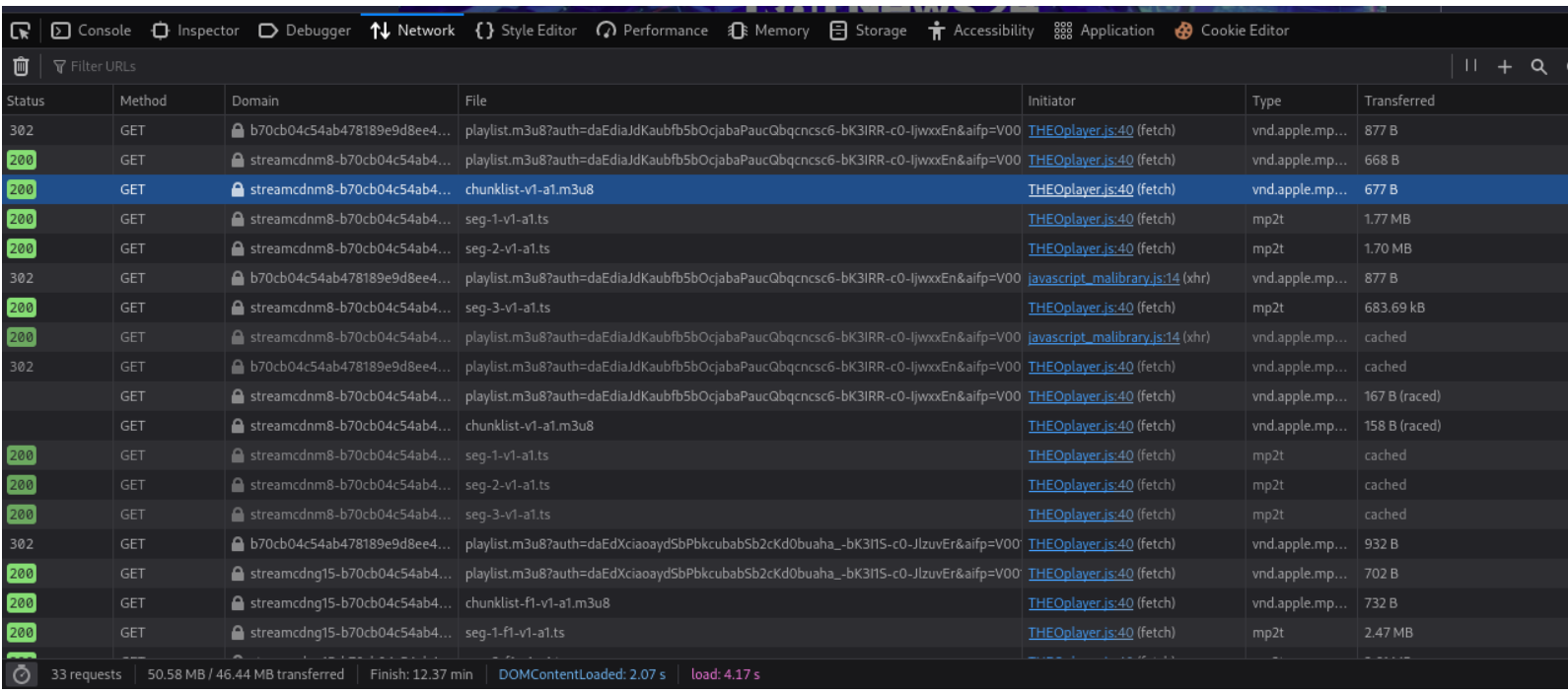
Rainews reverse-engineering guide

Tools I will be using:

Firefox developer edition
Insomnia HTTP client
Vscodium (Vscode without bloatware)



