Does exposed JSON data make paid e-songs vulnerable?

Jimut Bahan Pal[1]

Abstract

JavaScript Object Notation (JSON) is a popular way of interchanging data between the client and servers. It is easy for computers to scan and create JSON objects; is it a secure way of transferring data? If the JSON is exposed, then the answer is no as JSON makes scraping easier. When raw data is scraped from the web it is useless, unless we get the meaningful items from it. With the help of automated bots some companies regularly scrap their competitors' website for continuous monitoring opponents' progress, without any human interaction. Google is a good example of scraper which scraps the entire internet at a very fast rate to store in their database, with an intention to implement page rank algorithm that indexes the pages. This paper investigates a popular website, for exposed JSON, to download paid e-songs free of cost. Results show that sites lack security for which it may result in the loss of their earnings.

Key words: Scraping, JSON, Website, e-Songs, Security, and Loss of earnings

Introduction

Several programming languages and software for scraping and extracting data from the web are available but Python 3 being the best till now. The act of collecting specific data and storing it in the local database is called scraping. It is used both for academic purposes such as Natural Language Processing (NLP), which scraps Wikipedia to train on their contents, it is usually needed for analysis of sentiment and Machine learning etc., and for

[1] 2nd Year B.Sc. (Comp), St. Xavier's College, 30 Mother Teresa Sarani, Kolkata – 700016, India. Contact: +91-8902209098. The author acknowledges for free use of the DNS picture obtained from https://www.threatcrowd.org/domain.php?domain=akdls3re.hungama.com.

illegal purposes such as stealing paid content for free, hacking, gaining user's private data for analysis, which is unethical. Web scraping programs written in Python 3 may access the www directly using the HTTP. Even files may be downloaded using wget module from a website, that may include file extension such as iso, pdf, jpeg, png, mp3, mp4 etc. Wget module is also found in Python 3.

Several measures may be taken to protect a website against scraping, some of them are:

1. Setting up robots.txt – It is an agreement to prevent crawling bots to access sensitive information from websites.
2. Filtering request by user agent – The request is filtered before the content is served to the request.
3. Blacklisting the IP – it is much easier than other methods by which IP addresses are constantly monitored and if someone tries to access more information than what is necessary, the site blocks it.
4. Throwing Captcha – It generally reduces the traffic to a website by acting as a wall between the user-agents/bots and the data.
5. Setting up honeypots – These are traps set up to catch the bots which scraps.

Literature Review

New softwares are constantly developing to scrap website easier. Social media scraping is common now-a-days. It is illegal, but some platforms like twitter allows it through API. Twitter's "gardenhouse" API is used to query and collect information for particular tweets (Hernandez-Saurez et al., 2018). Those tweets can be then used for sentiment analysis, forecasting, predicting political elections, etc. An activist Swartz (2008) scraped the database of public access to Court Electronic Records (PACER) - which charges a small fee to access the US Federal Court records. He scraped and released 2.7 million of PACER's approx., 500 million documents to public.resource.org. A couple of years later he scraped 450 K documents from JSTOR corpus. He committed suicide on being arrested and was still faced 50 years' imprisonment with a million dollars fine.

Chelsea Manning used similar techniques to scrap the whole significant activities portion of Army's intelligence database and she gave it to the WikiLeaks. Snowden too amassed an archive of 1.7 million classified documents.

They used wget module. Wget module is open sourced in mid-1990's. When wget is pointed to a network it scraps all the data present there and all the related networks that is

present. The National Security Agency (NSA) scraps and amasses enormous database of global communications data, while Google constantly crawls the internet, copying and indexing everything it can reach. For Google it helps to find the page rank and for NSA that helps to identify pattern of interaction, behaviours and possibly spot the threats before they happen (Foster, 2014).

I-macro Crawler is also capable of harnessing every information which is accessible through the browser from the Facebook website within the legal framework authorized by the Facebook (Wani, et al., 2018).

