Semester Project – Blackbird

CSCI – 4321 -Computer Security

Abigail Rodriguez Vazquez, Taja Hicks, Luis Morales

September 24,2025

Report on Blackbird: Introducing Code and functions of Blackbird

Abstract

This paper presents the goal, use cases, and results of the OSINT tool, Blackbird. Blackbird is a tool with key features including free AI-powered insights, metadata extraction, WhatsMyName integration, export options and customizable filters. Practical applications of Blackbird are digital investigations, social media research, compliance and verification, and data collection and analysis. This paper goes over the goal and purpose of Blackbird, why someone would want to use this tool, installation, and examples of using the tool.

Introduction to Blackbird

Blackbird is an OSINT tool that combines username and email searches across more than 600 platforms (Antoniaci, 2020). This is done using AI to profile. By utilizing community-driven projects (like WhatsMyName), it lowers false positive rates and provides high-quality results. Some main Features include smart filters, polished PDF/CSV exports, and more (Antoniaci, 2020).

Goal and Purpose of Blackbird

The goal of blackbird is to use AI to analyze the sites where a username or email is found and returns a behavioral and technical profile of the user. This Helps you understand more about someone whether it be what language their learning, what platforms they participate in. As well as some of Risk they might encounter, all with less effort (Antoniaci, 2020).

Why would you want to use Blackbird When it comes to Computer Security?

- Detecting Possible Data Breaches: You can use Blackbird to better locate the websites or apps you are associated with and help you see if you might have been impacted by a breach.

- This can also help you ensure that you use different passwords for different accounts on different platforms.

- Becoming more Aware/Mindful: Since Blackbird returns a behavioral and technical profile of the user, we can become more aware of our online presence.

  - This awareness can help you better understand some of the vulnerabilities you may encounter and mitigate them.

  - In addition to being more mindful of our computer security and implementing more secure login systems, such as two-factor authentication.

Getting Started: [Colab Link here]

First we clone the blackbird program from github.

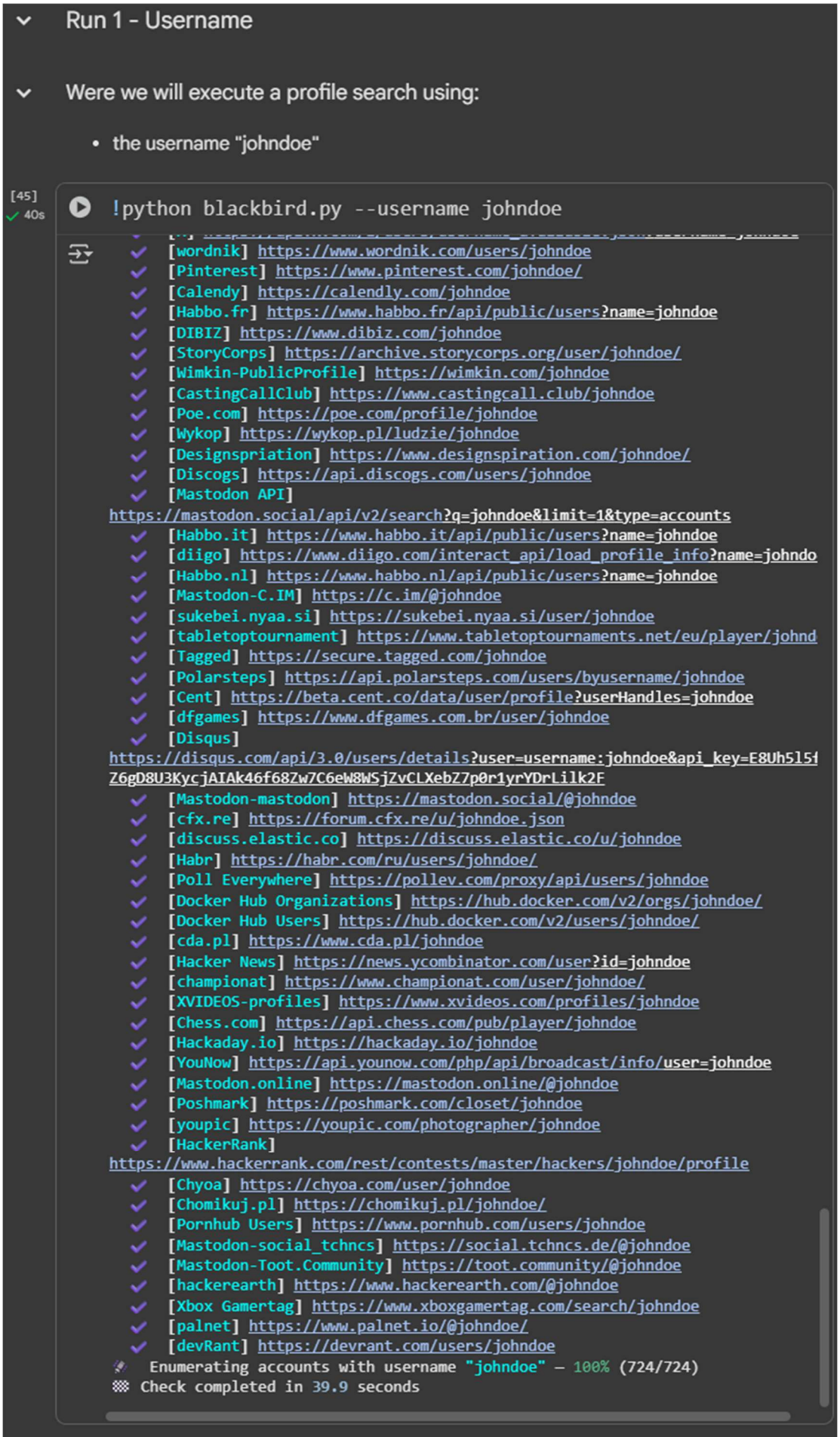
```
!git clone https://github.com/p1ngul1n0/blackbird

Cloning into 'blackbird'...
remote: Enumerating objects: 2890, done.
remote: Counting objects: 100% (580/580), done.
remote: Compressing objects: 100% (173/173), done.
remote: Total 2890 (delta 524), reused 407 (delta 407), pack-reused 2310 (from 3)
Receiving objects: 100% (2890/2890), 14.34 MiB | 21.13 MiB/s, done.
Resolving deltas: 100% (1638/1638), done.
```

Once the program files are installed, we change the working directory and install the requirements into the blackbird directory.

```
%cd blackbird
%ls

[Errno 2] No such file or directory: 'blackbird'
/content/blackbird
assets/       data/       docs/      requirements.txt  tests/
blackbird.py  Dockerfile  README.md  src/
```

