Cybersecurity for Beginners

CS50's Introduction to Cybersecurity — Complete Course Notes

SECTION 1: Introduction — The Cybersecurity Mindset

1.1 What This Course Covers

This course — CS50's Introduction to Cybersecurity — is designed for both technical and non-technical learners. It builds the foundational knowledge needed to protect your accounts, data, systems, and software against today's threats, and equips you to reason about tomorrow's threats even as technology evolves.

- Securing accounts — passwords, multi-factor authentication, passkeys
- Securing data — hashing, encryption, encryption in transit, end-to-end encryption
- Securing systems — networks, ports, firewalls, penetration testing, cookies, DNS
- Securing software — web vulnerabilities (XSS, CSRF), buffer overflows, code injection
- Preserving privacy — VPNs, Tor, DNS-over-HTTPS, app permissions, location data

1.2 The Core Cybersecurity Mindset

The most important insight in this course is that cybersecurity is NOT about achieving perfection — it is about managing relative risk. Security is always a trade-off with usability.

- The defender must secure EVERY door and window. The adversary only needs to find ONE mistake.
- Think in terms of: risk vs. reward for the adversary; cost vs. benefit for you
- The goal is not absolute protection but raising the cost and risk to the adversary until they lose interest in you as a target
- Prevention is important but so is detection — monitoring and auditing mean you can identify breaches quickly and minimize damage
- AI tools are increasingly used to detect patterns of adversarial behavior that humans might miss
- If you raise the bar high enough, many adversaries will simply move on to easier targets

Key principle: Security is a trade-off. More security often means less convenience. The right balance depends on context — personal use vs. corporate policy.

1.3 Key Vocabulary

Term	Definition
Authentication	Proving who you are (e.g. entering a password to prove you are you)
Authorization	Determining what you are allowed to access once your identity is proven
Adversary / Hacker	Someone attempting to gain unauthorized access to systems or data
Threat	A potential attack or vulnerability that could be exploited
Defense	A measure taken to prevent, detect, or recover from an attack
Usability	How easy and convenient a system is to use — often in tension with security
Credential	A username and/or password combination used to authenticate
Encryption	Scrambling data so only authorized parties can read it
Hash	A one-way mathematical transformation of data into a fixed-length output
Penetration Testing	Ethical hacking — intentionally attempting to break into systems to find vulnerabilities

SECTION 2: Securing Accounts — Passwords

2.1 How Passwords Work

Passwords are the most common authentication mechanism. A username (often an email address) is public — it identifies you. A password is private — it proves you are who you claim to be. The assumption is that only you know both your username AND your password.

- Username: public identifier — your email or a chosen name
- Password: private secret — must be known only to you
- Authentication: providing both username and password to prove your identity to a system

2.2 Password Attacks

Attack Type	How It Works & Defence
Dictionary Attack	The attacker uses a file containing real words (a dictionary) and tries each word as your password one by one. Defense: never use a real dictionary word as your password.
Brute Force Attack	Software systematically tries EVERY possible combination of characters until it finds the correct password. Defense: use long, complex, random passwords — the longer the password, the exponentially more combinations must be tried.
Credential Stuffing	The attacker takes username/password pairs leaked from one breached website and tries them on other websites, exploiting the fact that people reuse passwords. Defense: use a UNIQUE password for every website and app.
Social Engineering	The attacker manipulates you psychologically to reveal your password — e.g. a convincing phone call, email, or in-person request pretending to be a trusted authority. Defense: never share your password with anyone, regardless of how legitimate they seem.
Key Logging (Malware)	Malicious software records every keystroke you type (including passwords) and uploads it to the attacker. Defense: only use your own trusted devices; keep software updated; use antivirus/antimalware tools.

2.3 Password Strength — The Maths

Password security is measured by the number of possible combinations — the more combinations, the longer a brute force attack takes. The formula is: (number of possible characters) raised to the power of (password length).

Password Type	Possible Combinations
4-digit PIN (0–9)	$10^4 = 10{,}000$
4 lowercase letters (a–z)	$26^4 = 456{,}976$
4 upper + lower letters (a–z, A–Z)	$52^4 \approx 7.3$ million
4 chars: letters + digits + punctuation (94 chars)	$94^4 \approx 78$ million
8 chars: letters + digits + punctuation (94 chars)	$94^8 \approx 6$ quadrillion

Key insight: each additional character multiplies the difficulty exponentially — not additively. Length matters far more than just adding one special character.

