

Homework — Wordlist Generation

Three tasks to deepen your understanding of wordlist creation and optimization.

Document your commands, outputs, and observations for each task. Submit a text file or screenshot of your terminal session.

TASK 1

Build a Comprehensive Attack Wordlist

Create a multi-source wordlist by combining at least three different sources:

1. **Crunch-generated:** Generate a pattern-based wordlist of 8-character passwords following the pattern: one uppercase letter, five lowercase letters, two digits (e.g., "Abcdef12"). Use Crunch's `-t` flag with the pattern `,@000000%0`.
2. **CeWL-sourced:** Run CeWL against two different public websites of your choice (news sites, Wikipedia articles, company pages). Use a minimum word length of 5 and a spider depth of 2.
3. **Manual additions:** Create a small file of 20-30 common password bases (e.g., "password", "admin", "welcome", "letmein", "qwerty", "dragon", "master", "monkey", "shadow").

Combine all sources, deduplicate, and report:

- How many unique words does each source contribute?
- What is the total combined wordlist size after deduplication?
- How much overlap exists between sources?

```
# Combine and analyze
cat crunch_out.txt cewl_site1.txt cewl_site2.txt manual.txt | sort -u > combined.txt
wc -l crunch_out.txt cewl_site1.txt cewl_site2.txt manual.txt combined.txt
```

TASK 2

Write and Benchmark Custom Hashcat Rules

Create a custom rule file called `my_rules.rule` with at least 10 rules targeting common password patterns:

```
# Example my_rules.rule content:
:                # passthrough (original word)
c                # capitalize first letter
c $1            # capitalize + append "1"
c $1 $2 $3      # capitalize + append "123"
$!              # append "!"
$1 $!           # append "1!"
sa@ se3 si1 so0 # leetspeak substitutions
c $2 $0 $2 $4   # capitalize + append "2024"
u               # uppercase all
^1 ^2 ^3        # prepend "321"
```

Benchmark against built-in rules

Create 50 MD5 hashes from known passwords, then compare crack rates:

1. Attack with your wordlist + **no rules**
2. Attack with your wordlist + **my_rules.rule**
3. Attack with your wordlist + **best64.rule**
4. Attack with your wordlist + **d3ad0ne.rule**

For each run, record: number of hashes cracked, time taken, and total candidates tested. Present results in a comparison table.

Tip: Use `hashcat --stdout wordlist.txt -r my_rules.rule | wc -l` to count candidates without actually attacking. This helps predict runtime.

TASK 3

CeWL Profiling and Targeted Attack

Choose a public-facing website (a company site, university, or open-source project) and perform a thorough CeWL analysis:

1. Run CeWL with varying parameters and compare the results:

```
# Minimum word length 4, depth 1
cewl -m 4 -d 1 -w short_shallow.txt http://target-site.com
wc -l short_shallow.txt

# Minimum word length 6, depth 3
cewl -m 6 -d 3 -w medium_deep.txt http://target-site.com
wc -l medium_deep.txt

# Minimum word length 8, depth 2, with emails
cewl -m 8 -d 2 -e --email_file found_emails.txt -w long_medium.txt http://target-site.com
wc -l long_medium.txt found_emails.txt
```

2. Analyze the extracted words:

- What types of words did CeWL find (technical terms, names, products)?
- Which words would make plausible password bases?
- Were any email addresses or metadata found?

3. Create a targeted attack plan:

- Select the best CeWL output and apply it with `best64.rule`
- Create 10 test hashes based on words you expect CeWL would find on the target site (e.g., company name + digits)
- Run the attack and verify your targeted wordlist cracks them

Write a short report (half a page) summarizing: how many useful password candidates CeWL extracted, which parameters gave the best results, and how the targeted approach compares to using a generic wordlist like RockYou.