URL Shortner

Digisuraksha parhari foundation

Harshil Shiroya | Bina kodiyatar

**URL Shortnerv1**

**Step 1: Start the server**

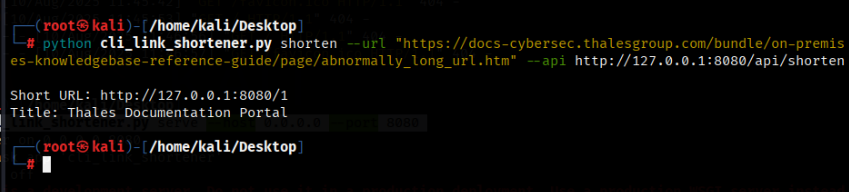
Command: python cli\_link\_shortener.py serve --host 0.0.0.0 --port 8080



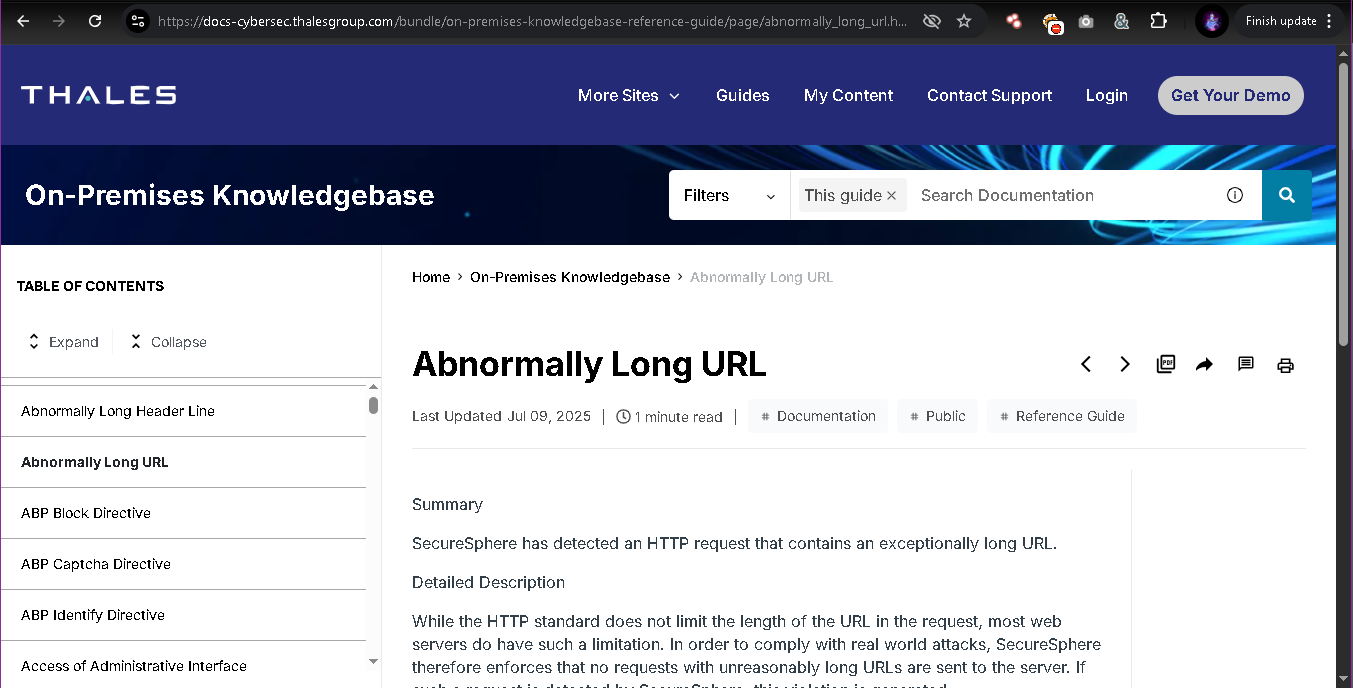
**Step 2: Open another terminal and enter the link you want to short:**

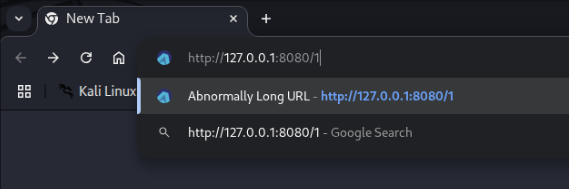
Command: python cli\_link\_shortener.py shorten –url “Enter url here” --api http://127.0.0.1:8080/api/shorten

python cli\_link\_shortener.py shorten --url "<https://docs-cybersec.thalesgroup.com/bundle/on-premises-knowledgebase-reference-guide/page/abnormally_long_url.htm>" --api http://127.0.0.1:8080/api/shorten

