Detection of Malicious Documents Utilizing XMP Identifiers

GIAC (GMON) Gold Certification

Author: Josiah Smith, josiahraysmith@gmail.com
Advisor: Bryan Simon

Accepted: 2020-August-02

Abstract

Modern digital documents are often composed of multiple other documents and images. Malware authors often produce malicious documents while reutilizing graphical assets or other components that can be uniquely identified with the Adobe E**x**tensible **M**etadata **P**latform (XMP). XMP IDs define a standard for mapping asset relationships and can be utilized to track, pivot, and cluster malicious campaigns, identify new TTPs, and possibly provide attribution against adversaries.

1. Introduction

Phishing and malspam are delivery techniques heavily used to distribute malware, and Office documents still tend to be the malware filetype of choice (Verizon, p18, 2020). However, the execution of active content is necessary for the majority of techniques that are used to invoke the malicious intent of the author. Considering the necessity of the user to perform actions such as “enable editing” or “enable content”, the incorporation of a graphical element that compels the user to perform those actions is often included. These malware lures take on many different forms and appear in different file types and variations of written language. However, they often share a unique metadata attribute that can be used to track their existence and identify other potentially related instantiations.

These unique metadata attributes belong to the Extensible Metadata Framework and are called XMP Identifiers. This research aims to prove that malicious samples can be detected using these anchors. To adequately document a known directory of suspicious indicators, they will need to be derived from malicious files. The testing methodology for this process will generally follow the defined procedure:

1. Develop YARA rule to source files with XMP IDs and detection level > 10
2. Deploy YARA rule to VirusTotal Live Hunt operation.
3. Acquire known bad samples and download them to the lab environment.
4. Examine/Document XMP IDs from known harmful files.
5. Develop a new rule with suspicious XMP IDs.
6. Deploy back into live hunt rule for conditioning.
7. Deploy into continuous security operations.

XMP IDs can provide the security researcher or operations team the ability to track, pivot, and cluster malicious campaigns, identify new TTPs, and possibly offer attribution against adversaries.

2. What is XMP?

The Extensible Metadata Platform (XMP) provides a standard format for the creation, processing, and exchange of metadata within an assortment of applications. XMP was developed by Adobe to solve the problem of incorporating a breadth of different metadata into various file formats while using a standard approach (Adobe, 2001). XMP allows tracking of parent-child relationships and different revisions and is most often stored in an XML format (Adobe, 2014). As described by Adobe (2001), there are three distinct properties within a packet, and together they define a managed asset.

Identifier	Property	Description
DcoumentID	xmpMM:DocumentID	Uniquely identifies an asset
InstanceID	xmpMM:InstanceID	Uniquely identifies a specific version of an asset
OriginalDocumentID	xmpMM:OrignalDocumentID	This value links an asset to its original source.

Figure 1: XMP Property Descriptions

While XMP IDs are not solely embedded within graphical assets and indeed can be used to identify other components of multiple file types, the re-utilization of the malware lure proves to be a useful anchor for malware discovery as confirmed by Amini and Remen (2019).

The recognition for the concept of utilizing XMP IDs to track malicious documents and their graphical assets is endowed to Michael Remen. When presented with the question, “What gave you the idea to start tracking XMP IDs?” He replied, “I noticed there was a lack of AV detection on this coercive type of malicious documents. While the images were benign, they were unique and being reused throughout multiple samples” (M. Remen, personal communication, 2020). The XMP IDs within the images have proved to be a useful anchor for detection.

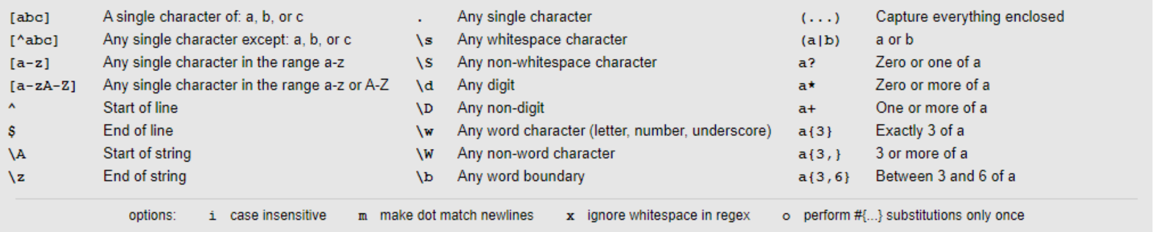
3. Rule Generation

There are two different formats of XMP IDs that have been observed throughout this research. The Hash format and GUID format. The XMP definition (Adobe, 2008), states, “An ID should be guaranteed to be globally unique... Typically 128 or 144-bit numbers are used, encoded as hexadecimal strings.” The typical length of 128 or 144-bit numbers infer the occurrence of 16 or 18-byte sequences within the ID and dictate the length that will be searched for within the regular expression. However, the predominant format is easily the 128-bit hash format based on the MD5 algorithm. The prevalence was dominant within this analysis of 9725 occurrences.

```
$ cat xmpidlist | grep xmp_ | cut -d: -f2 | distribution
    Key|Ct    (Pct)      Histogram
$xmp_md5|9725 (100.00%) -----------------------------------------------
```

3.1. Regular Expressions

A regular expression, or regex, is a type of text pattern that can be utilized within a variety of different programming languages and applications. In order to develop a regex to match on an XMP-ID, it is imperative to understand some of the basics. While reviewing the regex quick reference provided by Rubular (2020), it appears this regex can be quite simple with character classes.



