A Multi-leveled Approach for Detection of Coercive Malicious Documents Employing Optical Character Recognition

Josiah Smith
A Multi-leveled Approach for Detection of Coercive Malicious Documents Employing Optical Character Recognition

GIAC (GCIH) Gold Certification

Author: Josiah Smith, josiahraysmith@gmail.com
Advisor: Lenny Zeltser

Accepted: February 12, 2021

Abstract

Authors of malicious documents often include a graphical asset used to lure the potential victim to “enable editing” and to “enable content” to activate the macro’s embedded logic. While these graphical lures vary in theme, language, and content, they commonly have similar coercive text. Using Optical Character Recognition to produce text files of the images provides the ability to anchor the images’ contents. While attackers have been known to intentionally manipulate images to bypass OCR-based detection, some additional techniques can surface the textual contents. Optical Character Recognition can be utilized to track, pivot, and cluster malicious campaigns, identify new TTPs, and possibly provide attribution against adversaries.
1. Introduction

There is no question that document-based malware is a popular attack vector that puts an organization’s posture at risk. *A Multi-leveled Approach for Detection of Coercive Malicious Documents Employing Optical Character Recognition (OCR)* details the attempts to evade OCR-based detection by manipulating embedded graphical assets while maintaining the subtlety of the lure.

OCR is a technology used to identify text within images. OCR-based detection is complementary to other detection methodologies used by security operations teams. Based on research of modern attack campaigns, multiple threat actors use these malware lures to convince users to enable active content. Detection at this point can potentially break the kill chain and prevent the next organizational data breach.

This testing methodology in this research is defined by utilizing a constant set of graphical assets to measure the effectiveness of two different OCR products on the images. The resulting text is processed with a Yara rule designed to detect coercive language instructing users to activate the embedded logic. The images will then be modified through multiple techniques ranging from transparency, colors, blurriness, and font types, and then processed through the OCR engines and evaluated again. In addition to the pre- and post-manipulation results, some alternative detection techniques are assessed for use in the detection chain.

While some images are intentionally manipulated images to bypass OCR-based detection, additional techniques can surface the textual contents. Optical Character Recognition can be utilized to track, pivot, and cluster malicious campaigns, identify new TTPs, and possibly provide attribution against adversaries.

2. Optical Character Recognition

Optical Character Recognition has become an important and widely used technology. Among its many practical applications are money changing machines, mobile device check deposit, office scanners, and automation within the postal and shipping industry. Although the technology has been a topic of interest and research for many years, developing OCR with capabilities comparable to human interpretation is a sustainable challenge. OCR technology

Josiah Smith, josiahraysmith@gmail.com
presents a complex problem set due to the variety of languages, fonts, and styles in which text can be written. Other nuances ranging from image quality to intentional tampering can cause inconsistent results throughout the process.

The process of OCR is a complex activity that can be described as a sequence of phases. As described by Islam, Islam, and Noor (2017), the steps are as follows:

- Image acquisition: The process of acquiring the image.
- Pre-processing: Enhances the quality of the image
- Character segmentation: Separates the image into component characters
- Feature Extraction: Extracts other features from the image
- Classification: Classifying characters into their appropriate category
- Post-processing: Improve accuracy of results

While there are many intricacies and various approaches across different OCR technologies, this research’s scope will cater to the application of detecting coercive lures for malicious documents. The two specific OCR engines that will be described and utilized to derive text from the graphical lures are Tesseract and iDRS.

Tesseract is an open-source text recognition engine that is freely available under the Apache 2.0 license (Tesseract, 2020). The tool can be used directly from the command line, through its full-featured API, or various third-party contributed graphical user interfaces. Additionally, there are dozens of languages supported throughout its versions.

The installation process for Tesseract is relatively straightforward and contains two portions to include the engine itself and the training data for the selected language. The package is directly available for many Linux distributions. Simply running the following commands will install on Ubuntu systems:

```bash
# apt install tesseract-ocr
# apt install libtesseract-dev
```

The second OCR engine used to extract text from malware graphical lures is iDRS from IRIS. The iDRS OCR toolkit is a commercial application that offers the capability to convert all images into indexed and editable files. Considering the installation is a bit more extensive, and the installation media is not freely available since it is a licensed application, the installation steps are not detailed.

