**Introduction**

Portable Document Format, or PDF, has become a democratized standard for document sharing and distribution in recent years. This is owing to its qualities such as portability and flexibility between platforms. The widespread use of PDF has instilled in average consumers a false sense of inherent safety. However, the properties of PDF encouraged hackers to exploit numerous types of vulnerabilities and circumvent security protections, making the PDF format one of the most efficient harmful code attack vectors. As a result, detecting fraudulent PDF files effectively is critical for information security. Several static and dynamic analysis techniques have been proposed in the literature to extract the primary properties that allow malware files to be distinguished from genuine files. Because traditional analytic techniques may be limited in the event of zero-day vulnerabilities, machine-learning-based techniques are emerging as a means of automatic PDF-malware detection from a set of training samples. These techniques are themselves facing the challenge of evasion attacks where a malicious PDF is transformed to look legitimate.

Malicious software (malware) is without a doubt one of the biggest threats to the global IT infrastructure that exists today. It is now a frequent tool in data theft, corporate and national espionage, spamming, and distributed attacks to hinder the availability of IT assets. Malicious attackers breach systems to install malware that allows them to acquire access and privilege, compromise personal or sensitive data, sabotage systems, or utilize them in other assaults like DDOS. It is nearly impossible to 100 percent prevent information system compromise. In fact, attackers plot assaults in a variety of ways, including drive-by downloads from websites exploiting browser vulnerabilities or network-accessible vulnerabilities. Furthermore, social engineering attacks such as Phishing and malicious email attachments enable user-authorized installation of harmful programs.(Sherly Abraham)

In 2010 Symantec (Karthik Selvaraj, 2010)reported a significant rise in PDF attacks, justifying it with a corresponding rise in the vulnerabilities identified in the Adobe Reader software. primarily due to an increase in the vulnerabilities detected in the Adobe Reader program. More recently, Ke Liu reported (Liu, 2017) on his discovery of more than 150 vulnerabilities in the most popular PDF reader software solutions since December 2015. This later story demonstrates how, even today, PDF is an important infection vector with a vast attack surface. As illustrated in Figure I, the number of PDF attacks registered by Symantec has increased, indicating that the PDF file type is being targeted more frequently. The spikes in these graphs correspond to the release of various PDF-related CVEs (Common Vulnerabilities and Exposures).

Figure 1
SOURCE: https://studylib.net/doc/18609162/the-rise-of-pdf-malware

Figure 1

Source : <https://studylib.net/doc/18609162/the-rise-of-pdf-malware>

The attacks described above are merely a small sample of the options available to an attacker in order to exploit the vulnerabilities of PDF readers. Nonetheless, they clearly demonstrate how sophisticated and diverse such attacks may be, as well as the difficulty of detection.

Despite the efforts of software vendors like Adobe, PDF software is frequently vulnerable to zero-day attacks, exploiting the integration of PDF file format's connection with third-party technologies (e.g., JavaScript or Flash), making the construction of ad-hoc patches difficult. Furthermore, due to the architectural complexity of PDF files and the wide range of code obfuscation techniques used by criminals, antivirus providers struggle to provide protection against innovative or even recognized attacks. Attackers adopt JavaScript code to exploit PDF vulnerabilities, by leveraging on well-known techniques, such as Return Oriented Programming and Heap Spraying

**Static and Dynamic Malware Analysis**

Due to the creative techniques threat actors employ to spread malware, Malware Detection, Endpoint Detection and Response (EDR) is a huge focus in industry and academia. Malware analysis and detection is an essential technology that extracts the runtime behavior of malware and supplies signatures to detection systems and provides evidence for recovery, cleanup, and forensics. Malware analysis is the first stage in effectively tackling malware spread, its goal is to extract the content of malware which is used to detect it. There are mainly two kinds of malware analysis: static analysis and dynamic analysis.

Static Analysis is a malware analysis technique in which the malware is not executed or run. It includes the following basic Static Analysis methods: Generic Information Analysis, Code Analysis, Control Flow Analysis and File Structure Analysis, these are the general stages for Static Analysis, however other research may include some additional stages. Generic Information aka Metadata analysis is usually the first thing that is done, then a code analysis which is a crucial technique in static malware analysis that involves inspecting the code of a malware sample to understand its behavior, and potential vulnerabilities. Reverse engineers and security analysts perform code analysis to gain insights into how malware behaves and to develop effective countermeasures. To understand the rationale and flow of execution, analysts examine the control flow of the malware's code. They look for loops, conditionals, function calls, and other control structures to figure out how malware works. File structure analysis is a (Tree based Analysis) which analyzes the interconnection between the different objects of the PDF. PeePDF is a publicly available software that automatically performs this operation. Origami is like PeePDF in that it allows you to visualize the PDF file structure. It also includes procedures for encrypting and decrypting files, extracting metadata etc.

