

Forum:[Hack Forums](#)

Thread: [{Tutorial} Basic OSINT - Sherlock - Username Hunting](#)

Pr0m3th3us - October 13, 2026, 5:57 pm

Username Hunting: Tracking Digital Footprints with Sherlock

Part 1: Introduction & The Mechanics of Sherlock

Today, we are looking at a highly specialized, surgical tool: Sherlock. Sherlock does exactly one thing, but it does it better than almost anything else: Username Enumeration.

We as humans are creatures of habit. If someone has a favorite handle for gaming, they likely use it for Reddit, Spotify, GitHub, and random forums. Sherlock takes a single username and queries over 400 social networks and websites simultaneously to see where that identity exists. To show you how dangerous username reuse can be, I will be demonstrating in this TUT.

How Sherlock Works (No APIs Required)

Unlike SpiderFoot, which relies heavily on API keys, Sherlock is beautifully simple. It doesn't need passwords or APIs. It simply takes the username you give it and rapidly constructs profile URLs for hundreds of sites (e.g., `twitter.com/username`, `reddit.com/user/username`). It then checks the HTTP response codes from those servers. If a page exists, it flags it as a hit! Because it only looks at public-facing URLs, it is incredibly fast and completely legal.

Installation on Linux

Most Linux's usually has this in its repo, making installation a breeze.

Code `sudo apt update && sudo apt install sherlock`

If you prefer to grab the absolute latest version straight from the source, you can clone the GitHub repo instead:

Code `git clone https://github.com/sherlock-project/sherlock.git`

`cd sherlock`

`python3 -m pip install -r requirements.txt`

The Hunt & Analyzing the Data

Phase 1: Running the Scan

In my terminal, I run the following simple, basic command

Code `sherlock --csv --timeout 5 ALIAS`

Command Breakdown: `--csv`: This tells Sherlock to export all the positive hits into a clean spreadsheet file, which is great for reporting.

--timeout 5: This tells the tool to give up on a website if it doesn't respond in 5 seconds. This speeds up the scan significantly, preventing the tool from hanging on dead servers.

ALIAS: The target username.

Advanced Command-Line Flags

While the basic command gets the job done, Sherlock has a few powerful flags you can append to your command to customize your hunt.

1. Filtering by Site (--site)

If you don't want to wait for Sherlock to query over 400 sites and only care about specific platforms, you can restrict the search. You can stack this flag multiple times.

Code `sherlock --site twitter --site reddit --site github ALIAS`

2. OPSEC: Routing Through Tor (--tor)

If you are investigating a sensitive target and need to mask your IP address, you can route Sherlock's traffic through the Tor network. (You must have Tor installed and running for this to work).

Code `sherlock --tor ALIAS` Pro-Tip: Use the `--unique-tor` flag instead if you want Sherlock to request a brand new Tor circuit for every single website it checks. This makes you virtually untrackable, though it does significantly slow down the scan.

3. Decluttering the Terminal (--print-found)

By default, Sherlock prints every single site it checks, which results in a massive wall of red "Not Found!" text. If you only want your terminal to display the successful hits (the green pluses), append this flag:

Code `sherlock --print-found ALIAS`

4. NSFW Targets (--nsfw)

Sherlock maintains a list of Not Safe For Work (adult) websites in its database, but it excludes them from the default scan to keep things clean. If your investigation requires checking these platforms, you must explicitly enable them

Code `sherlock --nsfw ALIAS`

5. Instant Review (--browse)

If you are doing a quick manual review and want to instantly look at the profiles Sherlock finds, use the browse flag. It will automatically open a new tab in your default web browser for every successful hit it gets.

Code `sherlock --browse ALIAS` Warning: If Sherlock finds 50 accounts, it will launch 50 browser tabs simultaneously! LOL! Use this flag carefully.

Phase 2: The Live Output

Sherlock uses a simple color-coded system (Green for hits, Red for misses).

Quote:[*]Checking username: ALIAS

[+] GitHub: <https://github.com/ALIAS>

[+] Reddit: <https://www.reddit.com/user/ALIAS>

[+] Spotify: <https://open.spotify.com/user/ALIAS>

[-] Facebook: Not Found!

[+] HackerNews: <https://news.ycombinator.com/user?id=ALIAS>

[+] Pinterest: <https://www.pinterest.com/ALIAS>

[*]

Phase 3: The Analysis

Within two minutes, Sherlock checked over 400 sites and found 52 accounts tied to my old gaming alias. Here is why this is an OSINT goldmine

The Spotify Pivot: People forget that Spotify is a social network. By clicking the Spotify link Sherlock found, an attacker could see the public playlists. Those playlists were created by their real, legal name (which Spotify linked to the account), instantly unmasking the anonymous alias.

The Reddit History: The Reddit account link provided years of comment history. OSINT investigators can feed a Reddit history into linguistic analysis tools to determine a target's time zone, political leanings, hobbies, and even the city they live in based on the local subreddits they frequent.

The GitHub Connection: The GitHub profile listed an old personal email address [EMAIL]@gmail.com.

Conclusion

Notice what just happened in that last bullet point? Sherlock found an email address.

This is the core of OSINT. You take the output of one tool and feed it into another. I can now take that newly discovered Gmail address, plug it back into Maltego or SpiderFoot or a litany of other programs and go from there.

A single reused username is all it takes to unravel someone's entire digital life.