Vulnerabilities in PDF

Kyu Doun Sim

kdsim@connect.ust.hk

**Motivation**

Phil Ydens, Adobe’s VP Engineering for Document could, estimates that there are 2.5 trillion PDF documents in the world [1]. A lot of contents including academic journals and paper works are in pdf format. It is so widely used that people will send their resumes or cover letters in a pdf format for job applications. It would be one of the most widely used document file format in the world. It was Adobe’s goal to make PDF universal and has achieved its goal. Just from being a text format, PDF allows digital signatures, file attachments, and metadata to work in government organizations. The success of PDF is undoubtful. However, it generally led me to wonder if there are any vulnerabilities the PDF file format. A lot of security vulnerabilities often come from maliciously crafted user inputs that could take advantage of the implementation flaws of a software. What users can at most do with PDF files are to either read or write texts on them. This generally led me to investigate what security vulnerabilities that it could have and what we could learn from it.

**Introduction**

The following written report explores the cybersecurity aspect in PDF file formats. It will start off my introducing what PDFs are and any relevant terminologies or technologies that need to be explained in order to explain without hinderance. I will present what vulnerabilities that PDF has shown so far and introduce any impactful examples or attacks performed by exploiting these vulnerabilities. I will further introduce patches done by the company that created the file format, Adobe, and third-party users’ or organization’s contribution to help users to use the file format without security vulnerabilities.

**Relevant Terminologies or Technologies**

**Portable Document Format (PDF)**

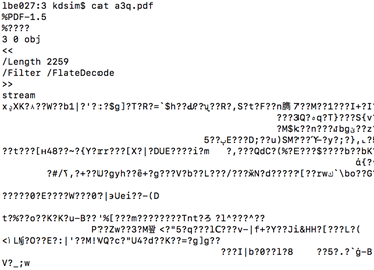
The Portable Document Format, in short PDF, is a file format created by Adobe, a firm famously known for its range of software that it provides. Any file that has a .pdf extension will be available to be opened by various software. There are numerous software applications that can read PDF files.

PDF depending on the operating system. It was developed by Adobe in the 1990s and the company has been pushing the public to use PDF as a standard. One of the main problems of document files back in the 1990s was if users were opening documents in a different operating system, document viewing software, and even a different version of the document reader, the fonts and graphics will not be consistent in these readers. Adobe has been pushing to solve this main concern that users had back then. Newspaper companies were very excited with such technology because printing these documents were also a problem back then. As the file format were able to successful to give consistent text and image quality regardless of the software reading the file and the operating system, more people found the appeal to it and started to widely adopt the format.

Portal Document Format, pdf, is a type of file created from the software company Adobe. The company has created various software including Photoshop, Adobe Premiere and has been creating a lot of software that are used for creators. PDF is no exception, and it is probably one of the most widely used text format. The following is the internal structure of the format.

