Blog

Cybersecurity DNA

MirageFox: APT15 Resurfaces With New Tools Based On Old Ones

Share:

PDF created with the PDFmyURL **web to PDF API**!

PDFmyURL

Jay Rosenberg
14.06.18 | 3:26 pm

APT15 Background

Coincidentally, following the recent hack of a US Navy contractor and theft of highly sensitive data on submarine warfare, we have found evidence of very recent activity by a group referred to as APT15, known for committing cyber espionage which is believed to be affiliated with the Chinese government. The malware involved in this recent campaign, MirageFox, looks to be an upgraded version of a tool, a RAT believed to originate in 2012, known as Mirage.

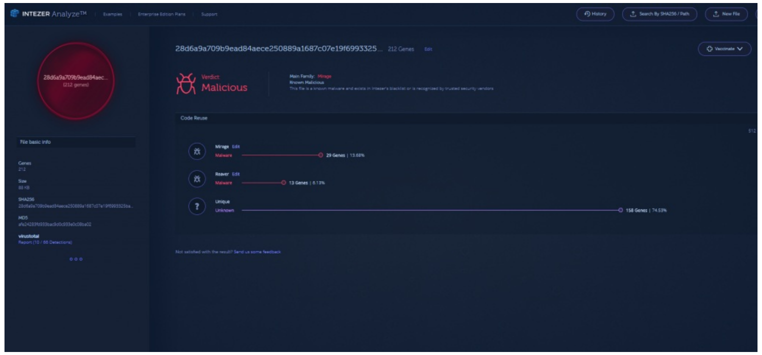
APT15 is known for committing cyberespionage against companies and organizations located in many different countries, targeting different sectors such as the oil industry, government contractors, military, and more. They are known for "living off the land," meaning they use already available tools and software installed on the computer to operate, and once inside a target network, they will tailor their malware specifically to the target. Other names for the group are Vixen Panda, Ke3chang, Royal APT, and Playful Dragon.

There are many articles and researches online about APT15 and their activities, the most recent one by NCC Group; although posted in March 2018, it refers to a campaign in 2017. In addition, although the 2017 campaign has been documented, during our research regarding MirageFox, we found a recently uploaded binary (6/8/2018) from the 2017 campaign, pretty much identical to a RAT mentioned in their RoyalAPT report, **barely detected with only 7/66 detections on VirusTotal.**

APT15 Code Reuse

We found the new version of the RAT on VirusTotal hunting, by a YARA signature we created based off code *only found in* Mirage and Reaver, both attributed to Chinese government affiliated groups. After seeing that these binaries were new uploads to VirusTotal, with very few detections, we analyzed them using Intezer Analyze™ to see if we could find any code reuse.

PDF created with the PDFmyURL **web to PDF API**!



28d6a9a709b9ead84aece250889a1687c07e19f6993325... 212 Genes Edit

Verdict: Malicious

Main Family: Mirage
Known Malicious
This file is a known malware and exists in Intezer's blacklist or is recognized by trusted security vendors

Code Reuse

Mirage Edit
Malware 29 Genes | 13.68%

Reaver Edit
Malware 13 Genes | 6.13%

Unique
Unknown 158 Genes | 74.53%

File basic info

Genes
212

Size
88 KB

SHA256
28d6a9a709b9ead84aece250889a1687c07e19f6993325ba...

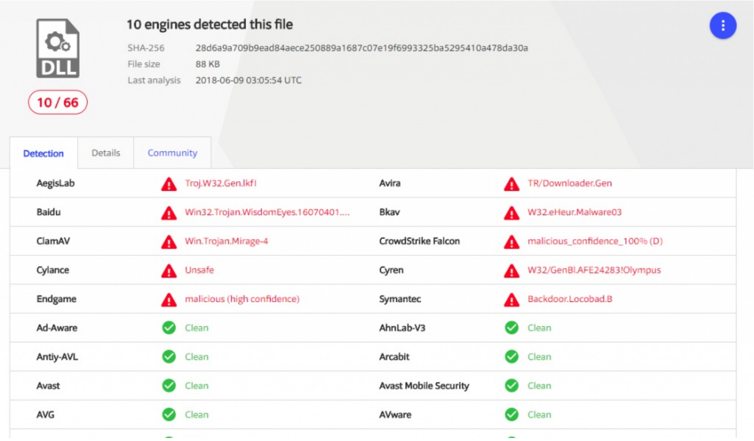
MD5
afe24283fd933bac9d0c933e0c08ba02

virustotal
Report (10 / 66 Detections)