Status	Method	Domain	File	Initiator	Type	Transferred
302	GET	b70cb04c54ab478189e9d8ee4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	877 B
200	GET	streamcdnm8-b70cb04c54ab4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	668 B
200	GET	streamcdnm8-b70cb04c54ab4...	chunklist-v1-a1.m3u8	THEOplayer.js:40 (fetch)	vnd.apple.mp...	677 B
200	GET	streamcdnm8-b70cb04c54ab4...	seg-1-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	1.77 MB
200	GET	streamcdnm8-b70cb04c54ab4...	seg-2-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	1.70 MB
302	GET	b70cb04c54ab478189e9d8ee4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	javascript_malibrary.js:14 (xhr)	vnd.apple.mp...	877 B
200	GET	streamcdnm8-b70cb04c54ab4...	seg-3-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	683.69 kB
200	GET	streamcdnm8-b70cb04c54ab4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	javascript_malibrary.js:14 (xhr)	vnd.apple.mp...	cached
302	GET	b70cb04c54ab478189e9d8ee4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	cached
	GET	streamcdnm8-b70cb04c54ab4...	playlist.m3u8?auth=daEdiaJdKaubfb5bOcjabaPaucQbqcncsc6-bK3IRR-c0-IjwxxEn&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	167 B (raced)
	GET	streamcdnm8-b70cb04c54ab4...	chunklist-v1-a1.m3u8	THEOplayer.js:40 (fetch)	vnd.apple.mp...	158 B (raced)
200	GET	streamcdnm8-b70cb04c54ab4...	seg-1-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	cached
200	GET	streamcdnm8-b70cb04c54ab4...	seg-2-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	cached
200	GET	streamcdnm8-b70cb04c54ab4...	seg-3-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	cached
302	GET	b70cb04c54ab478189e9d8ee4...	playlist.m3u8?auth=daEdXciaoaydSbPbkcubabSb2cKd0buaha_-bK3I1S-c0-JlzuvEr&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	932 B
200	GET	streamcdng15-b70cb04c54ab4...	playlist.m3u8?auth=daEdXciaoaydSbPbkcubabSb2cKd0buaha_-bK3I1S-c0-JlzuvEr&aifp=V00	THEOplayer.js:40 (fetch)	vnd.apple.mp...	702 B
200	GET	streamcdng15-b70cb04c54ab4...	chunklist-f1-v1-a1.m3u8	THEOplayer.js:40 (fetch)	vnd.apple.mp...	732 B
200	GET	streamcdng15-b70cb04c54ab4...	seg-1-f1-v1-a1.ts	THEOplayer.js:40 (fetch)	mp2t	2.47 MB

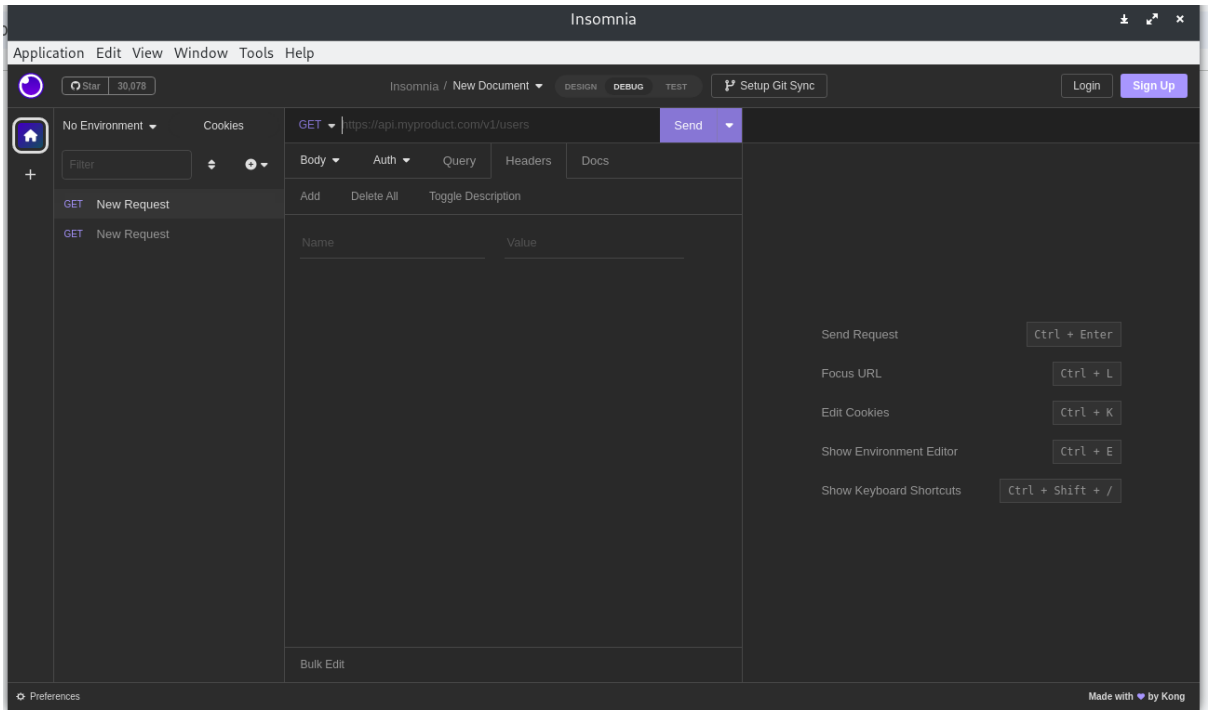
First I opened the developer tools on firefox and took a look at the network traffic when streaming a video off the website. One thing to note is most web applications that involve streaming implement a way to stream using chunks on the client side.