[abc]	A single character of: a, b, or c	.	Any single character	(...)	Capture everything enclosed
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character	(a|b)	a or b
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character	a?	Zero or one of a
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit	a*	Zero or more of a
^	Start of line	\D	Any non-digit	a+	One or more of a
$	End of line	\w	Any word character (letter, number, underscore)	a{3}	Exactly 3 of a
\A	Start of string	\W	Any non-word character	a{3,}	3 or more of a
\z	End of string	\b	Any word boundary	a{3,6}	Between 3 and 6 of a

options: i case insensitive m make dot match newlines x ignore whitespace in regex o perform #{...} substitutions only once

Figure 2: Regular Expression

Concerning the XMP-ID in MD5 format, if a regex was developed to match on the instance ID *xmp.iid:B3D4F1219157E911B37B9950729CB11D,* the following regex can be used to identify it as well as other identifiers in this format. This regex was easily generated using the notation within the square brackets defined as a character class (Goyvaerts & Levithan, 2012, p. 34). A character class matches a single character out of a range of possible characters, which covers the different categories of IDs and allows the specification of the hash value.

```
xmp\.[dio]id[-: _][A-Fa-f0-9]{32}
```

3.2. YARA Rule Development

YARA is a tool that was developed to help information security researchers identify and classify malware samples. Often proclaimed as the pattern matching swiss army knife (Van Impe,2015), its practicality is a derivative of the author's imagination and can be used in a variety of practical applications, from data-loss detection to deep-dive forensics. Each rule is comprised of a set of strings and a condition of Boolean expressions that determine its detection logic.

In order to develop the first YARA rule, the two potential formats for XMP identifiers must be taken into consideration. The strings $xmp_md5 and $xmp_guid are defined with a regex to account for the different identifier categories and possible values. With the intent to scale down the data and provide a defined constant within this research, only Microsoft Office documents will be examined. The approach is applied with the definition of the $magic string. A magic number is a sequence of numbers or bytes that are embedded at or near the beginning of a file and can be used to indicate the file format. According to Microsoft (2020), the byte sequence {D0 CF 11 E0 A1 B1 1A E1 00 00 00} defines a Microsoft Office document and can be attributed to 21 different file extensions ("File signature database," n.d.).

The following derived rule is telling YARA that any file containing any of the strings defining the XMP IDs and that has the magic byte sequence for Office Documents, will trigger the signature.

```
rule Adobe_XMP_Identifier_Within_Offic_Document
{


   meta:
      Description = "This signature identifies Adobe Extensible Metadata Platform (XMP) identifiers embe
dded within Office Documents."
      Ref = "http://wwwimages.adobe.com/content/dam/acom/en/products/xmp/Pdfs/XMPAssetRelationshi
ps.pdf"


   strings:
   $xmp_md5  = /xmp\.[dio]id[-: _][a-f0-9]{32}/  nocase ascii wide
   $xmp_guid = /xmp\.[dio]id[-: _][a-f0-9]{36}/ nocase ascii wide
```

```
    $magic = { D0 CF 11 E0 A1 B1 1A E1 00 00 00 }

    condition:
        (any of ($xmp_*) and  $magic in (0..1024))
}
```

Figure 3: Office Document with XMP ID YARA Rule

3.3. VirusTotal Enterprise

To effectively source unique indicators from a massive corpus of malicious, benign, and potentially undetected files, this research is utilizing VirusTotal (VT) Enterprise. VT Enterprise is comprised of a multitude of valuable services including: VT Intelligence, VT Hunting, VT Graph, and VT API. Of specific utility, identification, parsing, and the acquisition will be accomplished with the Hunting and CLI services.

A useful component of the Hunting service is established with Livehunt and its respective notifications. Livehunt occurs when a users' YARA rules are uploaded and applied against all files sent to VirusTotal from around the world. The resulting near real-time notifications are provided and can be accessed programmatically or through the REST API.

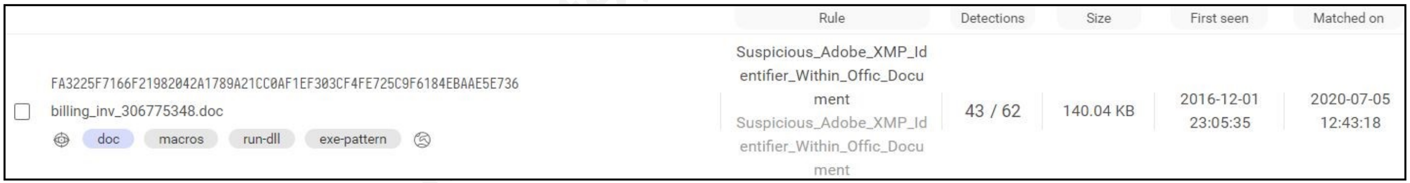
3.3.1. Deploying Live Hunt Rules

In order to employ a YARA rule to a Livehunt operation, a premium account is required. Within the VT Hunting Solution, simply navigate to the *Rulesets* sub-tab and generate a new ruleset that will be comprised of the necessary YARA rule(s). Since the intent of the research is to find graphical assets relating to malicious documents, the YARA rule will be modified to detect samples that have an arbitrary consensus between an assortment of AV vendors. This approach can be completed using a file search modifier within the YARA rule. Specifically, the modifier *positive:x* filters the results according to the number of antivirus vendors that were detected upon scanning it within the VirusTotal platform (VT, nd). For the initial sourcing of XMP-IDs found within graphical assets in malicious documents, the arbitrary number of positive detections was set modestly at a count of greater than ten. The resulting YARA condition was modified to resemble the following condition.

```
condition:
        (any of ($xmp_*) and  $magic in (0..1024) and positives > 10)
```

3.3.2. Explanation of results

After generating the Livehunt rule, it is necessary to wait until there are notifications relating to that ruleset. The notifications can be configured to alert via email or through the user interface. Figure 4 depicts one such notification. Of note, the FileName, SHA256 hash, Rule Name, and detection quantity is represented along with additional information. Another option to source results more quickly is with the RetroHunt capability. Retrohunt provides the ability to scan all the files sent to VirusTotal in the past 12 months with defined rulesets.