Not satisfied with the result? Send us some feedback

(https://analyze.intezer.com/#/analyses/d00b6787-0078-4148-aec3-a66779a22ba5)

As can be seen in this code reuse analysis report (SHA256: 28d6a9a709b9ead84aece250889a1687c07e19f6993325ba5295410a478da30a), there is shared code with Mirage and Reaver. The compilation timestamp is from June 8, 2018 while the upload date to VirusTotal was June 9, 2018.



10 engines detected this file

SHA-256 28d6a9a709b9ead84aece250889a1687c07e19f6993325ba5295410a478da30a
File size 88 KB
Last analysis 2018-06-09 03:05:54 UTC

AegisLab	Troj.W32.Gen.lkfI	Avira	TR/Downloader.Gen
Baidu	Win32.Trojan.WisdomEyes.16070401....	Bkav	W32.eHeur.Malware03
ClamAV	Win.Trojan.Mirage-4	CrowdStrike Falcon	malicious_confidence_100% (D)
Cylance	Unsafe	Cyren	W32/GenBl.AFE24283!Olympus
Endgame	malicious (high confidence)	Symantec	Backdoor.Locobad.B
Ad-Aware	✓ Clean	AhnLab-V3	✓ Clean
Antiy-AVL	✓ Clean	Arcabit	✓ Clean
Avast	✓ Clean	Avast Mobile Security	✓ Clean
AVG	✓ Clean	AVware	✓ Clean

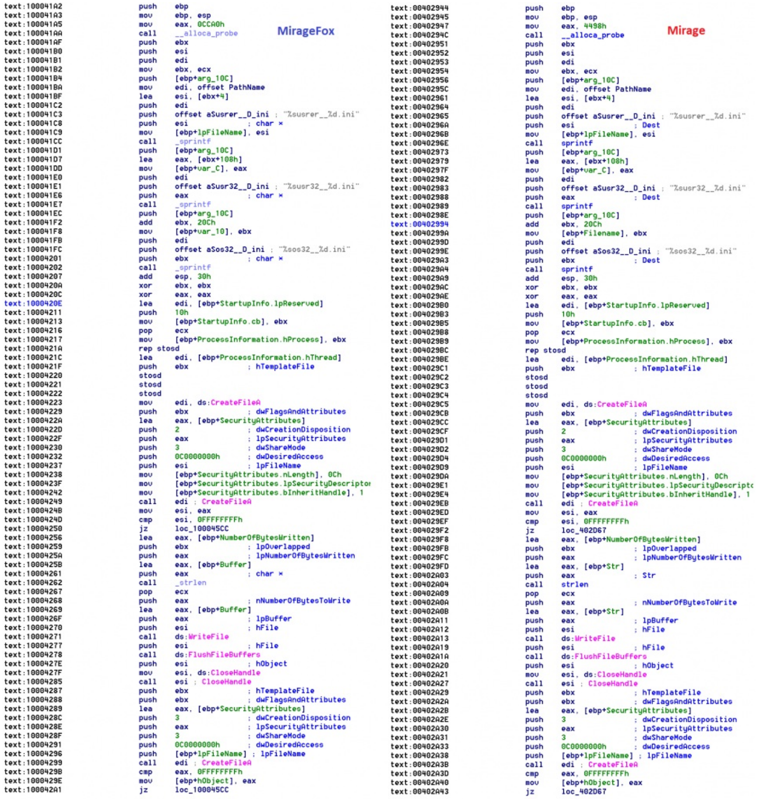
(VirusTotal)

On VirusTotal, we can see there are only 10/66 detections for this binary, 11/66 for another similar version of MirageFox (SHA256: 97813e76564aa829a359c2d12c9c6b824c532de0fc15f43765cf6b106a32b9a5), and 9/64 for the third MirageFox binary that was uploaded (SHA256: b7c1ae10f3037b7645541acb9f7421312fb1e164be964ee7acd6eb1299d6acb2).