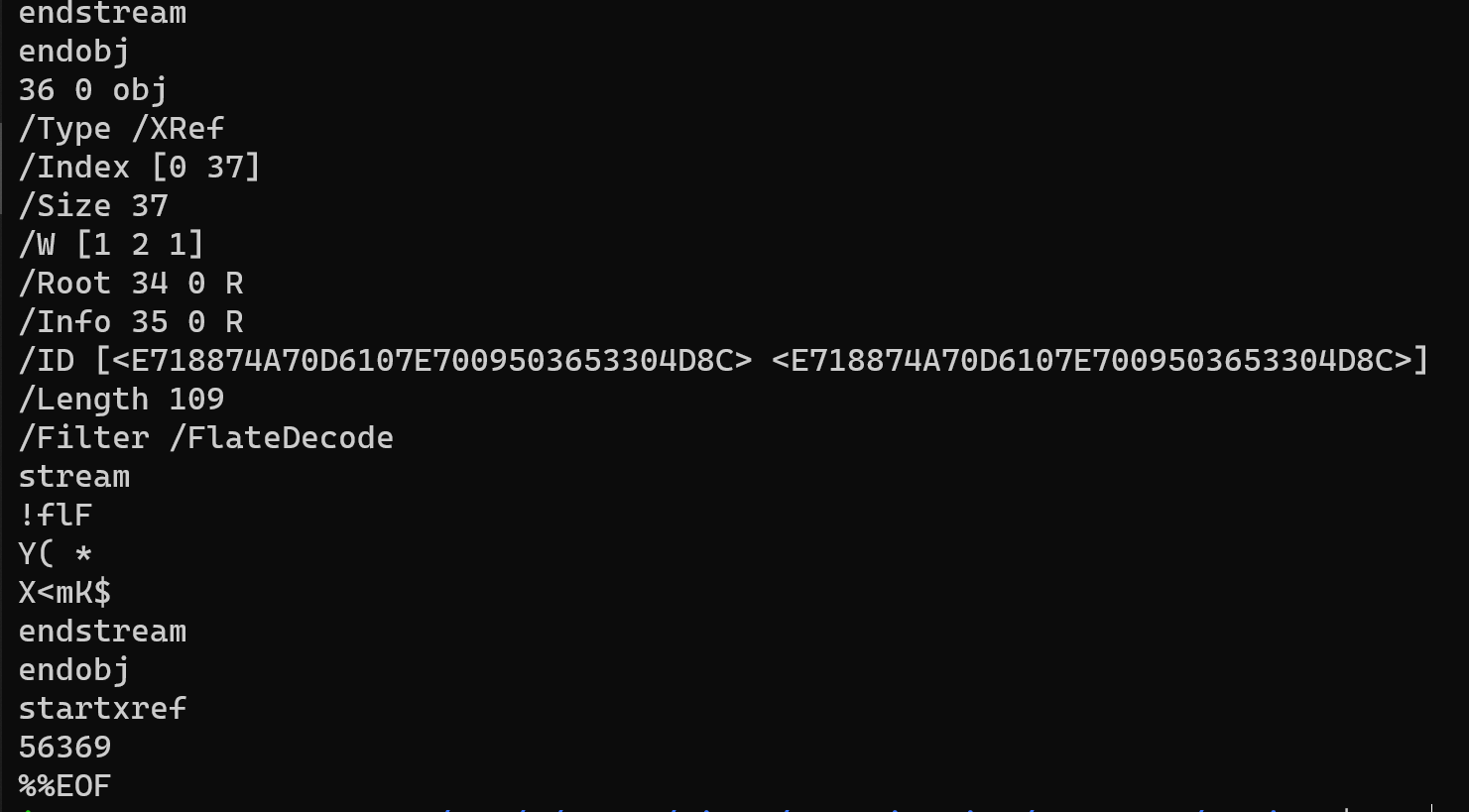
|  |  |
| --- | --- |
| Header | Contains the version information of the PDF file |
| Body | Sequence of indirect objects representing the contents of the document |
| Cross-Reference | Information that allows random access to indirect objects |
| Trailer | Allows readers to quickly find the cross-reference table |

*Figure 1. Structure of the PDF. Analogous to HTML, there are different compartments that constitute the PDF format that contain different categories of information on the PDF.*

****

*Figure 2. A screenshot if you run “cat a3q.pdf” command. Notice how the beginning starts as %PDF-1.5 notifying that this is a pdf format.*

The header shows what version of the PDF it is. Body component will often be the actual content of the file. Notice how there could be JavaScript incorporated in the body section of the PDF. The capabilities of JavaScript will further be discussed as many of the exploits take advantage of this.



*Figure3. Another screenshot showing at the very end of a PDF file. Notice How there is the /Type /XRef section and at the very end it ends with EOF.*

**JavaScript**

JavaScript is a scripting language that mostly used on frontend of the Internet. JavaScript has been added to the PDF in 2000 [reference] allowing the users to create a robust document that could incorporate programming capabilities and interact with any web page more easily. Whether the incorporation of the programming language served better than introducing more vulnerabilities is a debatable topic, as it will be discussed.

*Figure 4. A screenshot of a PDF document that incorporates JavaScript code.*

**Adobe Acrobat Reader**

The most widely used PDF viewer that Adobe has created and distributed. It allows various functionalities and gives more choice for paid versions of the software. Notice that the Adobe Acrobat Reader is not the only software that allows users to read PDF files. Browsers also provide the capability to read PDF files. Different software provides different capabilities of editing PDF files. This all depends on the operating system and the software that users are running on the machine.

