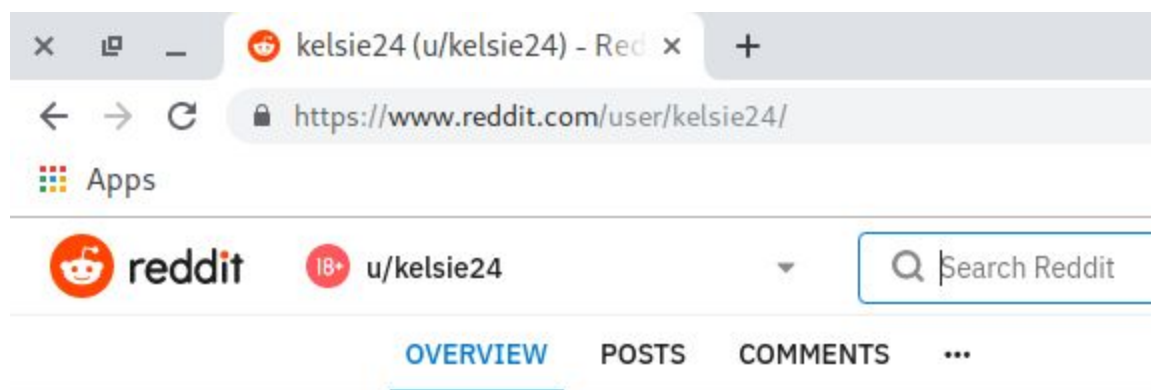


# Build An OSINT Username Search Tool Using SULTAN

In this article I will show you how to build your own custom OSINT username search tool using a python script that I call SULTAN.

Usernames are one of the major points of exploitation for any cyber investigation. Many websites utilize usernames as a way to identify individual accounts and users often keep the same, or similar, usernames across many of the websites and apps in which they hold accounts. This makes it very easy to link users across platforms. It comes as no surprise then that lots of OSINT websites offer ways to do username searches across multiple platforms. However, a major issue in developing such a site is that it is impossible to collect a list of all currently active platforms. Additionally, some of the websites might change their URL structure or how they display profiles. This will oftentimes “break” the username search tools and usually requires users to wait on the programmer to fix the broken queries.

Users might also want to run a custom list of websites that do not currently have username search tools to support them or only want to run a handful of niche sites for a more targeted search. All of these reasons are why I set out to build SULTAN (Sin’s Username Lookup Tool Ad Nauseam), a simple (bare-bones) Python script to run username searches on an expandable list of websites curated by the user or via crowdsourcing. SULTAN is designed to give the end-user the ability to create their own custom username search tool using only the platforms they want, without having to wait on the owner of a site or platform to make updates.



## Identifying Websites

Not all websites will work with SULTAN. Websites that include the username in their URL are the ones that we will be targeting (ex: <https://www.reddit.com/user/USERNAME/>) as we can use a script to easily swap out USERNAME with whatever username we are currently searching for.

Some websites prevent us from doing this by adding in a user id or other numerical identifier into the URL either with or without the username. In these instances we cannot run them through SULTAN as we cannot predict what the numerical id will be to rebuild the URL for scraping. An example of a website that does not work would be Meetup which utilizes user ids in place of a username (ex

<https://www.meetup.com/NAMEOFMEETUPGROUP/members/11401753/profile/>)

