

Homework

Complete these tasks before Lesson 03. Focus on understanding your hardware capabilities and establishing a reliable working environment.

Task 1: Full Benchmark Report

Goal: Understand how cracking speed varies dramatically across different hash algorithms and learn to read benchmark output.

Instructions

1. Run a full benchmark across all hash types:

```
hashcat -b
```

This may take 10–20 minutes depending on your hardware.

2. From the output, identify and record:
 - The **3 fastest** hash types (highest speed in H/s, MH/s, or GH/s)
 - The **3 slowest** hash types (lowest speed)
3. Calculate the ratio between the fastest and slowest hash type. How many orders of magnitude separate them?
4. Present your findings in a table:

Deliverable

Category	Hash Type	Mode Number	Speed
Fastest #1	(your result)	...	...
Fastest #2	(your result)	...	...
Fastest #3	(your result)	...	...
Slowest #1	(your result)	...	...

Category	Hash Type	Mode Number	Speed
Slowest #2	(your result)	...	...
Slowest #3	(your result)	...	...

Plus a one-sentence answer: "The fastest hash type is approximately ____x faster than the slowest."

Self-check

- ✓ You ran the full benchmark to completion.
- ✓ You correctly identified 3 fastest and 3 slowest hash types.
- ✓ You calculated the speed ratio and it makes sense (typically 10,000x to 1,000,000x difference).
- ✓ You can explain why some hashes are intentionally slow (key derivation functions).

Task 2: Environment Verification

Goal: Ensure your hashcat environment is properly configured and will persist across sessions.

Instructions

1. Add the following environment variables to your shell configuration file

(`~/.bashrc`, `~/.zshrc`, or Windows environment variables):

```
export HASHCAT_HOME="/path/to/hashcat"
export HASHCAT_LOG="$HOME/hashcat-workspace/logs"
export HASHCAT_WORKLOAD_PROFILE=3
```

2. Close your terminal completely and open a new one.
3. Verify each variable is set:

```
echo "HASHCAT_HOME=$HASHCAT_HOME"
echo "HASHCAT_LOG=$HASHCAT_LOG"
echo "HASHCAT_WORKLOAD_PROFILE=$HASHCAT_WORKLOAD_PROFILE"
```

4. Verify the workspace directory structure exists:

```
ls -la ~/hashcat-workspace/
```

5. Create a simple test script that checks your setup:

```
#!/bin/bash
echo "=== Hashcat Environment Check ==="
echo "Version: $(hashcat --version)"
echo "HASHCAT_HOME: ${HASHCAT_HOME:-NOT SET}"
echo "HASHCAT_LOG: ${HASHCAT_LOG:-NOT SET}"
echo "HASHCAT_WORKLOAD_PROFILE: ${HASHCAT_WORKLOAD_PROFILE:-NOT SET}"
echo ""
echo "=== Workspace Directories ==="
for dir in hashes wordlists rules masks potfiles sessions logs; do
    if [ -d "$HOME/hashcat-workspace/$dir" ]; then
        echo "    [OK] $dir"
    else
        echo "    [MISSING] $dir"
    fi
done
echo ""
echo "=== Compute Devices ==="
hashcat -I 2>/dev/null | grep -E "(Name\\.\\.\\.|Type\\.\\.\\.|Memory\\.Total)"
```

6. Run the script and confirm all checks pass.

Deliverable

The output of your environment check script showing all variables set and all directories present.

Self-check

- ✓ All three environment variables are set after opening a new terminal.
- ✓ All workspace directories exist (hashes, wordlists, rules, masks, potfiles, sessions, logs).
- ✓ The check script runs without errors.
- ✓ At least one compute device is detected by `hashcat -I`.

Task 3: Why Are Some Hashes Slow?

Goal: Understand the security implications of hash algorithm speed from both the attacker's and defender's perspective.

Instructions

In your benchmark results, you observed that some hash types (MD5, NTLM) are cracked at billions of hashes per second, while others (sha512crypt, bcrypt, Argon2) are cracked at only thousands or hundreds of hashes per second. This is not a hashcat limitation — it is by design.

Research and write a short explanation (200–300 words) that addresses:

1. **Purpose:** Why were fast hashes like MD5 and SHA-1 created? What were they designed for?
2. **Problem:** Why is a fast hash function a poor choice for storing passwords?
3. **Solution:** How do key derivation functions (bcrypt, scrypt, Argon2, sha512crypt) solve this problem? What specific techniques do they use (iteration count, memory hardness, etc.)?
4. **Attacker perspective:** As someone learning hashcat, why does this distinction matter for your work?

Deliverable

A written explanation (200–300 words) that a fellow security student could understand.

Self-check

- ✓ You correctly distinguish between general-purpose hash functions and key derivation functions.
- ✓ You mention at least two specific techniques that slow hashes use (e.g., iteration count, memory hardness, salting).
- ✓ You connect the concept to practical hashcat usage (time estimates, attack feasibility).
- ✓ Your explanation is clear enough for a non-expert to follow.

Bonus Challenge

Bonus: Multi-Device Comparison

Goal: Compare cracking performance across different compute devices on your system.

Instructions

If your system has both a CPU and a GPU (check with `hashcat -I`):

1. Run an MD5 benchmark using only the GPU:

```
hashcat -b -m 0 -d 1
```

2. Run the same benchmark using only the CPU:

```
hashcat -b -m 0 -d 2
```

(Device IDs may differ on your system — check `hashcat -I` for the correct numbers.)

3. Calculate the GPU-to-CPU speed ratio for MD5.
4. Repeat for sha512crypt (`-m 1800`). Is the ratio different? Why?

Self-check

- ✓ You benchmarked both devices individually.
- ✓ You calculated the speed ratio for at least two hash types.
- ✓ You can explain why the GPU advantage varies across hash types.

Submission

There is no formal submission for this course. These tasks are for your own development. However:

- Save your benchmark results — you will reference them in later lessons when estimating crack times.

- Keep your environment check script — it is useful for quickly verifying your setup on new machines.
- Your Task 3 explanation will be relevant again in Lesson 04 (Hash Types & Identification).

*Next lesson: **Lesson 03 — Command Structure & Parameters***