If you take a look at what is going on with the traffic you will notice that initially a request (highlighted request)is sent to return all the chunks (segments). After this the client requests each chunk individually. (seg-1-c1-a1.ts)

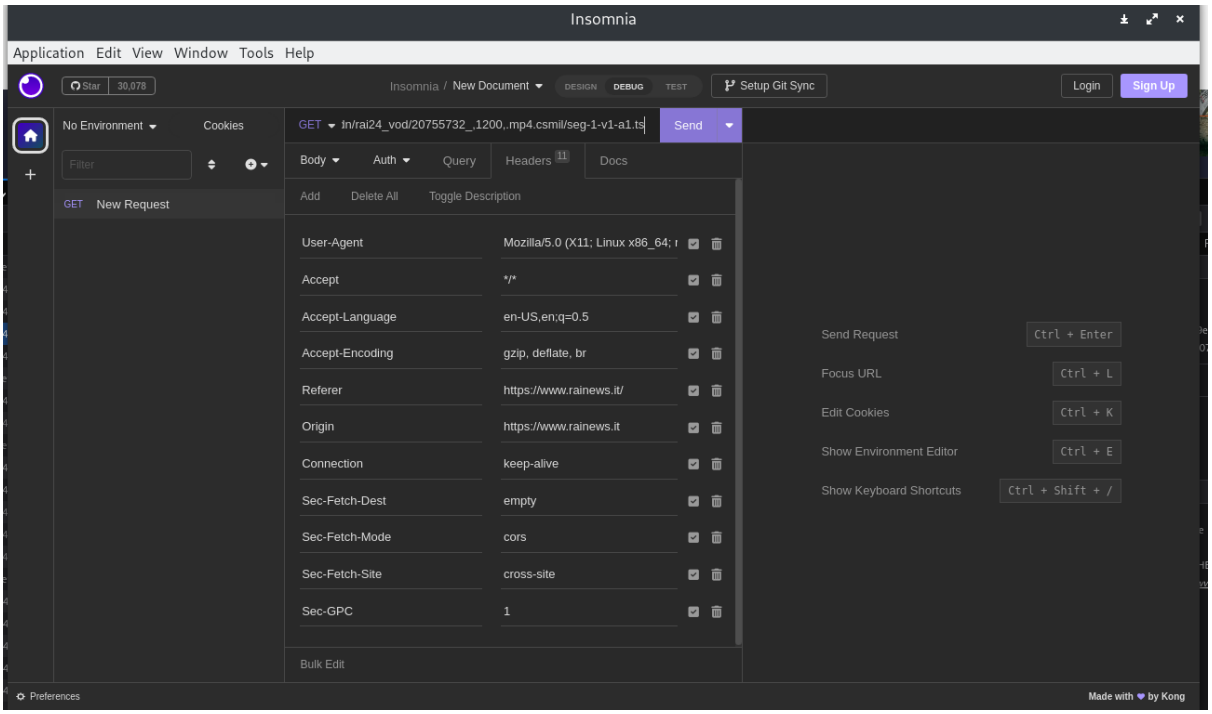
Let’s take a look at the chunks, we can right click on a chunk request and click copy url. If we paste this url on a new tab and it

starts to downloads a file this means there is no authentication for this file. And guess what! … It downloaded it! Good new for us.

