Development of a Malware Scanner (Sweeper) for PDF Files

Oluwatosin FASANMI
Saajan MAHARJAN
Shiron DEV
Abdallah Ahmed Hamdan AHMED

EPITA Graduate School of Computer Science

Advisor
Mohammad-Salman Nadeem

July 12, 2023

Acknowledgements

This project is dedicated to our families for their support as we study overseas.

Abstract

Companies face significant challenges in mitigating malware through PDF files received via email attachments or uploaded on their front-facing file servers, particularly those used for CV applications. The primary problem lies in effectively identifying and neutralizing malware embedded in PDF files.

PDF files received via email can bypass initial security filters, leading to unsuspecting users opening infected attachments. Similarly, when PDF files are uploaded to front-facing file servers, they become accessible to multiple users, increasing the risk of malware spreading throughout the network.

Successful malware attacks can have severe consequences, including data breaches, financial losses, reputation damage, and legal ramifications. To address these challenges, organizations must develop effective strategies to reduce the possibility of cyber-attacks.

This project aims to propose improved approaches for detecting and mitigating malware in PDF files sent via email attachments or uploaded to front-facing file servers. By developing a tool to perform an extensive check on PDFS, provide the valuable insights and recommendations to enhance companies' security posture and protect critical assets from cyber threats.

Table of Contents

1. Introduction

1.1 Motivation

As the world transitions every day to a global village the widespread use of PDF as a standard format for document exchange has raised concerns about the security risks associated with PDF attachments. Threat actors often exploit this vector to propagate malware and compromise enterprise networks. The motivation behind this project is to develop an effective solution, named Sweeper, that safeguards enterprise internal networks from such PDF-based malware threats.

1.2 Context of the Project

The context of the project is scoped to enterprise networks with front facing file servers , email servers and any network which has file share capabilities. The context is to have Sweeper setup in strategic locations to scan files as they hit the network.

1.3 Objectives

The primary objective of this project is to develop and deploy Sweeper, an advanced scanning script, which utilizes state-of-the-art technologies to detect and isolate malware within PDF files. The specific objectives of the project are as follows:

Integrate the ClamAV antivirus engine to perform malware signature checks and leverage its extensive virus signature database.

Incorporation of YARA: Implement YARA rule-based scanning to detect and classify specific malware and threats within PDF files.

Object Extraction with pyPDF2: Utilize pyPDF2 to extract various objects and components from PDF files, enabling comprehensive analysis and identification of potentially malicious elements.

Seamless Workflow: Design a workflow that seamlessly integrates Sweeper into enterprise systems, allowing automatic scanning and detection upon file upload.

Real-time Reporting: Develop a front-end application that provides authenticated users with real-time scan results, enabling prompt action in case of malware detection.

1.4 Overview of the Thesis

This thesis presents the development and implementation of Sweeper, a powerful and effective scanning script for detecting malware within PDF files. Chapter 2 provides an in-depth review of related work, including existing malware detection techniques and approaches. Chapter 3 describes the methodology used in designing and developing Sweeper, highlighting the integration of ClamAV, YARA, and pyPDF2, as well as the implementation of the workflow. Chapter 4 presents the results and evaluation of Sweeper's performance, including its effectiveness in detecting malware and its impact on system performance. Finally, Chapter 5 concludes the thesis, summarizing the achievements, discussing the limitations, and proposing potential avenues for future enhancements and research in the field of PDF-based malware detection.

2. Literature Review

2.1 Introduction of Literature Review

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 in order 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, mainly 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 considerably, 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.

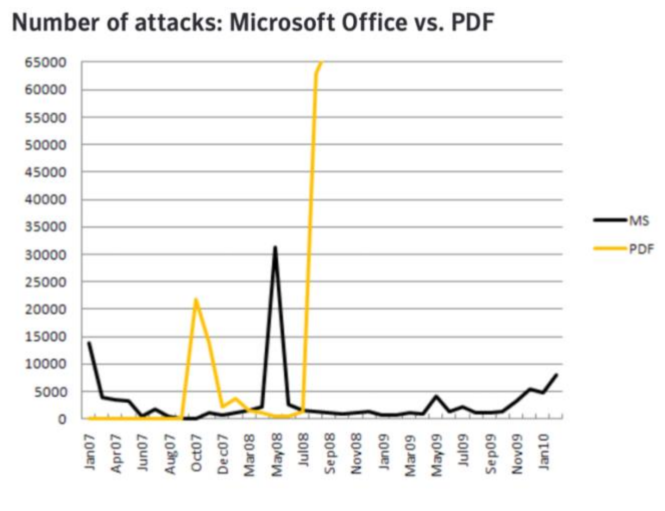


Figure 1

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 mainly adopt JavaScript code to exploit PDF vulnerabilities, by leveraging on well-known techniques, such as Return Oriented Programming and Heap Spraying

2.2 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 of 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 the 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 similar to 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 possible impact of malware on a machine or network It 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 however 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 actions—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 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 of supplementing signature matching with characteristic based analysis.

2.3 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.

2.4 Signature based malware 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 is affordable and for the most part gets 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 focus on reinforcing the signature-based method with condition-based technique for analyzing PDF.

2.5 Conclusion of Literature Review

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 compliment 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 signature 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.