**Vulnerabilities Categorization / Impactful Events**

The following discusses prominent vulnerabilities or hacks that are involved with PDF. It is not a comprehensive list of all vulnerabilities and patches and has been selected widely from different categories and different portions of the file exploited. There are various ways how a malicious PDF can reach a person. It is usually by people downloading the PDF file on their local machines. There are numerous PDF documents that could be the textbook or the paper that you needed for. In order to attack using PDF, the adversaries first need to find a way to compromise the software that reads the PDF file. Since users cannot read PDF files without the reading software, there is no way to open the file. Does that mean the users are safe if they never open the PDF file? Would a page in the Internet that is a PDF file safe? The answer is the moment you click on the link, the web browser becomes the users’ PDF reading software. Therefore, they do not necessarily need to download the file and open it with Adobe Acrobat Reader. The essence of a malicious PDF is using the rich text feature in the PDF file to exploit the PDF reader to read sensitive information or execute any malicious code. Adversaries will insert malicious code in the PDF file itself and would exploit user from the reader software. The following screenshot is from Adobe’s website showing the vulnerabilities that the company has discovered and patched so far. The vulnerabilities of PDF could be categorized as the following. Attackers have been exploiting using JavaScript, launch actions, embedded files in the PDF format, GoToE actions, embedded flash applications, encryption, and parser “flexibility”.

<https://www.pdfscripting.com/public/FreeStuff/PDFSamples/PDFS_CopyPastListEntries.pdf>

**JavaScript Exploits**

PDF has gained its fame by allowing rich text, graphics, and dynamic formats that allow users to produce forms that could allow users to input data effectively. The JavaScript engine in PDF are slightly different from the one used on the web. The different API serves to help users to modify PDF content dynamically or to control some viewer features. Potentially dangerous features are restricted for security reasons. However, this means that PDF documents are not purely static, and for example some actions may be used to fool a user (popups) or to send e-mails and HTTP requests automatically. Furthermore, experience shows that many recent vulnerabilities have been exploited using JavaScript in PDF. The following is one example or a use case that a user went through.

Example1 using JavaScript

In some kinds of malicious PDF attacks, the PDF reader itself contains a vulnerability or flaw that allows a file to execute malicious code. Remember that PDF readers aren’t just applications like Adobe Reader and Adobe Acrobat. Most browsers contain a built-in PDF reader engine that can also be targeted. In other cases, attackers might leverage AcroForms or XFA Forms, scripting technologies used in PDF creation that were intended to add useful, interactive features to a standard PDF document.

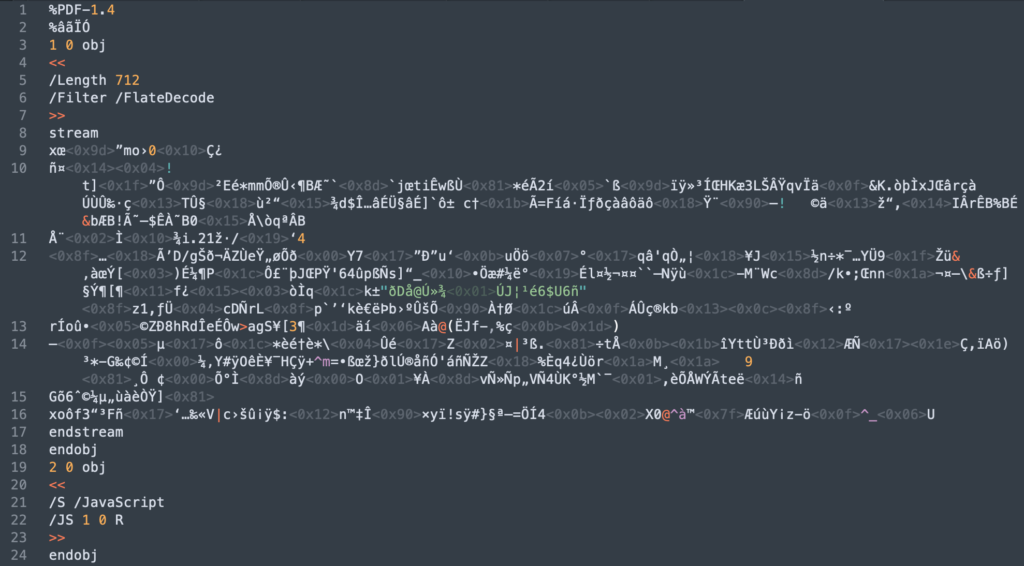


Figure. Shows the entire PDF file in 24 lines. Notice

However, with a bit of knowledge of PDF file structure, we can start to see how to decode this without too much trouble. The body or contents of a PDF file are listed as numbered “objects”. These begin with the object’s index number, a generation number and the “obj” keyword, as we can see at lines 3 and 19, which show the start of the definitions for the first two objects in the file:

1 0 obj  
2 0 obj

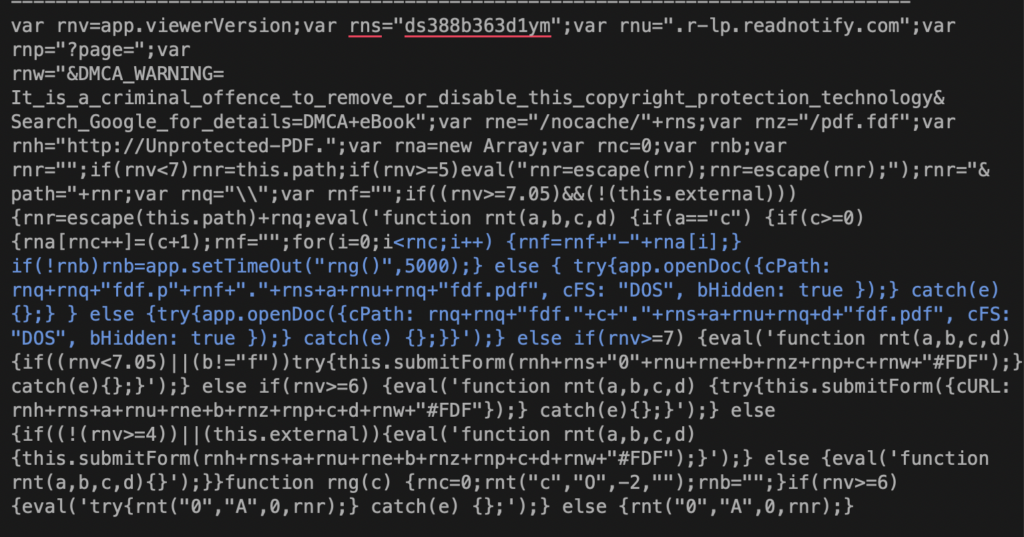
The end of each object is signalled with the keyword endobj, as seen at lines 18 and 24 for Object 1 and Object 2, respectively.

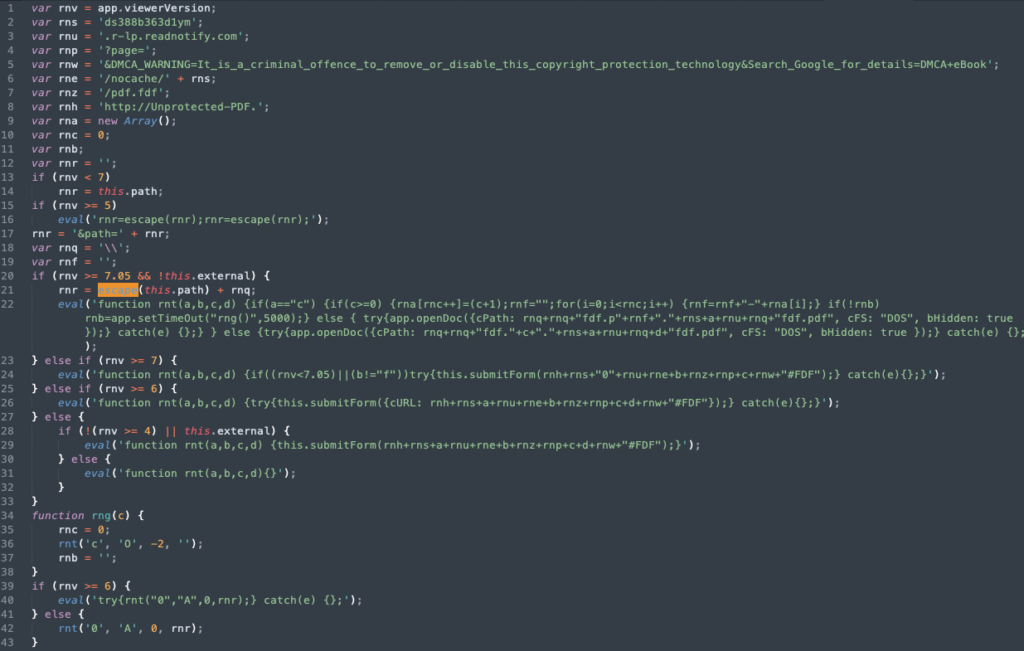
Object 2 immediately offers us some clues. We can see that it contains a dictionary (signalled by the chevrons << and >>. The dictionary has an entry for a JavaScript stream and a reference to Object 1:

JS 1 0 R

This tells us that the “garbage” code in Object 1 between the keywords stream (line 8) and endstream (line 15) is a JavaScript stream. Even better, Object 1’s dictionary is kind enough to tell us how to decode it. Line 6 specifies a “filter” of value “FlateDecode”. We can now write a quick-and-dirty Python script that decompresses the stream into plain JavaScript:

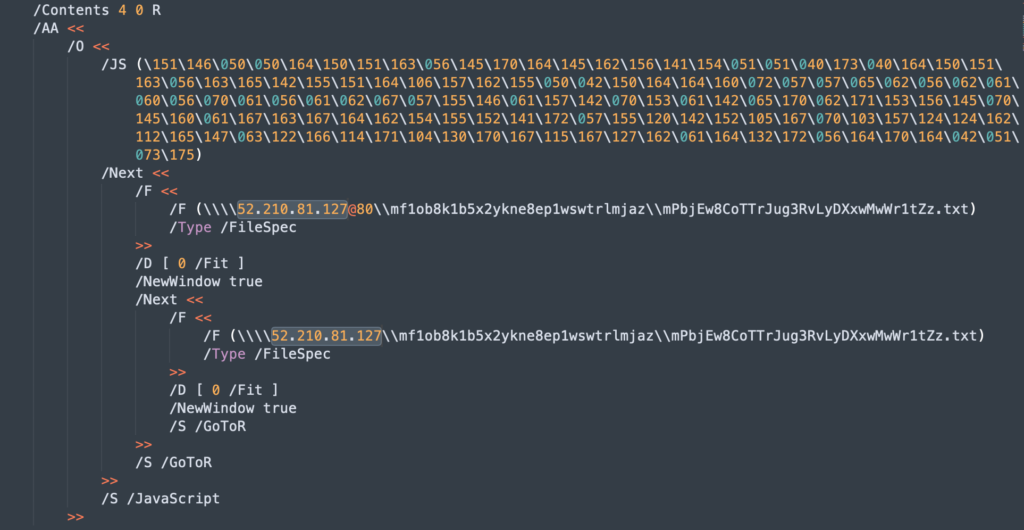
Our Python script churns out the JavaScript perfectly but not exactly beautifully:





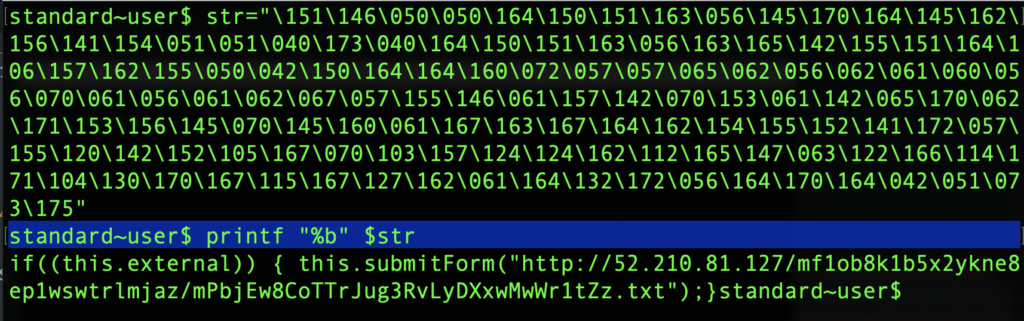
Now we can read the JavaScript and determine if it’s malicious or not. In this case, the code appears to be contacting a domain called “readnotify.com”. Making callbacks (“phoning home”) without user consent shows at least a lack of concern for user privacy. For people working in journalism or in politically sensitive areas this could be a serious issue, as this kind of callback can reveal the user’s IP address, operating system and browser version to a remote server.