Static analysis examines the code or file structure without running it. As a result, it is incapable of detecting dynamic malware behavior such as runtime activities, network communication, or interaction with the operating system. This limitation makes it difficult to identify certain kinds of malware that exhibit behavior only during runtime. Polymorphic or metamorphic (Samociuk, 2023) malware can dynamically modify its structure or appearance, making static analysis methods difficult to recognize patterns or signatures. These types of malwares can generate different versions of themselves, making it difficult to create static detection rules. If the malware protects its payload with encryption or compression, static analysis may struggle to extract and analyze the contents efficiently. Encrypted or compressed payloads can go undetected during static analysis, limiting its ability to detect malware. More recently, malware authors use obfuscation techniques to make the code unreadable and evade static analysis. Static analysis can be weak in detecting obfuscated code.

To overcome these drawbacks, a combination of static and dynamic analysis techniques, as well as other security measures like sandboxing, behavior monitoring, and network analysis, is often employed to enhance malware detection and analysis capabilities.

Contrary to Static Analysis, the Dynamic Malware Analysis executes the malware in run-time to prevent the disturbance of Shell Code, Polymorphism and Metamorphism.

It entails examining and comprehending the behavior and functionality of malware by executing it in an isolated controlled environment in contrast to static analysis, which includes inspecting malware's code or file structure without running it, dynamic analysis involves running malware in a controlled environment to examine its behaviors, interactions, and consequences on a system.  
  
Dynamic malware analysis is often carried out in a customized sandbox or virtual machine environment, where the malware can be safely executed without causing harm to the host system. This enables malware analysts to track malware operations such as file system calls, system alteration, network communications and changes in system state.  
The behavior, functioning, and impact of malware on a machine or network contributes to security. It assists security researchers and analysts in understanding the malware's techniques and tactics, identifying its capabilities (such as information theft, system compromise, or network spread), and developing countermeasures and mitigation plans.

Various monitoring and analysis techniques are utilized during dynamic analysis to record and evaluate malware behavior. System monitoring utilities, network traffic analyzers, code debuggers, and behavior analysis frameworks are examples of such tools. Analysts can discover harmful behaviors such as the generation of new files, attempts to modify system settings, and connection with command-and-control servers by attentively analyzing the malware's actions and interactions.

Dynamic analysis, although more advanced, is tricky and there is yet to be complete control over the extent of testing compared to Static Analysis. Analyzing malware whilst it runs can lead to uncontrollable and disruptive outcomes; to minimize the risk of infection and produce accurate results during dynamic analysis, the following characteristics and more must be met.

* It must be trusted; Data provided by the analysis framework must not be compromised by the malware (N. Nissim, 2018). A defensive mechanism should be in place to prevent the malware from gaining control of the system.
* It must be undetectable by the analyzed file: The malware could terminate itself or execute only non-malicious commands, if it detects it is being analyzed (M. Egele, 2012).
* It must capture as much relevant data as possible regarding the malware's action, this information can include function calls, parameters, networking, file system alterations, hardware measurements. (Bell, 1999). It must meet the malware’s expectations to expose its full behavior. The relevant operating system(s) must be installed, the appropriate hardware must be connected, and the vulnerable application must be installed.

Despite all its benefits, analyzing malware dynamically has a few limitations:

* Only the executed code is observable. This means that if a required condition is not met precisely, then some code might not be executed, therefore go unanalyzed.
* Dynamic analysis requires computational overhead, which will slow down execution.
* The analysis must be performed on the specific OS and/or hardware targeted by the malware.

For these reasons, this project is focused on supplementing signature matching with characteristic based analysis.

**Industry Status for characteristic-based PDF analysis**

A lot of advancements have been made on static and characteristic based approach as these techniques act even before the PDF is executed, hence it is decent first check.

Majority of PDF malware relies on embedded malicious JavaScript code. For this reason, specific features have been exploited to detect evidence of such behavior. The detection approach named PJScan (Srndic, 2011)aims to detect the presence of malicious (obfuscated) JavaScript code by considering occurrences of suspicious API calls like eval or replace, and of string-chaining operators like +, among others. Lux0R [ (I. Corona, 2014) leverages code instrumentation to detect the presence of API calls in JavaScript code that are specifically used for PDF-related operations. Wepawet dynamically executes the embedded JavaScript code using JSand, and then extracts features mostly related to method calls and shellcode memory allocation.

Analyzing the PDF file reveals that executables can be stored as an EmbeddedFile object. Investigators can quickly summarize the most important properties of a PDF document using Didier Stevens PDFiD. In a recent attack, (Schläpfer, 2022) a PDF included a malicious url stored in a docx file embedded in the pdf. The malicious PDF was quickly intercepted by HP Sure Click which ran the file in an isolated micro virtual machine, preventing their system from being infected. Further static analysis was carried out with PDFiD and pdf-parser.

