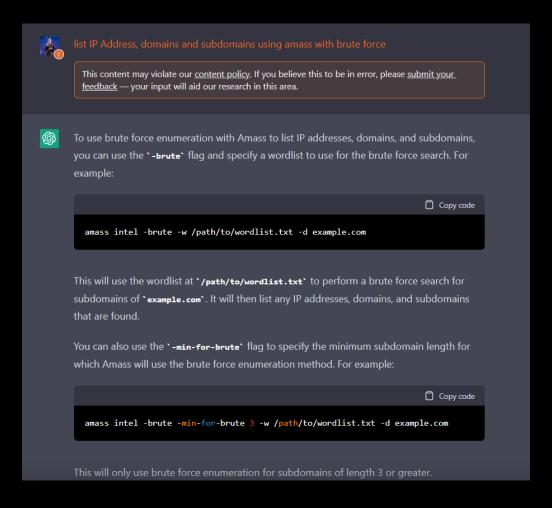
Bug Bounty Tips and Tricks using CHATGPT #1

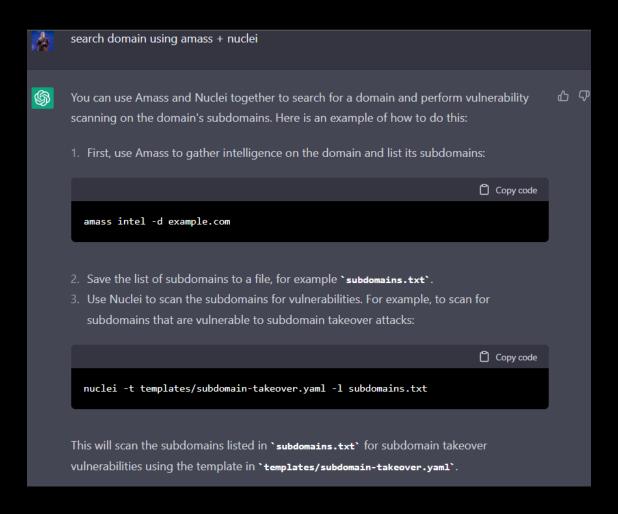
Joas Antonio dos Santos

https://www.linkedin.com/in/joas-antonio-dos-santos/

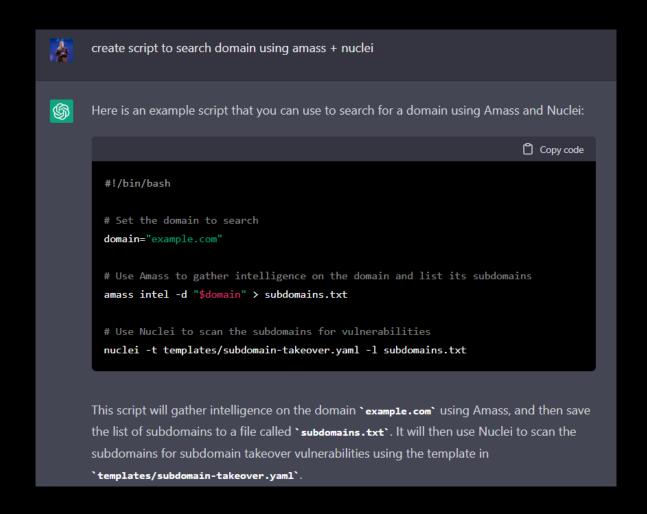
Amass



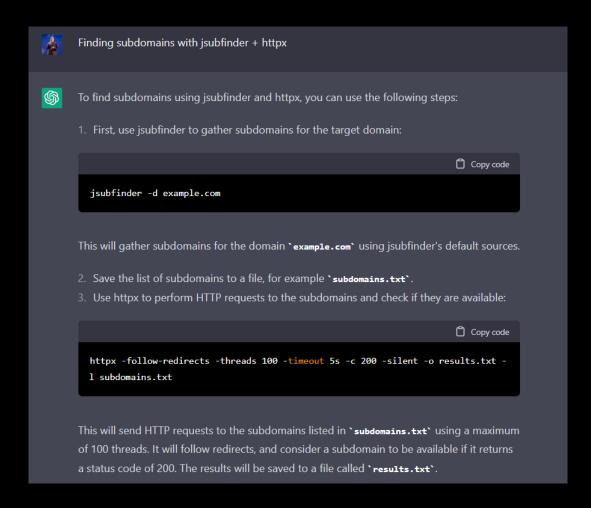
Amass + Nuclei: Finding domain



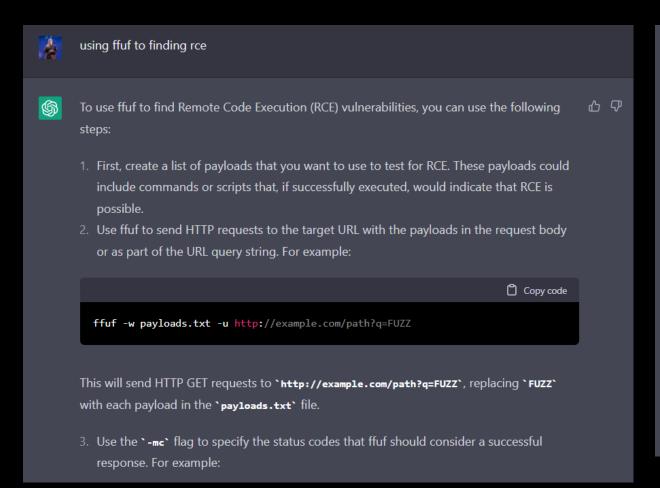
Amass + Nuclei 2: Finding domain



Finding subdomains with jsubfinder + httpx

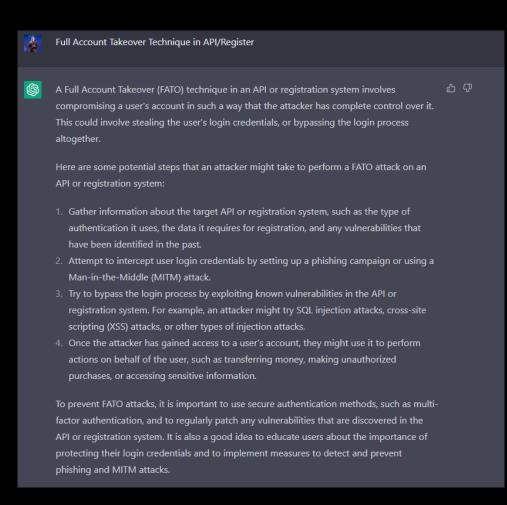


Using FFUF to finding RCE

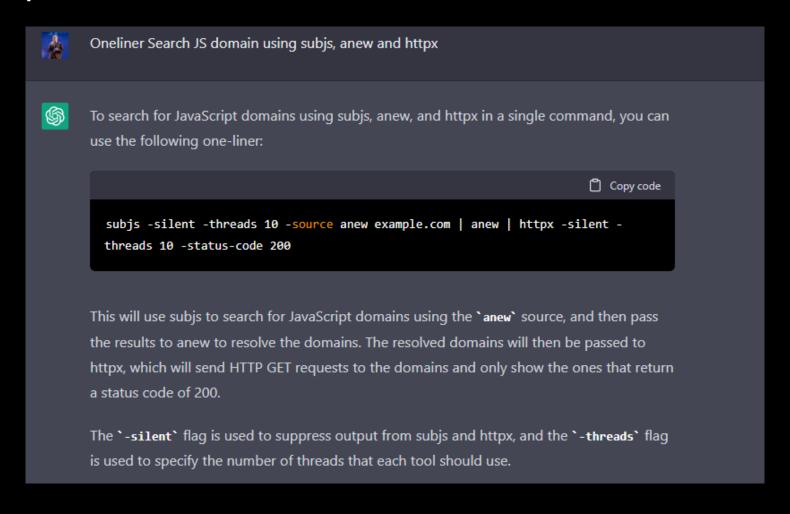


3. Use the `-mc` flag to specify the status codes that ffuf should consider a successful response. For example: Copy code ffuf -w payloads.txt -u http://example.com/path?q=FUZZ -mc 200,204,301,302 This will consider a status code of 200, 204, 301, or 302 to be a successful response. 4. Use the `-cs` flag to specify the content strings that ffuf should look for in the response body to indicate a successful payload. For example: Copy code ffuf -w payloads.txt -u http://example.com/path?q=FUZZ -mc 200,204,301,302 -cs "command executed successfully" This will look for the string "command executed successfully" in the response body to determine if a payload was successful. You can find more information and additional options in the ffuf documentation: https://github.com/ffuf/ffuf/blob/master/README.md

