CCNA CYBERSECURITY · EXAM 200-201

CCNA Cybersecurity 200-201

A Complete Learning Guide for Beginners

5 CHAPTERS · 25 SUB-SECTIONS · 125 QUESTIONS

Covers every topic in the 200-201 exam blueprint

Independent · Unofficial Study Guide

JOZEF BAROŠ

AUTHOR

CCNA CYBERSECURITY CERTIFICATION · EXAM 200-201

CCNA Cybersecurity

200-201

A Complete Learning Guide for Beginners

Every exam topic explained from the ground up — with plain-English analogies, runnable examples, exam tips, and Cisco-style practice quizzes.

</>

5 CHAPTERS · 25 SUB-SECTIONS · 125 PRACTICE QUESTIONS

Covers every topic in the 200-201 (CCNACBR) exam blueprint

An independent, unofficial study guide — not affiliated with or endorsed by Cisco Systems, Inc.

JOZEF BAROŠ
AUTHOR

CCNA Cybersecurity 200-201 — Complete Learning Guide

First edition · 2026

About This Book

This guide prepares you for the **Understanding Cisco Cybersecurity Operations Fundamentals (CCNACBR 200-201)** exam — the certification that proves you understand security concepts, security monitoring, host-based analysis, network intrusion analysis, and security policies and procedures: the core skills of a **Security Operations Center (SOC) analyst**. It is written for complete beginners — no prior security or networking experience is assumed.

The teaching method is simple: every concept is first explained as if to a curious newcomer, using an everyday analogy you can picture, before adding the precise detail the exam tests. The goal is understanding that sticks, not facts you cram and forget.

How each sub-section is built

To make the material varied and easy to digest, every sub-section follows the same friendly rhythm, with colored boxes that flag the most useful information at a glance:

> 💡 **Did you know? — Did you know?**
> A non-obvious fact or deeper insight that rewards your curiosity.

> ✅ **Exam Tip — Exam Tip**
> Exactly what the exam tests here — the distinctions and mnemonics that score points.

> ⚠️ **Common Pitfall — Common Pitfall**
> A mistake learners actually make, and how to avoid it.

> 🌍 **From the Field — From the Field**
> How the concept is really used in a real Security Operations Center.

Each sub-section then closes with a **Summary**, a bolded **Key Takeaways** recap, and a **five-question, Cisco-style Knowledge Check** so you can test yourself immediately — **125 practice questions** in all, with full explanations.

How to use it

1. Read a sub-section, paying attention to the analogies — they are the hooks your memory will hang on to.
2. Try the commands and code yourself in a safe lab; the examples are written to be realistic. A free Cisco DevNet Sandbox, a Linux virtual machine, and Wireshark with sample captures are ideal.
3. Take each Knowledge Check with the answers covered, then review the explanations for anything you miss — the explanations teach as much as the questions.
4. Revisit the Summary and Key Takeaways boxes for fast revision before exam day.

The 200-201 Exam Blueprint

The exam is organized into five official domains, each carrying a percentage weight. This book devotes one chapter to each domain, in the blueprint's own order, so your study maps one-to-one onto what Cisco tests. The table below shows each chapter, its official domain, and weight.

Book chapter	Official exam domain	Weight
Chapter 1	1.0 Security Concepts	20%
Chapter 2	2.0 Security Monitoring	25%
Chapter 3	3.0 Host-Based Analysis	20%
Chapter 4	4.0 Network Intrusion Analysis	20%
Chapter 5	5.0 Security Policies and Procedures	15%

The exam is a **120-minute** assessment. Because Security Monitoring (Chapter 2) carries the most weight at 25%, give it proportional study time — but note that the four analysis-heavy domains (Chapters 2–4) together make up 65% of the exam and reward the kind of hands-on practice with logs, packets, and tools that this book emphasizes.

> ✅ **Exam Tip — Always confirm the current blueprint**
>
> Cisco can revise exam objectives at any time. Before you book the exam, download the latest official 200-201 exam topics from Cisco's certification site and confirm nothing has shifted since this edition (2026).

Table of Contents

Chapter 1 — Security Concepts

Every security operations career starts with vocabulary. Before you can read an alert, triage an incident, or argue with a firewall, you need to know exactly what security people mean when they say **threat**, **vulnerability**, **risk**, or **defense in depth** — because the 200-201 exam tests these words with surgical precision, and so will every interview you ever sit.

This chapter covers **Domain 1 of the exam blueprint, worth 20% of your score**. It is the foundation everything else stands on: the CIA triad, the families of security products you will meet in a SOC (Security Operations Center — the team that monitors and defends an organization around the clock), the working vocabulary of security operations, access control models, the CVSS scoring system, and the two great philosophies of detection — rules versus behavior.

Sub-section	Focus	Blueprint topics
1.1	The CIA triad; risk, threat, vulnerability, and exploit	1.1, 1.4
1.2	Security deployments: network, endpoint, SIEM/SOAR, cloud	1.2
1.3	The SOC vocabulary: TI, hunting, threat modeling, DevSecOps	1.3
1.4	Defense in depth and access control models	1.5, 1.6
1.5	CVSS, visibility, data loss, 5-tuple, detection approaches	1.7 – 1.11

> ✅ **Exam Tip — How Domain 1 questions look**
>
> Domain 1 questions are almost all **"compare" and "describe"** questions: two terms that sound alike, four answers that differ by one word. The winning strategy is to learn each term as a contrast pair — threat **vs.** vulnerability, SIEM **vs.** SOAR, MAC **vs.** DAC — exactly as this chapter presents them.

1.1 The CIA Triad, Risk, Threat, Vulnerability, and Exploit

The CIA triad — the three promises of security

Imagine a bank vault. The bank makes three promises about what is inside: nobody unauthorized can look at it, nobody can tamper with it, and the rightful owner can get to it whenever the bank is open. Information security makes exactly the same three promises about data, and we call them the **CIA triad**: **Confidentiality**, **Integrity**, and **Availability**.