3. State of the Art

3.1 YARA in the Modern Threat Landscape

The creation of AI and machine learning tools has been essential in identifying zero-day threats in the face of hostile actors' constantly developing evasion tactics. Organizations frequently face difficulties implementing such tools due to their cost and complexity. This is where YARA's rule set comes in handy, providing a cost-efficient and efficient method of detecting fresh malware without only depending on AI-driven solutions. To identify encrypted payloads, encrypted JavaScript, or encrypted files inside PDFs, YARA can be used to develop rules. This proactive approach ensures that encrypted malware is detected and thwarted, safeguarding your system from potential risks. A 2022 project by Reversing Labs (Labs, 2022) project gives a demonstration to utilize YARA to detect the 2021 GwisinLocker zero-day encryption malware, which was targeted at pharmaceutical companies in South Korea. Reversing Labs in 2023, have an effective YARA project to detect Lorenz ransomware which surfaced in 2021. Several months after its inception, the Lorenz operation had found a new attack avenue to exploit. Based on research from Arctic Wolf Labs (Wolf, 2022), the operation used a critical vulnerability in Mitel MiVoice VOIP appliances to breach enterprises. The vulnerability, CVE-2022-29499, is a remote code execution vulnerability that allows the threat actor to obtain a reverse shell, followed by using Chisel as a tunneling tool to pivot into the victim’s environment.

YARA can be leveraged to create rules that identify encrypted payloads, encrypted JavaScript, or encrypted files within PDFs. This proactive approach ensures that encrypted malware is detected and thwarted.

In their 2022 demonstration (Abrams, 2022), Abrams breaks down the Windows MoTW bypass zero-day flaw which was first reported by Hewlett-Packard. The zero-day vulnerability leverages a flaw in the Windows Shell, enabling threat actors to execute arbitrary JavaScript code without raising the usual security prompts or warnings. By exploiting this vulnerability, attackers can evade detection mechanisms and potentially compromise vulnerable systems. This loophole poses a significant risk as it bypasses the built-in security measures designed to protect users from executing potentially harmful scripts. YARA can effectively identify patterns which include specific function names, variable assignments e.g *wget* or *curl.* A rule can focus on specific functionality triggers that are commonly associated with the exploit of similar nature hence covering all related attacks based on a behavior match.

Despite the critical role that AI and machine learning tools play in identifying and reducing dangers, many organizations may find it difficult to afford and use it effectively. With the help of the YARA, enterprises may identify zero-day risks without having to exclusively rely on expensive AI-driven technologies. Large amounts of data can be handled effectively with YARA. It can quickly process files, making it suited for batch mode or real-time scanning of many PDF files. Additionally, the scanning process can be fine-tuned with its programmable rules, which minimizes false positives. To ensure precise detection, you can modify the criteria to match traits of malicious patterns.

Finally, YARA benefits from a collaborative community of security professionals who continuously contribute to its rule set. When a new exploit or vulnerability like the one described arises, security experts promptly develop and share YARA rules to detect and mitigate it. This collaborative approach ensures that organizations have access to the latest rules to defend against emerging threats.

3.2 CLAMAV in the Modern Threat Landscape

ClamAV is an open source (ClamAV, 2023) (GPLv2) anti-virus toolkit, designed especially for e-mail scanning on mail gateways. It provides a flexible and scalable multi-threaded daemon, a command line scanner and advanced tool for automatic database updates. The core of the package is an anti-virus engine available in a form of shared library. Real time protection (Linux only). The ClamOnAcc client for the ClamD scanning daemon provides on-access scanning on modern versions of Linux. This includes an optional capability to block file access until a file has been scanned on-access prevention. ClamAV's strength lies not only in its core capabilities but also in its timely signature updates. The ClamAV community, comprising security professionals and researchers worldwide, actively contributes to maintaining a vast database of malware signatures. This collaborative effort ensures that ClamAV users receive regular updates containing the latest detection patterns, effectively combating emerging threats, including obfuscated, JavaScript-based, and encrypted malware.

3.3 PYPDF2

The power of YARA is evident however, to utilize YARA to its full potential, we integrated our source code with PyPDF2. PyPDF2 is a pure-python PDF library that is free and open-source and can split, merge, crop, and alter the pages of PDF files. Additionally, it may give PDF files password protection, bespoke data, and viewing options. From PDFs, PyPDF2 can also extract text and metadata. In the Sweeper code, we utilize PyPDF2 to extract JavaScript and OpenAction objects of the PDF, parse these objects to YARA for a final check against the rule set. (PyPDF2, 2023)

The goal of the project Sweeper is to create an effective and accurate pdf scanner and provide a functional user interface on which strong technical controls to further reduce attack vectors. Sweepers front end allows a file submission form which checks the validates the file type, size and most importantly sanitizes the file name to prevent injection which could lead to path traversal on the host server.

3.4 Magic Module

The Sweeper web application uses the magic module (Daniel Mendler, 2023) to determine the MIME type of file content. By checking that the MIME type is "application/pdf," it ensures that the uploaded file truly contains PDF data. The application also extracts the file name without any directory components. This validation helps guard against potential attacks where an attacker may attempt to upload a file with a misleading extension and path traversal attacks, respectively.

3.5 Access Control Application

In Enterprise networks, there have been multiple issues of Access Control misconfiguration; most cyber-attacks are usually targeted at users with privileged access. Security administrators can be rest assured that once, files have been isolated, only authorized users will have access to the directory which contains the suspicious PDFs. With the implementation of a simple access control web application, which provides authenticated access via username and a salted bcrypt password hash. Credential hash authentication allows for granular Authentication, enabling privileged users to access the quarantine directory. This will prevent erroneous access to harmful files by other users on the network.

3.6 Conclusion of State of the Art

By incorporating ClamAV, a highly reputable and widely used antivirus engine, Sweeper benefits from its extensive virus signature database and sophisticated scanning algorithms. This integration ensures on time access to a broad and updated database, to identify known viruses, and other malicious elements within PDF files.

Furthermore, the integration of YARA, a powerful pattern matching tool, enhances the effectiveness and versatility of your script. YARA enables the creation of custom rules and signatures that can be tailored to specific threats or targeted attacks. YARA supports regular expressions, which are powerful tools for pattern matching. Regular expressions enable the detection of complex obfuscation patterns, such as character substitutions or encoding schemes. YARA rules can be easily shared and reused, fostering collaboration within the security community. There is a vast collection of pre-existing YARA rules available, covering a wide range of obfuscation techniques and other security-related patterns.