	Rule	Detections	Size	First seen	Matched on
□ FA3225F7166F21982042A1789A21CC0AF1EF303CF4FE725C9F6184EBAAE5E736 billing_inv_306775348.doc doc macros run-dll exe-pattern	Suspicious_Adobe_XMP_Identifier_Within_Offic_Document Suspicious_Adobe_XMP_Identifier_Within_Offic_Document	43 / 62	140.04 KB	2016-12-01 23:05:35	2020-07-05 12:43:18

Figure 4: Livehunt Result

If Livehunt notifications are not producing enough content for the given ruleset, the Retrohunt capability is a compelling feature that will scan a volume of more than 420M files (~680TB worth of data) within a few hours (Retrohunt, n.d.). However, this method does not allow for the inclusion of search modifiers (such as positive count), and the matches are capped at 10,000 per job. Fortunately, there is an automated way to resolve this uncertainty by using the VT CLI tool. The first requirement is to derive only the sha256 hash values that are appended to the end of the name provided for the retrohunt job. The most effective way to parse these hashes out is with an extended grep command:

```
      $ egrep -o [a-z0-9]{64} retrohunt_results > retrohunt_hashes
```

Now the resulting list of hashes can be scrutinized using the VT *file-report* and by defining the search modifier for positives in this script.

```
#!/bin/bash
read -p 'What are the minimum detections requested? ' detections
cat retrohunt_hashes | while read HASH ; do
   p=$(vt file-report ${HASH} | jq '.results.positives')
   if [ $p -ge $detections ] ; then
```

```
      echo $HASH has $p detections.
   fi
done
```

3.4. Generate List of Suspicious XMP IDs

After curating a listing of malicious Office documents that have an occurrence of an XMP ID, the next objective would be to generate a list of suspicious XMP IDs. This process of producing that list would entail downloading the corpus of files, parsing out the identifiers with YARA and other command-line tools, and then developing a successive YARA rule with the suspicious IDs either manually, or preferably programmatically.

The most effective way to download a large assortment of files from VirusTotal would be to utilize the VT CLI and script out the process. Within the Livehunt notifications for the particular ruleset, there is an option to export hashes. Using the list of hashes, the following Bash one-liner will loop through the list and download the individual files with the *file-download* argument (VirusTotal, 2020).

```
    $ cat hashlist | while read HASH; do vt file-download $HASH; done
```

One method used to parse out the XMP IDs from the corpus of malicious documents would be to run YARA recursively with the -s (--print-strings) option and then grep, cut, and sort the output. Using the distribution tool, it is apparent that the XMP IDs frequently occur throughout the collection of files (Ellis, Stearns, & Vivero, 2020).

```
$ yara ../rules/Adobe_XMP_Identifier_Within_Office_Document.rule . -rs
\| grep xmp_ | cut -d ":" -f3-4 | distribution

                                       Key|Ct  (Pct)   Histogram
 xmp.did:D8830B7209206811822AD4D0C71569CB|684 (0.14%) -----------------
------
 xmp.iid:04801174072068118C14D51DCC23285E|624 (0.13%) -----------------
-----
```

```
 xmp.iid:F77F117407206811808386AE8045B185|623 (0.13%) -----------------
----
 xmp.iid:058011740720681183D1FBCD990FA4A3|404 (0.08%) ---------------
 xmp.iid:088011740720681192B0E7E160E4461F|354 (0.07%) -------------
 xmp.iid:048011740720681192B0E7E160E4461F|354 (0.07%) -------------
 xmp.did:048011740720681192B0E7E160E4461F|354 (0.07%) -------------
 xmp.did:FB7F11740720681197A5F036D900E89E|352 (0.07%) -------------
 xmp.iid:A06C7ECD3B2068118083A844A13C9CD3|312 (0.06%) ------------
```

Now that there is a collection of XMP IDs that have been derived from the malicious Office documents, a new YARA rule can be generated. While the strings could be incremented manually, it would most likely be a timely task. Instead, the utilization of this novel_rule_generator will quickly facilitate the job (King, 2020).

```
#!/usr/bin/env python
# Script to build a Yara rule with an input file & one string per line
# Author: Rob King (@TheKingAdRob)

import re
import sys
YARA_TEMPLATE = """
    rule {rule_name}
    {{
        strings:
        {string_defs}
        condition:
            any of them
    }}
"""
if len(sys.argv) != 3:
    sys.exit("usage: %s RULE_NAME INPUT" % sys.argv[0])
strings = []
with open(sys.argv[2], "r") as fp:
    for number, line in enumerate(fp):
        strings.append("$s%d = /%s/" % (number,
re.escape(line.strip())))
```

```
print(YARA_TEMPLATE.format(rule_name=sys.argv[1],
string_defs="\n\t".join(strings)))
```

Figure 5: Rule Generation Script

To generate a new rule, begin by executing the script in Figure 5 while providing a name for the output rule and an input file containing the list of XMP IDs that were derived from the Malicious Office documents.

```
$ ./novel_rule_generator.py Suspicious_XMPID.rule xmpidlist

    rule Suspicious_XMPID.rule
    {
        strings:
        $s0 = /xmp\.did\:0180117407206811822A8F29126FB70E/
        $s1 = /xmp\.did\:02528A3E07B4E911A469955DC09DF99B/
        $s2 = /xmp\.did\:03801174072068118083F44BDB59DEDD/
        $s3 = /xmp\.did\:042493DF32C0E4118D9B8BA7F9641847/
        $s4 = /xmp\.did\:0880117407206811822A9F1610E837E9/
        $s5 = /xmp\.did\:08801174072068119231FAC1EE99744A/
        …
```

Now that the YARA rule containing strings of the identified suspicious XMP IDs has been generated, the process can reiterate with the addition into the VirusTotal Hunting rulesets. After creating a new ruleset, the Livehunt notifications can be used aggregate samples containing graphical assets that were found within the malicious Office documents.

