

Security Fundamentals

Reading time: ~13 min · Prereqs: Lessons 01–10 · *Snapshot 2026* — *concepts are stable; specific breach examples and ransomware stats age fast.*

Why this matters to you

Every system you've learned about in this course — websites, cloud accounts, codebases, databases, deployment pipelines, networks, and virtualized compute environments — can be turned against its users. Cybercrime *as of 2026* is among the largest illegal industries on the planet, measured in the trillions of dollars annually (per Cybersecurity Ventures' running estimate). Even if you'll never write security code, you'll touch security daily: the password manager in your browser, the MFA prompt on your bank app, the breach-notification email that arrives every few months, the suspicious-looking text from "your bank".

This lesson gives you the full shape — broader than "Security Basics." Lessons 08 and 09 built the network and compute layers you now rely on. Security isn't something you bolt on afterward; it runs through those layers from the start. We'll cover how attacks happen, then how the network itself becomes the first line of defense, then how shared virtualized compute introduces its own set of risks. By the end you'll know which questions to ask at every layer.

1. Three intuitions to anchor everything

Three ideas explain most of modern security. First: **attackers take the cheapest path**. They are economically rational. If phishing a human takes ten minutes and exploiting a zero-day vulnerability takes ten months, phishing wins — every time. Second: **humans are the most-attacked surface**. Every major incident report — Verizon's **Data Breach Investigations Report** (DBIR), Mandiant's **M-Trends**, IBM's **X-Force Threat Intelligence** — has put the "human element" in the majority of breaches every year for over a decade (*as of 2026*). Third: **security is layered**. No single fix is enough. The industry phrase is *defense in depth*: assume any one layer will fail, and stack enough layers that the attacker gives up before reaching the prize.

2. The shape of attacks — five archetypes

Phishing — trick a human into giving up credentials or running your code. Still the dominant initial-access vector *as of 2026* (Verizon DBIR). **Spear-phishing** is the targeted version: an attacker studies one specific person and crafts a message tailored to them. AI-generated spear-phishing is now indistinguishable from a colleague's tone, and deepfake voice calls ("vishing") have moved from novelty to routine.

Credential stuffing — attackers buy databases of leaked username/password pairs (billions of them are in circulation), then try each pair against every service. Because most people reuse passwords, a 1% hit rate is enormous.

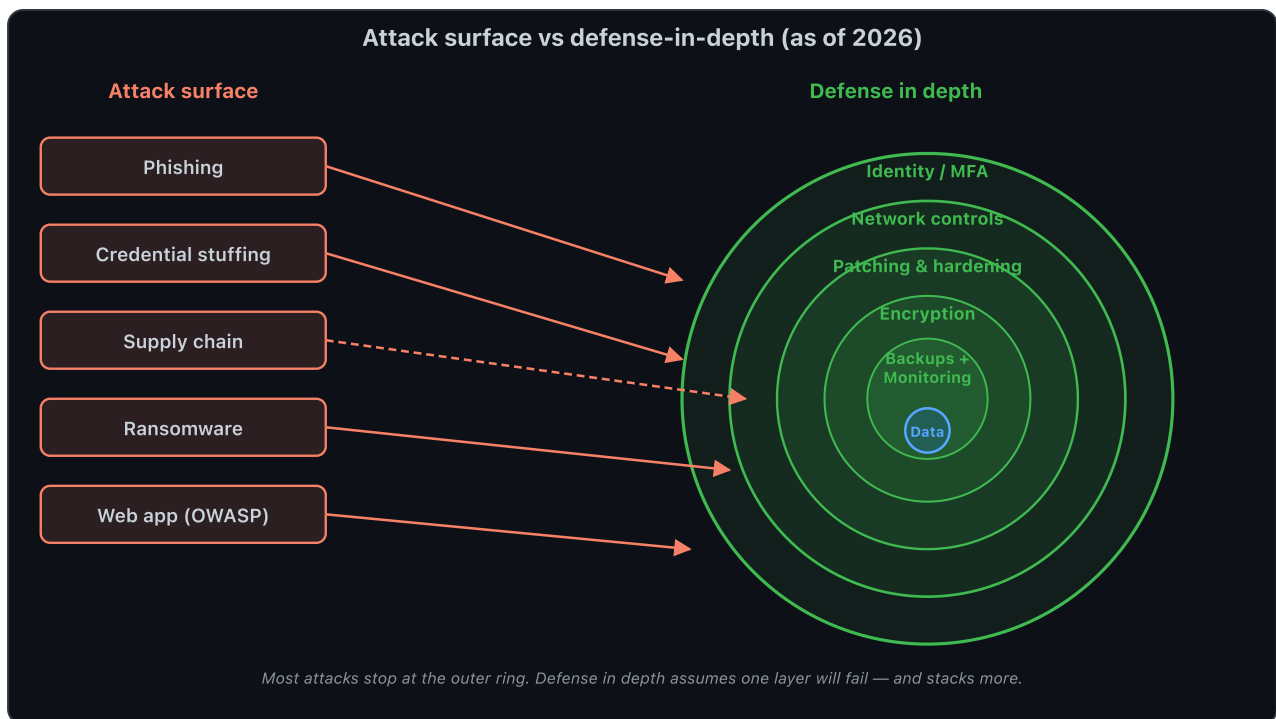
Supply-chain attacks — instead of attacking you, attack a library you depend on. Famous cases: **SolarWinds** (2020), **Log4Shell** (2021), the **xz-utils** backdoor (caught in 2024, cited by Mandiant as the canonical near-miss), and ongoing npm and PyPI compromises. A modern app pulls in hundreds of indirect dependencies; any one of them is a foothold.