PDF created with the PDFmyURL web to PDF API!

Here's a couple examples of code reuse similarities found in the Mirage family between one of the newer binaries and older ones.

<u>Remote Shell</u>:



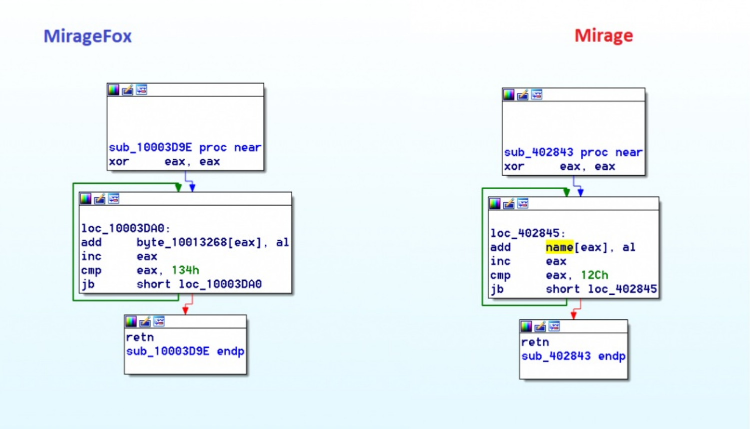
```
text:100041A2                 push    ebp
text:100041A3                 mov     ebp, esp
text:100041A5                 mov     eax, 0CCA0h
text:100041AA                 call    __alloca_probe
text:100041AF                 push    ebx
text:100041B0                 push    esi
text:100041B1                 push    edi
text:100041B2                 mov     ebx, ecx
text:100041B4                 push    [ebp+arg_10C]
text:100041BA                 mov     edi, offset PathName
text:100041BF                 lea     esi, [ebx+4]
text:100041C2                 push    edi
text:100041C3                 push    offset aSusrer__D_ini ; "%susrer__%d.ini"
text:100041C8                 push    esi             ; char *
text:100041C9                 mov     [ebp+lpFileName], esi
text:100041CC                 call    _sprintf
text:100041D1                 push    [ebp+arg_10C]
text:100041D7                 lea     eax, [ebx+108h]
text:100041DD                 mov     [ebp+var_C], eax
text:100041E0                 push    edi
text:100041E1                 push    offset aSusr32__D_ini ; "%susr32__%d.ini"
text:100041E6                 push    eax             ; char *
text:100041E7                 call    _sprintf
text:100041EC                 push    [ebp+arg_10C]
text:100041F2                 add     ebx, 20Ch
text:100041F8                 mov     [ebp+var_10], ebx
text:100041FB                 push    edi
text:100041FC                 push    offset aSos32__D_ini ; "%sos32__%d.ini"
text:10004201                 push    ebx             ; char *
text:10004202                 call    _sprintf
text:10004207                 add     esp, 30h
text:1000420A                 xor     ebx, ebx
text:1000420C                 xor     eax, eax
text:1000420E                 lea     edi, [ebp+StartupInfo.lpReserved]
text:10004211                 push    10h
text:10004213                 mov     [ebp+StartupInfo.cb], ebx
text:10004216                 pop     ecx
text:10004217                 mov     [ebp+ProcessInformation.hProcess], ebx
text:1000421A                 rep stosd
text:1000421C                 lea     edi, [ebp+ProcessInformation.hThread]
text:1000421F                 push    ebx             ; hTemplateFile
text:10004220                 stosd
text:10004221                 stosd
text:10004222                 stosd
text:10004223                 mov     edi, ds:CreateFileA
text:10004229                 push    ebx             ; dwFlagsAndAttributes
text:1000422A                 lea     eax, [ebp+SecurityAttributes]
text:1000422D                 push    2               ; dwCreationDisposition
text:1000422F                 push    eax             ; lpSecurityAttributes
text:10004230                 push    3               ; dwShareMode
text:10004232                 push    0C0000000h      ; dwDesiredAccess
text:10004237                 push    esi             ; lpFileName
text:10004238                 mov     [ebp+SecurityAttributes.nLength], 0Ch
text:1000423F                 mov     [ebp+SecurityAttributes.lpSecurityDescriptor
text:10004242                 mov     [ebp+SecurityAttributes.bInheritHandle], 1
text:10004249                 call    edi ; CreateFileA
text:1000424B                 mov     esi, eax
text:1000424D                 cmp     esi, 0FFFFFFFFh