Materials and Methods

There have been several methodologies and various software, which can scrap the web for different purposes. Like ‘https://musicpleer.fm/’ have their own scraper which can scrap the metadata of free e-songs available and store it in their own database. It doesn’t need to store the actual song in their database, when a user requests a e-song’s name, it searches in their database and directs them to the url in which the e-song is actually present. Various servers kept their free data through which the site is accessing the e-songs Scraping generally implements machine learning, classification, prediction related works. Thus we can scrap stock market data easily and then use those data to predict the prices of future stock using neural networks, etc. Ajax crawling can be blocked by disallowing ‘*_escaped_fragment_’, Captcha can be blocked by disallowing ‘Googlebot-Image’. There are various kind of commercial bots like ‘bingbot’, ‘Yandex’, ‘linkdexbot’, which can be blocked easily by setting up robots.txt and for disallowing them. Therefore, considerable steps can be taken to disallow such bots.

Results and Discussions

We investigated how to download paid e-songs for free using python 3 script. Python 3’s urllib, beautiful soup 4 and wget modules helps to scrap the web easily. The urllib uses ssl module to ignore the certificates and errors. It also has re (regex) to implement regular expressions – which are a replacement to huge code for small code to find needle in haystack.

We searched ‘hungama.com’, a popular website, to create an account. We visited a e-song’s main page. We tried to download the paid e-songs (**Fig. 1**) in mp3 or mp4 version to check

for vulnerabilities. The e-song could be run on browser but for downloading requires payment. Since the e-song is running in the browser, it is using a source to get the e-song. If anybody could get the source, they could scrap the e-song by using the simple wget module in Python 3. Let's consider this example, we went to http://www.hungama.com/song/__complicated/28175886/'. After examining multiple examples like another one: 'http://www.hungama.com/song/flames/34177568/', we found that there is a pattern in which the e-songs are kept, i.e., '<http://www.hungama.com/song/>' '<song-name/>' '<song-id/>'. Now this gets really interesting from here. We checked the page-source and found another pattern. Some of the pages contains: <'https://secure.hungama.com/newTwitter_Audio/audio/index-g.php?songid='> '<song-id>'. On visiting this site, we found that it contains an exposed JSON (**Fig. 2**), which sources the mp3 or mp4 e-song. On visiting the site, we found that it has the mp3 or mp4 e-song. The former url of the e-song has something like this: 'http:\/\/akdls3re.hungama.com\/s3\/r\/ms2\/28175886\/4\/250\/Complicated.mp3?__gda__=1529308028_892338fa9676705891439aa173c748cd' ; the latter being: 'http:\/\/akdls3re.hungama.com\/s3\/r\/ms2\/34177568\/4\/250\/Flames.mp3?__gda__=1529308204_b63639387256ec38fe943957a96cb9c4'and it will be auto-corrected in browser to get the mp3 or mp4 e-song but Python 3 wget module cannot directly scrap it from the web. Some url are encoded in JavaScript such as: 'http:\/\/hungama.pc.cdn.bitgravity.com\/wsasecure\/audio\/9\/c3\/17986985_b32_a2.mp4?e=1529223731&h;=65362aeba4ca6312b340bcee92081a8f&track;=songs.mp3' which have song id 17986985. Here we will do small decryption - such as '&h;' by '&h' and '&track;' by '&track'. This is automatically converted by the browser and can be accessed, but this cannot be directly accessed by Python 3 script. So after implementing the conversion we get the following: 'http:\/\/hungama.pc.cdn.bitgravity.com\/wsasecure\/audio\/9\/c3\/17986985_b32_a2.mp4?e=1529223731&h=65362aeba4ca6312b340bcee92081a8f&track=songs.mp3', which still have some problems. For example, '\ /' pair is ignored by the browser but this cannot be accessed via a Python 3 script so another transition is needed – which replaces "\" by "/". After this transition we get 'http:////hungama.pc.cdn.bitgravity.com//wsasecure//audio//9//c3//17986985_b32_a2.mp4?e=1529223731&h=65362aeba4ca6312b340bcee92081a8f&track=songs.mp3', which too cannot be accessed by Python 3 directly as shown in **Fig. 3**, as unformatted url links. Just a minor replacement '////' to '//' makes it possible to directly and securely access the website

using Python 3 script. Now, we use the wget module to directly download it and we get the paid e-songs free of cost. However, most of the e-songs can be downloaded, which have 'newTwitter_Audio' API, but about 10% cannot be downloaded as it uses some different or latest twitter API as JSON. On checking the domain of the server, we find that it was akdls3re server. We get the Domain Name System (DNS) and structure of the network as shown in **Fig. 4** when searching it on Google.