The inclusion of pyPDF2 for object extraction adds another layer of functionality and feasibility. By leveraging the capabilities of pyPDF2, Sweepers extracts PDF objects from PDF files, facilitating a comprehensive analysis of their contents. This enables deeper inspection of potentially malicious or suspicious elements within the files, contributing to the overall robustness of the script.

Leveraging the power of ClamAV, YARA, and pyPDF2, demonstrates impressive strengths, robustness, and feasibility in creating a lightweight malware scanner for PDF files. Its utilization of state-of-the-art technology, combined with an open-source approach, ensures effectiveness, adaptability, and transparency. In addition, the implementation of the access control prevents unauthenticated access to the isolated files.

4. System Architecture

4.1 Workflow

- File upload from the webapp
- File validation done on the client-side.
- Sweeper scanner script is set as a job on the server.
- The Scan is triggered when a new file is uploaded.
- The script connects to the ClamD to start the clam scan for malware signature check.
- If there is a positive signature check, the file is moved to quarantine.
- The YARA check follows the ClamAV, if there is a positive match It moves the file to quarantine.
- The file is isolated on detection of a malware signature or YARA condition match.
- The scan result is then displayed on the front-end app to authenticated users.

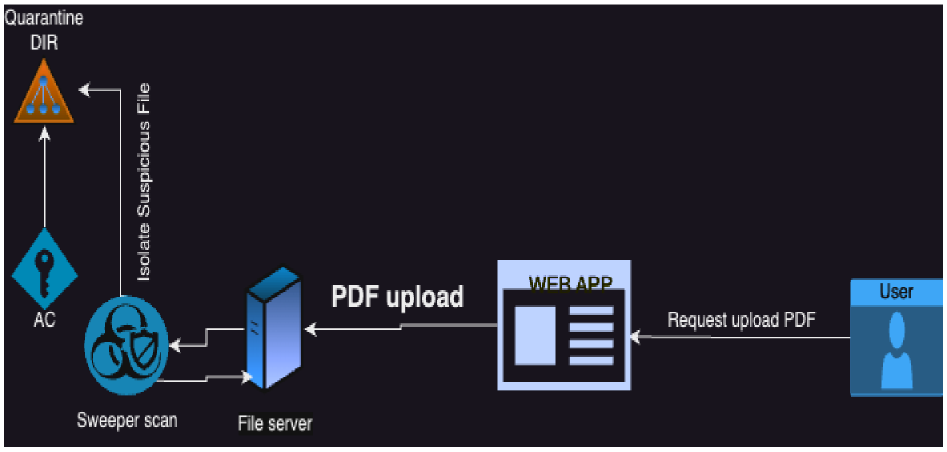


Figure 2: Workflow

4.2 Technologies

4.2.1 Linux based host server (kali 2023.01)

Kali Linux is a popular Linux distribution specifically designed for penetration testing and security auditing. It provides a wide range of pre-installed security tools, making it an ideal choice for your project. Kali Linux offers a stable and secure environment, enabling you to perform various security-related tasks efficiently.

4.2.2 Python3

Python is a programming language known for its simplicity as it is a high-level programming language. Python has tons of libraries and packages makes it an excellent choice for developing security-related scripts. We chose Python over Bash because Python's syntax is more expressive and intuitive, making it easier to write, understand and troubleshoot. It seamlessly interacted with system commands, external tools(PDFid), and worked perfectly for connecting to ClamD and invoking YARA check. Python has built-in error handling mechanisms, such as try-except blocks, which allow you to handle exceptions gracefully and ensure the script continues running even if errors occur.

4.2.3 Yara: 4.2.3

YARA's capability to create and modify rules makes it well-suited for the project, as it is very customizable.

4.2.4 ClamAV: 1.0.1

ClamAV 1.0.1 is the latest version with full support for all features. Integrating ClamAV into the project allows for extensive signature database and advanced scanning algorithms, enhancing the overall security of the scanning process.

4.2.5 Flask Web Framework: 2.3.3

Flask is a lightweight and flexible web framework for Python. It allows you to build web applications quickly and efficiently, providing essential features for handling HTTP requests and responses. Flask's simplicity makes it an excellent choice for developing the web application for the PDF upload. Its modular structure and support for various extensions enable seamless integration with other components of your project.

4.2.6 MAGIC Module MIME Checker

The "magic" module, also known as python-magic, is a Python interface to the libmagic library. It allows to identify the type of a file by analyzing its contents, based on magic number signatures and other file characteristics.

It was used to perform MIME type checking client-side to prevent the application from accepting any file that is not a PDF. This adds an extra layer of validation and verification to ensure the integrity of the files being scanned.

4.2.7 PYPDF2: 3.1.0

PYPDF2 is used in the pdf_analysis function of the code to, extract and parse the OpenAction and JavaScript objects to the yara_check function.

5. Methodology

5.1 Client-Side Validation

The PDF files are downloaded to a server from a website which has technical controls in place to perform file validation by verifying that the extension is .pdf. If the file extension matches, then a file size check by a client-side validation using JavaScript or HTML5 file input attributes in addition to a server-side validation. The ideal server handling implementation is to create a secure directory that stores files.

The project is concerned with the directory where the PDFs are stored. Sweeper runs in the background and is triggered immediately a file is uploaded to the server. It first scans the file against the ClamAV signature base then checks against YARA ruleset. The script runs as a logical 'OR' check and on identification of any signature or YARA condition, the file is isolated in a directory which access is limited to only authorized IT security personnel.

Client-side validation is implemented using JavaScript to ensure that the uploaded file meets the specified criteria before submitting the form to the server. Where client send the file by form, which is used to collect data, precisely upload file It uses a form POST method which specifies the form is used to send data to web server.