Slayer relies on PDFiD, the updated version (Slayer NEO) (D. Maiorca, 2015)employs PeePDF for a more in-depth examination of embedded files, multiple versions, and streamed objects, and Origami for file structure and content integrity checks. These analyses are valuable for detecting PDF malware concealed by clever embedding techniques, as well as anomalous or corrupted files.

Library-based parsing is dependent on specific PDF libraries, which are also supported by open-source PDF readers. Poppler, a comprehensive PDF library used by the famous open-source reader XPDF, is the most well-known example.

The open-source pattern matching program YARA (Yet Another Recursive Acronym) is largely used in malware research and detection. It enables analysts to develop and implement bespoke rules or patterns for identifying and categorizing files or data based on certain traits or indicators.

The patterns to be matched are defined by a series of strings, regular expressions, and conditionals in YARA rules. File names, sizes, hashes, texts, entropy values, or specific byte sequences can all be used to generate rules. Furthermore, YARA supports the use of metadata and tags to offer context or information about the matched samples. YARA is designed to efficiently handle large datasets. It uses optimized algorithms for pattern matching and can process files in parallel, making it suitable for scanning repositories.

**Signature based antimalware solutions**

Malware detection commonly uses a signature-based approach: known malware is described by purely syntactic characteristics mostly bit strings or simple patterns defined by regular expressions which are stored in a signature database. This project focuses on Static Analysis as it is still a primary form of malware detection, thousands of enterprises also do not have the resources to invest in a robust dynamic analysis framework as it requires ever changing machine learning and trained models which in turn lead high computational overhead. However, the fact that you have a dynamic analysis implementation in place does not guarantee protection from zero-day malware, hence companies invest more in signature-based solutions which are affordable and get the job done. Signature-based detection has several shortcomings: Firstly, weak detection of obfuscation, secondly a malware signature only detects known malware, hence zero-day threats will go undetected. Regardless of these setbacks, signature-based malware detection is ideal for certain cases, where access to PDF is crucial, hence this project focuses on reinforcing the signature-based method with condition-based technique for analyzing PDF.

**Conclusion**

In this project, focuses on improving the static analysis of malware in PDF. We employ the use of a broad signature-based tool ClamAV combined with a broad rule set (YARA) to complement the signature scan by obfuscation and evasion detection. This tool eases the dilemma that is detecting malicious from legitimate PDF and preventing access to legitimate PDFs. Signature based analysis is indeed an efficient and quick way of detecting malware as it matches patterns to a known set of malicious entities i.e., the malware signature. However due to novel techniques and trickiness of threat actors, known malware signatures can go undetected, even with renown anti-virus of end point detection and Intrusion systems. Thus, we will employ a rule set that checks for signs of obfuscation, key words for remote calls, indicators of polymorphism. Hence this Project is a characteristic-based AV complimented with signature-based antivirus.

This is a decent and affordable solution as opposed to a cost unfriendly, time-consuming dynamic analysis which might not be feasible in an enterprise as users need quick access to PDF files.

# Bibliography

Bell, T. (1999). *The concept of dynamic analysis.* ACM SIGSOFT Software Engineering Notes.

D. Maiorca, D. A. (2015). *A structural and content-based approach for a precise and robust detection of malicious PDF files.* 1st Int’l Conf. Information Systems Security and Privacy.

I. Corona, D. M. (2014). Lux0R: Detection of malicious PDF-embedded Javascript code through discrim- inant analysis of API references. New York: Workshop on Artificial Intell. and Sec., ser. AISec ’14.

Karthik Selvaraj, N. F. (2010). *studylib.* Retrieved from the-rise-of-pdf-malware: https://studylib.net/doc/18609162/the-rise-of-pdf-malware

Liu, K. (2017). *The-Attack-Surface-Of-PDF-And-Gain-100-CVEs-In-1-Year-wp.pdf.* Retrieved from blackhat: https://www.blackhat.com/docs/asia-17/materials/asia-17-Liu-Dig-Into-The-Attack-Surface-Of-PDF-And-Gain-100-CVEs-In-1-Year-wp.pdf

M. Egele, T. S. (2012). *A survey on automated dynamic malware-analysis techniques and tools.* ACM Computing Surveys.

N. Nissim, Y. L. (2018). *Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining. Knowl.-Based Syst. 153, .*

Samociuk, D. (2023, January 2023). *Antivirus Evasion Methods in Modern Operating Systems*. Retrieved from MDPI: https://doi.org/10.3390/app13085083

Schläpfer, P. (2022, May). Retrieved from https://threatresearch.com: https://threatresearch.ext.hp.com/pdf-malware-is-not-yet-dead/

Sherly Abraham, I. C.-S. (n.d.). An overview of social engineering malware: Trends, tactics, and implications. *ScienceDirect*.

Srndic, P. L. (2011). Static detection of malicious javascript- bearing PDF documents. *27th Annual Computer Security Appli- cations Conf.*