text:10004250                 jz      loc_100045CC
text:10004256                 lea     eax, [ebp+NumberOfBytesWritten]
text:10004259                 push    ebx             ; lpOverlapped
text:1000425A                 push    eax             ; lpNumberOfBytesWritten
text:1000425B                 lea     eax, [ebp+Buffer]
text:10004261                 push    eax             ; char *
text:10004262                 call    _strlen
text:10004267                 pop     ecx
text:10004268                 push    eax             ; nNumberOfBytesToWrite
text:10004269                 lea     eax, [ebp+Buffer]
text:1000426F                 push    eax             ; lpBuffer
text:10004270                 push    esi             ; hFile
text:10004271                 call    ds:WriteFile
text:10004277                 push    esi             ; hFile
text:10004278                 call    ds:FlushFileBuffers
text:1000427E                 push    esi             ; hObject
text:1000427F                 mov     esi, ds:CloseHandle
text:10004285                 call    esi ; CloseHandle
text:10004287                 push    ebx             ; hTemplateFile
text:10004288                 push    ebx             ; dwFlagsAndAttributes
text:10004289                 lea     eax, [ebp+SecurityAttributes]
text:1000428C                 push    3               ; dwCreationDisposition
text:1000428E                 push    eax             ; lpSecurityAttributes
text:1000428F                 push    3               ; dwShareMode
text:10004291                 push    0C0000000h      ; dwDesiredAccess
text:10004296                 push    [ebp+lpFileName] ; lpFileName
text:10004299                 call    edi ; CreateFileA
text:1000429B                 cmp     eax, 0FFFFFFFFh
text:1000429E                 mov     [ebp+hObject], eax
text:100042A1                 jz      loc_100045CC
```

```
text:00402944                 push    ebp
text:00402945                 mov     ebp, esp
text:00402947                 mov     eax, 4498h
text:0040294C                 call    __alloca_probe
text:00402951                 push    ebx
text:00402952                 push    esi
text:00402953                 push    edi
text:00402954                 mov     ebx, ecx
text:00402956                 push    [ebp+arg_10C]
text:0040295C                 mov     edi, offset PathName
text:00402961                 lea     esi, [ebx+4]
text:00402964                 push    edi
text:00402965                 push    offset aSusrer__D_ini ; "%susrer__%d.ini"
text:0040296A                 push    esi             ; Dest
text:0040296B                 mov     [ebp+lpFileName], esi
text:0040296E                 call    sprintf
text:00402973                 push    [ebp+arg_10C]
text:00402979                 lea     eax, [ebx+108h]
text:0040297F                 mov     [ebp+var_C], eax
text:00402982                 push    edi
text:00402983                 push    offset aSusr32__D_ini ; "%susr32__%d.ini"
text:00402988                 push    eax             ; Dest
text:00402989                 call    sprintf
text:0040298E                 push    [ebp+arg_10C]
text:00402994                 add     ebx, 20Ch
text:0040299A                 mov     [ebp+Filename], ebx
text:0040299D                 push    edi
text:0040299E                 push    offset aSos32__D_ini ; "%sos32__%d.ini"
text:004029A3                 push    ebx             ; Dest
text:004029A4                 call    sprintf
text:004029A9                 add     esp, 30h
