SECURITY HUBS

WE SAY WHAT WE DO AND DO WHAT WE SAY

PDF Documents Metadata and Practical Examples of How to Handle It

Version 1.1, August 2021

Security Hubs, Inc.
2093, Philadelphia Pike
Claymont, Delaware
United States, 19703
302 330 2121
contact@securityhubs.io

Table of Contents

1. Introduction

1.1. Metadata

Wikipedia defines Metadata to be "data that provides information about other data". Overall, there are six distinct types of metadata, descriptive Metadata, legal Metadata, administrative Metadata, reference Metadata, statistical Metadata, and structural Metadata.

From all the above, one of the most important is descriptive Metadata that describes information about a resource, and it is used for discovery and identification with elements such as title, abstract, author, and keywords coming into the picture.

Often Metadata is used to enhance data reuse, and it plays an important role in the data discoverability process and data relationships.

As an example, the OpenDataSoft company describes that Metadata represents the following information:

- What
- When
- Where
- Who
- How
- Which
- Why

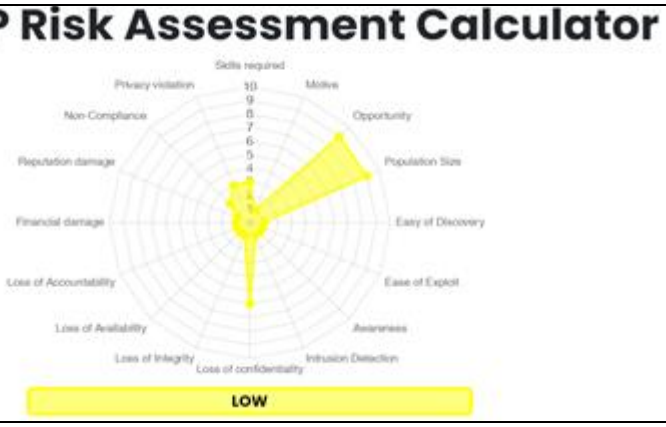
1.2. Background

In 2008, Larry Pesce published for SANS a whitepaper named "Document Metadata, the Silent Killer...".

Overall, his paper represents by far the most comprehensive guide for anyone looking to understand the risks associated with the exposure to unwanted metadata information within published resources. In addition, the document outlines several techniques and tools that can still be used.

The scope of this blog post is not going too deep in that direction, though. Instead, we will explore a couple of technical solutions that any business can consider during the process of limiting information exposure through its public documents.

This finding is often underrated and is usually missing completely from an Appsec pentesting engagement report. However, when this issue is detailed, it often does look like this.

Name	***Document Metadata Section Exposes Information***
Affected resource(s)	
Technical details	
Impact Level	
Reproduction steps	
Recommendations	**CWE-200**: Exposure of Sensitive Information to an Unauthorized Actor - https://cwe.mitre.org/data/definitions/200.html **CWE-1230**: Exposure of Sensitive Information Through Metadata - https://cwe.mitre.org/data/definitions/1230.html

Today, more than 70% of the publicly exposed documents are in pdf format, so that we will focus only on this format.

As a client, you might ask, okay, is there any easy way to deal with this matter without breaking the bank? Well, yes, you can consider a couple of options we are trying to detail further.

2. Practical Solutions

2.1. Using Bash

In 2017, Josh Lemon released a whitepaper detailing a process of handling the PDF metadata information exposure. Assuming the **qpdf** is already installed on an Ubuntu machine, his blog post contains all the steps necessary to script the automation, so here is the barebone script for use.

```
#!/usr/bin/env bash

# WORD AHEAD
# This represents a barebone bash script that does a basic metadata cleaning job. Nothing more, nothing
less.

# You DON'T like it, then DON'T use it.

# Tested on Ubuntu 18.04+
# Remeber to check for exiftool and qpdf libraries and install them before trying to use it.


clear
read -p "[-] Please enter the file name:= " FILE
echo "[-] Initializing step 1..."
qpdf ${FILE} ${FILE}_CLEAN
echo "[-] Initializing step 2..."
exiftool -all:all= ${FILE}_CLEAN
echo "[-] Initializing step 3..."
qpdf --linearize --encrypt "" "<YOUR_INSANE_LONG_PASSWORD>" 128 --print=full --modify=none --extract=n
--use-aes=y -- ${FILE}_CLEAN ${FILE}_P
echo "[-] Initializing step 4..."
mv ${FILE} ${FILE}.bck
echo "[-] Initializing step 5..."
mv ${FILE}_P ${FILE}
echo "[-] Verifying file"
exiftool ${FILE}
```

Cons

It seems the final output might be reversed under specific circumstances. While it does its job, we don't consider this solution a good fit for corporations or enterprise-grade businesses.