- **Confidentiality** — data is readable only by those authorized to read it. Enforced with **encryption**, access controls, and data classification. Violated by: a database breach that leaks customer records, a packet sniffer on an unencrypted link, a stolen unencrypted laptop.
- **Integrity** — data is accurate and has not been altered by unauthorized parties. Enforced with **hashing** (a one-way mathematical fingerprint of data — change one bit and the fingerprint changes completely), digital signatures, and version control. Violated by: an attacker changing a payment account number in transit, malware modifying a system file.
- **Availability** — data and services are reachable when legitimate users need them. Enforced with redundancy, backups, failover, and anti-DDoS measures. Violated by: a **denial-of-service attack**, ransomware encrypting your files, or simply a failed disk with no backup.

Every security control you will ever deploy protects at least one leg of the triad, and every attack you will ever investigate violates at least one. Train yourself to ask, for any incident: which letter of CIA did this hurt?

Triad goal	Typical control	Classic attack against it
Confidentiality	Encryption (TLS, disk encryption), ACLs	Sniffing, data breach, shoulder surfing
Integrity	Hashing (SHA-256), digital signatures	Man-in-the-middle data tampering, file modification
Availability	Redundancy, backups, DDoS protection	DoS/DDoS, ransomware, sabotage

> 💡 **Did you know? — The evil twin: the DAD triad**
>
> Each CIA goal has a named opposite, together called the **DAD triad**: **D**isclosure breaks confidentiality, **A**lteration breaks integrity, and **D**estruction (or denial) breaks availability. Some exam questions describe an attack and ask which CIA property was violated — mapping through DAD makes the answer instant.

Threat, vulnerability, exploit — three words that are not synonyms

Picture your house. The **vulnerability** is the broken lock on the back door — a weakness in your defenses. The **threat** is the burglar walking the neighborhood — anything with the potential to cause harm. The **exploit** is the burglar's crowbar technique that actually takes advantage of the broken lock — the method or code used against a specific vulnerability. No burglar? The broken lock is still a vulnerability, but there is no active threat. Solid lock? The burglar remains a threat, but has no vulnerability to exploit.

- **Vulnerability** — a weakness in a system, configuration, or process (an unpatched server, a default password, a missing input check). Catalogued publicly as **CVEs** (Common Vulnerabilities and Exposures — a dictionary of publicly known flaws, each with an ID like CVE-2026-12345).
- **Threat** — any potential danger to an asset: a hacker group, malware, a careless employee, even a flood. The person or group behind an intentional threat is the **threat actor**.
- **Exploit** — the concrete mechanism (code, command sequence, technique) that takes advantage of a vulnerability. A vulnerability with a publicly available exploit is far more dangerous than one without.

Risk — where threat and vulnerability meet an asset

Risk is the likelihood that a threat will exploit a vulnerability, multiplied by the impact if it does. It is the business word in the set: executives don't buy firewalls because of exploits, they buy them to reduce risk. The exam expects three risk activities:

- **Risk assessment** — the process of identifying assets, their vulnerabilities, the threats against them, and estimating likelihood and impact.
- **Risk scoring / risk weighting** — turning that estimate into a number (or High/Medium/Low) so risks can be compared and prioritized. Weighting means giving some factors — say, asset criticality — more influence on the final score.
- **Risk reduction** (mitigation) — applying controls to lower likelihood (patching, hardening) or impact (backups, segmentation). Risk is rarely eliminated; it is reduced to an accepted level.

A minimal risk-scoring model in Python — the same logic, however dressed up, sits inside every GRC tool

```
# risk_score.py - likelihood x impact, weighted by asset criticality
assets = [
    # name,            likelihood(1-5), impact(1-5), criticality weight
    ("Public web server",        4,          4,           1.5),
    ("HR database",              2,          5,           2.0),
    ("Lab switch",               3,          2,           0.5),
]

for name, likelihood, impact, weight in assets:
    score = likelihood * impact * weight
    level = "HIGH" if score >= 20 else "MEDIUM" if score >= 10 else "LOW"
    print(f"{name:20s} risk={score:5.1f}  -> {level}")

# Output:
# Public web server    risk= 24.0  -> HIGH
# HR database          risk= 20.0  -> HIGH
# Lab switch           risk=  3.0  -> LOW
```

> ⚠️ **Common Pitfall — "Threat" and "vulnerability" swapped in answers**
>
> The most common Domain 1 trap is an answer that defines a vulnerability using threat language or vice versa. Anchor on this: a **vulnerability is internal** (a weakness you have); a **threat is external** (a danger that exists whether or not you are weak). Risk only exists where the two overlap on an asset that matters.

🌍 From the Field — How a SOC actually uses these words

In a real ticket you will write: “Threat actor scanned our range (threat), found an unpatched Apache server (vulnerability, CVE-XXXX-YYYY), and ran a public exploit to gain a shell (exploit). Business risk: high — the server holds customer PII.” Four terms, one sentence, and management instantly understands the severity. Precision with vocabulary is not pedantry — it is how SOC analysts communicate.

Summary

The CIA triad defines security's three promises: confidentiality (only authorized eyes), integrity (no unauthorized changes), availability (usable when needed), with the DAD triad naming their violations. A vulnerability is an internal weakness; a threat is an external potential for harm; an exploit is the concrete technique that uses a vulnerability; and risk is the likelihood-times-impact consequence of a threat meeting a vulnerability on an asset. Organizations manage risk through assessment, scoring/weighting, and reduction rather than elimination.

Key Takeaways

CIA = Confidentiality, Integrity, Availability — every control protects one, every attack violates one. **Encryption → confidentiality; hashing → integrity; redundancy/backups → availability. Vulnerability = internal weakness; threat = external danger; exploit = the technique; risk = likelihood × impact. Risk is reduced, not eliminated** — assessment finds it, scoring ranks it, mitigation lowers it. **DAD (Disclosure, Alteration, Destruction)** are the named violations of C, I, A.

Knowledge Check — 1.1 The CIA Triad and Core Terms

Q1. An attacker intercepts unencrypted traffic and reads usernames and passwords. Which element of the CIA triad is violated?

- **A.** Integrity
- **B.** Availability
- **C.** Confidentiality
- **D.** Authentication

Correct answer: C. Reading data you are not authorized to see is a disclosure — a confidentiality violation. Integrity would require altering data; availability would require making it unreachable. Authentication is not part of the CIA triad.

Q2. Which statement best defines a vulnerability?