text:004029AC                 xor     ebx, ebx
text:004029AE                 xor     eax, eax
text:004029B0                 lea     edi, [ebp+StartupInfo.lpReserved]
text:004029B3                 push    10h
text:004029B5                 mov     [ebp+StartupInfo.cb], ebx
text:004029B8                 pop     ecx
text:004029B9                 mov     [ebp+ProcessInformation.hProcess], ebx
text:004029BC                 rep stosd
text:004029BE                 lea     edi, [ebp+ProcessInformation.hThread]
text:004029C1                 push    ebx             ; hTemplateFile
text:004029C2                 stosd
text:004029C3                 stosd
text:004029C4                 stosd
text:004029C5                 mov     edi, ds:CreateFileA
text:004029CB                 push    ebx             ; dwFlagsAndAttributes
text:004029CC                 lea     eax, [ebp+SecurityAttributes]
text:004029CF                 push    2               ; dwCreationDisposition
text:004029D1                 push    eax             ; lpSecurityAttributes
text:004029D2                 push    3               ; dwShareMode
text:004029D4                 push    0C0000000h      ; dwDesiredAccess
text:004029D9                 push    esi             ; lpFileName
text:004029DA                 mov     [ebp+SecurityAttributes.nLength], 0Ch
text:004029E1                 mov     [ebp+SecurityAttributes.lpSecurityDescripto
text:004029E4                 mov     [ebp+SecurityAttributes.bInheritHandle], 1
text:004029EB                 call    edi ; CreateFileA
text:004029ED                 mov     esi, eax
text:004029EF                 cmp     esi, 0FFFFFFFFh
text:004029F2                 jz      loc_402D67
text:004029F8                 lea     eax, [ebp+NumberOfBytesWritten]
text:004029FB                 push    ebx             ; lpOverlapped
text:004029FC                 push    eax             ; lpNumberOfBytesWritten
text:004029FD                 lea     eax, [ebp+Str]
text:00402A03                 push    eax             ; Str
text:00402A04                 call    strlen
text:00402A09                 pop     ecx
text:00402A0A                 push    eax             ; nNumberOfBytesToWrite
text:00402A0B                 lea     eax, [ebp+Str]
text:00402A11                 push    eax             ; lpBuffer
text:00402A12                 push    esi             ; hFile
text:00402A13                 call    ds:WriteFile
text:00402A19                 push    esi             ; hFile
text:00402A1A                 call    ds:FlushFileBuffers
text:00402A20                 push    esi             ; hObject
text:00402A21                 mov     esi, ds:CloseHandle
text:00402A27                 call    esi ; CloseHandle
text:00402A29                 push    ebx             ; hTemplateFile
text:00402A2A                 push    ebx             ; dwFlagsAndAttributes
text:00402A2B                 lea     eax, [ebp+SecurityAttributes]
text:00402A2E                 push    3               ; dwCreationDisposition
text:00402A30                 push    eax             ; lpSecurityAttributes
text:00402A31                 push    3               ; dwShareMode
text:00402A33                 push    0C0000000h      ; dwDesiredAccess
text:00402A38                 push    [ebp+lpFileName] ; lpFileName
text:00402A3B                 call    edi ; CreateFileA
text:00402A3D                 cmp     eax, 0FFFFFFFFh
text:00402A40                 mov     [ebp+hObject], eax
text:00402A43                 jz      loc_402D67
```

