inspecting binaries plotted as grey-scale images, we can observe the textural and structural similarities or dissimilarities on the *static* features of malware. By contrast, there are fewer vision-based *dynamic* malware classification techniques. [7] proposed a hierarchical ensemble neural network scheme on dynamic telemetries collected from Intel® Processor Trace, where the control flow packets are converted into time series of images and demonstrated the superior performance compared with other popular dynamic malware classifiers.

For security applications, besides classification efficacy, model explanation is equally important for security researchers and practitioners to deploy the model in the wild. Sensible interpretation from the model on why a sample is predicted as malicious or benign can generate valuable insights to triage malware families, identify new malware sig-

natures, understand the evolution of polymorphic malware, and enhance the practitioner's trust in the model. When malware is represented as images, interpretation schemes for natural images [17] can be extended to explain malware classification.

Unlike natural images, where interpretation fidelity can be assessed via human eyes, interpretation fidelity on malware images remains to be validated through security domain expertise. In this paper, via two case studies for dynamic malware classification, we investigate the effectiveness of local-interpretable model-agnostic explanation (LIME) framework [17] specifically for image-based dynamic malware analysis. Our first case study examines dynamic malware images generated from predictions on sequences of instructions. The interpretation framework identifies indirect calls that uniquely characterize the underlying exploit behavior of a malware family. In our second case study, we consider three types of malware images generated from API existence, API sequence, and API frequency features. The interpretation framework provides insightful information such as crypto-related APIs when applied on images created from API existence, but generates ambiguous information on images created from API sequences and frequencies. Our findings indicate that current image-based interpretation techniques are promising for vision-based malware classification. We plan to develop image-based interpretation schemes specifically for malware images in security applications.

Our contributions are summarized as follows:

- To the best of the authors' knowledge, we are the first to validate the interpretation fidelity of a model-agnostic interpretation framework, using security domain expertise, on dynamic image-based malware classification.
- We use deep transfer learning on dynamic malware images generated from instruction sequence predictions, API existence, API sequence, and API frequency features and demonstrate that dynamic malware image analysis is highly effective.
- Our case studies present a valuable combination of machine learning and domain expertise to fully understand the effectiveness of malware classification algorithms.
- We advocate that interpretation is another important dimension to evaluate malware classifiers. Vision-based interpretability highlights the advantage of approaching the malware problem from a computer vision direction so that interpretation becomes concrete as to indicate the actual locations of potential malicious signals.

2. Background and Related Work

In the interpretation frameworks for image classification, the explanation method provides interpretation by identifying the most contributing pixel regions to the prediction result [9, 10, 14, 17]. While there is an abundance of vision and natural language based interpretation frameworks, few exist specifically for security applications. In [11], the authors proposed non-linear approximation on the local decision boundary to explain malware detection algorithms for security applications. The method is primarily for multi-layer perception (MLP) and recurrent neural networks (RNN) on non-image based data representations for malware classification. [6] employed interpretation frameworks such as the local-interpretable model-agnostic explanation (LIME) [17] for natural images on static malware images.

When we represent dynamic malware as images, natural image explanation schemes can be applied. In two case studies here, we extend LIME to image-based dynamic malware classification and thoroughly examine the interpretation fidelity using security domain knowledge.

3. Case Studies

3.1. Case Study I

Our first case study is concerned with detecting anomalies in dynamic program control-flow traces. The task is to examine whether PDF files opened by Adobe Acrobat Reader are malicious or not in a Windows® system. The source of our control-flow data is Intel® Processor Trace (Intel® PT), which is a hardware feature present in modern Intel® processors [7]. Intel® PT produces a large volume of data within a short time period. For example, tracing Acrobat Reader for one minute yields over *2 million* indirect control-flow transfers including returns and indirect calls and jumps. It becomes a daunting task for human analysts to examine such a high volume of transfers for signs of exploitation. Hence it is desired to employ the automated interpretation framework to extract an explanation.

