By Falko Timme
Published: 2007-08-19 19:43

Build Your Own Video Community With Lighttpd And FlowPlayer (Debian Etch)

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 08/14/2007

This article shows how you can build your own video community using **lighttpd** with its `mod_flv_streaming` module (for streaming `.flv` videos, the format used by most major video communities such as **YouTube**) and its `mod_secdownload` module (for preventing hotlinking of the videos). I will use **FlowPlayer** as the video player, a free Flash video player with support for lighttpd's `mod_flv_streaming` module. I will also show how you can encode videos (`.mp4 .mov .mpg .3gp .mpeg .wmv .avi`) to the FLV format supported by Adobe Flash.

This document comes without warranty of any kind! I want to say that this is not the only way of setting up such a system. There are many ways of achieving this goal but this is the way I take. I do not issue any guarantee that this will work for you!

1 Preliminary Note

In this tutorial I use the hostname `server1.example.com` with the IP address `192.168.0.100`. These settings might differ for you, so you have to replace them where appropriate.

We need a lighttpd installation with PHP support, as shown in this tutorial: **Installing Lighttpd With PHP5 And MySQL Support On Debian Etch**. I won't cover this here, so please refer to this tutorial if you haven't already set up lighttpd with PHP support.

2 Installing LAME

LAME is an MPEG Audio Layer III (MP3) encoder. We need this so that our videos don't lose their sound while they are being converted to FLV. Unfortunately, LAME isn't available as an official Debian Etch package, so we must compile it manually. First, we install the tools we need for the compilation:

```
apt-get install build-essential
```

Then we go to the */tmp* directory and download the latest LAME version from **SourceForge**, e.g. like this:

```
cd /tmp

wget http://mesh.dl.sourceforge.net/sourceforge/lame/lame-3.97.tar.gz
```

Then we unpack and compile LAME:

```
tar xvfz lame-3.97.tar.gz

cd lame-3.97

./configure --enable-shared --prefix=/usr

make

make install
```

3 Installing ffmpeg

We will use ffmpeg to convert our video files to the FLV format. First, we install ffmpeg and a few plugins like this:

```
apt-get install ffmpeg libavcodec0d libavformat0d libavifile-0.7c2 libpostproc0d libasound2-plugins avifile-player avifile-utils
avifile-mad-plugin avifile-mjpeg-plugin avifile-vorbis-plugin
```

The problem with Debian's ffmpeg package is that is doesn't come with MP3 encoding support, which means that our FLV videos will lose their sound after the conversion. Therefore we will recompile Debian's ffmpeg source package with mp3lame support (which is why we had to install LAME in the previous

chapter).

First we download the ffmpeg source package to */usr/src*:

```
cd /usr/src/

apt-get source ffmpeg
```

Then we change to the ffmpeg source directory:

```
cd ffmpeg-0.cvs20060823
```

and edit the file *debian/rules*. Towards the beginning of that file, you will find two *confflags* lines. Add the configuration switch *--enable-mp3lame* to one of them and save the file:

```
vi debian/rules
```

```
[...]
confflags += --enable-gpl --enable-pp --enable-pthreads --enable-mp3lame
confflags += --enable-vorbis --enable-libogg --enable-a52 --enable-dts --enable-libgsm
[...]
```

Now we can build our new ffmpeg package:

```
dpkg-buildpackage
```

dpkg-buildpackage will most likely complain about missing packages that it needs to build the new ffmpeg *.deb* package:

```
server1:/usr/src/ffmpeg-0.cvs20060823# dpkg-buildpackage
  dpkg-buildpackage: source package is ffmpeg
  dpkg-buildpackage: source version is 0.cvs20060823-8
  dpkg-buildpackage: source changed by Sam Hocevar (Debian packages) <sam+deb@zoy.org>
  dpkg-buildpackage: host architecture i386
  dpkg-buildpackage: source version without epoch 0.cvs20060823-8
  dpkg-checkbuilddeps: Unmet build dependencies: debhelper (>= 4.0) quilt libogg-dev libvorbis-dev liba52-dev libdts-dev
zlib1g-dev libsdl1.2-dev libfreetype6-dev libimlib2-dev texi2html libraw1394-dev libdc1394-13-dev libtheora-dev (>>
0.0.0.alpha4) libgsm1-dev
  dpkg-buildpackage: Build dependencies/conflicts unsatisfied; aborting.
  dpkg-buildpackage: (Use -d flag to override.)
server1:/usr/src/ffmpeg-0.cvs20060823#
```