When the user selects a file using the file input field, the validateForm() function is triggered when the form is submitted.

The validateForm() function performs the following validations:

File extension validation: It checks if the selected file has a .pdf extension using the regular expression (\.pdf)$/i. If the file's extension does not match this pattern, an error message is displayed.

File size validation: It checks if the size of the selected file exceeds the specified limit of 5 MB (5 * 1024 * 1024 bytes). If the file size is larger than the limit, an error message is displayed.

If any of the validations fail, the corresponding error message is displayed in the <span> element with the id="error-message". The error message is shown in red. If all the validations pass, the form is submitted, and the file is uploaded to the server. The return true statement in the validateForm() function allows the form submission to proceed. If any validation fails, the return false statement prevents the form from being submitted. When the user submits the form, the validateForm() function is called to check if the uploaded file meets certain criteria. If the file meets the criteria, the form data is sent to the server at the specified URL ('upload' route). If the file doesn't meet the criteria, an error message may be displayed, and the form is not submitted. The client-side validation provides immediate response to the user, preventing them from submitting invalid files and reducing unnecessary server-side processing. It ensures that the uploaded file has a PDF extension and is within the specified size limit before proceeding with the file upload.

5.2 PDF Structure

To effectively handle malware propagation via PDF files, it is important to understand the structure of a PDF file. A PDF document is a collection of objects which describe how one or more pages must be displayed.

In general, a PDF document consists of four main parts.

- One-line header or Header
- Body
- Cross-reference table
- Trailer

The header identifies that it is PDF, specifying the PDF file format version, the trailer points to the cross-reference table (starting at byte position 642 into the file), and the cross-reference table points to each object (1 to 7) in the file (byte positions 12 through 518).

The objects are ordered in the file: 1, 2, 3, 4, 5, 6 and 7.

A PDF file's logical structure is hierarchical in nature, and the trailer identifies the root object. As shown in Figure 3, item 1 is the root, and objects 2 and 3 are its children.

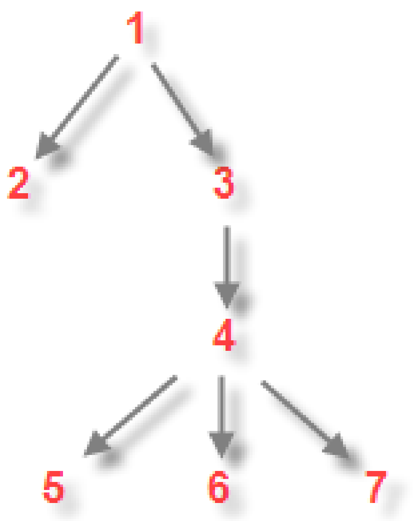
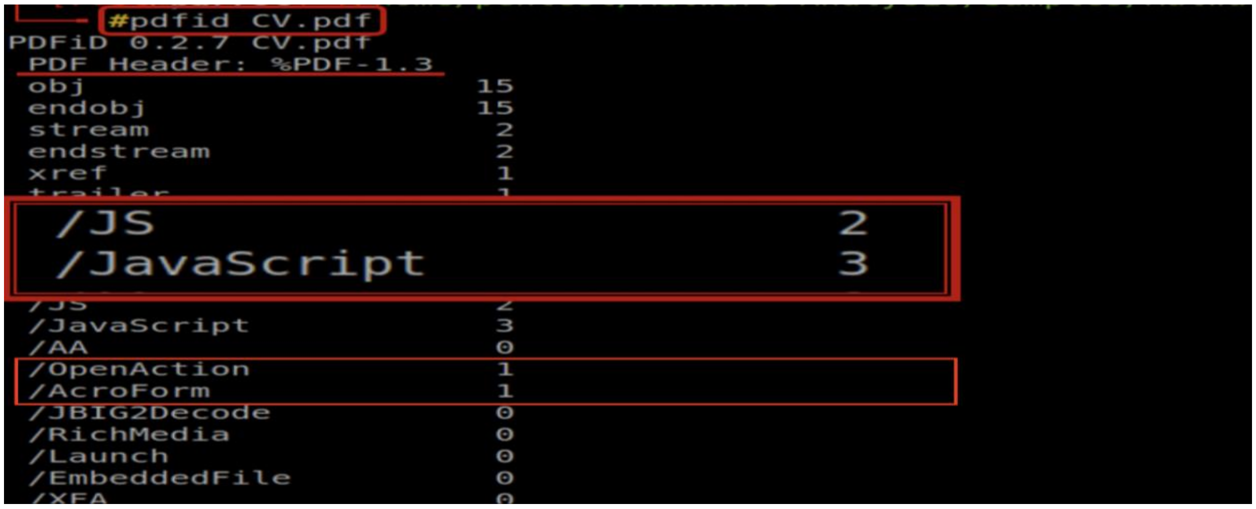


Figure 3

The symbols /JS and /JavaScript denote the presence of JavaScript in the PDF file. JavaScript is used in almost all malicious PDF files we discovered (to perform a heap spray and/or attack a JavaScript vulnerability). Of course, JavaScript can also be found in PDF files without being used maliciously. When a page or document is viewed, an automatic action will be performed as indicated by the options /AA and /OpenAction..



```
#pdfid CV.pdf
PDFiD 0.2.7 CV.pdf
 PDF Header: %PDF-1.3
 obj                    15
 endobj                 15
 stream                  2
 endstream               2
 xref                    1
 trailer                 1
  /JS                                  2
  /JavaScript                          3
 /JS                     2
 /JavaScript             3
 /AA                     0
 /OpenAction             1
 /AcroForm               1
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /EmbeddedFile           0
 /XFA                    0
```

