## Title

When A Tree Falls Using Diversity In Ensemble Classifiers To Identify Evasion In Malware Detectors

## Authors

Charles Sumutz & Angelos Stavrou from GMU

## Abstract

Machine learning classifiers are a vital component of modern malware and intrusion detection systems. However, past studies have shown that classifier based detection systems are **susceptible to evasion attacks** in practice. **Improving the evasion resistance of learning based systems is an open problem.** To address this, we introduce **a novel method for identifying the observations on which an ensemble classifier performs poorly.** During detection, when a sufficient number of votes from individual classifiers disagree, the ensemble classifier prediction is shown to be unreliable. The proposed method, **ensemble classifier mutual agreement analysis, allows the detection of many forms of classifier evasion without additional external ground truth.**

**We evaluate our approach using PDFrate, a PDF malware detector.** Applying our method to data taken from a real network, we show that **the vast majority of predictions can be made with high ensemble classifier agreement.** However, most classifier evasion attempts, including nine targeted mimicry scenarios from two recent studies, **are given an outcome of uncertain indicating that these observations cannot be given a reliable prediction by the classifier.** To show the **general applicability of our approach,** we tested it **against the Drebin Android malware detector where an uncertain prediction was correctly given to the majority of novel attacks.** Our evaluation includes **over 100,000 PDF documents and 100,000 Android applications.** Furthermore, we show that our approach can be generalized to **weaken the effectiveness of the Gradient Descent and Kernel Density Estimation attacks against Support Vector Machines.** We discovered that **feature bagging** is the most important property for enabling ensemble classifier diversity based evasion detection.

## I. INTRODUCTION

The use of machine learning has emerged as one of the primary techniques employed to address a wide range of malfeasance and malicious activities. Applications of machine learning include **clustering of malware families** [7], [20], **detection of malicious downloads** [12], [34], **detection of account misuse in social networks** [14], [44], **and detection of commonly exploited file formats such as Java archives** [36] and **documents** [24], [25], [39]. Moreover, statistical or machine learning techniques have been used successfully **for years to identify SPAM** [11], [21], [35].

One of the main weaknesses of systems that **employ machine learning classification in adversarial environments is their susceptibility to evasion attacks**. With evasion attacks, we refer to the classes of attacks that take advantage of knowledge of how the machine learning system operates, and in many cases utilize access to the training set and features, to **evade detection passively or actively** [8], [9], [15], [33], [45].

A common technique used in **evasion attacks** against machine learners is **mimicry**. Mimicry attacks thwart detection by making the **attack data appear benign** according to the model used by **the intrusion detection system**. Often this is achieved by **hiding overtly malicious content through encoding or encryption** [28], [42] or minimizing the footprint of malicious content **through data misuse or code re-use attacks** [17], [37]. For instance, content aligning with a benign observation is added to cover up or drown out **the malicious content**. **Many detection systems are evaded by exploiting differences in the detection system and the systems being protected** [16], [19]. **Even if operational details of defense systems are kept secret, enough knowledge to conduct evasion can often be obtained solely from external testing** [18]. With all of these potential **evasion vectors**, **preventing detection evasion remains an open problem**.

Our approach is not to prevent all possible evasion attacks, but to introduce a mechanism that **provides detection of poor classifier performance**. We analyze the use of introspection in an ensemble classifier to detect **when the classifier provides unreliable results at classification time. The use of ensemble classifier mutual agreement analysis relies on the intuition that when individual classifiers in an ensemble vote for the same prediction, the prediction is likely to be accurate.** When a sufficient number of the votes are in opposition, then the classifier prediction is not trustworthy. In this state of internal classifier disagreement, the detector returns the outcome of uncertain **instead of a prediction of benign or malicious**. In operation, confidence in the predictions of the classifier is improved at the cost of a small portion of the samples being labeled as uncertain, indicating that the classifier is not fit to provide an accurate response. This separation of accurate predictions from uncertain predictions is possible because the majority of the misclassifications, including evasion attempts, have a classifier voting score distribution distinct from the accurate predictions.

