DEVELOPER'S GUIDE

ADFILTER PLUGIN

Table of Contents

About AdFilter

AdFilter is a browser plugin which blocks embarrassing image ads belonging to the following categories- dating ads, matrimony ads, nightwear ads, adult site ads. The plugin is a modified version of the widely popular Adblock plus tool (https://adblockplus.org).

The main motivation behind the tool was to address the concerns of web users towards being shown embarrassing online ads as well as provide a better alternative to advertisers, publishers and the users which current ad-blocking tools fail to provide. Online advertisements allow the advertisers to advertise their products to millions of users at a relatively cheaper costs. Also they are a good source of revenue for many website publishers and help in keeping the users updated with the new products in the market. Current ad-blocking tools block all ads and do not give an option to the users for selective ad-blocking. Hence there is a need for a browser plugin which blocks embarrassing ads only in contrast to blocking all ads.

Downloading the development build

AdFilter is a modified version of Adblock Plus Tool. The Adblock Plus development build can be downloaded using the following link:

```
https://adblockplus.org/downloads/adblockplus-2.2.3-source.tgz
```

The AdFilter development build can be downloaded from:

https://dl.dropboxusercontent.com/u/11874439/Ad-block/AdFilter%20code.zip

Creating an AdFilter build

In order to create a build, please make sure that python (http://www.python.org/getit/) and jinja2 (http://jinja.pocoo.org/docs/intro/) module is installed. It is recommended to install the latest version of python. Installing jinja2 is optional. I had installed python 2.7 and didn't install jinja2 and was able to work with the development build.

A development build can be created by entering the "adfilter" directory (your local copy of the repository) and running the following command from the command line:

```
python build.py build
```

The script will create a development build with a name in the form adblockplus-1.0.0.0.xpi where 1.0.0.0 is the current version number. This file will contain the source code currently in the repository and all available locales. The xpi file can be directly uploaded in the Firefox window by going to the Add-ons Manager ("Tools-> Add-ons-> Extensions") and selecting the option “Install add-on from file”. Locate the xpi file and upload it.

Testing the changes

In order to test the changes, it is suggested to install the extension Auto-installer in the browser from the following link (https://addons.mozilla.org/en-US/firefox/addon/autoinstaller/). This will help to avoid creating and uploading xpi files again and again. You can make changes in the code and this extension will automatically update the changes in the browser. Assuming that the extension auto-installer is configured to use port 8888 (the default value), the changes can be pushed to the browser by running:

```
python build.py autoinstall 8888
```

The browser will automatically update the changes.

More information on working with the development build can be found at:
https://adblockplus.org/en/source

AdFilter file structure

AdFilter Code contains the following folders:

lib – It contains all the libraries required by AdFilter including libraries for implementing matching algorithms, hiding elements, blocking HTTP requests etc. The file “contentPolicy.js” contains content policy implementation, responsible for blocking things. It contains functions which gets called whenever there is HTTP request being sent. The function checks the address of the request to the filter lists to see if the request is to be blocked or not and calls the appropriate function accordingly.

chrome- It contains all the user interface related files including the XUL files, ad-filter icons and properties files.

Understanding the AdFilter code

Building the filter lists-

“Once the plugin is installed and the filter subscriptions are added, all filters are translated into regular expressions internally, even the ones that haven't been specified as such. For example, the filter ad*banner.gif| will be translated into the regular expression /ad.*banner\.gif$/”. (http://adblockplus.org/en/filters)

Also In order to increase the processing speed, AdFilter tries to create a unique string of eight characters (a “shortcut”) for matching. All shortcuts are put into a lookup table. For filters less than eight characters, it is not possible to create shortcuts and they are used as it is. Thus when a specific address has to be tested, AdFilter will first look for known shortcuts there. However, filters without a shortcut still have to be tested one after another which slows down the process. Please look at the file “websites_referred.txt” to know about the websites which were referred for creating the filter list for each category.

More information about ad-filter filters which uses the same filters as adblock plus can be found at:
http://adblockplus.org/en/faq_internal
http://adblockplus.org/en/filters
http://adblockplus.org/en/filter-cheatsheet

Blocking HTTP requests

AdFilter acts like a proxy between the browser and the web server. It monitors the HTTP request being sent to the web server and blocks HTTP requests based on the source addresses of the request. A JavaScript object called content policies gets called by AdFilter whenever there is something to be loaded in the browser. It checks the address and matches it with a list of ad-blocking filters to decide whether it should be allowed to load.