Additionally, the current rendition of the suspicious XMP ID rule can be applied within an organization's continuous security monitoring environment, with consideration for the intention of this research is to identify files that are potentially malicious based on similarities found between them. Since the graphical assets that coerce users to enable active content are often reused, their reappearance in ensuing files that are undetected by AV solutions should be investigated. The continuous curation of the ruleset is also wise; add additional indicators as discovered and remove any indicators that are prevalent within benign instances.

4. Results and Analysis

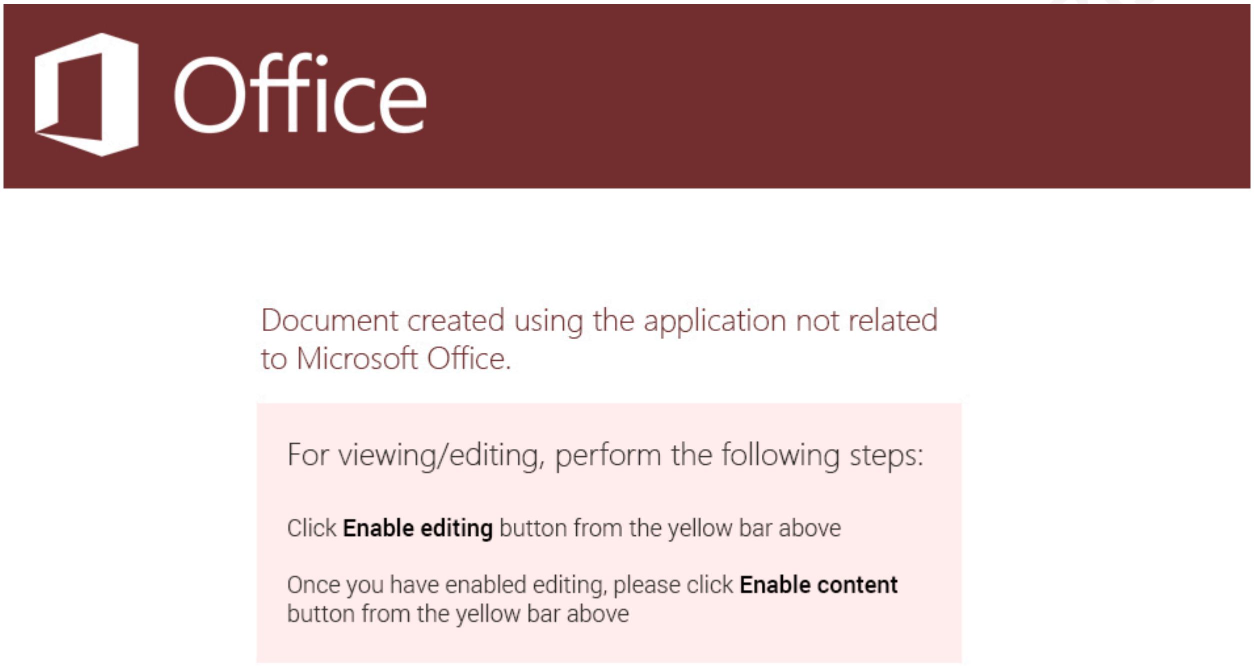
4.1. Sample A

The first file that will be discussed is a Microsoft Excel Spreadsheet with the SHA-1 hash of 6568d702e83bf5d1067b8bfadfcac6e696a57b49. This file was chosen for examination based on the presence of an XMP ID that reoccurred six times within the sample set *xmp.did:EB9F13B8B630E71191CDA99F6C9671D8*. This suspicious ID was also found within the following documents identified by the SHA-256 hashes.

```
193cb0caa096ce0b9c6ea2ac6875d1dbf12d4db1d630a100bd1c5fb74288bf3e
96d95668ee4138a14d8802b00fe76ae09b8fe7f27356288ab9e50a5346bcf988
98e4695eb06b12221f09956c4ee465ca5b50f20c0a5dc0550cad02d1d7131526
9e848ebc3af4449e73845e27238fb28eae7f9d4727da3eafb794ec6559d6f27b
c296f8c5522b03890d2c3681a19035781401f1094529089e405138eb25e9da72
e9f1dab486c5cb784f23d59f4b598bcea88dc4876f8246e63a8e4f033ebbff54
```

A more comprehensive listing of files containing this XMP ID can be acquired from InQuest Labs research portal at the following URL: https://labs.inquest.net/dfi/search/ioc/xmpid/xmp.did%3ABEB05E267EBBE6118943A1A884F74A77

The ensuing image is the graphical asset utilized within this Office document (Yes, now this research paper includes suspicious images). The image invokes the social engineering effort to coerce the pitiable user into enabling the active content and executing the embedded logic. This particular instance is utilizing the re-birthed techniques of obfuscated Excel 4.0 macros. The additional analysis derives high confidence that this file is part of an invoice-themed malspam campaign and the resulting payload found at *hxxps://merystol[.]xyz/6ng688x8* which was removed at the time of this research.



Document created using the application not related to Microsoft Office.

For viewing/editing, perform the following steps:

Click **Enable editing** button from the yellow bar above

Once you have enabled editing, please click **Enable content** button from the yellow bar above