Josiah Smith, josiahraysmith@gmail.com
It is important to note that while there is an opportunity to compare the two OCR engines that will be utilized as tools to extract the textual content from graphical lures, this research intends not to compare the two products. Instead, they will be used as standard toolsets for the experimentation of detecting malware lures utilizing OCR technologies. Additionally, there are many different pre-processing features of both engines that could be tuned to achieve higher efficacy of text extraction and identification. When considering each toolset’s extensiveness and consistent effects, the default configurations will be used and will not be tuned throughout the process.

### 2.1. Malware Graphical Lures

After installing the OCR engines, the next step in the research methodology is to acquire samples of coercive graphical lures used in malicious documents. There are many ways to address this step which range from an automated to a manual approach. One solution to acquire the coercive lure without opening the weaponized document is downloading the image from some repositories or websites focused on this topic. A particular relevant source of these malware lures is Dr. Josh Stroschein’s maldoc template GitHub repository (Stroschein, 2021). Another thoroughly developed source for images to perform OCR on is the Malware Lures Gallery from InQuest (InQuest, 2020).

If the image’s extraction is sourced directly from a document, there are a few different tools and techniques to extract them, but the effort and effectiveness are dependent on that specific file type. In the case of .docx and .docm files, those are actually disguised zip files, and the images are found within the media directory of the unzipped contents. The following bash command will loop through every .docx file and extract the images to their respective and newly-created directory.

```bash
for L in *.doc*; do mkdir -p "$L"-images && unzip "$L" "word/media*" -d "$L"-images; done
```

The process to extract the images is very similar for .xlsx files. The differing aspect is the images are stored in the “xl/media/” portion of the unzipped document. The following script can automate the extraction of images from multiple files.

Josiah Smith, josiahraysmith@gmail.com
#!/bin/bash
for L in *
do
    mkdir -p "$L"-images
    unzip "$L" "xl/media/*" -d "$L"-images
done

While the image extraction from the aforementioned file types proved trivial, the extraction from other file types like .doc can be a bit more challenging due to the streams associated with the Object Linking and Embedding (OLE) file format. Ogden, Roberts, and Sayre (2018) describe OLE, “a complex binary format that is not easily manipulated”, but represent the compressed archives formats’ flexibility for manipulating the contents of the file.

Another favorite tool to assist in carving images out of the documents is Foremost. Originally developed by the United States Air Force Office of Special Investigations and the Naval Post-Graduate School Center for Cybersecurity and Cyber operations, Foremost is a console program to recover files based on their data structures. One practical use-case shown by Smith (2020) is the extraction of executables embedded within images. However, this use case will show the carving images from documents. Looking at the malicious document b93c1b5898ee3d02d1f7996c90256099 (VirusTotal, 2021) masquerading as Ginny Hergott’s resume, Foremost carves out one image.

```
$ foremost b93c1b5898ee3d02d1f7996c90256099 && md5sum output/{png,jpg}/* 2> /dev/null
0e822212fa479958692131134e28de53 output/png/00000008.png
```

Figure 1 shows the rendering of the extracted .png.

Josiah Smith, josiahraysmith@gmail.com
Another approach to carving out the images found within OLE files is Didier Stevens’s oledump.py (Stevens, 2021). However, this technique is generally more difficult as the stream containing the image is unknown at the beginning of the analysis, and on occasion, there is some data or other chaff bytes ahead of the image’s magic. Demonstrating with the file md5: aaa33b12b551840719b1df0d6a8ac731 (VirusTotal, 2021), it is apparent that some bytes need to be removed before the image can be rewritten. Furthermore, while earlier analysis found the data in stream 5 of 14, it is necessary to identify the correct stream.

```
$ oledump.py -a -s 5 aaa33b12b551840719b1df0d6a8ac731 | more
00000000: 9E 3F 00 00 44 00 64 00 00 00 00 00 00 08 00 ..D........
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 51 26 C0 11 ........Q&..
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
00000050: 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
00000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........
00000070: 06 01 02 00 00 00 00 FF 01 00 00 08 00 36 00 37 00 .........6.7.
00000080: 31 00 32 00 38 00 30 00 30 00 00 00 00 00 00 00 10 F0 1.2.8.0.2....
00000090: 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..../.......@
000000A0: 06 06 0B 86 88 8A 37 D9 75 0B 03 08 A5 2B DC C1 4A ....7.u....+.J
000000B0: 59 92 FF 00 DA 3E 00 00 01 00 00 00 44 00 00 00 Y......D..n.....
000000C0: 00 00 7C 00 00 00 00 1B F0 D2 3E 00 00 8B 86 9A 37 ....|......7
000000D0: D9 75 OB 03 08 A5 2B DC C1 4A 59 92 FF 89 50 4E u....+.JY..PF
000000E0: 47 0D 08 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 02 G.........IHDR...
000000F0: 8E 00 00 01 2F 08 06 00 00 00 CB A6 40 36 00 00 ....../......@
```