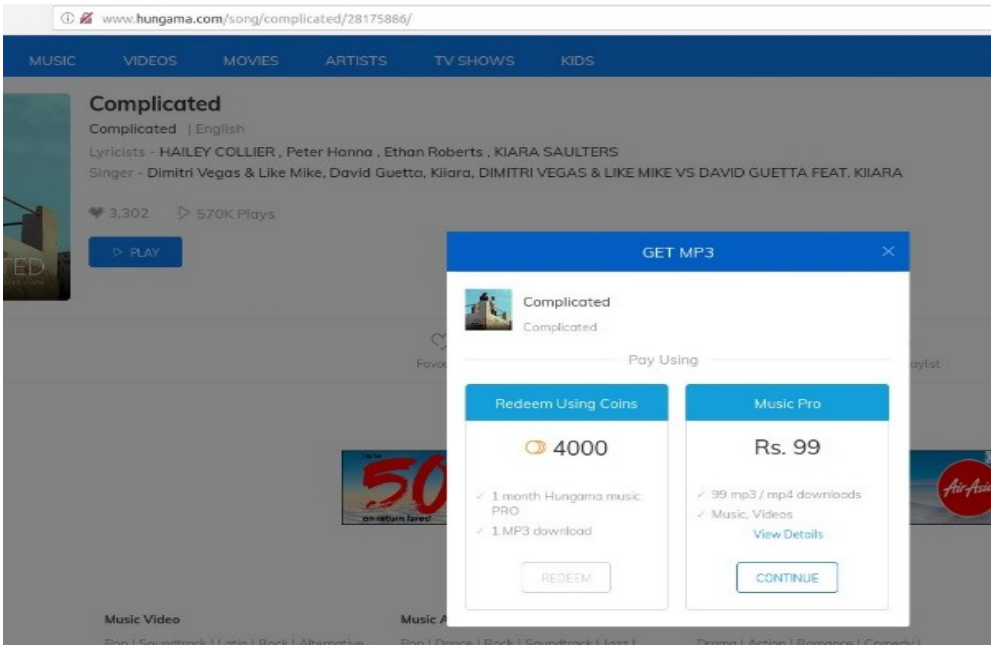

Fig. 1: A sample of paid e-songs.

JSON Raw Data Headers

Save Copy Pretty Print

{"uuid":"28175886","tracks":[{"uuid":"28175886","title":"Complicated","artwork":"https:\/\/content.hungama.com\/audio%20album\/display%20image
\/300x300%20jpeg\/2396802804.jpg","sources":[{"bitrate":"progressive","url":"http:\/\/akdls3re.hungama.com\/s3\/r\/ms2\/28175886\/4\/250
\/Complicated.mp3?__gda__=1529307730_dc7badd3d72e2ac9731c588e7d63bf9a"}],"cta_value":"http:\/\/www.hungama.com\/song\/complicated\/28175886","artist":
{"name":"Dimitri Vegas & Like Mike, David Guetta, Kiiara, DIMITRI VEGAS & LIKE MIKE VS DAVID GUETTA FEAT. KIIARA","handle":"Dimitri Vegas & Like Mike, David
Guetta, Kiiara, DIMITRI VEGAS & LIKE MIKE VS DAVID GUETTA FEAT. KIIARA"},"beacons":[{"key":"start","url":"https:\/\/secure.hungama.com\/newTwitter_Audio
\/audio\/beacons.php?track=28175886&eventType=audio&action=start&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},
{"key":"pause","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=audio&action=pause&anonymous_id=
{{user_id}}&page_name={{page}}&title=Complicated"},{"key":"resume","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio\/beacons.php?track=28175886&
eventType=audio&action=resume&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},{"key":"stop","url":"https:\/\/secure.hungama.com
\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=audio&action=stop&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},
{"key":"complete","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=audio&action=complete&anonymous_id=
{{user_id}}&page_name={{page}}&title=Complicated"},{"key":"ctaImpressionSignup","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio
\/beacons.php?track=28175886&eventType=impression&action=impression_download&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},
{"key":"ctaImpressionOpen","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=impression&
action=impression_open&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},{"key":"ctaClickSignup","url":"https:\/\/secure.hungama.com
\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=click&action=click_download&anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"},
{"key":"ctaClickOpen","url":"https:\/\/secure.hungama.com\/newTwitter_Audio\/audio\/beacons.php?track=28175886&eventType=click&action=click_open&
anonymous_id={{user_id}}&page_name={{page}}&title=Complicated"}]}]}