The requirements will be installed and may need to be reset to update the changes.

```
!pip install -r requirements.txt

Requirement already satisfied: aiohappyeyeballs==2.6.1 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 1)) (2.6.1)
Requirement already satisfied: aiohttp==3.12.13 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 2)) (3.12.13)
Requirement already satisfied: aiosignal==1.3.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 3)) (1.3.2)
Requirement already satisfied: attrs==25.3.0 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 4)) (25.3.0)
Requirement already satisfied: certifi==2025.6.15 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 5)) (2025.6.15)
Requirement already satisfied: chardet==5.2.0 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 6)) (5.2.0)
Requirement already satisfied: charset-normalizer==3.4.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 7)) (3.4.
Requirement already satisfied: frozenlist==1.7.0 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 8)) (1.7.0)
Requirement already satisfied: idna==3.10 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 9)) (3.10)
Requirement already satisfied: markdown-it-py==3.0.0 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 10)) (3.0.0)
Requirement already satisfied: mdurl==0.1.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 11)) (0.1.2)
Requirement already satisfied: multidict==6.6.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 12)) (6.6.2)
Requirement already satisfied: pillow==11.3.0 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 13)) (11.3.0)
Requirement already satisfied: propcache==0.3.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 14)) (0.3.2)
Requirement already satisfied: Pygments==2.19.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 15)) (2.19.2)
Requirement already satisfied: python-dotenv==1.1.1 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 16)) (1.1.1)
Requirement already satisfied: reportlab==4.4.2 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 17)) (4.4.2)
Requirement already satisfied: requests==2.32.4 in /usr/local/lib/python3.12/dist-packages (from -r requirements.txt (line 18)) (2.32.4)
```


Run 1 - Username
Were we will execute a profile search using:
the username "johndoe"
[45]
40s
!python blackbird.py --username johndoe
[wordnik] https://www.wordnik.com/users/johndoe
[Pinterest] https://www.pinterest.com/johndoe/
[Calendly] https://calendly.com/johndoe
[Habbo.fr] https://www.habbo.fr/api/public/users?name=johndoe
[DIBIZ] https://www.dibiz.com/johndoe
[StoryCorps] https://archive.storycorps.org/user/johndoe/
[Wimkin-PublicProfile] https://wimkin.com/johndoe
[CastingCallClub] https://www.castingcall.club/johndoe
[Poe.com] https://poe.com/profile/johndoe
[Wykop] https://wykop.pl/ludzie/johndoe
[Designspriation] https://www.designspiration.com/johndoe/
[Discogs] https://api.discogs.com/users/johndoe
[Mastodon API]
https://mastodon.social/api/v2/search?q=johndoe&limit=1&type=accounts
[Habbo.it] https://www.habbo.it/api/public/users?name=johndoe
[diigo] https://www.diigo.com/interact_api/load_profile_info?name=johndo
[Habbo.nl] https://www.habbo.nl/api/public/users?name=johndoe
[Mastodon-C.IM] https://c.im/@johndoe
[sukebei.nyaa.si] https://sukebei.nyaa.si/user/johndoe
[tabletoptournament] https://www.tabletoptournaments.net/eu/player/johnd
[Tagged] https://secure.tagged.com/johndoe
[Polarsteps] https://api.polarsteps.com/users/byusername/johndoe
[Cent] https://beta.cent.co/data/user/profile?userHandles=johndoe
[dfgames] https://www.dfgames.com.br/user/johndoe
[Disqus]
https://disqus.com/api/3.0/users/details?user=username:johndoe&api_key=E8Uh5l5
Z6gD8U3KycjAIAk46f68Zw7C6eW8WSjZvCLXebZ7p0r1yrYDrLilk2F
[Mastodon-mastodon] https://mastodon.social/@johndoe
[cfx.re] https://forum.cfx.re/u/johndoe.json
[discuss.elastic.co] https://discuss.elastic.co/u/johndoe
[Habr] https://habr.com/ru/users/johndoe/
[Poll Everywhere] https://pollev.com/proxy/api/users/johndoe
[Docker Hub Organizations] https://hub.docker.com/v2/orgs/johndoe/
[Docker Hub Users] https://hub.docker.com/v2/users/johndoe/
[cda.pl] https://www.cda.pl/johndoe
[Hacker News] https://news.ycombinator.com/user?id=johndoe
[championat] https://www.championat.com/user/johndoe/
[XVIDEOS-profiles] https://www.xvideos.com/profiles/johndoe
[Chess.com] https://api.chess.com/pub/player/johndoe
[Hackaday.io] https://hackaday.io/johndoe
[YouNow] https://api.younow.com/php/api/broadcast/info/user=johndoe
[Mastodon.online] https://mastodon.online/@johndoe
[Poshmark] https://poshmark.com/closet/johndoe
[youpic] https://youpic.com/photographer/johndoe
[HackerRank]
https://www.hackerrank.com/rest/contests/master/hackers/johndoe/profile
[Chyoa] https://chyoa.com/user/johndoe
[Chomikuj.pl] https://chomikuj.pl/johndoe/
[Pornhub Users] https://www.pornhub.com/users/johndoe
[Mastodon-social_tchncs] https://social.tchncs.de/@johndoe
[Mastodon-Toot.Community] https://toot.community/@johndoe
[hackerearth] https://www.hackerearth.com/@johndoe
[Xbox Gamertag] https://www.xboxgamertag.com/search/johndoe
[palnet] https://www.palnet.io/@johndoe/
[devRant] https://devrant.com/users/johndoe
Enumerating accounts with username "johndoe" — 100% (724/724)
Check completed in 39.9 seconds