Josiah Smith, josiahraysmith@gmail.com
Overcoming this uncertainty is simple enough with some rudimentary scripting skills. The following Python script will match, carve, and write the embedded PNG image back into the current working directory.

```python
#!/usr/bin/env python3
import re
import sys
with open(sys.argv[1], "rb") as fp:
    contents = fp.read()
    match = re.search(b"\x89\x50\x4e\x47\x0d\x0a\x1a\x0a", contents)
    if not match:
        sys.exit("No match")
    with open("carved_image", "wb") as carved_image:
        carved_image.write(contents[match.start():])
    sys.exit("PNG Image Found")
```

There is still a problem identifying which stream to look for the image data, but a simple BASH loop here will write out all the streams to be looped through the PNG image finder.

```bash
$ for L in `oledump.py -d aaa33b12b551840719b1df0d6a8ac731 | cut -d: -f1`; do oledump.py aaa33b12b551840719b1df0d6a8ac731 -d -s$L > stream$L; done

$ ls stream*| while read L; do ./pngfind.py $L; done
No match
No match
No match
No match
No match
No match
No match
No match
No match
No match
No match
No match
PNG Image Found
No match
No match
No match
No match
No match

$file carved_image && md5sum carved_image

carved_image: PNG image data, 654 x 303, 8-bit/color RGBA, non-interlaced
7490be961ba7e861590271c6e033f431 carved_image
```

The resulting image shown in figure 2 was found in the stream and carved with oledump.py after removing the chaff bytes at the head of the data stream.

Josiah Smith, josiahraysmith@gmail.com
2.2 YARA Detection Logic

Now that multiple approaches have been described for the acquisition of images from documents, the next step is to perform OCR on the images with the two engines and acquire the first pass results. The variable measured is the output of the OCR tools from multiple controlled images and two different OCR products. After impacting the detection from manipulating images, the detection logic is modified to detect the images. The following YARA rule can detect malware lure coercion and may be improved to cover the difference in output.

```yara
rule OCR_Rule {
    strings:
    $enable1 = "Enable Content" nocase ascii wide
    $enable2 = "Enable Editing" nocase ascii wide
    condition: any of ($enable*) and filesize < 1KB
}
```

Figure 2: 7490be961ba7e861590271c6e033f431
3. Pre-manipulation

3.1. Image 1 Pre-manipulation

The first image analyzed with OCR and the YARA based detection rule will be the following MS Office Word-themed Lure originally derived from file md5: 138e3164dd69df539809a0bc5cd37d36. As previously discussed, this Microsoft Word 2007+ file can have the image extracted by unzipping the file and searching in the media directory.

```
$ unzip 138e3164dd69df539809a0bc5cd37d36 "word/media*" -d images
Archive:  138e3164dd69df539809a0bc5cd37d36
extracting: images/word/media/image1.png
```

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td>W Document created in earlier version of MS Office Word To view this content, please click “Enable Editing” from the yellow bar and then click “Enable Content”</td>
<td>Document created in earlier version of MS Office Word To view this content, please click “Enable Editing” from the yellow bar and then click “Enable Content”</td>
</tr>
</tbody>
</table>

**YARA Signature Hits**

```
$ yara OCR.rul image1.png.ocr -s
OCR_rul image1.png.ocr
0x96:$enable1: Enable Content
0x5f:$enable2: Enable Editing
```

```
$ yara OCR.rul image1.png.txt -s
OCR_rul image1.png.txt
0x93:$enable1: Enable Content
0x5f:$enable2: Enable Editing
```

**Coercive Detection Success**

Yes

Yes

Figure 3: Image 1 Summary

3.2. Image 2 Pre-manipulation

The second image that will be analyzed with OCR and the YARA-based detection rule will be the following MS Office 365 originally derived from file md5: 12453bda42e40501077b1d202a338648.

