IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 23 March 2023, accepted 13 April 2023, date of publication 17 April 2023, date of current version 21 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3267717

RESEARCH ARTICLE

Content Disarm and Reconstruction of PDF Files

RAN DUBIN, (Member, IEEE)
Department of Computer Science, Ariel University, Ariel 4070000, Israel
Ariel Cyber Innovation Center, Ariel University, Ariel 4070000, Israel
e-mail: rand@ariel.ac.il

ABSTRACT Content Disarm and Reconstruction (CDR) is a zero-trust file methodology that proactively extracts threat attack vectors from documents and media files. While extensive literature on CDR emphasizes its importance, a detailed discussion of how the CDR process works, its effectiveness, and its drawbacks is not presented. Therefore, this paper presents PdfCDR, the first PDF CDR system in which the validation, the prevention rate, and the received visual similarity effect of disarming and reconstruction are presented and measured. Furthermore, PdfCDR suggests for the first time a novel method dealing with new emerging exploits by automatically converting detection rules to disarm and reconstruction rules. As a result, PdfCDR can prevent evasive attacks without any software upgrades and utilize the cyber security community knowledge to prevent cyber attacks as soon as they are advertised. The effectiveness of the novel PdfCDR against well-known PDF datasets shows that it disarmed not only the malicious components, but the reconstructed file is also usable and functional. However, since CDR relies on understanding the file format, any CDR solution should handle each supported file type separately due to the vast difference in each file format. Hence, this paper focuses on the Portable Document Format (PDF) file type that attackers commonly exploit. The results indicate that PdfCDR successfully CDR 90% of the malicious files while the remaining 10% were encrypted or had abnormal structures compared to the standard and were quarantined.

INDEX TERMS Adobe PDF, attack prevention, CDR, malware, sensitization, threat disarm, zero-trust.

I. INTRODUCTION

In recent years, Portable Document Format (PDF) positioning is a standard for document exchange and dissemination. PDF [1] is flexible, easy to customize, and portable across platforms. However, the characteristics of PDF motivated hackers to exploit various types of vulnerabilities and file features to overcome security safeguards [2], [3]. It was reported [4] that about 30% of malicious web downloads in 2020 were delivered through documents such as Rich Text Format (RTF), Portable Document Format (PDF), Microsoft Office Word, Excel, and PowerPoint [5]. Furthermore, a report based on Checkpoint threat cloud data [6] collected between January 1st and December 31st 2020, indicates that PDF is the third most common malicious file type responsible for 8% of the file-based attacks coming from Web browsing.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet.

Several detection analysis techniques have been proposed in the literature that use signature-based detection [7], behavioral heuristic [8] (dynamic), sandboxing (dynamic) [9], and Artificial Intelligence (AI) [10]. All these methods are detection mechanisms that rely on file detection or file behavior detection, meaning file/ behavior trust. However, it is still possible that evasive malware will not be detected. For example, a recent Snake Keylogger attack used a malicious PDF with an embedded Microsoft Office DocX file to lure the user into clicking on the malicious embedded object [11] and evade detection. Recent antivirus detection tests show that the best solution achieves a 96.3% online detection rate [12], which is insufficient. Therefore, a zero-trust prevention layer is needed.

Content Disarm and Reconstruction (CDR) [13], [14], [15], [16], [17] is a zero-trust file security that is currently gaining more popularity in Industrial Control Systems (ICS), file upload protection, and email security. CDR's ability to disarm and reconstruct relies on understanding the file format specification and being aware of its weakness. Therefore, CDR should handle each supported file type separately and validate the effectiveness. The CDR utilizes in-depth understanding of the file structure to break down any file type into its discrete components. The CDR strips away anything that does not conform to the file type's original specification, International Organization for Standardization (ISO) standard, company policy, and after rebuilds a "clean" version that has a high safety level against zero/one-day attacks.

CDR works by following the file format specification and sanitizing/ disarming or removing all attack vectors found in the file [3] and increasing user security. An attack vector, or threat vector, is a way for attackers to enter a network or cause a denial-of-service (DoS) [18], [19]. Common attack vectors include social engineering attacks [20], credential theft [21], vulnerability exploits [22], and insufficient protection against insider threats [23]. A significant part of information security is closing off attack vectors whenever possible. In the scope of our work, PDF may contain lure images with instructions to execute the attack, hyperlinks, active components, and exploits. All the attack vectors must be handled to secure the users. Further information about PDF attack vectors is found in the Appendix Section.

CDR is not a new solution in the cybersecurity industry and is mainly used in Industrial Control Systems (ICS) that cannot rely only on antivirus detection [12] and need the most accurate prevention. However, CDR is not limited to ICS, and the technology adaptation has been increasing and is expected to be very significant in the upcoming years in the IT sectors [24]. Furthermore, Gartner's analysts [24] identify CDR technology as particularly useful where files cross organizational boundaries on platforms such as email, web, and file content-sharing sites. Therefore, our goal is to increase the research depth in the CDR area.

CDR has many advantages, such as zero-trust, speed, and risk reduction, and it can be added as another defense level before or after the Antivirus (AV) /sandbox detection. However, some inherent limitations exist with CDR, such as:

- a decrease in document usability caused by the removal of active code and functionality
- inappropriate handling of signed documents
- failure to disarm an encrypted document (unknown password)
- limited understanding of how the evasive malware attack vectors work (new zero-day or new attack vectors) [25]
- not suitable for disarming code files, executable, and scripts. This is because disarming will affect those files' ability to operate. However, documents and multimedia files could disarm safely due to their defined structure and lack of dependencies between the different file sections.

In light of the above, we can conclude that CDR is not a silver bullet: however, it can significantly reduce risks since it does not rely on antivirus signatures or AI detection, which malware can bypass [26], [27], [28].

Though there is vast literature on CDR (e.g., [13], [14], [16]) that emphasizes the need for the technology at a high level, it does not discuss how the CDR process works in detail, does not demonstrate the method's effectiveness, and does not present the drawbacks in terms of usability and user-perceived visible quality. These details are essential for adopting the technology and increasing the research in this area.

Closing this gap is this paper's main novelty, which expands the in-depth research by measuring the effectiveness of the prevention and exploring the limitations of CDR. This paper is not about protecting the organization from attacks due to failed sanitization/privacy of files going from the organization or malware threat detection. Instead, this paper is focused on zero-trust prevention by file CDR. To the best of our knowledge, this is the first work on the PDF file type that proposes a research methodology, validates file zero-trust disarm effectiveness, and evaluates the disarm effect on the user-perceived visible quality/similarity to the original file.

The PDF file type is commonly exploited. For example, there are hundreds of Common Vulnerabilities and Exposures (CVE) for PDF readers and almost 300 known vulnerabilities for Adobe Acrobat Reader alone [29]. A significant novelty of this work is the ability to create disarm and reconstruction rules from detection rules. Detection rules such as Yara [7] are used by researchers to stop emerging threats and CVEs. The rules are shared by social media and other Internet channels once a threat is discovered [30]. By automatically using detection rules and converting them to disarm and reconstruction rules, we stop evasive threats as soon as they are discovered. As a result, we do not need to wait for expensive and slow software updates, and prevent them as soon as an attack is discovered. The above gave us the motivation to start with the PdfCDR system.

The rest of this paper is organized as follows. First, section II describes the related works. Section III presents the PDF structure and summarizes the different attack vectors in the PDF file. Section IV offers the PdfCDR solution architecture, and Section VI provides our dataset. In Section V, we present our research evaluation and validation methodology. Section VII discusses our disarm prevention results and reconstruction measured image similarity. Then in Section VIII, we present PdfCDR limitations and future work. Finally, Section IX provides our conclusions and directions for future work. Detailed information about our disarm and reconstruction steps are found in the Appendix Section.

II. RELATED WORK

CDR technologies are typically deployed at a network edge where their primary focus is to extract and disarm threats in the file. In this section, we begin by reviewing the general works related to CDR (section II-A) and then those related to PDF CDR (section II-B).

A. GENERAL CDR RELATED WORK

Sim et al. [14] and Sunshine et al. [16] present the need for CDR for different uses but do not discuss how to build and validate the technology and do not present its effectiveness.

Han et al. [15] present CDR technology for PDF and Microsoft Office files by saving the original document and converting it to a JPEG image file. This method provides a 100% prevention rate, but the usability and the file format, which is changed to an image, make the document impossible to edit and work on. Furthermore, their methods rely on screen capturing and should be done in an isolated and safe environment since the document is opened and may contain evasive malware.

Wiseman [13] presents the value of CDR combined with Content Threat Removal (CTR) that also acts as a Data Leakage Protection (DLP) by preventing unseen sensitive information from leaking out. However, the author does not discuss how this is done and does not provide algorithms, datasets, methodology, or prevention rate, and doesn't discuss the CDR similarity changes effect.

General CDR solutions lack up-to-date open-source projects, free cloud solutions, and prevention benchmarks of industry-grade solutions. Open-source projects exist, such as DocBleach [31], a Java tool for CDR that supports Microsoft Office, RTF, PDF, and Zip Archive. A recent initiative [32] suggested a proof of concept for CDR file protection that supports Excel, PowerPoint, Word, PDF, and several image file formats. However, it has limited PDF disarm support and focuses only on JavaScript and embedded objects. Currently, both projects are archived and outdated. Furthermore, the effectiveness of zero-trust disarm technology was not validated, and the effects on the image similarity to the original document after CDR were not measured. Our previous work suggested a novel CDR solution for RTF files [22] and described how to prevent malicious attack vectors in detail. However, this work is the first CDR work that provides a solution for updating CDR engines from detection rules shared in the community.

B. PDF CDR

Many works in the past have focused on malicious PDF file detection [3], [26], [33], [34], [35], [36], [37], [38], [39]. Others focus on specific PDF attack vectors such as Malicious URL detection [40], and PDF JavaScript detection [41]. However, all the above are trust-based solutions where the anomaly, static signature, or AI can detect the malicious file. However, this work is zero-trust prevention and not detection. Our goal is to disarm any threat in any file without prior knowledge if the file is malicious. The same disarm and reconstruction actions are always done on every file.

Others focus on privacy content in the file or sanitization [42], [43], [44], [45], [46]. Close related works [42], [47] focus on hidden data detection in PDF that can be used to collect threat intelligence on the entity that created the file. The extracted knowledge, such as printer names [47], internal domain names, operating systems, and personal information can be used against the organization. However, those did not focus on disarming the attacks in the PDF file and focus on removing sensitive data from files going out of the organization.

The NSA [48] reviewed the attack vector focused on the PDF. Inspired by the NSA guidelines, this work is the first work that suggests a zero-trust disarm and reconstruction method, validates the prevention rate (disarming malware), usability (reconstruction), and measures the document's visual changes caused by the CDR based on well-known image visual similarity metrics.

The full description of our CDR steps is found in the Appendix Section. Our proposed solution is different from all previous related works because it enables the user to work on the same file type as the original after the disarming process while ensuring that the user can continue to work on it. In other words, the disarmed document is similar to the original one in our solution, but it is safe to use.

We do not trust the detection ability to detect and stop the malware. Therefore, our zero-trust focus is on attack prevention by always disarming and reconstructing the file regardless of whether it is malicious.

III. BACKGROUND

As mentioned above, this paper focuses on the PDF file type. Consequently, we first explain the PDF file type and survey the different attack vectors for PDF files before we present our novel CDR architecture. Note that, because CDR systems rely on understanding the file format, any CDR solution should handle each supported file type separately.