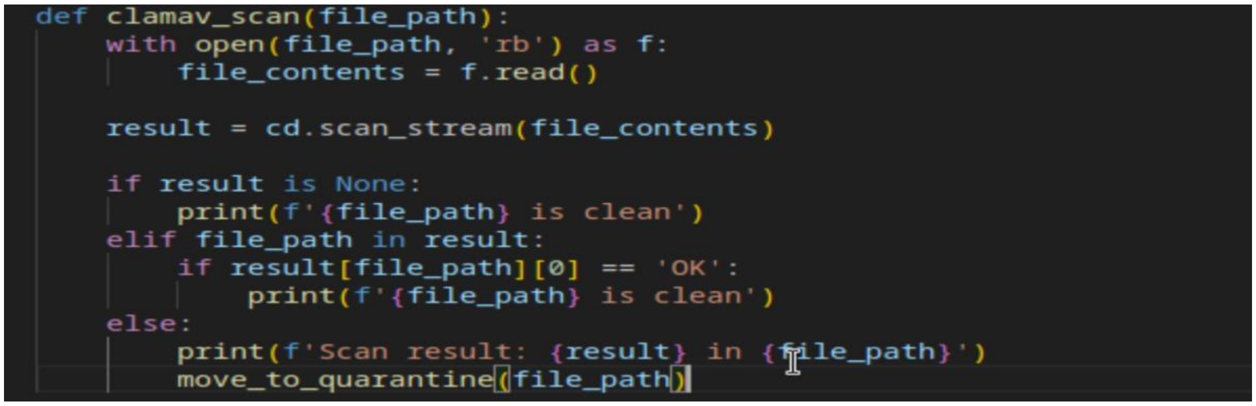
Figure 4

5.3 ClamAV Implementation and Functions

ClamAV and its dependencies were first configured on the Linux server. ClamAV can be configured to use a network socket (UnixSocket) or a daemon "ClamD". Sweeper uses the ClamD because it enables improved resource management by keeping the ClamAV engine loaded in memory and eliminating the need to load the engine for each request.

ClamD (ClamAV Daemon) is a distinct process that functions as a background service and offers a more effective and scalable method of scanning files. Instead of communicating with the ClamAV engine directly using Unix sockets. The host machine is configured to listen for clam calls, which will be initiated by Sweeper and handled ClamD. Sweeper invokes ClamAV daemon with pyclamd.ClamdUnixSocket(), which creates an instance of ClamdUnixSocket from the pyclamd library. This instance represents a connection to the ClamAV daemon using Unix socket communication, after which it loads the signature database into memory.

ClamAV verifies the PDF file's general structure to make sure it complies with the PDF specification. It searches for any anomalies or differences that might point to potential problems. It analyzes each object in the content stream (the PDF's actual content) as well as its connections to other objects. It confirms that there are no errors and that the structure is valid. It compares the output of the scan to its signature base the CLAM VIRUS DATABASE (CVD). It compares them against the signatures in the signature stream. It performs pattern matching algorithms to detect if any part of the file matches the patterns in the signature stream. If a match is found, it indicates the presence of a known malware threat. The clamav_scan function in the Sweeper source code as shown in Figure 4.



```python
def clamav_scan(file_path):
    with open(file_path, 'rb') as f:
        file_contents = f.read()

    result = cd.scan_stream(file_contents)

    if result is None:
        print(f'{file_path} is clean')
    elif file_path in result:
        if result[file_path][0] == 'OK':
            print(f'{file_path} is clean')
    else:
        print(f'Scan result: {result} in {file_path}')
        move_to_quarantine(file_path)
```

Figure 5

In the ClamAV scan, when a file is infected, the result dictionary returned by cd.scan_stream contains the infected file as a key, and the value associated with that key is a tuple representing the infection result. The first element of the tuple is the infection status, such as 'FOUND', 'ERROR', etc.

If file_path exists in the result dictionary, it further checks if the first element of the list associated with file_path is equal to 'OK'. If it is, it prints a message indicating that the file is clean. If the first element of the list is not equal to 'OK', it means the file is infected with a specific detection. The code prints a message indicating the file is infected with the detection and calls the move_to_quarantine function to move the infected file to the quarantine directory.

5.4 YARA Rule set

5.4.1 How It works

Yara rules generally works through matching the strings patterns and detecting suspicious activity within that file or data. The structure of the rule with using the proper format for the following strings , condition along with meta data will represent the specific content and pattern searching the pdf file. e.g. – including strings based on java script injected strings will help if the pdf has any malicious or doubtful content within it. So the method of Yara rule help classify and analyze the pdf files to identify the threat level of an infected pdf files.

Firstly, we install YARA and YARA to Python binder on the host machine, i.e the kali linux. YARA to Python is the official Python binding for YARA, which enables interaction with YARA using Python code. This enables YARA to be imported in the Sweeper code.

5.4.2 What the rule checks for in a pdf

The techniques used in the project mainly targeted finding suspicious threats, embedded content, and JavaScript within the files.

Strings:

- $magic = { 25 50 44 46 }
  - The rule represents the magic byte of the file which is the identification type of the file. So the rule contains the format byte number if the " %PDF" also the signature of the file
- $attrib = /\/Launch/
  - This variable or signature represents the launch function within the pdf files. The rule contains this string to find any suspicious or irrelevant launch action inject within the pdf file. Successful string match will be considered for further analysis protocol of the files.
- $attrib = /\/URL /
  - The attribute signature is applied to find any link or phishing link might be provided or attached to the object which will direct to the threat sites.

$reg0 = /trailer\r?\n?.{1,7}\/Size.{,5}\r?\n?\.{1,7}/
$reg1 = /\/Root.{,5}\r?\n?.{x,10}startxref\r?\n?.{x,9}*\r?\n?%%EOF/

- This rule set specifically written to identify the validity including the root object and startxref offset within pdf
- By using the root and trailer character and including the optional link breaks " (\r?) or (\r?\n?)" the pattern will check the verification of patterns on trailer and root structure for any potential modification to it.