Josiah Smith, josiahraysmith@gmail.com
The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the template.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>'mOffice 365</td>
<td>You are attempting to open a file that was created in an earlier version of Microsoft Office. If the file opens in Protected View, click Enable Editing, and then click Enable Content</td>
<td>n</td>
</tr>
<tr>
<td>&quot;Office365&quot;</td>
<td>You are attempting to open a file that was created in an earlier version of Microsoft Office. If the file opens in Protected View, click Enable Editing, and then click Enable Content</td>
<td>Office365</td>
</tr>
<tr>
<td>YARA Signature Hits</td>
<td>$ yara ../OCR.rul image1.jpg.ocr -s OCR_Rule image1.jpg.ocr 0xbb:$enable1: Enable Content 0x9c:$enable2: Enable Editing</td>
<td>$ yara ../OCR.rul image1.jpg.txt -s OCR_Rule image1.jpg.txt 0xbd:$enable1: Enable Content 0x9e:$enable2: Enable Editing</td>
</tr>
<tr>
<td>Coercive Detection Success</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 4: Image 2 Summary

3.3. Image 3 Pre-manipulation

The third image analyzed with OCR and the YARA based detection rule was the following Amazon gift card-themed Lure originally derived from file md5: 678aff9ae66a0fb94043261eeecae5. As previously described, the embedded image is extracted with oledump.py and pngfind.py.

Josiah Smith, josiahaysmith@gmail.com
The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ oledump.py 678aff9ae66a0fb94043261eeecae5 -d -s5 &gt; stream5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ ./pngfind.py stream5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ file output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output: PNG image data, 1017 x 1092, 8-bit/color RGBA, non-interlaced</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The fourth image analyzed with OCR and the YARA based detection rule was the following Happy New Year - Apple Store Gift Voucher Lure originally derived from file md5: 5e870ca9e50adec114a98eab76f5b49c.

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

<table>
<thead>
<tr>
<th>Coercive Detection Success</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
</table>

Figure 5: Image 3 Summary

3.4. Image 4 Pre manipulation

The fourth image analyzed with OCR and the YARA based detection rule was the following Happy New Year - Apple Store Gift Voucher Lure originally derived from file md5: 5e870ca9e50adec114a98eab76f5b49c.

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.
A Multi-Leveled Approach for Detection of Coercive Malicious Documents
Employing Optical Character Recognition

Josiah Smith, josihaysmith@gmail.com

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
</table>
| ![Image](image1.png) | - d AppleStore
"Gift Voucher
figggy New Year
MWH$MWX$$L
I
- Office
This document created with Microsoft Office newest version; """"
Please, press """"Enable Content"""" to resolve.
"Fit-hum"
Adam Information
Invoice Number Ivoice Dale terms Ship Ode
"13003 mm""""
Tìsorderisswietbflwe'sSdsadRemPoiies.
, w1m.wecWWies.hm . | 9 Apple Store Gift Voucher
You make us proud every year.
•
ice
This document created with Microsoft Office newest version.
Please, press """"Enable Content"""" to resolve.
For a total of $25.50
Additional Information
Invoice Nm DPI
Invoice Date
111UIIS
Ship Date
9299613083
Dec, 2020
Credit Card
1 his order is suit,ect to Apple's Sales and Refunds Policies [http://store apple.com/CatalogfJ S/11P;iges.1s;itespo11c1es
html](http://store apple.com/CatalogfJ S/11P;iges.1s;itespo11c1es.html) |

<table>
<thead>
<tr>
<th>YARA Signature Hits</th>
<th>$ yara ../OCR.rul image1.png.ocr -s OCR_Rule image1.png.occr 0xbb:$enable1: Enable Content</th>
<th>$ yara ../OCR.rul image1.png.txt -s OCR_Rule image1.png.txt 0x99:$enable1: Enable Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coercive Detection Success</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 6: Image 4 Summary

3.5. Image 5 Pre-manipulation

The fifth image analyzed with OCR and the YARA based detection rule was the following MS Office Word-themed Lure originally derived from file md5:

b1d4a944e6a71d17037664135b3498eb. Unfortunately, detection capability was foiled by introducing Spanish text within the coercive lure.

