Introduction to Cybersecurity

Notes

Security conversations go wrong when they stop at labels.

- “High risk.”
- “Vulnerable.”
- “Needs to be secured.”

Those statements don’t tell a team what to change, what to test, or what success looks like.

The goal here is to describe risk in a way that supports real decisions: prioritizing work, choosing controls, and being able to explain (and defend) why a fix is the proper fix.

The core idea

Security risk is not a score. It is a path to an asset.

Every meaningful security concern can be explained as:

Asset → Entry Point → Weak Spot → Damage

This model is used throughout.

The four parts

Asset (what must not be harmed)

An asset is a specific system object or function that matters.

Examples:

- A database table containing customer records
- A payroll calculation function
- A report generation pipeline

Not assets:

- Trust
- Reputation
- Compliance

Those are consequences. Always ask:

What is the thing that, if changed or exposed, causes the problem?

If you forget everything else, the asset is the thing in the system, not the feeling about it.

Entry Point (where typical activity begins)

An entry point is a routine, legitimate way to start an interaction.

Examples:

- A user logging in
- A scheduled background job
- A support workflow

Not entry points (attacks, not workflows):

- “SQL injection on the login form”
- “Privilege escalation via exploit”

Correct entry points (workflows):

- “User authentication workflow”
- “Administrative task scheduler”

Entry points are workflows, not attacks. They matter because of what they make reachable.

If you forget everything else, entry points are how typical work begins, not how attacks happen.

Weak Spot (the condition that lowers resistance)

A weak spot is a present-tense condition that exists right now.

Weak spots are conditions, not attack names.

Examples:

- Access is broader than responsibility
- Identities have authority without attribution
- Changes can occur without review or evidence
- Validation relies only on client-side enforcement
- State transitions lack authorization checks
- Dependencies can fail in ways that block required outcomes

Friction note: weak spots are often places where friction is too low for unauthorized use. The wrong action is cheap instead of expensive.

If you forget everything else, a weak spot is something you could point to and check today.

Damage (what actually goes wrong)

Damage is the observable outcome if the path completes.

Good damage descriptions:

- “Invoices contain incorrect totals.”
- “Private records are visible to unauthorized staff.”
- “Reports are not delivered before the deadline.”

Bad damage descriptions:

- “The system is compromised.”
- “This could be hacked.”

Ask:

If this happened, what would someone notice?

If you forget everything else, if you can’t describe what someone would notice, you don’t have damage yet.

Why systems drift

Most environments become risky without malicious intent.

Common reasons:

- A contractor needs temporary access, and it is never removed.
- A shared credential exists because rotation is painful.
- A migration leaves old paths active “just in case.”
- Automation grows faster than ownership.

These are ordinary decisions under time pressure. Security risk emerges when they accumulate.

CIA

Once you can describe damage clearly, you may want a shorthand for referring to the type of failure. In security work, the most common shorthand is CIA.

CIA is not a way to find problems. It is a way to categorize outcomes after you already understand the path.

CIA describes what kind of damage occurred:

- Confidentiality: the wrong party saw the information
- Integrity: information or behaviour is incorrect
- Availability: a system failed to deliver when needed

Rule: CIA labels apply only after you can describe the damage clearly.

A CIA label without a path describes symptoms rather than explaining the risk.

Observable vs. vague damage

Vague: “The database could be compromised.”
Observable: “Customer email addresses are visible to unauthorized staff accounts.”

Vague: “The service might go down.”
Observable: “Orders cannot be processed during peak hours, causing missed deliveries.”

The Acid Test

A claim is ready for review only if it passes all three tests:

- Concrete Test: Is the asset a specific thing?
- Current Test: Is the weak spot a real, present-tense condition?
- Witness Test: Could a non-technical manager recognize the damage?

If any test fails, rewrite only that part and recheck.

Examples

Shared Report Link Exposes Private Records

A customer support team uses an internal portal to view customer profiles and generate PDFs. To speed up collaboration, the portal allows staff to create “view-only” links to a customer report and paste them into chat. The link does not require a login and does not expire.

Customer profile PDFs

→ “Share report link” workflow

→ Report links are unexpired and unauthenticated

→ Unauthorized individuals can view a private customer report, leading to the disclosure of private customer details.

This is a confidentiality issue because the system continues to function, and no data is changed or destroyed. The failure is that information becomes visible to someone who should not have access to it. The observable damage is unauthorized viewing of private customer details.

Billing Rule Change Produces Incorrect Invoices

A billing service calculates invoice totals using a shared “pricing rules” configuration. A non-owner team can edit that configuration to “help” with last-minute changes. The service remains online and continues generating invoices.

Invoice totals in the billing system

→ Routine configuration update workflow

→ Pricing rules are editable by non-owners without review

→ Invoices are generated with incorrect totals, and customers receive invoices with incorrect totals that appear valid.

This is an integrity issue because the system remains available and produces output, but that output is no longer correct or trustworthy. The observable damage is that customers receive invoices with incorrect totals that appear valid.

Appointment Booking Fails Under Load

A clinic uses an online booking system. During a seasonal surge, many patients try to book at once. The booking service depends on a single scheduling database. Under load, requests time out and bookings fail.

Ability to book appointments on time

→ Standard booking workflow

→ Single scheduling database becomes a bottleneck under load

→ Bookings time out and appointments cannot be scheduled, so patients cannot complete bookings during peak demand.

This is an availability issue because the primary failure is not exposure or incorrectness. It is a failure to deliver a required function when needed. The observable damage is that patients cannot complete bookings during peak demand.

Shared Service Credential Enables Multiple Harms

Scenario A data analytics dashboard pulls metrics from multiple internal systems. To simplify integration, it uses a shared service credential stored in a shared configuration location used by several teams. The credential has broad permissions: it can read sensitive data and also write updates (for “data corrections”).

Analytics dashboard datasets

→ Routine access to shared configuration

→ Shared credentials are readable by too many identities and have broad permissions

→ An unauthorized person can view sensitive data, modify metrics and reports while they still appear legitimate, and trigger disruption while the credential is rotated and services are adjusted.

This is primarily an integrity issue because the system continues to operate and produce results. Yet its behaviour and outputs can be altered in unauthorized ways. Metrics and reports can be changed while still appearing legitimate. The same weak spot also harms confidentiality by enabling unauthorized access to sensitive data, and it harms availability because containment requires credential rotation and service changes that interrupt dashboards and reporting.

The evidence rule

If a weak spot allows an action to occur without reliable attribution, the damage is not just the event itself.

It also includes:

- inability to prove what happened
- inability to bound what else could have happened
- broader, more disruptive containment

Example:

- If multiple people share an admin account and records are changed, investigators cannot determine who acted or how to prevent it from happening again.

Judgment and prioritization

Not all paths matter equally. Consider:

- Proximity: How many steps from the entry to the asset?

- Blast radius: How many stakeholders are affected?
- Reversibility: Can the harm be undone?
- Detectability: Is the failure loud or silent?

Friction rule

If you forget everything else, good security makes the wrong thing expensive and the right thing cheap.

Good security increases the cost of unauthorized use by ~100× while increasing the cost of authorized use by <1.1×.