If you see an error like this, install the missing packages, e.g. like this:

```
apt-get install debhelper quilt libogg-dev libvorbis-dev liba52-dev libdts-dev zlib1g-dev libsdl1.2-dev libfreetype6-dev libimlib2-dev texi2html
libraw1394-dev libdc1394-13-dev libtheora-dev libgsm1-dev
```

Afterwards, run *dpkg-buildpackage* again:

```
dpkg-buildpackage
```

The *dpkg-buildpackage* command should now compile ffmpeg again and create new *.deb* packages (ffmpeg plus some plugins) in the */usr/src* directory. This can take some time, so please be patient. It's possible that you get some warnings about signatures at the end - you can ignore them.

Afterwards, we go to the */usr/src* directory and install our new .deb packages:

```
cd ..

dpkg -i *.deb
```

That's it for ffmpeg.

4 Installing flvtool2

When we convert videos to the FLV format, we need to add some metadata such as the duration of the video to the FLV file so that FlowPlayer can properly display the length of the video. We can add this metadata with **flvtool2**. flvtool2 is written in Ruby, so we must install Ruby first:

```
apt-get install ruby
```

Then we download the latest version of flvtool2 to the */tmp* directory, e.g. like this:

```
cd /tmp

wget http://rubyforge.org/frs/download.php/17497/flvtool2-1.0.6.tgz
```

Afterwards, we install it:

```
tar xvfz flvtool2-1.0.6.tgz

cd flvtool2-1.0.6

ruby setup.rb config

ruby setup.rb setup

ruby setup.rb install
```

5 Creating Video Directories

In this tutorial I'm assuming that your lighttpd document root for your video web site is */var/www* (the default document root for lighttpd on Debian). Of

course, we don't want to store the original videos and the FLV videos in the document root (or a subdirectory) to prevent that anyone can download them directly (if he knows the link). Therefore we create a directory for the original videos (e.g. */var/videos/incoming*) and a directory for the FLV videos (e.g. */var/videos/flv*) outside the document root:

```
mkdir -p /var/videos/incoming

mkdir -p /var/videos/flv
```

You (or your users) can then upload their original videos to */var/videos/incoming* (e.g. through FTP or some web interface that you program), and you can then encode the videos to FLV (either manually or through some script), as shown in the next chapter.

6 Encoding Videos To FLV

Let's assume we have a video called *video.avi* in */var/videos/incoming* (works for the extensions *.mp4 .mov .mpg .3gp .mpeg .wmv* as well). We want to convert it to the file *video.flv* and store it in the directory */var/videos/flv*. I want *video.flv* to have a size of 320x240 pixels with an audio sampling frequency of 44100 Hz and a frame rate of 12 fps. This is how we do it:

```
ffmpeg -i /var/videos/incoming/video.avi -s 320x240 -ar 44100 -r 12 /var/videos/flv/video.flv
```

(For more options, take a look at

```
man ffmpeg
```

)