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure. While the OCR engines appeared to efficiently carve out the text of “Abilita modifiche” and “Abilita contenuto”, the YARA rule did not include those strings. This situation is an excellent example of identifying False Negatives “in the wild” and using them to improve the efficacy of an organizations signature set.
A Multi-Leveled Approach for Detection of Coercive Malicious Documents Employing Optical Character Recognition

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Questo file (*9 state creato con una versione precedente di Microsoft Office Word</td>
<td>Questo file è stato creato con una versione precedente di Microsoft Office Word</td>
</tr>
<tr>
<td></td>
<td>Per visualizzare il contenuto è necessario fare clic sul pulsante &quot;Abilita modifiche&quot;, situato sulla barra gialla in alto, e poi cliccare su &quot;Abilita contenuto&quot;</td>
<td>Per visualizzare il contenuto è necessario fare clic sul pulsante &quot;Abilita modifiche&quot;, situato sulla barra gialla in alto, e poi cliccare su &quot;Abilita contenuto&quot;</td>
</tr>
<tr>
<td>YARA Signature Hits</td>
<td>No hits</td>
<td>No hits</td>
</tr>
<tr>
<td>Coercive Detection Success</td>
<td>FAIL</td>
<td>FAIL</td>
</tr>
</tbody>
</table>

Figure 7: Image 5 Summary

4. Attempts to Evade OCR Detection by Manipulating Images

The following section details the attempts to evade the detection produced by OCR text extraction and the provided YARA rule. While the evasion would be easily accomplished by completely removing the suggestive instructions to enable the embedded logic, the spirit of the coercive image is designed to stay the same while subtly editing the image to maintain the apparent legitimacy and aspect of social engineering. Additionally, the results are described after the first attempt and not continuously tuned until the evasion is successful. Much like malware authors who test their attacks to gauge success against antivirus solutions before the campaign begins, multiple iterations are likely to be more successful.

Josiah Smith, josiahraysmith@gmail.com
4.1. Image 1 Post-manipulation

The first attempt at manipulating an image to evade the previously successful OCR derivative string detection is performed on the following image. For this example, the technique used was to change the strings’ font color, providing instruction for enabling the document’s macros. The color was changed to a similar color as the image background while maintaining the text’s readability. The technique was successful against the first OCR engine.

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="w.png" alt="Image" /></td>
<td>W Document created in earlier version of MS Office Word To view this content, please click from the yellow bar and then click “Enable Editing”</td>
<td>Document created in earlier version of MS Office Word To view this content, please click “Enable Editing” from the yellow bar and then click “Enable Content”</td>
</tr>
<tr>
<td>YARA Signature Hits</td>
<td>None</td>
<td>$ yara ../../OCR.rul Image1-manipulated.png.txt -s OCR_Rule Image1-manipulated.png.txt 0x93:$enable1: Enable Content 0x5f:$enable2: Enable Editing</td>
</tr>
<tr>
<td>Coercive Detection Success</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 7: Altered Image 1 Summary

4.2. Image 2 Post-manipulation

For the second image, a yellow box was drawn around the instructive strings to accentuate them to the unbeknownst user. Additionally, a lighter vertical line was drawn between the words to inject another character into the output. The effects disrupted the Tesseract engine output accuracy and eliminated half of the detected strings from iDRS.

Josiah Smith, josiahraysmith@gmail.com
The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'mOffice 365</td>
<td>Office365</td>
</tr>
<tr>
<td></td>
<td>You are attempting to open a file that was created in an earlier version of Microsoft Office.</td>
<td>You are attempting to open a file that was created in an earlier version of Microsoft Office.</td>
</tr>
<tr>
<td></td>
<td>If the file opens in Protected View, click <strong>Enable Editing</strong> and then click <strong>Enable Content</strong></td>
<td>If the file opens in Protected View, click Enable Editing, and then click Enable/Content</td>
</tr>
</tbody>
</table>

**YARA Signature Hits**

None

$ yara ../../OCR.rul image2-manipulated.jpg.txt -s
OCR_Rule image2-manipulated.jpg.txt
0x9a:$enable2: Enable Editing

**Coercive Detection Success**

No

Yes. 1 of two strings

Figure 8: Altered Image 2 Summary

4.3. Image 3 Post-manipulation

The third image that was manipulated to avoid detection is the Amazon Gift Card-themed image. For this example, the image was opened in the standard Microsoft Paint Application, and the text was replaced with a different text box containing a zero ASCII character instead of the “o” in the word “content.” Additionally, a thin eraser was used against the text to lower the letters’ “crispness.”
The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure. While the extracted strings were close, this manipulation effort was able to bypass detection based on both engines’ output.

