**A Proposal to Automatically Evasion for Classifiers**

A Case Study of PDF Classifiers Evasion Attack

Tengyi Wang

Rochester Institute of Technology

**A Proposal to Automatically Evasion for Classifiers**

**1. Introduction**

Machine learning is a specific application of artificial intelligence that allows for computers and programs to learn how to improve themselves without being programmed to do so. Machine learning is a process that includes the usage of data and the patterns of that data. The objective is to make sure that computers learn without any type of help by humans. It is one of the most state-of-art multidisciplinary subjects, which help humans solve many problems in many fields, including gaming, chatting robots, speaker recognition, results prediction, malware detection, adversarial example implementing, etc. Machine learning comes up with big data, meaning that machine learning needs a lot of data supporting it to have better performance.

Machine learning has been widely applied for malware detection because of its powerful ability in solving enormous data. Many researchers, for example, Yerima et al (April 2015), have claimed that they have achieved over 99% accuracy with very low false positive rates in PDF malware detection[[[1]](#endnote-1), [[2]](#endnote-2)], which is a hot field since PDF (Portable Document Format) is the most widely used format for consistent contents.

However, most of the researchers having done their works are lacking consideration that those works are susceptible to some situations due to the fact that almost all the works are done based on specific datasets which can be made use of by attackers. When machine learning comes into security field, the attackers had come up with the idea of adversarial examples. Adversarial examples, unlike Generative Adversarial Networks, are to add up some imperceptible subtle changes by human to the original images to result in immense changes in neural networks by machines, making them wrongly classify by evaluating confidence[[[3]](#endnote-3)].

Evasion attacks, including white-box attack and black-box attack, is to perturb a normal input to be an adversarial example into the classifier. As for now, since many researchers have not provided models and methods against adversaries, I am willing to simulate an adaptive attacker who is attending to fool a classifier using immune algorithm, which is a branch in genetic algorithm. I aim to use such algorithm to generate noises automatically without hurting the malware functionality. This can be considered as a method to assess whether a classifier is robust or not.

**2. Background**

In this section, I will briefly introduce PDF, PDF format, PDF malware classifiers and automatically evading for PDF malware classifiers.

**2.1 PDF**

Portable Document Format (PDF) is a file developed by Adobe with a consistent content regardless of software, hardware and operating systems and therefore, it has become one of the most widely used file formats worldwide because of its archivability[[[4]](#endnote-4), [[5]](#endnote-5)]. According to the figure of *“PDF as a percentage of document formats on the web”,* the PDF has had dominant position in document formats of over 85% by 2018[[[6]](#endnote-6)].

**2.2 PDF Structure**

Single PDF file consists of two structures including file structure and document structure. In file structure, there are four sections: header, body, cross-reference table, and trailer. Header situates the first line of a PDF file with version number as well as a magic number. The body consists of a variety of objects, including text streams, images, other elements, etc. Simply speaking, body section is the content of this PDF file shown to the user. The cross-reference table usually locates near the end of the file and contains the references to all the objects that allows users to have access to different objects quickly and simply. Trailer, the last part but the first part where application starts to read, instructs the application which is reading such PDF document to find the cross-reference table, and must contain a dictionary within two entries: /Root and /Size. The document structure, which is directed by /Root in trailer, contains document catalog, pages, and page object these three parts.

In the file structure, the body part is usually the target of attackers due to its tree-like structure.

**2.3 PDF Malware Classifiers**

There have existed four types of machine learning: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning, while supervised learning has been the most widely used one, where each sample in a dataset will be with an identified label certified by humans.

Sharma, Krishna and Sahay (2019) details that machine learning technique usage across networks has been used primarily for malware detection due to the fact that feature selection has been a central part of the machine learning technique[[[7]](#endnote-7)]. Following feature selection, the aim has been to obtain the finest classifier possible to make sure there is accurate detection of any advanced malware. Most anti-malware solutions have been a combination of dynamic and static analysis approaches.

Classifiers prefer to use machine learning analyzing the file behavior, file structure, file size, etc. to judge whether the file is malicious or not. In general, one of the most efficient way to do so is to analyze dependencies and behaviors because compared to benign files, those malicious files usually have different behaviors and dependencies rarely seen.