To evaluate our technique, we applied **mutual agreement analysis** to **two well-studied malware detection systems**: PDFrate [40] and Drebin [4]. PDFrate uses features derived from document structure and metadata fed into a Random Forest classifier to detect Trojan PDFs. **PDFrate is used in real world intrusion detection systems and can be evaluated by the public through submissions to pdfrate.com**. PDFrate was selected because it is **publicly accessible**, well documented, uses an ensemble classifier which returns the raw voting score, and **has been subjected to（饱受） multiple recently published mimicry attacks** [26], [27], [43]. Our evaluation includes over 100,000 documents sourced from an operational environment and hundreds of malicious documents in **nine unique evasion scenarios** from two independent evasion studies. To demonstrate the general applicability of our approach, we apply **mutual agreement analysis** to the Drebin Android malware detector using over 100,000 applications, including over 5,000 malicious applications in over 20 labeled malware families. We find that mutual agreement analysis enables the identification of novel malware that would otherwise not be detected reliably.

In building **an evasion resistant ensemble using Support Vector Machines (SVM) as base classifiers**, we find that feature bagging, or constructing many individual classifiers with randomized subsets of the whole feature set, is crucial to providing this discriminatory power. Using this method, we counter the Gradient Descent and Kernel Density Estimation (GD-KDE) attack, which is highly successful against a traditional SVM classifier.

## II. RELATED WORK

Previous work on the detection of document malware shares many common ideas with the methods for detecting drive-by-downloads. This is not surprising, since the exploitation techniques underlying both kinds of malware are the same. Existing methods can be classified, with some degree of overlap, into *dynamic* analysis methods, in which documents are opened in a specially instrumented environment, and *static* methods, in which detection is carried out without malware execution.

Several key ideas have fueled the development of dynamic analysis methods. Early work followed the emulation-based approach in which a suspicious payload was executed using abstract payload execution [36] or software emulation [1, 26]. However, software emulation does not have full coverage of the instruction set and hence can be detected and evaded. To overcome this problem and improve scalability, the recently proposed system SHELLOS uses hardware virtualization instead of emulation for controlled execution of shellcode [34]. Implemented as an operating system kernel, SHELLOS is able to effectively detect shellcode in any buffer allocated by an application. However, this effectiveness has its price. While SHELLOS performs with outstanding bandwidth in detecting network level attacks, its application to document malware suffers from high latency (on the order of seconds). Such latency is due to the fact that detection is carried out at the level of memory buffers which must be allocated by an application before they can be analyzed.

Another group of dynamic analysis methods has focused on detection of malicious behavior during execution of JavaScript. JSAND uses 10 carefully designed heuristic features to train models of benign JavaScript and detect attacks as large deviations from these models [9]. A similar approach has been successfully applied for detection of Ac- tionScript 3 malware [24]. CUJO is built on top of a specialized JavaScript sandbox and automatically learns models of sequences of events affecting the state of the JavaScript interpreter [31]. JavaScript-specific dynamic analysis methods improve on the performance of the methods focused on shellcode detection, bringing it in the range of hundreds of milliseconds per file, while maintaining high detection accuracy and an extremely low false positive rate.

Early static methods based on n-gram analysis [21, 32] have never been evaluated on modern PDF malware. Since they do not address some essential properties of the PDF format, such as encoding, compression and encryption, they can be easily evaded by modern PDF malware using tech- niques similar to those used against conventional signature- based antivirus systems. PJSCAN was the first method that demonstrated feasibility of anomaly-based static detection of PDF malware focused on JavaScript content [19]. For the sake of efficiency, the JavaScript extractor of PJSCAN only searches for locations where the presence of JavaScript is prescribed by the PDF Standard. Unfortunately, this ex- traction strategy can be defeated by placing JavaScript code into an arbitrary location accessible via the PDF JavaScript API and fetching it with an eval()-like function call. An- other recently proposed system, MALWARE SLAYER [23], is based on the pattern recognition methods applied to tex- tual keywords extracted from PDF documents using the PDFID tool. It exhibits excellent detection and false alarm rates on real PDF data but is limited to the extraction func- tionality of PDFID and can handle neither multiple revision numbers nor objects hidden in object streams. PDFRATE is a recent learning-based, static PDF classifier operating on simple PDF metadata and byte-level file structure evalu- ated on a large dataset of PDF files with excellent classifica- tion performance [33]. However, it does not extract object streams, a feature that could be used to hide features from the detector.