2.4 NIST Password Recommendations

The US National Institute of Standards and Technology (NIST) issues widely followed best-practice guidelines for password security:

- Minimum 8 characters in length — to ensure enough combinations to resist brute force
- Allow at least 64 characters maximum — so users who want very long passwords (passphrases) are not blocked
- Allow all printable ASCII characters — letters, digits, spaces, and punctuation symbols
- Do NOT require periodic forced password changes — forced changes cause users to make predictable, incremental changes (e.g. Password1 → Password2 → Password3) and actually reduce security
- Rate limiting — limit the number of failed login attempts (e.g. lock out after 10 wrong attempts, with increasing delays). This makes brute force attacks impractical even for short passwords.
- Do NOT require specific character types — forcing 'must include a capital letter and a number' tends to make passwords more predictable, not more secure
- Check passwords against known-breached password lists — reject passwords that appear in publicly known data breaches

2.5 Rate Limiting — Why It Matters

Rate limiting drastically increases the time needed for a brute force attack. Without rate limiting, software can try all 10,000 four-digit PINs in milliseconds. With rate limiting:

- iPhone/Android: locks out after 10 wrong attempts, then enforces increasing wait times (1 min, 2 min, 5 min...)
- Some devices can be configured to wipe themselves after a set number of failed attempts
- This turns a millisecond attack into hours, days, or years — even for short passwords
- It also raises the physical risk to the attacker: the longer they need the device, the greater the chance of being caught

2.6 Password Managers

A password manager is software that generates, stores, and automatically fills in long, complex, unique passwords for every site you use — solving the impossible task of remembering hundreds of different passwords.

- Generates truly random, long passwords for every site
- Stores all passwords encrypted behind one strong master password
- Auto-fills credentials to reduce risk of typing passwords on fake (phishing) sites
- Examples: 1Password, Bitwarden, LastPass, Apple Keychain, Google Password Manager
- Even if one website is breached, all your other accounts remain safe because passwords are unique

Best practice: Use a password manager. Let it generate 20+ character random passwords for every account. Only remember your one strong master password.

SECTION 3: Multi-Factor Authentication (MFA)

3.1 What is MFA?

Multi-Factor Authentication (MFA) — also called Two-Factor Authentication (2FA) — requires users to provide two or more independent types of proof of identity before gaining access. Even if an attacker learns your password, they still cannot log in without the second factor.

- Single-factor: only a password — vulnerable to theft, guessing, or phishing
- Two-factor: password PLUS one additional factor — exponentially harder to compromise
- The key insight: the factors must be fundamentally DIFFERENT types — not just two passwords

3.2 The Three Factor Types

Factor Type	What It Is / Examples / Strengths & Weaknesses
Something You Know (Knowledge Factor)	Information only you should know. Examples: password, PIN, security question answer. Weakness: can be guessed, stolen, phished, or observed.
Something You Have (Possession Factor)	A physical object only you should possess. Examples: hardware key fob (TOTP token), authenticator app on your phone (Google Authenticator, Authy), SMS text message code. Weakness: physical theft of the device, or SIM swapping (for SMS). Strength: attacker must be physically near you, not just online.
Something You Are (Inherence / Biometric Factor)	A physical characteristic unique to you. Examples: fingerprint, face scan, iris scan. Weakness: biometrics cannot be changed if compromised (unlike a password). Strength: very hard to forge.

3.3 One-Time Passwords (OTP)

A One-Time Password is a short code (typically 6 digits) that is valid for only one login session or a brief time window. It is automatically generated and synchronized between your device and the server.

- Generated by: hardware key fobs (code changes every 30 seconds), authenticator apps (Google Authenticator, Microsoft Authenticator, Authy), or sent via SMS
- Even if an attacker intercepts the OTP, it expires within seconds making it useless
- Authenticator apps are MORE secure than SMS-based OTPs — see below

3.4 Why SMS-Based OTP is Less Secure

- SMS messages travel over the cellular network, which was not designed with security as a priority
- SIM Swapping attack: an attacker calls your mobile carrier, impersonates you using personal information, and convinces them to transfer your phone number to the attacker's SIM card
- Result: all future SMS texts — including your OTP codes — go to the attacker's phone instead of yours
- Defense: use an authenticator app (not SMS) whenever possible; place a SIM lock/PIN with your carrier

Best practice: If a site offers both SMS and app-based 2FA, always choose the authenticator app.