As the image from VT makes clear, this is trojan that’s exploiting CVE-2018-4993. Let’s open it up and look inside.

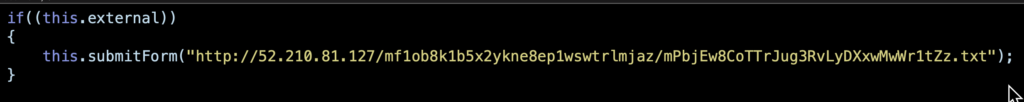


This is a very small file. There are only 4 objects, but the one that interests us is Object 3 and the value for the dictionary key /AA. Note that this contains a child dictionary with key name /O. That’s important because the /O key specifies actions that should occur when a document is opened. And the value of this key is itself another dictionary containing /JS, indicating yet again some encoded JavaScript.

Unlike our previous file, however, this one does not specify a filter. Luckily, the value of “JS” is clearly recognizable as octal encoding. Octal (or “oct”) uses three digits between 0 and 7 to specify a single value. The best thing about oct is we don’t need to roll up our Python sleeves to interpret it; we can just print it out directly on the command line:



As printf shows, the octals represent the same kind of JavaScript call that we saw in the previous example, leveraging the this.submitForm() function.



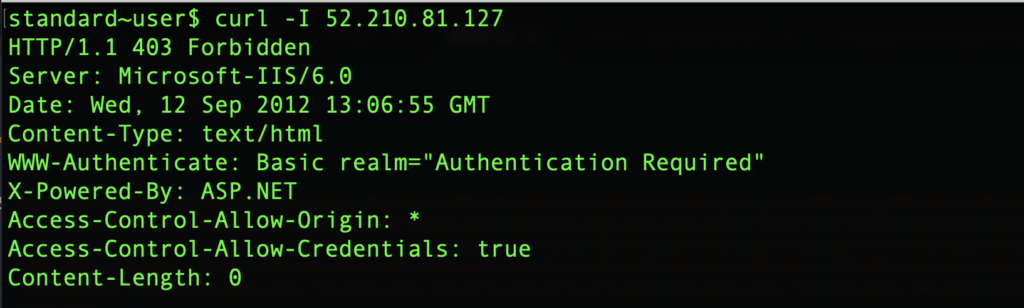
Going back to the /AA dictionary in the PDF, note the two lines which specify

/S /GoToR

This code issues the “Go To Remote” action, telling the reader application to jump to the destination specified under the /F key.

## Stealing Credentials with an SMB Attack

We can use cURL to grab the headers from that IP address to see what we can learn.



Looks like we need some authentication to get past the server, and that’s exactly where the danger lies for Windows users. If the attacker has set up the remote file as an SMB share, then the crafted PDF’s attempt to jump to that location will cause an exchange between the user’s machine and the attacker’s server in which the user’s [NTLM](https://docs.microsoft.com/en-us/windows/desktop/secauthn/microsoft-ntlm) credentials are leaked.

This happens because when a user tries to access SMB shared files, Windows sends the user name and a hashed password to automatically try to log in. Although the hashed password is not the user’s actual password, the leaked credentials can both be used to set up [SMB Relay](https://pentestlab.blog/tag/smb-relay/) attacks and, if the password is not [particularly strong](https://www.sentinelone.com/blog/7-signs-weak-password/), the plain-text version can easily be retrieved from the hash by automated password-cracking tools.

**Launch Actions**

A PDF file may launch any command on the operating system, after user confirmation (popup message). Different command lines may be specified for Windows, Unix and Mac. On Windows only, parameters can be provided for the command. Until Adobe Reader 9.3.2, the [CVE-2010-1240](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-12) vulnerability made it possible to fool users by modifying the text of the popup message. Since Adobe Reader 9.3.3, a blacklist restricts file formats that can be opened, blocking executable files by default (but [a way to bypass it](http://blog.bkis.com/en/adobe-fix-still-allows-escape-from-pdf/) has been found, and finally fixed in v9.3.4).