Fig. 2: The exposed JSON data for easy download.

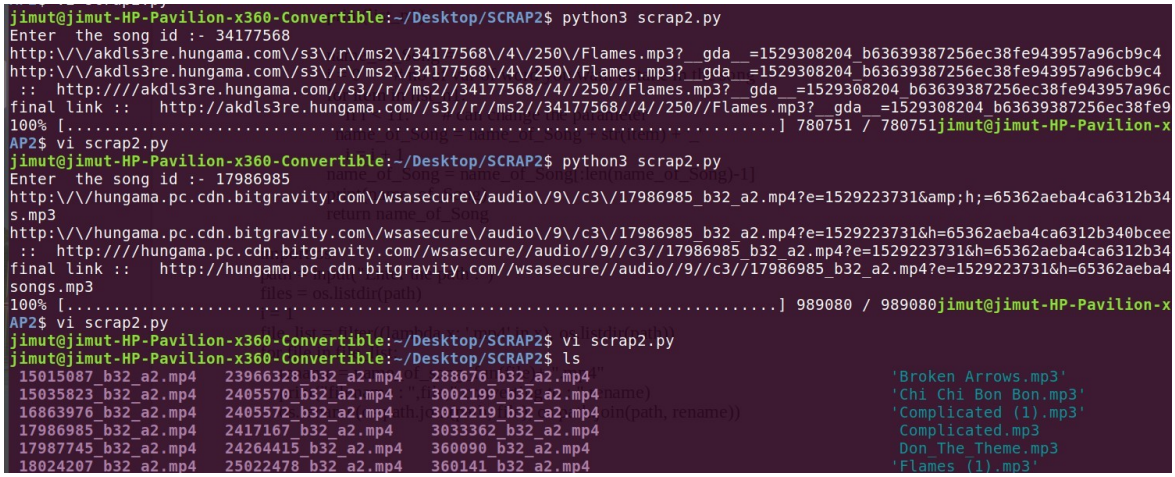

Fig. 3: The unformatted url links, being formatted by applying transformations.