For this paper, I choose two targeted state-of-art, open-source classifiers: PDFrate and Hidost.

***PDFrate***

PDFrate[[[8]](#endnote-8)] is one of the state-of-art classifiers based on ensemble learning, which is often used in supervised learning, using Random Forest classification method while is based on R[[[9]](#endnote-9)] environment. Within Random Forest, they achieved the best results of True Positives over 99% while False Positive less than 0.2%8. Based on dataset Contagio[[[10]](#endnote-10)], PDFrate randomly selects respectively 5000 benign and 5000 malicious samples to train itself. Activated by Weixin Yu, Yanjun Qi, and David Evans[[[11]](#endnote-11)], I choose the same reimplemented version of PDFrate named Mimicus[[[12]](#endnote-12)] developed by Nedim et al[[[13]](#endnote-13)].

**Hidost**

Hidost[[[14]](#endnote-14)] is another state-of-art targeted classifier in this paper. Compared to PDFrate, Hidost is using Support Vector Machine (SVM) classification method being supervised learning model instead of Random Forest method. There also exist some decision trees helping for reducing variance. Hidost collects the dataset from VIRUSTOTAL[[[15]](#endnote-15)] mostly and google search, with the same amount of 5000 samples for both benignity and maliciousness. The authors in the paper claimed that Hidost had achieved great results against even the strongest conceivable mimicry attacks within only 2 added misclassifications14. It has even better performance than PDFrate that it has achieved over 99.8% accuracy of true positive and less than 0.06% false positive.

**2.4 PDF malware**

Nowadays, operating systems have become more and more secure, because “security fixes are constantly released, and the possibility of finding Zero-Day Vulnerabilities is reduced”[[[16]](#endnote-16)]. However, attackers will never give up finding ways to burst into user’s computers. Beneath such circumstance, attackers have become paying more attention to third-party applications and vulnerabilities in their structure.

Because of the dominant position PDF has in electronic document format6, the attackers are more willing to aim attacking PDF files since more people are willing to keep files in PDF and read them in PDF. PDF malware usually aims to make the operating system run its shellcode, which might crash PDF reader, make content chaotic, installing other dictionaries for other works, etc.

While the most commonly used one is supervised learning that most of the malware classifiers will adopt it, the classifiers have higher accuracy to detect existing malware or malware similar to existing malware. However, without updating from humans, they can hardly detect malware new or with noise since they cannot stretch out the features stored in the database within designed features and labels. For example, attackers used to like to exploit PDF based on JavaScript (e.g. Wepawet[[[17]](#endnote-17)]) while defenders have developed robust-enough tools for detecting. Therefore, JavaScript are being made use of less than before and attackers have developed more methods to hide JavaScript[[[18]](#endnote-18)] responding to updated techniques, while those updated detection mechanisms can hardly detect updated techniques used by attackers if they find something new. Therefore, there has been automatic evasion for malware detection for supervised-learning classifiers.

In this work, I assume the attacker aims to have an automatic evasion attack based on Immune Algorithm (IA), which can be considered as an extension of genetic algorithm, to the PDF malware classifiers mentioned above in black-box model, which means that the attacker can only obtain little information, like part of the labels, about the classifiers, without damaging the original functionality of malware.

***Genetic Algorithm***

Genetic Algorithm (GA) derives from evolutionary algorithms, which was inspired by evolution of animals. It is a method to search for the optimal results by simulating the process of natural selection-according to Darwin, only the variants (in this work) more suitable to the environment will be selected for the following “evolutions”. The basic idea of it is just simple: first an individual of the population is initialized in a computer followed by evolving with the principles of variation, selection, and inheritance[[[19]](#endnote-19)]. There are two operators called crossover and mutation.