Ransomware — encrypt the victim's data, demand payment for the key. *As of 2026* it remains one of the highest-impact attack categories per the annual Sophos **State of Ransomware** and IBM X-Force reports. Hospitals, municipalities, and manufacturers are recurring targets. Modern ransomware is usually **double-extortion**: encrypt AND threaten to leak the data publicly if unpaid.

Web application attacks — the **OWASP Top 10** has been the canonical industry list since 2003, refreshed every few years. As of the 2021 edition (still current heading into the 2026 refresh window), the top three are *broken access control*, *cryptographic failures*, and *injection* (including SQL injection — the same vulnerability that has been ranked highly for 25 years).

3. The attack surface vs the defense onion

Here is the same idea drawn as a diagram. On the left, the five attack archetypes. On the right, the layered defenses an attacker must traverse to reach the protected asset in the middle. Most attackers stop at the first locked layer; the rare persistent ones test every layer in turn — which is exactly what "defense in depth" is designed for.



2026 threat-landscape advisory

The actively hot threat categories as of 2026, per the most-recent annual incident reports: **AI-augmented phishing & vishing** (deepfake voice calls impersonating executives); **ransomware against critical infrastructure** (hospitals, water utilities, ports); **long-tail supply-chain compromises** (small upstream packages with huge transitive blast radius); and **identity-provider attacks** (single-sign-on tenants are a high-value target). Re-check the most recent Verizon DBIR and Mandiant M-Trends for next year's hot list — this paragraph ages fastest.

4. Security in the network layer — the first line of defense

Before an attacker touches your servers, they have to get through the network. This is where Lesson 08's networking infrastructure becomes a security story. The classic tool is the **firewall**: a device (or software) that looks at every packet and decides whether to let it pass based on a rulebook. A **perimeter firewall** sits at the edge of your organization's network and filters traffic coming from the internet. A **host-based firewall** runs on the individual machine itself — so even if an attacker slips past the perimeter, each server still has its own gatekeeper.

Network segmentation is the architectural principle of cutting a network into isolated zones — so that someone who compromises a printer in the guest Wi-Fi zone can't simply walk across to the payroll database. You've seen this in Lesson 08 as VLANs and subnets; the security framing calls it a *DMZ* (demilitarized zone) for public-facing services, and an *internal segment* for sensitive data that never needs to be reached from the internet at all.

NAC — Network Access Control is the policy that says "you can only join this network if your device meets a set of conditions" (up-to-date antivirus, OS patch level, valid certificate). It's the reason a hospital or bank will reject your personal laptop from joining their corporate Wi-Fi even if you know the password.

IDS and IPS — Intrusion Detection and Intrusion Prevention Systems — sit on the network watching traffic patterns, not just headers. An IDS raises an alert when it sees something suspicious (a port scan, an unusual flood of DNS queries, a known malware signature). An IPS goes further and blocks the traffic in real time. Open-source examples include **Snort** and **Suricata**; commercial equivalents are built into most enterprise firewall products.