Figure 2: Coercive Lure A

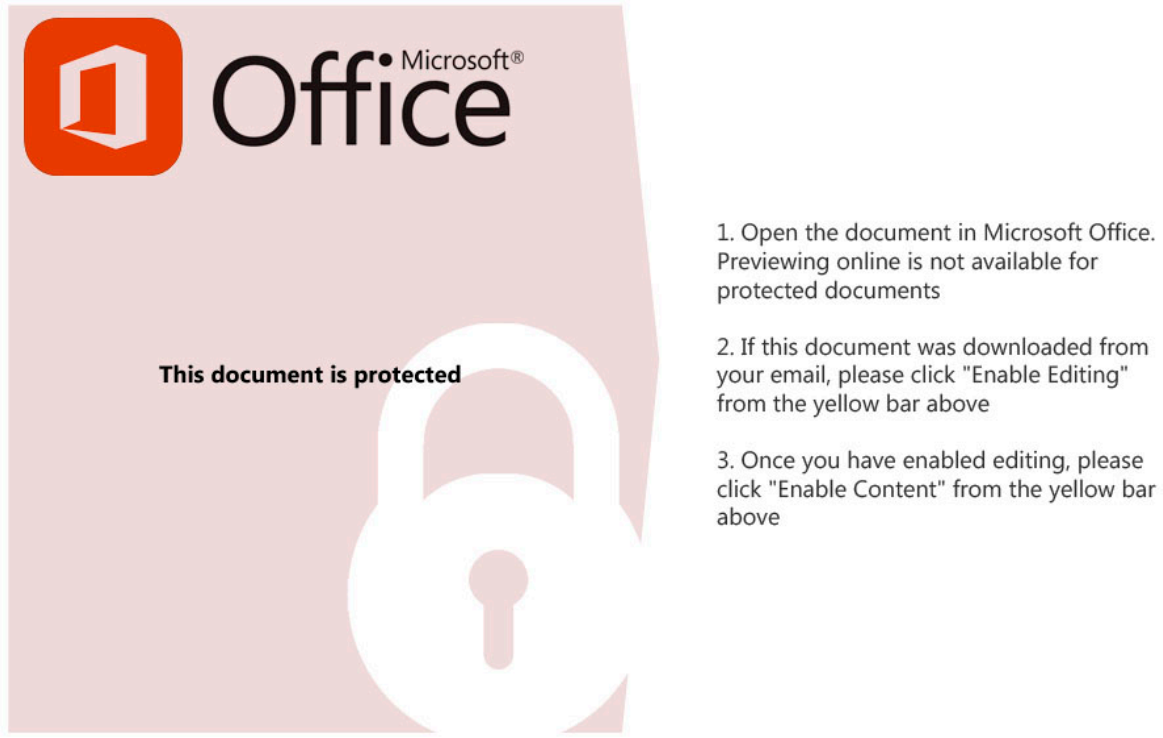
While the pivot to additional files was shown using the XMP ID above, 13 identifiers were found to be associated with the image.

```
$ yara Adobe_XMP_Identifier carving-000.jpg -s | cut -d: -f 3-4 | \
sort | uniq
 xmp.did:2F3641AF08ABE611BF55F001BDA26C84
 xmp.did:32A1613310ABE611BF55F001BDA26C84
 xmp.did:346765EC20ABE611BF55F001BDA26C84
 xmp.did:397D4A9586BBE6118943A1A884F74A77
 xmp.did:6096798C1B28E711A303AD291C1DD996
 xmp.did:71CD87DED406E611B390B6F985243EBE
 xmp.did:BEB05E267EBBE6118943A1A884F74A77
 xmp.did:C4B05E267EBBE6118943A1A884F74A77
 xmp.did:EB9F13B8B630E71191CDA99F6C9671D8
 xmp.iid:EB9F13B8B630E71191CDA99F6C9671D8
 xmp.iid:EC9F13B8B630E71191CDA99F6C9671D8
 xmp.iid:ED9F13B8B630E71191CDA99F6C9671D8
```

4.2. Sample B

The second file reviewed was a malicious Microsoft Word document with a SHA1 hash of 936252aa4f656eb5def659a00b9a818233d6933b. This graphical asset has the following XMP IDs associated with its presence. Pivoting of the first identifier, xmp.did:3BA1613310ABE611BF55F001BDA26C84, dozens of malicious documents were observed.

```
xmp.did:3BA1613310ABE611BF55F001BDA26C84
xmp.iid:306765EC20ABE611BF55F001BDA26C84
xmp.iid:3BA1613310ABE611BF55F001BDA26C84
xmp.iid:3CA1613310ABE611BF55F001BDA26C84
```



1. Open the document in Microsoft Office. Previewing online is not available for protected documents

2. If this document was downloaded from your email, please click "Enable Editing" from the yellow bar above

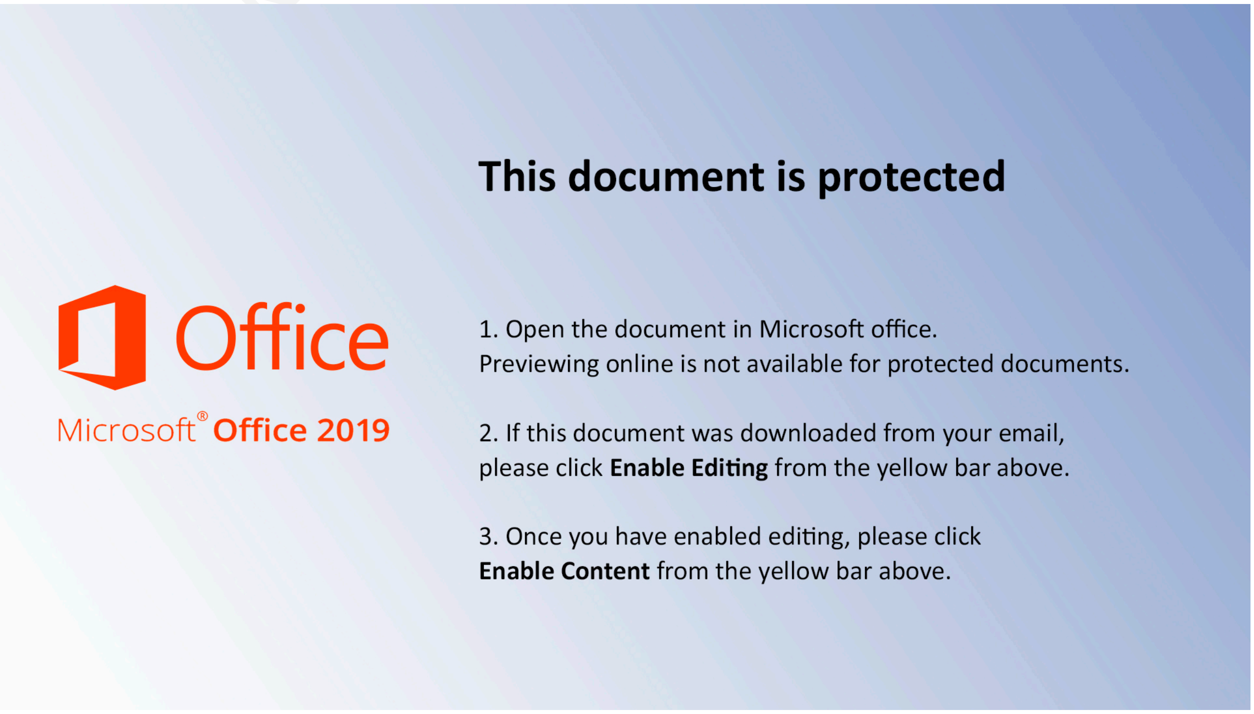
3. Once you have enabled editing, please click "Enable Content" from the yellow bar above