Run 2 - E-mail

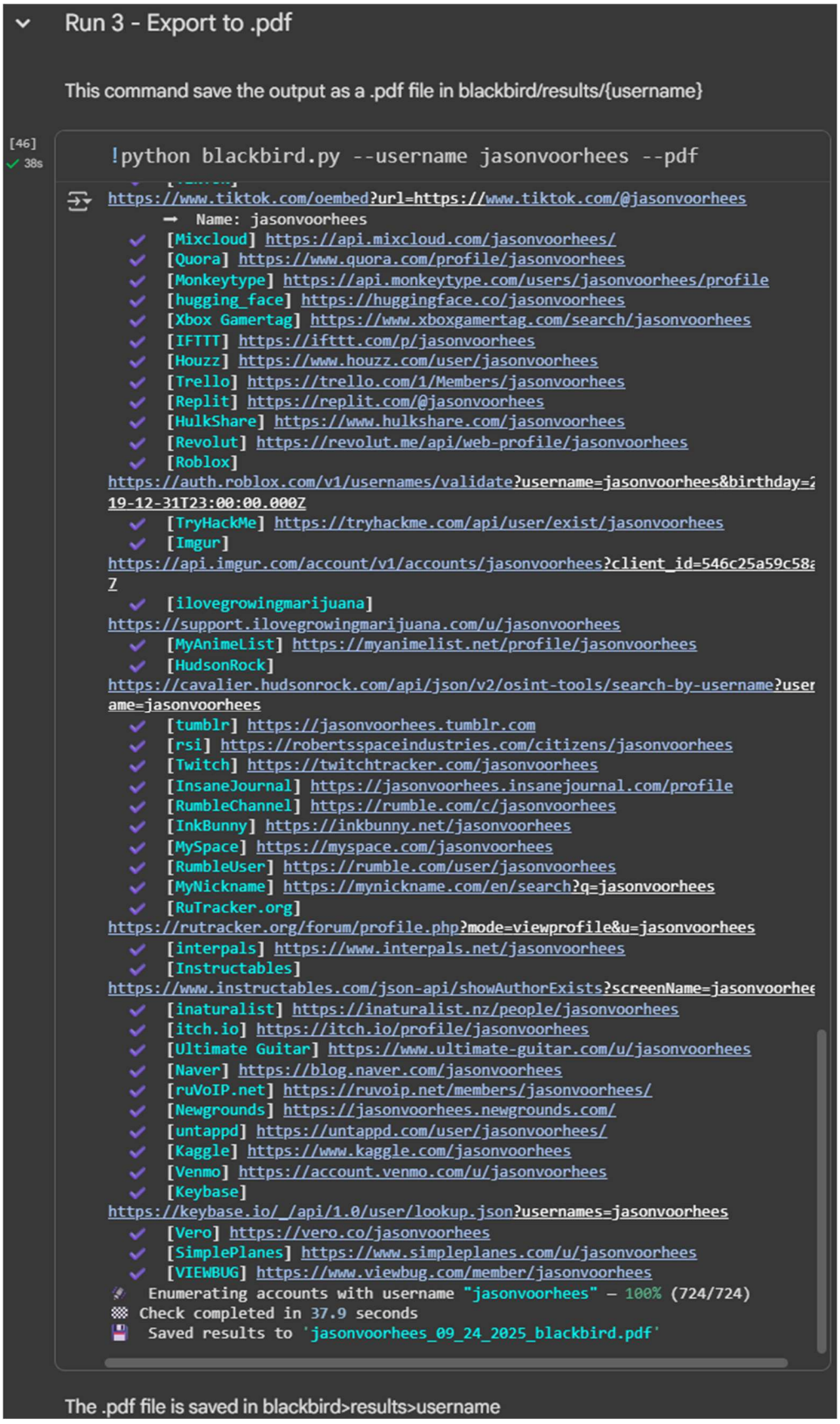
Here we will execute a profile search using:

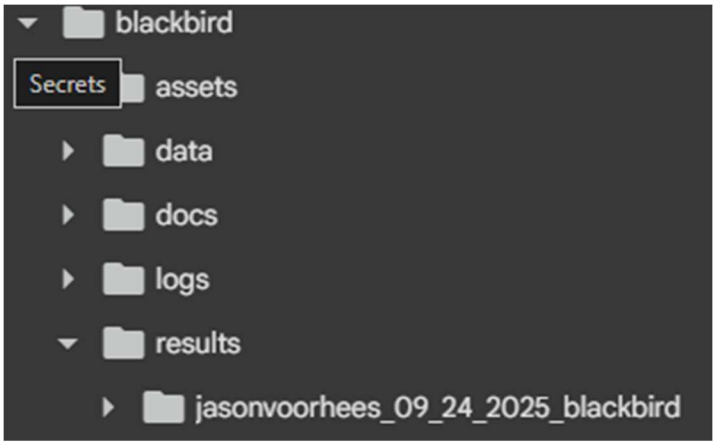
- the e-mail address johndoe@example.com

```
[ ]  !python blackbird.py --email johndoe@example.com
```

```
            Use --filter to limit search scope. | by Lucas Antoniaci
 Checking for updates...
  Sites List is up to date
   [Twitter]
https://api.twitter.com/i/users/email_available.json?email=johndoe@example.com
   [Eventbrite] https://www.eventbrite.com/api/v3/users/lookup/
      → User ID: 63904742633
   [Duolingo]
https://www.duolingo.com/2017-06-30/users?email=johndoe@example.com
   [Xvideos]
https://www.xvideos.com/account/checkemail?email=johndoe@example.com
   [Chess.com]
https://www.chess.com/callback/email/available?email=johndoe@example.com
   [Picsart]
https://api.picsart.com/users/email/existence?email_encoded=0&emails=johndoe@exa
mple.com
   Enumerating accounts with email "johndoe@example.com" — 100% (16/16)
 Check completed in 0.9 seconds (16 sites)
```

The results of this search are less due to the use of only 16 online platforms.