3.5 Two-Step vs. Two-Factor Authentication

- Two-Step Authentication: two sequential steps that may both be the same type (e.g. password + security question = two knowledge factors)
- Two-Factor Authentication (true 2FA): two steps from fundamentally different categories (e.g. password + OTP from your phone = knowledge + possession)
- True 2FA is significantly stronger because compromising one factor does not automatically compromise the other

SECTION 4: Passkeys — The Future of Passwordless Authentication

4.1 What are Passkeys?

Passkeys (technically an implementation of the Web Authn standard) are a modern replacement for passwords that use public-key cryptography instead of shared secrets. They eliminate the password entirely.

- Based on public-key cryptography — the same mathematics behind HTTPS and digital signatures
- No password to remember, steal, phish, or breach
- Increasingly supported by major platforms: Apple, Google, Microsoft, and growing numbers of websites

4.2 How Passkeys Work

- Registration (creating an account):
  - Your device prompts you for a local verification — fingerprint, face scan, or PIN
  - Your device generates a unique public/private key pair just for this website
  - Your device sends ONLY the public key to the website — the private key never leaves your device
  - The website stores your public key alongside your user ID
- Login (returning to the site):
  - The website sends your browser a random 'challenge' — a piece of random data
  - Your device prompts your biometric/PIN to unlock
  - Your device signs the challenge using your private key
  - Your device sends the signed challenge back to the website
  - The website uses your previously stored public key to verify the signature
  - If the signature is valid, you are authenticated — without ever transmitting a password

4.3 Why Passkeys Are More Secure

- Phishing-resistant — there is no password to steal from a fake login page
- Breach-resistant — the website only stores your public key, which is useless to an attacker without your private key
- No reuse — each passkey is unique to one website; compromise of one does not affect others
- Device-bound — the private key is stored securely on your device (or in a synced cloud keychain)
- Synchronisation — Apple, Google, and Microsoft can sync passkeys across your devices securely

Note: Passkeys are not yet universal — adoption is growing but many websites still only support passwords. Expect this to change significantly in the next few years.

SECTION 5: Securing Data — Hashing and Encryption

5.1 Hashing vs. Encryption — Key Difference

Property	Hashing
Direction	One-way — cannot be reversed
Purpose	Verifying data integrity; storing passwords safely
Key required?	No
Same input = same output?	Yes (always)
Use case	Password storage, file integrity checks

5.2 Password Hashing

Websites should NEVER store your password in plain text. Instead, they store a hash of your password — a fixed-length fingerprint generated by a mathematical function. When you log in, the website hashes what you type and compare it to the stored hash.

- Good hash functions: produce completely different outputs for even tiny input changes; are computationally infeasible to reverse
- Common hash functions: bcrypt, scrypt, Argon2 (modern, slow-by-design — ideal for passwords); SHA-256 (faster — better for file integrity, not passwords alone)
- If an attacker steals the hash database, they cannot simply read the passwords — they must crack them
- Cracking methods against hashes:
  - Dictionary attack — hash every word in a dictionary and compare to stolen hashes
  - Brute force — hash every possible password combination and compare
  - Rainbow table — use a pre-computed table of password-to-hash mappings for instant lookup

5.3 Salting — Defending Against Rainbow Tables

A salt is a unique random value generated for each user that is combined with their password before hashing. This ensures that even if two users have the same password, their stored hashes will be completely different.

- How it works: hash (password + salt) = unique hash value per user
- The salt itself is stored alongside the hash (it does not need to be secret)
- Why it works: rainbow tables become useless — an attacker would need to pre-compute a separate table for every possible salt value
- When you log in: the server retrieves your salt, re-hashes your input with it, and compares the result to your stored hash
- Modern password hash functions (bcrypt, Argon2) include salting automatically — and are deliberately slow to compute, making brute force attacks extremely expensive

Best practice: Never implement your own password hashing scheme. Use a well-vetted library (bcrypt, Argon2) — security researchers have already stress-tested these.