***Immune Algorithm***

Immune Algorithm (IA)[[[20]](#endnote-20)] belongs to Artificial Immune System, and introduces another operator named immune operator within two types of Full Immunity and Target Immunity based on genetic algorithm. The approximate procedure of the IA is shown in figure 2.4-1 below. The approximate procedure can be summed up in text:

1. Input Antigen, which means the problem needing to be solved.
2. Randomly generate parent group A1.
3. Extract vaccine, which is basic feature information. Vaccine can be various, and operator can select one vaccine, as well as a number of combined vaccines following specific logics for the next step.
4. Adaptability calculation followed by judging whether having best individual. If yes, end and output result. If no, continue to 4.
5. Do C*rossover operation* for the kth parent group Ak to get the result Bk.
6. Do *Mutation operation* for the Bk to get the result Ck.
7. Do *Vaccination operation* for the Ck toe get the result Dk. This operation aims to improve the adaptability.
8. Do *Immune Selection operation* for the Dk to get the new parent group Ak+1 backing to process 4. This operation aims to prevent the group from degeneration. There are two steps here: first check the adaptability, if the adaptability is lower than parent, then using the relative parent individual Ak-x to replace this individual Dk-x to get Ek group. If Dk-x’s adaptability performs better than parent individual, then using annealing selection[[[21]](#endnote-21)], meaning randomly (following specific function) choose individual xi to be in the new parent group Ak+1.

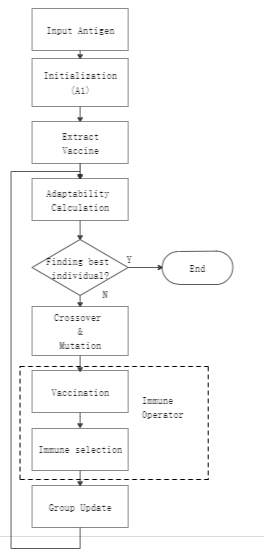


Figure 2.4-1 Flow Chart of Immune Algorithm

**3. Related Work**

Hu & Tan (2017)[[[22]](#endnote-22)] discussed how machine learning algorithms, even with their great functionality, are often vulnerable to intentionally targeted attacks. Machine learning based malware detection algorithms are integrated into software either at the installation level or on the cloud side. These still provide a black box system to authors of malware, though. It creates much less opportunity for exploitation because the authors have less understanding of the classifiers and parameters associated with the classifier. The malware authors have to, in some instances, carefully create test cases, which help to assess the features of the malware detection algorithm.

Machine learning algorithms have been applied much more to address and meet the requirements for proper security because of the constant changes in standards. This has allowed for an addressing of any parsed information of the file. There has been an automatic tuning of the meaning with any new data that is presented without a specific need for expert knowledge of a certain domain (Vinayakumar et al., 2018)[[[23]](#endnote-23)].

Goodfellow, Shlens & Szegedy (2015)[[[24]](#endnote-24)] furthered the discussion of how machine learning is highly exploitable and vulnerable to adversarial examples. The reason behind the high vulnerability remains something that many researchers have tried to assess, but have not fully been able to determine outright. Most explanations put forth have reasoned that because of deep neural network nonlinearity, that a combination of insufficiency along with that have created the problem with machine learning algorithms. Many frameworks have been easy to train because of how they are constructed and therefore, by creating a more robust optimization technique in the frameworks, there may be a lessening of the vulnerability of machine learning models to adversarialexamples.

Kolosnjaji et al. (2018)[[[25]](#endnote-25)] explains that detecting malicious binaries is something that remains a computer security task. Machine learning has become a much more adopted method in order to counter the attacks. While research is showing that a substantial amount of discussion on malware detection, recent studies are looking at the usage of deep-learning algorithms on raw bytes in order to add to the accuracy of the samples. Research has shown that deep-learning methods, which are a branch of machine learning, are very vulnerable to evasion attacks for most of the parts.

In more specific terms, Kolosnjaji et al. (2018)[25] identify that through changing bytes that were related in particular to debug information, the executable could be changed without damaging or destroying its function. Additional changes require a bit more complexity in order to make sure that there is no damage that was done to the functionality. Another option debated was packing, which is highlighted through a compression of the executable files. The executable files would then be decompressed. The problem with packing was stated is that to perform the necessary modifications to any files, the process has to be very specific and fine-tuned to achieve the point of being a good option for using.