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>amazon</td>
<td>$500</td>
</tr>
<tr>
<td></td>
<td>fl Office</td>
<td>amazon</td>
</tr>
<tr>
<td></td>
<td>This document created with Microsoft Office newest version.</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Please, press “Enable Content” to resolve.</td>
<td>•</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This document created with Microsoft Office newest version.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please, press “Enable Content” to resolve.</td>
</tr>
<tr>
<td>YARA Signature Hits</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Coercive Detection Success</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

Figure 9: Altered Image 3 Summary

4.4. Image 4 Post-manipulation

The Apple Gift Store image was manipulated with a blur filter over the “Enable Content” string. The filter swapped the text and background color to implement a black box around the white text. This specific technique disrupted the accuracy of the IDRS engine, but Tesseract still provided effective translation for the subsequent YARA inspection.

The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.
Table 1: Comparison of Optical Character Recognition Engines for Image 4

<table>
<thead>
<tr>
<th>Image</th>
<th>Tesseract Output</th>
<th>iDRS Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppleStone Gift Voucher baggy New Year mmfimew</td>
<td>9 Apple Store Gift Voucher You make us proud every year.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Office This document created with Microsoft. Office newest version; &quot;Enable Content&quot; to resolve.</td>
<td>This document created with Microsoft Office newest version. Please, press &quot;Enable Content&quot; to resolve.</td>
</tr>
<tr>
<td></td>
<td>‘rm us a m’ AdditlmlNtation Invoice Nmb: invoice Dale Tums Ship Ode</td>
<td>For a total of $25.50 Additlml Information Invoice NmDPf Invoice Date</td>
</tr>
<tr>
<td></td>
<td>9299813003 the m Cad m, m TNsoMefissnfiectbApple’sSdaaflReltldsm.</td>
<td>Shup Date 9299613083 Dec, 2020 Credit Card</td>
</tr>
<tr>
<td></td>
<td>wyme.we.cwwsnmuhm.</td>
<td>1 his order is subject to Apple’s Sales and Refunds Policies hl1p //store apple.com/CatalogflJ</td>
</tr>
<tr>
<td>YARA Signature Hits</td>
<td>$ yara ../../OCR.rul image4-manipulated.png.ocr -s OCR_Rule image4-manipulated.png.ocr 0xb1:Enable Content</td>
<td>None</td>
</tr>
</tbody>
</table>

| Coercive Detection Success | Yes | No |

Figure 10: Altered Image 4 Summary

4.5. Image 5 Post-manipulation

The Spanish language-themed Microsoft word image’s initial results were found to have exceptional extraction from both OCR engines. However, the multi-language support was not included in the YARA rule used for detection. For this post-manipulation analysis, the individual strings were added into the rule to support the Spanish strings. The manipulation of the image was done by blurring all of the text. While a bit more aggressive, the text is still readable, and this technique has been seen “in the wild”, proving the technique is successful in both aspects.

Neither OCR engines’ derived text was even close to accurate and ultimately evaded detection from both engines. The summary of the results below describes efficient OCR extraction and YARA detection from both OCR engines on the graphical lure.

Josiah Smith, josiahraysmith@gmail.com
5. Evasion Detection with DHASH

The previous image manipulations have proven the capacity of evading OCR-based detection against a standard set of OCR engines and a YARA rule. It’s responsible to acknowledge some mitigative factors to still contribute detection on the altered images. One such technique that can be used is perception hashing. This difference hash can be useful for detecting duplicate images and the slightly modified renditions used throughout the research. The following algorithm is known as DHASH, based on Neal Krawetz’s study (2013). The algorithm works by:

- Converting image to grayscale
- Reducing to a 9x9 thumbnail
- Producing a 64-bit row hash and 64-bit column hash
- Combining the two values

The installation and use for DHASH is relatively trivial. Analysts can run:

Josiah Smith, josiahraysmith@gmail.com
The following results depict the similarities between the images derived from the original maldoc and the edited graphics. Note the similarity between the perception hash despite the differing cryptographic MD5 hash. The threshold for defining similar based images based on perception hashing is subjective based on the implementation. For this use-case, the difference of 10% or less will be defined as a similar or manipulated image. With respect to the utilization of DHASH-based threat hunting or security monitoring, the images from malicious documents will need to be acquired, analyzed, and curated into a derived detection rule to be effective.