5.4 Encryption in Transit vs. End-to-End Encryption

Type	What It Means / Who Can Read Data / Examples
Encryption in Transit (TLS/HTTPS)	Data is encrypted between your browser and the server. BUT: the server (e.g. Gmail) can still read your data once it arrives. Protects against eavesdroppers on the network but not against the service provider itself. Examples: HTTPS websites, standard email services.
End-to-End Encryption (E2EE)	Data is encrypted from sender to recipient. The intermediary server can only see scrambled data — even the service provider cannot read it. Examples: WhatsApp, iMessage (when enabled), Signal. Strongest privacy guarantee.

Important: Alice having a secure connection to Gmail + Bob having a secure connection to Gmail does NOT mean Alice has a secure connection to Bob. Security does not work through transitivity — Gmail in the middle could technically read everything.

5.5 Public-Key Cryptography — The Foundation of Modern Security

Public-key (asymmetric) cryptography uses mathematically linked key pairs. What one key encrypts, only the other can decrypt.

- Public key — shared openly with anyone; used to encrypt data or verify signatures
- Private key — kept secret; used to decrypt data or create signatures
- Key property: even knowing the public key, it is computationally infeasible to derive the private key
- Applications:
  - HTTPS — your browser uses the server's public key to establish an encrypted connection
  - Digital signatures — signing with your private key; anyone can verify with your public key
  - Passkeys — your device signs challenges with a private key; websites verify with your public key
  - Email encryption (PGP/GPG) — encrypt a message with the recipient's public key

SECTION 6: Securing Systems — Networks and Infrastructure

6.1 IP Addresses and Ports

Every device on the internet has an IP address — a unique numeric identifier like a postal address. When your computer sends data, it is wrapped in a 'virtual envelope' that includes the source IP address and the destination IP address.

- IPv4 addresses: 4 numbers separated by dots (e.g. 192.168.1.1)
- IPv6 addresses: longer format to support the growing number of connected devices
- Port numbers: a number on the outside of the envelope indicating WHICH service the data is destined for on the target computer

Port Number	Service
80	HTTP — standard unencrypted web traffic
443	HTTPS — encrypted web traffic (TLS)
22	SSH — secure remote access to computers
53	DNS — domain name resolution
25	SMTP — email sending

6.2 Domain Name System (DNS)

DNS translates human-readable domain names (e.g. harvard.edu) into machine-readable IP addresses. Your browser cannot connect to a server by name alone — it needs the IP address.

- When you type a URL, your device queries a DNS server: 'What is the IP address for harvard.edu?'
- DNS lookup hierarchy: device cache → home router/ISP DNS → authoritative DNS servers
- DNS responses are cached for efficiency — your device remembers recent lookups

6.3 DNS Privacy Problem

- Standard DNS (port 53) is UNENCRYPTED — your DNS queries are sent in plain text
- This means: your Internet Service Provider (ISP) can see every domain name you visit — even if the actual website uses HTTPS
- Coffee shops, airports, and other public Wi-Fi providers also see all your DNS queries
- ISPs commonly log this data, which can be used for advertising, government surveillance, or sold to third parties

6.4 DNS Solutions for Privacy

Solution	How It Works
DNS-over-HTTPS (DoH)	DNS queries are sent encrypted inside HTTPS connections (port 443). Your ISP cannot see which domains you are querying. A trusted DNS provider (e.g. Cloudflare 1.1.1.1, Google 8.8.8.8) resolves queries instead of your ISP.
DNS-over-TLS (DoT)	DNS queries are encrypted using TLS (but not wrapped in HTTP). Similar privacy benefit to DoH — encrypted queries that ISPs cannot inspect.

6.5 Firewalls

A firewall is software (or hardware) that monitors and controls network traffic based on defined rules — deciding what is allowed in or out of a network.

- Blocks unwanted incoming connections — prevents attackers from reaching open services
- Blocks unwanted outgoing connections — prevents malware from sending data outside
- Traffic filtering methods:
  - IP address-based — block specific IP addresses (e.g. block access to known malicious servers)
  - Port-based — block all traffic except on specific ports (e.g. only allow port 443 for HTTPS and port 22 for SSH)
  - Deep Packet Inspection (DPI) — examine the actual content of packets to identify and block malicious traffic even if it uses an allowed port
- Home routers have built-in basic firewalls that block unsolicited inbound connections
- Enterprise firewalls are far more sophisticated and configurable
- Firewalls are not foolproof — if malware is already inside, the firewall may not catch outbound traffic from it