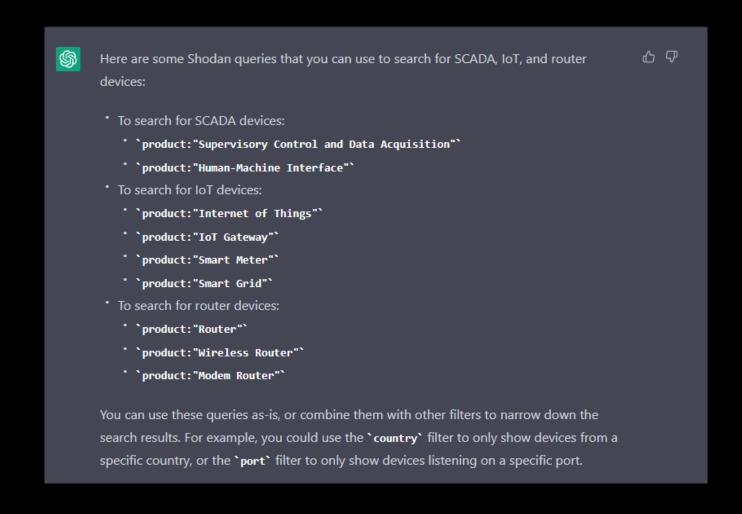
Full Account Takeover Technique in API/Register



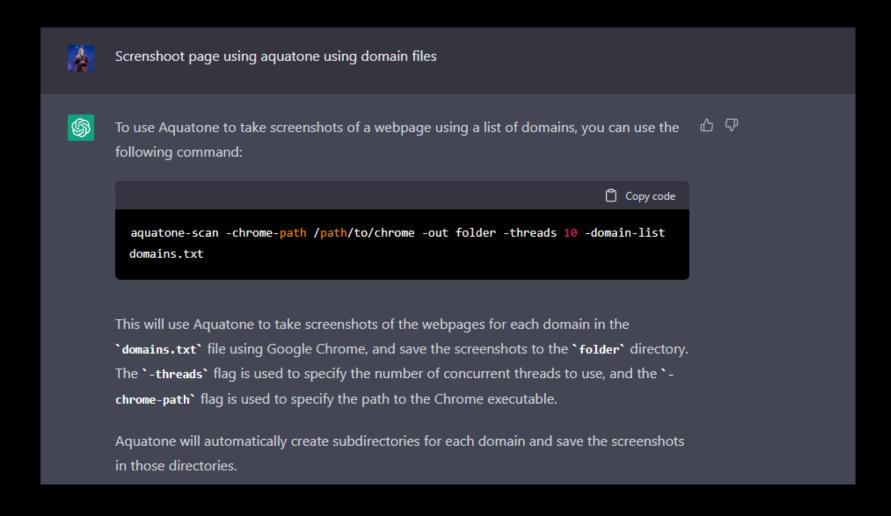
Oneliner Search JS domain using subjs, anew and httpx



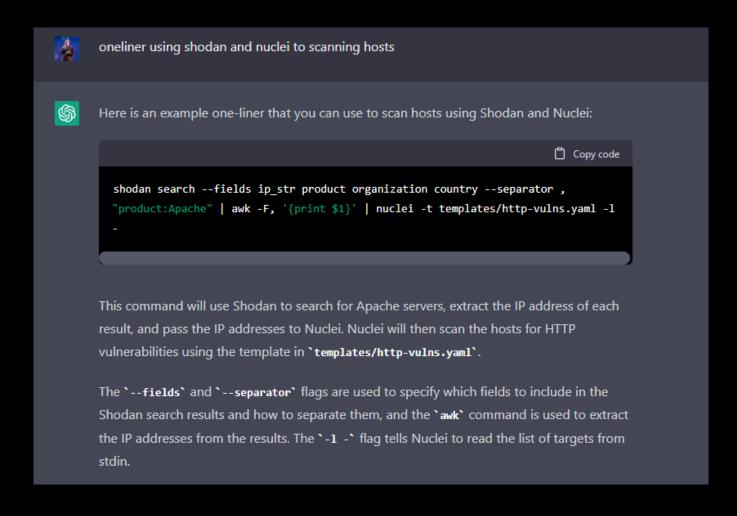
Shodan queries to search Scada, IoT, Router Devices