The file “contentPolicy.js” contains an implementation of the nsIContentPolicy xpcom Interface for controlling loading of various types of content. This allows the http requests being sent to be filtered if they match the filter list.

Whenever there is a http request being sent, the shouldLoad() function of the nsIContentPolicy interface calls the processNode() function. The processNode() function checks if the address of the request belongs to a whitelist filter. If it belongs to the whitelist filter it doesn’t block the request. Otherwise it calls the matchesAny() function present in “matcher.js” which checks if the address matches with any of the items in the filter list. If a match occurs, the processNode()

functions returns “false” to the calling function shouldLoad() which then simply blocks the requests from being sent.

The value returned from processNode() is (!match || match instanceof WhitelistFilter). So every time there is a match, which means that the value of the variable match will not be null, and hence the value “false” is returned.

Modifications done to Ad-Block Plus’s code

Since Adblock plus blocks HTTP requests, the requests for loading any iframes and javascripts are also blocked. We did some modification to adblock plus’s code to block only certain ads which match the list of filters defined by us. In order to allow these elements to load, we checked if the request was being sent to load an iframe or a JavaScript and if it was true we allowed it to load. The following code was used for the same:-

```
If ((node.nodeName=="IFRAME")||(node.nodeName=="SCRIPT"))
        match=null;
```

This allowed the content of the iframes and JavaScript to be loaded in the browser.

Another modification done was to get the source URLs for the image ads. Usually for image ads, when a request is sent to the server to fetch the images, the address on which the HTTP requests are sent is the image address and not the ad source URL.

For example, in the HTML code given below for an image advertisement, the HTTP request is sent to the address http://cdn.adnxs.com/p/51.jpg which is the source of the image. However, we require the ad source url (http://nym1.ib.adnxs.com/click? ww2.shaadi.com) for doing the matching.

```
<a href="http://nym1.ib.adnxs.com/click? ww2.shaadi.com" target="_blank">
<img width="468" height="60" style="border-style: none" src="http://cdn.adnxs.com/p/51.jpg">
</a>
```

The ad source url is obtained by using the following code:

```
If (node.nodeName=="IMG")
        locationText=node.parentNode.getAttribute('href');
```

We see if the request is being sent for an image element and if it is so we get the ad-src url by accessing the parent node's href attribute.

Getting the landing page URL of an ad

The matching is first done on the source URL. In most of the cases the landing page URL is present inside the source URL in a parameter called as adurl. For the cases where the landing page URL is not present inside the source URL we send a HTTP get request on the ad-source URL to get the landing page URL and then do a matching on the landing page URL.

We used the XMLHttpRequest (XHR) API for the same which sends HTTP get requests directly to a web server and returns the HTTP response from the server. The Location field in the response header contains the landing page URL. This feature is implemented in the geturl() function in "contentPolicy.js".

However, to allow redirects, we need to set the redirectionLimit to 0 as shown in the code below.

```
if (request.channel instanceof Ci.nsIHttpChannel)
        request.channel.redirectionLimit = 0;
```

Creating the filter lists

The filter list used for blocking embarrassing ads contain urls of websites belonging to basically 4 broad categories- Dating, Matrimony, Lingerie and Adult websites. The URLs for these websites are all compiled in one list but with a separate section for each of them. We referred to alexa.com, ranker.com and many other websites on the internet to create a comprehensive lists of such websites. We were able to create a list with around 200 unique websites for each category.

Apart from the URLs, some addition filters options have been added to each URL. The option "$image" has been added to determine that only image ads directing to such domains should be blocked. The option "domain" has been included to implement domain restrictions and not allow HTTP requests to be blocked from a particular site, when the user is actually visiting that particular site.