Another two contributions must be mentioned that combine static and dynamic analysis techniques. MDSCAN [37] employs static analysis of the PDF file in order to extract all chunks of JavaScript code that can serve as an entry point to the JavaScript execution. To this end, a special-purpose parser was developed for MDSCAN, which attempts to extract additional information from a file including objects omitted from a cross-reference table as well as potentially malformed objects. The extracted scripts are executed in a JavaScript engine which emulates the engine of Acrobat Reader. During the controlled execution, all memory buffers are checked using a tool for shellcode detection based on binary emulation (NEMU). In ZOZZLE [10], the roles of the static and the dynamic components are reversed. The dynamic part of ZOZZLE extracts parts of JavaScript from the JavaScript engine of Internet Explorer before their execution, which naturally unfolds JavaScript obfuscation. The static part of ZOZZLE uses Bayesian classification built on top of the syntactic analysis of detected JavaScript code.

The comparison of related work shows a clear trade-off exhibited in the up-to-date static and dynamic systems for document malware detection. While dynamic systems demonstrate excellent detection accuracy and low false positive rates, these advantage come at the cost of latency, performance overhead and the need for specially instrumented environments. The new method proposed in the paper attempts to bridge this gap from the static side, by boosting the detection performance while retaining the simplicity of design and computational efficiency typical for static methods. To achieve this goal, we develop the methodology for a *comprehensive static analysis* of PDF documents using an off-the-shelf PDF parser (POPPLER). Furthermore, we pay a special attention to potential evasion strategies and experimentally evaluate the robustness of the proposed method to selected attack strategies.

对相关工作的比较表明，在当前用于文档恶意软件检测的静态和动态系统间有明显的权衡（trade-off）。纵然动态系统表现出高准确度和低误报率，这种优势是以高延迟，性能瓶颈和需要特定的受控制系统（instrumented environments）为代价的。在本文中提出的新方法，旨在从静态方法处填补空隙。此技术通过从静态方法处，增强检测性能同时保持设计的简约和计算的高效。为了达到这样的目标，我们使用