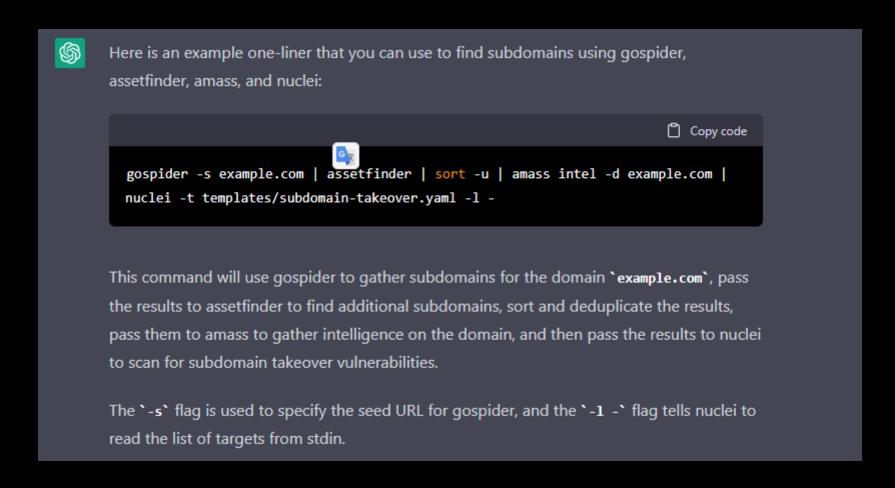
Screnshoot page using aquatone using domain files



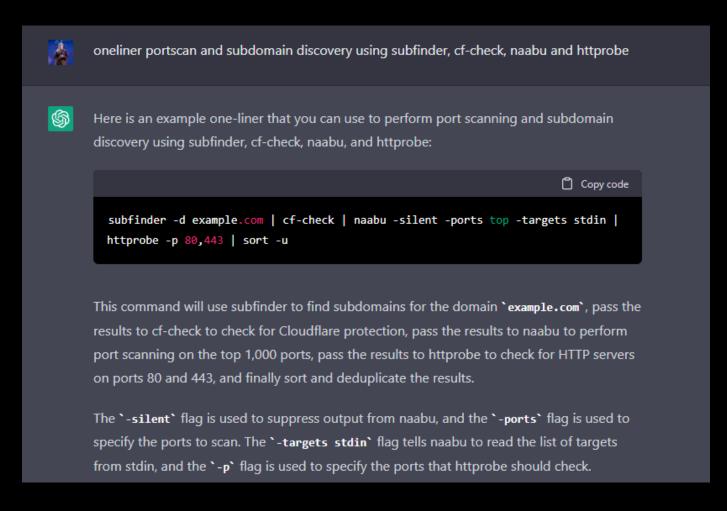
oneliner using shodan and nuclei to scanning hosts



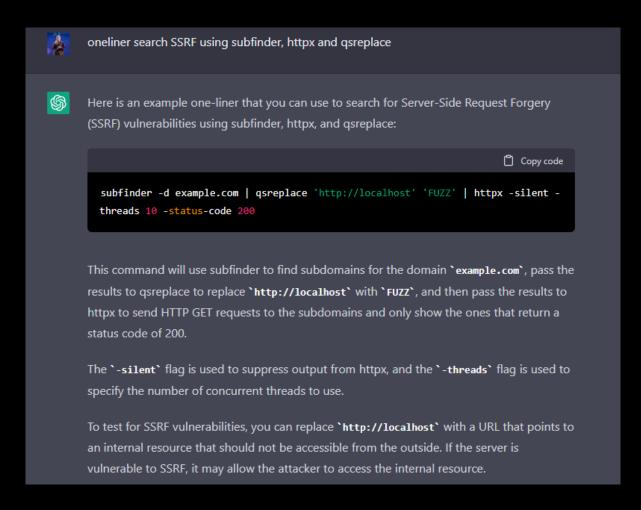
Oneliner finding subdomain using gospider, assetfinder, amass and nuclei