This can take some time, and the output should look something like this:

```
server1:~# ffmpeg -i /var/videos/incoming/video.avi -s 320x240 -ar 44100 -r 12 /var/videos/flv/video.flv
FFmpeg version SVN-rUNKNOWN, Copyright (c) 2000-2004 Fabrice Bellard
  configuration:  --enable-gpl --enable-pp --enable-pthreads --enable-mp3lame --enable-vorbis --enable-libogg
```



```
 --enable-a52 --enable-dts --enable-libgsm --enable-dc1394 --disable-debug --enable-shared --prefix=/usr
  libavutil version: 0d.49.0.0
  libavcodec version: 0d.51.11.0
  libavformat version: 0d.50.5.0
  built on Aug 14 2007 15:02:25, gcc: 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)
Input #0, avi, from '/var/videos/incoming/video.avi':
  Duration: 00:10:53.8, start: 0.000000, bitrate: 5455 kb/s
  Stream #0.0: Video: mpeg4, yuv420p, 1024x576, 24.00 fps(r)
  Stream #0.1: Audio: ac3, 48000 Hz, 5:1, 448 kb/s
Output #0, flv, to '/var/videos/flv/video.flv':
  Stream #0.0: Video: flv, yuv420p, 320x240, q=2-31, 200 kb/s, 12.00 fps(c)
  Stream #0.1: Audio: mp3, 44100 Hz, stereo, 64 kb/s
Stream mapping:
  Stream #0.0 -> #0.0
  Stream #0.1 -> #0.1
No accelerated IMDCT transform found
Press [q] to stop encoding
frame= 7847 q=2.0 Lsize=   21682kB time=653.8 bitrate= 271.7kbits/s
video:16061kB audio:5108kB global headers:0kB muxing overhead 2.427536%
server1:~#
```

Please make sure that in the *Output* section, you see two streams, one for video, one for audio. If you see video only, this means that the sound gets lost which means you've probably done something wrong in chapters two and three.

After the conversion, we can now add metadata to *video.flv* with flvtool2:

```
cat /var/videos/flv/video.flv | flvtool2 -U stdin /var/videos/flv/video.flv
```

7 Configuring Lighttpd

Now we have to open lighttpd's main configuration file, */etc/lighttpd/lighttpd.conf*, and enable the modules *mod_secdownload* and *mod_flv_streaming* in it. It is very important that *mod_secdownload* is listed before *mod_flv_streaming* in the *server.modules* stanza. When I did it

the other way round, I found that fast-forwarding the video in FlowPlayer didn't work!

```
vi /etc/lighttpd/lighttpd.conf
```

```
[...]
server.modules              = (
        "mod_access",
        "mod_alias",
        "mod_accesslog",
        "mod_fastcgi",
#        "mod_rewrite",
#        "mod_redirect",
#        "mod_status",
#        "mod_evhost",
#        "mod_compress",
#        "mod_usertrack",
#        "mod_rrdtool",
#        "mod_webdav",
#        "mod_expire",
        "mod_secdownload",
        "mod_flv_streaming",
#        "mod_evasive"
 )
[...]
```

In the same file, we add also add the following configuration (you can add it right at the end of `/etc/lighttpd/lighttpd.conf`):

```
[...]
flv-streaming.extensions = ( ".flv" )
```

```
secdownload.secret          = "somesecret"
secdownload.document-root   = "/var/videos/flv/"
secdownload.uri-prefix      = "/dl/"
secdownload.timeout         = 120
```

Please replace *somesecret* with your own secret string (you can choose one).

What *mod_secdownload* does is this: a web application (e.g. a PHP script) can have a link in it of the following form:

<uri-prefix>/<token>/<timestamp-in-hex>/<rel-path>

e.g.

/dl/d8a8cb150f7e5962f6a8443b0b6c6cc2/46c1d9f6/video.flv

where *<token>* is an MD5 of
- a secret string (user supplied)
- *<rel-path>* (starts with /)
- *<timestamp-in-hex>*

mod_secdownload will then map this link to the appropriate file in the *secdownload.document-root* (which is outside the document root of the web site) and allow access to that file for *secdownload.timeout* seconds. After *secdownload.timeout* seconds, the link isn't valid anymore, and access is denied.

After we have installed FlowPlayer, we will use a PHP script to generate the appropriate video links for *mod_secdownload*.

You can find more information about *mod_secdownload* here: **http://trac.lighttpd.net/trac/wiki/Docs%3AModSecDownload**

Don't forget to restart lighttpd after your changes to */etc/lighttpd/lighttpd.conf*:

```
/etc/init.d/lighttpd restart
```

8 Installing FlowPlayer

Go to **http://flowplayer.org/download** and download the latest FlowPlayer version to your */tmp* directory, e.g. like this:

```
cd /tmp

wget http://belnet.dl.sourceforge.net/sourceforge/flowplayer/flowplayer-1.19.zip
```

FlowPlayer comes in *.zip* format, so we must install unzip to uncompress it:

```
apt-get install unzip
```