- **A.** A weakness in a system or its configuration that could be taken advantage of
- **B.** Any potential danger that could cause harm to an asset
- **C.** The code or technique used to compromise a system
- **D.** The likelihood that an asset will be damaged

Correct answer: A. A vulnerability is the internal weakness itself. The “potential danger” is the threat, the “code or technique” is the exploit, and the “likelihood × impact” is the risk.

Q3. A company applies critical patches within 48 hours and segments its network to limit blast radius. Which risk activity does this describe?

- **A.** Risk assessment

B. Risk scoring

C. Risk acceptance

D. Risk reduction

Correct answer: D. Applying controls that lower likelihood (patching) or impact (segmentation) is risk reduction/mitigation. Assessment identifies risk, scoring ranks it, and acceptance is choosing to live with it unchanged.

Q4. Ransomware encrypts a hospital's patient records, making them unusable until a ransom is paid. Which CIA property is primarily attacked?

A. Confidentiality

B. Availability

C. Non-repudiation

D. Integrity

Correct answer: B. The defining harm of ransomware is that legitimate users can no longer access their own data — an availability attack (modern ransomware may also steal data, adding a confidentiality breach, but the primary classic impact is availability).

Q5. Which combination correctly pairs the term with its example?

A. Threat: an unpatched server; vulnerability: a criminal group; exploit: a risk score

B. Vulnerability: code abusing a flaw; exploit: an unpatched server; threat: a backup policy

C. Threat: a criminal group scanning internet ranges; vulnerability: an unpatched server; exploit: code abusing the missing patch

D. Exploit: a careless employee; threat: a missing patch; vulnerability: attack code

Correct answer: C. The criminal group is the external danger (threat), the unpatched server is the internal weakness (vulnerability), and the code that abuses the missing patch is the exploit. All other options scramble the definitions.

1.2 Security Deployments: From Firewalls to SIEM, SOAR, and the Cloud

Three places to stand guard: network, endpoint, application

Think of an airport. Perimeter fences and passport control guard the whole site; an air marshal protects one specific plane; and the cockpit door protects one specific function. Security products divide the same way — by **where they stand guard**:

- **Network security systems** sit in or beside the traffic path and protect everything behind them: firewalls, **IDS/IPS** (Intrusion Detection/Prevention Systems — sensors that match traffic against attack signatures; detection alerts, prevention also blocks), VPN concentrators, email and web security gateways.
- **Endpoint security systems** run on the device itself — laptop, server, phone: antivirus/antimalware, host-based firewalls, **EDR** (Endpoint Detection and Response — a recorder and remote-control agent for hosts), disk encryption.
- **Application security systems** protect one application's logic and data: **WAF** (Web Application Firewall — inspects HTTP requests for attacks like SQL injection), code analysis tools, runtime application self-protection.

> ✅ **Exam Tip — Pick the layer from the wording**
>
> If the question says “inspects traffic between networks” → network security. “Installed on each host” → endpoint. “Protects against SQL injection / inspects HTTP payloads” → application (WAF). The exam loves making you place a product at the right layer.

Agent-based vs. agentless protection

An **agent** is a small piece of software installed on the monitored device that reports in and enforces policy. **Agent-based** protection gives deep visibility (processes, memory, files) and works even off-network, but you must deploy and update the agent on every host. **Agentless** protection observes from outside — via network traffic, APIs, WMI/SSH queries, or hypervisor introspection — so there is nothing to install, which is perfect for devices you cannot install on (IoT, printers, appliances, unmanaged guests), at the price of less depth.

	Agent-based	Agentless
Visibility	Deep: processes, memory, files, registry	Shallower: network behavior, API/remote queries
Deployment	Install + maintain software on every host	Nothing installed on targets
Coverage	Only hosts that can run the agent	Anything visible on the network/API
Offline devices	Keeps protecting off-network	Blind when device leaves the observed path

Legacy antivirus vs. modern antimalware

Legacy antivirus is a signature librarian: it keeps a catalogue of fingerprints (hashes, byte patterns) of known malware and blocks exact matches. Fast and cheap, but blind to brand-new (**zero-day**) or self-modifying (**polymorphic**) malware — change one byte and the fingerprint no longer matches. **Modern antimalware / next-gen AV** adds **heuristics** (suspicious traits), **behavioral analysis** (watch what the program does — "it just encrypted 500 files in 10 seconds"), machine learning, sandboxing, and cloud reputation lookups, so it can catch malware nobody has ever named.

SIEM, SOAR, and log management — the SOC's nervous system

Every device in your network produces logs. Three product families deal with them, and the exam will test the difference:

- **Log management** — the warehouse: collect, store, index, and search logs at scale, with retention for compliance. No real analytics; it answers "show me the logs."
- **SIEM** (Security Information and Event Management) — the analyst's brain amplifier: aggregates logs from many sources, **normalizes** them into a common format, **correlates** events across sources ("5 failed logins on the VPN and then a success from a new country"), and raises **alerts** with dashboards and reports. SIEM is about detection and visibility.
- **SOAR** (Security Orchestration, Automation and Response) — the robot hands: takes alerts (often from the SIEM) and executes **playbooks** — automated workflows that enrich the alert (look up the IP's reputation), contain the threat (isolate the host, block the hash), open tickets, and notify humans. SOAR is about response and automation.

> 🌍 **From the Field — A day in the pipeline**
>
> A real chain looks like this: a domain controller logs 200 failed Kerberos logins → log management stores them → the SIEM correlation rule "N failures + 1 success within 5 min" fires an alert → the SOAR playbook auto-pulls the user's recent sign-ins, checks the source IP against threat intelligence, disables the account, and posts a summary into the SOC channel — all before a human opens the ticket. The analyst starts with context instead of raw logs.

> 💡 **Did you know? — Why normalization matters**
>
> A Cisco ASA, a Windows DC, and a Linux server all describe a "failed login" in completely different formats. SIEM normalization maps them all to common fields (user, source IP, action, outcome) — which is the only reason cross-source correlation rules are even possible.

Securing containers, VMs, and the cloud

A **virtual machine (VM)** emulates a whole computer with its own OS on top of a **hypervisor**; a **container** packages just an application and its dependencies while sharing the host's OS kernel. That sharing makes containers light and fast — and means one kernel vulnerability can endanger every container on the host. Container security therefore focuses on **image scanning** (check the container image for known CVEs before it runs), registry hygiene, runtime monitoring, and host/kernel hardening; VM security looks more like traditional server security plus hypervisor patching and VM isolation.

