

Lab Guide

Time estimate: 30–45 minutes · Prerequisites: Terminal/command-line access, internet connection for downloads

Setup

This lab walks you through installing hashcat, verifying the installation, detecting your hardware, and running your first benchmark. Follow the steps for your operating system.

Before You Begin

Ensure your GPU drivers are up to date. Run `nvidia-smi` (NVIDIA) or `rocm-smi` (AMD) to verify. If you do not have a discrete GPU, hashcat will use CPU-only mode — this is fine for learning, but cracking speeds will be significantly lower.

Exercise 1: Install Hashcat

Goal

Install hashcat on your system and verify it runs correctly.

Steps — macOS

1. Open Terminal.
2. Install hashcat using Homebrew:

```
brew install hashcat
```

3. Verify the installation:

```
hashcat --version
```

4. Confirm the binary location:

```
which hashcat
```

Steps — Linux (Debian/Ubuntu)

1. Update your package list and install:

```
sudo apt update  
sudo apt install hashcat
```

2. Verify:

```
hashcat --version
```

Steps — Windows

1. Download the latest release from hashcat.net/hashcat/.
2. Extract the archive to `C:\Tools\hashcat` (or your preferred location).
3. Open PowerShell and navigate to the directory:

```
cd C:\Tools\hashcat  
.\hashcat.exe --version
```

Expected Output

```
$ hashcat --version  
v6.2.6
```

(Version number may vary — any version 6.x or later is fine.)

Troubleshooting

- If `hashcat: command not found` — ensure the installation directory is in your PATH.
- On macOS, if Homebrew is not installed, install it first: `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

- On Linux, if the package is outdated, consider building from source (see Theory page).

Exercise 2: Device Detection

Goal

Identify all compute devices available to hashcat on your system.

Steps

1. Run the device information command:

```
hashcat -I
```

2. Read the output carefully and note:
 - How many compute platforms were detected?
 - What type is each device (CPU or GPU)?
 - What is the device name and VRAM (for GPUs)?
 - What driver version is reported?
3. Record this information — you will need it for the homework.

Expected Output (Example)

```
$ hashcat -I

hashcat (v6.2.6) starting in backend information mode

OpenCL Info:
=====

OpenCL Platform ID #1
  Vendor...: NVIDIA Corporation
  Name.....: NVIDIA CUDA
  Version..: OpenCL 3.0 CUDA 12.2.138

Backend Device ID #1
  Type.....: GPU
```

```
Vendor.ID.....: 32
Vendor.....: NVIDIA Corporation
Name.....: NVIDIA GeForce RTX 3080
Version.....: OpenCL 3.0 CUDA
Processor(s)....: 68
Clock.....: 1710 MHz
Memory.Total....: 10240 MB (10 GB)
Memory.Free.....: 9856 MB
Local.Memory....: 48 KB
OpenCL.Version..: OpenCL C 1.2
Driver.Version..: 535.129.03
```

CPU-Only Systems

If you only see a CPU device, that is normal for systems without a discrete GPU. You can still complete all exercises. Performance will be lower, but the learning experience is the same. Use the `--force` flag if hashcat warns about running on CPU-only.

Exercise 3: Run Your First Benchmark

Goal

Measure your system's hash cracking speed across common hash types and establish a performance baseline.

Steps

1. Run a targeted benchmark on MD5 (hash mode 0):

```
hashcat -b -m 0
```

2. Run a benchmark on NTLM (hash mode 1000):

```
hashcat -b -m 1000
```

3. Run a benchmark on sha512crypt (hash mode 1800):

```
hashcat -b -m 1800
```

4. Record the speed for each hash type in a table:

Hash Type	Mode	Speed
MD5	0	(your result)
NTLM	1000	(your result)
sha512crypt	1800	(your result)

5. Calculate the ratio: how many times faster is MD5 compared to sha512crypt?

Key Observations to Note

- MD5 and NTLM are very fast because they are simple, single-iteration hash functions.
- sha512crypt is orders of magnitude slower because it uses 5000 rounds by default — this is by design to resist brute-force attacks.
- The speed difference illustrates why modern systems use slow hash functions (bcrypt, scrypt, Argon2) for password storage.

Exercise 4: Environment Setup

Goal

Configure your hashcat working environment with a proper directory structure and environment variables.

Steps

1. Create a hashcat workspace directory:

```
mkdir -p ~/hashcat-workspace/{hashes,wordlists,rules,masks,potfiles,session}
```

2. Verify the structure:

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

3. Add environment variables to your shell configuration. For Bash:

```
cat >> ~/.bashrc << 'EOF'

# Hashcat environment
export HASHCAT_HOME="$(which hashcat | xargs dirname)"
export HASHCAT_LOG="$HOME/hashcat-workspace/logs"
export HASHCAT_WORKLOAD_PROFILE=3
EOF

source ~/.bashrc
```

For Zsh (macOS default):

```
cat >> ~/.zshrc << 'EOF'

# Hashcat environment
export HASHCAT_HOME="$(which hashcat | xargs dirname)"
export HASHCAT_LOG="$HOME/hashcat-workspace/logs"
export HASHCAT_WORKLOAD_PROFILE=3
EOF

source ~/.zshrc
```

4. Verify the variables are set:

```
echo $HASHCAT_HOME
echo $HASHCAT_LOG
echo $HASHCAT_WORKLOAD_PROFILE
```

Exercise 5: Test Crack

Goal

Perform a simple test crack to confirm that your entire pipeline works end to end.

Steps

1. Create a test hash file with a known MD5 hash:

```
# MD5 hash of "password123"
echo "482c811da5d5b4bc6d497ffa98491e38" > ~/hashcat-workspace/hashes/test.h
```

2. Create a small test wordlist:

```
cat > ~/hashcat-workspace/wordlists/test.dict << 'EOF'
admin
letmein
password
password123
qwerty
123456
EOF
```

3. Run hashcat in dictionary attack mode:

```
hashcat -m 0 -a 0 ~/hashcat-workspace/hashes/test.hash ~/hashcat-workspace/
```

4. Verify the result:

```
hashcat -m 0 ~/hashcat-workspace/hashes/test.hash --show --potfile-path ~/r
```

Expected Output

```
482c811da5d5b4bc6d497ffa98491e38:password123

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 482c811da5d5b4bc6d497ffa98491e38
Time.Started.....: ...
Time.Estimated....: ...
Recovered.....: 1/1 (100.00%) Digests
```

Finishing Up

Once you have completed all five exercises:

1. Confirm hashcat is installed and reports a version number.
2. Confirm `hashcat -I` shows at least one compute device.
3. Confirm benchmarks ran successfully and you recorded the speeds.
4. Confirm your workspace directory exists with all subdirectories.
5. Confirm the test crack found the correct password.

Self-Check

- ✓ `hashcat --version` returns a version number.
- ✓ `hashcat -I` lists at least one backend device (CPU or GPU).
- ✓ You can explain why MD5 benchmarks are orders of magnitude faster than sha512crypt.
- ✓ Environment variables are set and persist after opening a new terminal.
- ✓ The test crack successfully recovered `password123` from the MD5 hash.