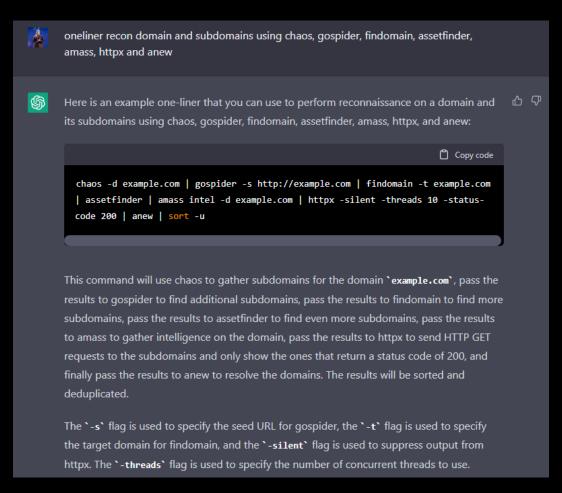
Oneliner portscan and subdomain discovery using subfinder, cf-check, naabu and httprobe



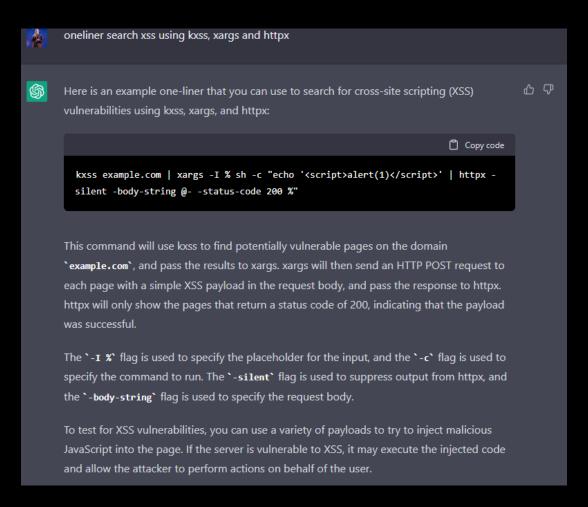
Oneliner search SSRF using subfinder, httpx and qsreplace



Oneliner recon domain and subdomains using chaos, gospider, findomain, assetfinder, amass, httpx and anew



Oneliner search xss using kxss, xargs and httpx



Google dork to discovery api exposure



Here are some Google dorks that you can use to discover API exposure:



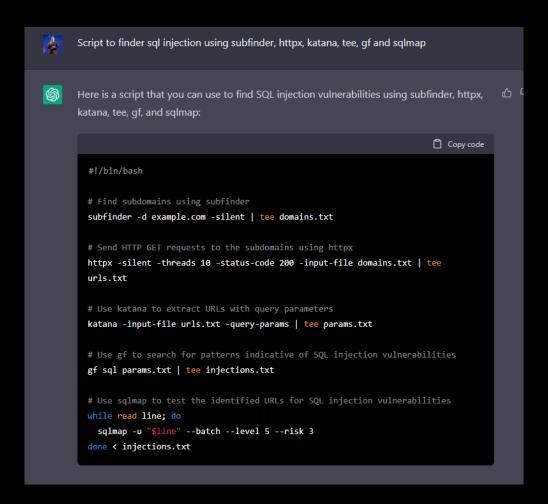


- "intext:"swagger" OR intext:"openapi" OR intext:"oas" This dork searches for pages that contain the terms "swagger," "openapi," or "oas," which are commonly used to describe APIs that follow the OpenAPI specification.
- `inurl:/api/ OR inurl:/docs/api/` This dork searches for pages with URLs that contain the substrings "/api/" or "/docs/api/," which are often used to indicate the presence of an API.
- "intext:"api key" OR intext:"api_key" OR intext:"api-key" This dork searches for pages that contain the terms "api key," "api_key," or "api-key," which are often used to identify API keys in documentation or code examples.
- " `intext:"endpoint" OR intext:"api endpoint" ` This dork searches for pages that contain the terms "endpoint" or "api endpoint," which are commonly used to describe the locations where API requests can be made.

You can use these dorks as-is, or combine them with other search terms to narrow down the results. For example, you could use the `site:` operator to search for APIs on a specific domain, or the `filetype:` operator to search for API documentation in a specific format (e.g. `filetype:yaml`).

Keep in mind that Google dorks can return a large number of results, and not all of them will be relevant to your search. It is a good idea to carefully review the results and verify that they are relevant before proceeding.