Cloud security deployments follow the **shared responsibility model**: the provider secures the cloud infrastructure (physical hosts, hypervisors); the customer secures what they put in it (data, identities,

configurations) — with the split moving as you go IaaS → PaaS → SaaS (Infrastructure/Platform/Software as a Service — renting raw machines, a managed platform, or a finished application, respectively). Cloud-specific tooling includes **CASB** (Cloud Access Security Broker — a policy checkpoint between users and cloud apps for visibility, DLP, and compliance) and **CSPM** (Cloud Security Posture Management — continuously hunts for misconfigurations like public S3 buckets).

> ⚠️ **Common Pitfall — "The cloud provider secures everything"**
>
> The single most-failed cloud idea: in every cloud model the **customer always owns the security of their data, identities, and access configuration**. A world-class provider cannot save you from a database you exposed to the internet. If an exam scenario blames an exposed cloud storage bucket, the responsible party is the customer.

Summary

Security products are classified by where they stand: network systems protect everything behind a chokepoint, endpoint systems protect the host they run on, and application systems (like WAFs) protect one application's logic. Protection is delivered agent-based (deep, installed, works offline) or agentless (nothing to install, network/API-observed). Legacy AV matches known signatures, while modern antimalware adds heuristics, behavior, ML, and sandboxing to catch the unknown. Log management stores events, SIEM correlates them into alerts, and SOAR automates the response with playbooks. Containers share the host kernel and demand image scanning and runtime controls, and cloud security always splits duties via the shared responsibility model — with the customer forever owning data, identity, and configuration.

Key Takeaways

Network / endpoint / application = protect everything behind me / this host / this app. **Agent-based = deep + offline; agentless = no install, shallower. Legacy AV = known signatures only; NGAV/antimalware = behavior, heuristics, ML, zero-days. Log mgmt stores, SIEM detects (correlates + alerts), SOAR responds (playbooks + automation). Containers share the kernel → scan images, harden the host. Shared responsibility: customer always owns data, identity, configuration.**

Knowledge Check — 1.2 Security Deployments

Q1. Which platform aggregates logs from many sources, normalizes them, correlates events, and generates alerts for analysts?

- **A.** SOAR
- **B.** SIEM
- **C.** Log management
- **D.** CASB

Correct answer: B. Correlation + alerting on aggregated, normalized logs is the definition of a SIEM. SOAR automates the response to alerts, log management only stores/searches, and a CASB governs cloud application use.

Q2. A SOC wants failed-login alerts to automatically trigger account disablement, IP reputation lookups, and ticket creation via playbooks. Which technology provides this?

- **A.** SIEM
- **B.** IPS

C. Log management

D. SOAR

Correct answer: D. Automated playbook-driven response and orchestration across tools is SOAR. The SIEM raises the alert; SOAR acts on it.

Q3. Which is an advantage of agentless protection over agent-based protection?

A. It can cover devices on which no software can be installed

B. It provides deeper visibility into processes and memory

C. It continues protecting laptops when they leave the network

D. It can quarantine individual files on the host

Correct answer: A. Agentless monitoring observes via network/APIs, so it covers IoT, appliances, and unmanaged devices where agents are impossible. Deep host visibility, offline protection, and on-host actions are agent-based strengths.

Q4. Why can legacy signature-based antivirus miss polymorphic malware?

A. Polymorphic malware only travels over encrypted channels

B. Legacy antivirus cannot scan executable files

C. The malware changes its code so its fingerprint no longer matches any known signature

D. Signatures only exist for Linux binaries

Correct answer: C. Signatures are fingerprints of known samples; polymorphic malware mutates with each copy, producing a new fingerprint that matches nothing in the database. Behavioral/heuristic engines were invented exactly for this gap.

Q5. Under the cloud shared responsibility model, which item is the customer's responsibility in all service models?

A. Physical security of the data center

B. Security of their data, identities, and access configuration

C. Patching the provider's hypervisors

D. Maintaining the provider's storage hardware

Correct answer: B. Data, identity, and configuration never transfer to the provider, regardless of IaaS/PaaS/SaaS. Physical facilities, hypervisors, and hardware are always the provider's side.

1.3 The SOC Vocabulary: Intelligence, Hunting, Modeling, and DevSecOps

Threat intelligence and threat hunting — knowing vs. seeking

Threat intelligence (TI) is processed, contextualized knowledge about threats: who the actors are, what tools and infrastructure they use, and — most practically — lists of **IOCs** (Indicators of Compromise: malicious IPs, domains, file hashes, URLs) your tools can match against. Raw data (an IP address) becomes intelligence only when context is added ("this IP is a C2 server for group X, active since May"). TI arrives via **feeds**, often in the **STIX** format over the **TAXII** transport protocol — think "STIX is the language, TAXII is the postal service."

Threat hunting is the opposite of waiting for an alert: a skilled analyst forms a **hypothesis** ("if an actor were abusing PowerShell here, I would see X") and proactively searches logs and endpoints for evidence that automated detection missed. Hunting assumes the attacker is already inside; alerts assume your rules already know the attack.

> ✅ **Exam Tip — Reactive vs. proactive**
>
> Alert triage is **reactive** (the tool found something, you respond). Threat hunting is **proactive and hypothesis-driven** (no alert exists; you go looking). If the question says "without any alert, an analyst searches for signs of…" — that is hunting.

Malware analysis and reverse engineering

Malware analysis answers "what does this sample do?" in two complementary ways. **Static analysis** examines the file without running it: hashes, embedded strings, imported functions, packers. **Dynamic analysis** runs it in an instrumented, isolated environment — a **sandbox** or **detonation chamber** — and records every file write, registry change, process spawn, and network connection. **Reverse engineering** is the deepest static form: disassembling or decompiling the binary to reconstruct its logic, used when you must understand exactly how the malware works (e.g., to build a decryptor or extract its C2 protocol).