$js = /\/JavaScript/

- The variable used in the string define to identify the potential javascript codewithin the pdf file.
- This string will code including the finding the correct java version within the pdf document structure.
- Checks for the JavaScript which is outside the range of the pdf version and then provides the potential warning when the signatures were found.

$embed = /\/EmbeddedFiles/

- Rule is to find any embedded objects within the pdf files.
- Purpose of this rule is to find the embedded contains the pdf document, which is not the correct version of the pdf, so the rule specifies by adding the second rule the version check " $ver = /%PDF-1\.[3-9]/"

$UUID V4 = {d8b94d86-5d88-46aa-86a8-9717f1dc23a8}

- This detects the following strings in UUID `$\():`

Conditions :

1. $magic in (0..1024)
    - Condition checks the magic number values within the header which matches "%PDF" signature. The magic number values condition should be met within the first 1024 bytes
    - Used for the verification of the PDF file format.

2. $magic in (0..1024) and $js and not $ver
    - Continuation of this rule add up to the string's headers "$js" and "$ver" and uses the "and" and "not" operation.
    - First it checks for the version of file through magic byte.'
    - Uses the "and" operation to include to find and JavaScript content within the pdf file.
    - Add the operation "not" to check and request to find if the version content is within correct numbers if not the result find and gives a threat indication.

3. $magic in (0..1024) and $embed and not $ver
    - Same as like the previous rule to find JavaScript within pdf the only difference is the rule is applied to find embedded content within pdf file.
    - Check with whether the version of the file which has embedded content is within out range of the pdf version.

Yara rule will be differ its function most of the time depends on the scope of the project and the type of the content the rule is written for, which explains that strings and conditions should be matched accordingly to the type of rule and the signatures written on it.
The research will continue identifying more rule patterns and difference of malware signatures found within the pdf file. Grateful of the research guidance from the official Yara rule

documentation (YARA, 2023) and the official open source github (Rules, 2022)repository of a collection of YARA rules.

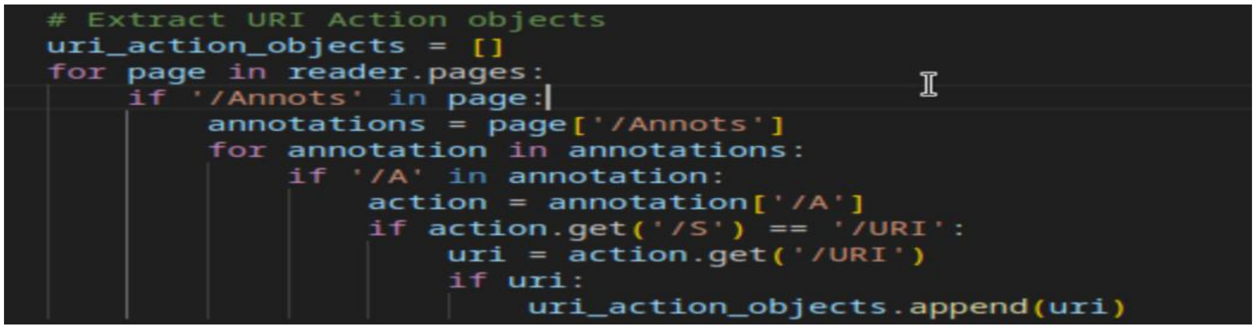
5.5 Object Extraction with PYPDF2

The holistic approach for parsing in PYPDF2 through the PdfReader. PdfReader is the PDF file parser used by pypdf. The fundamental structure of parsing is represented by the method PdfReader.read

In the Sweeper code the file is opened in Read and Binary mode (rb mode), When a file is opened in binary mode ('rb'), it allows reading the file's content as raw bytes instead of text. This is useful for handling binary data, as malware can be written in binary to evade detection from antivirus.

The PyPDF2.PdfReader(file) creates a PdfReader object from the opened file.The function then extracts JavaScript objects from each page of the PDF (PyPDF2, 2023). It iterates over the reader.pages collection and checks if a page contains '/JavaScript'. If it does, the JavaScript object is appended to the js_objects list. Same procedure is done for the OpenAction objects. And extracted objects are stored in the `openaction_objects` The reader.pages collection is the sequence of pages in the PDF.

Lastly, seeing the spike in malware being executed by enabling a remote connection to a C2 server, it is imperative to check for URI within the Action Objects, hence the Sweeper code performs a granular check by recusing through the Action Object of each page in each PDF, as seen in Figure 6.



```
# Extract URI Action objects
uri_action_objects = []
for page in reader.pages:
    if '/Annots' in page:
        annotations = page['/Annots']
        for annotation in annotations:
            if '/A' in annotation:
                action = annotation['/A']
                if action.get('/S') == '/URI':
                    uri = action.get('/URI')
                    if uri:
                        uri_action_objects.append(uri)
```

Figure 6

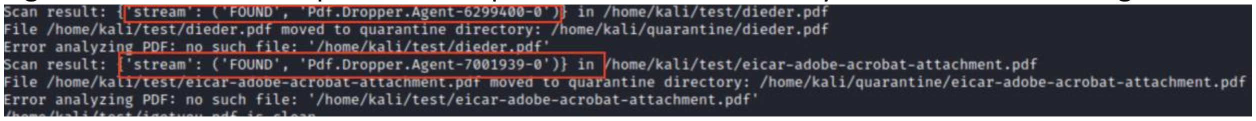
For each page in reader.pages, it checks if the page contains '/Annots' (annotations). If annotations exist, it iterates over each annotation and checks if it contains '/A' (action). If an action is present and has '/S' (subtype) equal to '/URI', it retrieves the URI value and appends it to the `uri_action_objects` list. All 3 lists are combined and passed as a parameter to the YARA function to compare against the rule set.