Figure 3: Coercive Lure B

4.3. Sample C

Examining the file 061269ffd3fba696a1da5363f1723525e468e6ef presented the following XMP-IDs associated with the graphical lure that coerces execution.

```
$ yara Adobe_XMP_Identifier carving-000.jpg -s | grep xmp_ |cut -d: \
-f 3-4
 xmp.did:03801174072068118083F44BDB59DEDD
 xmp.did:08801174072068119231FAC1EE99744A
 xmp.did:4FD4E7F18A2068118527A7954BFDAD06
 xmp.did:542CEC94152068119231FAC1EE99744A
 xmp.did:788A2A26D02511E7A0AEC879B62BAB1D
 xmp.did:CA0879F7C71211E89876B7EB71914955
 xmp.did:D27578FF33206811871FD2CE7718D7F3
 xmp.did:F77F117407206811A961EA0D9E399328
 xmp.did:FB301D6D11206811871FD2CE7718D7F3
 xmp.did:FC301D6D11206811871FD2CE7718D7F3
```



This document is protected

1. Open the document in Microsoft office.
Previewing online is not available for protected documents.

2. If this document was downloaded from your email,
please click **Enable Editing** from the yellow bar above.

3. Once you have enabled editing, please click
Enable Content from the yellow bar above.

Figure 4: Coercive Lure C

Author Name, email@addressjosiahraysmith@gmail.com

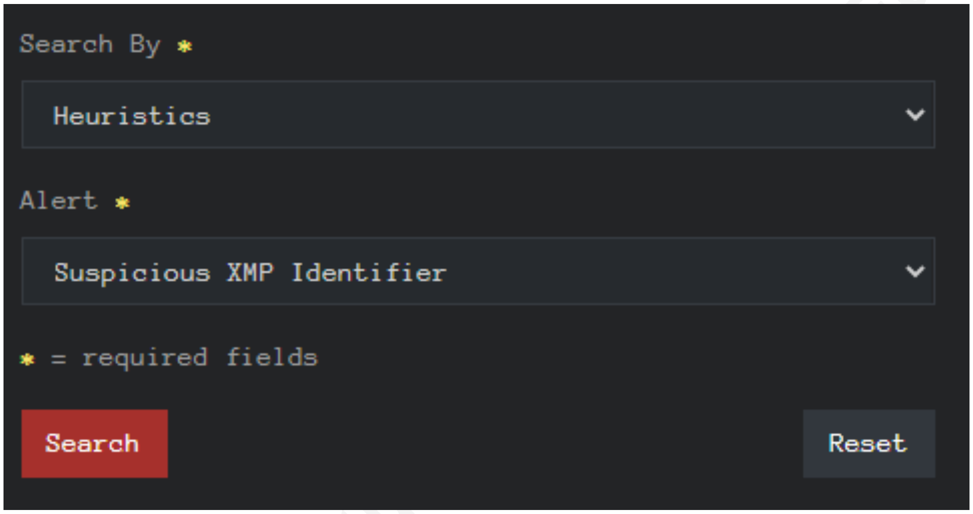
While pivoting on xmp.did:542CEC94152068119231FAC1EE99744A, there were 14 related malicious files available.

```
d5456147ea4579c7f22a48ae4f62d6432ff605707d9490a450702f0b307485d4
bb424b8755195c9d109ba274be45be4baf847c36a2394b70e8025cb320c4c347
96d35357643338b355adee447f84543496659e1c7c3be3049f232bcb18c71b9b
9d7dd89932a65caf83b3300043452b7f14b56ae7175cb526f1368ddf2b39d50e
a47db516311d4647223114c2b54bd5a33bf6e156c71486ba42bb2c2f9c16db52
6f0b09444670d89ec825e151c95e522c60bd764906995371c25aa0faf516775c
637a2678016822c45a019b3764ca8d4f9b5d4cc64bdfa52e2f91bcf3b4063d92
48c31d2f33fe11cb1e0135fe4d763c43a7c2745944a9fa0d76ca49dd885c74cc
0a55f500b5ba0c4c3ff9da59230302690be9542923aff4fb0af6ce8a2e593dfd
e68c8166954d6baca0f5e61a64fdaadd8f5744e909c6944262c17997d42c0fe6
e8ae56d47083be03044c12c2b4d719f3b8cef747ebbb908c4822dd9c6fc4f24b
a2d83a3acc4ce11608bcd8893eb7b71eba2292127e51b5edccde4c4ee47a354c
6badea2bc1cf4c96e7cb2b07255584cf592487cc54d27bda55028270a2f897c7
14869a8ab8ddc7e9fe39a4891236a1e87a5776190f43327f04db13cf02d2b37d
```

5. InQuest Labs

While most data observed was sourced from VirusTotal, it is prudent to mention the capabilities of the open-source research portal, InQuest Labs (InQuest, 2020). Capable of ingesting malware at scale, samples are fed through a lightweight and less featured version of Deep File Inspection to extract embedded logic, semantic content, metadata, and IOCs such as URLs, domains, IPs, e-mails, and file names.

