Web Scanner Documentation

Overview

This Python script is a multi-purpose web scanner designed for security professionals and penetration testers. It provides several scanning capabilities including:

- Subdomain enumeration
- Directory brute-forcing
- Parameter fuzzing (GET/POST)

The tool is highly configurable with support for rate limiting, custom delays, status code filtering, and keyword matching in responses.

Features

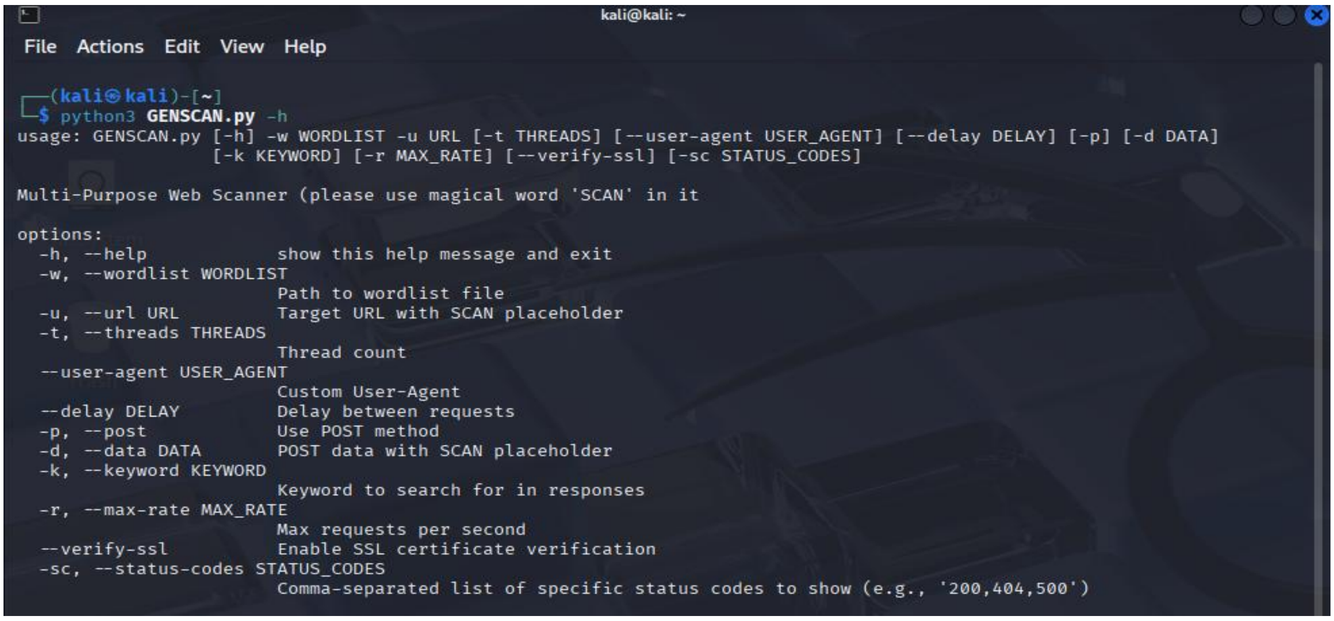
Core Functionality

- **Subdomain Enumeration**: Finds valid subdomains by replacing "SCAN" in the target URL
- **Directory Brute-Force**: Tests for existing directories by appending words to a base URL
- **Parameter Fuzzing**: Tests GET/POST parameters by replacing "SCAN" in requests
- **Keyword Matching**: Searches responses for specific content patterns
- **Status Code Filtering**: Only shows results matching specified HTTP status codes

Technical Features

- Multi-threaded for performance
- Configurable rate limiting
- Custom delays between requests
- SSL verification toggle
- Custom user agent support
- POST data handling
- Context-aware scanning modes