Run 3 - Export to .pdf
This command save the output as a .pdf file in blackbird/results/{username}
[46]
38s
!python blackbird.py --username jasonvoorhees --pdf
https://www.tiktok.com/oembed?url=https://www.tiktok.com/@jasonvoorhees
→ Name: jasonvoorhees
[Mixcloud] https://api.mixcloud.com/jasonvoorhees/
[Quora] https://www.quora.com/profile/jasonvoorhees
[Monkeytype] https://api.monkeytype.com/users/jasonvoorhees/profile
[hugging_face] https://huggingface.co/jasonvoorhees
[Xbox Gamertag] https://www.xboxgamertag.com/search/jasonvoorhees
[IFTTT] https://ifttt.com/p/jasonvoorhees
[Houzz] https://www.houzz.com/user/jasonvoorhees
[Trello] https://trello.com/1/Members/jasonvoorhees
[Replit] https://replit.com/@jasonvoorhees
[HulkShare] https://www.hulkshare.com/jasonvoorhees
[Revolut] https://revolut.me/api/web-profile/jasonvoorhees
[Roblox]
https://auth.roblox.com/v1/usernames/validate?username=jasonvoorhees&birthday=2
19-12-31T23:00:00.000Z
[TryHackMe] https://tryhackme.com/api/user/exist/jasonvoorhees
[Imgur]
https://api.imgur.com/account/v1/accounts/jasonvoorhees?client_id=546c25a59c58
7
[ilovegrowingmarijuana]
https://support.ilovegrowingmarijuana.com/u/jasonvoorhees
[MyAnimeList] https://myanimelist.net/profile/jasonvoorhees
[HudsonRock]
https://cavalier.hudsonrock.com/api/json/v2/osint-tools/search-by-username?user
ame=jasonvoorhees
[tumblr] https://jasonvoorhees.tumblr.com
[rsi] https://robertsspaceindustries.com/citizens/jasonvoorhees
[Twitch] https://twitchtracker.com/jasonvoorhees
[InsaneJournal] https://jasonvoorhees.insanejournal.com/profile
[RumbleChannel] https://rumble.com/c/jasonvoorhees
[InkBunny] https://inkbunny.net/jasonvoorhees
[MySpace] https://myspace.com/jasonvoorhees
[RumbleUser] https://rumble.com/user/jasonvoorhees
[MyNickname] https://mynickname.com/en/search?q=jasonvoorhees
[RuTracker.org]
https://rutracker.org/forum/profile.php?mode=viewprofile&u=jasonvoorhees
[interpals] https://www.interpals.net/jasonvoorhees
[Instructables]
https://www.instructables.com/json-api/showAuthorExists?screenName=jasonvoorhee
[inaturalist] https://inaturalist.nz/people/jasonvoorhees
[itch.io] https://itch.io/profile/jasonvoorhees
[Ultimate Guitar] https://www.ultimate-guitar.com/u/jasonvoorhees
[Naver] https://blog.naver.com/jasonvoorhees
[ruVoIP.net] https://ruvoip.net/members/jasonvoorhees/
[Newgrounds] https://jasonvoorhees.newgrounds.com/
[untappd] https://untappd.com/user/jasonvoorhees/
[Kaggle] https://www.kaggle.com/jasonvoorhees
[Venmo] https://account.venmo.com/u/jasonvoorhees
[Keybase]
https://keybase.io/_/api/1.0/user/lookup.json?usernames=jasonvoorhees
[Vero] https://vero.co/jasonvoorhees
[SimplePlanes] https://www.simpleplanes.com/u/jasonvoorhees
[VIEWBUG] https://www.viewbug.com/member/jasonvoorhees
Enumerating accounts with username "jasonvoorhees" — 100% (724/724)
Check completed in 37.9 seconds
Saved results to 'jasonvoorhees_09_24_2025_blackbird.pdf'
The .pdf file is saved in blackbird>results>username

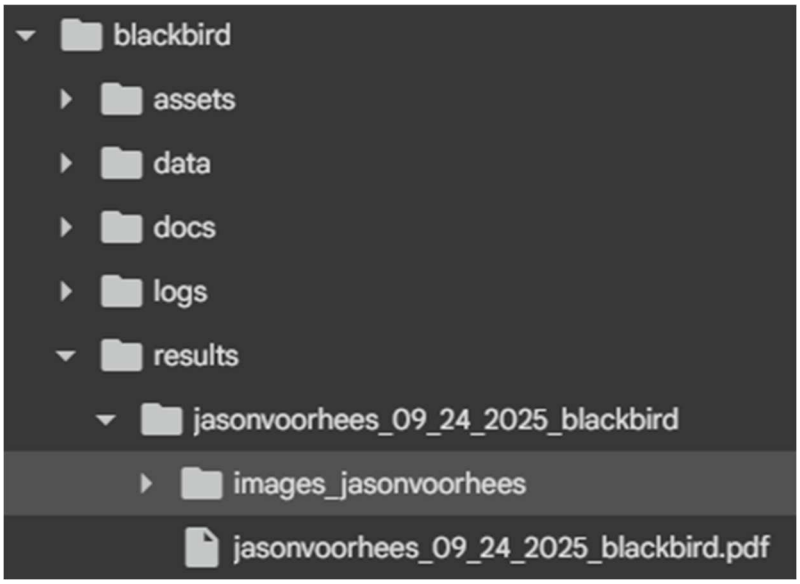

blackbird
Secrets
assets
data
docs
logs
results
jasonvoorhees_09_24_2025_blackbird

Run 4 - Dumping

The dump attribute puts all of the found account HTTP responses into a folder for later examination

```
[ ] !python blackbird.py --dump --username jasonvoorhees
```

```
✔ [StreamElements] https://api.streamelements.com/kappa/v2/channels/jasonvoorhees
    ➞ Avatar: https://yt3.ggpht.com/ytc/AAUvwnjDqSnvGQrl-iJ1GiQBHuoJIgopLyRXNF6YCd9_Bg=s88-c-k-c0x00ffffff-no-rj
    ➞ Name: Jason Voorhees
✔ [pikabu] https://pikabu.ru/@jasonvoorhees
✔ [StreamLabs] https://streamlabs.com/api/v6/user/jasonvoorhees
✔ [Habbo.com] https://www.habbo.com/api/public/users?name=jasonvoorhees
✔ [Habbo.com.br] https://www.habbo.com.br/api/public/users?name=jasonvoorhees
✔ [Filmweb] https://www.filmweb.pl/user/jasonvoorhees
✔ [fansly] https://apiv2.fansly.com/api/v1/account?usernames=jasonvoorhees
✔ [Stripchat] https://stripchat.com/api/front/users/checkUsername?username=jasonvoorhees
✔ [Habbo.com.tr] https://www.habbo.com.tr/api/public/users?name=jasonvoorhees
✔ [Calendly] https://calendly.com/jasonvoorhees
✔ [Habbo.de] https://www.habbo.de/api/public/users?name=jasonvoorhees
✔ [Habbo.es] https://www.habbo.es/api/public/users?name=jasonvoorhees
✔ [Flipboard] https://flipboard.com/@jasonvoorhees
✔ [Wykop] https://wykop.pl/ludzie/jasonvoorhees
✔ [Poe.com] https://poe.com/profile/jasonvoorhees
✔ [Mastodon API] https://mastodon.social/api/v2/search?q=jasonvoorhees&limit=1&type=accounts
✔ [Pinterest] https://www.pinterest.com/jasonvoorhees/
✔ [diigo] https://www.diigo.com/interact_api/load_profile_info?name=jasonvoorhees
✔ [CastingCallClub] https://www.castingcall.club/jasonvoorhees
✔ [Discogs] https://api.discogs.com/users/jasonvoorhees
✔ [X] https://api.x.com/i/users/username_available.json?username=jasonvoorhees
✔ [Polarsteps] https://api.polarsteps.com/users/byusername/jasonvoorhees
✔ [Disqus] https://disqus.com/api/3.0/users/details?user=username:jasonvoorhees&api_key=E8Uh5l5fHZ6gD8U3KycjAIAk46f68Zw7C6eW8WSjZvCLXebZ7p0r1yrYDrLilk2F
✔ [Fotolog Archived Profile] https://archive.org/wayback/available?url=https://www.fotolog.com/jasonvoorhees
✔ [Docker Hub Organizations] https://hub.docker.com/v2/orgs/jasonvoorhees/
✔ [Docker Hub Users] https://hub.docker.com/v2/users/jasonvoorhees/
✔ [chatango.com] https://jasonvoorhees.chatango.com
✔ [championat] https://www.championat.com/user/jasonvoorhees/
✔ [XVIDEOS-profiles] https://www.xvideos.com/profiles/jasonvoorhees
✔ [Hacker News] https://news.ycombinator.com/user?id=jasonvoorhees
✔ [Habtium] https://habtium.es/jasonvoorhees
✔ [Chess.com] https://api.chess.com/pub/player/jasonvoorhees
✔ [cda.pl] https://www.cda.pl/jasonvoorhees
✔ [Pornhub Users] https://www.pornhub.com/users/jasonvoorhees
✔ [YouNow] https://api.younow.com/php/api/broadcast/info/user=jasonvoorhees
✔ [Chomikuj.pl] https://chomikuj.pl/jasonvoorhees/
✔ [HackerRank] https://www.hackerrank.com/rest/contests/master/hackers/jasonvoorhees/profile
✔ [Telegram] https://t.me/jasonvoorhees
✔ [Donation Alerts] https://www.donationalerts.com/api/v1/user/jasonvoorhees/donationpagesettings
✔ [Xbox Gamertag] https://www.xboxgamertag.com/search/jasonvoorhees
Enumerating accounts with username "jasonvoorhees" — 100% (724/724)
Check completed in 45.9 seconds
Dump content saved to 'jasonvoorhees_09_24_2025_blackbird/dump_jasonvoorhees'
```