6. Results

6.1 Findings

Sweeper runs as expected and makes detections as specified with all its combined components.

The ClamAV database is as broad as it boasts and accurately detects and matches the malware signatures in the dieder.pdf and eicar.pdf which are files used by researchers for testing.



```
Scan result: {'stream': ('FOUND', 'Pdf.Dropper.Agent-6299400-0')} in /home/kali/test/dieder.pdf
File /home/kali/test/dieder.pdf moved to quarantine directory: /home/kali/quarantine/dieder.pdf
Error analyzing PDF: no such file: '/home/kali/test/dieder.pdf'
Scan result: {'stream': ('FOUND', 'Pdf.Dropper.Agent-7001939-0')} in /home/kali/test/eicar-adobe-acrobat-attachment.pdf
File /home/kali/test/eicar-adobe-acrobat-attachment.pdf moved to quarantine directory: /home/kali/quarantine/eicar-adobe-acrobat-attachment.pdf
Error analyzing PDF: no such file: '/home/kali/test/eicar-adobe-acrobat-attachment.pdf'
```

Figure 7

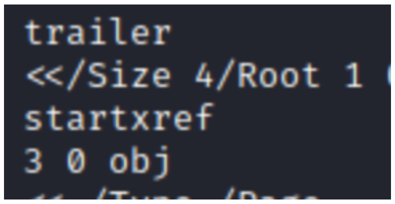
The PYPDF2 extracts the objects as a list. We are considering making the EmbeddedFile object a special case for the PYPDF2 analysis as the EbeddedFile Object cannot be parsed to the YARA function. Hence if a file has an EmbeddedFile it will be isolated for further analysis.

6.2 Challenges and Resolutions

The clamunix socket was unreliable, it kept tripping off after each scan. Hence, we used the clamd instead.

Choosing the right PDF parser to extract the Objects was a challenge, we first used the pymupdf library, but could not properly extract the objects. PDFID by Dieder was also attempted however integrating into the Sweeper code was difficult as it had to run as a sub-process because it is a stand-alone tool and not a python library. Finally, PYPDF2 successfully extacted URI objects and we proceeded with it.

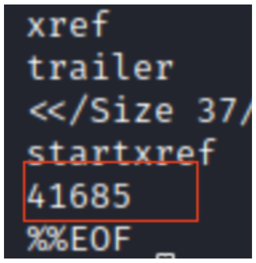
Another challenge related to extracting the objects is the mode in which the pdf was created, some pdfs are not set with an EOF marker, i.e End of File to specify the end of the document, similarly some PDFs do not have the startxref specified, PYPDF2 sees such files as broken and cannot read them. This is an ongoing challenge as also YARA cannot completely parse such PDFs, however, we have set a YARA rule to flag invalid objects structure as this in most cases is the case in a malicious PDF based on experience.



```
trailer
<</Size 4/Root 1
startxref
3 0 obj
```

Figure 8

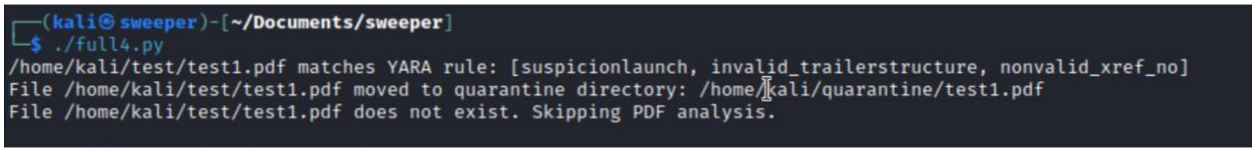
In Figure 8, The startxref line is missing the actual offset value. The startxref keyword should be followed by the byte offset indicating the starting position of the cross-reference table in the PDF file. In Figure 9, we can see the xref number that contains information about the location and structure of objects within the file.



```
xref
trailer
<</Size 37/
startxref
41685
%%EOF
```

Figure 9

Finally, in Figure 10 below we use specified a YARA rule that checks for malformed objects in the PDF, as this is a sign of malicious intent.



```
┌──(kali㉿sweeper)-[~/Documents/sweeper]
└─$ ./full4.py
/home/kali/test/test1.pdf matches YARA rule: [suspicionlaunch, invalid_trailerstructure, nonvalid_xref_no]
File /home/kali/test/test1.pdf moved to quarantine directory: /home/kali/quarantine/test1.pdf
File /home/kali/test/test1.pdf does not exist. Skipping PDF analysis.
```

Figure 10

For the front-end there is the ongoing challenge of displaying the scan result to the end user, as the web page is a static and currently displays a directory list of the infected files. We are working on making it dynamic so a user can select files and rescan by interfacing with the webapp, this feature does not affect the Sweeper code in any way, however it is a value adding factor as it makes our project more marketable, and this can be updated in the future.

7. Conclusion

Sweeper is a project to curb the success rate of malware propagation via PDF in enterprise networks. The project has proven to be as effective as anticipated thanks to the combination of the vast CLAMAV signature base, the PYPDF object extraction feature, YARA rule for detecting various patterns which can be misused for malicious intent. It is lightweight, simple to integrate in an existing environment and offers defense-in-depth protection. This tool is essential for all companies because they interact PDF documents from various external sources, and it can be integrated on all front facing file servers. In the future more CLAMAV plugins can be integrated with Sweeper as well as an updated YARA rule. To improve administration, a feature on the access control web app to rescan and review all scan logs is better to prevent interreacting directly with a command line interface.