5.1. Image 1 Perception Hashing

```bash
$ md5sum image1*
931ac00fcbf461fae4fd3ee9f1f8ff02  image1-manipulated.png
d5fb7eb635c3c6fe0b2cfc4118627f08  image1.png

$ python -m dhash image1-manipulated.png
9069484c69b0c081107f7f80cf7f0003

$ python -m dhash image1.png
9069484c69b4c081107f7f80cf7f0003

$ python -m dhash image1.png image1-manipulated.png
1 bit differs out of 128 (0.8%)
```

5.2. Image 2 Perception Hashing

```bash
$ md5sum image2*
2602785981f896ab4371f40690370ac7  image2.jpg
4b8326ba41996cb41d36b6c0fce699c0  image2-manipulated.jpg

$ python -m dhash image2.jpg
202b2b230049512d80607fff000d40ff

$ python -m dhash image2-manipulated.jpg
202b2b230049532d80607fff000d40ff

$ python -m dhash image2.jpg  image2-manipulated.jpg
1 bit differs out of 128 (0.8%)
```

5.3. Image 3 Perception Hashing

```bash
$ md5sum image3*
2602785981f896ab4371f40690370ac7  image3.jpg
4b8326ba41996cb41d36b6c0fce699c0  image3-manipulated.jpg

$ python -m dhash image3.jpg
202b2b230049512d80607fff000d40ff

$ python -m dhash image3-manipulated.jpg
202b2b230049532d80607fff000d40ff

$ python -m dhash image3.jpg  image3-manipulated.jpg
1 bit differs out of 128 (0.8%)
```

Josiah Smith, josiahraysmith@gmail.com
6. Alternative Approach for Detecting Images (Halogen)

Throughout the research, a handful of alternative tools proved interesting in similarity for defensive purposes. An alternative approach for detecting files that have been embedded within malicious documents is the command-line tool Halogen. The tool automates the creation of YARA rules against image files embedded within documents (Eaton & Roersma, 2020).

Josiah Smith, josiahraysmith@gmail.com
This technique can provide additional coverage for maldoc lures that are not detected with OCR. Similar to Amini and Remen’s (2019) method to track and detect malicious documents while anchoring the XMP IDs, the previous examination of a document containing the graphical asset must be sourced to develop the detection logic. The tool is user friendly; after installation and a look at the help menu, the Yara rule is generated with the five maldoc samples used for the initial acquisition of the graphical lures.

```plaintext
python3 halogen.py -h

Halogen: Automatically create yara rules based on images embedded in office documents.

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  to parse
  -d DIR, --directory DIR
directory to scan for image files.
  -n NAME, --rule-name NAME
  specify a custom name for the rule file
  --png-idat
  For PNG matches, instead of starting with the PNG file
  header, start with the IDAT chunk.
  --jpg-sos
  For JPG matches, skip over the header and look for
  the Start of Scan marker, and begin the match there.
```

Running the tool against the original files produced the following YARA rule.

```plaintext
$ python3 halogen.py -d ~/5901/halogen/
rule halo_generated: maldoc images {
  meta:
    tlp = "amber"
    author = "Halogen Generated Rule"
    date = "2021-01-24"
    md5 = "[12453bda42e40501077b1d202a338648',
           '138e3164dd69df539809a0bc5cd37d36', '678aff9ae66a0fb94043261eeecaeec5',
           '5e870ca9e50adec114a98eab76f5b49c', 'bd4a94e6a71d7037664135b3498eb']"
    family = "malware family"
    scope = "["detection', 'collection']"
    intel = "['']"
  strings:
    $jpg_img_value_0 =
    (ffd8ffe000104a46494600010101060006000000fe1004b45786966000049492a000800000100098820200270000001a00000004100000047786e6764386761357533395a746a733434623aa346a74324337b7837)"
}
```

Josiah Smith, josiahraysmith@gmail.com
A Multi-Leveled Approach for Detection of Coercive Malicious Documents
Employing Optical Character Recognition

Josiah Smith, josiahraysmith@gmail.com

7. Conclusion

The rise of document-based malware has driven the necessity of effective measures to prevent, detect, and respond to this attack vector. When considering the colloquial proverb

condition: any of them

When using the Halogen generated rule, positive detection is confirmed against the initial file set. Additionally, running the derived YARA rule against a VirusTotal Retrohunt operation matched over 1300 samples with the same embedded graphical asset. A truncated list of 50 MD5 hashes is included in Appendix B. Using this approach for threat detection within an organization’s defensive posture does require the initial acquisition and curation for threats to maintain the following rule.