Threat actors — know your adversary

A **threat actor** is the person or group behind a threat. Categories differ by motivation and capability, and exam questions describe one and ask you to name it:

Actor	Motivation	Capability
Script kiddie	Curiosity, bragging	Low — runs others' tools
Hacktivist	Ideology, protest	Low–medium — defacement, DDoS
Organized crime	Money (ransomware, fraud, theft)	High — professional, well-funded
Nation-state / APT	Espionage, sabotage, geopolitics	Very high — patient, custom tooling
Insider	Revenge, profit, or pure accident	Varies — already has legitimate access

> 💡 **Did you know? — Why "APT" means patience**
>
> **APT** stands for Advanced Persistent Threat: advanced (custom malware, zero-days), persistent (months or years of quiet access, not smash-and-grab), threat (a funded, organized group — typically nation-state sponsored). The defining trait the exam tests is the long-term, stealthy persistence.

Run book automation, anomaly detection, threat modeling, DevSecOps

- **Run book automation (RBA)** — a **run book** is a documented, step-by-step procedure for a known situation ("phishing email reported: extract URL, check reputation, search who else received it, quarantine"). RBA encodes those steps as automated workflows so they execute consistently and instantly — the conceptual ancestor of SOAR playbooks.
- **Sliding window anomaly detection** — instead of comparing traffic to a fixed baseline, the system continuously recomputes "normal" from the **most recent window of time** (say, the last 30 days) and flags what deviates from that. The baseline slides forward, so gradual legitimate change (a growing company) is absorbed, while sudden change (a 3 a.m. 10 GB upload) stands out.
- **Threat modeling** — a structured design-time exercise: diagram the system, enumerate what can go wrong, prioritize, and plan mitigations — before the attacker does it for you. The classic framework is **STRIDE** (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege), and outputs feed risk assessment.
- **DevSecOps** — the practice of integrating security into every stage of the DevOps pipeline ("shift left" — move security earlier in development): automated code scanning, dependency checks, container image scanning, and policy-as-code in CI/CD, making security a shared, automated responsibility instead of a final gate.

DevSecOps in one glance — a CI pipeline stage that fails the build on critical vulnerabilities

```
# .gitlab-ci.yml (excerpt) - security gate inside the pipeline
stages: [build, security, deploy]

dependency_scan:
  stage: security
  script:
    - pip install safety
    - safety check -r requirements.txt --full-report   # known CVEs in libs

container_scan:
  stage: security
  script:
    - trivy image --severity HIGH,CRITICAL --exit-code 1 myapp:latest
  # exit-code 1 on findings => the build FAILS before anything ships
```

> 🌍 **From the Field — Shift-left economics**
>
> Fixing a vulnerability found by a pipeline scanner costs minutes of a developer's time. The same flaw found in production costs an emergency change window, possible incident response, and — if exploited — breach costs. That cost curve is the entire business case for DevSecOps, and why "shift left" became a mantra.

Summary

Threat intelligence is contextualized knowledge about adversaries, delivered as feeds of IOCs (STIX/TAXII); threat hunting is the proactive, hypothesis-driven search for intrusions no alert has caught. Malware analysis splits into static (examine without executing) and dynamic (detonate in a sandbox), with reverse engineering reconstructing binary logic in depth. Threat actors range from script kiddies through hacktivists and organized crime to nation-state APTs and insiders, distinguished by motivation and capability. RBA automates documented procedures, sliding-window detection re-baselines "normal" continuously, threat modeling (e.g., STRIDE) finds design flaws before attackers do, and DevSecOps shifts automated security checks left into the CI/CD pipeline.

Key Takeaways

TI = processed knowledge + IOCs (STIX format, TAXII transport). Hunting = proactive + hypothesis-driven; triage = reactive. Static analysis = don't run it; dynamic = sandbox/detonation chamber; reverse engineering = reconstruct the code. APT = advanced, persistent (long-term stealth), state-grade. Insiders already have legitimate access — hardest to detect. RBA automates run books; sliding window = moving baseline; STRIDE = threat modeling; DevSecOps = security shifted left into CI/CD.

Knowledge Check — 1.3 SOC Vocabulary

Q1. An analyst, without any triggering alert, forms the hypothesis that attackers may be abusing scheduled tasks and searches endpoint logs for evidence. What is this activity called?

- **A.** Alert triage
- **B.** Vulnerability scanning
- **C.** Run book automation
- **D.** Threat hunting

Correct answer: D. Proactive, hypothesis-driven searching with no prior alert is threat hunting. Triage reacts to alerts, vulnerability scanning looks for weaknesses (not active intrusions), and RBA automates procedures.

Q2. Which threat actor is best described as well-funded, focused on long-term stealthy access, and typically sponsored by a government?

- **A.** Script kiddie
- **B.** Nation-state APT
- **C.** Hacktivist
- **D.** Opportunistic insider

Correct answer: B. Advanced Persistent Threats are the signature of nation-state sponsorship: patient, custom-tooled, espionage-driven. Script kiddies lack skill, hacktivists are ideology-driven and noisy, insiders act from within.

Q3. What distinguishes dynamic malware analysis from static analysis?

- **A.** The analyst examines strings and imports without running the file
- **B.** It can only be performed on source code
- **C.** The sample is actually executed in an isolated, instrumented environment
- **D.** It requires disassembling the binary into assembly

Correct answer: C. Dynamic = detonate and observe behavior (sandbox). Examining strings/imports without execution is static analysis, and disassembly is reverse engineering — a deep static technique.

Q4. Which statement describes sliding window anomaly detection?

A. The behavioral baseline is continuously recomputed from the most recent period, and deviations from it are flagged

B. Traffic is compared against a fixed baseline captured once at deployment

C. Packets are matched against a database of known attack signatures

D. Alerts fire only when a static threshold is crossed

Correct answer: A. The "sliding" part is the moving time window from which normal behavior is constantly re-learned. Fixed baselines, signatures, and static thresholds are precisely what it is not.

Q5. A development team adds automated dependency and container-image vulnerability scanning to the CI/CD pipeline so builds fail on critical findings. Which practice is this?

A. Threat hunting

B. Reverse engineering

C. Mandatory access control

D. DevSecOps

Correct answer: D. Embedding automated security controls into the development pipeline — shifting security left — is DevSecOps.