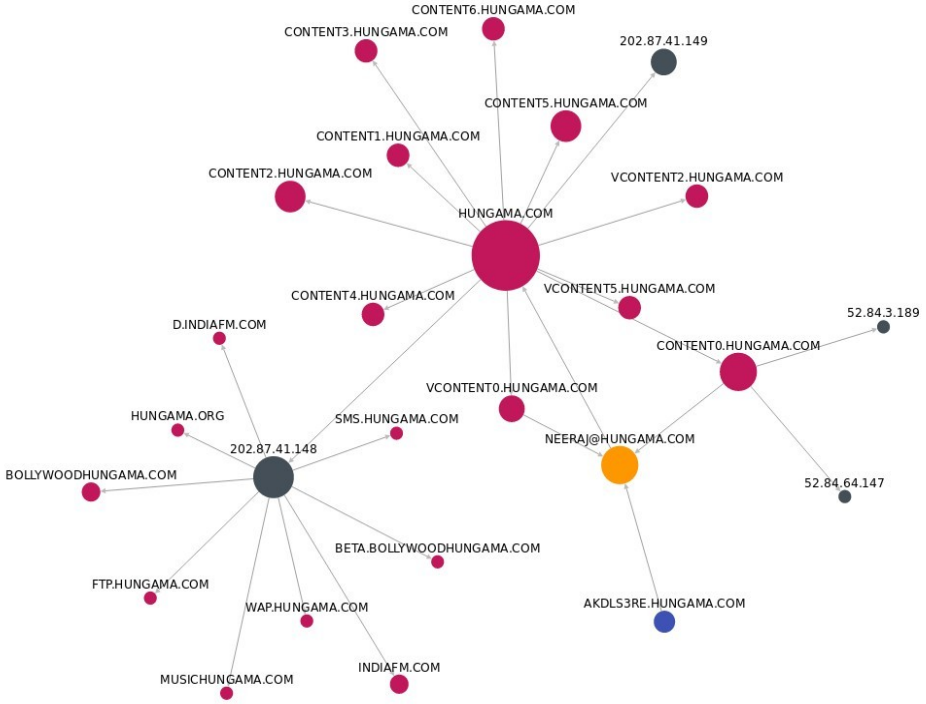

Fig. 4: DNS picture of hungama.com

The Python 3 script:

'''

To get the raw mp3 or mp4 songs from hungama.com
__author__ : Jimut Bahan Pal
__date__ : June 11 2018

'''

```
import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup
import ssl
```

```
import re

# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

id1 = input('Enter  the song id :- ')                    # The first url that is entered, required
for ignition
url = 'https://secure.hungama.com/newTwitter_Audio/audio/index-g.php?songid='+str(id1)
#print(url)
html = urllib.request.urlopen(url, context=ctx).read()          # html parser
soup = BeautifulSoup(html, 'html.parser')
#print(soup)
heap = str(soup)
heap1 = heap.split('\"')

import wget

try:
   link_needed = heap1[27]
except:
   print("Sorry! song cannot be downloaded!")
   exit(3)
#print(link_needed)
fin_link = str(link_needed)
#print(fin_link)
string_1 = fin_link.replace('&amp;h;','&h')
string_2 = string_1.replace('&amp;track;','&track')


string_3 = string_2.replace("\\","/")
#print(" :: ",string_2)
string_4 = string_3.replace("////","//")
#print("final link ::  ",string_4)

try:
   filename = wget.download(string_4)
except:
   print("Sorry! song cannot be downloaded!")
   exit(4)
```

It is further investigated that some mp4 e-songs have improper name instead they have ‘3002199_b32_a2.mp4’ which is <hash-pattern><.mp4> name. The e-songs can’t be known directly as those names are meaningless and named arbitrarily as shown in **Fig. 5**. We tried the ID3, tag-lib, file-metadata modules but it didn’t provide any fruitful result. These mp4 songs are binary encrypted. In addition, we used the io module in Python 3 and opened it in binary using open (fname, 'rb') and converted it into string and found a pattern.

The e-song name is about 300 characters before the ‘hungama.com’ Unicode character. Applying careful regex and slices in Python 3, we can extract the name as shown in **Fig. 6**. Subsequently, we modify the script so that it can use the os.listdir() module to get the path of the currently placed directory using pwd command and change every mp4 e-songs name to the name it have extracted. It’s a rough way of naming those e-songs which works good as depicted in **Fig. 7**.