blackbird
assets
data
docs
logs
results
jasonvoorhees_09_24_2025_blackbird
images_jasonvoorhees
jasonvoorhees_09_24_2025_blackbird.pdf

Run 5 - Filters

Many filters can be applied to searches, such as a categorical search.

```
[42]
✓ 32s
!python blackbird.py --filter "cat=social" --username elonmusk

[SEOClerks] https://www.seoclerks.com/user/elonmusk
[slides] https://slides.com/elonmusk
[untappd] https://untappd.com/user/elonmusk/
[Snapchat] https://www.snapchat.com/@elonmusk
[Cent] https://beta.cent.co/data/user/profile?userHandles=elonmusk
[Slideshare] https://www.slideshare.net/elonmusk
[Clubhouse] https://www.clubhouse.com/@elonmusk
[KnowYourMeme] https://knowyourmeme.com/users/elonmusk
[Geocaching] https://www.geocaching.com/p/?u=elonmusk
[Gettr] https://api.gettr.com/s/user/elonmusk/exist
[Mastodon-pol.social] https://pol.social/@elonmusk
[chatango.com] https://elonmusk.chatango.com
[Mastodon-social_tchncs] https://social.tchncs.de/@elonmusk
[Wattpad] https://www.wattpad.com/api/v3/users/elonmusk
[Coub] https://coub.com/api/v2/channels/elonmusk
[Pillowfort] https://www.pillowfort.social/elonmusk
[pikabu] https://pikabu.ru/@elonmusk
[Minds] https://www.minds.com/api/v3/register/validate?username=elonmusk
[Kwai] https://www.kwai.com/@elonmusk
[Mastodon-Toot.Community] https://toot.community/@elonmusk
[Letterboxd] https://letterboxd.com/elonmusk/
[Lemon8] https://www.lemon8-app.com/elonmusk?region=us
[TikTok] https://www.tiktok.com/oembed?url=https://www.tiktok.com/@elonmusk
    → Name: elonmusk
[about.me] https://about.me/elonmusk
[Pinterest] https://www.pinterest.com/elonmusk/
[Linktree] https://linktr.ee/elonmusk
[Wikidot] http://www.wikidot.com/user:info/elonmusk
[Disqus]
https://disqus.com/api/3.0/users/details?user=username:elonmusk&api_key=E8Uh5l5f
HZ6gD8U3KycjAIAk46f68Zw7C6eW8WSjZvCLXebZ7p0r1yrYDrLilk2F
[Substack] https://substack.com/@elonmusk
[mssg.me] https://elonmusk.mssg.me/
[Speaker Deck] https://speakerdeck.com/elonmusk/
[Weblancer] https://www.weblancer.net/users/elonmusk/
[MyAnimeList] https://myanimelist.net/profile/elonmusk
[Instagram_archives]
https://archive.org/wayback/available?url=https://instagram.com/elonmusk/
[Hometech.social (Mastodon Instance)]
https://hometech.social/api/v1/accounts/lookup?acct=elonmusk
[Habr] https://habr.com/ru/users/elonmusk/
[X] https://api.x.com/i/users/username_available.json?username=elonmusk
[Albicla] https://albicla.com/elonmusk/post/1
[MySpace] https://myspace.com/elonmusk
[allmylinks] https://allmylinks.com/elonmusk
[Pr0gramm] https://pr0gramm.com/api/profile/info?name=elonmusk
[Telegram] https://t.me/elonmusk
[Wykop] https://wykop.pl/ludzie/elonmusk
[YouNow] https://api.younow.com/php/api/broadcast/info/user=elonmusk
[Pronouny] https://pronouny.xyz/api/users/profile/username/elonmusk
[masto.ai] https://masto.ai/@elonmusk
[Pronouns.Page] https://pronouns.page/api/profile/get/elonmusk?version=2
[Naver] https://blog.naver.com/elonmusk
[Neocities] https://neocities.org/site/elonmusk
[anonup] https://anonup.com/@elonmusk
[palnet] https://www.palnet.io/@elonmusk/
Enumerating accounts with username "elonmusk" — 100% (213/213)
Check completed in 32.3 seconds
```

Multiple usernames with site filter search

```
!python blackbird.py --username donaldtrump jeffreyepstein melaniatrump gmaxwell --filter "n
```

```
Investigate deeper with https://sherlockeye.io | by Lucas Antoniaci
Checking for updates...
 Sites List is up to date
Applied "name=Snapchat" filter to sites [1]
  [Snapchat] https://www.snapchat.com/@donaldtrump
 Enumerating accounts with username "donaldtrump" — 100% (1/1)
Check completed in 0.3 seconds
Applied "name=Snapchat" filter to sites [1]
  [Snapchat] https://www.snapchat.com/@jeffreyepstein
 Enumerating accounts with username "jeffreyepstein" — 100% (1/1)
Check completed in 0.1 seconds
Applied "name=Snapchat" filter to sites [1]
  [Snapchat] https://www.snapchat.com/@melaniatrump
 Enumerating accounts with username "melaniatrump" — 100% (1/1)
Check completed in 1.6 seconds
Applied "name=Snapchat" filter to sites [1]
  [Snapchat] https://www.snapchat.com/@gmaxwell
 Enumerating accounts with username "gmaxwell" — 100% (1/1)
Check completed in 0.1 seconds
```

AI is avaible to summarize the findings of each search subject.

