import xml.etree.ElementTree as etree

import base64

from struct import unpack, pack

import sys

import io

import os

import time

import itertools

import xbmcaddon

import xbmc

import urllib2,urllib

import traceback

import urlparse

import posixpath

import re

import socket

from flvlib import tags

from flvlib import helpers

from flvlib.astypes import MalformedFLV

import zlib

from StringIO import StringIO

import hmac

import hashlib

import base64

addon\_id = 'script.video.F4mProxy'

selfAddon = xbmcaddon.Addon(id=addon\_id)

\_\_addonname\_\_ = selfAddon.getAddonInfo('name')

\_\_icon\_\_ = selfAddon.getAddonInfo('icon')

downloadPath = xbmc.translatePath(selfAddon.getAddonInfo('profile'))#selfAddon["profile"])

#F4Mversion=''

class interalSimpleDownloader():

outputfile =''

clientHeader=None

def \_\_init\_\_(self):

self.init\_done=False

def thisme(self):

return 'aaaa'

def openUrl(self,url, ischunkDownloading=False):

try:

post=None

openner = urllib2.build\_opener(urllib2.HTTPHandler, urllib2.HTTPSHandler)

if post:

req = urllib2.Request(url, post)

else:

req = urllib2.Request(url)

ua\_header=False

if self.clientHeader:

for n,v in self.clientHeader:

req.add\_header(n,v)

if n=='User-Agent':

ua\_header=True

if not ua\_header:

req.add\_header('User-Agent','Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36')

#response = urllib2.urlopen(req)

if self.proxy and ( (not ischunkDownloading) or self.use\_proxy\_for\_chunks ):

req.set\_proxy(self.proxy, 'http')

response = openner.open(req)

return response

except:

print 'Error in getUrl'

traceback.print\_exc()

return None

def getUrl(self,url, ischunkDownloading=False):

try:

post=None

openner = urllib2.build\_opener(urllib2.HTTPHandler, urllib2.HTTPSHandler)

if post:

req = urllib2.Request(url, post)

else:

req = urllib2.Request(url)

ua\_header=False

if self.clientHeader:

for n,v in self.clientHeader:

req.add\_header(n,v)

if n=='User-Agent':

ua\_header=True

if not ua\_header:

req.add\_header('User-Agent','Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36')

#response = urllib2.urlopen(req)

if self.proxy and ( (not ischunkDownloading) or self.use\_proxy\_for\_chunks ):

req.set\_proxy(self.proxy, 'http')

response = openner.open(req)

data=response.read()

return data

except:

print 'Error in getUrl'

traceback.print\_exc()

return None

def init(self, out\_stream, url, proxy=None,g\_stopEvent=None, maxbitRate=0):

try:

self.init\_done=False

self.init\_url=url

self.clientHeader=None

self.status='init'

self.proxy = proxy

self.maxbitRate=maxbitRate

if self.proxy and len(self.proxy)==0:

self.proxy=None

self.out\_stream=out\_stream

self.g\_stopEvent=g\_stopEvent

if '|' in url:

sp = url.split('|')

url = sp[0]

self.clientHeader = sp[1]

self.clientHeader= urlparse.parse\_qsl(self.clientHeader)

print 'header recieved now url and headers are',url, self.clientHeader

self.status='init done'

self.url=url

#self.downloadInternal( url)

return True

#os.remove(self.outputfile)

except:

traceback.print\_exc()

self.status='finished'

return False

def keep\_sending\_video(self,dest\_stream, segmentToStart=None, totalSegmentToSend=0):

try:

self.status='download Starting'

self.downloadInternal(self.url,dest\_stream)

except:

traceback.print\_exc()

self.status='finished'

def downloadInternal(self,url,dest\_stream):

try:

url=self.url

fileout=dest\_stream

self.status='bootstrap done'

while True:

response=self.openUrl(url)

buf="start"

firstBlock=True

try:

while (buf != None and len(buf) > 0):

if self.g\_stopEvent and self.g\_stopEvent.isSet():

return

buf = response.read(200 \* 1024)

fileout.write(buf)

#print 'writing something..............'

fileout.flush()

try:

if firstBlock:

firstBlock=False

if self.maxbitRate and self.maxbitRate>0:# this is for being sports for time being

print 'maxbitrate',self.maxbitRate

ec=EdgeClass(buf,url,'http://www.en.beinsports.net/i/PerformConsole\_BEIN/player/bin-release/PerformConsole.swf',sendToken=False)

ec.switchStream(self.maxbitRate,"DOWN")

except:

traceback.print\_exc()

response.close()

fileout.close()

print time.asctime(), "Closing connection"

except socket.error, e:

print time.asctime(), "Client Closed the connection."