In general, malicious attacks could be concealed in the different file components such as scripts, images, videos, embedded objects, forms, and links, which illustrates the standard generic discrete components that should be disarmed. It is important to note that some objects should be disarmed recursively since they contain nested objects. Nested objects are objects embedded in a file, whereas each can consist of additional embedded objects. For example, a PDF can have an embedded object, such as a Word document containing an additional embedded object (nested) with a malicious macro. CDR should aim to remove the parent object or recursively remove all possible child objects that are threats. The complexity and running time of the CDR increase when handling all nested objects compared to removing the root object.

PDF is a versatile file format, created by Adobe in 1992. PDF gives people an easy, reliable way to present and exchange documents regardless of the software, hardware, or operating systems used by anyone who views the document. PDF was standardized as ISO 32000 in 2008 [49] the last edition as ISO 32000-2:2020, also named PDF 2.0 [50].

The PDF file structure comprises four sections: the header, the body, the cross-reference table, and the trailer. Figure 1 illustrates the PDF representation and basic contents that exist in each section, while Table 1 presents the list of abbreviations for important PDF suspicious and evasive content.

TABLE 1. PDF common keywords in relation to malware list of abbreviations. For the full list please review the standard [51].

Abbreviation	Meaning
%pdf	Indication of the start of the PDF file. After that, there is an indication of the PDF version.
obj	Indication of indirect object.
endobj	Indication of indirect end of object.
endstream	Indication of stream object end.
cross-reference table	Containing information about the indirect object in the file.
xref	Each cross-reference section shall begin with a line containing the keyword xref.
trailer	Giving the location of the cross-reference table and certain special objects within the file's body.
startxref	The value following the startxref keyword shall be the offset of the cross-reference stream.
/Encrypt	Indicates that the PDF document has DRM or needs a password to be read.
/ObjStm	Object stream can contain other objects. Malware authors use it to obfuscate objects (by using different filters to bypass detection).
/JS	Execute a JavaScript script.
/JavaScript	Execute a JavaScript script.
/OpenAction	Indicates an automatic action when the page/document is viewed. Usually followed with JavaScript.
/AA	Indicates an automatic action when the page/document is viewed. Usually followed with JavaScript.
/JBIG2Decode	Decompresses data encoded using the JBIG2 standard.
/FlateDecode	Decompresses data encoded using the zlib/deflate compression method.
/RichMedia	Used for embedding flash object.
/Launch	Launch an application, usually to open a file.
/URI	Resolve a Uniform Resource Identifier (URI).
/XFA	Indicates XML Form.
%%EOF	End of File marker.

PDF docs can contain links and buttons, form fields, audio, video, and business logic. They can be signed electronically and easily viewed on Windows or macOS using the free Adobe Acrobat Reader software. PDF can include hidden and potentially dangerous data while still appearing as legitimate-looking documents. As a result, it is essential to investigate the PDF file format and standard for data hiding and disclosing vulnerabilities.

The first line of a PDF (Header) file specifies the document's version number of the used PDF specification. The following section is the body of the PDF document, and some objects typically include text streams, images, and other multimedia elements. The body section holds all the document's data being shown to the user. The cross-reference (Xref) table contains the references to all the objects in the document. A cross-reference table allows random access to objects in the file, so we do not need to read the whole PDF document to locate the particular object. In the example above, in the first line, we can see that the first number in those lines corresponds to the object number, while the second line (column in that line) states the number of objects in the current subsection (in this example 271 objects).

The following line presents each object and is represented by one entry, 20 bytes long (including the CRLF). The first

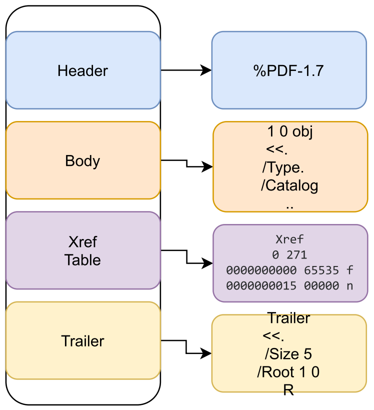


FIGURE 1. PDF file format.

10 bytes are the object's offset from the start of the PDF document to the beginning of that object. Then, a space separator with another number specifying the object's generation number follows. After that, another space separator is followed by a letter "f" or "n" to indicate whether the object is free or in use.

The PDF trailer specifies how the application reading the PDF document should find the cross-reference table and other special objects. All PDF readers should start reading a PDF from the end of the file. The last line of the PDF document contains the end of the "%%EOF" file string. Before the end of the file tag, a line with a \startxref string specifies the offset from the beginning of the file to the cross-reference table. The PDF structure understanding is important and can discover different anomalies that can be a sign of malicious activity [34].

NSA provides a list of hidden data, and embedded content that may be found in PDF files [48]. Based on NSA recommendations, our reverse engineering research, known threat research, and the standard, we established PdfCDR disarm capabilities. Table 2 summarizes our recommended disarm and reconstruction steps for the different PDF attack vectors and their effect on the user. The full detailed description per attack vector can be found in the Appendix Section. After all the attack vectors are handled and detection signatures are removed, the PDF file will be safe from malware.

IV. PdfCDR ARCHITECTURE

In this section, we present the PdfCDR system architecture. The user can upload a file using HTTP Rest Application Programming Interface (API). The file or files will be uploaded to PdfCDR backend architecture illustrated in Fig.2.

In step (1), the user selects the folder/file to disarm using the desktop application. Then, using Rest API, step (2), the backend application receives the files and inserts them into the RabbitMQ [52] queue containers, step (3), one for the detection worker container (4) and one for the PdfCDR worker container (6). The detection container has three tasks: PDFiD, Yara, and VirusTotal reputation (all are described in Section V). The output of this step is written in the shared volume log file (5) per incoming file and will be used to

TABLE 2. Summary of PDF attack vectors, disarm steps, and their effect on the user.

#	Attack Vector	Recommended Action	The Effect
1	Metadata	Sanitize (remove) metadata content fields.	Reduces the risk without affecting the image quality since the data is not visible. There is no effect on the document's visual appearance.
2	Code Comments	Remove code comments.	Since the data is not visible, there is no effect on the document's visual appearance.
3	Links	Remove links	Possible effect on the document's visual appearance.
4	Dynamic Resources and Scripts	Remove JavaScript and ActionScript.	Possible effect on the document's visual appearance.
5	Embedded Content	Recursively remove embedded content.	Possible effect on the document's visual appearance.
6	Encryption	Encryption at this moment is not handled.	The encrypted file will be marked as unsafe, and we plan to extend the support in future work.
7	Multimedia and Image Content	The solution replaces the multimedia and image-embedded object with re-encoded and harmless content (not in the scope of this work).	Possible effect on the document's visual appearance.
8	Annotations	Remove annotations.	Possible effect on the document's visual appearance.
9	Triggers and Actions	Remove all Triggers and Actions.	Possible effect on the document's visual appearance.
10	Acrobat Forms - Acro-Forms	Remove AcroForms.	Possible effect on the document's visual appearance.
11	Acrobat Forms- Adobe XML Forms Architecture (XFA) Form	Remove XFA.	Possible effect on the document's visual appearance.
12	PDF Functions	Remove the function dictionary and related data streams and related references to the function dictionary.	Possible effect on the document's visual appearance.

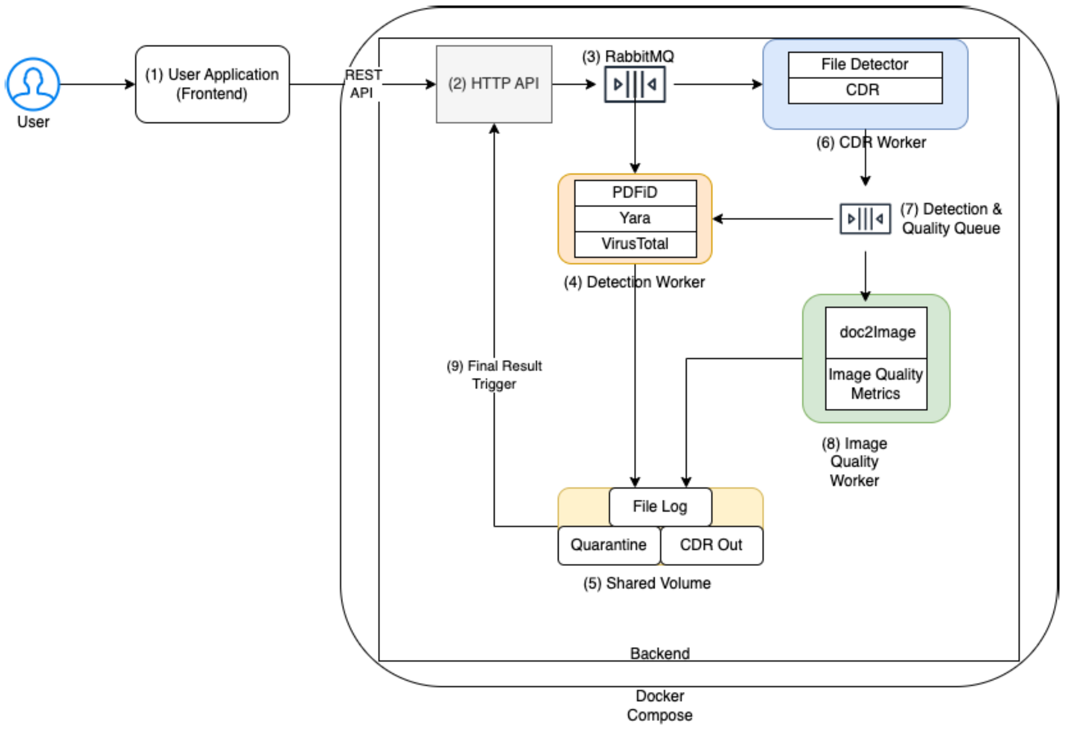


FIGURE 2. PdfCDR general architecture overview.

compare the results at the end of the analysis. The PdfCDR worker (6) receives the file, detects its actual file type, and applies the CDR methodology to the file. Since each file type is different, there is a need for a CDR worker to use the correct disarm function per file type. Any file that fails to be disarmed is Zip compressed with password protection and moved to the quarantine folder in the shared volume. The PdfCDR steps are discussed in the Appendix Section and Fig. 3.

Files successfully disarmed are sent to the detection and quality queues (7). One queue is for the detection after the CDR process, and one queue is for the image quality comparison (8). Finally, the detection docker worker pulls a file from the detection and quality queue (7) and verifies the success of the disarming process. If the file is found to be malicious, it is Zip compressed with password protection and moved to the quarantine folder in the shared volume (5). If the file is secure, meaning the disarming was successful, the file is transferred to the CDR out folder located in the shared volume (5).

The image quality worker (8) pulls the file from the queue (7) and fetches the original file (not depicted in Fig. 2 for simplicity reasons). The first pages of the original and the CDR constructed file are converted to a JPEG file. Then the image quality metrics are calculated and stored in the shared volume, as explained in the Results section. After the file analysis is complete, the application receives (9) the status. It receives the log with all the disarm and reconstruction steps and the detection status (4 before and after PdfCDR to validate the correctness of the proposed solution, including the image quality metrics (8), which were calculated for the file.

The research and validation methodology presented in Fig. 2 is essential since it validates the file's capabilities before PdfCDR and after comparing the original file similarity to the PdfCDR output. In addition, Yara validation and VirusTotal ensure the file is safe for the user. As far as we know, this is the first work that validates the correctness and safety of CDR, and none of the previous work benchmarks the disarm and reconstruction rate, which limits the adoption of CDR solutions.

Fig. 3 illustrates the PdfCDR steps from detection rules and is designed to enhance the CDR's ability to remove new exploits and attacks unrelated to the file structure and focus on vulnerabilities in the file reader. For example, Cisco Talos [53] recently detected four Foxit Reader PDF viewer exploits with high severity that could lead to arbitrary code execution. In those attacks, a specially crafted PDF document can trigger the reuse of previously freed memory, leading to arbitrary code execution. An attacker must trick a user into opening a malicious file to trigger these vulnerabilities.