References

* [1]  P. Akritidis, E. Markatos, M. Polychronakis, and K. Anag- nostakis. STRIDE: Polymorphic sled detection through in- struction sequence analysis. In *20th International Confer- ence on Information Security*, pages 375–392, 2005.
* [2]  M. Barreno, B. Nelson, A. Joseph, and J. Tygar. The secu- rity of machine learning. *Machine Learning*, 81(2):121–148, 2010.
* [3]  M. Barreno, B. Nelson, R. Sears, A. Joseph, and J. Tygar. Can machine learning be secure? In *ACM Symposium on Information, Computer and Communication Security*, pages 16–25, 2006.
* [4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.
* [5] L. Breiman, J. Friedman, J. Olshen, and C. Stone. *Classifi- cation and Regression Trees*. Wadsworth, 1984.
* [6] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *International Conference on World Wide Web (WWW)*, pages 197–206, 2011.
* [7] W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning (ICML)*, pages 115–123, 1995.
* [8] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
* [9] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *International Conference on World Wide Web (WWW)*, pages 281–290, 2010.
* [10] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. ZOZ- ZLE: Fast and precise in-browser JavaScript malware detec- tion. In *USENIX Security Symposium*, pages 33–48, 2011.
* [11] M. Engelberth, C. Willems, and H. T. MalOffice – analysis of various application data files. In *Virus Bulletin Interna- tional Conference*, 2009.
* [12] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through IDS- driven dialog correlation. In *USENIX Security Symposium*, pages 167–182, 2007.
* [13] Google warns of using adobe reader - particularly on linux. http://www.h-online.com/open/news/item/Google- warns-of-using-Adobe-Reader-particularly-on-Linux- 1668153.html.
* [14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer series in statistics. Springer, New York, N.Y., 2009. 2nd edition.
* [15] Vorsicht bei angeblicher telekom-onlinerechnung. http://heise.de/-1545909.
* [16] S. Jana and V. Shmatikov. Abusing file processing in mal- ware detectors for fun and profit. In *IEEE Symposium on Security and Privacy*, pages 80–94, 2012.
* [17] S. Kaplan, B. Livshits, B. Zorn, C. Siefert, and C. Cursinger. “nofus: Automatically detecting” + string.fromcharcode(32) + “obfuscated ”.tolowercase() + “javascript code”. Technical report, Microsoft Research, 2011.
* [18] P. Laskov and M. Kloft. A framework for quantitative secu- rity analysis of machine learning. In *Proceedings of the 2nd ACM Workshop on AISec*, pages 1–4, Nov. 2009.
* [19] P.LaskovandN.Sˇrndic ́. Staticdetectionofmalicious JavaScript-bearing PDF documents. In *Annual Computer Security Applications Conference (ACSAC)*, pages 373–382, 2011.
* [20] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pages 120–132, 1999.
* [21] W.-J. Li, S. Stolfo, A. Stavrou, E. Androulaki, and A. Keromytis. A study of malcode-bearing documents. In *Detection of Intrusions and Malware & Vulnerability As- sessment (DIMVA)*, pages 231–250, 2007.
* [22]  M.MahoneyandP.Chan.Learningrulesforanomalydetec- tion of hostile network traffic. In *International Conference on Data Mining (ICDM)*, 2003.
* [23]  D. Maiorca, G. Giacinto, and I. Corona. A pattern recogni- tion system for malicious pdf files detection. pages 510–524, 2012.
* [24]  T. V. Overveldt, C. Kruegel, and G. Vigna. FlashDetect: ActionScript 3 malware detection. In *Recent Adances in In- trusion Detection (RAID)*, pages 274–293, 2012.
* [25]  PDFReference. http://www.adobe.com/devnet/pdf/pdf reference.html, 2008.
* [26]  M.Polychronakis,K.Anagnostakis,andE.Markatos.Com- prehensive shellcode detection using runtime heuristics. In *Annual Computer Security Applications Conference (AC- SAC)*, pages 287–296, 2010.
* [27]  N.Provos,P.Mavrommatis,M.AbuRajab,andF.Monrose. All your iFRAMEs point to us. In *USENIX Security Sympo- sium*, pages 1–16, 2008.
* [28]  J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.
* [29]  Blackhole crimeware kit drives web threat spike. http://www.theregister.co.uk/2012/01/26/ sophos fakeav conficker/.
* [30]  K. Rieck, T. Holz, K. Willems, P. Du ̈ssel, and P. Laskov. Learning and classification of malware behavior. In *De- tection of Intrusions and Malware & Vulnerability Assess- ment (DIMVA), 5th International Conference*, pages 108– 125, July 2008.
* [31]  K. Rieck, T. Kru ̈ger, and A. Dewald. Cujo: Efficient de- tection and prevention of drive-by-download attacks. In *An- nual Computer Security Applications Conference (ACSAC)*, pages 31–39, 2010.
* [32]  Z. Shafiq, S. Khayam, and M. Farooq. Embedded malware detection using markov n-grams. In *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, pages 88–107, 2008.
* [33]  C. Smutz and A. Stavrou. Malicious PDF detection using metadata and structural features. In *Annual Computer Secu- rity Applications Conference (ACSAC)*, 2012. To appear.
* [34]  K. Z. Snow, S. Krishnan, F. Monrose, and N. Provos. Shel- lOS: Enabling fast detection and forensic analysis of code injection attacks. In *USENIX Security Symposium*, 2011.
* [35]  PDF malware writers keep targeting vulnerability. http://www.symantec.com/connect/blogs/pdf-malware- writers-keep-targeting-vulnerability.
* [36]  T. Toth and C. Kruegel. Accurate buffer overflow detection via abstract payload execution. In *Recent Adances in Intru- sion Detection (RAID)*, pages 274–291, 2002.
* [37]  Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. Markatos. Combining static and dynamic analysis for the detection of malicious documents. In *European Workshop on System Security (EuroSec)*, 2011.
* [38]  C. Willems, T. Holz, and F. Freiling. CWSandbox: Towards automated dynamic binary analysis. *IEEE Security and Pri- vacy*, 5(2):32–39, 2007.