We collect 1,249 benign and 1,314 malicious traces from the *pdfka* malware family, where each trace is collected from the targeted program opening a PDF document. Each trace is then disassembled, yielding a linear sequence of the executed basic blocks. A basic block is defined as a sequence of linear instructions ending with a branch, which can be a return, call, jump, conditional branch and so on. Each basic block is assigned a universally unique integer defined as BBID. A fixed length sliding window is moved over the sequence of BBIDs, and a subsequent long short term memory (LSTM) neural network is tasked with learning and predicting the next BBID for any sub-sequence ending with an indirect control-flow transfer. The intuition behind predicting only indirect transfers is that these are the

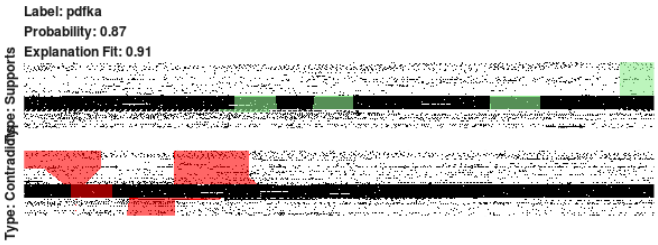

Figure 1. Interpretation for a *pdfka* sample. White pixels are correct predictions by the underlying model and black incorrect. Green denotes strong support whereas red is strong contradiction. The green regions shown highlight a suspicious control-flow loop.

only places where control-flow hijacking can occur during a program execution. The LSTM model is trained using only normal traces of the target program and then its performance is monitored over unlabeled traces. If an anomaly occurs in the trace, this will cause the model's performance to drop below a defined threshold and the trace will be labeled anomalous.

The dynamic malware images are generated from the prediction of the LSTM model on the BBIDs, where the white pixels are correct predictions and black pixels are incorrect predictions. On these malware images, we apply deep transfer learning using the pre-trained VGG model [20] on ImageNet, freeze the top layers and add an additional two fully connected layers, each with dropout, to retrain on the dynamic malware images. The training and test split is $0.8 : 0.2$. We set the number of epochs to be 50 with early stopping criterion if the validation loss does not decrease after 10 epochs. We use the model checkpointed at the 32-th epoch. The classification accuracy on the test set is 100%. This result demonstrates the effectiveness of vision-based deep transfer learning approach for dynamic malware detection and thus it makes sense to examine what interpretation can be generated using the decision boundaries from this model.

Next we apply the interpretation framework on the generated images to examine the fidelity of explanation. Figure 1 shows the interpretation of one of the *pdfka* family traces. Our model marks several spots within a large streak of incorrect predictions as strongly supporting that this trace belongs to the *pdfka* set. From here we can reverse these chunks of BBIDs to get back to the executed instructions. Upon manual inspection, we discover that at this point in the trace, the program makes one particular indirect call several hundred times in a row. By looking at the relative virtual address, we determine that this activity is happening inside the part of *AcroRd32.dll* that parses TIFF images. The most well-known vulnerability in this part of the program is CVE-2010-0188, which matches several online reports about this family's behavior [4]. We also manually confirm that this pattern does in fact appear in all the *pdfka* traces and none of the benign traces. To further verify, we also create and trace several benign documents containing benign TIFF images and confirm none of them produce the anomalous pattern. Although Acrobat Reader is closed source, making indisputable verification difficult, we believe our manual analysis strongly supports that our interpretation model successfully identified the subsection of the *pdfka* traces that uniquely characterizes the underlying exploit this family relies on. This case study demonstrates the usefulness of the interpretation method on dynamic malware images.

3.2. Case Study II

In our second case study, we evaluate three models designed to classify Windows^{R} Portable Executable (PE) files as either malicious or benign. All three models use dynamic features produced by malware and benign software during execution.

Our malware dataset is comprised of 13,394 Windows^{R} PE samples. These samples were collected from the Georgia Tech Research Institute (GTRI) using their internal malware collection and analysis platform APIARY [1]. Using AVClass [19], our malware dataset is made up of 247 families (demonstrating the diversity of our samples).