The steps illustrated in Fig. 3 are additional steps inside the CDR worker (step 6) in Fig. 2. The PdfCDR is done after the CDR worker steps (defined in the Appendix Section) are done and enhance PdfCDR to disarm the PDF from detection capabilities against file exploits that are detected or will be detected in the future.

The CDR worker receives the input file (1) (step (1) in Fig. 3), and the File Detector(2) is responsible for detecting the file type of the file and making sure it is supported and has a valid structure file type. The CDR Yara Engine (3) first checks for new and updated signatures (4); if it is updated, it runs each Yara rule against the file. Next, the CDR Yara engine (5) return offsets for the detected rules in the original file. In the next phase, we map each detected rule offset in the file and map it to the correct PDF file object. Then we create Disarm rules from the detected objects, which can be seen in step (6), and, based on the defined policy (7), we decide per PDF object type what the disarm rules and reconstruction rules for the object and the file are. Based on this step, we are disarming the detection rules(8). Finally, in step (9), we create reconstruction rules, also affected by the user policy (7), and enforce the reconstruction (10). The result is a clean, disarmed file that is usable, editable, and similar to the original source file, as we will demonstrate in the Evaluation Result Section VII.

The novel contribution of Fig. 3 is the ability to leverage the malware research community, social media, and commercial security companies' fast detection and response [54]. Yara rules are the de facto standard for newly detected vulnerability detection sharing. By using Yara detection rules and converting them to disarm and reconstruction rules, PdfCDR can stop threats as soon as they are discovered without code updates. Product code update takes much more time to release and install in the organization, while Yara rules in our solution work similarly to automated anti-virus signature updates, which are fast and straightforward.

V. EVALUATION METHODOLOGY

The malware research community uses well-known open-source tools and commercial solutions to research and validate if a file is malicious. Our evaluation methodology illustrated in Fig. 2 describes the entire flow from receiving the file to the resulting clean CDR output of PdfCDR. In this section, we explain the different evaluation components we are using.

First, we describe Fig. 2 Detection worker (4) components: Yara [7] engine in SectionV-A, PDFiD [55] in Section V-B, and VirusTotal [56]. The Detection worker is used on the original document and the output of PdfCDR.

Then we turn to describe our visual similarity metrics in Section V-D that are used in Fig. 2 in image worker (8). The similarity engine receives the document pages, converts them to images per input file, and estimates the average similarity between the matching input and output document image pages.

A. YARA

Yara [7] is a static tool that can help malware researchers to identify and classify malware samples and emphasize their abilities. Yara provides a flexible language for creating data signatures. Yara is used extensively in the industry and the research community and enables descriptions of malware families and behavioral attributions. Many open-source rules exist; we found that the following provides good visibility of the detected behaviors [57], [58], [59], [60], [61] for PDF files. The Yara signatures include common CVEs and common attacks, and suspicious behaviors. The final file we used is available in our GitHub repo [62].

B. PDFiD

PDFiD [55] is a Python module to detect specific PDF keywords, allowing researchers to identify PDF documents that contain interesting behavior that may be exploited, such as JavaScript, or execute an action when opened. PDFiD can handle name obfuscation. PDFiD is a good validation tool that helps develop and verify evasive objects' existence and enables a simple illustration of the file's capabilities. Fig. 4 illustrates a malicious file PDFiD analysis. We can observe that the file has two pages containing OpenAction

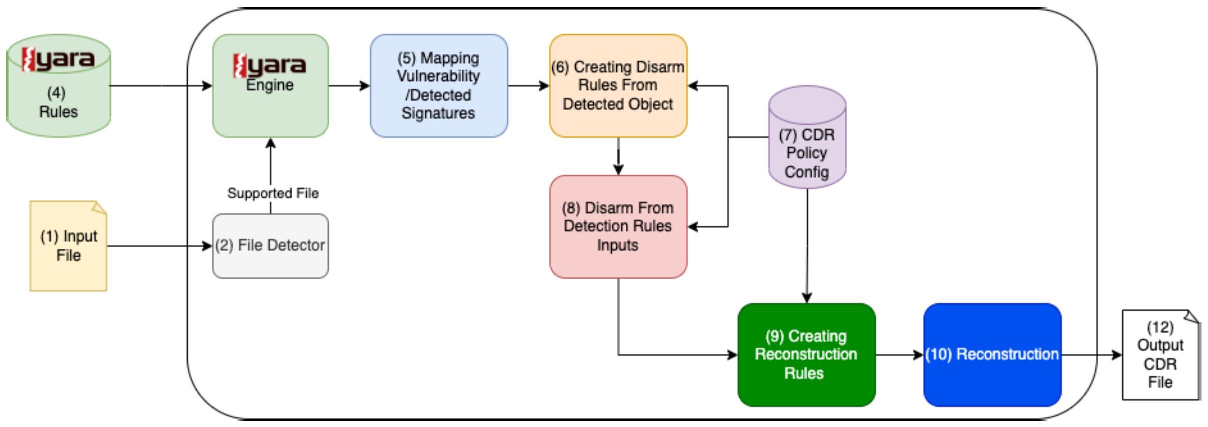

yara
(4) Rules
yara Engine
(5) Mapping Vulnerability /Detected Signatures
(6) Creating Disarm Rules From Detected Object
(7) CDR Policy Config
(8) Disarm From Detection Rules Inputs
Supported File
(1) Input File
(2) File Detector
(9) Creating Reconstruction Rules
(10) Reconstruction
(12) Output CDR File

FIGURE 3. PdfCDR disarm and reconstruction steps from detection rules architecture overview.

and JavaScript, a common attack combination that alerts researchers that this document may be malicious. In addition, the file comprises AcroFrom and XFA, which are known to hide malicious intent. However, PDFiD is a visibility tool and not a detection tool. As a result, we will regard it in the result section as removed objects visibility that helps to learn more about the datasets.

C. VirusTotal

VirusTotal [56] combines the knowledge of many antivirus vendors (static scans), online scan engines, and dynamic analysis (malware sandbox). The service continuously updates and currently contains more than 74 antivirus vendors. We used VirusTotal to verify the correctness of PdfCDR.

D. IMAGE SIMILARITY

PDF disarm changes the fonts and images and removes active and evasive content. The question is how much this change affects the user-perceived quality/similarity. We saved the original (reference) and the disarmed (modified) documents as JPEG images to validate this. The disarming process may cause an essential loss of information or quality. Image similarity evaluation methods can be subdivided into objective and subjective methods [63].

Subjective methods are based on human judgment and operate without reference to explicit criteria [64], and they are out of the scope of this work. On the other hand, objective methods are based on comparisons using explicit numerical criteria [65] and a ground truth image. In this work, we explore three objective methods: Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR) [66], and Structural Similarity Index Measure (SSIM) [66]. In our comparison, we use the original document's first-page image and its equivalent disarm PDF image from the first page to calculate the objective criteria. All images are scaled to the same size MxN. MSE is calculated as follows where f is the original image, and g is the disarmed image.

$$MSE(f,g) = \frac{1}{MN}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(f(i,j) - g(i,j))^2 \tag{1}$$

The PSNR is calculated as follows:

$$PSNR(f,g) = 10\log_{10}(255^2/MSE(f,g)) \quad [dB] \tag{2}$$

The PSNR value approaches infinity as the MSE approaches zero; this shows that a higher PSNR value provides a higher image similarity. On the other hand, a small value of the PSNR emphasizes high numerical differences between images. Please note that for identical images, we will receive ∞.

SSIM is used to measure the similarity between two images, and it is correlated with the quality perception of the Human Visual System (HVS). SSIM can model the image distortion based on three factors: loss of correlation, luminance distortion, and contrast distortion, as can be seen: 3.

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g) \tag{3}$$

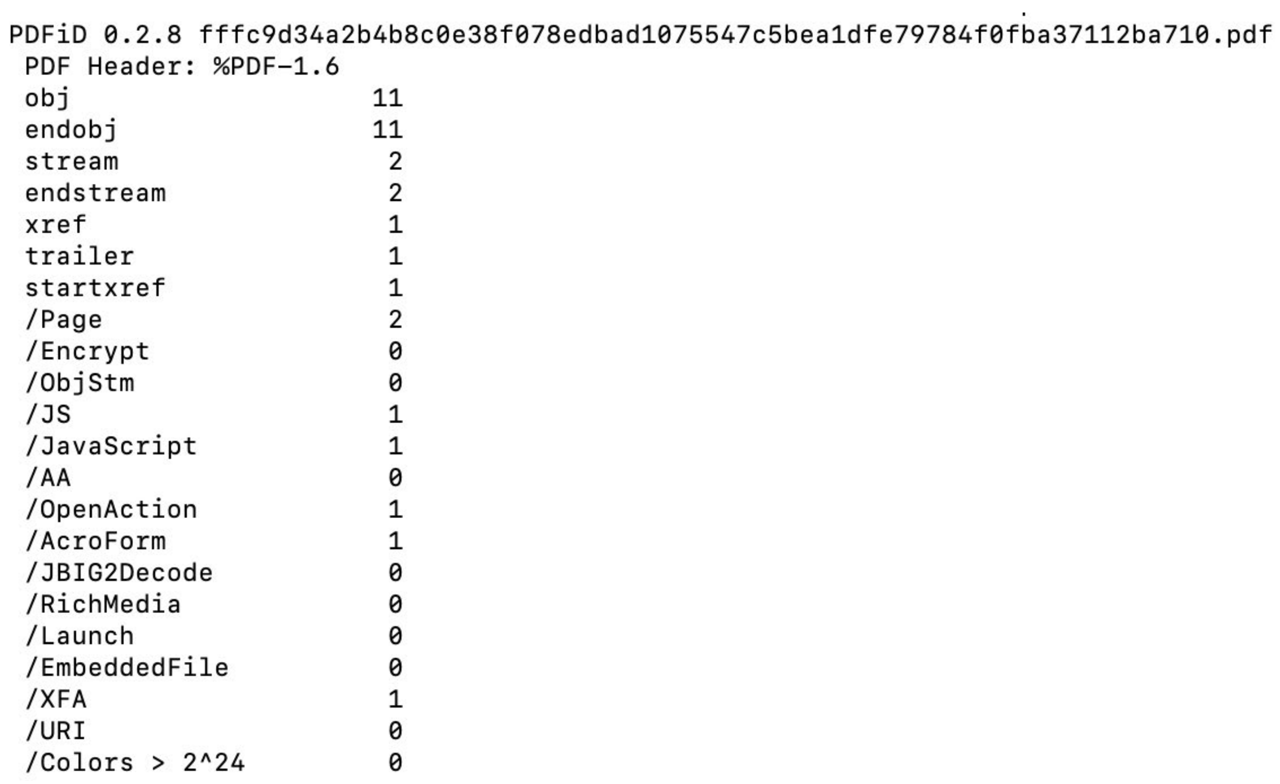
where the terms $l(f,g)$, $c(f,g)$, $s(f,g)$ appear in Eq. 4, the first term describes the luminance comparison function that expresses the closeness of the two images; when $\mu_f = \mu_g$ the maximum value will be 1. The second term describes the closeness of the contrast of the two images; when $\sigma_f = \sigma_g$ the maximal value will be 1. The last term measures the correlation coefficient between the two images; the positive constants C_1, C_2, and C_3 are used to avoid a null denominator.

$$\begin{aligned} l(f,g) &= \frac{(2\mu_f\mu_g + C_1)}{\mu_f^2\mu_g^2 + C_1} \\ c(f,g) &= \frac{(2\sigma_f\sigma_g + C_2)}{\sigma_f^2\sigma_g^2 + C_2} \\ s(f,g) &= \frac{(\sigma_f g + C_3)}{\sigma_f\sigma_g + C_3} \end{aligned} \tag{4}$$

VI. DATASET

We used two datasets to validate PdfCDR disarm capabilities and measure the reconstruction usability and similarity to the original file:

- The benign pdf files - LOC [67] which contains 1002 PDF benign/clean files,
- The malicious PDF files - VirusTotal dataset for academia was collected between 2018-2021 and



```
PDFiD 0.2.8 fffc9d34a2b4b8c0e38f078edbad1075547c5bea1dfe79784f0fba37112ba710.pdf
 PDF Header: %PDF-1.6
 obj                     11
 endobj                  11
 stream                   2
 endstream                2
 xref                     1
 trailer                  1
 startxref                1
 /Page                    2
 /Encrypt                 0
 /ObjStm                  0
 /JS                      1
 /JavaScript              1
 /AA                      0
 /OpenAction              1
 /AcroForm                1
 /JBIG2Decode             0
 /RichMedia               0
 /Launch                  0
 /EmbeddedFile            0
 /XFA                     1
 /URI                     0
 /Colors > 2^24           0
```

FIGURE 4. PDFiD analysis of object indicators for md5: 35ab21dd2409385f154ba2caaadf3bdc.

contained 11246 malicious PDF files provided by Google VirusTotal service [56].