1.4 Defense in Depth and Access Control Models

Defense in depth — the castle, not the eggshell

A medieval castle never relied on one wall. It had a moat, an outer wall, an inner wall, a keep, and guards at every gate — so a breach of any single defense left the attacker facing the next one. **Defense in depth** applies the same principle to security: deploy **multiple, layered, independent controls**, so no single failure is fatal. The opposite is the "eggshell": hard perimeter, soft inside — crack the firewall and everything is exposed.

- **Typical layers** (outside-in): perimeter (edge firewall, DDoS protection) → network (segmentation, IDS/IPS, NAC) → host (hardening, EDR, host firewall) → application (WAF, secure coding, input validation) → data (encryption, DLP, backups) — wrapped in policies, awareness training, and physical security.
- **Supporting principles**: **least privilege** (every user/process gets the minimum access needed), **separation of duties** (no single person controls a critical process end-to-end), and **diversity of controls** (administrative + technical + physical, so one bypass technique doesn't defeat them all).

> 💡 **Did you know? — Layers must be independent**
>
> Defense in depth only works if layers fail independently. Two firewalls from the same vendor with the same misconfiguration are one layer wearing two hats. True depth mixes control types — a phishing email must beat the mail filter, then user training, then EDR, then egress filtering, then DLP.

AAA — the gatekeeper's three questions

Before models, the framework: **AAA**. **Authentication** — who are you? (prove identity: password, token, biometric, certificate). **Authorization** — what may you do? (the permissions granted to that proven identity). **Accounting** — what did you do? (log the session and actions for audit). In Cisco networks AAA is typically delivered by **RADIUS** or **TACACS+** servers (e.g., Cisco ISE). Every access control model below is a different way of answering the authorization question.

The access control models

Discretionary access control (DAC) — the owner of a resource decides who gets access, at their discretion. This is everyday computing: you create a file, you share it. Flexible, but security depends on every owner's judgment. Classic example: Linux file permissions set by the file's owner with `chmod`.

DAC in practice — the file's owner grants and revokes access

```
$ ls -l report.txt
-rw-r----- 1 jojko analysts 5120 Jun 10 09:00 report.txt
#  ^owner jojko decides: group 'analysts' may read, others get nothing
$ chmod o+r report.txt      # owner discretionarily grants read to everyone
$ chown maria report.txt    # ownership (and the discretion) can be transferred
```

Mandatory access control (MAC) — the system, not the owner, enforces access based on **security labels** (Top Secret, Secret, Confidential…) and **clearances**, under a central policy nobody can override at will. A

user with Secret clearance cannot open a Top Secret file even if its creator wants to share it. Used by military/government systems and OS mechanisms like SELinux.

Nondiscretionary access control — the umbrella for models where a central authority (not the resource owner) defines access rules. Role-based access control is the textbook example, and the exam treats "nondiscretionary" as "centrally administered, typically via roles."

- **Role-based access control (RBAC)** — permissions attach to **roles** ("Helpdesk", "HR-Manager"), users are assigned roles, and access follows the role. New hire in accounting? Assign the Accounting role — done. Scales beautifully and maps to job functions.
- **Rule-based access control** — access decided by **system-wide if/then rules** applied to everyone regardless of identity, the canonical example being a **firewall ACL**: "permit tcp any host 10.1.1.10 eq 443."
- **Time-based access control** — access permitted only during defined **time windows** ("contractors authenticate 08:00–18:00 weekdays"), on Cisco devices implemented with `time-range` objects referenced by ACLs.
- **Attribute-based access control (ABAC)** — the most granular: decisions evaluate **attributes** of the user (department, clearance), the resource (classification), the action, and the environment (time, location, device posture). "Managers from Finance may open payroll files from corporate devices during business hours" — four attribute checks in one policy.

Model	Who decides	Memory hook
DAC	The resource owner	"My file, my call" — chmod
MAC	The system, via labels + clearances	Military labels; owner can't override
Nondiscretionary	A central authority	Centrally administered (roles)
RBAC	Central admin via job roles	Access follows the role
Rule-based	System-wide if/then rules	Firewall ACL for everyone
Time-based	Rules bound to clock windows	Cisco time-range + ACL
ABAC	Policy over many attributes	User+resource+action+environment

⚠️ **Common Pitfall — Rule-based is not role-based**

One letter, two models — and a deliberately cruel distractor pair. **Rule**-based = identity-blind if/then conditions (ACLs); **role**-based = permissions through assigned job roles. Read the stem twice; the exam counts on you skimming.

✅ **Exam Tip — Spot the model from one phrase**

"Owner decides" → DAC. "Security labels / clearance" → MAC. "Assigned to the Helpdesk role" → RBAC. "ACL applies to all traffic" → rule-based. "Only during business hours" → time-based. "Department + device posture + location evaluated together" → ABAC.

Summary

Defense in depth layers multiple independent controls — perimeter, network, host, application, data, plus people and process — so a single failure never equals a breach, supported by least privilege and

separation of duties. AAA frames access as authentication (who), authorization (what), accounting (what happened), delivered in Cisco environments by RADIUS/TACACS+. The authorization models differ by who decides: owners (DAC), the system via labels (MAC), or a central authority (nondiscretionary), with RBAC granting through roles, rule-based applying universal conditions like ACLs, time-based gating by clock windows, and ABAC evaluating attributes of user, resource, action, and environment together.

Key Takeaways

Defense in depth = layered, independent, diverse controls; no single point of failure. Least privilege = minimum necessary access, always. AAA: authenticate → authorize → account (RADIUS/TACACS+). DAC = owner decides; MAC = labels + clearances, system decides; RBAC = role decides; rule-based = if/then for everyone; time-based = clock windows; ABAC = multi-attribute policies. Rule-based ≠ role-based — the exam's favorite one-letter trap.

Knowledge Check — 1.4 Defense in Depth and Access Control

Q1. A file's creator uses chmod to grant read access to a colleague. Which access control model is in use?

- **A.** Discretionary access control
- **B.** Mandatory access control
- **C.** Role-based access control
- **D.** Attribute-based access control

Correct answer: A. The owner granting access at their own discretion is the definition of DAC. MAC would forbid owner-granted sharing across labels; RBAC and ABAC are centrally administered.

Q2. In which model does the operating system enforce access through security labels and clearances that even the data owner cannot override?

- **A.** Discretionary access control
- **B.** Rule-based access control
- **C.** Mandatory access control
- **D.** Time-based access control

Correct answer: C. Labels (Secret, Top Secret) plus clearances under an unoverridable central policy is MAC — the military/SELinux model.

