

Forum:[Hack Forums](#)

Thread: [Creating an uploading library](#)

dberube - November 1, 2012, 6:57 pm

Uploading Script Tutorial

In this tutorial, you will learn: How to handle openers

How to handle requests

How to handle multipart (file uploading)

Login in a website from python.

Get requests from a website and use them in python.

Much more...

In this tutorial, we will create a library that will enable an user to login from your script to minus.com and upload a file.

This tutorial targets the intermediate-advanced python user. You should be familiar with the python syntax, modules, functions and classes (understanding how http requests works is a good bonus though).

Summary

Required tool

Get the needed requests from the website

Teach you script how to use these requests

Full script

Required tool

When you want to upload something, you NEED to know how the website handle the requests you send to it using your browser AND you need to know the Headers used. This is where Charles enter. (any HTTP proxy / HTTP monitor can do the job just fine.)

"Charles is an HTTP proxy / HTTP monitor / Reverse Proxy that enables a developer to view all of the HTTP and SSL / HTTPS traffic between their machine and the Internet. This includes requests, responses and the HTTP headers (which contain the cookies and caching information)."

You can get Charles (mac,linux,windows) here : <http://www.charlesproxy.com>

An user also suggested Fiddler, you can get it here: <http://www.fiddler2.com/fiddler2/>

Get the needed requests from the website

Has you might guessed, you do not start coding just now. You need to get the needed requests and the website headers.

Before everything, we need to prepare our environment so close every unnecessary stuff on you computer. Open notepad (we will save the requests data & headers there), open charles and open your favourite web browser .

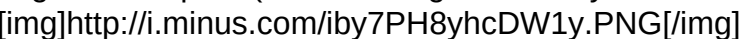
This done, lets move to the easiest part: getting the request headers!

We want our program to behave like an user on his browser. To do this we need to create the exact headers that are used on, let's say... firefox.

To get the headers go to <http://minus.com/> .

Now, on Charles, you should see new items.

Click on the "minus address", then "default" then go in "Request" and finally click on "raw". Now copy and paste everything into notepad. (see the image below if you need help)

Spoiler ([Click to View](#))

Now we have the headers that will be used later when we will be scripting.

Next step: Get the login request.

On the majority of file sharing website, you need an account and anyway we want a way to recover the file we uploaded. So all we need to do is to get the login request to be able to log in from our script.

I recommend to clear the session in Charles for better readability (CTRL+Backspace)

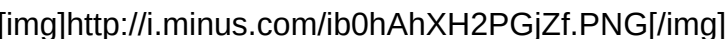
If you do not have an account, create one and logout.

Log In... now go check into charles

The request should be in <http://minus.com/api/login/login>

Like before, copy and paste the request data. (see picture and request below)

NOTE:As you can see, this is a POST request and it comes with parameters: disable_safety_mode,web-login,login-username and login-password

Spoiler ([Click to View](#))

`disable_safety_mode=0&web-login=true&login-username=dickbutt vomit&login-password=vomit dick butt`

Last Step: Get the upload request AND the create folder request.

Of course we need the upload request. The upload request, in most cases is nothing more than another POST request using multipart (that I will explain later).In minus case, there is 2 important requests: createFolder and upload. To need to get both of them.

So you know the drill: upload a file and get the requests.

See the picture if you still need help.

Spoiler [\(Click to View\)](#)

Teach you script how to use these requests

Now that we have our requests all saved within notepad, we can start to code. Since people tend to use python 2.7, this tutorial will use python 2.7.

First thing first, lets write the imports statement.

All we need is urllib2 to handle requests and cookielib to handle cookies and a couple more imports to handle multipart. So...

```
Code import cookielib
import urllib
import urllib2
import mimetools, mimetypes
import os, stat
```

Next we create a class that will hold our session

If you don't know what is a class, you are not enough experienced to follow this tutorial. We will call it "minusSession"

```
Code class minusSession():
    def __init__(self):
        pass #Placeholder
```

```
sessions = minusSession()
```

Add the requests headers.

Now that we have our class created, lets define 3 variables in the __init__() function:

self.loginHeaders=: dictionary (Hold the headers of the login requests we will send)

self.createFolderHeaders: dictionary (Hold the headers of the create folder requests we will send)

self.uploadHeaders: dictionary (Hold the headers of the upload requests we will send)