$ yara ../halo_generated:maldoc_images .
halo_generated ./12453bda42e40501077b1d202a338648
halo_generated ./138e316dd69df539809a0bc5cd37d36
halo_generated ./b1d4a944e6a71d17037664135b3498eb
halo_generated ./678aff9ae6a0fb94043261eeeecaec5
halo_generated ./5e870ca9e50a9c114a98eab76f5b49c
“Prevention is ideal, but detection is a must,” innovative approaches are being developed to detect these threats. Regarding maldocs, they have often been accompanied by coercive lures to social engineer the victim into enabling the dynamic content.

The approach of using OCR-derived test in conjunction with a tailored YARA rule to alert on this type of activity has been explored, with some effort to evade the positives by manipulating the coercive lures. Through the research, various success was identified in achieving this objective, but some additional detection techniques were explored to supplement the OCR-based detection.
References


VirusTotal. (2021). b93c1b5898ee3d02d1f7996c90256099. Retrieved from https://www.virustotal.com/gui/file/3c31bb37840e2413d56aec6497cf8f17a03df919713142a0937cbe2ce86fda/detection

Josiah Smith, josiahraysmith@gmail.com
Appendix A

Link to analyzed images and maldoc samples for download

https://github.com/JosiahRaySmith/STI_OCR_Maldoc_Samples

Appendix B Halogen- detected samples (MD5)

0327a4558ea7b4401f77e2e5148c0ff 750c4a8653659ef775a4160dcfc2d0cfc
12453bda4e2e0501077b1d202a338648 77e0fb05dafa8f607fb2e2b0d355ce5dd
138e3164dd69df538909a0bc5cd3736 79faaf40417d9217adf55909de9d2e9e9
195b3a109bd3c9276026507138f6e954 842ea5f93523fab661d1aa54d49ff0d8
1d7798702eb8689a080cb38a2c948ec1 84ca0a804bb4329fb0aece2592963fb
1df0b5bc020b7debc0d1a3634d2eece0f 8e15ea8a6bc8f38cc0f2643bdc84ae6c
20a938f804dd6ee995ee9cd923d75999 9c3489eabf0e7723726ae3d0051717b9
295a704c7ee659745ddc526dbfc625b ac5a832d975d248db09b6f61198a0e76
2c15ede996f60f1f28b345ec923eca44 b1d4a944e6a71d1703764135b3498eb
2f716680c59dd3a0343adf0667d1712 bab8427df24379ef969fefe14926cb9f
39b3250a21f54fdd83889f4e4a0d4932 bc8ca1e5d5df623f11a34278fe80d8a
3a9e19492e7c6e7052820563cd1a9f50 be400144480ff3d88322171914fcede1d
3ab2daf32a000abd8a5dc328235f15f9 c6d2e97d9f1b5d070d98e1b5249cd3ee6
3fe7bf47c10b57fd63d83f87b2e65ee c86077cbded356552749da8097c49f9b
53dbb5386352cc5f3c370cd12c7d4cc3 d1e2fc34470ec69519de6c08a9e13aa
57a2bfbce3d10610e9c0b2a5784f6df0 b3d951eddb8bd4d93814247ef3b05d6
5cad21a4ed485d9b56afa3857fe666f d4c84c161ba08890d3275429082504b
5e870ca9e50adecc114a98eb76f5b49c d5f2b5a1c76978cf586fa7287f03f6
5ed70d26731be613389933b056ad082 d60831c16d273efce98c2f99c45f858
632b845141ca60883026b2bf5ca082e e50c4fd673fc089e0212f3089d3adb
678aff9a6e0f9b4043261eeecaeac5 e510869a65f36548293ef2c01eaaad9
6b5a3f7f2ddaca3b89050b9425fd4b e5bd04c2680ed41ee96285d6f11366
726dbc2bfff09b63d781bb988a8f2e7c f3157b345a61c18ec4489005f70f44bc
748c844ad700769b4382496d3a3d2a5c feee9a27c8823963705749ec536cd5a4
74f241d54d4213229e29a105bd3f5582