6.6 Port Scanning & Penetration Testing

- Port scanning: systematically trying all port numbers (0–65,535) on a target to discover which services are running and listening
- Attackers use port scanning to identify attack surfaces — open ports that might be exploitable
- 'Security through obscurity' (running services on non-standard ports) is not effective — port scanners will find them
- Penetration testing (ethical hacking): a legitimate, paid security practice where professionals use hacker techniques to find and report vulnerabilities before real attackers do
  - Red team: tries to breach systems (attackers)
  - Blue team: defends systems (defenders)
  - Purple team: combines both for continuous improvement
- Pen testers may use: port scanning, brute force, social engineering, phishing simulations, code review
- Ethical hackers report vulnerabilities to the organization and receive payment — not prosecution

SECTION 7: Privacy Tools — VPNs and Tor

7.1 Virtual Private Networks (VPN)

A VPN creates an encrypted tunnel between your device and a VPN server. All your internet traffic passes through this encrypted tunnel before reaching its destination.

- What VPNs do:
  - Encrypt all traffic between your device and the VPN server — your ISP or local network cannot see your traffic content
  - Mask your real IP address — websites see the VPN server's IP address, not yours
  - Appear to be located in a different country — useful for accessing geo-restricted content
  - Protect you on public Wi-Fi — hotel, airport, and coffee shop networks cannot eavesdrop
- What VPNs do NOT do:
  - Make you completely anonymous — the VPN provider itself can see your traffic
  - Protect you once traffic leaves the VPN server — if you use HTTP (not HTTPS), traffic is still unencrypted after the VPN
  - Prevent malware that is already on your device from operating
  - Prevent browser fingerprinting or tracking cookies
- Business use case: allow remote employees to securely access internal company networks from home or while travelling

Key limitation: a VPN transfers trust from your ISP to your VPN provider. You must trust your VPN provider not to log or sell your data.

7.2 Tor (The Onion Router)

Tor is free software that routes your internet traffic through a series of volunteer-operated servers (nodes) around the world, encrypting it multiple times — one layer per node, like layers of an onion.

- How Tor works:
  - Your Tor client selects a random path of 3+ nodes through the Tor network
  - Your data is encrypted THREE times — once with each node's public key
  - Each node decrypts only ONE layer (its own), revealing only the next hop — not the full path or destination
  - Only the final 'exit node' sees the destination, but not your identity
  - Each connection uses a different path, making pattern analysis very difficult
- What Tor protects: makes it very difficult for any single node, your ISP, or a website to link your identity to your browsing activity
- Limitations:
  - Significantly slower than normal browsing — multiple hops add latency
  - Exit node can see unencrypted traffic if you use HTTP (not HTTPS)
  - If you are the ONLY Tor user on a network (e.g. a company network), that alone might flag you as suspicious
  - Determined government entities could potentially reconstruct paths by subpoenaing multiple nodes
- Tor does not delete logs by design — this is intentional to protect privacy
- Tor Browser is the easiest way to use Tor — available free from the Tor Project

7.3 VPN vs. Tor — Comparison

Property	VPN
Speed	Fast — single hop to VPN server
Who sees your traffic?	VPN provider sees all your traffic
Cost	Usually requires a paid subscription
Anonymity level	Moderate — trust the VPN provider
Best for	Privacy from ISP, public Wi-Fi security, corporate remote access
Setup complexity	Easy — one-click apps available

SECTION 8: Cookies, Browser Tracking and Privacy

8.1 What are Cookies?

Cookies are small pieces of data that a web server stores on your browser. They allow websites to remember you between visits — since HTTP is stateless (each page request is independent).

- A server 'plants' a cookie on your browser using the HTTP response header: Set-Cookie: session=1234ABCD
- Every subsequent visit to that site, your browser sends the cookie back: Cookie: session=1234ABCD
- This lets the server identify you without requiring you to log in on every page

8.2 Types of Cookies

Cookie Type	Purpose & Privacy Implications
Session Cookies	Temporary — expire when you close the browser or session ends. Used to maintain shopping carts, login sessions. Relatively privacy-friendly as they don't persist long.
Persistent / Tracking Cookies	Long-lived — stored for days, months, or years (e.g. Google Analytics cookie: max-age of 2 years). Used to track your behavior across visits and websites for analytics and advertising. The most privacy-invasive type.
First-Party Cookies	Set by the website you are visiting. Generally necessary for functionality (login, preferences, cart). Less invasive.
Third-Party Cookies	Set by external services embedded in the page (e.g. social media buttons, ad networks). Allow tracking across many different websites even without your knowledge.