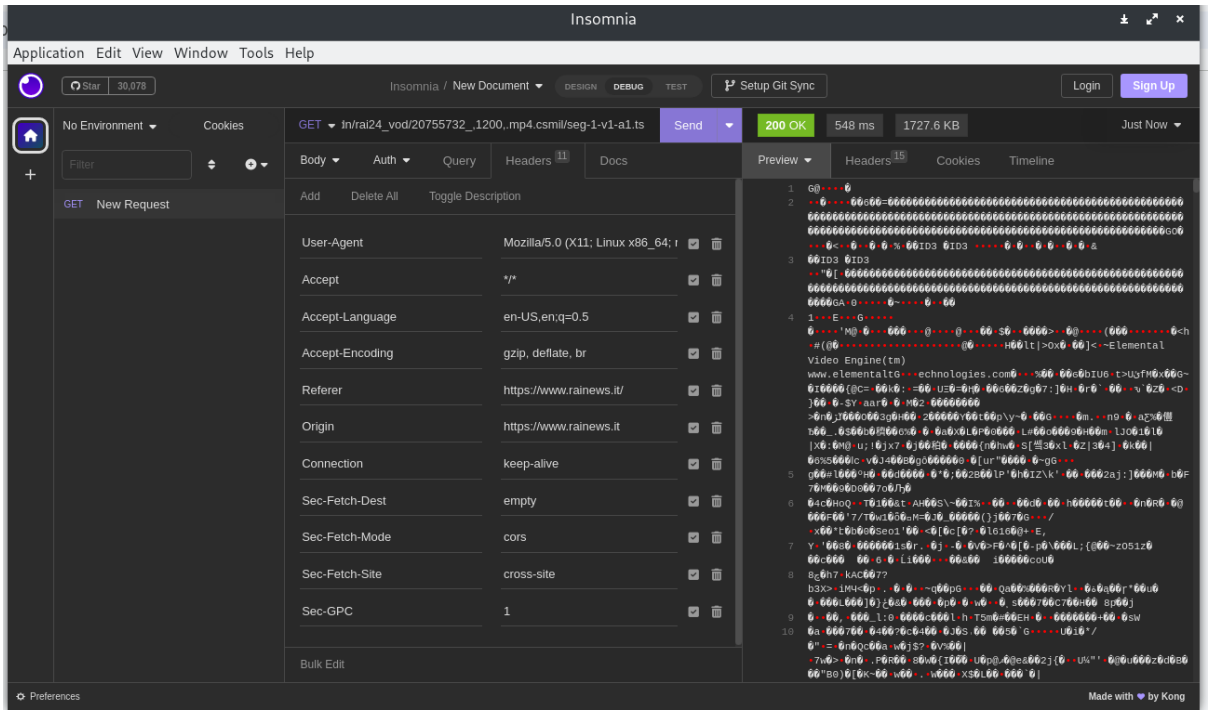
I am now going to show you a very, very important tool that I use. Insomnia is a free and open-source http client.
An important trick to know it to right click a request in the browser and then click copy as curl (or copy as curl-bash if you get that option)



Then paste into the insomnia url input. This will automatically import the whole request including headers, authentication and body.