Afterwards we can uncompress it:

```
unzip flowplayer-1.19.zip
```

This creates a directory called *flowplayer* in the */tmp* directory. I'd like to have that directory in the document root of my video web site (*/var/www*), so I move it there:

```
mv flowplayer /var/www/
```

9 Configuring FlowPlayer

FlowPlayer is now installed, so all that is left to do is create an HTML file that lets us watch our video. I will create a PHP file for this called */var/www/flowplayertest.php* which contains all parameters to start FlowPlayer in the user's browser and which also create valid video links for *mod_secdownload*:

```
vi /var/www/flowplayertest.php
```

```
<?php

$secret = "somesecret";
$uri_prefix = "/dl/";

# filename
$f = "/video.flv";

# current timestamp
$t = time();

$t_hex = sprintf("%08x", $t);
$m = md5($secret.$f.$t_hex);

?>
<html>
<head>
<title>Flowplayer Test</title>
</head>
<body text="#000000" bgcolor="#FFFFFF" link="#FF0000" alink="#FF0000" vlink="#FF0000">

<object type="application/x-shockwave-flash" data="/flowplayer/FlowPlayerThermo.swf"
   width="320" height="256" id="FlowPlayer">
   <param name="allowScriptAccess" value="sameDomain" />
   <param name="movie" value="/flowplayer/FlowPlayerThermo.swf" />
   <param name="quality" value="high" />
   <param name="scale" value="noScale" />
   <param name="allowFullScreen" value="true" />
   <param name="flashvars" value="config={videoFile: '<?php printf('%s%s/%s%s', $uri_prefix, $m, $t_hex, $f, $f); ?>', streamingServer: 'lighttpd', loop: 'false', useNativeFullScreen: true}" />
</object>
```

```
</body>
</html>
```

It's very important that *$secret* has the same value than *secdownload.secret* in */etc/lighttpd/lighttpd.conf*. Also, *$uri_prefix* and *secdownload.uri-prefix* must match. If this is fulfilled, the above script will generate valid links. *$f* must hold the filename of the FLV video, beginning with a slash (*/*). (The filename is hard-coded in the above example, but of course you can program whatever you like to dynamically generate the filename.)

The *<object></object>* stanza contains the FlowPlayer configuration. FlowPlayer comes with some different skins (see **http://flowplayer.org/documentation/quick+start**); I like the *FlowPlayerThermo* skin most, so I use that. Our FLV video has a size of 320x240px (see chapter six), and the control bar of the *FlowPlayerThermo* skin has a height of 16px. Therefore I specify a width of 320px and a height of (240 + 16 = 256)px. If you use another skin, adjust the height appropriately.

In the *<param name="flashvars"...* line we can configure the behaviour of FlowPlayer. The most important setting is the *videoFile* setting (which is set by PHP in the above script) which specifies the path to the FLV video.

Another important setting is *streamingServer: 'lighttpd'* which causes the video to be streamed through lighttpd's *mod_flv_streaming* module which allows better support for long videos.

The other two settings (*loop*, *useNativeFullScreen*) are optional in this setup. However, if you set *useNativeFullScreen* to *true* (adds support for **native Flash 9 full screen mode**), you must also include the line

```
<param name="allowFullScreen" value="true" />
```