8.3 Browser Fingerprinting

Even without cookies, websites can identify and track you using browser fingerprinting — collecting a unique 'fingerprint' from the combination of your browser settings and device characteristics.

- Data points used: browser type and version, operating system, screen resolution, installed fonts, browser plugins/extensions, time zone, language settings, graphics card renderer (via WebGL)
- Combined, these create a fingerprint that is often unique enough to identify you even without cookies
- Clearing cookies does NOT prevent fingerprinting — the fingerprint is derived from your device/browser configuration
- Defense: use the Tor Browser (standardizes many fingerprinting attributes), privacy-focused browsers (Firefox, Brave), or browser extensions that block fingerprinting
- Even changing your IP address (via VPN) may not defeat fingerprinting if your browser profile remains the same

8.4 App Permissions and Location Privacy

Modern operating systems (iOS, Android) prompt you to grant or deny specific permissions to apps — a fine-grained control mechanism that shifts security decisions to the user.

- Permission types: camera, microphone, contacts, location, photos, notifications
- Permission granularity on iOS: Allow Always / Allow While Using App / Ask Next Time / Never
- Location tracking: mapping apps (Google Maps, Apple Maps) need location access — but 'Allow Always' means the app can track you even when closed and, in your pocket,
- If you grant 'Always On' location to many apps, companies can build detailed movement profiles of everywhere you go
- Best practice: regularly audit your app permissions. Grant location 'While Using App' only. Revoke permissions from apps that don't need them.

Remember: your phone is essentially a sensor-packed tracking device. Every permission you grant is a potential data stream about your behavior and location.

SECTION 9: Securing Software — Common Vulnerabilities

9.1 Cross-Site Scripting (XSS)

XSS (Cross-Site Scripting) is a web vulnerability where an attacker injects malicious JavaScript code into a trusted website, which then executes in other users' browsers.

- Reflected XSS: malicious code is embedded in a URL. When the victim clicks the link, the server reflects the code back in the response and the browser executes it.
- Stored XSS: malicious code is saved in the website's database (e.g. in a comment field). Every user who views that content has the code executed in their browser.
- Impact: steal session cookies, redirect users to phishing sites, log keystrokes, deface websites
- Defenses:
  - Input escaping / output encoding: convert special characters (<, >, ', ") into their HTML entities before rendering — so script tags are displayed as text, not executed as code
  - Content-Security-Policy (CSP) HTTP header: instructs the browser to only execute JavaScript from trusted sources (external.js files from the same domain). Inline <script> tags are blocked.
  - Frameworks: modern web frameworks (React, Vue, Angular) escape output by default — reducing XSS risk

9.2 Cross-Site Request Forgery (CSRF)

CSRF tricks an authenticated user's browser into making unauthorized requests to a web application the user is currently logged into, using the user's existing session credentials.

- Example: you're logged into your bank. You visit a malicious website that contains hidden code that submits a transfer request to your bank using your active session cookie — without your knowledge.
- Defenses:
  - CSRF tokens: the server generates a unique secret token for each form. When the form is submitted, the token is validated. Malicious sites cannot guess this token.
  - Same Site cookie attribute: instructs browsers not to send cookies in cross-site requests
  - Checking the HTTP Referrer header to confirm requests originate from the expected domain

9.3 Buffer Overflow

A buffer overflow occurs when software receives more input than the memory buffer allocated to store it — causing data to 'overflow' into adjacent memory, potentially overwriting critical data including return addresses and machine code pointers.