header	Social Media	header	n/a
<a href="https://graph.facebook.com/">https://graph.facebook.com/</a>		Some of the al	jeff
<a href="https://twitter.com/">https://twitter.com/</a>		Sorry, that pag	twitter
<a href="https://www.instagram.com/">https://www.instagram.com/</a>		The link you f	twitter
<a href="https://myspace.com/">https://myspace.com/</a>		Not Found	jess
<a href="https://www.youtube.com/">https://www.youtube.com/</a>		404handler	jess
<a href="https://">https://</a>	.tumblr.com/	Unless you we	test
<a href="https://">https://</a>	.skyrock.com/	Page non trou	redboolf
<a href="https://mix.com/">https://mix.com/</a>		Discover the	jess
<a href="https://weheartit.com/">https://weheartit.com/</a>		Looks like this	wrong9
<a href="https://vk.com/">https://vk.com/</a>		404 Not Foun	ivan
<a href="https://www.pinterest.com/">https://www.pinterest.com/</a>		Discover recip	jess
<a href="https://ello.co/">https://ello.co/</a>		If all else fail	jeff
<a href="https://keybase.io/">https://keybase.io/</a>		Have you trie	jeff
<a href="https://vimeo.com/">https://vimeo.com/</a>		or try searchi	jeff

## Setting up the Spreadsheet

To organize our websites for the Python script we need to note down four major variables for each website we wish to check against. For each website you wish to search usernames on please note down the following into a spreadsheet similar to the one above (or use the spreadsheet I provided in my Github below):

**UrIA:** This is the first part of the URL and will include everything up until, but not including, the username. Be mindful not to forget your slashes or any leading symbols.

**UrIB:** This is the second part of the URL and will include everything following, but not including, the username. Many websites will be blank in this section. Tumblr is one of the platforms that use a format in which the username comes before the rest of the URL rather than following it.

**Error:** This is the bit of text that appears on a website when you navigate to a profile (username) that does not exist. Some commonly use "404", others redirect to the main page and some have their own unique, sometimes witty, text. This is the variable you will be checking against when running the Python script. Do NOT rely on what appears in the browser for your text. I use a small requests script to return the HTML so I can search through it and make sure the error text is there and will be the same as if I use requests in SULTAN.

**Known Working:** This is where you will keep track of a known working username so you can run validation testing later. If you search for this username and SULTAN is unable to find the account listed here on the platform you will need to go back and look for other unique error text

and try again or see if there are other issues with why the script is not picking up the known working account. Some sites will not work as they load the error text dynamically, and some use multiple versions of error text for different situations making it impossible to pick only one.

```
# If line is not a header (regular entry)
else:

    #Send request to page
    try:
        r = requests.get(url)
    except:
        print("Could not connect to: " + url)

    #returns the HTML as a string
    html = r.text

    # read the data from the URL and check for error

    notfound = (html.find(error[count]))

    #If profile with username exists
    if notfound == -1:
        print url          # Prints url to profile
        found += 1         # Increases found count
```

## Inspecting the Python Code

I am not going to get too far into the weeds here however I feel it is important to explain the coding logic that was used for the tool for those that would like to know a bit more behind how it works but don't have a background in programming.

The first thing we need to do import the necessary modules for the code to function (requests and xlrd). You might need to install requests and xlrd before running SULTAN for the first time if you do not regularly use Python on your machine. Next, we will prompt the user for their username and save that input as a variable. Using xlrd we will read the columns of our Excel file and then assign the variables we set up in the spreadsheet above to arrays we can pull from later. Each of these sections will go into their own array so make sure all arrays have the same number of items (cells in the columns) in them otherwise you will have errors later on.

Next, we will set up a loop to take the username and put it in between the two parts of the URL arrays (URLA and URLB). This (URLA + username + URLB) will be our variable to pass to

requests as our URL to connect to. Once it connects to the website it will obtain the HTML and parse it looking for the error text that should be present when a profile does not exist. If the error text is found then the account should not exist and it moves on to the next URL. Otherwise, if the HTML is missing the error text, the account should exist and it will print the full URL out to the user before going to the next platform.

## **Conclusion**

SULTAN remains a work in progress and is far from perfect; as some sites do like to throw it on its head from time to time. That being said the logic and code provided should be more than enough to get you up and running on some of the major sites. You can download the code and my sample Excel file (which out of the box searches ~100 platforms) to get started over on my Github [page](#). If you would like to contribute any websites or offer any suggestions, please feel free to reach out to me on Twitter.