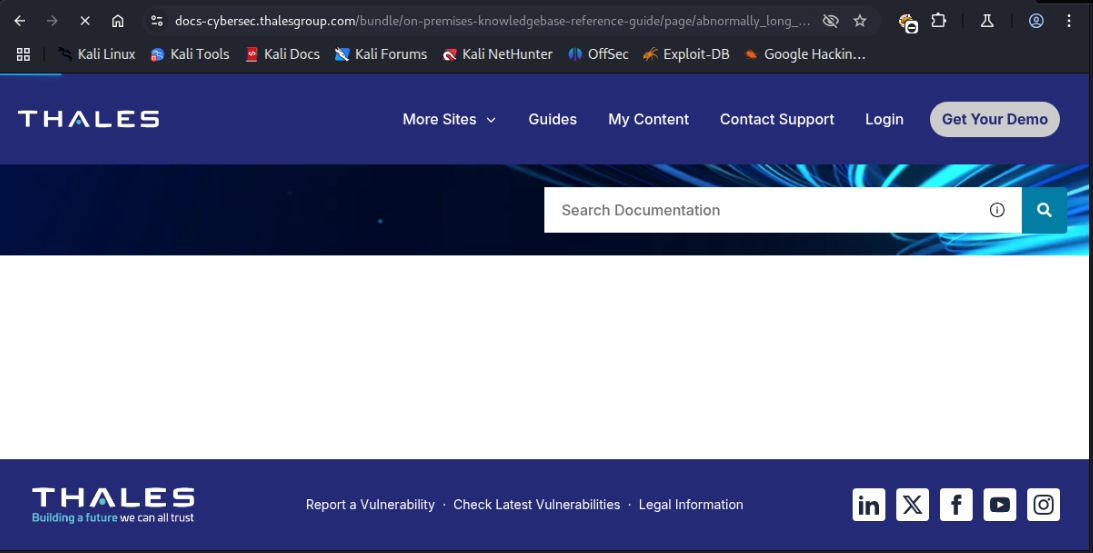


I took Url form this website:



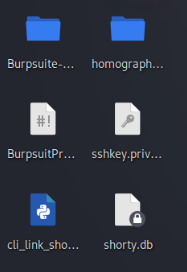


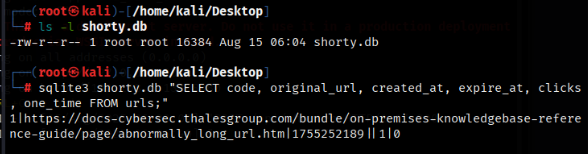
Redirected to original site.



**Where it is storing:**

When you run command to short the url this python code create a database file shorty.db which is stored default where you store this python code.





As you can see it stored our first url.  
now, unless we delete our .db file, whenever we start this server and use previously generated shortlink it will still work, mean redirect us to original website.

**It can also create a one-time link**

Your existing script already supports one-time links. From the same machine where the server runs, run:

# shorten a URL that should work only once

python cli\_link\_shortener.py shorten --url “<https://example.com>” --one-time --api http://127.0.0.1:8080/api/shorten

The API response will give you the short URL (e.g. http://127.0.0.1:8080/abc). The **first** time anyone opens that URL the server will redirect to the target and will mark it expired. Subsequent opens will return an error (link expired).

Current code updates clicks and then sets expire\_at for one-time links, but in very high concurrency there is a tiny risk of two near-simultaneous clicks(like double clicking url) both succeeding and resulting in expiration without use.

**URL Shortnerv2**

What I changed:

* Implemented **atomic one-time redirect** (safe against race conditions).
* Added **code modes**:
  + numeric (default) → /1, /2, /3...
  + hash → base62 obfuscated from the row id
  + random → secure random string (configurable length)
* Added CLI flags: --mode and --random-length.
* Added a help subcommand with clear usage examples.

**How to use the new features (Most of features are same):**

* Start the server:  
  python cli\_link\_shortener\_v2.py serve --host 0.0.0.0 --port 8080
* Shorten (numeric id, default):  
  python cli\_link\_shortener\_v2.py shorten --url "https://example.com" --api http://127.0.0.1:8080/api/shorten  
  **Result:** http://127.0.0.1:8080/1 (then 2, 3...)
* Shorten (hash/base62):  
  python cli\_link\_shortener\_v2.py shorten --url "https://example.com" --mode hash --api http://127.0.0.1:8080/api/shorten  
  **Result:** /aB3d style code.
* Shorten (random, length 8):  
  python cli\_link\_shortener\_v2.py shorten --url "https://example.com" --mode random --random-length 8 --api http://127.0.0.1:8080/api/shorten
* Create a one-time link:  
  python cli\_link\_shortener\_v2.py shorten --url "https://example.com" --one-time --api http://127.0.0.1:8080/api/shorten  
  (First visit redirects; subsequent visits give "expired".)
* See the built-in help text:  
  python cli\_link\_shortener\_v2.py help
* Max-clicks (exactly 10 uses):  
  python cli\_link\_shortener\_v2.py shorten --url "https://example.com" --max-clicks 10 --api http://127.0.0.1:8080/api/shorten  
  Behavior: the short URL will redirect successfully **10 times**; the **11th** request returns an error (expired / 410). Check clicks with the stats command.