Please note that VirusTotal is available for academic researchers upon request. Please refer to our GitHub [62] repository for hash and further information.

In this section, we will review the VirusTotal dataset detected threat types in Table 3 and review the differences between the content of the two datasets using PDFiD in Table 4.

Table 3 illustrates the common threats in the VirusTotal dataset. The dataset contains 373 different threat types. We decided to group all threats from the malicious dataset with fewer than four detected samples as threats named "Other" in the table for simplification purposes. The malicious dataset covers most of the PDF malicious file behaviors described in the Appendix Section.

From Table 4 we can observe the PDFiD object comparison between the VirusTotal and LOC datasets. The Norm VirusTotal column is the normalized quantities since the dataset sizes are imbalanced, which helps to emphasize the object differences between the malicious and benign datasets. We can observe that the number of pages in legitimate files on average is 14 in this LOC dataset, while in the malicious files, most of the PDF files have a single page. We can observe that JS, JavaScript, and OpenAction are much higher and more common in the VirusTotal dataset and are usually connected. The EmbeddedFiles, AcroForm/XFA, and URI are much more prevalent in malware, although they are used in legitimate files as well.

VII. EVALUATION RESULTS

This section is structured as follows: the Yara detection results are described in Section VII-A. We present VirusTotal detection results in Section VII-B, and PdfCDR run time analysis in Section VII-C, and we conclude the visual changes PdfCDR causes in Section VII-D based on image similarity metrics.

A. YARA RULES RESULTS

In this section, we will evaluate the LOC and VirusTotal datasets with the detection capabilities of Yara and VirusTotal multi-antiviruses and malware sandboxes. Table 5 illustrates the Yara rule detection rate before and after PdfCDR on the LOC dataset. We found that Yara detected 19 different types of signatures before our PdfCDR. The most common Yara rules show that 24.85% had a suspicious attack indicator, meaning some active component exists in the file. This indication is not necessarily malicious but shows that the dataset contains abilities that can be used for malicious activity. A surprising fact is that 16.68% of the LOC dataset before PdfCDR presents an invalid PDF structure. Our research found that deviations in the structure are common in legitimate PDFs. We can see that 2% of the data contain OpenAction activity. When the document is open, the action is executed. OpenAction is a known attack vector in malicious PDFs but is also used legitimately in many files.

TABLE 3. VirusTotal detected threat name histogram. We show only threats that have four samples and more. The full description is found on our GitHub.

Threat label	Count	Threat Label	Count	Threat Label	Count
Other	394	trojan.pdfka pidex	31	trojan.phish1	7
trojan.pdfka pidief	3265	trojan.expl pdfka	27	trojan.expl pidief	6
trojan.pdfka payload	2831	dropper.pidief smeo	22	trojan.pdfka smei	6
trojan.pidief pdfka	1314	trojan.pidief name	21	phishing.stealer	6
trojan.pdfka name	491	trojan.pdfka bloodhound	19	trojan.malphish	6
trojan.pdfka expl	289	trojan.swrort meterpreter	18	trojan.exploit	6
phishing.phishingx stealer	220	trojan.pidief pidief1	17	trojan.tiff pdfka	6
virus.eicar test	215	trojan.expl	16	trojan.pdfka tiny	6
phishing.phishingx	214	trojan.pidief tiff	15	trojan.pidief iframebof	6
trojan.pdfka tiff	184	trojan.pdfka utoti	14	trojan.fraud	6
trojan.pdfka crypted	181	trojan.payload pidief	13	pdfka expl	6
trojan.pdfka pdfjsc	127	trojan.pidief payload	13	trojan.pdfka blackhole	5
trojan.	95	trojan.name pidief	12	trojan.blacole pidief	5
trojan.pdfka shellcode	92	trojan.pdfka blacole	12	trojan.pdfka aaaaa	5
trojan.swrort cryptz	74	trojan.pidief pdfjsc	12	trojan.name iframebof	5
trojan.name pdfka	72	trojan.expl cve20102883	12	trojan.pidief pidrop	5
trojan.payload pdfka	68	trojan.pidief jmub	12	trojan.pdfka smiv	5
trojan.redir includer	61	trojan.pdfka nsanti	11	trojan.shellcode cve100188	4
trojan.pidief	58	phishing.pidief	11	tiff/pidief	4
trojan.pidief expl	54	trojan.scam	11	phishing.goriphish	4
trojan.goriphish	51	trojan.expl shellcode	11	trojan.blacoleref	4
trojan.pdfka pdfex	47	trojan.pdfka oneofchild	10	trojan.shellcode pidief	4
trojan.pidief shellcode	46	trojan.pdfka exploit	10	trojan.blacole pdfka	4
phishing.gen2 phishingx	45	trojan.expl shellcodesm	9	trojan.expl shellcode1	4
trojan.shellcode pdfka	41	trojan.pdfka	8	trojan.pdfjsc expl	4
trojan.pidief pdfex	40	trojan.shellcode expl	8	trojan.scam fraud	4
trojan.tiff pidief	38	trojan.zzxx	8	trojan.phishingx	7
trojan.pidief blacole	36	trojan.phishingx	7	trojan.pdfka smkw	7
trojan.pdfka jmub	35	phishing	31	trojan.pdfka embedded	7

TABLE 4. Comparison of PDFiD results between the LOC and VirusTotal datasets. Since the VirusToal is more than 11 times bigger, the Norm VirusTotal aligns the quantities between the dataset, so the malicious objects are emphasized more.

#	Detected Object	LOC	VirusTotal	Norm VirusTotal
1	Obj	265810	228574	20780
2	Endobj	265778	247136	22467
3	Stream	107028	57877	5262
4	Endstream	106725	64351	5850
5	Xref	1367	11081	1007
6	Trailer	1367	12074	1098
7	StartXref	1552	11584	1053
8	/Page	**14195**	20972	1907
9	/Encrypt	31	36	3
10	/Objstm	2734	2416	220
11	/JS	52	10838	**985**
12	/JavaScript	34	13429	**1121**
13	/AA	59	453	41
14	/OpenAction	195	7277	**662**
15	/AcroForm	104	2427	221
16	/Jbig2Decode	1746	17	2
17	/RichMedia	0	49	4
18	/Launch	0	230	**21**
19	/EmbeddedFile	40	7869	**715**
20	XFA	0	1418	**129**
21	/URI	2378	43744	**3977**
22	/Colors	0	9	1

After our PdfCDR, we can observe that all indicators except one Oldversion in PDF were left with 214 detected files, which is a small increase compared to the original files. However, this does not indicate an issue, and the files are opened and functional. We plan to improve the reconstruction validation in future work.

Yara rule detection rate before and after PdfCDR on VirusTotal dataset is illustrated in Table 6. Yara's detection over the VirusTotal dataset has 102 unique types of signatures detected. The Table presents the most common signature detected in more than 1% of the VirusTotal dataset. The full analysis is found in our GitHub repository. Before PdfCDR, we can conclude that 15.4% of the files contain a suspicious attack indicator signature consisting of a mixture of signatures like embedded files, OpenActions, and javascript.

We can observe that after PdfCDR, Yara detected only two signatures out of 102 previously detected before PdfCDR, meaning the disarm was successful. All the detected signatures after PdfCDR existed before the CDR and none of them are malicious. The suspicious version signature count has slightly decreased after CDR, while Oldversion in PDF was found 1546 times, which is a slight decrease compared to the 1557 detected files before PdfCDR. We leave that for future optimization work. All documents are functional after PdfCDR, and we measure the similarity to the original files in Section VII-D.

B. VALIDATION USING VirusTotal SERVICE

We used the VirusTotal service to validate the LOC and VirusTotal datasets before and after PdfCDR. The LOC dataset had one file with one engine, indicating it was a malicious file before PdfCDR, and after PdfCDR, the engine stopped detecting the file.

For the VirusTotal dataset, Fig. 5 illustrates the histogram of the number of detected engines per file before PdfCDR.

TABLE 5. Yara rule detection rate before and after PdfCDR on LOC dataset.

#	Signatures	Before %	Before Count	After %	After Count
1	Oldversion in PDF,	6.16	160	100	**214**
2	Multiple versions	14.49	376	0	0
3	PDF invalid structure	16.68	433	36.6	0
4	Invalid trailer structure	16.68	433	0	0
5	Action in PDF	0.96	25	0	0
6	PDF warning openaction	7.43	193	0	0
7	URI on OpenAction in PDF	2	52	0	0
8	Suspicious embedded external content	2.58	67	0	0
9	ASCIIDecode in PDF	2	53	0	0
10	Multiple filtering	1.11	29	0	0
11	Invalid xref numbers	3.4	88	3.26	60
12	Suspicious javascript object	0.38	10	0	0
13	XFA in PDF	0.38	1	0	0
14	Header evasion	0.15	4	0	0
15	PDF warning remote action	0.34	9	0	0
16	JS splitting	0.38	1	0	0
17	Suspicious obfuscation getAnnots access blocks	0.38	1	0	0
18	Suspicious attack	24.85	645	0	0
19	Suspicious embedded file	0.58	15	0	0

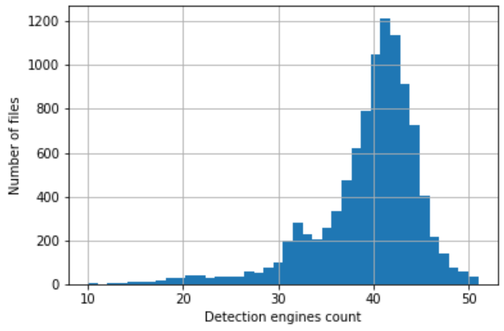


FIGURE 5. VirusTotal dataset histogram of detected engines before PdfCDR.

We can see that most of the VirusTotal dataset files were detected by 30 to 41 engines before PdfCDR.

Fig. 6 illustrates the histogram of the number of detected engines per file after PdfCDR. After PdfCDR, we can observe that the histogram changes, and most of the files after PdfCDR are detected with four engines (3806 files) and by zero engines. The low detection indicates that most engines think the file is clean.

When examining files that were detected based on four antivirus engines, we see that most of them have the same detection engines in all of them. The VirusTotal engine detection similarities lead us to think that the files are detected based on the same heuristics. Fig. 7 illustrates one of those files (md5: cff765b34c09b8c373ae86ca772ede68). We can see that the original file was detected by 45 engines and had the following signatures: invalid-Xref, autoaction, and js-embedded from VirusTotal analysis. At the same time, the PdfCDR result had no Yara detection signatures or suspicious objects left in the file, and the attack couldn't be executed. We tried several files in a sandbox and found no malicious activity while monitoring them.