Usage



```
┌──(kali㉿kali)-[~]
└─$ python3 GENSCAN.py -h
usage: GENSCAN.py [-h] -w WORDLIST -u URL [-t THREADS] [--user-agent USER_AGENT] [--delay DELAY] [-p] [-d DATA]
                  [-k KEYWORD] [-r MAX_RATE] [--verify-ssl] [-sc STATUS_CODES]

Multi-Purpose Web Scanner (please use magical word 'SCAN' in it

options:
  -h, --help            show this help message and exit
  -w, --wordlist WORDLIST
                        Path to wordlist file
  -u, --url URL         Target URL with SCAN placeholder
  -t, --threads THREADS
                        Thread count
  --user-agent USER_AGENT
                        Custom User-Agent
  --delay DELAY         Delay between requests
  -p, --post            Use POST method
  -d, --data DATA       POST data with SCAN placeholder
  -k, --keyword KEYWORD
                        Keyword to search for in responses
  -r, --max-rate MAX_RATE
                        Max requests per second
  --verify-ssl          Enable SSL certificate verification
  -sc, --status-codes STATUS_CODES
                        Comma-separated list of specific status codes to show (e.g., '200,404,500')
```

Example of usage

1. For directory brute force

   Python3 GENSCAN -w wordlist.txt -u http://exmaple.com/SCAN (optional argument) --status-codes "301,403" --max-rate 5 --delay 0.2

2. For subdomain

   Python3 GENSCAN -w wordlist.txt -u http://SCAN.exmaple.com (optional argument) --status-codes "301,403" --max-rate 5 --delay 0.2

3. For parameter fuzzing

   Python3 GENSCAN -w wordlist.txt -u http://exmaple.com/q=SCAN (optional argument) --status-codes "301,403" --max-rate 5 --delay 0.2

   OR

   Python3 GENSCAN -w wordlist.txt -u http://exmaple.com/login.php -p -d "name=SCAN"&pass=pass" (optional argument) --status-codes "301,403" --max-rate 5 --delay 0.2

Required Arguments

-w, --wordlist - Path to wordlist file (required)

-u, --url - Target URL with "SCAN" placeholder (required)

Optional Arguments

-t, --threads	Number of threads to use	default -10
--user-agent	Custom User-Agent string	default -"Mozilla/5.0"
--delay	Delay between requests in seconds	0
-p, --post	Use POST method instead of GET	default -False
-d, --data	POST data with SCAN placeholder	default -None
-k, --keyword	Keyword to search for in responses	default -None
-r ,--max-rate	Maximum requests per second	default -10
--verify-ssl	Enable SSL certificate verification	Default -False
-sc, --status-codes	Comma-separated list of status codes to show	default -None (shows 200)

Implementation Details

Key Functions

1. **scan_post()** - Parses POST data strings into dictionaries
2. **scan_ratelimit()** - Implements rate limiting logic
3. **scan_keyword()** - Handles keyword searching in responses
4. **scan_subdomain()** - Detects subdomain scanning mode
5. **scan_target()** - Core scanning logic for all modes
6. **scan_worker()** - Thread worker function
7. **main_scan()** - Main function handling argument parsing and execution

Rate Limiting

- The script implements a token bucket-style rate limiter:
- Tracks timestamps of recent requests
- Calculates wait times when approaching max rate
- Ensures minimum delay between requests

Threading Model

- Uses Python's threading module
- Creates a queue of words to test
- Spawns worker threads to process the queue
- Implements thread-safe operations with locks

Error Handling

- Handles common network errors (timeouts, DNS failures)
- Provides descriptive error messages
- Gracefully handles keyboard interrupts

Security Considerations

- **SSL Verification** - Disabled by default for testing, but can be enabled with --verify-ssl
- **Rate Limiting** - Important to avoid overwhelming servers or triggering protections
- **Legal Compliance** - Only use on systems you have permission to test
- **Disclosure** - The tool includes a custom User-Agent that identifies it as a scanner

Limitations

- **Basic Authentication** - Doesn't currently support authenticated scans
- **Cookie Handling** - No built-in session management
- **JavaScript** - Doesn't execute or analyze JavaScript content
- **Recursion** - Doesn't automatically follow or scan discovered paths