VPN security matters because remote-access VPNs are now the front door to corporate networks. A VPN that isn't patched, uses weak authentication, or has a publicly disclosed vulnerability is essentially a locked door with the key taped to the outside — a major ransomware entry point every year. **Encrypted DNS (DoH/DoT)** — DNS-over-HTTPS and DNS-over-TLS — closes a different gap: ordinary DNS lookups travel in plain text and can be intercepted or forged to redirect you to a fake site. Encrypted DNS tunnels those lookups so eavesdroppers can't see what domains you're resolving, and so the responses can't be silently altered.

"The perimeter is dead" — and what comes after

A much-repeated observation in security circles is that "the perimeter is dead" — meaning the old castle-and-moat model of one big firewall around a trusted network no longer works when employees work from anywhere and services live in the cloud. This is true as a critique of naive perimeter thinking. But it doesn't mean firewalls and segmentation are useless — it means they aren't *sufficient on their own*. The replacement model is **zero trust**: assume every request is untrusted regardless of where it comes from, and verify by identity and device

posture every time. Zero trust and network controls coexist; one doesn't replace the other.

5. Security in computing and virtualization — shared compute is shared risk

Lesson 09 showed how multiple workloads share physical hardware through hypervisors and containers. That sharing is efficient — and it introduces a category of security questions that don't exist when every workload runs on its own dedicated machine.

Hypervisor isolation is the core guarantee of virtualization: your virtual machine should not be able to see or affect your neighbor's. Most of the time, this holds well. But the guarantee has had real cracks. **VENOM** (2015) was a vulnerability in a widely-used virtual floppy-disk driver that let an attacker escape their virtual machine and run code on the hypervisor host itself. **L1TF / Foreshadow** (2018) was a CPU speculative-execution flaw (in the same family as Spectre and Meltdown) that let a process read memory from another VM sharing the same physical core. These aren't reasons to distrust virtualization — they're why hypervisor vendors treat every CVE as an emergency patch. The lesson: isolation guarantees come from software running on hardware, and both layers can have bugs.

Container escapes are the container equivalent of a VM breakout. Containers share the host OS kernel (unlike VMs, which have their own kernel). That thinner boundary makes them lightweight but also means a kernel vulnerability can let a container process break out to the host. In practice, well-configured containers with seccomp profiles, AppArmor/SELinux policies, and non-root users inside are very hard to escape from — most successful container escapes exploit misconfigurations rather than kernel zero-days.

Secrets management is where most real-world compute security problems live, not in spectacular VM escapes. A *secret* in this context means any credential or key that grants access to something: a database password, an API key, a TLS certificate. The number-one practical failure mode is **committing secrets to version control** — a developer pastes an AWS API key into a configuration file and pushes it to GitHub. Automated bots scan public repositories for credential patterns and find them within

minutes. The right pattern is to store secrets in a dedicated secrets manager (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault) and inject them into the environment at runtime, never baking them into images or code.

RBAC at the orchestrator — Role-Based Access Control in Kubernetes and similar platforms — is the question of who is allowed to do what inside the compute environment, not just who can reach it from outside. In Kubernetes, this means controlling who can run `kubectl exec` to drop a shell into a running container (equivalent to "log in to the running server"), who can read secrets, and who can deploy new workloads. Poor RBAC configuration — such as granting wildcard permissions in a cluster to make things "just work" — is one of the most common security findings in cloud-native infrastructure audits.

6. The shape of defense — five layers

Identity is the new perimeter. The old model (firewall around the corporate network, everything inside is trusted) is dead — your employees are working from coffee shops and your services are spread across clouds. The defining shift of the past decade is **zero trust**: never trust based on network location; verify every request based on identity. Practically that means **MFA** everywhere (multi-factor auth: password + a second factor like an authenticator app or a hardware key — both far stronger than SMS), **SSO** (single sign-on, to reduce password proliferation), and **least privilege** (every account gets only what it strictly needs).

Patch fast. Most successful breaches exploit vulnerabilities with patches already available — sometimes for months. The **CVE** system (cve.org) is the public registry of known vulnerabilities; the **CVSS** score grades severity. Auto-update where you can; prioritize the rest by severity and known-exploited status (CISA's **KEV** catalog — the short-list of vulnerabilities actively being used in attacks *right now*).