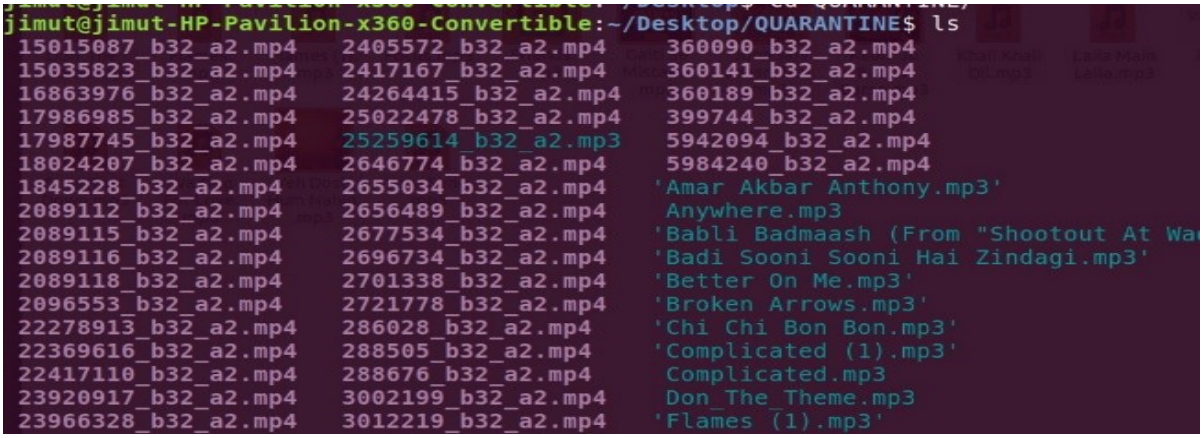

Fig. 5: Songs named arbitrarily

Fig. 6: Extracting the name from the binary encoded e-song.

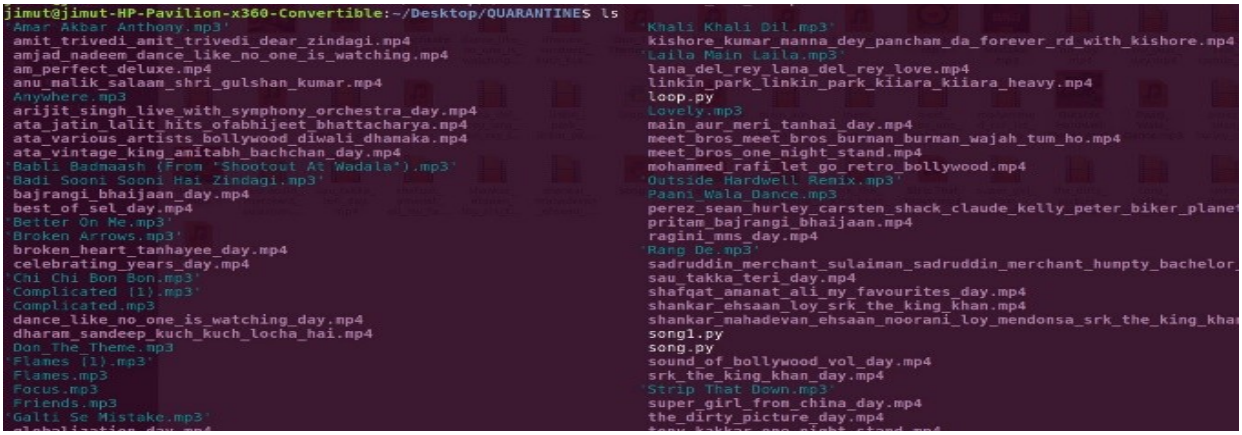

Fig. 7: Change of song names.

The Python 3 script:

```
'''
To change the name of mp3 songs which have hash-names
__author__ :: Jimut Bahan Pal
__date__ :: June 17 2018
'''
def name_of_song_fun(i):
        import re
        fname = i
        with open(fname, 'rb') as f:
            lines = [str(x.strip()) for x in f.readlines()]
        v=0
        for line in lines:
            tmp = line.strip().lower()
            if 'hungama.com' in tmp and v != 1:
                str_ = tmp
                v=1

        pos_f = str_.find('hungama.com')
        string_cut = str_[pos_f-300:pos_f]
        print(string_cut)
        p = re.compile('[a-zA-Z]+')
        list_n = p.findall(string_cut)

        print(list_n)

        list_n1 = []

        for list_1 in list_n:
            if len(list_1) > 1 and "lb" not in list_1 and "too" not in list_1 and "xa" not in
list_1 and "data" not in list_1 and "cmt" not in list_1 and "alb" not in list_1 and "lavf" not
in list_1:
                list_n1.append(list_1)

        print(list_n1)

        name_of_Song = ''
        i = 0       # no of var of word you want to take in the song
        for item in list_n1:
            if i < 11:       # can change the parameter
            name_of_Song = name_of_Song + str(item) + '_'
            i = i + 1
        name_of_Song = name_of_Song[:len(name_of_Song)-1]
        print(name_of_Song)
        return name_of_Song

import os
```

```
path = input("Enter the path :")
files = os.listdir(path)
i = 1
file_list = filter((lambda x: '.mp4' in x), os.listdir(path))
for file in file_list:
    rename = name_of_song_fun(file)+ ".mp4"
    print("filename : ",file," converting to : ",rename)
    os.rename(os.path.join(path, file), os.path.join(path, rename))
```

