

Homework — Security Fundamentals

Estimated time: ~50 min · No code · A browser, your real email address, and a willingness to look at uncomfortable truths

Three tool-tour exercises. You'll learn four things by doing them, in this order: (1) your data is almost certainly already in a breach somewhere — this is normal, and the question is what to do about it; (2) a password manager plus **MFA** on one important account takes ten minutes and removes most of the risk; (3) what your home network's firewall exposes to the internet — connecting the theory page's "network-layer defense" section to something you can see; (4) how the secrets-management problem looks in practice, through GitHub's automated scanning tools.

Before you start

Keep a one-page notebook open (paper or a text file). For each task, write the short answers under (a)/(b)/(c). You are not turning this in. Nothing here requires you to install software or run a command — every task is a few clicks in a browser on sites that are free to everyone.

The three tasks

Task 1 — Check if your email is in a breach

Open haveibeenpwned.com (HIBP — the industry-standard breach lookup service run by Troy Hunt). Enter your most-used email address in the search box. Read the results.

If you've used the internet for more than ~3 years, expect *multiple* hits — this is normal, not alarming on its own. It just means a site you once signed up for got breached at some point and your email (and sometimes more) ended up in a public dump.

Now click into one of the named breaches and read the "**Compromised data**" list. This is the most useful part: it tells you specifically what about you leaked from that one site — just an email and password hash? Or also phone numbers, addresses, dates of birth, security questions?

Self-check — answer all three:

- **(a)** How many breaches does your email appear in?
- **(b)** What's the date of the most recent breach in the list?
- **(c)** For that most-recent breach, name two or three classes of data that were exposed (e.g. passwords, phone numbers, physical addresses).

Bonus: click **Notify me** and subscribe your address to future-breach alerts — you'll get an email the next time your address shows up in a new dump.

✓ Take a screenshot of the result page for your own notes. (You can redact your email if you'd like to keep it private.)

Task 2 — Set up (or audit) a password manager + MFA on one account

If you don't already have a password manager, sign up for a free tier of any of these:

- **Bitwarden** — free, open source (bitwarden.com)
- **1Password** — free trial (1password.com)
- **Proton Pass** — free (proton.me/pass)
- Your browser's built-in — **Apple iCloud Keychain**, Chrome / Edge / Firefox built-in password manager

Install the browser extension if your choice has one. Then pick *one* important account — your primary email, your bank, or your main work account — and do these three things:

1. Check whether its current password is unique to that account (the password manager will usually flag reused passwords for you).
2. If it isn't unique, change it to a fresh password generated by the password manager.

3. Turn on **MFA** (multi-factor auth, sometimes called 2FA / two-factor authentication). Look in *Account Settings* → *Security* → *Two-Factor Authentication*.

Which MFA method to choose?

Strongly prefer an **authenticator app** (Google Authenticator, Microsoft Authenticator, Authy, 1Password / Bitwarden's built-in TOTP) over SMS. SMS-based MFA is much better than nothing, but is vulnerable to SIM-swap attacks. If the account offers a **hardware key** option (YubiKey, Google Titan), that's the strongest choice of all.

Self-check — answer all three:

- **(a)** Which password manager did you choose, and one sentence on why?
- **(b)** Which MFA method did you turn on (SMS, authenticator app, hardware key)? If you picked SMS, write one sentence on what you'd need to switch to an authenticator app.
- **(c)** Take a screenshot of the account's security settings page showing that MFA is now *on*.

✓ **Ten minutes of work on one important account removes most credential-stuffing risk for that account. The lesson from Attack 2 in the theory page is: do this for your email *first*, because your email is the password-reset path for everything else.**

Task 3 — Tour network and compute security in the real world

Two quick browser visits that connect what you just read (Themes A and B) to tools real security teams use.

Part A — ShieldsUP! (network-layer exposure)

Open grc.com/shieldsup (Gibson Research Corporation's ShieldsUP! tool — free, no account required). Click "**Proceed**", then run the "**All Service Ports**" scan. The tool probes your public IP address from the outside to see which ports are open, closed, or in "stealth" mode.

What you're seeing is exactly what an attacker running an internet-wide port scanner sees when they reach your router or firewall. Each *open* port is a service the internet can try to talk to. Each *stealth* result means the firewall is silently dropping probes rather than replying at all — the preferred configuration, because it reveals less to scanners.

Self-check — answer all three:

- **(a)** How many ports are shown as *open* vs *stealth*? (Write down the rough split.)
- **(b)** Were any ports *open*? If so, look up one of them on speedguide.net/port.php?port=XXXX (replace XXXX with the port number) and write one sentence on what service typically uses that port.
- **(c)** In one or two sentences, describe which defense concept from the "Security in the network layer" section of the theory page this scan is demonstrating.

✓ Most home routers put everything in stealth mode by default — that's the firewall doing its job. If you see open ports you didn't expect, write them down; it's worth investigating what's listening on them.

Part B — GitHub's secrets-scanning alert page (compute/secrets management)

Open docs.github.com/en/code-security/secret-scanning/about-secret-scanning. Read the first two sections (roughly 5 minutes). GitHub scans every public repository push for recognized credential patterns — API keys, OAuth tokens, private keys for dozens of providers — and alerts the repository owner (and sometimes the affected service provider directly) when one is found.

This is the automated response to the problem described in the "Secrets management" part of the theory page: developers accidentally commit credentials to code and push them to version control, where automated bots (both benign like GitHub's scanner and malicious) find them almost immediately.

Self-check — answer all three:

- **(a)** According to the page, what happens when a supported secret pattern is detected in a public repository?

- **(b)** What is "push protection," and in one sentence, why does it matter more than after-the-fact alerts?
- **(c)** Write one or two sentences connecting what you read to the "Secrets management" section of today's theory page — specifically, why a secrets manager (Vault, AWS Secrets Manager) solves the problem at the source rather than at detection.

✓ You don't need a GitHub account to read the documentation page. The takeaway: even the platforms we use to store code have had to build detective controls because committing secrets is genuinely easy to do by accident.

What "done" looks like

You're done when, in your notebook, you have:

- a breach count, a most-recent breach date, and one screenshot from Task 1 (HIBP)
- a password manager chosen, an MFA method chosen, and one screenshot of MFA enabled from Task 2
- a port-scan summary + one open-port service identified + one connecting sentence from Task 3 Part A (ShieldsUP!)
- three answers about GitHub secret scanning from Task 3 Part B

The aim was never to make you a security engineer in 45 minutes. It was to do the two things that actually move the needle for almost everyone — unique passwords and MFA on the accounts that matter — and to leave you with a concrete picture of how network and compute-layer defenses look from the outside.

Where this goes next

Lesson 11 closes Module 3 (How Software Runs & Stays Safe). Module 4 opens the AI era — Lesson 12 covers AI & ML Foundations, and Lesson 13 covers the LLM & agent era. Security comes back as a cross-cutting concern there too:

prompt injection, model abuse, and AI-augmented social engineering are all live topics as of 2026.