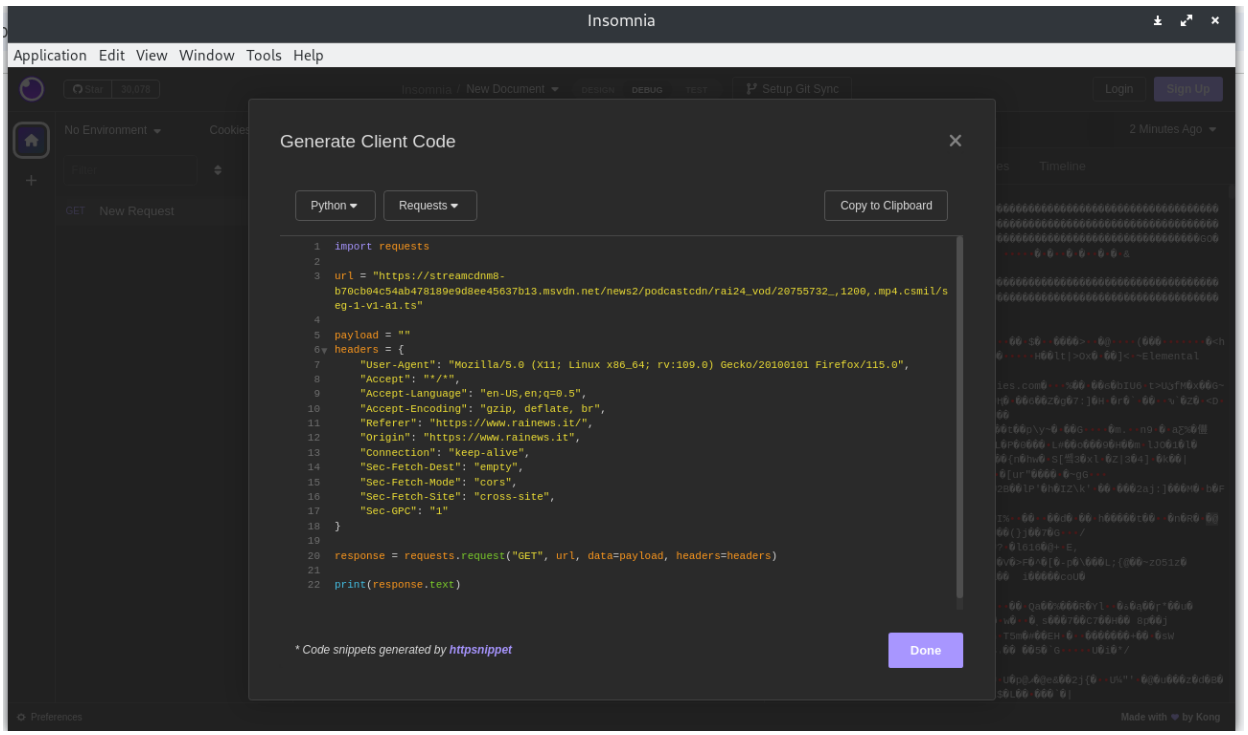


As you can see, the request is now in the insomnia client. We can now click the send button to see if the request is working properly in isolation.



We get a bunch of binary data which is exactly what we are looking in this example.

We can now do a sneaky trick to get it into python. We can hover over the ‘new request’ box, click the arrow and then click generate code.



Copy this to clipboard, and paste straight into python.
If we run this code, it will just print out a bunch of random binary. This is not what we want. We need to download it to a video file. We can do this by adding the stream=true parameter to the request and also adding a way to save the data to a file.

Below is the updated code.

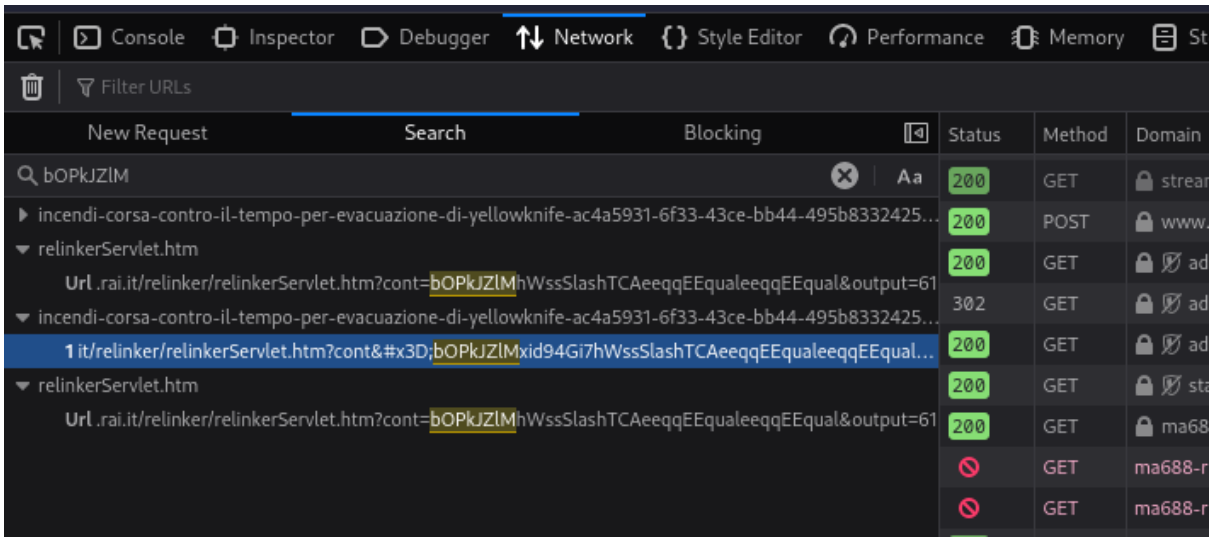
```
import requests

def download_ts_video(url, save_path):
    try:
```

```
        headers = {
            "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0",
            "Accept": "*/*",
            "Accept-Language": "en-US,en;q=0.5",
            "Accept-Encoding": "gzip, deflate, br",
            "Referer": "https://www.rainews.it/",
            "Origin": "https://www.rainews.it",
            "Connection": "keep-alive",
            "Sec-Fetch-Dest": "empty",
            "Sec-Fetch-Mode": "cors",
            "Sec-Fetch-Site": "cross-site",
            "Sec-GPC": "1"
        }
        response = requests.get(url, stream=True, headers=headers)
        response.raise_for_status()

        with open(save_path, 'wb') as file:
            for chunk in response.iter_content(chunk_size=8192):
                file.write(chunk)

        print(f"Download complete. Video saved to {save_path}")
    except requests.exceptions.RequestException as e:
        print(f"Error downloading the video: {e}")

url =
"https://streamcdnm8-b70cb04c54ab478189e9d8ee45637b13.msvdn.net/news2/podcas
tcdn/rai24_vod/20755732_,1200,.mp4.csmil/seg-1-v1-a1.ts"

save_path = "path_to_save_video.ts"

download_ts_video(url, save_path)
```