Bibliography

Abrams, L. (2022). *Exploited Windows zero-day lets JavaScript files bypass security warnings*. Retrieved from Bleeping Computer: https://www.bleepingcomputer.com/news/security/exploited-windows-zero-day-lets-javascript-files-bypass-security-warnings/

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

ClamAV. (2023). Retrieved from ClamAV: https://docs.clamav.net/Introduction.html

ClamAV. (2023). Retrieved from ClamAV: https://docs.clamav.net/Introduction.html

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.

Daniel Mendler, J. W. (2023). Retrieved from https://github.com/mimemagicrb/mimemagic#

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

Labs, R. (2022, 11). Retrieved from From the Labs: YARA Rule for Detecting GwisinLocker: https://www.reversinglabs.com/from-the-labs/yara-rule-for-detecting-gwisinlocker

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, .*

PyPDF2. (2023). *How pypdf parses PDF files*. Retrieved from readthedocs.io: https://pypdf2.readthedocs.io/en/stable/dev/pypdf-parsing.html

Rules, Y. (2022). Retrieved from https://github.com/Yara-Rules/rules

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

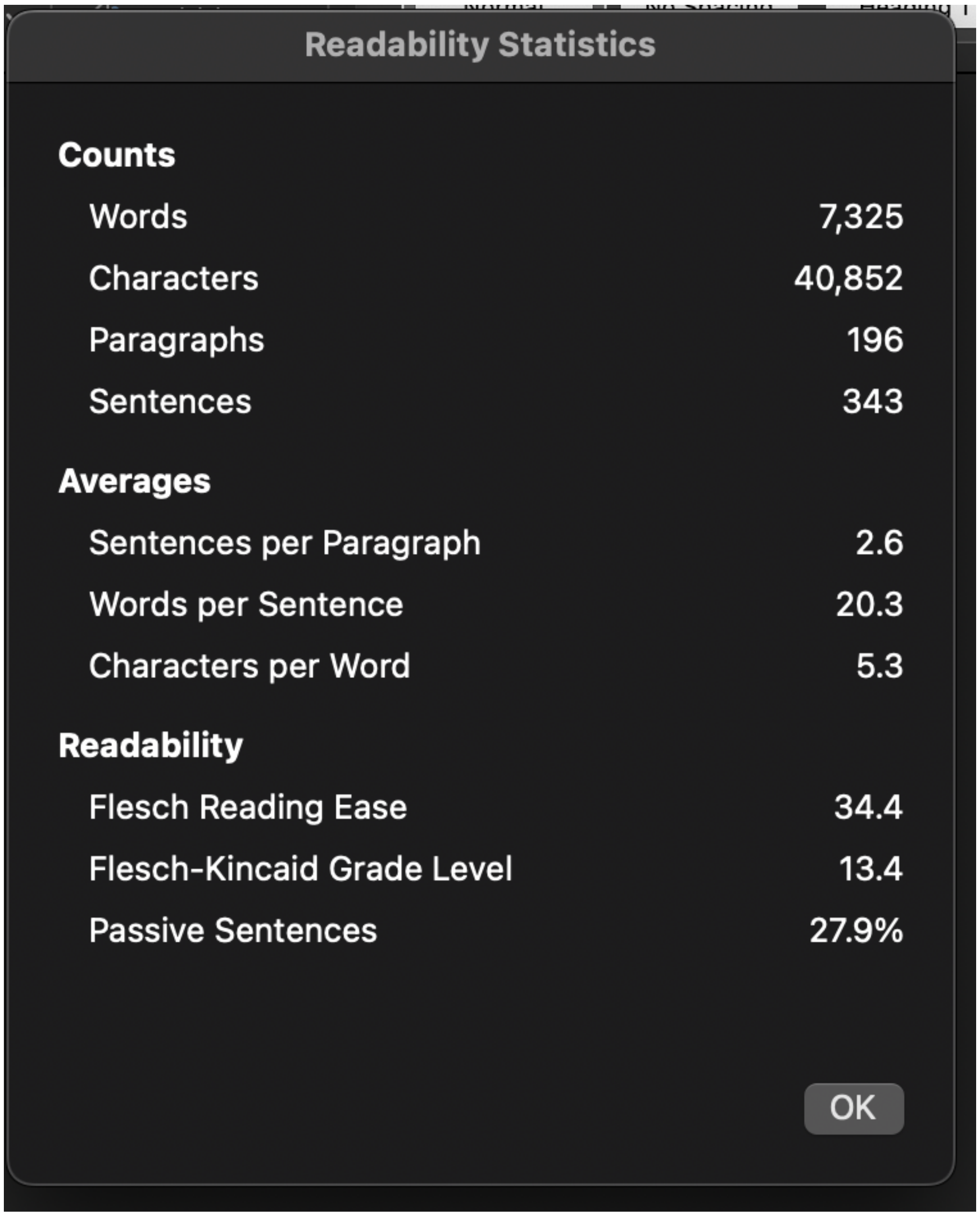
Schläpfer, P. (2022, May). Retrieved from https://threatresearch.com: https://threatresearch.ext.hp.com/pdf-malware-is-not-yet-dead/

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.*

Wolf, A. (2022, 09). *Chiseling In: Lorenz Ransomware Group Cracks MiVoice And Calls Back For Free*. Retrieved from Chiseling In: Lorenz Ransomware Group Cracks MiVoice And Calls Back For Free: https://arcticwolf.com/resources/blog/lorenz-ransomware-chiseling-in/

YARA. (2023). Retrieved from https://yara.readthedocs.io/en/stable/ : https://yara.readthedocs.io/en/stable/


Readability Statistics

Counts

Words	7,325
Characters	40,852
Paragraphs	196
Sentences	343

Averages

Sentences per Paragraph	2.6
Words per Sentence	20.3
Characters per Word	5.3

Readability

Flesch Reading Ease	34.4
Flesch-Kincaid Grade Level	13.4
Passive Sentences	27.9%