We can observe from Analyzing md5:
c07436906953f1bb3f81bd73aa8bfe19 in Fig. 8 that the original file had 33 detections and contained the following

TABLE 6. Yara rule detection rate before and after PdfCDR on VirusTotal dataset.

#	Signatures	Before %	Before Count	After %	After Count
1	Suspicious attack	15.4	10743	0	0
2	Suspicious javascript object	12.83	8952	0	0
3	PDF warning OpenAction	10.26	7141	0	0
4	Action In PDF	9.26	6462	0	0
5	PDF invalide structure	5.28	3685	0	0
6	Invalid trailer structure	5.28	3685	0	0
7	Invalid xref numbers	2.5	1744	0	0
8	Oldversion in PDF	2.23	1557	75.34	1546
9	Suspicious embedded file	2.14	1493	0	0
10	JS wrong version	2.12	1480	0	0
11	XFA in PDF	1.85	1301	0	0
12	Suspicious creator	1.76	1227	0	0
13	Suspicious obfuscation using unescape	1.51	1055	0	0
14	Suspicious author	1.31	916	0	0
15	Suspicious creation	1.28	899	0	0
16	Suspicious title	1.19	833	0	0
17	Suspicious obfuscation using substr	1.14	798	0	0
18	XFA with JS in PDF	1.11	776	0	0
19	Suspicious javascript in XFA block	1.11	776	0	0
20	Shellcode hash Load LibraryA	1.04	726	0	0
21	Shellcode hash Win Exec	1.04	726	0	0
22	Shellcode hash URL download to FileA	1.04	726	0	0
23	Malicious author	1.03	718	0	0
24	Shellcode hash exit process	1.02	716	0	0
25	Suspicious version	0.78	550	24.65	506

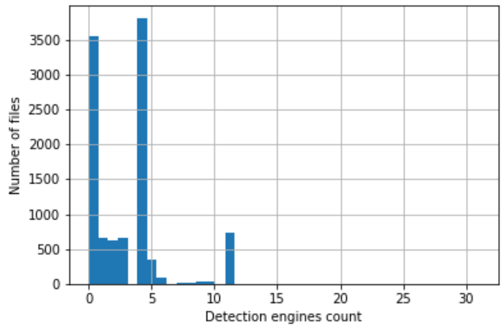


FIGURE 6. VirusTotal dataset histogram of detected engines after PdfCDR.

capabilities: javascript, auto action, invalid-Xref, acroform, and CVE-2010-2883. After PdfCDR, VirusTotal detected the file with 31 engines. From our analysis, we successfully removed all the signatures except for the CVE-2010-2883 signature, for which we do not have the specific signature used to detect the file. However, the detected exploit can't be executed without the capabilities that PdfCDR removed. Therefore CDR action was successful in preventing the attack.

From analyzing the result, we can see that files that received a high detection rate after PdfCDR are SPAM. For example, md5: bf53d5def67420fad1a72c71b1ea1f50 is illustrated in Fig. 9 where 27 engines detected the original file in VirusTotal and after PdfCDR 17 engines detected it in VirusTotal platform. The different engines indicate this is SPAM. Since PdfCDR removed the hyperlink, the file is no longer a threat. However, different artifacts that engines are using as a signature are left. We should be aware of the

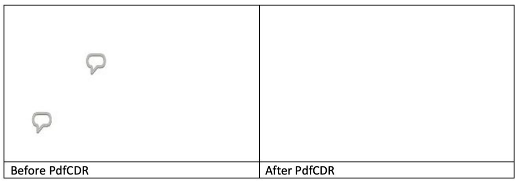


FIGURE 7. Example of one of the files detected by four engines after PdfCDR without any suspicious objects detected. The original file on the left had 45 detections and contained malicious objects. Note: to reduce space, we cut only the visible part of the document.

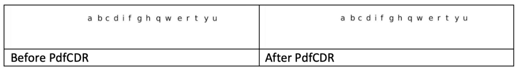


FIGURE 8. CVE-2010-2883 example, the original file on the left had 33 detections and contained malicious objects, while 31 engines detected the disarmed file without malicious active objects to execute the CVE attack. Note that we cut only the visible part of the document to reduce space.

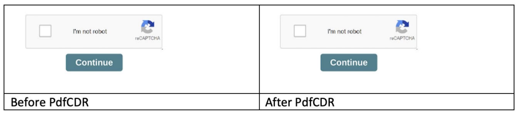


FIGURE 9. SPAM example: the original file on the left had 27 detections, and the disarmed file had 17 detected engines after PdfCDR (the link behind the button was successfully removed). Note that we cut only the visible part of the document to reduce space.

signature they are using to remove the remaining VirusTotal engine-detected heuristics and use our Yara detection to CDR engine to remove the indicators.

C. RUN TIME RESULTS

Speed is important when handling large-scale services that receive thousands of files. Some PDF files can have a complex nested structure and contain many pages. The disarming of a complex file structure takes longer. Table. 7 summarizes the LOC dataset PdfCDR run-time in seconds. We can see that the mean duration of disarming a file was 4.65 seconds, and the median was 0.73 seconds. The 90% percentile is 9.34 seconds, while the 95% increased to 31.1 seconds. When we examine the maximum value, we see anomalies in large and complex files. For example, the maximum duration for the LOC dataset was 221 seconds. The anomaly was caused by a complex file containing 1178 pages with 20117 objects and 2441 streams and PdfCDR handled each one of the components. We believe that we can further improve the timing of complex files in the future, but from the 75% percentile, most of the files are PdfCDR in less than 0.35 seconds. When analyzing the LOC CDR file complexity versus the number of pages (Fig. 10) and the number of objects and streams (Fig. 11), we can conclude that large files with more than 540 pages and 6000 objects take a long time to CDR. After investigating the root cause of the problem, we can conclude it is not due to the regular disarm and reconstruction flow (found in the Appendix Section). However, it is due to a less efficient implementation of our disarm-by-detection rules. We are sure this can be optimized in the future.

When examining the VirusTotal dataset in Table 8 the maximum file analysis duration recorded was 1837 seconds.

TABLE 7. Summary of analysis duration for the LOC dataset.

mean	4.65
median	0.73
std	14.1
min	0.17
25%	0.48
50%	0.73
75%	0.35
90%	9.34
95%	31.1
max	221

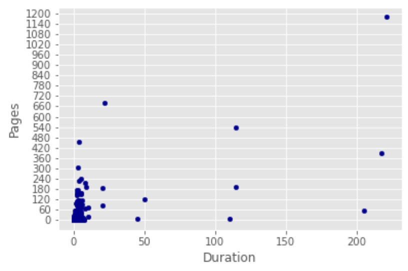


FIGURE 10. LOC dataset PdfCDR analysis duration compared to the number of pages in the dataset.

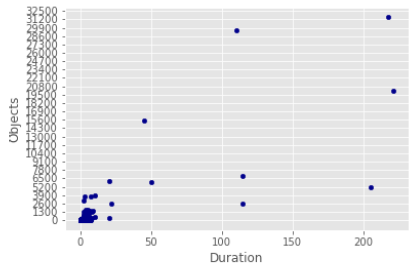


FIGURE 11. LOC dataset PdfCDR analysis duration compared to the number of objects and streams in the dataset.

The file contains 15404 objects, 1173 streams, multiple suspicious indicators, and 168 pages. But we can see that the 95% percentile is 0.14 and the median is 0.027, which indicates that overall the malicious dataset takes significantly less time compared to the LOC dataset due to the lower number of pages and objects in the VirusTotal dataset. We conclude that the detected time anomalies are rare and can be time reduced with software engineering steps.

When analyzing the VirusTotal CDR file complexity versus the number of pages (Fig. 12) and the number of objects and streams (Fig. 13), we can conclude that similar to the LOC dataset here, most of the files contain a low number of pages and the effect of the number of pages is not significant. However, the number of objects that need to be disarmed affects our implementation. The root cause of the problem is the same as the LOC dataset. Since the VirusTotal contains fewer objects, their analysis duration is shorter, and objects contribute to a higher analysis complexity.

D. OBJECTIVE IMAGE COMPARISON

Besides the detection rate, we want to check the reconstruction and similarity of the PdfCDR output to the original file. Therefore we are using image comparison similarity metrics

TABLE 8. Summary of analysis duration for the VirusTotal dataset.

mean	0.35
median	0.027
std	18.24
min	0.02
25%	0.024
50%	0.027
75%	0.03
90%	0.07
95%	0.14
max	1837

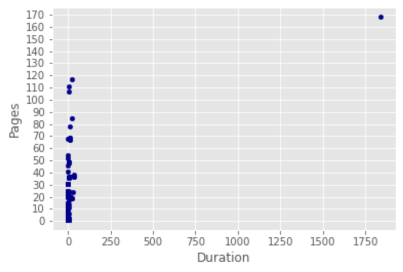


FIGURE 12. VirusTotal dataset PdfCDR analysis duration compared to the number of pages in the dataset.

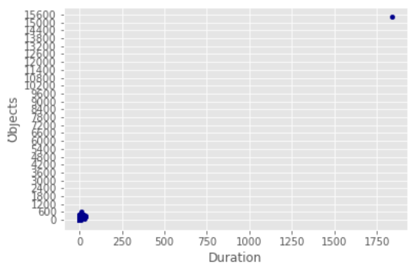


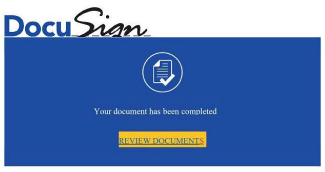
FIGURE 13. VirusTotal dataset PdfCDR analysis duration compared to the number of objects and streams in the dataset.

TABLE 9. Comparison of clean file images dataset.

	MSE	PSNR	SSIM
Mean	0.36	inf	0.999
Std	2.66	Nan	0.001
Min	0	32.2	0.973
Max	39.15	inf	1
Percentile 25%	0	Nan	1
Percentile 50%	0	Nan	1
Percentile 75%	0	Nan	1

TABLE 10. VirusTotal malicious dataset image similarity comparison.

	MSE	PSNR	SSIM
Mean	0.905	inf	0.999
Std	3.21	Nan	0.001
Min	0	32.11	0.962
Percentile 25%	0.0	47.22	0.9992
Percentile 50%	0.615	50.23	0.9996
Percentile 75%	1.23	62.99	0.9998
Max	39.96	inf	1



Your document has been completed

FIGURE 14. Phishing PDF file that contains link before PdfCDR. Due to the size, we cropped the image without empty white background.

to see if the disarm affects the page visualization. At the same time, the functionality is verified by first opening the PDF files, saving them as images, and validating their similarity before and after PdfCDR.

Table 9 summarizes the LOC dataset image similarity. We can see that the SSIM, MSE, and PSNR indicate that the resulting image is similar in 100% of the cases. Some files contain significant differences, such as was reported with a maximum MSE of 39.15. Overall, the differences could be more negligible. Note that the Inf value in PSNR was caused due to the MSE values equal to zero (divided in zero). MSE equal to zero also affects the PSNR STD and other parameters equal to Nan.

It is important to note that we assert that CDR changes the file appearance. However, in most cases, the disarm overall visual changes are insignificant compared to the security-added value.