The function above is seen throughout many of the binaries in the Mirage family and is executed when a command is sent from the C&C. It is responsible for executing commands in cmd.exe (later down in the functions, not seen in the screenshot, it looks for cmd.exe and executes it using *CreateProcessA*).

<u>Configuration Decryption:</u>

PDF created with the PDFmyURL **web to PDF API**!



```
sub_10003D9E proc near
xor     eax, eax

loc_10003DA0:
add     byte_10013268[eax], al
inc     eax
cmp     eax, 134h
jb      short loc_10003DA0

retn
sub_10003D9E endp
```

```
sub_402843 proc near
xor     eax, eax

loc_402845:
add     name[eax], al
inc     eax
cmp     eax, 12Ch
jb      short loc_402845

retn
sub_402843 endp
```

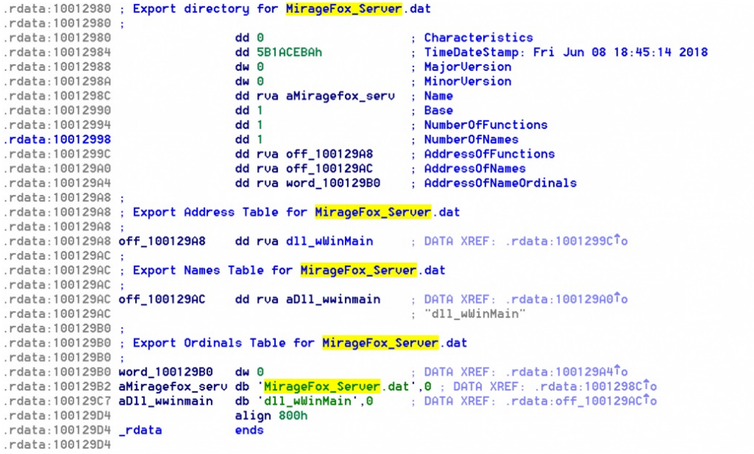
Another small, but same important function in the photo above, is the function for decrypting the data containing the C&C configuration. Similar to Reaver as posted by Palo Alto, it gets the IP or domain of the C&C server, the port, name of the binary, a sleep timer, and what Palo Alto calls a "campaign identifier."

Technical Details

At this moment, we were unable to retrieve the original infection vector and other information regarding what other tools the APT15 group is using to attack their targets. *We are able to come up with a few very interesting conclusions about what is going on here, although we cannot say for sure what the case is without the full context.*

Firstly, the reason this has been named MirageFox instead of just Mirage, is because in the Export directory for the modules, the name field is filled with a string MirageFox_Server.dat.

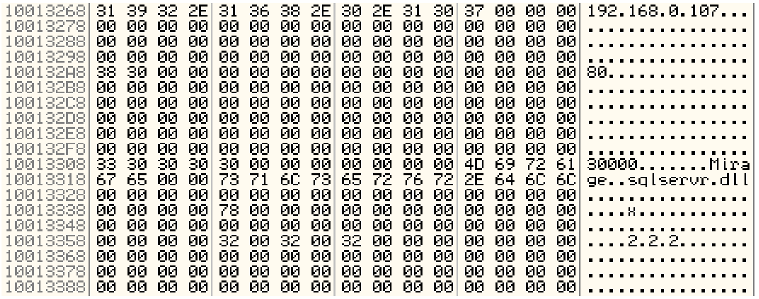
PDF created with the PDFmyURL **web to PDF API**!



```
.rdata:10012980 ; Export directory for MirageFox_Server.dat
.rdata:10012980 ;
.rdata:10012980                 dd 0                    ; Characteristics
.rdata:10012984                 dd 5B1ACEBAh            ; TimeDateStamp: Fri Jun 08 18:45:14 2018
.rdata:10012988                 dw 0                    ; MajorVersion
.rdata:1001298A                 dw 0                    ; MinorVersion
.rdata:1001298C                 dd rva aMiragefox_serv  ; Name
.rdata:10012990                 dd 1                    ; Base
.rdata:10012994                 dd 1                    ; NumberOfFunctions
.rdata:10012998                 dd 1                    ; NumberOfNames
.rdata:1001299C                 dd rva off_100129A8     ; AddressOfFunctions
.rdata:100129A0                 dd rva off_100129AC     ; AddressOfNames
.rdata:100129A4                 dd rva word_100129B0    ; AddressOfNameOrdinals
.rdata:100129A8 ;
.rdata:100129A8 ; Export Address Table for MirageFox_Server.dat
.rdata:100129A8 ;
.rdata:100129A8 off_100129A8    dd rva dll_wWinMain     ; DATA XREF: .rdata:1001299C↑o
.rdata:100129AC ;
.rdata:100129AC ; Export Names Table for MirageFox_Server.dat
.rdata:100129AC ;
.rdata:100129AC off_100129AC    dd rva aDll_wwinmain    ; DATA XREF: .rdata:100129A0↑o
.rdata:100129AC                                         ; "dll_wWinMain"
.rdata:100129B0 ;
.rdata:100129B0 ; Export Ordinals Table for MirageFox_Server.dat
.rdata:100129B0 ;
.rdata:100129B0 word_100129B0   dw 0                    ; DATA XREF: .rdata:100129A4↑o
.rdata:100129B2 aMiragefox_serv db 'MirageFox_Server.dat',0 ; DATA XREF: .rdata:1001298C↑o
.rdata:100129C7 aDll_wwinmain   db 'dll_wWinMain',0     ; DATA XREF: .rdata:off_100129AC↑o
.rdata:100129D4                 align 800h
.rdata:100129D4 _rdata          ends
.rdata:100129D4
```

