

Lab Guide — GPU Acceleration & Performance

Six hands-on exercises testing GPU detection, attack modes, workload tuning, thermals, and CPU vs GPU comparison.

Prerequisites: A system with a supported GPU (NVIDIA or AMD), up-to-date GPU drivers, and Hashcat installed. Prepare a test hash file with several MD5 hashes and have the `rockyou.txt` wordlist available.

Prepare test hashes: Create a file called `hashes.txt` with known MD5 hashes for testing. For example:

```
# Generate test hashes
echo -n "password" | md5sum | cut -d' ' -f1 > hashes.txt
echo -n "123456" | md5sum | cut -d' ' -f1 >> hashes.txt
echo -n "qwerty" | md5sum | cut -d' ' -f1 >> hashes.txt
echo -n "letmein" | md5sum | cut -d' ' -f1 >> hashes.txt
echo -n "welcome" | md5sum | cut -d' ' -f1 >> hashes.txt
```

Exercise 1 — Check GPU Availability

EX 1

Detect compute devices with Hashcat and system tools

Verify that your GPU is properly detected before running any attacks.

1. Run `hashcat -I` and note all detected devices, their IDs, types, and memory
2. For NVIDIA: run `nvidia-smi` and record the GPU name, driver version, CUDA version, and current temperature

3. For AMD or general OpenCL: run `clinfo` and note the platform and device details
4. Run `hashcat -b -m 0` to benchmark MD5 on your system and record the speed

```
# Step 1: Hashcat device info
hashcat -I

# Step 2: NVIDIA system info
nvidia-smi

# Step 3: OpenCL info (all vendors)
clinfo

# Step 4: Quick MD5 benchmark
hashcat -b -m 0
```

Checkpoint: You can identify your GPU by name and device ID. The benchmark completes without errors.

Exercise 2 — GPU Dictionary Attack

EX 2

Run a dictionary attack using the GPU

Crack the test hashes using `rockyou.txt` on the GPU.

1. Run the attack with GPU device type:

```
# Dictionary attack on GPU
hashcat -m 0 -a 0 -D 2 -o cracked_gpu.txt hashes.txt rockyou.txt
```

2. Observe the output: note the speed (H/s), progress, and time to completion
3. Check the results file:

```
# View cracked passwords
cat cracked_gpu.txt
```

Checkpoint: All test hashes are cracked. The speed shows millions or billions of H/s (depending on your GPU).

Exercise 3 — GPU Mask Attack

EX 3

Run a brute-force mask attack on the GPU

Use a mask to brute-force short passwords on the GPU.

1. Create a hash of a short known password (e.g., "test1"):

```
# Create a test hash for mask attack
echo -n "test1" | md5sum | cut -d' ' -f1 > mask_test.txt
```

2. Run a mask attack targeting 5-character lowercase+digit passwords:

```
# Mask attack: 4 lowercase letters + 1 digit
hashcat -m 0 -a 3 -D 2 -o cracked_mask.txt mask_test.txt ?l?l?l?l?d
```

3. Note the speed, keyspace size, and estimated time to exhaustion
4. Verify the result:

```
cat cracked_mask.txt
```

Checkpoint: The mask attack finds "test1". You can explain the keyspace size ($26^4 * 10 = 4,569,760$ candidates).

Exercise 4 — Workload Profile Comparison

EX 4

Compare speeds across all four workload profiles

Test each workload profile and record the hash rate.

1. Run the benchmark for each profile and record the speed:

```
# Test workload profile 1 (Low)
hashcat -b -m 0 -D 2 -w 1

# Test workload profile 2 (Medium)
hashcat -b -m 0 -D 2 -w 2

# Test workload profile 3 (High)
hashcat -b -m 0 -D 2 -w 3

# Test workload profile 4 (Maximum)
hashcat -b -m 0 -D 2 -w 4
```

2. Record the results in a comparison table:

Profile	Speed (H/s)	System Responsiveness
-----	-----	-----
-w 1	_____	_____
-w 2	_____	_____
-w 3	_____	_____
-w 4	_____	_____

3. Note the speed increase percentage between profiles

Checkpoint: You have a completed comparison table. Speed increases with each profile level.

Exercise 5 — Temperature Limits

EX 5

Set thermal safeguards and observe throttling behaviour

Configure temperature limits and monitor GPU thermals during an attack.

1. Open a second terminal and start monitoring GPU temperature:

```
# NVIDIA: refresh every 1 second
nvidia-smi -l 1
```

```
# Linux alternative (AMD/NVIDIA)
watch -n1 sensors
```

2. In the first terminal, run an attack with thermal limits:

```
# Attack with temperature controls
hashcat -m 0 -a 3 -D 2 -w 3 \
  --gpu-temp-abort=85 \
  --gpu-temp-retain=75 \
  hashes.txt ?a?a?a?a?a?a
```

3. Observe the Hashcat status output — look for temperature readings and any throttling messages

4. Press `s` during the attack to see the current status including temperature

5. Press `q` to quit when you have observed enough data

Checkpoint: Temperature stays at or below the retain value. You can see Hashcat adjusting speed to manage thermals.

Exercise 6 — CPU vs GPU Performance

EX 6

Compare CPU and GPU cracking speeds

Run the same attack on CPU and GPU to quantify the performance difference.

1. Benchmark MD5 on CPU only:

```
# CPU benchmark
hashcat -b -m 0 -D 1
```

2. Benchmark MD5 on GPU only:

```
# GPU benchmark
hashcat -b -m 0 -D 2
```

3. Run the same dictionary attack on both and compare wall-clock time:

```
# CPU dictionary attack (time it)
time hashcat -m 0 -a 0 -D 1 -o cracked_cpu.txt hashes.txt rockyou.txt

# GPU dictionary attack (time it)
time hashcat -m 0 -a 0 -D 2 -o cracked_gpu2.txt hashes.txt rockyou.txt
```

4. Calculate the speedup factor: GPU speed / CPU speed

5. Repeat the comparison for a slower hash type like SHA-256 (`-m 1400`) or
bcrypt (`-m 3200`)

```
# Benchmark SHA-256 on CPU vs GPU
hashcat -b -m 1400 -D 1
hashcat -b -m 1400 -D 2
```

Checkpoint: You can state the GPU speedup factor for at least two hash types.
GPU is significantly faster for both.

Tip: Clean up temporary output files between exercises to avoid Hashcat's "all hashes already cracked" message. Use `--potfile-disable` or remove the potfile between runs: `rm -f ~/.local/share/hashcat/hashcat.potfile`