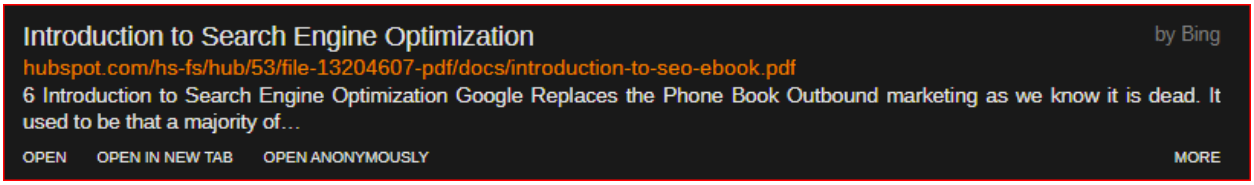
2.2. Using Python PyPDF2

Luckily, Python does have various libraries that allow PDF manipulation, PyPDF2, ReportLab, pdfkit, etc.

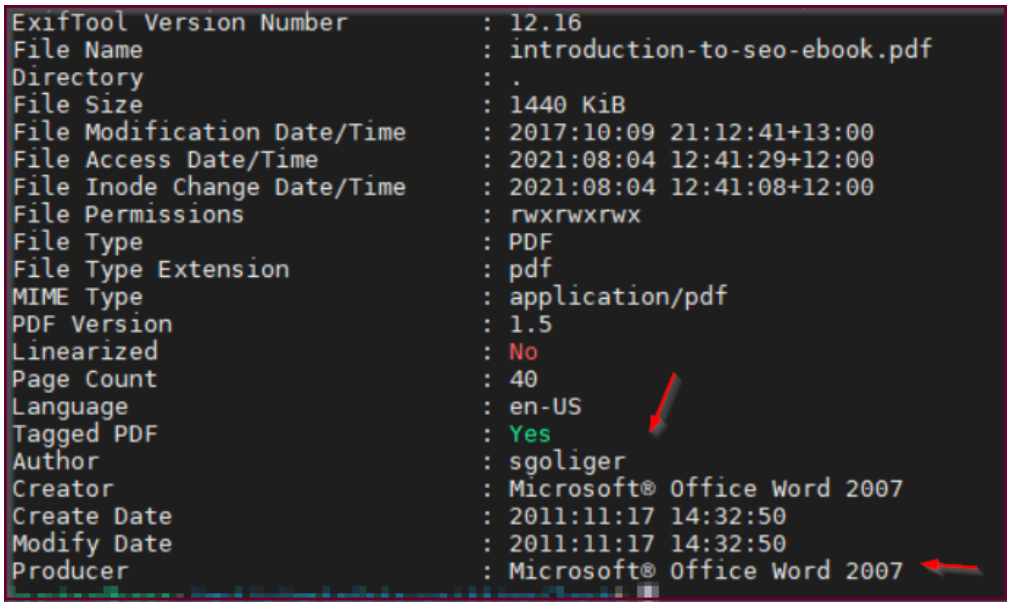
All of them are great libraries, but we found PyPDF2 friendly enough during the goal of achieving PDF metadata changes.

For the sake of this article, we will randomly pick a CRM company, like HubSpot. For the sake of this article, we will randomly pick a CRM company, like HubSpot. They are one of the major players in their industry, with many public documents published.

We picked up a file named "***Introduction to SEO eBook***" through a quick search engine wizardry. Good content, by the way, and the people wrote it did a great job.



Checking the metadata of the file, we got the following:



To summarize the info, the original file is from 2011, initially wrote in Microsoft Word 2007, and then exported in PDF format by ***sgoliger***, as its Author. The current details expose timestamps associated with 2011 and an obsolete 2007 MS Office version; beyond that, the document proved to be a good candidate pinpointing the amount of information stored inside the metadata fields.

Hint

*If the Producer field returns "**Skia/PDF m83**", the PDF was exported using Google Docs. More details can be found in the **Reference** section.*

Ideally, any business should sanitize all its documents before publishing. The internet is a vast ocean, and while you can still take down and remove some of your content, the process is complicated and time-consuming.

Altering a PDF metadata through a Python script can be achieved through a snippet like this:

```python
#!/usr/bin/env python3

from PyPDF2 import PdfFileReader, PdfFileWriter

.....

fin = open('your_original.pdf', 'rb')
reader = PdfFileReader(fin)
writer = PdfFileWriter()

writer.appendPagesFromReader(reader)
metadata = reader.getDocumentInfo()
writer.addMetadata(metadata)

# Write your custom metadata here:
writer.addMetadata({
    '/Creator': '',
    '/Creator Tool': '',
    '/Document ID': '',
    '/Instances ID': '',
    '/Author': ''
})

fout = open('your_original.pdf', 'ab')
writer.write(fout)

fin.close()
fout.close()
```

Cons

We tested this option quite intensively, and it was found that the results were not consistent. Also, the pypdf2 project does not seem maintained actively, with a plethora of forks and derivates.

From the supply chain security perspective, adopting this solution might not be the best call, so it is up to you if you want to dig deeper or not.

2.3. Using Python + PDFTK

We remembered that every story has three heroes back in childhood, so here is the third potential practical solution. We will use Python again, just because it is much easier to understand and deal with. There are other ways of doing it, and you can use whatever makes you comfortable to get the job well done.