The next step is to see how the site generated this link. I looked through the traffic and found this url: https://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=bOPkJZlMxid94Gi7hWssSlashTCAeeqqEEqualeeqqEEqual&output=61

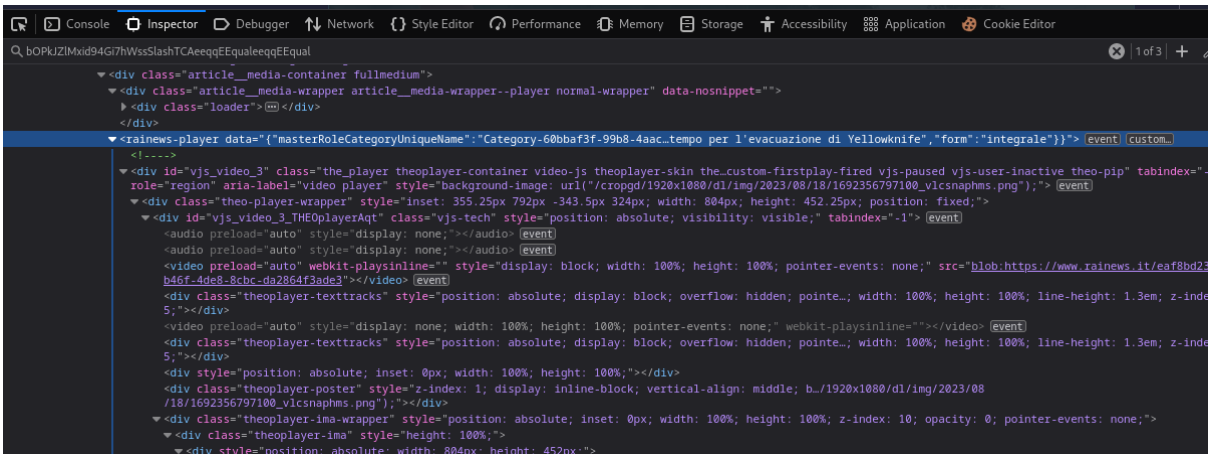
This URL returns some data, including the URL for the video data files. VERY GOOD NEWS!. However we have some bad news, we need to figure out how the site

generates an encoded string (the highlighted section of the URL) as this is very important.
A way of doing this is to copy the highlighted section (bOPkJZlMxid94Gi7hWssSlashTCAeeqqEEqualeeqqEEqua) and use it on the developer tools search function.



Great! We found a request that returns the encoded string we need (Highlighted request). Lets paste this into insomnia client and take a deeper look. This request seems to be the main request for the actual HTML. Click generate code and paste into python (We can delete some of the headers in the request as they are not necessary). We need to use the beautifulsoup library in python to parse this HTML data and return the encoded strin we need so we can download the video. First, to parse the data we need to know where it actually is in the HTML, we can do this by going to inspect

element and searching the encoded string.



We can see that the TAG is named ‘rainews-player’. We can use this in beautiful soup like this:

```
def get_video_url(url):
    querystring = {"nxtep":""}
    payload = ""
    headers = {
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0",
        "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,*/*;q=0.8",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate, br",
        "Referer": "https://www.rainews.it/",
        "Connection": "keep-alive",
        "Upgrade-Insecure-Requests": "1",
        "Sec-Fetch-Dest": "document",
        "Sec-Fetch-Mode": "navigate",
        "Sec-Fetch-Site": "same-origin",
        "Sec-Fetch-User": "?1",
        "Sec-GPC": "1"
    }
    response = requests.request("GET", url, data=payload,
headers=headers, params=querystring)
    soup = BeautifulSoup(response.text, 'html.parser')
    print(soup.find('rainews-player')['data'])
```

This prints out the JSON data containing the encoded string like this:

{"masterRoleCategoryUniqueName":"Category-60bbaf3f-99b8-4aac-8a6d-69a98e11bfc1","title":"Incendi, corsa contro il tempo per l'evacuazione di Yellowknife","image":"/cropgd/1920x1080/dl/img/2023/08/18/1692356797100_vlcsnaphms.png","adv":{"nobanner":false,"adv_label":null,"nofloorad":false,"tema1":"Ambiente","tema2":"Disastriambientali","tema3":"Incendi","isVideoEmbedded":true},"mediapolis":"http://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=bOPkJZlMxid94Gi7hWssSlashTCAeeqqEEqualeeqqEEqual","live":false,"audio":false,"stream_type":"","autoplay":true,"pip":true,"content_url":"http://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=bOPkJZlMxid94Gi7hWssSlashTCAeeqqEEqualeeqqEEqual","subtitles":"","dash":"","uhd":"","track_info":{"id":"ContentItem-ac4a5931-6f33-43ce-bb44-495b83324256","domain":"rainews","media_type":"vod","page_type":"video","editor":"rainews","section":"video","sub_section":null,"content":"pagina video","title":"Incendi, corsa contro il tempo per l'evacuazione di Yellowknife","channel":"rainews digital","create_date":"2023-08-18","date":"2023-08-18","typology":"informazione","genres":["ambiente|disastriambientali|incendi"],"sub_genres":["Incendio,Yellowknife,canada"],"program_title":"rainews","season":"2023","episode_title":"Incendi, corsa contro il tempo per l'evacuazione di Yellowknife","form":"integrale"}}

However we need to parse the data so it only prints out the encoded string. To do this we need to convert the datatype from a string to a JSON/dict object. We can do this with the JSON library like this:

```
    response = requests.request("GET", url, data=payload,
headers=headers, params=querystring)
    soup = BeautifulSoup(response.text, 'html.parser')
    data = soup.find('rainews-player')['data']
    data = json.loads(data)
    print(data['content_url'])
```

This will return this url:
http://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=bOPkJZlMxid94Gi7hWssSlashTCAeeqqEEqualeeqqEEqual
Which, as you can see, contains the encoded string we need.

Lets test to see if this works on a different video page:
https://www.rainews.it/video/2023/08/granchio-blu-goro-invasa-allevatori-di-vongole-in-crisi-db6ac663-a088-437c-968b-1c3a1a5b8b73.html

Great! It worked, we got this output:
http://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=x8wyzxgMYtssSlashWFvH8l7p1EAeeqqEEqualeeqqEEqual

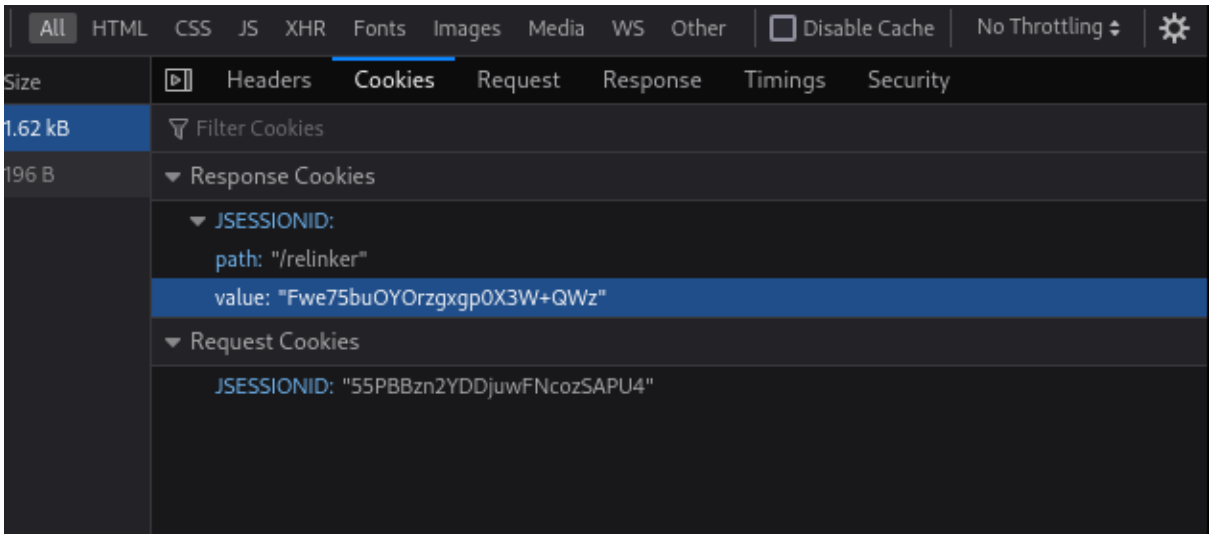
You can see the encoded string in this output is different to the other one!
Now we should make the script return the encoded string in the data url, not the content url

```
    data = data['content_url'].split('=')[1]
    data_url =
f'https://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont={data}
&output=61'
    return data_url
```