Table 10 demonstrates that some images were identical, yielding a PSNR with ∞. The SSIM was close to 1, and the maximum MSE was very low compared to the clean dataset. There are two explanations for these results. The first is because most malicious files are based on exploits and do not contain highly detailed perceptual information. The second reason is that the malicious PDF attacks are hidden and are not visible in the original and disarm versions, which also explains the results. The similarity results in the LOC dataset were better, but the dataset is smaller. We believe that the diversity will be higher in a larger dataset like the VirusTotal dataset. Therefore, this is why the similarity metrics are lower (less similar) in the VirusTotal dataset, but we can see both show excellent similarity results.

To emphasize the above results, Fig. 14 illustrates a phishing PDF (md5: 43eb32086cbc695c3dd62bc07e9a06d7) which was detected by the VirusTotal platform by 30 detection engines and contains a lure image and a lure link. The result of the PdfCDR can be seen in Fig. 15, where the lure link was removed behind the blue link button (yellow background). As a result, the document can't risk the user anymore. Furthermore, the disarmed file was detected by only two engines meaning that those engines probably considered the lure image in their detection methodology and not the phishing link we removed.

The image similarity metrics for those images were: MSE was 0, and SSIM was 1. Therefore, we can conclude that PdfCDR, in this case, removes the threat, but the image similarity was 100% similar to the source.

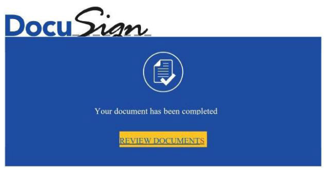


FIGURE 15. Phishing PDF file after PdfCDR. Due to the size, we cropped the image without empty white background.

When we examine a different file that uses embedded javascript and auto-open action
(md5: 479089f4110a9152245bae59df06c367) we can see it was detected in VirusTotal by 32 engines. After PdfCDR, all suspicious objects were handled, and only a single engine caught the file. We will not show the image of the file since we do not know the source of the original painting (we do not want to risk the painting infringement copyrights). However, the image similarity after PdfCDR was SSIM 0.973, and the MSE was 7.94, which is relatively high. The difference is black colored string content in the middle of the image that existed in the picture before PdfCDR removed it. The added security is much more significant because the current online detection rate for antivirus engines [12] shows a 96.3% accuracy. However, this low detection rate means that the problem still needs to be solved, and CDR is needed to increase user security.

VIII. LIMITATIONS AND FUTURE WORK

This research used Yara rules and PDFiD, which are static analysis solutions. Yara is a rule-based signature that was created based on known attacks. The limitation of Yara is the ability to adapt it to new permutations. The rules also may cause false positives and should be updated regularly. PDfiD is not a detection tool but is used for visibility and research. Not all disarm actions will remove a specific PDF section even after CDR. As a result, PDFiD indicates capabilities and shouldn't be regarded as a detection tool since the behavior may be sanitized.

It is important to note that CDR changes the file. Therefore, it is different from the original file and it is impossible to sign the document, which is the drawback of the method for some enterprise use cases.

Regarding validation, the dataset is from VirusTotal [56], and the benign dataset was already found in VirusTotal databases. This work aims to check if PdfCDR managed to disarm and remove all attack vectors from the PDF dataset. As a result, testing it against VirusTotal multi-antivirus engines that are constantly updated is the most accurate valuation we have available. One may argue that validating the VirusTotal dataset against VirusTotal is biased. However, the PdfCDR output is a modified file with different characteristics and hash function representation. As a result, it is unknown to VirusTotal when uploading it, and it is scanned as a new file. As far as we know, this is the most accurate method of validating the CDR output in terms of detection.

This work measures the effect of PdfCDR using well-known image similarity metrics. However, it is essential to emphasize that the functionality may change. For example, PDF Forms will look similar perception-wise, but the functionality will be altered, and active code components will be disabled. Therefore, the user experience and usability may be decreased and not considered at the moment. We plan to extend the image similarity to subjective user experience evaluation in our future work [68].

IX. CONCLUSION

In this work, we investigated the Adobe PDF file format and reviewed the different attack vectors that it may contain. We proposed a novel zero-trust CDR solution named PdfCDR designed to protect users from files entering the organization. We shared our research methodology, which includes a static examination of the file using PDFiD, Yara detection, and VirusTotal service that contains multiple antivirus detection and malware sandboxes. We also confirmed that the malicious files are not detected in the malware sandbox and that all files are usable. Furthermore, the PdfCDR methodology enables verification that the threats were removed as part of the development and operation cycles.

While our methodology can be extended to other file formats, we decided to focus on PDF due to its popularity and cross-platform support. As a result, PDFs are exploited by malware and phishing campaigns. As far as we know, this is the first work focusing on zero-trust malware disarm for PDF files, allowing us to continue working on the resulting file after the disarming process. Research concerning CDR for PDF-type files has been conducted primarily in the industry, and there is a minimal number of previous related works in academia [15], and in the open-source community [31].

PdfCDR was shown to be effective for large-scale deployments, and the disarm duration was very low. For the first time, we investigated the effect of the disarm process based on professional multi-scan engines (VirusTotal) and based on objective image similarity metrics that validate that the files are working and are similar to the original documents. We found that the overall effect of the disarm can be significant. However, in most cases, the differences are insignificant. Therefore, the security benefits of PdfCDR are much higher than the image similarity changes caused by using it and removing/sanitizing parts of the documents.

Our solution proposes a novel method that enables updating PdfCDR to disarm and reconstruct new vulnerabilities detected and shared via social media or any other sharing method. The technology does not need to update the disarm or the reconstruction code logic, which is expensive. Instead, the updates are done only by pushing and updating the signature database in real-time. The automatic updates significantly reduce the organization's ability to prevent evasive malware as soon as the vulnerability is detected without any need for software upgrades. Since new vulnerabilities are always found, the PdfCDR method reduces the vulnerability update gap and offers zero-trust prevention as soon as a vulnerability detection rule is shared. The results indicate that PdfCDR successfully CDR 90% of the malicious files and validated they are clean. At the same time, the remaining 10% of the dataset were encrypted (36 files) or had abnormal structures compared to the standard, and all of them were quarantined. PdfCDR was able to CDR all LOC dataset files except 31 encrypted files that were quarantined.

APPENDIX

The following sections describe the different attack vectors found in the PDF file format and discuss how to disarm them in detail.

A. METADATA

The Metadata section provides information about that document that is not directly visible in the PDF reader software and is used in cataloging and searching for documents in external databases. Several previous works suggest using the metadata content as a fingerprinting mechanism [69], and for extracting private information [70]. This section contains information such as the document properties, authors, subject, creator (name of the application software that created the original document), producer (the name of the application that produced the PDF), CreationDate, ModDate, and trapped (indicating whether the document has been modified to include trapping information [48]).

The threat: Metadata fields can't be trusted and may contain malicious code.

The solution: Sanitize the Metadata section. Since the data is not visible, there is no effect on the document's visual appearance.

Example: The following malware [71] used JS script and Metadata Producer section which use three vulnerabilities inside the Producer information:

1) Collab.collectEmailInfo() uses CVE-2007-5659
2) Util.printf() uses CVE-2008-2992
3) Collab.getIcon() uses CVE-2009-0927

The Producer code was executed as follows:

```
\JS(var a=this.producer;)
```

To disarm these groups of CVE attacks, we remove the JS script from the file and sanitize all Metadata fields by replacing the content with empty strings. However, there could be other initiators in addition to JS; therefore, our algorithm sanitizes all Metadata sections.

B. CODE COMMENTS

Code comments can be used and inserted anywhere in the PDF file. PDF code comment starts with % symbol.

The threat:PDF file viewers ignore those lines of code, but they can be used as a source of information about the authors and also as malicious code hidden in comments that can be used.

The solution: Code comments are removed. Since the data is not visible, there is no effect on the document's visual appearance.

C. LINKS

The most common PDF attack vectors are malicious links that can hide under regular-looking buttons inside the PDF file [72], [73]. The button could be a Fake CAPTCHA, coupon, file sharing download/request access button, static picture with a video play button, or any lure trick. When the non-suspicious users click on the button, they are taken to an attacker-controlled website.

The threat: Clicking on the URL can lead to various web attacks, file download, or phishing attacks.

The solution: Removing the link. Since the data is visible, there is an effect on the document's visual appearance. Therefore, the link's removal is a trade-off between increased security and decreased visual appearance and functionality. Removing the link behind a button will disarm the malicious activity in phishing attacks. However, the visual appearance change will not be detected, but the document's functionality may be damaged. For legitimate files, the appearance will stay intact, but some functionality may not work.

Example: In this example [72], the malicious website uses different web re-directions that will lead the user to adult dating and, in other cases, to gambling or adult content. A different common attack sends the user to a malicious fake Office 365 domain and asks the user for his email username and password to send the attached PDF to his email [72]). Many similar attacks can be used as a lure of well-known file-sharing fake websites to deceive the user into providing his username and password for that service.

D. DYNAMIC RESOURCES AND SCRIPTS

JavaScript and ActionScript are two scripting languages widely adopted in PDF files and websites, due to their high flexibility and interpreted nature.

The threat: They allow attackers to use them for exploiting vulnerabilities.

The solution: Remove JavaScript and ActionScript. Since the data is visible, there is an effect on the document's visual appearance. Therefore, the JavaScript and ActionScript removal is a trade-off between increased security and decreased visual appearance and functionality, and some dynamic content will be unavailable.

Example: Initial attacks were reported in 2008 [74] when PDF files containing JavaScript attacks started to be used in PDF files. An interesting case is an attack against the Hacking Team, an Italian security company, which caused a loss of more than 400 GB of classified data [75]. The scripting languages can be easily obfuscated and interpreted in run time to evade antivirus and AI-based malware detection. The recommended step is to remove the script because the static analysis can miss the evasive malicious act. Furthermore, evasive scripts can bypass dynamic (sandbox) analysis in a never-ending cat and mouse game. As a result, we recommend removing the script from the files.

E. EMBEDDED CONTENT

A PDF file may contain additional objects or file types, including JavaScript, SWF, HTML, Microsoft Office, EXE, PDF, image, video, etc.

The threat: This functionality can be used to embed a malicious file inside a seamless-looking file and leverage vulnerabilities of other file types to perform its malicious activity.

The solution: Recursive CDR for each embedding object that can be handled and removal of objects that can not be supported. In this work, we removed the embedded object and will extend this capability in the future.

Example: An attack can be a simple malicious office file embedding that lures the user to open it or a more complex exploit like flash file embedding (CVE-2010-3654). In that CVE, a malicious Flash movie file (.SWF) was embedded in the PDF, and JavaScript or PDF commands (*/Launch*, */Action*, or */EmbeddedFiles*) were used combined with social engineering/image lure [76]. Not only embedded objects are considered as risk concerns, but also fonts [77] and media objects. Therefore, we replace the standard fonts. Disarming the image can be done by using image transcoding [78] or image format changing. We recommend disarming the image and focusing multimedia/images on that in our future work. However, image modification may decrease the user quality of experience slightly.

F. ENCRYPTION

PDF encryption [48] exists at multiple levels in a PDF document. Document-level encryption applies to all strings and streams in the PDF document, except for the ID entry in the trailer, strings defined in the Encryption dictionary, and strings inside content streams and object streams, which are encrypted. However, the encryption does not include other object types, such as Integers and Boolean values that illustrate the document structure. At the same time, strings and streams are encrypted and contain the content of the PDF.

The problem: Encryption may contain/hide malicious content.

The solution: Encryption at the moment is not handled, and an encrypted file will be marked as unsafe, and we plan to extend the support in future work.

G. MULTIMEDIA AND IMAGE CONTENT

Sound and Movie objects were the first embedding objects in PDF, while as embedded objects types, they were deprecated in PDF version 1.5, and now they are embedded through Annotation objects.