Xu, Qi & Evans (2016)[11] find that machine learning is a technique that has shown to be helpful in malware classification. The authors created adversarial examples against two state-of-art classifiers within 100% possibility to at least create one sample evading the classifiers. They observed that in order to allow for the system to use machine learning, it requires feature extraction, as algorithms are for the most part represented as a future space, where each feature is indicated as a vector. One of the more popular types of algorithms the authors explained was supervised learning, which provides the data sets with specific labels alerting the training sample classes. Genetic programming was also discussed, which was created initially to generate computer programs which can be executed to perform certain tasks. A key part of the discussion was on PDF Malware, which was noted as something that has become much more widespread in recent years, especially because it is a user-friendly format that is widely accepted as the typical way of signing a form or viewing a document. PDF malware has been shown to take over JavaScript objects and exploit their vulnerabilities. In their assessment, the authors identified that in order to evade classifier, they needed to parse the file as a tree-like representation. The authors then generated several variants in order to manipulate the files at the level of the object structure. The authors additionally observed that through an automatic evasion method, they could enable more target efficiency in challenging the classifiers. To effectively capture an algorithm, they found that through using a probabilistic model will help them find the good mutations for more efficiency since a totally random search algorithm might generate corrupted useless variants changing too much resulting in the fact that malware loses functionality. (Cuan et al., 2018)[[[26]](#endnote-26)]. Such attacking method has arisen the threat level of PDF malware by **automatically evading**. Definitely, the defenders should pay more attention to coming up with an automatically defending classifier.

Besides genetic algorithm, immune algorithm, developed by Wang, Pan & Jiao (2000)[20], is also a great algorithm based on genetic algorithm. Compared to basic genetic algorithm, by adding immune operator within two more steps, immune algorithm greatly decreases the fluctuation phenomenon of genetic algorithm late stage, and profoundly accelerates the convergence speed of genetic algorithm, improving the whole performance and accurate of genetic algorithm.

Adversarial examples have been a hot field attracting many experts and attackers since they are definitely hard to detect[[[27]](#endnote-27)] and have become a difficult problem for white hats. There have been many ways to detect adversarial examples[[[28]](#endnote-28), [[29]](#endnote-29)]. However, it is still very hard to make classifiers keep pace with adaptive attacks, while typical adaptive evaluations are not perfect[[[30]](#endnote-30)]. In addition, such detection is usually limited in specific field or specific examples. Chinavle et al. argued that the adversarial is a well-studied field in machine learning that considers data distributions which change over time[[[31]](#endnote-31)]. Creating adversaries is rather simple while detecting is rather complicated.

Recently, Chen et al (2019, December)[[[32]](#endnote-32)] think even the state-of-art classifiers are susceptible to even a simple adversary, and they have developed a classifier increasing the evasion cost of unbounded attackers by automatically eliminating simple evasion attacks, and they have claimed that they have achieved better performance even than any of the state-of-art classifiers existing nowadays. Such classifier, though, has not achieved the best status since it can just eliminate some simple adversaries, while attackers may use more complicated techniques to generate more complicated adversaries.

**4. Project Idea**

In this work, I am planning to implement an PDF malware based on immune algorithm developed by Wang, Jin & Jiao (2000), which was the first version of immune algorithm detached from previous works of “genetic algorithm with immune features”, to create an adversarial example which has been certified as malware by state-of-art classifiers aiming to automatically evade them without any perturbations to its original malicious functionalities.

This work was inspired by Xu, Qi & Evans’ work (2016)11. They have developed an automatically evading for classifiers based on genetic algorithm. However, genetic algorithm has severe fluctuating problem approaching to the end stage, resulting in lower convergence speed and worse results in performance and accuracy. That inspires me to try immune algorithm, which is an extension of genetic algorithm, to implement a new automatically evading for the state-of-art classifiers.

This work will find sample PDF malware from Contagio10 database. The sample PDF malware should have been detected as malware by state-of-art classifiers. Based on the malware, I am planning to create an adversary using immune algorithm.

PDFrate8 and Hidost14 will be being the two targeted state-of-art classifiers in this work. As those mentioned in Xu, Qi & Evans’ work (2016)11, I will be using the same codes used by them as the classifiers. That means that for PDFrate classifier, I will be using Mimicus12 version of PDFrate which can be found online with open-source codes.