Encrypt at rest and in transit. HTTPS / TLS 1.3 for network traffic; disk encryption (BitLocker, FileVault, LUKS) for laptops; encrypted backups in storage. **Back up — and test the restore.** Most ransomware victims discover their backups didn't work *during* the incident (Sophos *State of Ransomware*). Offline / immutable backups beat cloud-only backups when the cloud account itself is what got owned. **Monitor.** A **SIEM** (Security Information & Event Management — Splunk, Microsoft Sentinel, Elastic, Google Chronicle) aggregates logs and alerts on suspicious patterns. **Dwell time** —

how long an attacker is inside before detection — is the most-watched metric; the 2026 median is still measured in weeks per Mandiant M-Trends.

7. Who does security in a company?

Security engineers build defensive systems — they write the IAM rules, harden cloud accounts, integrate scanners. **SOC analysts** sit in a **Security Operations Center** and watch alerts in real time, triaging which are real incidents. **Incident responders** (sometimes the same people, sometimes a specialized **DFIR** team — Digital Forensics & Incident Response) handle a breach end-to-end: contain, eradicate, recover, write the post-mortem. **Penetration testers** (pentesters) and **red teams** are paid to attack the company's own systems to find weaknesses before real attackers do. The **CISO** (Chief Information Security Officer) owns the security program at the executive level. As of 2026 the explicit expectation is that every developer carries security responsibility — the **DevSecOps** label captures the shift (Lesson 10's pipeline, plus security automation baked in: dependency scanning, secret scanning, infrastructure-as-code linting, all running on every pull request).

What people argue about

- **Pay the ransom or not?** Law enforcement says no — payments fund the next attack and there's no guarantee the decryption key works. Insurers and CFOs sometimes say yes — it's cheaper than the downtime. The honest answer depends on whether your backups actually work, which most organizations only find out *during* the incident.
- **Security through obscurity.** Universally derided in textbooks; quietly relied on in many real systems. The mature view: obscurity is fine as one layer in defense in depth — fatal as the only layer.
- **Open source vs proprietary for security tools.** "Many eyes make all bugs shallow" vs "obscurity buys time"; both camps have war stories (and the xz-utils near-miss was cited by both). Most modern security stacks blend the two.
- **Perimeter defense: dead or alive?** "The perimeter is dead" is a useful provocation against naive castle-and-moat thinking — but in practice, most organizations still have firewalls, VPNs, and segmented networks, and they

still catch real attacks. The honest answer: a perimeter alone is not enough; a perimeter plus zero-trust identity plus monitoring is still the dominant model in 2026.

- **Are containers a security sandbox?** Containers are not a security boundary in the same way VMs are — they share the host kernel. Whether that matters in practice depends entirely on how well the rest of the configuration is hardened. The debate is less "containers vs VMs" and more "what's your threat model?" For most workloads, well-configured containers are secure enough; for high-sensitivity multi-tenant environments (e.g., running untrusted user code), you want VM-level isolation or specialized sandboxing (gVisor, Kata Containers).
- **Zero trust — paradigm shift or rebranded segmentation?** Critics point out that "zero trust" often describes what careful practitioners were already doing with strong identity, MFA, and least-privilege — repackaged as a marketing term by vendors. Proponents argue the label matters because it forces a explicit policy conversation about every trust boundary. Both are partly right.

Reading on

- **Verizon Data Breach Investigations Report (DBIR)** — free annual; the canonical "what actually happened last year" report, with a stable methodology that lets you compare year to year.
- **Mandiant M-Trends** — free annual; complements the DBIR with an incident-response perspective (dwell time, attacker tooling, ransomware trends).
- **OWASP Top 10** — owasp.org/Top10. The canonical web-application vulnerability list, updated every few years; the one industry artifact every developer should be able to recognize.
- **Krebs on Security** — krebsonsecurity.com. Independent journalist Brian Krebs; the easiest-to-read running diary of the cybercrime ecosystem, with the human stories behind the headlines.
- **Suricata IDS/IPS** — suricata.io. The leading open-source network intrusion detection and prevention engine. The project website has readable documentation on how rules are written and what network-layer threats look like from a defender's perspective — good context even if you'll never run it yourself.

- **HashiCorp Vault documentation — "What is secrets management?"** — developer.hashicorp.com/vault/docs/what-is-vault. A concise, vendor-neutral framing of the secrets-management problem and the pattern for solving it.
- **CISA Zero Trust Maturity Model** — cisa.gov/zero-trust-maturity-model. The U.S. government's practical guide to zero trust, written for practitioners but readable as a framework overview. Useful for understanding what "zero trust" means operationally, beyond the vendor marketing.

LAST-REVIEWED: 2026-06-02