And finally, for every parameters of the first header we found, we add a key to the dictionary corresponding to the parameter name with a value corresponding to the parameter value. (Your parameters may be different from mine)

```
Code def __init__(self):
    self.loginHeaders = {}
```

```
    self.loginHeaders['Host'] = 'minus.com'
    self.loginHeaders['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0)
Gecko/20100101 Firefox/16.0'
    self.loginHeaders['Accept'] = 'application/json, text/javascript, */*; q=0.01'
    self.loginHeaders['Accept-Language'] = 'fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3'
    self.loginHeaders['Accept-Encoding'] = 'gzip, deflate'
    self.loginHeaders['Content-Type'] = 'application/x-www-form-urlencoded; charset=UTF-8'
    self.loginHeaders['X-Requested-With'] = 'XMLHttpRequest'
    self.loginHeaders['Referer'] = 'http://minus.com/'
    self.loginHeaders['DNT'] = '1'
    self.loginHeaders['Pragma'] = 'no-cache'
    self.loginHeaders['Cache-Control'] = 'no-cache'

    self.createFolderHeaders = {}
```

```

self.createFolderHeaders['Host'] = 'minus.com'
self.createFolderHeaders['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0)
Gecko/20100101 Firefox/16.0'
self.createFolderHeaders['Accept'] = 'application/json, text/javascript, */*; q=0.01'
self.createFolderHeaders['Accept-Language'] = 'fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3'
self.createFolderHeaders['Accept-Encoding'] = 'gzip, deflate'
self.createFolderHeaders['Referer'] = 'http://minus.com/'
self.createFolderHeaders['Cookie'] = ''
self.createFolderHeaders['DNT'] = '1'

```

```

self.uploadHeaders = {}
self.uploadHeaders['Host'] = 'minus.com'
self.uploadHeaders['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0)
Gecko/20100101 Firefox/16.0'
self.uploadHeaders['Accept'] = 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
self.uploadHeaders['Accept-Language'] = 'fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3'
self.uploadHeaders['Accept-Encoding'] = 'gzip, deflate'
self.uploadHeaders['Referer'] = 'http://minus.com/'
self.uploadHeaders['Content-Type'] = 'multipart/form-data; boundary=--36022435010291'
self.uploadHeaders['Cookie'] = ''
self.uploadHeaders['DNT'] = '1'
self.uploadHeaders['Pragma'] = 'no-cache'
self.uploadHeaders['Cache-Control'] = 'no-cache'
self.cjar = ''

```

Creating the login function #1

We have all our requests headers ready. Now, we want our user to be able to login so, we will create a function called "openSession" and we will give it two parameters: username and password for obvious reasons.

```
Code def openSession(self,username,password):
```

```
    pass #Placeholder
```

Creating the login function #2

First assign the username and the password to the class.

then create a variable that will hold the POST request data we will name it "data". The value it will hold will be the unformatted parameters of the login request we saved earlier. We will format it with the username and the password provided by the user using the ".format" string method.

Last thing we need to do is to define a variable that will hold a CookieJar object. A cookieJar object is nothing more than an object that will save cookies when when sent with an http opener.

```
Code def openSession(self,username,password):
```

```
    self.user = username
```

```
    self.psw = password
```

```
    data =
```

```
"disable_safety_mode=0&web-login=true&login-username={0}&login-password={1}".format(userna
me,password)
```

```
    cj = cookielib.CookieJar()
```

Creating the login function #3

Now that we have our variables all settled up, we need to send a request to the website AND keep the cookies. To do this we will build an opener (that will use an HTTPCookieProcessor and an HTTPHandler) and a request. The opener is nothing more than something that will help you send

requests. After we built everything, we send the request via the opener and we read() the response.

```
Code def openSession(self,username,password):
```

```
    self.user = username
```

```
    self.psw = password
```

```
    data =
```

```
"disable_safety_mode=0&web-login=true&login-username={0}&login-password={1}".format(userna  
me,password)
```

```
    cj = cookielib.CookieJar()
```

```
    #Create an opener to open pages using the http protocol and to process cookies.
```

```
    opener = urllib2.build_opener(HTTPCookieProcessor(cj), HTTPHandler())
```

```
    req = urllib2.Request("http://minus.com/api/login/login",data = data,headers = self.loginHeaders)
```

```
    #Create a post request
```

```
    f = opener.open(req)
```

```
    html = f.read()
```

```
Creating the login function #4
```

We have the response and a filled CookieJar. So lets get those cookies and make that function return a value depending if the login was a success or not. When we sign In successfully the response = '{"success": true}' so we can achieve this with a simple "if" loop and a "for" loop to iterate over the cookies saved in the cookieJar object.

```
Code def openSession(self,username,password):
```

```
    self.user = username
```

```
    self.psw = password
```

```
    data =
```

```
"disable_safety_mode=0&web-login=true&login-username={0}&login-password={1}".format(userna  
me,password)
```

```
    cj = cookielib.CookieJar()
```

```
    #Create an opener to open pages using the http protocol and to process cookies.
```

```
    opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj), urllib2.HTTPHandler())
```

```
    req = urllib2.Request("http://minus.com/api/login/login",data = data,headers = self.loginHeaders)
```

```
    #Create a post request
```

```
    f = opener.open(req)
```

```
    html = f.read()
```

```
    if html == '{"success": true}':
```

```
        self.uploadHeaders['Cookie'] = " #clear cookies if already logged
```

```
        for cookie in cj:
```

```
            cookie = str(cookie).split(' ')[1]
```

```
            self.uploadHeaders['Cookie'] += cookie + ';' 
```

```
            self.createFolderHeaders ['Cookie'] += cookie + ';' 
```

```
        print self.uploadHeaders['Cookie'],self.createFolderHeaders ['Cookie']
```

```
        return 1
```

```
    else:
```

```
        return 0,html
```

```
Creating the upload function #1
```