Our benign dataset is made up of 5,772 samples and was collected by crawling CNET [2]. Specifically, our samples are a mix of Windows^{R} PE and Windows^{R} Installer (MSI) files under 22 different categories (according to CNET) ranging from Audio to Education to Business-related software.

We ran all of our samples for 2 minutes using a modified version of Cuckoo [3] version 1.2 in Windows 7 32-bit KVM virtual machines with network and random-access memory (RAM) hardware extensions. We used KVM and hardware extensions to introduce as few artifacts indicative of a malware analysis environment as possible. Malware authors have been known to check for system and network-related artifacts (e.g., registry key values and network timing) which they can use to evade analysis (e.g., by performing innocuous activities or terminating early) [13, 12, 8, 18]. To improve the quality of our malware dataset, we only include samples which ran for the full 2 minutes without terminating early. We also executed the malware samples with 3 days of them being collected by the organization to improve the chances that the malware would perform malicious activities. Finally, to improve the quality of our benign dataset, we only include samples which did not have more than 2 antivirus companies label them as malicious via VirusTotal [5]. We also use Cuckoo to automatically interact with the benign software (via fake mouse-clicks on its GUI) to cause it to reveal a variety of behaviors (namely the installation process).

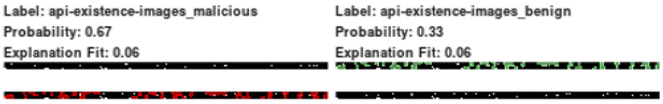

Figure 2. Interpretation for a dynamic malicious image created from API existence. The large light green areas in the top left image denote support regions for the malicious class. The supported regions that contribute to malicious prediction focused on cryptography-related API calls and HTTP-related calls.

On each set of the dynamic malware images generated from API existence, sequence, and frequency, we again apply transfer learning via VGG network pre-trained on ImageNet, where we freeze the top layers and add customized two additional fully connected layers and a softmax layer to produce the classification result. The training and validation split is 0.8:0.2. We set the number of epochs to be 50 with early stopping criterion if the validation loss does not decrease after 10 epochs. The classification results on three models are the same within statistically significance with an accuracy at 95%. Then we apply the interpretation scheme on each of the three datasets and the decision boundaries generated by the corresponding classifiers.

The interpretation framework on API call frequencies and existence generate similar insights, where we find that one of our malicious sample interpretations focused on cryptography-related API calls and HTTP-related calls, both of which are common ways for malware to communicate with their command-and-control (C&C servers). We note that these extracted insights are not exclusive to malware, since legitimate Internet browser applications perform similar activities. In fact, this lends insight into the shortcomings of our benign dataset and what types of benign software we may be missing to improve the reliability of our classifier.

The API sequence call has been shown to be a weak feature in past work [21]. Using the interpretation framework, we examine whether the API sequence is a weak feature without relying on domain expertise. Although training this model resulted in a validation accuracy of 94%, our interpretation results are not intuitive. While the model was confident in classifying one the benign samples (Fig 3), the interpretation on its boundary approximations is ambiguous at interpreting why this was the case. There are large sections highlighted as contributing to the classification of the sample. The most heavily weighted areas make frequent calls to *FindNextFileW* and *GetProcAddress* (among others), but this isn't indicative of benign *or* malicious behavior. When looking at malicious samples, the results are even more ambiguous. It seems the model memorized at least one of the samples entirely as seen in Fig 4.

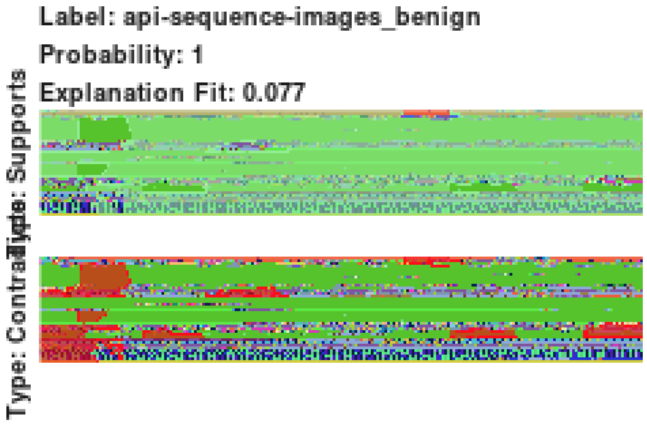

Figure 3. Interpretation for a benign sample. Each color represents a unique Windows API call during execution. The large light green areas in the top image denote support for the benign class. The dark red areas in the bottom image contradict the support for the benign class.