- Memory layout of a running program:
  - Machine code (program instructions) stored in upper memory
  - Stack (growing upward from lower memory): stores local variables, function data, and return addresses
  - Return address: a pointer telling the CPU where to resume execution after a function completes
- The attack: attacker supplies input longer than the allocated buffer; excess data overwrites the return address; the attacker substitutes a return address that points to their own malicious code
- Result: the program executes the attacker's code — potentially giving remote control of the system
- Common in: older C/C++ programs that use unsafe functions (gets(), strcpy()) without bounds checking

- Defenses:
  - Use memory-safe programming languages (Python, Java, Rust) that automatically manage bounds
  - Address Space Layout Randomization (ASLR): randomizes where code is loaded in memory, making it hard to predict addresses to overwrite
  - Stack canaries: a random value placed between the buffer and return address — if overwritten, execution is aborted
  - Data Execution Prevention (DEP/NX): marks memory regions as non-executable so injected code cannot run
  - Keep software updated — buffer overflow patches are released regularly

9.4 Social Engineering

Social engineering is not a technical attack but a human one — manipulating people psychologically to reveal confidential information, grant access, or take harmful actions.

- Phishing: fraudulent emails, messages, or websites that impersonate trusted entities to steal credentials or install malware
- Vishing: voice phishing — phone calls impersonating banks, government agencies, or IT support
- Pretexting: creating a fabricated scenario (e.g. 'I'm from IT and need your password to fix your account') to gain trust
- In-person: someone physically claiming to be a maintenance worker, colleague, or authority to gain access
- The demonstration: being asked to write down a password — even by a teacher in class — and complying is being socially engineered
- Defenses:
  - Healthy skepticism: any request for your password, credentials, or sensitive data — regardless of who is asking — should trigger suspicion
  - Verify independently: if someone calls claiming to be your bank, hang up and call the official number yourself
  - Never share passwords: no legitimate IT department, bank, or website will ever ask for your password
  - Security awareness training: organizations should regularly train employees to recognize and report social engineering attempts

SECTION 10: Key Concepts Quick Reference

10.1 Attack Types — Cheat Sheet

Attack	Description / Defense
Dictionary Attack	Trying words from a word list as passwords. Defense: avoid real words; use random passwords.
Brute Force Attack	Trying every possible character combination. Defense: long complex passwords; rate limiting.
Credential Stuffing	Using leaked credentials on other sites. Defense: unique password per site; password manager.
Phishing	Fake sites/emails to steal credentials. Defense: check URLs; use passkeys/hardware keys.
SIM Swapping	Tricking carrier to move phone number. Defense: use authenticator apps, not SMS 2FA; add a SIM PIN.
Key Logging	Malware recording your keystrokes. Defense: use only your own devices; keep software updated.
Rainbow Table	Pre-computed hash lookup tables. Defense: salted hashes; use bcrypt/Argon2.
Social Engineering	Manipulating humans to reveal info. Defense: healthy skepticism; verify identities independently.
XSS	Injecting malicious scripts into trusted sites. Defense: input escaping; CSP headers.
CSRF	Forging requests using victim's session. Defense: CSRF tokens; Same Site cookies.
Buffer Overflow	Overflowing input to execute attacker code. Defense: memory-safe languages; ASLR; DEP.
Port Scanning	Discovering open services on a target. Defense: firewalls; close unnecessary ports.
DNS Spoofing	Returning fake IP addresses from a DNS server. Defense: DNSSEC; DNS-over-HTTPS.

10.2 Defense Technologies — Cheat Sheet

Defence	What It Does
Strong Password (8+ chars, random, unique)	Resists brute force and credential stuffing attacks
Password Manager	Generates and stores unique random passwords for every site
Rate Limiting	Limits failed login attempts — makes brute force impractical
Multi-Factor Authentication (MFA)	Requires a second factor (possession or biometric) beyond a password
Passkeys / WebAuthn	Replaces passwords with public-key cryptography — phishing-resistant
Password Hashing (bcrypt/Argon2)	Stores passwords as slow, salted hashes — protects if database is breached
Salting	Adds unique random data per user before hashing — defeats rainbow tables
HTTPS / TLS	Encrypts data in transit between browser and server
End-to-End Encryption	Encrypts data so only sender and recipient can read it — not the service provider
Firewall	Controls what network traffic is allowed in and out
VPN	Encrypts all traffic and masks your IP from your ISP and local network
Tor	Routes traffic through multiple encrypted nodes for high anonymity
DNS-over-HTTPS (DoH)	Encrypts DNS queries so ISPs cannot see which sites you visit
Content-Security-Policy (CSP)	HTTP header restricting which scripts/styles the browser can execute
CSRF Token	Server-issued secret token validating that form submissions originate legitimately
App Permissions (iOS/Android)	Fine-grained control over what data and hardware each app can access

— End of Course Notes —