Currently, Microsoft and Open Office documents, spreadsheets, and presentations are available for search and download. In the future, there will be an expansion of the public data set to include Adobe PDF documents, Java / Flash applets, and scriptlets, such as PowerShell. Security researchers can search extracted layers and IOCs by keyword, download samples, and pivot between samples by heuristic detections and IOCs. InQuest Labs has documented thousands of suspicious XMP identifiers, and the user interface provides the ability to search through a vast set of samples. Furthermore, the pivot functionality of IOCs provides for the correlation of samples with matching attributes like XMP IDs.


Search By *

Heuristics

Alert *

Suspicious XMP Identifier

* = required fields

Figure 5: Suspicious XMP Search

6. Conclusion

Cyber threats have become commonplace throughout the world and have heightened the importance of security monitoring, threat hunting, and incident response. Considering the most prevalent attacks require some action from the target to be effective, detection and timely intervention are more critical than ever. One useful technique to detect threats and related campaigns is the examination and tracking of the XMP identifiers used within the malicious documents. Diving into a process to acquire, curate, and redeploy suspicious indicators has been examined. By leveraging XMP IDs for signature development, effective threat detection can be enveloped within an organization's security monitoring posture.

References

Adobe. (2001). *A Manager's Introduction to Adobe eXtensible Metadata Platform, The Adobe XML Metadata Framework* [PDF file]. Retrieved from https://www.adobe.com/content/dam/acom/en/products/xmp/Pdfs/xmp_whitepaper.pdf

Adobe. (2014). *Introduction to Asset Relationships* [PDF file]. Retrieved from https://www.adobe.com/content/dam/acom/en/products/xmp/Pdfs/xmp_whitepaper.pdf

Adobe. (2008). *XMP Specification Part 2: Standard Schemas* [PDF file]. Retrieved from https://www.adobe.com/content/dam/acom/en/products/xmp/Pdfs/xmp_whitepaper.pdf

Adobe. (2010). *XMP Specification Part 3: Storage in FIles* [PDF file]. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.174.1092&rep=rep1&type=pdf

Amini, P., & Remen, M. (2019, September 30). Adobe XMP: Tales of an Overlooked Anchor. InQuest.net. https://inquest.net/blog/2019/09/30/Adobe-XMP-Tales-of-an-Overlooked-Anchor

Ellis, T., Stearns, T., & Vivero, P. (2020). Distribution (Version 1.2.2) [Software]. Available from https://github.com/wizzat/distribution.

File signature database:: D0cf11e0a1b11ae1 file signatures. (n.d.). File Signature Database:. https://www.filesignatures.net/index.php?page=search&search=D0CF11E0A1B11AE1&mode=SIG

Goyvaerts, J., & Levithan, S. (2012). *Regular expressions cookbook.* O'Reilly Media.

InQuest. (2020). InQuest Labs. Retrieved June 18, 2020, from https://labs.inquest.net.

King, R. (2020). Novel_Rule_Generator.py (Version 1.0) [Software]. Available from https://github.com/InQuest/yara-rules/blob/master/novel_rule_generator.py.

Microsoft. (2020, March 30). *The header*. Technical documentation, API, and code examples | Microsoft Docs. https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cfb/68530324-9d3d-4441-9ea9-66a2c8f79567

Remen, M. (2020, May 8). Personal interview.

Author Name, email@addressjosiahraysmith@gmail.com

Retrohunt. (n.d.). VirusTotal. Retrieved June 19, 2020, from https://support.virustotal.com/hc/en-us/articles/360001293377-Retrohunt

Rubular. (2020, June 18).Rubular: A Ruby Regular Expression Generator. Retrieved from https://rubular.com/

Van Impe, K. (2015, June 24). *Signature-based detection with YARA*. Security Intelligence. https://securityintelligence.com/signature-based-detection-with-yara/

Verizon. (2020). *2020 Data Breach Investigations Report* [PDF file]. Retrieved from https://enterprise.verizon.com/resources/reports/2020/2020-data-breach-investigations-report.pdf

VirusTotal. (2020). VirusTotal Command Line Interface (Version 1.0) [Software]. Available from https://github.com/VirusTotal/vt-cli.

Appendix

(SHA256 Hashes derived from XMP IDs)