Examples from the link

**Embedded file**

A PDF file may contain attached files, which can be extracted and opened from the reader. This trick may be used to hide malicious executables in order to bypass some antivirus and content analysis engines. Fortunately, Adobe Reader refuses to open embedded files if their extension is part of a blacklist, such as EXE, BAT, CMD, etc. However, this blacklist is not perfect and formats such as HTML or Python scripts may be embedded in PDF and launched from Adobe Reader.

A way to execute the embedded file is the **GoToE Action function in the PDF.** A PDF file may be embedded inside another PDF file, and a GoToE action may be used so that Adobe Reader opens the embedded PDF file automatically without notifying the user. This feature may be used to hide a malicious PDF file within a normal PDF file, to fool many antivirus engines.

**Example**

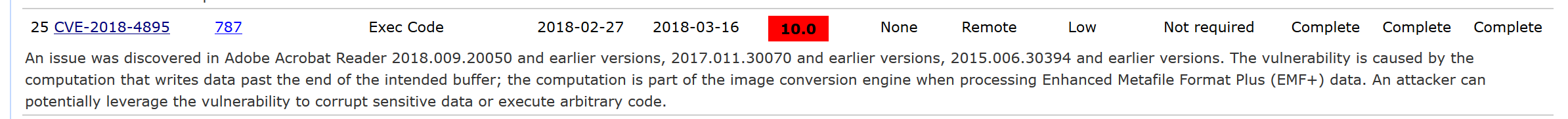
**Encryption**

A PDF file may be encrypted with a password. However, if an empty password is used, Adobe Reader will open it directly without asking the user. This trick may be used to fool many antivirus and analysis engines that do not support decryption.

**Parser "flexibility"**

PDF specifications, Adobe Reader and possibly other applications are very flexible about the structure of PDF files.

* + For example, most people think that PDF files have to start with the "%PDF" magic number, whereas the specifications only say this header has to be in the first 1024 bytes. See the [Adobe PDF 1.7 Reference, Appendix H.3, page 1102](http://www.adobe.com/content/dam/acom/en/devnet/pdf/pdf_reference_archive/pdf_reference_1-7.pdf#page=1102): "Acrobat viewers require only that the header appear somewhere within the first 1024 bytes of the file". It is therefore possible to insert around 1000 random bytes at the beginning of a PDF file. This trick may be used to bypass too strict antivirus or content analysis engines, because a fake file header (for example JPEG or HTML) can be inserted.
  + Another example is that the catalog at the end of the PDF structure may not point exactly to each object: Adobe Reader is able to reconstruct malformed files even if some content has been inserted within or between PDF objects.



An issue was discovered in Adobe Acrobat Reader 2018.009.20050 and earlier versions, 2017.011.30070 and earlier versions, 2015.006.30394 and earlier versions. The vulnerability is caused by the computation that writes data past the end of the intended buffer; the computation is part of the image conversion engine when processing Enhanced Metafile Format Plus (EMF+) data. An attacker can potentially leverage the vulnerability to corrupt sensitive data or execute arbitrary code.

**Miscellaneous**

**Embedded Flash applications**

A PDF file may contain Flash applications (stored as embedded SWF files), which bring their own security issues, such as ActionScript content and Adobe Flash Player vulnerabilities. Adobe Reader contains its own Flash Player, independent from the one installed in web browsers. For example the [CVE-2010-1297](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1297) vulnerability was first patched in the Flash Player on the [10 June 2010](http://www.adobe.com/support/security/bulletins/apsb10-14.html), whereas the Flash Player shipped with Adobe Reader was only patched on the [29 June 2010](http://www.adobe.com/support/security/bulletins/apsb10-15.html).

**Third Party Contributions**

One prominent project created by jesparza is the “peepdf” tool created in Python and is used to analyze whether a pdf could be harmful or not. There are numerous libraries and work done by people to mitigate the potential damage a PDF file could cause.

Web browsers have also constantly been pushing to support a more secure software not to get compromised any malicious PDFs. The key feature that they ensure is to provide the sandboxing environment.