Evidently in the image, you can see there is an exported function. The MirageFox binaries export a function called *dll_wWinMain,* the name of an export in *vsodscpl.dll*, a module by McAfee that is loaded by a few of their executables that import and call this function. This most likely means there is some type of DLL hijacking going on by distributing a legitimate McAfee binary with MirageFox to load up the DLL properly into a legitimate looking process. DLL hijacking techniques have been seen in the past with the APT15 group. The problem here is that once the export is called the first time, the module renames itself to sqlsrver.dll and there is no evidence within the module of any type of persistence. By renaming it to this, the future executions of the RAT will not be through a McAfee binary. The future persistence could be setup through another component of the malware or even a command sent by the C&C to the infected computer.

The most interesting part is the decrypted C&C configuration, as can be seen in the image below.



```
10013268|31 39 32 2E|31 36 38 2E|30 2E 31 30|37 00 00 00|192.168.0.107...
10013278|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013288|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013298|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
100132A8|38 30 00 00|00 00 00 00|00 00 00 00|00 00 00 00|80..............
100132B8|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
100132C8|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
100132D8|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
100132E8|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
100132F8|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013308|33 30 30 30|30 00 00 00|00 00 00 00|4D 69 72 61|30000.......Mira
10013318|67 65 00 00|73 71 6C 73|65 72 76 72|2E 64 6C 6C|ge..sqlservr.dll
10013328|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013338|00 00 00 00|78 00 00 00|00 00 00 00|00 00 00 00|....x...........
10013348|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013358|00 00 00 00|32 00 32 00|32 00 00 00|00 00 00 00|....2.2.2.......
10013368|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013378|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
10013388|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|................
```

PDF created with the PDFmyURL **web to PDF API**!

Decrypted Config:

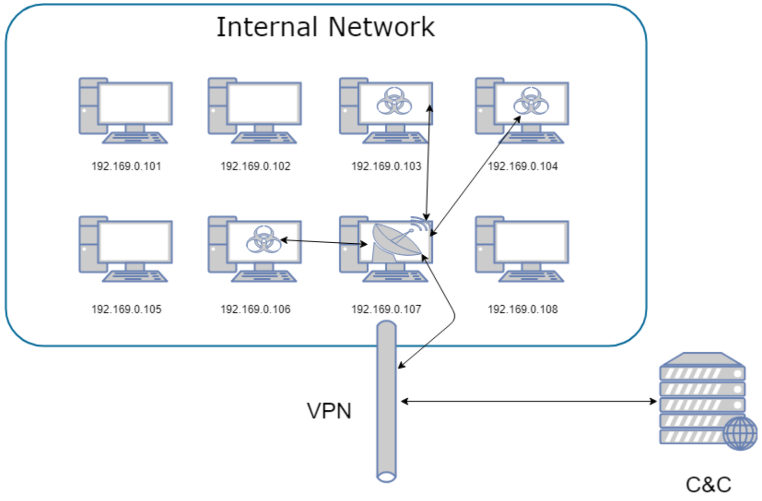
C&C IP: 192.168.0.107

Port: 80

Sleep Timer: 30000

Campaign Identifier: Mirage

If you look at it the decrypted configuration, you may notice that the ***IP being used for the C&C is an internal IP address***. If you read the report mentioned above about RoyalAPT by NCC Group, it is mentioned that APT15 infiltrated an organization again after stealing a VPN private key, ***therefore we can assume this version was tailor made to an organization they have already infiltrated and are connecting to the internal network using a VPN***.