Q3. Access to a payroll system is granted only when the user is in the Finance department, on a compliant corporate device, and connecting during business hours from the office network. Which model evaluates all these factors?

- **A.** Role-based access control
- **B.** Attribute-based access control
- **C.** Discretionary access control
- **D.** Mandatory access control

Correct answer: B. Combining user, resource, and environmental attributes (department, device posture, time, location) in one policy decision is ABAC — the most granular model.

Q4. Within AAA, which component answers the question "what actions did the user perform during the session?"

- **A.** Authorization
- **B.** Authentication
- **C.** Auditing labels
- **D.** Accounting

Correct answer: D. Accounting records what an authenticated, authorized user actually did — the audit trail. Authentication proves identity; authorization sets permissions.

Q5. Which principle states that users and processes should receive only the minimum access required to perform their function?

- **A.** Least privilege
- **B.** Separation of duties
- **C.** Defense in depth
- **D.** Implicit deny

Correct answer: A. Minimum-necessary access is least privilege. Separation of duties splits critical processes among people; defense in depth layers controls; implicit deny is the default-block behavior at the end of an ACL.

1.5 CVSS, Visibility Challenges, Data Loss, the 5-Tuple, and Detection Philosophies

CVSS — a common language for "how bad is it?"

When a new vulnerability appears, every team on Earth asks the same question: how bad? The **Common Vulnerability Scoring System (CVSS)** answers it with a standardized 0.0–10.0 score built from defined metrics, so a "9.8 critical" means the same thing in Bratislava and in Tokyo. The exam tests the **base metrics** by name — learn what each one measures:

- **Attack Vector (AV)** — from where can it be exploited? **Network** (remotely over the internet — worst), Adjacent (same LAN), **Local** (needs local access), **Physical** (hands on the device — least severe).
- **Attack Complexity (AC)** — how hard is a successful attack beyond the attacker's control? **Low** (works reliably, point-and-shoot — worse) vs. **High** (requires special conditions, race wins, target-specific preparation).
- **Privileges Required (PR)** — what access must the attacker already have? **None** (worst), Low (user account), High (admin).
- **User Interaction (UI)** — must a victim do something (click a link, open a file)? **None** (worst — fully unattended, wormable) vs. Required.
- **Scope (S)** — does the exploit's impact stay inside the vulnerable component (**Unchanged**) or break out to affect other components (**Changed**), like a VM-escape that compromises the hypervisor?

Two further metric groups refine the base score over time. **Temporal metrics** change as the threat landscape evolves: Exploit Code Maturity (is a working exploit public?), Remediation Level (is there an official patch yet?), Report Confidence. **Environmental metrics** adapt the score to your organization: how critical are confidentiality/integrity/availability for the affected asset in your environment, with modified base metrics to reflect your compensating controls.

> ✅ **Exam Tip — Base vs. temporal vs. environmental**
>
> **Base** = intrinsic, never changes (AV, AC, PR, UI, Scope, C/I/A impact). **Temporal** = the world changed (exploit released, patch published). **Environmental** = your context (asset criticality, your controls). A question about "score changed after a public exploit appeared" is temporal; "score adjusted because the server holds critical data" is environmental.

> 💡 **Did you know? — Why "Scope: Changed" hurts so much**
>
> A scope change means the damage escapes the vulnerable component's own security authority — e.g., code in a sandboxed browser process compromising the OS. CVSS weights these flaws sharply higher because one foothold suddenly endangers a whole different trust zone.

The visibility problem — you can't defend what you can't see

Detection depends on **data visibility**, and each environment hides things differently. On the **network**, encryption (TLS everywhere) hides payloads from inspection, NAT hides true sources behind one address, and east-west traffic between hosts in the same segment may never cross a sensor. On the **host**, you only

see what an agent reports — unmanaged/BYOD/IoT devices are dark, and attackers using legitimate tools ("living off the land" — abusing built-in binaries like PowerShell so no malware ever touches disk) blend into normal noise. In the **cloud**, you don't own the infrastructure: visibility shrinks to whatever logs the provider exposes (and you must enable them), shadow IT spins up unsanctioned services, and ephemeral workloads may live and die between two log polls.

> 🌍 **From the Field — The encrypted-traffic dilemma**
>
> Around 95% of web traffic is now TLS-encrypted. SOCs choose between TLS interception (decrypt-inspect-re-encrypt proxies — powerful but a privacy, performance, and certificate-management burden) and **encrypted traffic analytics** — inferring maliciousness from metadata (packet sizes, timing, TLS fingerprints) without decrypting. Most mature environments mix both: decrypt where policy allows, analyze metadata where it doesn't.

Spotting data loss in traffic profiles

Data exfiltration — an attacker smuggling data out — leaves fingerprints in traffic profiles even when the content is hidden. A **traffic profile** is the statistical pattern of who talks to whom, how much, when, and over which protocols. Watch for:

- **Volume anomalies** — a workstation that normally uploads megabytes suddenly pushing gigabytes outbound; outbound/inbound byte ratios flipping.
- **Time anomalies** — large transfers at 03:00 from an office that sleeps at night.
- **Destination anomalies** — sustained flows to never-before-seen external IPs, rare countries, or personal cloud storage.
- **Protocol misuse** — data hidden in **DNS tunneling** (thousands of long, high-entropy DNS queries carrying encoded data to an attacker-controlled domain) or ICMP payloads; "DNS" traffic measured in megabytes is a flashing red light.

The 5-tuple — isolating a compromised host in a pile of logs

Every network conversation is uniquely identified by five fields — the **5-tuple**: **source IP, source port, destination IP, destination port, protocol**. When logs from many systems are dumped in front of you, the 5-tuple is your scalpel: filter on one suspect value (say, a known-malicious destination IP), and the internal source addresses that remain are your compromised candidates; pivot again on each source to reconstruct everything else it talked to.