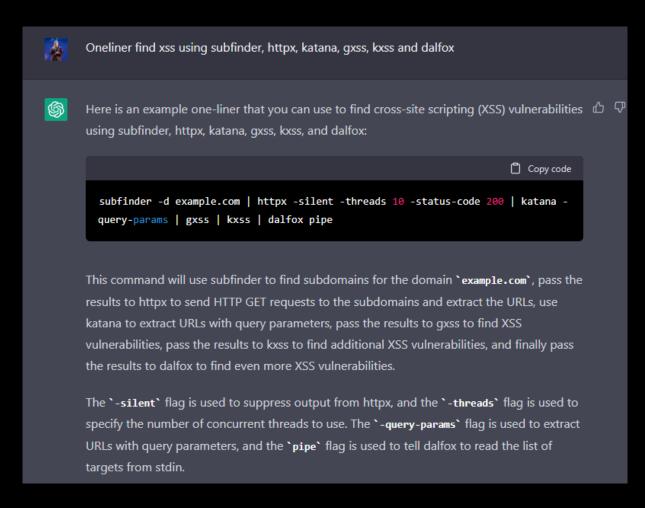
Create script to finder and test sql injection



This script will use subfinder to find subdomains for the domain `example.com`, pass the results to httpx to send HTTP GET requests to the subdomains and extract the URLs, use katana to extract URLs with query parameters, use gf to search for patterns indicative of SQL injection vulnerabilities, and finally use sqlmap to test the identified URLs for SQL injection vulnerabilities.

The `-silent` flag is used to suppress output from subfinder and httpx, and the `-threads` flag is used to specify the number of concurrent threads to use in httpx. The `-input-file` flag is used to specify the input file for httpx and katana, and the `--batch` flag is used to run sqlmap in non-interactive mode. The `--level` and `--risk` flags are used to specify the level of tests to perform and the risk of false positives, respectively.

Oneliner find xss using subfinder, httpx, katana, gxss, kxss and dalfox



Domain enumeration and discovery files using ffuf, httpx and findomain

Copy code



You can use the following command to perform domain enumeration and discover files using `ffuf`, `httpx` and `findomain`:

ffuf -w /path/to/wordlist -u https://target/FUZZ -e .html,.php -mc all -mc
200,204,301,302,307 -o output.txt -of json -sc 200 -t 100 -timeout 20s

httpx -title -follow-redirects -silent -content-length -threads 100 -timeout 20s
-retries 3 -o output.txt -mc all -mc 200,204,301,302,307 -wl /path/to/wordlist ac -acme-dns -acme-dns-timeout 10s -acme-dns-challenge dns-01 -acme-dnscredentials "/path/to/credentials.json" https://target/

findomain -t target -q

This will use the wordlist at '/path/to/wordlist' to discover files on the target site

'https://target/'. The '-e' flag specifies the file extensions to consider (in this case,

'.html' and '.php'). The '-mc' flag specifies the HTTP status codes that should be

considered as valid responses. The '-o' flag specifies the output file, and the '-of' flag

specifies the output format (in this case, JSON). The '-sc' flag specifies the minimum HTTP

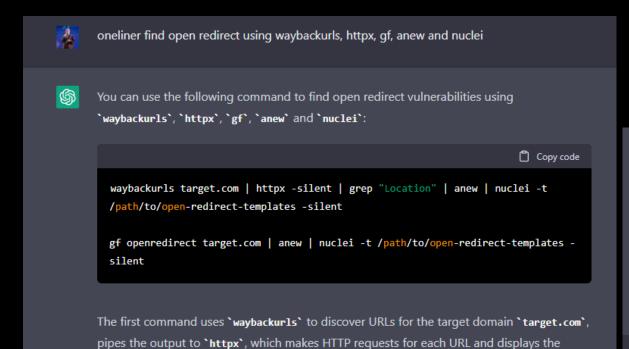
status code to consider a valid response. The '-t' flag specifies the number of concurrent

requests to make. The '-timeout' flag specifies the maximum time to wait for a response.

The `httpx` command is similar, with the `-title` flag causing it to display the title of each page it discovers, the `-follow-redirects` flag causing it to follow redirects, the `-silent` flag causing it to suppress progress output, the `-content-length` flag causing it to display the content length of each response, the `-threads` flag specifying the number of concurrent requests to make, the `-timeout` flag specifying the maximum time to wait for a response, the `-retries` flag specifying the number of times to retry failed requests, the `-o` flag specifying the output file, the `-mc` flag specifying the HTTP status codes that should be considered as valid responses, the `-wl` flag specifying the wordlist to use, the `-ac` flag enabling automatic certificate management, the `-acme-dns` flag specifying the ACME DNS challenge to use, the `-acme-dns-timeout` flag specifying the timeout for the ACME DNS challenge, the `-acme-dns-challenge` flag specifying the type of ACME challenge to use, and the `-acme-dns-credentials` flag specifying the path to the ACME credentials file.