The rest of MirageFox functions similarly to previous malware created by APT15, first collecting information about the computer like the username, CPU information, architecture, and so forth. Then it sends this information to the C&C, opens a backdoor, and sits waiting for

PDF created with the PDFmyURL **web to PDF API**!

commands from the C&C with functionality such as modifying files, launching processes, terminating itself, and more functionality typically seen in APT15's RATs.

Conclusion

There is high confidence that MirageFox can be attributed to APT15 due to code and other similarities in the MirageFox binaries. As is known about APT15, after infiltrating their target, they conduct a lot of reconnaissance work, send the commands from the C&C manually, and will customize their malware components to best suit the environment they have infected.

IOCs

MirageFox

28d6a9a709b9ead84aece250889a1687c07e19f6993325ba5295410a478da30a

97813e76564aa829a359c2d12c9c6b824c532de0fc15f43765cf6b106a32b9a5

b7c1ae10f3037b7645541acb9f7421312fb1e164be964ee7acd6eb1299d6acb2

RoyalAPT

016948ec7743b09e41b6968b42dfade5480774df3baf915e4c8753f5f90d1734

RoyalAPT C&C

buy.healthcare-internet[.]com

Mirage (w/ Same C&C Config Decryption)

5787eff4b4d6ee5b4c5457c9af5e2f2e3c75204b5989606b1faa9b944dc49a30

b6bd5d8f5a824db05c37dde459b60a5571df87966e00390f2df56628da49b856

b9403fb1e3743617bcdf8c1e5dd332c325c1e1f2e79bef166261fec0091880cf

PDF created with the PDFmyURL **web to PDF API**!

ffaddb93042243926a964864e21a28365807ac5be843f5e690f9916cddbbd55b

b0a2923e817ac982c89510e4bd8eab68892ee51e5fa625bd806508a4b701aa78

da4dbc738d069fbcc9b96ab4af2bd3f7a87c7b69a4b47071e099e36b481dfa01

f633df1fb42666f62eb23fd70dac4e3c0c4908af123f9335f3b58e6ea205df8a

e67e58bc736bd54e6915cb43af5f3c332da3592839a5a4884ba141b089310815

1534432fafb21c0479343bc2d9f3991e56c75baa41c54b3470d41055bb578f8f

27a0ce9761363f6a1eafc719b96bbe1f9a426e50e8b5abf84db963efddb89a8d

d22c2ef1453d5575e05a673777931e07c44734fe467a77969bebe86e26aacf98

f85023ae81917a7fae0d987134a968ffad346d5c3b35d3a98e237419dd334696

24b3c3527a2431d1c1dd27fe6566ddcaa8e4b92e31e468bb733e827350830a14

57550ab2d20a757b24137ab764a2e9bf644fd8e1f4313bca22e04db7fa608cc2

4d45ddc35abf77cded21bafe5483d345585c1d84f52a101a65ebfda88be5ad7d

421f4c83898ff3ae9b2a94621140ef770888a8a0914b163cdae4690433173899

c27fb5fd362fdaec2e344564f76197d01e1dc463ee3552b16452fc41e67f2709

cec9c4e48fad6e4c2b7cf4bc34d357893ef878e8be076c9f680f297e06184c20

By Jay Rosenberg

Jay Rosenberg is a self-taught reverse engineer from a very young age (12 years old), specializing in Reverse Engineering and Malware Analysis. Currently working as a Senior Security Researcher in Intezer.

Share:

PDF created with the PDFmyURL **web to PDF API**!

Register to our free community

Digital Certificates- When The Chain Of T.

PDF created with the PDFmyURL **web to PDF API**!

PDFmyURL