In this way, fraudster can store the e-songs that is in their database now in our local machine.

Conclusions

Python 3 technologies developed easy web scraping procedures. Some of them are Scrapy, Django and customized daemons. It is used for escalating the volume of information retrieved from web crawlers. We have also seen that using simple libraries like wget, urllib, ssl, beautiful soup 4 and doing a little bit of observation, to collect the sensitive information. We have also observed that about 70% of the paid e-songs available in hungama.com can be scraped for free by making simple programs in Python 3. If someone finds the JSON in the haystack, then everything is found. It is important to take measures to hide the static JSON. They should elevate the security measures else their business is at stake. Thus several e-commerce site, which can be scraped easily by the fraudster, for which they may suffer huge economic loss of their business.

References

1. Ashiwal, P., Tandan, S. R., Tripathi, P., and Miri, R. (2016). Web Information Retrieval Using Python 3 and BeautifulSoup. IJRASET. https://www.ijraset.com/fileserve.php?FID=4999, accessed on 14.06.2018.
2. Brewster, T. (2014). Snowden Used Basic Web Scraping Tools in NSA Breach. Silicon. https://www.silicon.co.uk/workspace/snowden-web-crawlers-nsa-insider-attack-138576?inf_by=5b2821ac671db8cc7e8b4f24, accessed on 18.06.2018.

3. DNS picture of hungama.com. (2018). https://www.threatcrowd.org/domain.php?domain=akdls3re.hungama.com, accessed on 18.06.2018.

4. Foster, R. (2014). When Programmers Scraped By. The New Yorker. https://www.newyorker.com/tech/elements/when-programmers-scrape-by, accessed on 20.06.2018

5. Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, K., Martinez-hernandez, V., Sanchez, V., and Perez-Meana, H. (2018). A Web Scraping Methodology for Bypassing Twitter API Restrictions. arXiv:1803.09875v1 [cs.IR]. https://arxiv.org/abs/1803.09875, accessed on 18.06.2018.

6. Pavan (2015). Why Common Measures Taken to Prevent Scraping Aren't Effective. Shield Square. https://www.shieldsquare.com/why-common-measures-taken-to-prevent-scraping-arent-effective/, accessed on 18.06.2018.

7. Ron, A., Shulman-Peleg, A., and Bronshtein, E. (2015). No SQL, No Injection? Examining NoSQL Security. arXiv:1506.04082v1 [cs.IR]. https://arxiv.org/abs/1506.04082, accessed on 15.06.2018.

8. Severance, C.R. (2013). Python for Everybody: Exploring Data Using Python 3. http://www.pythonlearn.com, accessed on 20.06.2018.

9. Severance, C.R. (2017). Python for Everybody. A Specialization course offered by University of Michigan through Coursera. https://www.coursera.org/specializations/python, accessed on 2017.

10. Wani, M. A., Agarwal, N., Jabin, S., and Hussain, S. Z. (2018). Design and Implementation of iMacros-based Data Crawler for Behavioral Analysis of Facebook Users. arXiv:1802.09566[cs.SI]. https://arxiv.org/abs/1802.09566, accessed on 14.06.2018.

11. Wikipedia. (Unknown). Web Scrapping. https://en.wikipedia.org/wiki/Web_scraping, accessed on 18.06.2018.