The problem: Multimedia objects are a concern because they contain information used for a data attack (malformed file), particularly for movie files. In addition, movies and sounds may also contain hidden information embedded within the file format. Multimedia objects may include additional metadata such as author or source information subject to a data disclosure risk or content that can hide evasive attacks. Image has similar data hidden behavior.

The solution: The solution replaces the multimedia and image embedded object with re-encoded and harmless content. However, we plan to extend our PdfCDR platform to support this activity in the future. Furthermore, current multimedia and image disarm algorithms are out of this research scope since each media content needs a dedicated disarm algorithm since some standards contain various attack vectors.

H. INTERACTIVE OBJECTS

This part is split into annotations and triggers/actions and presents them in detail.

1) ANNOTATIONS

Annotations allow a user to interact with the document in multiple ways, such as inserting comments into the PDF, visible icons like a paperclip, caret, drawing, etc. An annotation with a visible icon is called a markup annotation.

The problem: From a security perspective, annotations are dangerous, contain hidden data/functionality, and sometimes can be revealed only based on the user's action.

The solution: Remove the annotations from the document to avoid zero/one day's exploits that are evasive and hard to detect. However, since the data is visible, there is an effect on the document's visual appearance. Therefore, the removal of the annotations is a trade-off between increased security and decreased visual appearance and functionality, and some document capabilities will be unavailable.

Example: For example, a mouse hovering over a specific area to expose functionality. It can also be rendered invisible to the user. Another concern is that the annotation can contain media content /objects that contain vulnerabilities in the software interpreting the file. Researcher [79] found that annotations can be used to obfuscate malicious JavaScript code. In addition, recent research [80] found that digitally signed PDF documents can be altered by the receiver in such a way that the changes are undetectable, either in all PDF applications or in a subset of them. One of the methods is called Evil Annotation Attack (EAA). The researchers found three types of annotations capable of hiding and adding text and images. All three can be used to modify a certified document and inject malicious content stealthily. To execute the attack, the attacker modifies a certified document by including the annotation with the malicious content at a position of the attacker's choice.

2) TRIGGERS AND ACTIONS

Trigger and action events can invoke actions that will execute on the user's system and cause data hiding and data attack risks [48]. There are many types of triggers, and they are considered one of the most popular and straightforward attacks in PDF documents.

PDF actions can cause an event on the system reading the file through the viewer application. Actions include linking or resolving to websites, accessing embedded files, launching applications, and executing scripts [48]. Actions can be attached to the document, individual pages, annotations, outline items, and interactive form fields and triggered through various events. Similar to triggers, actions can act as a starting point to execute a chain of actions.

The problem: Common attack vectors are hard to detect by static and dynamic (sandbox) detections.

The solution: Remove all triggers and actions. However, since the data is visible, there is an effect on the document's visual appearance. Therefore, the triggers and actions removal is a trade-off between increased security and decreased visual appearance and functionality, and some document capabilities will be unavailable.

Example: For example, [81] uses CVE-2018-4993 and works as follows [81]: a trigger action (*/AA*) with a child dictionary with key name */O* that specifies that the action should occur when a document is opened. In the next set, */JS* (JavaScript) comes into action.

I. ACROBAT FORMS

The interactive form is a collection of fields used to collect information from the user inside a PDF file.

There are two interactive methods:

1) Acrobat Form (AcroForm): It is a simple PDF object containing form fields that can store objects like buttons, images, and code. In addition, the code can be stored inside other objects or object streams (large chunks of compressed data), making it a great place to hide malware attacks.
   The problem: Complex attack vectors may contain hidden evasive data that can be changed dynamically.
   The solution:Remove all AcroForm objects. However, since the data is visible, there is an effect on the document's visual appearance. Therefore, AcroForm's removal is a trade-off between increased security and decreased visual appearance and functionality, and some document capabilities will be unavailable.
2) Adobe XML Forms Architecture (XFA) Form: XFA is a group of proprietary XML specifications used to enhance the processing of web forms and dynamic resize fields within the document. Introduced by Adobe in PDF v1.5 and was deprecated in PDF V2.0.
   The problem: Complex attack vectors may contain hidden evasive data that can be changed dynamically.
   The solution:Remove all XFA objects. However, since the data is visible, there is an effect on the document's visual appearance. Therefore, the XFA's removal is a trade-off between increased security and decreased visual appearance and functionality, and some document capabilities will be unavailable.
   Example: Researchers have found [82] malicious PDF with XFA with obfuscated JavaScript inside of it. The disarm algorithm removes the XFA and the associated JavaScript to prevent this attack.
   POC of the XFA attack was suggested [83] and used a stream that contains an XML stylesheet that can also be used to initiate a direct connection to a remote server or SMB share (Fig. 16). The application will parse the URL and immediately attempt a connection to stop it.

```
1 0 obj <<>>

stream

<?xml version="1.0" ?>

<?xml-stylesheet href="\\example.com\share\whatever.xslt" type="text/xsl" ?>

endstream
endobj
```

FIGURE 16. XML stylesheet used to initiate a direct connection to a remote server or SMB share.

J. PDF FUNCTIONS

Function objects [48] permit the representation of a numerical transformation and can take multiple values and output any number of values. There are four categories of functions in the standard:

1) **Type 0 Functions**: A sampled function that uses a table of values to define the function.
2) **Type 2 Functions**: An exponential interpolation function that uses a set of coefficients.
3) **Type 3 Functions**: A stitching function, which is a combination of other functions, partitioned across a domain [84].
4) **Type 4 Functions**: A PostScript calculator function which uses operators from the PostScript language to define an arithmetic expression [84].

The problem: The concern with functions is their ability to manipulate input and generate output and should be validated in order to make sure no boundaries are crossed since they present a data attack risk. Furthermore, *Function Type 4* also involves arithmetic operations with the user-supplied operands. Attacks such as invalid instructions overflow, or illegal mathematical operations could be exploited by attackers.

The solution: Remove the function dictionary and related data streams and related references to the function dictionary. The visible drawback is that the removal may break and affect the visible appearance of the content.

Example: For example, CVE-2019-5042 [85] discovered a Use-After-Free vulnerability exists in the way *Function Type 0* PDF elements are processed in Aspose.PDF for C++. A specially crafted PDF can cause a dangling heap pointer, resulting in a use-after-free. An attacker can send a malicious PDF to trigger this vulnerability.

ACKNOWLEDGMENT

The author would like to thank Ofek Alon, Amit Dvir, and Chen Hajaj for their comments and feedback, VirusTotal for granting them access to their cloud service, malware collection for educational use, and Aspose for providing an academic license for their ASPOSE total product libraries for parsing and manipulating PDF files. This work is based on a patent pending request 63/408631.

REFERENCES

[1] Adobe. (2021). *ISO 32000-2:2020 Document Management—Portable Document Format—Part 2: PDF 2.0*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.iso.org/standard/75839.html

[2] Dark Reading. (2022). *New Attack Shows Weaponized PDF Files Remain a Threat*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.darkreading.com/attacks-breaches/weaponized-pdf-files-remain-a-threat-research-shows

[3] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, "Detection of malicious PDF files and directions for enhancements: A state-of-the art survey," *Comput. Secur.*, vol. 48, pp. 246–266, Feb. 2015.

[4] *What is Content Disarm and Reconstruction (CDR)*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-content-disarm-and-reconstruction-cdr/

[5] R. Future. (2018). *Microsoft Targeted by 8 of 10 Top Vulnerabilities*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.recordedfuture.com/top-vulnerabilities-2018/

[6] Statista. (2020). *Most Common Malicious File Types Received Globally Via Web and E-Mail in 2020*. Accessed: Sep. 17, 2022. [Online]. Available: https://www.statista.com/statistics/1238996/top-malware-by-file-type/

[7] VirusTotal. (2022). *Yara: The Pattern Matching Swiss Knife, Google Open Source Project*. Accessed: Jan. 15, 2022. [Online]. Available: https://opensource.google/projects/yara

[8] E. Amer, I. Zelinka, and S. El-Sappagh, "A multi-perspective malware detection approach through behavioral fusion of API call sequence," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102449.

[9] D. Serpanos, P. Michalopoulos, G. Xenos, and V. Ieronymakis, "Sisyfos: A modular and extendable open malware analysis platform," *Appl. Sci.*, vol. 11, no. 7, p. 2980, Mar. 2021.

[10] A.-A. M. Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, "A review of artificial intelligence based malware detection using deep learning," *Mater. Today, Proc.*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785321048586, doi: 10.1016/j.matpr.2021.07.012.

[11] E. Montalbano. (2022). *Snake Keylogger Spreads Through Malicious PDFs*. Accessed: Dec. 3, 2022. [Online]. Available: https://threatpost.com/snake-keylogger-pdfs/179703/

[12] AV-Comparatives. (2021). *Malware Protection Test March 2021 Date*. Accessed: Feb. 19, 2022. [Online]. Available: https://www.av-comparatives.org/tests/malware-protection-test-march-2021/

[13] S. Wiseman, "Content security through transformation," *Comput. Fraud Secur.*, vol. 2017, no. 9, pp. 5–10, Sep. 2017.

[14] G. Sim, "Defending against the malware flood," *Netw. Secur.*, vol. 2018, no. 5, pp. 12–13, 2018.

[15] H. Jaehyeok, Y. Yoon, G. Hur, J. Lee, J. Choi, S. Hong, and S. Lee, "Secure file transfer method and forensic readiness by converting file format in network segmentation environment," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 29, no. 4, pp. 859–866, 2019.

[16] Y. Sunshine, "The rise of MSP & CSP vulnerabilities: Storehouses for secure data," *Comput. Fraud Secur.*, vol. 2021, no. 2, pp. 15–19, Jan. 2021.

[17] M. Baker. (2022). *Content Disarm and Reconstruction—A Proactive Stance on Cybersecurity*. Accessed: Jan. 15, 2022. [Online]. Available: https://uktechnews.co.uk/2021/10/28/content-disarm-and-reconstruction-a-proactive-stance-on-cybersecurity/

[18] X. Cai, K. Shi, K. She, S. Zhong, Y. Soh, and Y. Yu, "Performance error estimation and elastic integral event triggering mechanism design for T–S fuzzy networked control system under DoS attacks," *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 4, pp. 1327–1339, Apr. 2023.

[19] X. Cai, K. Shi, K. She, S. Zhong, and Y. Tang, "Quantized sampled-data control tactic for T-S fuzzy NCS under stochastic cyber-attacks and its application to truck-trailer system," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7023–7032, Jul. 2022.

[20] F. Salahdine and N. Kaabouch, "Social engineering attacks: A survey," *Future Internet*, vol. 11, no. 4, p. 89, Apr. 2019.

[21] J. M. Esparza, "Understanding the credential theft lifecycle," *Comput. Fraud Secur.*, vol. 2019, no. 2, pp. 6–9, Jan. 2019.

[22] R. Dubin, "Content disarm and reconstruction of RTF files a zero file trust methodology," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1461–1472, 2023.

[23] A. Kim, J. Oh, J. Ryu, and K. Lee, "A review of insider threat detection approaches with IoT perspective," *IEEE Access*, vol. 8, pp. 78847–78867, 2020.

[24] S. Handa and P. Shoard. (2021). *Hype Cycle for Network Security*. Accessed: Feb. 19, 2022. [Online]. Available: https://www.gartner.com/en

[25] E. Montalbano. (2020). *Hackers Update Age-Old Excel 4.0 Macro Attack*. [Online]. Available: https://threatpost.com/hackers-update-age-old-excel-4-0-macro-attack/154898/

[26] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, and Y. Vorobeychik, "Improving robustness of ML classifiers against realizable evasion attacks using conserved features," in *Proc. 28th USENIX Secur. Symp. (USENIX Security)*, 2019, pp. 285–302.

