

Lab 3 — Wordlist Generation

Install wordlist generation tools, create custom dictionaries, and run attacks with them.

Prerequisites: You need a working Hashcat installation (from Lab 1) and a terminal with root/sudo access for package installation. All commands assume a Linux or macOS environment.

Exercise 1 — Install Crunch

EX 1

Install the Crunch wordlist generator

Crunch is available in most package managers. Install it for your platform:

```
# Debian / Ubuntu / Kali
sudo apt-get update
sudo apt-get install crunch

# macOS (Homebrew)
brew install crunch

# Verify installation
crunch --help
```

If the installation succeeds, you should see Crunch's help output listing the available options.

Checkpoint: Running `crunch --help` displays usage information without errors.

Exercise 2 — Install CeWL

EX 2

Install the CeWL web wordlist generator

CeWL is a Ruby gem. You need Ruby installed on your system first:

```
# Check Ruby is available
ruby --version

# Install CeWL
gem install cewl

# On Kali Linux, CeWL is pre-installed
which cewl

# Verify
cewl --help
```

Note: If you encounter permission errors with `gem install`, try `sudo gem install cewl` or use a Ruby version manager like `rvm` or `rbenv`.

Checkpoint: Running `cewl --help` displays usage information.

Exercise 3 — Generate a Wordlist with Crunch

EX 3

Create custom wordlists using Crunch

Create a working directory and generate several wordlists:

```
# Create a lab directory
mkdir -p ~/hashcat_lab3
cd ~/hashcat_lab3

# Generate a small wordlist: 4-digit PINs
crunch 4 4 0123456789 -o pins.txt
wc -l pins.txt

# Expected: 10,000 lines
```

```
# Generate passwords 6-8 chars, lowercase only
crunch 6 8 abcdefghijklmnopqrstuvwxyz -o alpha_dict.txt
# WARNING: This will be very large! Cancel with Ctrl+C after a few seconds
# to see it working, then clean up

# More practical: use a pattern
# 4 lowercase letters + 4 digits (e.g., "pass1234")
crunch 8 8 -t @@@@% % -o pattern_dict.txt
wc -l pattern_dict.txt

# Generate passwords 8-10 chars with default charset
crunch 8 10 -o custom_dict.txt
# WARNING: extremely large output — cancel quickly or limit with -c flag

# Practical approach: limit line count
crunch 8 8 abcdefghijklmnopqrstuvwxyz0123456789 -c 100000 -o limited_dict.txt
```

Examine the output

```
# Check sizes
ls -lh *.txt

# Preview the first and last entries
head -5 pins.txt
tail -5 pins.txt

head -5 pattern_dict.txt
tail -5 pattern_dict.txt
```

Checkpoint: You have at least two Crunch-generated wordlists in your lab directory. `pins.txt` contains exactly 10,000 lines.

Exercise 4 — Generate a Wordlist with CeWL

EX 4

Spider a webpage and build a wordlist

Use CeWL to extract words from a public website. We will use example.com as a safe target:

```
cd ~/hashcat_lab3

# Basic CeWL run against example.com
cewl -w web_dict.txt http://example.com
wc -l web_dict.txt
cat web_dict.txt

# Try with a more content-rich page (Wikipedia)
cewl -m 6 -d 1 -w wiki_dict.txt https://en.wikipedia.org/wiki/Password
wc -l wiki_dict.txt
head -20 wiki_dict.txt

# Extract longer words (minimum 8 characters)
cewl -m 8 -d 2 -w long_words.txt https://en.wikipedia.org/wiki/Cryptograph
wc -l long_words.txt
```

Tip: In a real engagement, you would run CeWL against the target organization's website, blog, about page, and public documentation. The extracted words become high-value password candidates.

Checkpoint: `web_dict.txt` and `wiki_dict.txt` contain extracted words. The Wikipedia-sourced list should have dozens to hundreds of unique words.

Exercise 5 — Run Attacks with Custom Wordlists

EX 5

Use your custom wordlists with Hashcat

First, create some test hashes, then attack them with your wordlists:

```
cd ~/hashcat_lab3

# Create test hashes (MD5) from known passwords
echo -n "pass1234" | md5sum | awk '{print $1}' > test_hashes.txt
echo -n "word5678" | md5sum | awk '{print $1}' >> test_hashes.txt
echo -n "test0000" | md5sum | awk '{print $1}' >> test_hashes.txt
```

```
# On macOS, use md5 instead of md5sum:
# echo -n "pass1234" | md5 >> test_hashes.txt

# Attack with the pattern dictionary (no rules)
hashcat -m 0 test_hashes.txt pattern_dict.txt --potfile-disable

# Attack with CeWL wordlist + best64 rules
hashcat -m 0 test_hashes.txt wiki_dict.txt -r /usr/share/hashcat/rules/bes

# Attack with PIN dictionary
hashcat -m 0 test_hashes.txt pins.txt --potfile-disable

# Show cracked results
hashcat -m 0 test_hashes.txt --show
```

Combine wordlists and attack

```
# Merge all your wordlists
cat pins.txt pattern_dict.txt web_dict.txt wiki_dict.txt | sort -u > combined.txt
wc -l combined.txt

# Attack with combined wordlist + best64 rules
hashcat -m 0 test_hashes.txt combined.txt -r /usr/share/hashcat/rules/bes
```

Checkpoint: At least "pass1234" should be cracked by the pattern dictionary. The combined attack should crack additional hashes.

Exercise 6 — Analyze Results

EX 6

Compare wordlist effectiveness

Document your findings by answering these questions:

1. How many hashes did each wordlist crack individually (pins, pattern, CeWL, combined)?
2. What was the speed (hashes per second) for each attack?
3. How does adding rules (best64.rule) change the number of candidates and the results?

4. Which wordlist strategy was most effective for the test hashes you created?

Test rule impact

```
# See how many candidates a wordlist + rules generates
hashcat --stdout wiki_dict.txt -r /usr/share/hashcat/rules/best64.rule |

# Compare with the raw wordlist
wc -l wiki_dict.txt

# Try with d3ad0ne rules (much larger)
hashcat --stdout wiki_dict.txt -r /usr/share/hashcat/rules/d3ad0ne.rule
```

Expected observation: The best64 rule set multiplies your wordlist by 64x. A 1,000-word CeWL list becomes 64,000 candidates. The d3ad0ne rules multiply it by ~34,000x — turning 1,000 words into 34 million candidates.

Checkpoint: You have documented which wordlist/rule combination cracked the most hashes and understand the trade-offs between wordlist size, rules, and attack speed.

Clean up: Remove large generated wordlists when done to free disk space: `rm`

```
~/hashcat_lab3/alpha_dict.txt ~/hashcat_lab3/custom_dict.txt
```