**Defenses against Attacks**

As discussed in the vulnerabilities portion, often the vulnerabilities come from the software that allows you to read the pdf. If users are using Adobe Reader to read their PDFs, they should make sure they are updating their software whenever there are relevant security patches. If users are using their web browsers to read PDF files, they should regularly update their browsers to ensure that the code will not exploit the browser as well. Another way user could be careful is to be aware of the sources of the PDF that they are downloading from. Websites that pirate textbook pdfs or information could often contain such malicious codes. On the other hand, this could be coupled with a spear phishing attack that could come from e-mail attachments. Therefore, users should always check whom send the e-mail, and also to double check the format of the attached files.

**Conclusion**

So far, the prominent vulnerabilities and patches introduced by Adobe, and efforts from third party users to detect harmful pdf were discussed. PDF will continuously use be successful as it has been pushing its boundaries in terms of the graphical capabilities of it and the

Figure x. A rough screenshot what Google News return when “pdf hack” was searched. The materials are still very recent and relevant. Notice malicious e-mails are coupled with PDF.

The long history and exposure of PDFs often let the user’s guard down. The static nature of a PDF is rather deceiving, as the vulnerabilities exposed shows how the users would not be too suspicious of the file they received. These PDF vulnerabilities coupled with social engineering or phishing attacks even makes it more dangerous as the users do not suspect that the sender would send PDF files that has a malicious intent. This means that people will continuously try to exploit PDFs. As PDF has already established as the main document reader format, unless other companies or organization comes up with a good file format that provides a significant advantage over pdf files, people will continuously use the file format. As users, we should be aware of such vulnerabilities of PDF formats and take caution on what e-mail attachment we open or what documents we are downloading from. PDF will continue to be used, so there is really not much the users can do then to take caution and sanitize PDFs before they open them.

**References**

PDF related facts information

1. <https://itextpdf.com/en/blog/technical-notes/do-you-know-how-many-pdf-documents-exist-world>
2. <https://en.wikipedia.org/wiki/History_of_the_Portable_Document_Format_(PDF)>
3. <https://www.decalage.info/file_formats_security/pdf>
4. <http://www.malwaretracker.com/pdfthreat.php>
5. <https://www.investintech.com/resources/blog/archives/84-13-ways-the-pdf-is-vulnerable.html>
6. <https://www.welivesecurity.com/2018/05/15/tale-two-zero-days/>
7. <https://securityxploded.com/pdf_vuln_exploits.php>
8. <https://security.stackexchange.com/questions/64052/can-a-pdf-file-contain-a-virus>
9. <https://www.foxitsoftware.com/blog/understanding-security-vulnerabilities-in-pdfs/>
10. <http://pralab.diee.unica.it/sites/default/files/aisec09f-corona.pdf>
11. <https://www.zdnet.com/article/windows-10-or-mac-user-patch-adobe-reader-and-acrobat-now-to-fix-9-critical-security-flaws/>
12. <https://www.brightfort.com/pdffix.html>
13. <https://techcrunch.com/2010/07/23/an-analysis-of-a-pdf-exploit/>
14. <https://techxplore.com/news/2019-10-major-vulnerabilities-pdf.html>
15. <https://www.menlosecurity.com/blog/critical-vulnerability-in-firefox-built-in-pdf-viewer>

Vulnerability in the Firefox PDF viewer.

1. <http://www.h-online.com/security/features/CSI-Internet-PDF-timebomb-1038864.html>
2. <https://www.sentinelone.com/blog/malicious-pdfs-revealing-techniques-behind-attacks/>
3. <https://proprivacy.com/privacy-news/pdf-security-guide-vulnerable-exploit>
4. <https://www.cvedetails.com/vulnerability-list/vendor_id-53/product_id-497/Adobe-Acrobat-Reader.html>
5. <https://www.cvedetails.com/product/497/Adobe-Acrobat-Reader.html?vendor_id=53>
6. <https://blog.didierstevens.com/2010/03/29/escape-from-pdf/>
7. <https://nsfocusglobal.com/chrome-pdf-file-parsing-0-day-vulnerability-threat-alert/>
8. <https://www.forbes.com/sites/zakdoffman/2019/10/05/critical-pdf-warning-new-threats-leave-millions-at-riskupdate-all-pdf-apps-now/#4afb585f739d>

Tools

1. <https://github.com/jesparza/peepdf>