Here comes the hardest part. To upload a file to minus, we need to send 2 requests: first the "create folder" request then the actual upload request. To keep our code clean and accessible we will first define a new function within our minusSession class called "upload". To this new function we will add a parameter called "file" (file will accept a file object (NOT A STRING)).

Inside this new function we will define 2 other functions:

1:createMinusFolder(), it will handle the create folder request and will return the created folder ID (which is needed in the upload request)

2:uploadFile(file,folderID), it will handle the upload function.

Finally we call them in this order at the end of our upload function. Don't forget to assign variable to them.

Code def upload(self,fileName):

```
def createMinusFolder():  
    pass
```

```
def uploadFile(fileName,folderID):  
    pass
```

```
folderID = createMinusFolder()  
rep = upload(fileName,folderID)
```

```
return rep
```

Creating the upload function #2 (createMinusFolder)

The create folder function is no different than the login function. Actually, it is even simpler: we don't need to log cookies nor to use request data. Do has you did for the login function: create an handler (with no cookie processor this time), create the request and send it.

Code def createMinusFolder():

```
opener = urllib2.build_opener(urllib2.HTTPHandler())  
req = urllib2.Request("http://minus.com/api/CreateGallery_Web/?d=1351809273415",headers =  
self.createFolderHeaders)  
f = opener.open(req)
```

Creating the upload function #3 (createMinusFolder)

This request will return a dictionary like value representing the new folder ID so all we need to do is to "eval" the html text and return the key (which is named "editor_id"). The create folder function is now finished!

Code def createMinusFolder():

```
opener = urllib2.build_opener(urllib2.HTTPHandler())  
req = urllib2.Request("http://minus.com/api/CreateGallery_Web/?d=1351809273415",headers =  
self.createFolderHeaders) #Create a post request  
f = opener.open(req)  
html = eval(f.read())
```

```
return html['editor_id']
```

Creating the upload function #4 (multiparts)

To upload a file we will need to use multiparts. To make it simple, multiparts are file that are appended to a request. Sadly urllib and urllib2 do not directly supports multiparts BUT someone

created a function that implement it within urllib2. You can get it here: <http://pastebin.com/rnNB7njd> . All you need to do is to copy the 2 classes at the beginning of your script.

Creating the upload function #5 (uploadFile)

As always, you need to send a request with an opener but this time the opener will use the function we added in the last step (MultipartPostHandler) and the request will be slightly different: the parameters are with the url. So instead of formatting a "data" variable, we will format a "url" variable. We format the url with "folderID" and "os.path.basename(fileName.name)" (which will return the name of the file). We append a file to our request by creating a dictionary containing the fileName object with "file" as key. Finally we try to send the request with the parameter containing the file if there is an error, it return False(0) else it return True(1).

Code def uploadFile(fileName,folderID):

```
    opener = urllib2.build_opener(MultipartPostHandler)

    url =
'http://minus.com/api/UploadItem_Web/?editor_id={0}&key=-&filename={1}'.format(folderID,os.path.
basename(fileName.name))
    req = urllib2.Request(url,headers = self.uploadHeaders)
    params = {"file" : fileName }

    try:
    f = opener.open(req, params)
    return 1
    except:
    return 0
```

Testing your script

The only thing left to do is to test your script. Create a minus session in the end of your script, connect and try to upload a file

```
Code session = minusSession()
session.openSession('username','password')
#ALWAYS USE "rb" TO OPEN A FILE
session.upload(open("FILEPATH",'rb'))
Full Script: http://pastebin.com/mb8wCnrZ
```

Hydra_ - November 2, 2012, 6:15 am

Good Work dberube buddy ;)

Keep it up

creepycreeper - November 2, 2012, 2:24 pm

One of few really good tutorials on hackforums...

You should post it on your blog or something like that, because honestly, here it will be appreciated only by few people (who mostly don't need it), because most of people here are forever beginners who struggle with basics...

dberube - November 2, 2012, 2:49 pm

Thanks, that means alot to me.

And I'm porting it to my blog right now. :D