Sample A: xmp.did:EB9F13B8B630E71191CDA99F6C9671D8
193cb0caa096ce0b9c6ea2ac6875d1dbf12d4db1d630a100bd1c5fb74288bf3e
97f28fae05e1872a55d5004b52b7e1f2ceef04742ea4e5926b39bd6866f295f6
af743bc10cecfde85c96ba0dfdfb50ed043743b3a426f40432b39d15c9cbd09f
fc1a3658d1e9c45ef564c3515e081635581a8a7fcf41fa4927f0056995c6ec68
fae3b23661a6db77a28dabe5f52367e3c0c927f4205bbcb64ab88a249eb53d98
fb731105a96db1882a3d16975edd56c80f9ddbcd2132805ddeec45ca06ab20bd
f72bbb3c15f527a736b5778cd9f4bb4b7be73b2329bd6ea86281db092f803b73
f1c70ec5dd94681bf1fb21f512bed1ca4afbf9f0a140877a7e78daa6559707de
f09f0636552fb22b7b9dfd9701cadcde955565b38d1f2d2be032ed9baee71f26
fc0e361cf10c03d5511c01ff89f94ccfc721335328d6073d1640a7dcd4ab5d2c
fb4260fa5c521a984c6611fddff1a96d789129b6ba40ed228b924b66d5305ec6
eb1ce9fec03db92784ab3ba45abf206a5485f9dd2a8d64758313450f21d93214
eb33734ee9f2175c2b77319702bd26118516743d65be427d88981b15342e3729
f7db246e98ae037cbf0d2ae041d359af086945c54c7c501354e093907647cffd
f67de045e8a9166f781cf9482e814e46f68bbc0872b2b526c46c4ae9c4f0dff2
f600cc0a6a9b10ead20d714f601735b867708c9377100a0e368c10051d34ea36
ff9b0ad63eee1f1809f2a367df9c0f762f6ed6cc03ae5faf3f526d1ea3d11bff
ff15dfdea413825023c755b3bb6bd224ec84e49af3072b25358225f10e951112
fe1c4320f5a47b87c2c3e6de18e2da30d8ae568eac904d4f6f6cb28553cc725e
fc77b2870ce98e2462ee8b1eba4e00954e2072055e7a8974c6365ff93b481f5c
fc2ce43c0bcfb91bd701e5ef9034a40a7f7133f0cbd935543cff5d2d5bdaa9f1
e5b3378f01190f87f291294db0e6a414c4c9dfa427252d286c8c17202044550f
fa6ebeee9a373a0a771ae440f0480d56f098f0183bd0e80efba491bba4351a18
e496f32a3b90c99d3858990a7d095a889e3b00c702d23116e838ceaf8cdaa243
ed497fab01bbd56f8d38c79b06db6b32e6c7d9f2772088cb85234e75a9d34122

Sample B: xmp.did:3BA1613310ABE611BF55F001BDA26C84
c4e28ef5ebdaec42756f0cff70e8c28fa19f086cf63b31db662cb90470088c86
86b08775eda705cb6c9767abab0c8765033e32b535cc575299891a9b15ff206f
fe8ece8443136e3c70f7b8dd9ec936204c0417ce9f1bca13e8c6efb816017b75
ed1e104b79b03e1d361e36b4228a336d4112e9f2581aeaa2c2c2e995277f574d
f0375d9f34a059618b5e185c21f81ba3cc75ef02f36671558e3c10132039ddca
e6636ac757011d0cf2b6e68bbd3e33044df0bd088faeb2b504ba9c9efac84f64
d60dbc150c1ad9f052e4ed8c73d2dbcedf95ca02706697bb87b142afdc9351b6
d52d521b8b9b37d3f22929381f73279b29d92ffff73e62f5c5b6b9bd3615d655
d1debd993e6feae20c56849ea48284181059887a1904f6a501772071d8aa9527
c7d96679b5880519a8bbbcc79d8a294b75b5396221374d9866bf146b484a60d3
c31df85bf2293735b05d2dbe22ba1a2173fc7f7f524b790dddc8de47dad5f913
c283993f922697711fdafd42b44bc0538962687f944395c306198c63c1a37935
be041162fcb2f3dd21ea7c29005b709e5062ce376b3e64e019cac216c5471823

Author Name, email@addressjosiahraysmith@gmail.com

a4e8a31721224eb846df574a205995236860b99fcadca977c5eecd02bf68cac3
a10726236520a31f9dcb149f3d49a309406a860d2724492c000d4dd23ddfb3ac
9f358b978da3596d5fbf91a9d3d75ff2a589afba0ae5db1a4f1bf6db6d3002fb
9912554fde9fa47a09eb9513248cb1b7ea17afbc68c8dc563a370aeb6aa6f503
982c3a421c987120b8cc20ec1f715b9a8c933b9a648d40898390b8e46ff4525b
84dc860ced44a0997f2a5bd113e50fdac65eb2ddb76cca6ad35e967604f499e9
675f8677284fb042897019c27a9fad90a77223b5cc21c349e3f30654ac58f549
5e6ce529e078a7983bdcff4fb3f76273a8b0b1ce1b6e6e6857d8c3bad4bf9ff6
5c5299d81ef7d936f21a62c144a1873b9a2fe46b2d22fd8373c7b9887bb42623
543e02803efe428d9fd135db644d774dac1bc3250f07bdc7424ac82d192cc707
4d6ee4a0da756fbfa3d73e1fb2203dbc7ceb8727146e261dd6f4854a88f5ca9c
3f425d8dae077dd5cc747a2483bfc287294b4434f8c67f91863ebbf2e763223e

Sample C: xmp.did:542CEC94152068119231FAC1EE99744A
d5456147ea4579c7f22a48ae4f62d6432ff605707d9490a450702f0b307485d4
bb424b8755195c9d109ba274be45be4baf847c36a2394b70e8025cb320c4c347
96d35357643338b355adee447f84543496659e1c7c3be3049f232bcb18c71b9b
9d7dd89932a65caf83b3300043452b7f14b56ae7175cb526f1368ddf2b39d50e
a47db516311d4647223114c2b54bd5a33bf6e156c71486ba42bb2c2f9c16db52
6f0b09444670d89ec825e151c95e522c60bd764906995371c25aa0faf516775c
637a2678016822c45a019b3764ca8d4f9b5d4cc64bdfa52e2f91bcf3b4063d92
48c31d2f33fe11cb1e0135fe4d763c43a7c2745944a9fa0d76ca49dd885c74cc
0a55f500b5ba0c4c3ff9da59230302690be9542923aff4fb0af6ce8a2e593dfd
e68c8166954d6baca0f5e61a64fdaadd8f5744e909c6944262c17997d42c0fe6
e8ae56d47083be03044c12c2b4d719f3b8cef747ebbb908c4822dd9c6fc4f24b
a2d83a3acc4ce11608bcd8893eb7b71eba2292127e51b5edccde4c4ee47a354c
6badea2bc1cf4c96e7cb2b07255584cf592487cc54d27bda55028270a2f897c7
14869a8ab8ddc7e9fe39a4891236a1e87a5776190f43327f04db13cf02d2b37d