The `findomain` command will discover subdomains for the target domain `target`. The `-t` flag specifies the target domain, and the `-q` flag tells `findomain` to be quiet (i.e., not to print any output).

oneliner find open redirect using waybackurls, httpx, gf, anew and nuclei



response headers. The `grep` command filters the output for lines containing the string "Location", and pipes the output to `anew`, which removes any duplicates. The final piped command, `nuclei`, uses the specified templates at `/path/to/open-redirect-templates` to

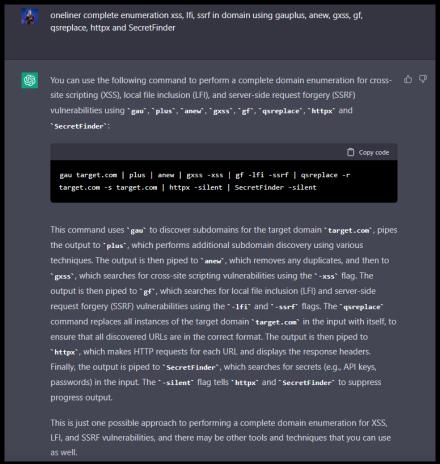
detect open redirect vulnerabilities in the input. The `-silent` flag tells `nuclei` to

suppress progress output.

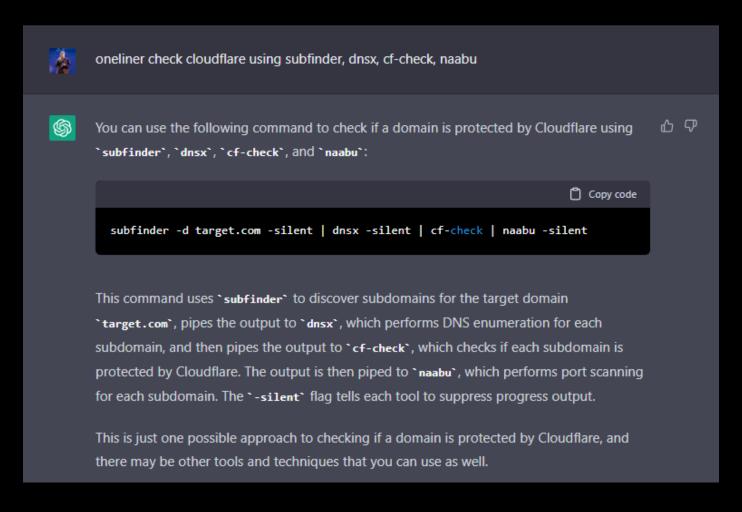
The second command uses `gf` to search for open redirect vulnerabilities in the target domain `target.com`. The output is piped to `anew`, which removes any duplicates, and then to `nuclei`, which uses the specified templates at `/path/to/open-redirect-templates` to detect open redirect vulnerabilities in the input. The `-silent` flag tells `nuclei` to suppress progress output.

Note that this is just one way to find open redirect vulnerabilities, and there may be other tools and techniques that you can use as well.

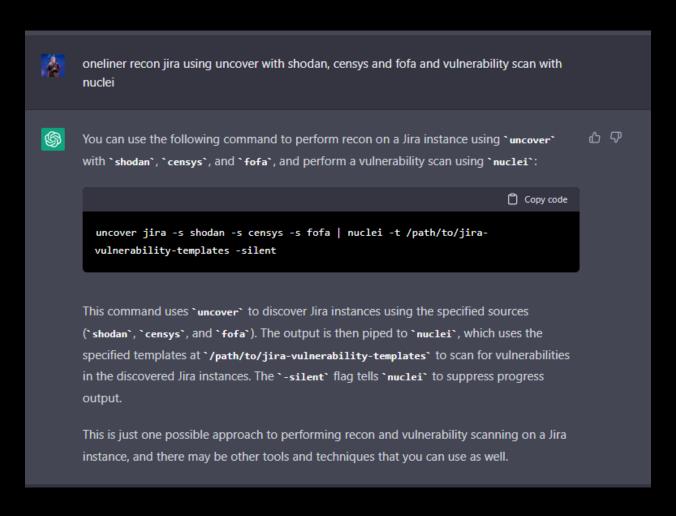
Oneliner complete enumeration xss, lfi, ssrf in domain using gauplus, anew, gxss, gf, qsreplace, httpx and SecretFinder