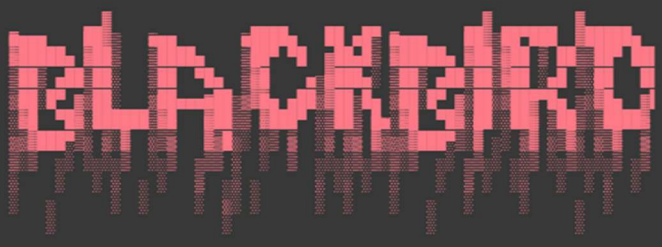
For this, an API key must be generated and agreement accepted:

```
[ ]
!python blackbird.py --setup-ai
```

```
Blackbird loves breadcrumbs. | by Lucas Antoniaci
Checking for updates...
 Sites List is up to date
! By continuing, you acknowledge that your IP is registered for API key
management and abuse prevention. [Y/n] >
```

```
!python blackbird.py --username innocentuser -ai
https://www.tiktok.com/oembed?url=https://www.tiktok.com/@innocentuser
      → Name: "🌸"
  ✔ [Letterboxd] https://letterboxd.com/innocentuser/
  ✔ [GitHub] https://api.github.com/users/innocentuser
  ✔ [lichess.org]
https://lichess.org/api/player/autocomplete?term=innocentuser&exists=1
  ✔ [GNOME GitLab] https://gitlab.gnome.org/api/v4/users?username=innocentuser
  ✔ [tumblr] https://innocentuser.tumblr.com
  ✔ [Habbo.com] https://www.habbo.com/api/public/users?name=innocentuser
  ✔ [Stripchat]
https://stripchat.com/api/front/users/checkUsername?username=innocentuser
  ✔ [Pinterest] https://www.pinterest.com/innocentuser/
  ✔ [Discogs] https://api.discogs.com/users/innocentuser
  ✔ [Disqus]
https://disqus.com/api/3.0/users/details?user=username:innocentuser&api_key=E8Uh
5l5fHZ6gD8U3KycjAIAk46f68Zw7C6eW8WSjZvCLXebZ7p0r1yrYDrLilk2F
  ✔ [Hacker News] https://news.ycombinator.com/user?id=innocentuser
  ✔ [Pornhub Users] https://www.pornhub.com/users/innocentuser
  ✔ [YouTube User] https://www.youtube.com/user/innocentuser/about
  ✔ [YouTube User2] https://www.youtube.com/@innocentuser
  ✔ [hugging_face] https://huggingface.co/innocentuser
  ✔ [Blogspot] http://innocentuser.blogspot.com
  ✔ [Roblox]
https://auth.roblox.com/v1/usernames/validate?username=innocentuser&birthday=201
9-12-31T23:00:00.000Z
  ✔ [Filmot Channel Search] https://filmot.com/channelsearch/innocentuser
  ✔ [WordPress Support] https://wordpress.org/support/users/innocentuser/
  ✔ [X] https://api.x.com/i/users/username_available.json?username=innocentuser
  ✔ [Newgrounds] https://innocentuser.newgrounds.com/
  ✔ [chaturbate] https://chaturbate.com/innocentuser/
  ✔ [Chess.com] https://api.chess.com/pub/player/innocentuser
  ✔ [Xbox Gamertag] https://www.xboxgamertag.com/search/innocentuser
  Enumerating accounts with username "innocentuser" — 100% (724/724)
  Check completed in 36.1 seconds
  Analyzing with AI...
[Summary]
>  The individual has a diverse online presence, engaging with various platforms for creative expression, social interaction, and perso

[Profile Type]
>  Tech Savvy Creator

[Insights]
>  - Engages with developer communities through GitHub and GNOME GitLab
>  - Participates in social media and content sharing on platforms like Tumblr, TikTok, and YouTube
>  - Shows interest in gaming through Roblox, Xbox, and Newgrounds
>  - Has a presence on chess-related platforms, indicating a strategic hobby
>  - Utilizes discussion forums like Disqus and Hacker News, suggesting an interest in technology and online discourse

[Risk Flags]
>  - Potential adult content exposure
>  - Online gaming security risks
>  - Publicly visible personal interests

[Tags]
>  Developer, Gamer, Content Creator, Social Media User, Chess Enthusiast

9 AI queries left for today
```

```
!python blackbird.py --username fearnobacon -ai
```



```
            You can run, but we cached. | by Lucas Antoniaci
  Checking for updates...
  ✔ Sites List is up to date
  ! By proceeding, you consent to share the found site names with Blackbird AI
for analysis. [Y/n] > Y
  ✔ [Xbox Gamertag] https://www.xboxgamertag.com/search/fearnobacon
  Enumerating accounts with username "fearnobacon" — 100% (724/724)
  Check completed in 36.1 seconds
⚠ Not enough accounts found for AI analysis. Skipping AI features.
```

The output shows the script running and the AI will attempt to generate a summary, but will indicate that not enough results were found

Example of Code verifying account existence and acting accordingly

Code	What's Happening?

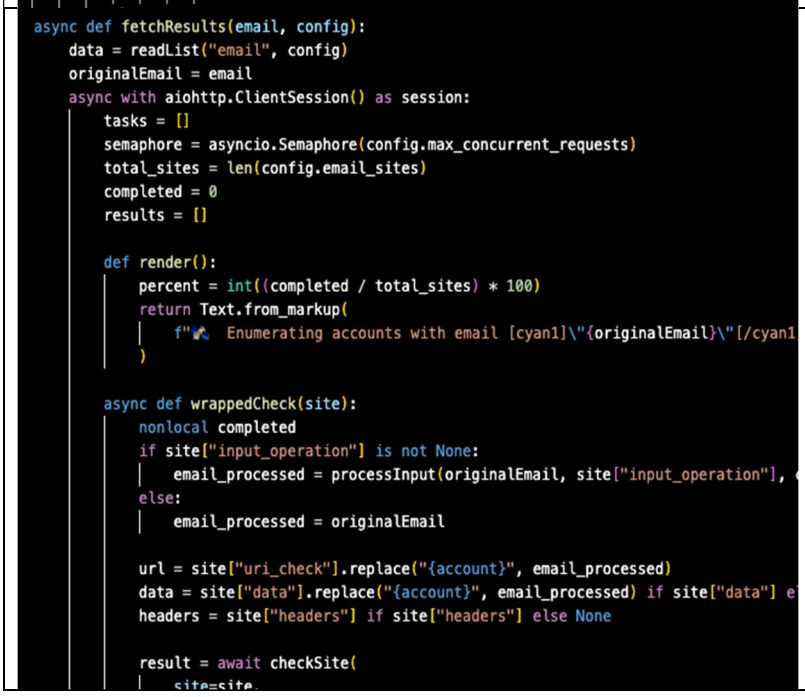
email.py
`# Verify account existence based on list args`
`async def checkSite(`
`site,`
`method,`
`url,`
`session,`
`semaphore,`
`config,`
`data=None,`
`headers=None,`
`):`
	This takes place on the email.py were
	1) It defines an asynchronous function (GeeksforGeeks, prajjqv, 2025). Checksite (), which checks for existing accounts on sites asynchronously
	2) Then it verifies existence based on site list args.