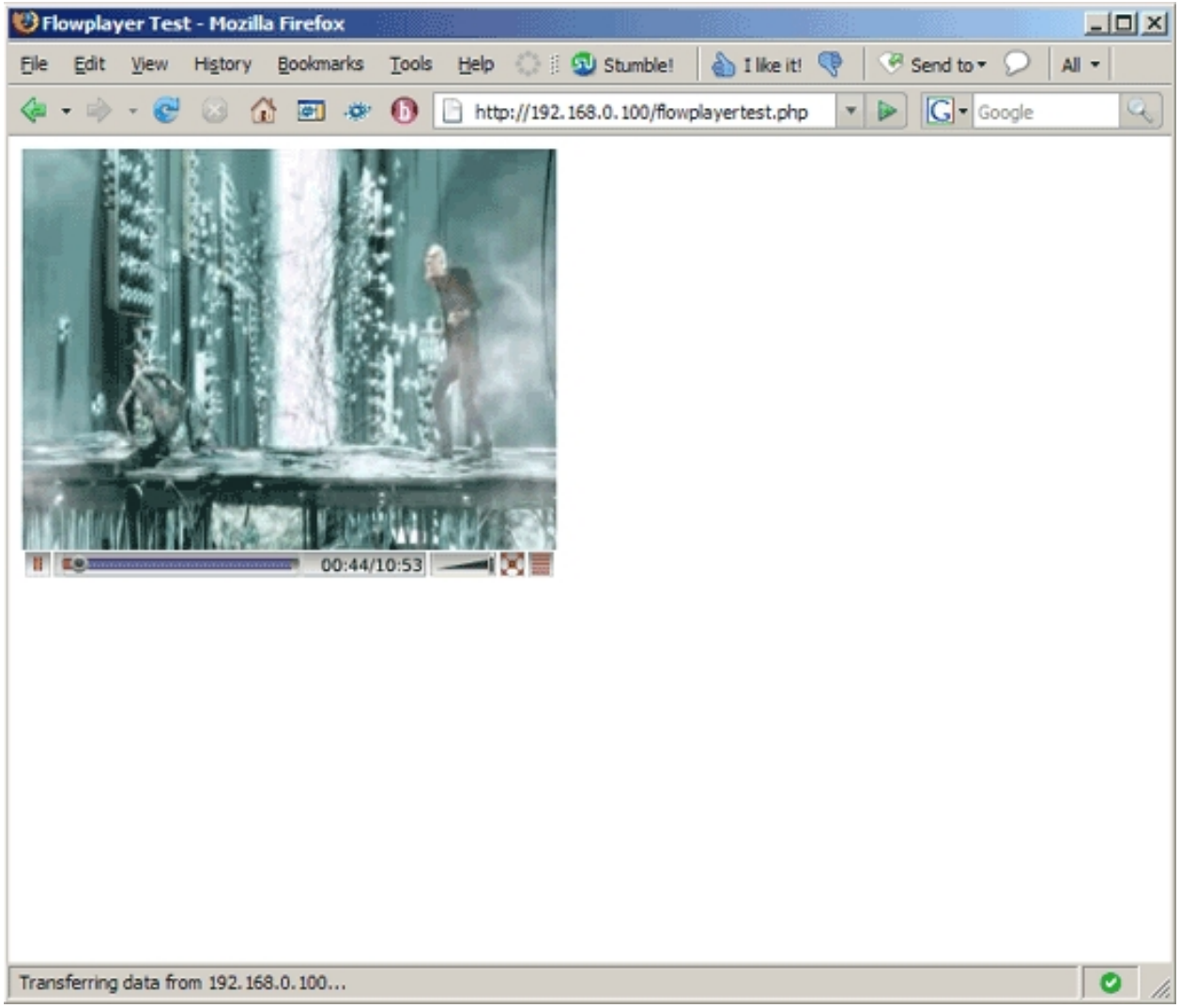
in the *<object></object>* stanza, and you cannot use

```
 <param name="wmode" value="transparent" />
```

then.

To learn more about all options that you can use in the *<param name="flashvars"...* line, take a look at **http://flowplayer.org/config/variables**.

Now it's time to test our setup. Direct your browser to *http://192.168.0.100/flowplayertest.php* or *http://server1.example.com/flowplayertest.php*, and your video should start to play in your browser (including sound):



This is how the native Flash 9 full screen mode looks:

10 Links

- Lighttpd: **http://www.lighttpd.net**
- mod_flv_streaming: **http://blog.lighttpd.net/articles/2006/03/09/flv-streaming-with-lighttpd**
- mod_secdownload: **http://trac.lighttpd.net/trac/wiki/Docs%3AModSecDownload**
- FlowPlayer: **http://flowplayer.org**
- LAME: **http://lame.sourceforge.net**

- ffmpeg: **http://ffmpeg.mplayerhq.hu**
- flvtool2: **http://rubyforge.org/projects/flvtool2**
- PHP: **http://www.php.net**
- Debian: **http://www.debian.org**