Related Work 相关工作

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.

现有方法约可分成两类，纵然这有一定程度上的重合。动态方法需要把文档放在某一个特定探针环境（instrumented environment）中打开运行；而静态方法则无需打开运行仅通过静态特征即可完成检测。

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 which must be allocated by an application before they can be analyzed.

一些核心想法促成了动态分析方法的兴起。早起的工作使用基于模拟器的方法，此方法把可疑包（suspicious payload）使用抽象包执行（abstract payload execution）【36】或使用软件模拟【1，26】。然而，软件模拟并不能有效覆盖全部指令集。正因如此，此方法会被检测到或被逃逸。为了解决上述问题和增强可扩展性，最近提出的系统SHELLOS在可控shellcode执行的过程中【34】中使用硬件虚拟化技术以取代模拟方法。通过实现为操作系统内核，SHELLOS能有效地检测出shellcode（此shellcode在被程序任意分配的缓存之中）。然而，此有效性有它的代价。纵然SHELLOS在检测网络层攻击时显示出优秀的吞吐量，他的检测恶意文档的应用却遭受严重的延迟（在秒级别）。这种延迟造成的原因是检测位于内存缓存中，此类检测需要程序首先开辟内存缓存，才能实施检测。

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 ActionScript 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.

另外一种动态分析的方法聚焦于检测在运行JavaScript时的恶意行为。JSAND使用10个仔细设计的启发性特征去训练JavaScript的好模型，然后使用此模型去检测与此模型基线有较大偏移的攻击【9】。一个相似的方法也被成功运用于ActionScript3恶意软件的检测当中【24】。CUJO建立在专门针对JavaScript的沙箱上，且可自动学习事件序列的模型（models of sequences of events），这些模型均对JavaScript的解析器产生影响【31】。专门的Java Script动态分析方法与shellcode检测方法相比，有很大的性能提升（单个文件的检测时间控制为数百毫秒），同时也保持了高准确度和低误报率。

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 techniques similar to those used against conventional signature-based antivirus systems.

早期基于n-gram的静态分析方法【21，32】从来不运用于评估现代PDF恶意软件。因为他们不会去碰触某些基本PDF格式的属性，例如编码（encoding），压缩（compression）和加密（encryption），他们同时也会轻易地被现代PDF恶意软件逃逸（通过使用一些与对抗传统基于签名的杀毒软件系统相同的技术）

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 extraction 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.

PJScan是第一个，成功实现基于JavaScript内容的PDF恶意软件检测【19】。为了提高效率，PJSCAN的JS提取器只搜索那些PDF标准预先指定好的位置。不幸的，这种提取方法会在以下情况下有局限，可把JS代码放在任何能用PDF JS API或能够使用类eval()函数进行操作的地方。

Another recently proposed system, MALWARE SLAYER [23], is based on the pattern recognition methods applied to textual 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 functionality of PDFID and can handle neither multiple revision numbers nor objects hidden in object streams.

另外一个近期提出的系统MALWARE SLAYER【23】，基于模式识别。这里的模式是指使用PDFID工具从PDF文档中提取的文本关键字。它在真实PDF数据上展现出高准确度和低误报率。这种方法被PDFID自身的函数所局限，且不能处理多版本（multiple revision numbers）和在对象流（object stream）中的对象隐藏（object hidden）。

PDFRATE is a recent learning-based, static PDF classifier operating on simple PDF metadata and byte-level file structure evaluated on a large dataset of PDF files with excellent classification performance [33]. However, it does not extract object streams, a feature that could be used to hide features from the detector.

PDFrate是最近的一个基于机器学习的静态PDF分类器。此分类器使用简单的PDF附带信息（meta info）和字节层面的文件格式。此PDFrate在大数据集中有出色的分类表现【33】。然而，此分类器不提取对象流（object streams），这是一个能向分类器隐藏特征的特征。

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).