`returnData = {`
`"name": site["name"],`
`"url": url,`
`"category": site["cat"],`
`"status": "NONE",`
`"metadata": None,`
`}`
`async with semaphore:`
`if site["pre_check"]:`
`authenticated_headers = perform_pre_check(`
`site["pre_check"], headers, config`
`)`
`headers = authenticated_headers`
`if headers == False:`
`returnData["status"] = "ERROR"`
`return returnData`
	3) A dictionary (W3Schools) returnData is created, which stores/contains anything from the site name, URL, category, status and metadata etc. To then return it in an easier to read format.

```
async with semaphore:
    if site["pre_check"]:
        authenticated_headers = perform_pre_check(
            site["pre_check"], headers, config
        )
        headers = authenticated_headers
        if headers == False:
            returnData["status"] = "ERROR"
            return returnData

    response = await do_async_request(method, url, session, config, data, headers)
    if response == None:
        returnData["status"] = "ERROR"
        return returnData
    try:
        if response:
            if (site["e_string"] in response["content"]) and (
                site["e_code"] == response["status_code"]
            ):
                if (site["m_string"] not in response["content"]) and (
                    site["m_code"] != response["status_code"]
                ):
                    returnData["status"] = "FOUND"
                    config.console.print(
                        rf"  ✓  \[[cyan1]{site['name']}[/cyan1]] [bright_white]{response['url']}[/bright_white]"
                    )
                    if site["metadata"]:
                        extractedMetadata = extractMetadata(
                            site["metadata"], response, site["name"], config
                        )
                        extractedMetadata.sort(key=lambda x: x["name"])
                        returnData["metadata"] = extractedMetadata
                    # Save response content to a .HTML file
                    if config.dump:
                        path = os.path.join(
                            config.saveDirectory, f"dump_{config.currentEmail}"
                        )

                        result = dumpContent(path, site, response, config)
                        if result == True and config.verbose:
                            config.console.print(
                                f"      Saved HTML data from found account"
                            )
```

4) It then works with shared resources (Foundation., P. S) and checks for pre_check, metadata. If there is no content found from the asynchronous function, then it gives you errors. If something is found, then it returns to the user their expected data.

5) It also Save response content to a HTML file



```
async def fetchResults(email, config):
    data = readList("email", config)
    originalEmail = email
    async with aiohttp.ClientSession() as session:
        tasks = []
        semaphore = asyncio.Semaphore(config.max_concurrent_requests)
        total_sites = len(config.email_sites)
        completed = 0
        results = []

        def render():
            percent = int((completed / total_sites) * 100)
            return Text.from_markup(
                f"  Enumerating accounts with email [cyan1]\"{originalEmail}\"[/cyan1
            )

        async def wrappedCheck(site):
            nonlocal completed
            if site["input_operation"] is not None:
                email_processed = processInput(originalEmail, site["input_operation"],
            else:
                email_processed = originalEmail

            url = site["uri_check"].replace("{account}", email_processed)
            data = site["data"].replace("{account}", email_processed) if site["data"] e
            headers = site["headers"] if site["headers"] else None

            result = await checkSite(
                site=site,
```

6) The fetchResults (email,config) is defined as an asynchronous function.

7) It then reads a list of sites and uses a Semaphore object. Which limits things happening at the same time

8) Then it returns the appropriate results to the user

```python
# Start email check and presents results to user
def verifyEmail(email, config):

    data = readList("email", config)
    sitesToSearch = data["sites"]
    config.email_sites = applyFilters(sitesToSearch, config)

    start_time = time.time()
    results = asyncio.run(fetchResults(email, config))
    end_time = time.time()
```

9) As the name states this starts the email check and presents result to user

10) It applies filters so we only search for certain sites

11) If the email doesn't exist it returns “⭕ No accounts were found for the given email”

username.py

```python
# Start username check and presents results to user
def verifyUsername(username, config, sitesToSearch=None, metadata_params=None):
    if sitesToSearch is None or metadata_params is None:
        data = readList("username", config)
        sitesToSearch = data["sites"]
        config.metadata_params = readList("metadata", config)
    else:
        config.metadata_params = metadata_params

    config.username_sites = applyFilters(sitesToSearch, config)

    start_time = time.time()
    results = asyncio.run(fetchResults(username, config))
    end_time = time.time()

    config.console.print(
        f":chequered_flag: Check completed in {round(end_time - start_time, 1)} seconds"
    )
```

This takes place on the username.py were

1) The email.py and username.py are almost identical the only difference is that this is searching for username

2) Another difference is that utilizes metadata parameters

3) It then presents result to user

pdf.py

This takes place on the

```python
def saveToPdf(foundAccounts, resultType, config):
    regularFontFile = os.path.join(
        os.getcwd(),
        config.ASSETS_DIRECTORY,
        config.FONTS_DIRECTORY,
        config.FONT_REGULAR_FILE,
    )
    boldFontFile = os.path.join(
        os.getcwd(),
        config.ASSETS_DIRECTORY,
        config.FONTS_DIRECTORY,
        config.FONT_BOLD_FILE,
    )
    try:
        pdfmetrics.registerFont(TTFont(config.FONT_NAME_REGULAR, regularFontFile))
        pdfmetrics.registerFont(TTFont(config.FONT_NAME_BOLD, boldFontFile))

        fileName = generateName(config, "pdf")
        path = os.path.join(config.saveDirectory, fileName)

        width, height = letter
        canva = canvas.Canvas(path, pagesize=letter)
        accountsCount = len(foundAccounts)

        canva.drawImage(
            os.path.join(
                os.getcwd(),
                config.ASSETS_DIRECTORY,
                config.IMAGES_DIRECTORY,
                "blackbird-logo.png",
```