First, I will be finding PDF parser, which parse the PDF file into tree-like structure. Here I use an open source python-based parser, pdfrw[[[33]](#endnote-33)]. Loosening the grammar checking, the actual version of pdfrw being used will be the same as the version can be found in <https://github.com/mzweilin/PDF-Malware-Parser>.

After that, I will be building up immune algorithm, including genetic operators of crossover and mutation, oracle which determines whether the malicious functionality is preserved, fitness function which outputs score of a variant, adaptability function which calculates adaptability, selection function with global annealing algorithm which selects the best individual without degeneration, vaccination which improves adaptability, immune operators, etc.

I might be spending one week in finding PDF malware, setting up classifiers, setting up environment, implementing genetic algorithm part (two operators), and another two or three weeks implementing immune algorithm, with oracle, fitness function, adaptability function, etc.

If time permits, I am willing to create an adversary against what Chen et al[32] did for improving classifiers against automatically evading methods by implementing robust models and see whether their work can defend my adversary evading attack.

For the final week, I am planning to work on the final essay.

1. **References**

   []C. Smutz and A. Stavrou. Malicious pdf detection using metadata and structural features. *Proceedings of the 28[th] annual computer security applications conference*, pages 239–248. ACM, 2012. [↑](#endnote-ref-1)
2. [] N. Šrndic and P. Laskov. Detection of malicious pdf files based on hierarchical document structure. *Proceedings of the 20[th] Annual Network & Distributed System Security Symposium*, pages 1–16. Citeseer, 2013. [↑](#endnote-ref-2)
3. [] [Christian Szegedy](https://research.google/people/ChristianSzegedy/), Wojciech Zaremba, Ilya Sutskever, Joan Bruna, [Dumitru Erhan](https://research.google/people/DumitruErhan/), Ian Goodfellow, Rob Fergus, *International Conference on Learning Representations* (2014) [↑](#endnote-ref-3)
4. [] Adobe Systems Incorporated, PDF Reference, Sixth edition, version 1.23 (53 MB), Nov 2006, p. 33. [↑](#endnote-ref-4)
5. [] Smith E. https://www.vice.com/en\_us/article/pam43n/why-the-pdf-is-secretly-the-worlds-most-important-file-format. Accessed March 27, 2020. [↑](#endnote-ref-5)
6. [] Johnson D. *PDF Days Europe 2018*.; 2018:40. https://www.pdfa.org/wp-content/uploads/2018/06/1330\_Johnson.pdf. Accessed March 27, 2020. [↑](#endnote-ref-6)
7. [] Sharma, S., Krishna, C. R., & Sahay, S. K. (n.d.). Detection of advanced malware by machine learning techniques. https://arxiv.org/ftp/arxiv/papers/1903/1903.02966.pdf [↑](#endnote-ref-7)
8. [] C. Smutz and A. Stavrou. Malicious pdf detection using metadata and structural features. *Proceedings of the 28[th] annual computer security applications conference*, pages 239–248. ACM, 2012. [↑](#endnote-ref-8)
9. [] R project. http://www.r-project.org/. [↑](#endnote-ref-9)
10. [] Stephan Chenette. *Malicious Documents Archive for Signature Testing and Research* - Contagio Malware Dump. http://contagiodump.blogspot.de/2010/08/malicious-documents-archive-for.html. [↑](#endnote-ref-10)
11. [] Xu, Weilin, Yanjun Qi, and David Evans. "Automatically evading classifiers." *Proceedings of the 2016 network and distributed systems symposium*. Vol. 10. 2016. [↑](#endnote-ref-11)
12. [] Nedim Šrndic and Pavel Laskov. Mimicus: *A Library for Adversarial Classifier Evasion*. https://github.com/srndic/mimicus. [↑](#endnote-ref-12)
13. [] Nedim Šrndic and Pavel Laskov. Practical Evasion of a Learning-Based Classifier: A Case Study. In *35th IEEE Symposium on Security and Privacy (Oakland)*, 2014. [↑](#endnote-ref-13)
14. [] Nedim Šrndic and Pavel Laskov. Detection of malicious pdf files based on hierarchical document structure. In *Proceedings of the 20[th] Annual Network & Distributed System Security Symposium*, pages 1– [↑](#endnote-ref-14)
15. [] VirusTotal. Free Online Virus, Malware and URL Scanner.

