

Custom Wordlist Generator (Python)

Project Overview

The **Custom Wordlist Generator** is a Python-based CLI tool designed to generate highly customizable password wordlists for **security testing, penetration testing, and SOC labs**.

It allows users to control **word length, character sets, base words, file size limits, and compression**, making it suitable for real-world password attack simulations.

```
PS D:\projects\wordlist_code> python .\wordlist_generator.py
🔧 Configure your custom wordlist generation

📦 Specify output wordlist size? (yes/no): yes
Enter size unit (MB/GB): GB
Enter the size in GB (e.g., 100): 2
🔪 Specify min/max word length? (yes/no): yes
Enter minimum word length: 3
Enter maximum word length: 3
🔤 Include specific letters? (yes/no): yes
Enter letters to use (e.g., abc...): ram
🔧 Include symbols? (yes/no): yes
Enter symbols to use (e.g., @$%): !@#
🔢 Include digits? (yes/no): yes
Enter digits to use (e.g., 0123456789): 123
🔪 Use base words in generation? (yes/no): yes
How many base words do you want to enter? 1
Enter base word 1: ra
🔧 Generating base word combinations...
✅ Wordlist generated: 12 words, 0.00 MB
🔧 Compressed size: 0.00 MB (gzip)
PS D:\projects\wordlist_code>
```

```
ra!
!ra
ra@
@ra
ra#
#ra
ra1
1ra
ra2
2ra
ra3
3ra
```

Problem It Solves

Traditional wordlists are either:

- Too large and inefficient, or
- Not customized for a specific target

This tool solves that by generating **targeted, size-controlled wordlists**, reducing attack time while increasing relevance.

Key Features

- Interactive CLI with validation
- Custom word length (min & max)
- Character set selection:
 - Letters
 - Digits
 - Symbols
- Base word mutation logic
- File size control (MB / GB)
- Duplicate prevention
- Automatic gzip compression

- Efficient memory handling (stream writing)
-

How It Works (High Level)

1. User configures generation options via CLI prompts
 2. Tool validates inputs to prevent invalid wordlists
 3. Wordlist is generated using:
 - Base word permutations OR
 - Brute-force character combinations
 4. Output is written incrementally to avoid memory overload
 5. Final wordlist is compressed using gzip
-

Input Options

Option	Description
Output Size	Limit wordlist size (MB / GB)
Word Length	Minimum & maximum length
Letters	Custom or default a-z
Digits	Custom digit sets
Symbols	Custom special characters
Base Words	User-defined keywords
Compression	Automatic gzip output

Word Generation Logic

1 Base Word Mutation

If base words are provided, the generator creates:

- Direct combinations
- Symbol-inserted variations
- Digit-based mutations

Examples

adminuser
admin@user
user@admin
admin123user
admin1user

2 Brute-Force Character Combinations

If no base words are provided:

- Uses `itertools.product`
 - Generates combinations within defined length
 - Stops automatically when target file size is reached
-

Validation & Safety

- Ensures valid input types
- Prevents empty character sets
- Avoids duplicate words

- Stops generation once size limit is reached
 - Supports graceful exit conditions
-

Output

- **Primary file:** custom_wordlist.txt
 - **Compressed file:** custom_wordlist.txt.gz
 - Displays:
 - Total words generated
 - Final file size
 - Compressed size
-

Technologies Used

- Python 3
 - itertools
 - string
 - gzip
 - shutil
 - os
-

How to Run

python wordlist_generator.py

Follow the interactive prompts to configure your wordlist.

Use Cases

- Password cracking (Hashcat / John the Ripper)
 - SOC & Blue Team labs
 - Red Team password audits
 - CTF challenges
 - Custom attack simulations
 - Security research
-

Performance Considerations

- Stream-based file writing (low memory usage)
 - Early stopping when size limit is reached
 - Optional compression for storage efficiency
-

Security & Ethical Note

This tool is intended **only for authorized security testing, educational purposes, and labs.**

Misuse against systems without permission is illegal.