pdf.py were

1) It first sets the font to the appropriate font styles.
2) It then creates a filename that generates a name for the file and saves it as a PDF.
3) It then saves it in your specified directory
4) It utilizes canvas to create an image of the report

```python
        "Blackbird can make mistakes. Consider checking the information.",
    )

    if config.ai_analysis:
        # Background do quadro
        canva.setFillColor("#F4F6F8")
        canva.setStrokeColor("#D0D5DA")
        canva.rect(40, height - 545, 530, 320, stroke=1, fill=1)

        # Cabeçalho
        canva.setFillColor("#000000")
        canva.drawImage(
            os.path.join(os.getcwd(), config.ASSETS_DIRECTORY, config.IMAGES_DIRECTORY, "ai
            55, height - 245, width=12, height=12, mask="auto"
        )
        canva.setFont(config.FONT_NAME_BOLD, 10)
        canva.drawString(70, height - 242, "AI Analysis - Behavioral Summary")
        canva.setFont(config.FONT_NAME_REGULAR, 8)
        canva.drawString(55, height - 255, "This behavioral summary was generated using AI |

        y_position = height - 270

        if config.ai_analysis.get("summary"):
            canva.setFont(config.FONT_NAME_BOLD, 10)
            canva.drawString(55, y_position, "Summary")
            y_position -= 12

            lines = simpleSplit(config.ai_analysis["summary"], config.FONT_NAME_REGULAR, 8,
            text = canva.beginText()
            text.setTextOrigin(55, y_position)
```

5) It then takes the info gather by the AI report and uses canvas to put the information into the crated pdf
6) It then saves results to the appropriate file
7) If there's no report it gives an error "Couldn't save results to PDF file!"”

email-data.json

```
"categories": [
    "archived",
    "art",
    "blog",
    "business",
    "coding",
    "dating",
    "finance",
    "gaming",
    "health",
    "hobby",
    "images",
    "misc",
    "music",
    "news",
    "political",
    "search",
    "shopping",
    "social",
    "tech",
    "video",
    "xx NSFW xx"
]
```

This is a list of all available categories to both categorize sites and be used for filtering search results.

```
"name": "Notion",
"uri_check": "https://www.notion.so/api/v3/getLoginOptions",
"data": "{\"email\":\"{account}\"}",
"method": "POST",
"headers": {
    "Content-Type": "application/json"
},
"e_code": 200,
"e_string": "\"hasAccount\":true",
"m_string": "\"hasAccount\":false",
"m_code": 404,
"known": [],
"cat": "misc",
"input_operation": null,
"metadata": null,
"pre_check": null
```

- This object defines how to interact with Notion's login API to check if an account (email) exists. The name of the service being checked — in this case, Notion.
- The API endpoint used to verify login options for a given email. This is where the POST request will be sent.
- The payload sent in the POST request. It's a JSON string with an email field. `{account}` is a placeholder that will be replaced with the actual email being tested.
- Specifies the HTTP method used to send the request — here, it's a POST.

wmn-data.json


```
"sites": [
    {
        "name": "21buttons",
        "uri_check": "https://www.21buttons.com/buttoner/{account}",
        "e_code": 200,
        "e_string": "profile-info__profile-data__name",
        "m_string": "This is not the page you're looking for",
        "m_code": 404,
        "known": [
            "patricialmendro",
            "ginamariahoffmann",
            "espeworkout"
        ],
        "cat": "social"
    },
```


- The name of the service being checked — 21buttons, a fashion-focused social network.
- The URL used to check if a username exists. `{account}` is a placeholder that gets replaced with the target username.
- Expected HTTP status code if the account exists. A `200 OK` means the profile page was successfully loaded.
-
- Expected HTTP status code if the account exists. A `200 OK` means the profile page was successfully loaded.
- Category label for the service. "social" indicates it's a social media platform.

```python
def initiate():
    if not os.path.exists("logs/"):
        os.makedirs("logs/")
    logging.basicConfig(
        filename=config.LOG_PATH,
        level=logging.DEBUG,
        format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
    )

    parser = argparse.ArgumentParser(
        prog="blackbird",
        description="An OSINT tool to search for accounts by username in social networks.",
    )
    parser.add_argument(
        "-u",
        "--username",
        nargs="*",
        type=str,
        help="One or more usernames to search.",
    )
    parser.add_argument(
        "-uf",
        "--username-file",
        help="The list of usernames to be searched.",
    )
    parser.add_argument(
        "--permute",
        action="store_true",
        help="Permute usernames, ignoring single elements.",
    )
```

The initiate() function is within the blackbird.py file. It sets up logging by creating a log directory if it doesn't exist and stores log files there. Then, it creates a parser for command line interface arguments. Overall, the initiate() function initializes the application, sets up logs and parses user inputs so that they can be used to search for the provided username(s) and email(s).

```python
    if config.email_file:
        if isFile(config.email_file):
            config.email = getLinesFromFile(config.email_file)
            config.console.print(
                f':glasses: Successfully loaded {len(config.email)} emails from "{config.email_file}"'
            )
        else:
            config.console.print(f'❌ Could not read file "{config.email_file}"')
            sys.exit()

    if config.email:
        for email in config.email:
            config.currentEmail = email
            if config.dump or config.csv or config.pdf or config.json:
                createSaveDirectory(config)
            verifyEmail(email, config)
            if config.ai:
                if len(config.emailFoundAccounts) > 2:
                    from modules.ai.client import send_prompt
                    site_names = [account.get("name", "") for account in config.emailFoundAccounts]
                    if (site_names):
                        prompt = ", ".join(site_names)

                        data = send_prompt(prompt, config)

                        if (data):
                            config.ai_analysis = data
                else:
                    config.console.print(
                        ":warning: Not enough accounts found for AI analysis. Skipping AI features."
                    )
```

This code is part of the Main function in blackbird.py. It is run after the command line arguments are parsed. It begins with checking if emails were entered in by the user, loops through each email and creates a save directory depending on user input. Then, it uses the verifyEmail() function to search for the provided email and stores what's found. There is also AI analysis if the user input it that takes the site names from accounts that were found and sends it to AI. There's similar code for checking usernames as well.

SOURCES

Antoniaci, L. (2020). BlackBird. https://p1ngul1n0.gitbook.io/blackbird

GeeksforGeeks, prajjqv. (2025, July 23). Python async. GeeksforGeeks, prajjqv. (2025, July 23). Python async. https://www.geeksforgeeks.org/python/python-async/

W3Schools. (n.d.). W3schools.com. W3Schools Online Web Tutorials.

https://www.w3schools.com/python/python_dictionaries.asp

Foundation., P. S. (n.d.). Synchronization primitives. Python documentation.

https://docs.python.org/3/library/asyncio-sync.html