    https://www.virustotal.com/. [↑](#endnote-ref-15)
16. [] Maiorca, Davide, Giorgio Giacinto, and Igino Corona. "A pattern recognition system for malicious pdf files detection." *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, Berlin, Heidelberg, 2012. [↑](#endnote-ref-16)
17. [] Marco Cova, Christopher Kruegel, and Giovanni Vigna. Detection and Analysis of Drive-By-Download Attacks and Malicious JavaScript Code. In 19th International World Wide Web Conference (WWW), 2010. [↑](#endnote-ref-17)
18. [] Karthik Selvaraj and Nino Fred Gutierrez. The Rise of PDF Malware. https://www.symantec.com/content/en/us/enterprise/media/security response/whitepapers/the rise of pdf malware.pdf, March 2010. [↑](#endnote-ref-18)
19. [] Stephanie Forrest. Genetic Algorithms: Principles of Natural Selection Applied to Computation. *Science*, 261(5123), 1993. [↑](#endnote-ref-19)
20. [] Wang, Lei, Jin Pan, and Li-Cheng Jiao. "The immune algorithm." *Acta Electronica Sinica* 28.7 (2000): 74-78. [↑](#endnote-ref-20)
21. [] Zhang, J., Xu, Z. & Leung, Y. Global annealing genetic algorithm and its convergence analysis. *Sci. China Ser. E-Technol. Sci.* **40,**414–424 (1997). https://doi-org.ezproxy.rit.edu/10.1007/BF02919428 [↑](#endnote-ref-21)
22. [] Hu, W., & Tan, Y. (2017, February 20). Generating adversarial malware examples for black-box attacks based on GAN. https://arxiv.org/abs/1702.05983 [↑](#endnote-ref-22)
23. [] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2018, December). Robust intelligent malware detection using deep learning. *IEEE Access*, *7*, 46717-46738. doi:10.1109/ACCESS.2019.2906934 [↑](#endnote-ref-23)
24. [] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015, March 20). *Explaining and harnessing adversarial examples*. https://arxiv.org/abs/1412.6572 [↑](#endnote-ref-24)
25. [] Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., & Roli, F. (2018, March 12). Adversarial malware binaries: Evading deep learning for malware detection in executables. https://arxiv.org/abs/1803.04173 [↑](#endnote-ref-25)
26. [] Cuan, B., Damien, A., Delaplace, C., & Valois, M. (2018). Malware detection in pdf files using machine learning. https://hal.archives-ouvertes.fr/hal-01704766/file/Malware%20Detection%20in%20PDF%20Files%20Using%20Machine%20Learning%20SECRYPT%2718.pdf [↑](#endnote-ref-26)
27. [] Carlini, Nicholas, and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017. [↑](#endnote-ref-27)
28. [] Liang, Bin, et al. "Detecting adversarial image examples in deep neural networks with adaptive noise reduction." *IEEE Transactions on Dependable and Secure Computing* (2018). [↑](#endnote-ref-28)
29. [] He, Warren, et al. "Adversarial example defense: Ensembles of weak defenses are not strong." *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*. 2017. [↑](#endnote-ref-29)
30. [] Tramer, Florian, et al. "On adaptive attacks to adversarial example defenses." *arXiv preprint arXiv:2002.08347* (2020). [↑](#endnote-ref-30)
31. [] Deepak Chinavle, Pranam Kolari, Tim Oates, and Tim Finin. Ensembles in Adversarial Classification for Spam. In 18th ACM Conference on Information and Knowledge Management (CIKM), 2009. [↑](#endnote-ref-31)
32. [] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana (2019, December 3). On Training Robust PDF Malware Classifiers. https://arxiv.org/pdf/1904.03542.pdf [↑](#endnote-ref-32)
33. [] Patrick Maupin. PDFRW: A Pure Python Library That Reads and Writes PDFs. https://github.com/pmaupin/pdfrw. [↑](#endnote-ref-33)