[27] A. Ashkenazy and S. Zini. (2019). *Cylance, I Kill You!* [Online]. Available: https://skylightcyber.com/2019/07/18/cylance-i-kill-you/

[28] H. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading machine learning malware detection," in *Proc. Black Hat*, 2017, pp. 1–6.

[29] I. Kringel. (2021). *PDF as a Weapon of Choice on the Cybersecurity Battlefield*. Accessed: Mar. 15, 2022. [Online]. Available: https://www.deepinstinct.com/blog/pdf-as-a-weapon-of-choice-on-the-cybersecurity-battlefield

[30] V. Kropotov and F. Yarochkin. (2019). *Hunting Threats on Twitter: How Social Media Can be Used to Gather Actionable Threat Intelligence*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.trendmicro.com/vinfo/it/security/news/cybercrime-and-digital-threats/hunting-threats-on-twitter

[31] *DocBleach Open Source CDR*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/docbleach/DocBleach

[32] D. Righetto. (2022). *Document Upload Protection*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/righettod/document-upload-protection

[33] C. Carmony, M. Zhang, X. Hu, A. Vasisht Bhaskar, and H. Yin, "Extract me if you can: Abusing PDF parsers in malware detectors," in *Proc. NDSS*, 2016, pp. 1–15.

[34] Y. Otsubo, M. Mimura, and H. Tanaka, "O-checker: Detection of malicious documents through deviation from file format specifications," in *Proc. Black Hat USA*, 2016, pp. 1–16.

[35] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, Dec. 2012, pp. 239–248.

[36] D. Stevens, "Malicious PDF documents explained," *IEEE Secur. Privacy Mag.*, vol. 9, no. 1, pp. 80–82, Jan. 2011.

[37] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. NDSS*, 2016.

[38] M. A. Munson and J. S. Cross, "Deep pdf parsing to extract features for detecting embedded malware," Sandia Nat. Lab. (SNL), Albuquerque, NM, USA, Tech. Rep. SAND2011-7982, 2011.

[39] D. Maiorca, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: Lessons learned from PDF-based attacks," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Jul. 2020.

[40] N. Hason, A. Dvir, and C. Hajaj, "Robust malicious domain detection," in *Cyber Security Cryptography and Machine Learning*. Be'er Sheva, Israel: Springer, Jul. 2020, pp. 45–61.

[41] Y. Xue, J. Wang, Y. Liu, H. Xiao, J. Sun, and M. Chandramohan, "Detection and classification of malicious Javascript via attack behavior modelling," in *Proc. Int. Symp. Softw. Test. Anal.*, Jul. 2015, pp. 48–59.

[42] S. Adhatarao and C. Lauradoux, "Exploitation and sanitization of hidden data in PDF files: Do security agencies sanitize their PDF files?" in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2021, pp. 35–44.

[43] T. Aura, T. A. Kuhn, and M. Roe, "Scanning electronic documents for personally identifiable information," in *Proc. 5th ACM Workshop Privacy Electron. Soc.*, Oct. 2006, pp. 41–50.

[44] Y. Feng, B. Liu, X. Cui, C. Liu, X. Kang, and J. Su, "A systematic method on PDF privacy leakage issues," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 1020–1029.

[45] S. L. Garfinkel, "Leaking sensitive information in complex document files- and how to prevent it," *IEEE Secur. Privacy*, vol. 12, no. 1, pp. 20–27, Jan./Feb. 2013.

[46] D. Sánchez, M. Batet, and A. Viejo, "Automatic general-purpose sanitization of textual documents," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 6, pp. 853–862, Jun. 2013.

[47] K. Mendelman, "Fingerprinting an organization using metadata of public documents," M.S. thesis, Inst. Comput. Sci. Cyber Secur. Curriculum, Univ. Tartu, Tartu, Estonia, 2018.

[48] *Inspection and Sanitization Guidance for Portable Document Format*, Central Security Service (CSS), NSA, Fort Meade, MD, USA, 2017.
[49] Adobe Systems Incorporated. (2008). *PDF, Version 1.7 (ISO 32000-1:2008)*. Accessed: Mar. 15, 2022. [Online]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000277.shtml
[50] Adobe. (2020). *PDF, Version 2.0 (ISO 32000-2:2020)*. Accessed: Mar. 15, 2022. [Online]. Available: https://www.iso.org/standard/75839.html
[51] *Document Management—Portable Document Format—Part 1: PDF 1.7*, Standard 32000-1:2008, International Organization for Standardization, Geneva, Switzerland, 2008. Accessed: Jan. 15, 2022. [Online]. Available: https://www.iso.org/standard/51502.html
[52] VMware. (2021). *RabbitMQ Message Broker*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.rabbitmq.com/
[53] K. Dontje. (2022). *Vulnerability Spotlight: Use-After-Free Vulnerabilities in Foxit Reader Could Lead to Arbitrary Code Execution*. Accessed: Jan. 15, 2022. [Online]. Available: https://blog.talosintelligence.com/vulnerability-spotlight-use-after-free-vulnerabilities-in-foxit-reader-could-lead-to-arbitrary-code-execution/
[54] S. Gatlan. (2022). *Google Releases 165 YARA Rules to Detect Cobalt Strike Attacks*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.bleepingcomputer.com/news/security/google-releases-165-yara-rules-to-detect-cobalt-strike-attacks/
[55] D. Stevens. (2020). *PDF Tools by Didier Stevens*. Accessed: Sep. 17, 2022. [Online]. Available: https://blog.didierstevens.com/programs/pdf-tools/
[56] VirusTotal. (2022). *Virustotal Website*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.virustotal.com/gui/home/upload
[57] Ditekshen. (2022). *Detection and Hunting Signatures*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/ditekshen/detection
[58] InQuest. (2022). *InQuest YARA Rules*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/InQuest/yara-rules
[59] LPART. (2022). *Static Analysis Malicious Files YARA Rules*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/lprat/static_file_analysis/blob/master/yara_rules1/pdf.yar
[60] YARA-Rules. (2022). *YARA Rules Project*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/Yara-Rules/rules/blob/master/maldocs/Maldoc_PDF.yar
[61] TyLabs. (2022). *Quicksand YARA Rules*. Accessed: Jan. 15, 2022. [Online]. Available: https://github.com/tylabs/quicksand/blob/main/src/quicksand/quicksand_pdf.yara
[62] R. Dubin. (2022). *Ariel-CDR GitHub Repository*. Accessed: Sep. 17, 2022. [Online]. Available: https://github.com/randubin/PDF-CDR
[63] I. Avcıbas, B. Sankur, and K. Sayood, "Statistical evaluation of image quality measures," *J. Electron. Imag.*, vol. 11, no. 2, pp. 206–223, Apr. 2002.
[64] J. E. Farrell, "Image quality evaluation," in *Color Imaging: Vision and Technology*, L. W. Macdonald and M. R. Luo, Eds. Hoboken, NJ, USA: Wiley, 1999, pp. 285–313.
[65] M. Cadik and P. Slavik, "Evaluation of two principal approaches to objective image quality assessment," in *Proc. 8th Int. Conf. Inf. Vis.*, 2004, pp. 513–518.
[66] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[67] LOC. (2022). *Library of Congress PDF Dataset*. Accessed: Jan. 15, 2022. [Online]. Available: https://www.loc.gov/item/2020445568/
[68] R. Shalala, R. Dubin, O. Hadar, and A. Dvir, "Video QoE prediction based on user profile," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 588–592.
[69] C. Alonso and J. Palzon, "Tactical fingerprinting using foca," in *Proc. DEF CON Hacking Conf.*, 2009, pp. 41–50.
[70] C. Alonso, E. Rando, F. Oca, and A. Guzmán, "Disclosing private information from metadata, hidden info and lost data," in *Proc. Black Hat Eur.*, 2009, p.102.
[71] *Cyren Security Blog, How PDF Files Hide Malware—example—PDF Scan From Xerox*. Accessed: Apr. 18, 2023. [Online]. Available: https://www.cyren.com/blog/articles/how-pdf-files-hide-malware-example-pdf-scan-from-xerox-1247
[72] A. Hosseini and A. Chitwadgi. (2021). *2020 Phishing Trends With PDF Files*. [Online]. Available: https://unit42.paloaltonetworks.com/phishing-trends-with-pdf-files/
[73] M. A. Ivanov, B. V. Kliuchnikova, I. V. Chugunkov, and A. M. Plaksina, "Phishing attacks and protection against them," in *Proc. IEEE Conf. Russian Young Res. Electr. Electron. Eng. (ElConRus)*, Jan. 2021, pp. 425–428.
[74] D. Maiorca, P. Russu, I. Corona, B. Biggio, and G. Giacinto, "Detection of malicious scripting code through discriminant and adversary-aware API analysis," in *Proc. 1st Italian Conf. Cybersecur. (ITASEC)*, vol. 1816, 2017, pp. 96–105.
[75] Ars Technica. (2015). *Hacking Teams Flash 0-Day: Potent Enough to Infect Actual Chrome User*. [Online]. Available: https://arstechnica.com/information-technology/2015/07/hacking-teams-flash-0day-potent-enough-to-infect-actual-chrome-user/
[76] R. Naraine. (2010). *Hacker Finds a Way to Exploit PDF Files, Without a Vulnerability*. [Online]. Available: https://www.zdnet.com/article/hacker-finds-a-way-to-exploit-pdf-files-without-a-vulnerability/
[77] J. Sejtko. (2011). *Another Nasty Trick in Malicious PDF*. [Online]. Available: https://blog.avast.com/2011/04/22/another-nasty-trick-in-malicious-pdf/
[78] B. Ines, J. S. Ben, and Z. Ezzeddine, "Online multi-sprites based video watermarking robust to collusion and transcoding attacks for emerging applications," *Multimedia Tools Appl.*, vol. 77, no. 11, pp. 14361–14379, Jun. 2018, doi: 10.1007/s11042-017-5033-y.
[79] B. Zdrnja. (2010). *Javascript Obfuscation in PDF: Sky is the Limit*. [Online]. Available: https://isc.sans.edu/diary/JavaScript+obfuscation+in+PDF+Sky+is+the+limit/8587
[80] S. Rohlmann, V. Mladenov, C. Mainka, and J. Schwenk, "Breaking the specification: PDF certification," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 1485–1501.
[81] P. Stokes. (2019). *Malicious PDFs Revealing the Techniques Behind the Attacks*. [Online]. Available: https://www.sentinelone.com/blog/malicious-pdfs-revealing-techniques-behind-attacks
[82] *Complex—PDF Hides Malware Inside XFA Which is Inside PNG—Not an Image*. [Online]. Available: https://www.cyren.com/blog/articles/complex-pdf-hides-malware-inside-xfa-which-is-inside-png-not-an-image-1229
[83] (Jan. 2019). *Alex Inführ, Adobe Reader—PDF Callback Via XSLT Stylesheet in XFA*. [Online]. Available: https://insert-script.blogspot.com/2019/01/adobe-reader-pdf-callback-via-xslt.html
[84] *Document Management—Portable Document Format—Part 1: PDF 1.7*, Int. Org. Standardization, ISO Central Secretariat, Geneva, Switzerland, 2008. [Online]. Available: https://www.iso.org/standard/51502.html
[85] Cisco. (2019). *Aspose.PDF for C++ Remote Code Execution Vulnerability*. [Online]. Available: https://talosintelligence.com/vulnerability_reports/TALOS-2019-0809

RAN DUBIN (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in communication systems engineering from Ben-Gurion University, Beer Sheva, Israel. He is currently a Faculty Member with the Computer Science Department, Ariel University, Israel. His research interests include zero-trust cyber protection, malware disarms and reconstruction, encrypted network traffic detection, deep packet inspection (DPI), bypassing AI, natural language processing, and AI trust.