Lets make a request to this URL in python and see if we get the data we need.

```
└─$ /bin/python /home/larry/Documents/dev/fiverr/cosimopp/rainews/main.py
https://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont=x8wyzxgMYtssSlashWFvH8l7p1EAeeqqEEqualeeqqEEqual&outpu
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>

┌──(larry㉿1azza)-[~/…/dev/fiverr/cosimopp/rainews]
└─$
```

Oh no, access denied. This most likely means that we need a cookie to access this resource. We can go to the URL in the browser and inspect the traffic and look at the cookies section.



Looks like we need a lookie called JSESSIONID. Lets copy the cookie and add it straight to our request.
Hmm, it still doesn't work. Sometimes if we use http.client instead of requests it works. Let’s try it in a new file:

```
import http.client

conn = http.client.HTTPSConnection("mediapolisvod.rai.it")

payload = ""

headers = {
   'cookie': "JSESSIONID=S4Qx1U6kUZ3DAHzVi%2BVCd8V-",
   }

conn.request("GET",
"/relinker/relinkerServlet.htm?cont=x8wyzxgMYtssSlashWFvH8l7p1EAeeqqEEq
ualeeqqEEqual&output=61", payload, headers)

res = conn.getresponse()
data = res.read()

print(data.decode("utf-8"))
```

It worked!
Now let's add this back to our main script.
The data type that the URL returns is XML which is very similar to HTML. This means that we can use beautiful soup to parse it and return the URL with the auth string.

```
soup = BeautifulSoup(data.decode("utf-8"), 'html.parser')
print(soup.find('url').decode_contents()[10:][:-4])
```

Like this.
Now we need to write some code to download the video.
Here is the complete code:

```
import requests
from bs4 import BeautifulSoup
import json
import http.client
```

```
def get_data_url(url):
   querystring = {"nxtep":""}
   response = requests.request("GET", url, params=querystring)
   soup = BeautifulSoup(response.text, 'html.parser')
   data = soup.find('rainews-player')['data']
   data = json.loads(data)
   data = data['content_url'].split('=')[1]
   data_url =
f'https://mediapolisvod.rai.it/relinker/relinkerServlet.htm?cont={data}
&output=61'
   return data_url



def download_ts_video(url, save_path):
   try:
       headers = {
           "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0",
           "Accept": "*/*",
           "Accept-Language": "en-US,en;q=0.5",
           "Accept-Encoding": "gzip, deflate, br",
           "Referer": "https://www.rainews.it/",
           "Origin": "https://www.rainews.it",
           "Connection": "keep-alive",
           "Sec-Fetch-Dest": "empty",
           "Sec-Fetch-Mode": "cors",
           "Sec-Fetch-Site": "cross-site",
           "Sec-GPC": "1"
       }
       response = requests.get(url, stream=True, headers=headers)
       response.raise_for_status()

       with open(save_path, 'wb') as file:
           for chunk in response.iter_content(chunk_size=8192):
               file.write(chunk)

       print(f"Download complete. Video saved to {save_path}")
   except requests.exceptions.RequestException as e:
       print(f"Error downloading the video: {e}")
```

```
def download_mp4(url, save_path):
   response = requests.get(url, stream=True)
   if response.status_code == 200:
       with open(save_path, 'wb') as file:
           for chunk in response.iter_content(chunk_size=8192):
               file.write(chunk)
       print(f"Download complete. Saved as '{save_path}'")
   else:
       print(f"Failed to download. Status code:
{response.status_code}")




def get_download_url(url):
       conn = http.client.HTTPSConnection("mediapolisvod.rai.it")
       payload = ""
       headers = {
           'cookie': "JSESSIONID=S4Qx1U6kUZ3DAHzVi%2BVCd8V-",
           }
       conn.request("GET", data_url, payload, headers)
       res = conn.getresponse()
       data = res.read()
       soup = BeautifulSoup(data.decode("utf-8"), 'html.parser')

       download_url =soup.find('url').decode_contents()[10:][:-4]
       return download_url

url =
"https://www.rainews.it/video/2023/08/granchio-blu-goro-invasa-allevato
ri-di-vongole-in-crisi-db6ac663-a088-437c-968b-1c3a1a5b8b73.html"
data_url = get_data_url(url)
download_url = get_download_url(data_url)



save_file_path = "video.mp4"

download_mp4(download_url, save_file_path)
```