try:

response.close()

fileout.close()

except Exception, e:

return

except Exception, e:

traceback.print\_exc(file=sys.stdout)

response.close()

fileout.close()

except:

traceback.print\_exc()

return

class EdgeClass():

def \_\_init\_\_(self, data, url, swfUrl, sendToken=False, switchStream=None):

self.url = url

self.swfUrl = swfUrl

self.domain = self.url.split('://')[1].split('/')[0]

self.control = 'http://%s/control/' % self.domain

self.onEdge = self.extractTags(data,onEdge=True)

self.sessionID=self.onEdge['session']

self.path=self.onEdge['streamName']

print 'session',self.onEdge['session']

print 'Edge variable',self.onEdge

print 'self.control',self.control

#self.MetaData = self.extractTags(data,onMetaData=True)

if sendToken:

self.sendNewToken(self.onEdge['session'],self.onEdge['streamName'],self.swfUrl,self.control)

def getURL(self, url, post=False, sessionID=False, sessionToken=False):

try:

print 'GetURL --> url = '+url

opener = urllib2.build\_opener()

if sessionID and sessionToken:

opener.addheaders = [('User-Agent', 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:14.0) Gecko/20100101 Firefox/14.0.1' ),

('x-Akamai-Streaming-SessionToken', sessionToken ),

('x-Akamai-Streaming-SessionID', sessionID ),

('Content-Type', 'text/xml' )]

elif sessionID:

opener.addheaders = [('User-Agent', 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:14.0) Gecko/20100101 Firefox/14.0.1' ),

('x-Akamai-Streaming-SessionID', sessionID ),

('Content-Type', 'text/xml' )]

else:

opener.addheaders = [('User-Agent', 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:14.0) Gecko/20100101 Firefox/14.0.1' )]

if not post:

usock=opener.open(url)

else:

usock=opener.open(url,':)')

response=usock.read()

usock.close()

except urllib2.URLError, e:

print 'Error reason: ', e

return False

else:

return response

def extractTags(self, filedata, onEdge=True,onMetaData=False):

f = StringIO(filedata)

flv = tags.FLV(f)

try:

tag\_generator = flv.iter\_tags()

for i, tag in enumerate(tag\_generator):

if isinstance(tag, tags.ScriptTag):

if tag.name == "onEdge" and onEdge:

return tag.variable

elif tag.name == "onMetaData" and onMetaData:

return tag.variable

except MalformedFLV, e:

return False

except tags.EndOfFile:

return False

f.close()

return False

def decompressSWF(self,f):

if type(f) is str:

f = StringIO(f)

f.seek(0, 0)

magic = f.read(3)

if magic == "CWS":

return "FWS" + f.read(5) + zlib.decompress(f.read())

elif magic == "FWS":

#SWF Not Compressed

f.seek(0, 0)

return f.read()

else:

#Not SWF

return None

def MD5(self,data):

m = hashlib.md5()

m.update(data)

return m.digest()

def makeToken(self,sessionID,swfUrl):

swfData = self.getURL(swfUrl)

decData = self.decompressSWF(swfData)

swfMD5 = self.MD5(decData)

data = sessionID+swfMD5

sig = hmac.new('foo', data, hashlib.sha1)

return base64.encodestring(sig.digest()).replace('\n','')

def sendNewToken(self,sessionID,path,swf,domain):

sessionToken = self.makeToken(sessionID,swf)

commandUrl = domain+path+'?cmd=sendingNewToken&v=2.7.6&swf='+swf.replace('http://','http%3A//')

self.getURL(commandUrl,True,sessionID,sessionToken)

def switchStream(self, bitrate, upDown="UP"):

newStream=self.path

print 'newStream before ',newStream

newStream=re.sub('\_[0-9]\*@','\_'+str(bitrate)+'@',newStream)

print 'newStream after ',newStream,bitrate

sessionToken =None# self.makeToken(sessionID,swf)

commandUrl = self.control+newStream+'?cmd=&reason=SWITCH\_'+upDown+',1784,1000,1.3,2,'+self.path+'v=2.11.3'

self.getURL(commandUrl,True,self.sessionID,sessionToken)