This time, we will not import any python pdf libraries but invoke an external program, pdftk (on an Ubuntu 18.04+, "***sudo apt install pdftk***" should do the trick).

```python
#!/usr/bin/env python3

import subprocess, sys

def overrideMetadata():
    """ This function deals with the metadata overriding process """

    cleanFile = targetFile[:-4] + '_CLEAN' + '.pdf'
    try:
        subprocess.run(['pdftk', targetFile, 'update_info', 'clean_metadata.txt', 'output', cleanFile])
    except OSError:
        print("[!] Cannot generate the clean_metadata file.")

def collectMetadataInfo(targetFile):
    """ This function extracts all the metadata forma pdf file """

    subprocess.run(['pdftk', targetFile, 'dump_data', 'output', 'metadata_info.txt'])

def main():
    """ The main function """

    def writeInFile():
        """ This function writes the metadata back to the PDF file """
        fileHandle.write('%s\n' % i)

    try:
        file = open('metadata_info.txt', encoding='utf-8')
    except  OSError:
        print("[!] Cannot open the metadata file, maybe was not created properly?")
    else:
        lines = file.read().splitlines()

        with open('clean_metadata.txt', 'w') as fileHandle:
            print("[!] Cannot open the clean metadata file!")
            for i in lines:
                if "InfoValue:" in i:
                    print("[!] Found one and shredded -> " + i + '\n')
                    i = "InfoValue:"
                    writeInFile()
                else:
                    writeInFile()
        file.close()


targetFile = sys.argv[1]

collectMetadataInfo(targetFile)

main()

overrideMetadata()
```

Cons

We cannot pinpoint any at this point. However, additional testing might be required. What worked for us does not necessarily work for you. 😊

Q. Is there any way to automate this process even further?

A. Yes. It can be done, and an option on the table would be using GitHub's "Actions" feature. We performed some testing and the execution time for a 100 pdf files list was under one minute. That gives for free 2000 minutes free execution time, say, more enough to automate the process.

Note

A barebone Github Actions YAML file skeleton might look like this. All you need to do is to replace the missing bits with your code, and it should be ready to roll.

```yaml
name: pdf-metadata-fixer

on:
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repo
        uses: actions/checkout@master

      - name: Seting up things and install missing components if any
        run:  sudo apt install -y unzip pdftk
        shell: bash

      - name: Reading the PDF file(s) from the target directory
        run:  |
        [YOUR BASH MAGIC]
        shell: bash

      - name: Create local changes
        run: |
          git add output/directory_listing.txt

      - name: Commit results to Github
        run: |
          git config --local user.email "your@user.email"
          git config --global user.name "your_github_user"
          git commit -m "PDF-Metadata-Fixer_Report" -a --allow-empty

      - name: Push changes
        uses: ad-m/github-push-action@master
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          branch: ${{ github.ref }}
```

3. Conclusion

As mentioned before, metadata provides a substantial set of information that could be used to develop further attack vectors. Also, the metadata can be used beyond the scope of a cybersecurity attack and for competitive intelligence. However, this aspect might be covered in the next parts.

This blog post is aimed at beginners. It does not present any novel research and only serves as an introduction, covering the basics and offering simple, practical free solutions to address the metadata information exposure.

4. References

- https://securityhubs.io/pdf_documents_metadata_and_practical_examples_of_how_to_handle_it
- https://sansorg.egnyte.com/dl/FU8X6REqUY/
- https://docs.github.com/en/actions
- https://www.digitalcitizen.life/remove-metadata-file/
- https://www.cnet.com/tech/services-and-software/remove-metadata-from-office-files-pdfs-and-images/
- https://blog.joshlemon.com.au/protecting-your-pdf-files-and-metadata/
- https://www.giac.org/paper/gpyc/400/pdf-metadata-extraction-python/170602
- https://github.com/JavierOlmedo/OWASP-Calculator
- https://stackoverflow.com/questions/60876952/use-python-to-determine-if-pdf-was-generated-by-google-docs

5. About Security Hubs

Security Hubs is a US-represented, tightly knit, client-focused IT security consultancy, helping businesses globally from Auckland HQ, New Zealand.

We have extensive experience in the United States and Australasia markets, providing a superior alternative to the current penetration testing approach.

Security Skills as a Service model emerged as the best of both worlds between standard penetration testing and bug bounty programs and is powered by an on-demand exclusive global network of security engineers.

We are highly specialized in providing:

- Web / API Application Penetration testing
- Infrastructure Penetration Testing - Mobile Security Testing
- Fat Client | Desktop Application Security Testing
- Security Source Code Review
- Vulnerability Assessment
- Threat Modelling
- Cloud Config Security Review
- Cloud Architecture Security Review
- Cloud Adoption and Migration Complete Services

6. Copyright

7. Disclaimer

While Security Hubs has attempted to ensure the information in this whitepaper is as accurate as possible, the information is for personal and educational use only. It is provided in good faith without any express or implied warranty. There is no guarantee given to the accuracy or currency of the information contained in this White Paper. Security Hubs does not accept responsibility for any loss or damage occasioned by using the information contained in this whitepaper.

Page intentionally left blank