另外两个须提及的贡献是把静态和动态分析技术结合起来。MDScan【37】对PDF文件进行静态分析，目标是提取所有的JS代码簇（chunk），这些代码簇可以作为执行的入口（entry point）。直到现今为止，一个特定用途的解析器（parser）被MDScan开发，这个解析器被用于抓取在文件里面的其他信息，这些信息包括被交叉引用图表（cross-reference table）所忽略的对象以及潜在的恶意对象。被抓取的代码会在JavaScript引擎中被执行（此引擎会模拟Acrobat Reader引擎）。在控制执行中（controlled execution），所有的内存缓存都会被一个工具所检测，此工具基于二值模拟（binary emulation NEMU）以用于sheelcode检测。

In ZOZZLE [10], the roles of static and 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.

在ZOZZLE【10】一文中，静态与动态部件的角色发生了变化。ZOZZLE的动态部件会在运行前提取JAVASCRIPT的片段，这些JavaScript源于IE浏览器中的JavaScript引擎，此引擎能自然而然地解决JavaSript代码混淆（obfuscation）的问题。ZOZZLE的静态分析部分使用Bayesian分类器，此分类器建立在对检测到的JavaScript源代码进行混合分析（syntactic analysis）上。

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  *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。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开始，增强检测性能同时保持系统设计的简约和计算时的高效。为了达到这样的目标，我们使用了一个现成的解析器（parser），构筑关于“全面静态分析PDF”（Comprehensive Static Analysis）的方法论。更进一步地，我们对样本潜在的逃逸策略有特别的关注，且用实验去评估所提及的方法对特定攻击策略的健壮性。

\*\*\* 凤娇

传统的恶意PDF 检测方法有基于病毒检测、基于签名的的检测方法等，这些方法存在识别率不高、无法及时更新恶意代码等问题。机器学习技术为恶意PDF 检测提供了新方向，PDF 文档的检测研究大多采用PDF 文档内容或结构为特征[11]，利用随机森林、SVM、决策树等分类器构建PDF 检测器,。例如，Charles 等人提出通过随机森林检测含恶意代码的PDF 文件技术PDFrate，作者从PDF 文档元数据以及文档结构中提取了135 个特征，使用已标记特征的训练数据，并采用10 倍的交叉验证，生成具有多个分类树的分类器，从待测PDF 文档提取特征，评估森林中的每个树，最后投票决定其分类。该方法初始训练过程计算开销较大，但一旦分类器构建完成，对待测的PDF 文档的分类速度很高，我们同时也对这135个特征进行了提取，利用随机森林的算法对文档进行预测分类。

Maiorca等人。研究了对PDFrate和其他PDF文档分类器的逃逸[23]。他们提出 反向模仿技术。为了使内容看似良性（如Mimicus所做的那样），他们将恶意内容嵌入到良性PDF中，而不是将内容添加到恶意文档中，而是尽可能少地修改。反向模仿攻击实施针对PDFrate的独立逃避方法, Maiorca等人提出了三种不同的逃避方案。在EXEembed方案中，恶意可执行文件被植入到现有的良性PDF文档中。打开文档时会执行恶意软件。这些文档利用CVE-2010-1240。在PDFembed方案中，恶意PDF被嵌入到良性PDF中。这些嵌入式文档在文档打开时自动呈现。为了评估，Maiorca等人 将利用CVE-2009-0927的文档嵌入到现有的良性PDF文档中。最后，在JSinject场景中，恶意JavaScript（与PDFembed嵌入式文档中使用的相同）直接注入根良性文档。

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.

. [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] D. Maiorca, D. Ariu, I. Corona, and G. Giacinto, “A Structural and Content-Based Approach for a Precise and Robust Detection of Malicious PDF Files”, in Proceedings of the 1st International Conference on Information Systems Security and Privacy. ScitePress Digital Library,2015, pp. 27-36

. [19] P.LaskovandN.Sˇrndic ́. Staticdetectionofmalicious JavaScript-bearing PDF documents. In *Annual Computer Security Applications Conference (ACSAC)*, pages 373–382, 2011.

. [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.

[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.

. [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.

. [29] Blackhole crimeware kit drives web threat spike. http://www.theregister.co.uk/2012/01/26/ sophos fakeav conficker/.

[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.

. [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.