A grouped log set — which internal host is compromised?

```
# firewall connection log (simplified)
# src_ip        src_port  dst_ip          dst_port  proto
10.10.5.23      51844     203.0.113.77    443       TCP
10.10.5.41      49733     198.51.100.9    53        UDP
10.10.5.23      51902     203.0.113.77    443       TCP
10.10.5.23      52011     203.0.113.77    8443      TCP
10.10.5.88      50122     93.184.216.34   443       TCP
10.10.5.23      52240     203.0.113.77    443       TCP

# TI feed says 203.0.113.77 is a known C2 server. Filter the 5-tuple:
```

```
$ grep "203.0.113.77" fw.log | awk '{print $1}' | sort | uniq -c
      4 10.10.5.23
# => 10.10.5.23 is beaconing to the C2 on 443/8443 - isolate that host.
```

Notice the pattern the 5-tuple revealed: **repeated, regular connections** from one internal host to one external IP — **beaconing**, the heartbeat of malware checking in with its **C2** (command-and-control) server. Source ports change every connection (they are ephemeral, picked from the high range); it is the stable source IP → destination IP:port relationship that convicts.

> ⚠️ **Common Pitfall — Don't pivot on the source port**
>
> Source ports are ephemeral — a new one per connection — so filtering on them tells you nothing. Investigations pivot on the stable fields: source IP, destination IP, destination port, protocol. An exam answer that "identifies the host by its source port" is wrong by design.

Rule-based vs. behavioral and statistical detection

Two philosophies power every detection product. **Rule-based (signature) detection** matches activity against **predefined patterns of known badness** — byte signatures, hash blocklists, exact correlation rules. It is precise, explainable, and low on false positives, but it cannot see an attack nobody has written a rule for. **Behavioral and statistical detection** first learns **normal** — baselines of users, hosts, and traffic — then flags **deviation**: a login from two countries in one hour, a server suddenly speaking a new protocol, traffic three standard deviations above its weekday mean. It can catch zero-days and insider misuse, but "unusual" is not always "malicious," so it pays in false positives and needs tuning time to learn the environment.

	Rule-based / signature	Behavioral / statistical
Detects	Known threats with written rules	Unknown threats, novel deviations
Zero-days	Missed (no signature exists)	Catchable (deviation from baseline)
False positives	Low — match is a match	Higher — unusual ≠ malicious
Needs	Constant signature updates	Learning period + tuning
Verdict clarity	Explains exactly which rule fired	"Anomalous" needs analyst interpretation

> ✅ **Exam Tip — Keyword radar for the two philosophies**
>
> "Signature", "known patterns", "database of attacks" → rule-based. "Baseline", "deviation", "anomaly", "learns normal behavior" → behavioral/statistical. Mature SOCs run **both** — signatures for the known, behavior for the unknown — and the exam knows it.

Summary

CVSS standardizes severity with base metrics — attack vector, attack complexity, privileges required, user interaction, and scope — refined by temporal metrics as exploits and patches appear and environmental metrics for your own context. Visibility is fought for differently in network (encryption, NAT, east-west blind spots), host (agentless devices, living-off-the-land), and cloud (provider-controlled logging, shadow IT, ephemeral workloads). Data loss betrays itself in traffic profiles through

volume, timing, destination, and protocol anomalies such as DNS tunneling. The 5-tuple — source IP/port, destination IP/port, protocol — isolates compromised hosts from grouped logs, exposing C2 beaconing. Rule-based detection precisely matches known badness; behavioral and statistical detection baselines normal and flags deviation, trading false positives for the power to catch the unknown.

Key Takeaways

CVSS base: AV, AC, PR, UI, Scope — intrinsic and fixed; temporal = threat evolves; environmental = your context. Scope: Changed = impact escapes the vulnerable component — scored harsher. Visibility killers: encryption & NAT (network), unmanaged devices & LOTL (host), provider-dependent logs & shadow IT (cloud). Exfiltration clues: volume spikes, odd hours, new destinations, DNS/ICMP tunneling. 5-tuple = src IP, src port, dst IP, dst port, protocol — pivot on stable fields, never source port. Rules catch the known precisely; behavior catches the unknown noisily — run both.

Knowledge Check — 1.5 CVSS, Visibility, and Detection

Q1. Which CVSS base metric expresses whether the flaw can be exploited remotely across the internet versus requiring physical access?

- **A.** Attack Complexity
- **B.** Privileges Required
- **C.** Attack Vector
- **D.** Scope

Correct answer: C. Attack Vector ranges Network → Adjacent → Local → Physical. Complexity measures conditions beyond attacker control, PR measures pre-existing access, Scope measures impact escape.

Q2. A vulnerability's CVSS score is updated after a working exploit is published and before any vendor patch exists. Which metric group changed?

- **A.** Base metrics
- **B.** Environmental metrics
- **C.** Impact metrics
- **D.** Temporal metrics

Correct answer: D. Exploit code maturity and remediation level are temporal metrics — they track how the threat landscape evolves over time. Base metrics never change; environmental metrics reflect a specific organization.

Q3. Security analysts observe thousands of unusually long, high-entropy DNS queries from one workstation, totaling hundreds of megabytes. What does this traffic profile most likely indicate?

- **A.** Data exfiltration via DNS tunneling
- **B.** Normal DNS cache refresh
- **C.** A misconfigured NTP client
- **D.** An IPv6 transition mechanism

Correct answer: A. DNS should be tiny queries; megabytes of long, random-looking queries to one domain is the classic signature of DNS tunneling smuggling data out.

END OF FREE SAMPLE

You just read Chapter 1 in full — one of five complete chapters.

What the full guide contains

- **5 complete chapters**, one per official 200-201 exam domain, in the blueprint's own order and weighting.
- **25 in-depth sub-sections** — every concept taught from first principles with plain-English analogies.
- **125 Cisco-style practice questions** with full answer explanations, balanced across A/B/C/D.
- **Realistic commands and captures**: tcpdump, NetFlow, Wireshark, Windows Event IDs, Linux logs, regex, forensic imaging — written to try in a safe lab.
- **Hundreds of colored value boxes**: Did you know?, Exam Tip, Common Pitfall, and From the Field, plus Summary and Key Takeaways recaps for fast revision.
- A full **glossary**, exam **blueprint map**, and a chapter-by-chapter **wrap-up** that ties every topic into a real SOC analyst's mindset.

Get the complete guide

Search “CCNA Cybersecurity 200-201 Complete Learning Guide” by Jozef Baroš

Available as a styled, fully searchable PDF — study on any device.