For example,

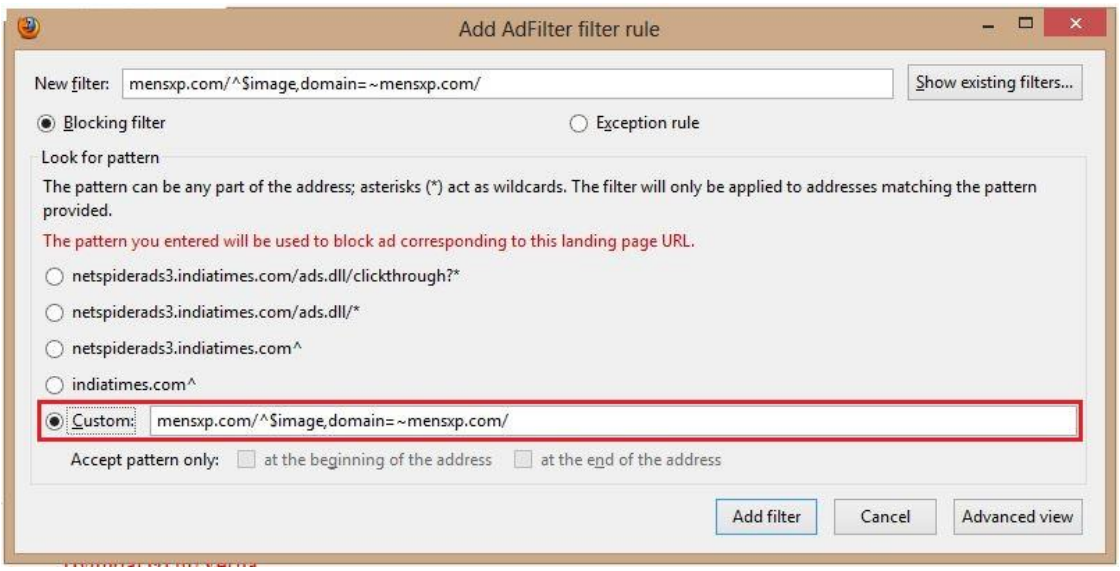
```
shaadi.com^$image, domain=~shaadi.com
```

Will block any image ad which redirects to the domain shaadi.com. However, no HTTP requests sent to shaadi.com is blocked when the user is visiting the website shaadi.com.

More information on writing ad-block plus filter lists can be found at:
http://adblockplus.org/en/filters

Allowing users to add landing page URLs to the filter list

AdFilter allows users to block an item by adding source URLs to the filter list. Since the filter list used for blocking embarrassing ads consists of landing page URL, we modified the code to give an option to the users to directly add the landing page URL to the filter list. The "Custom" option in the "Add AdFilter filter rule" dialog box contains this filter along with the additional filter options.



We added the geturl() function in the file "composer.js" in the folder chrome\content\ui. The geturl() functions processes a HTTP get requests and fetches the landing page URL from the response header. We get the domain of the landing page from the URL and add the options "$image" and "domain" to the filter.

```
geturl(item.location, function(locationres){
    if((!(locationres==null))&&(!(locationres==undefined))){
        locationres = locationres.replace(/^[\w\-]+:\/+(?:www\.)?/, "");
        let suffix = "^$image,domain=~"+ locationres;
        E("customPattern").value = locationres + suffix; }
```

```
    else {
        E("customPattern").value = item.location;
    }
});
```

Limitations of the tool

- The tool only blocks image ads. It does not work for flash and text ads. Due to the way the text and flash ads are implemented, it is not possible to get the landing page URL of these ad and hence not possible to block such ads with the current implementation of the AdFilter tool.
- The tool contains a predefined list of URLs. There still could be some embarrassing ads which may not be included in the list.
    - In such a case, the user will have to manually add that filter to the list using the tool.

Future Work

- Currently the embarrassing filter list blocks ads belonging to the following 4 categories- dating ads, matrimony ads, nightwear ads, adult site ads. We plan to extend the category list to wider range of topics which also include sensitive categories like health, religion etc.
- We plan to give topical preferences to the users where they can choose the categories from a list for which they want ads to be blocked. The user can then select the topics of ads they wish to block. The tool will then only block ads belonging to the categories selected by the users.
- AdFilter uses a filter list containing a list of URLs of ads to be blocked. In the near future, we plan to use a clustering algorithm to block ads based on topics instead of a filter list.
- Update the icon for the AdFilter tool.