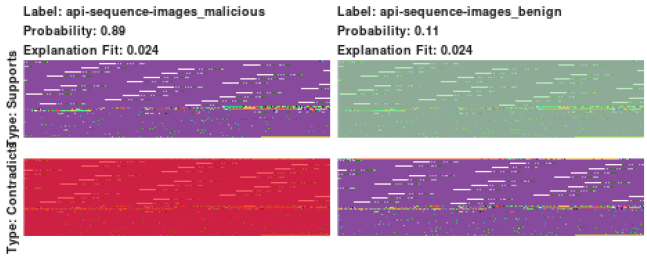

Figure 4. Interpretation on a malicious image generated from API sequences, where each pixel represents a unique Windows API call during execution. Even though the vision-based classification scheme correctly predicts this sample as malicious with high confidence, the interpretation method that approximates the boundaries provides ambiguous explanation.

4. Conclusion

In this paper, we demonstrate the effectiveness of using computer-vision based techniques for dynamic malware classification and employing vision-based interpretation frameworks to explain why the deep learning models make such predictions. Our discoveries on the two case studies indicate the promising advantages of applying vision-based interpretation frameworks to explain image-based dynamic malware classifiers. Security practitioners, based on the algorithmic interpretation findings, can check the code and verify whether the ML-identified locations contain a signatures unique to certain malware families. We plan to continue studying and proposing interpretation schemes specifically for image-based malware classification frameworks.

References

[1] Apiary. http://apiary.gtri.gatech.edu/. Accessed: 2019-03-28.

[2] Cnet. https://www.cnet.com/. Accessed: 2019-03-29.

[3] Cuckoo. https://cuckoosandbox.org/. Accessed: 2019-03-29.

[4] Exploit:win32/pdfjsc.aew. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Exploit:Win32/Pdfjsc.AEW. Accessed: 2019-03-27.

[5] Virustotal. https://www.virustotal.com/. Accessed: 2019-03-29.

[6] L. Chen. Deep transfer learning for static malware classification. *arXiv preprint arXiv:1812.07606*, 2018.

[7] L. Chen, S. Sultana, and R. Sahita. Henet: A deep learning approach on intel ® processor trace for effective exploit detection. *IEEE Symposium on Security and Privacy Workshop. arXiv preprint arXiv:1801.02318*, 2018.

[8] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62. ACM, 2008.

[9] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[10] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[11] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 364–379. ACM, 2018.

[12] D. Kirat, G. Vigna, and C. Kruegel. Barecloud: bare-metal analysis-based evasive malware detection. In *Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA*, pages 287–301, 2014.

[13] C. Kolbitsch, E. Kirda, and C. Kruegel. The power of procrastination: detection and mitigation of execution-stalling malicious code. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 285–296. ACM, 2011.

[14] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[15] A. Makandar and A. Patrot. Malware image analysis and classification using support vector machine. *International Journal of Trends in Computer Science and Engineering*, 4(5):01–03, 2015.

[16] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*, page 4. ACM, 2011.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[18] P. Royal. Entrapment: Tricking Malware with Transparent, Scalable Malware Analysis. *Talk at Blackhat*, 2012.

[19] M. Sebastin, R. Rivera, P. Kotzias, and J. Caballero. AVclass: A Tool for Massive Malware Labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264. ACM, 2002.

[22] S. Yue. Imbalanced malware images classification: a cnn based approach. *arXiv preprint arXiv:1708.08042*, 2017.