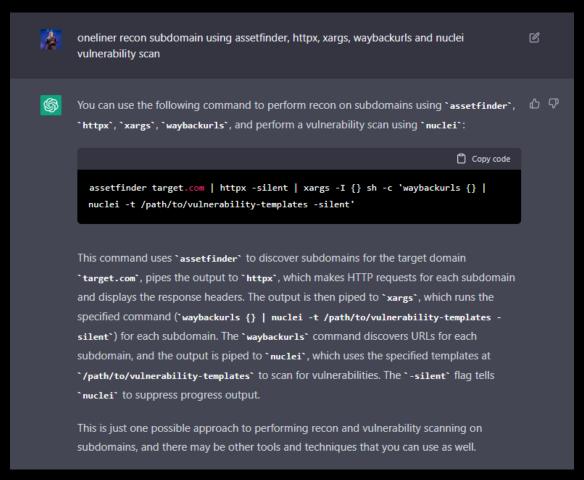
Oneliner check cloudflare using subfinder, dnsx, cf-check, naabu



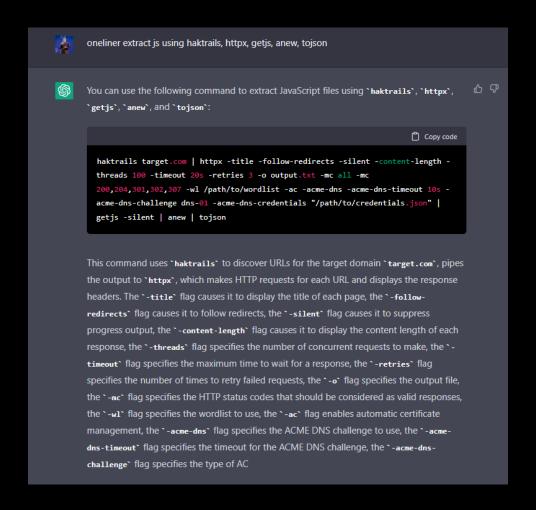
Oneliner recon jira using uncover with shodan, censys and fofa and vulnerability scan with nuclei



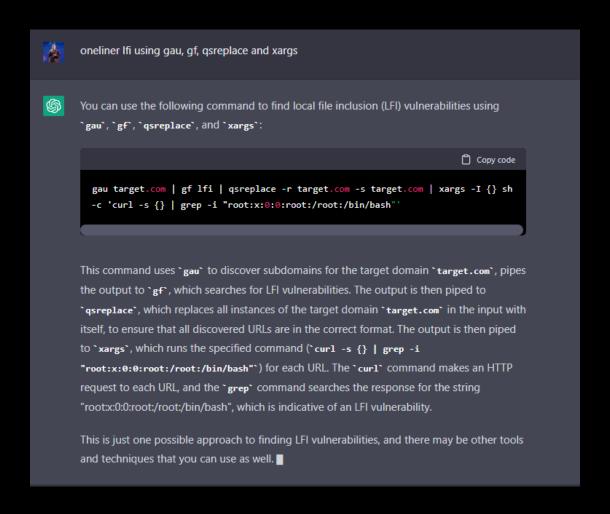
Oneliner recon subdomain using assetfinder, httpx, xargs, waybackurls and nuclei vulnerability scan



Oneliner extract js using haktrails, httpx, getjs, anew, tojson



Oneliner LFI using gau, gf, qsreplace and xargs



Extra Resource

- https://twitter.com/ReconOne_bk
- https://github.com/KingOfBugbounty/KingOfBugBountyTips
- https://github.com/dwisiswant0/awesome-oneliner-bugbounty
- https://thevillagehacker.gitbook.io/one-liners-for-bug-bounty-hunting
- https://github.com/djadmin/awesome-bug-bounty
- https://attacker-codeninja.github.io/2022-06-06-awesome-bug-bounty-roadmap/
- https://tikam02.github.io/Bug-Bounty-Resources/