All this text wasn’t written by me (Tiago a.k.a TigaxMT).

All of it was transcribed by me from HackerOne video lessons.

All credits and thanks go to them.

Secure Architecture Review

A secure architecture is, at its core, one that acknowledge that bugs invariably exist and seek to limit their impact and value.
By accepting that all code is fallible, the priority can shift to minimizing damage when a component is compromised.

ARCHITECTURE REVIEW

This is other side of the coin from threat modeling. Rather than trying to determine how attackers could get in, you assume that they already have and determine what they can access.

For each component in a system, starting with externally-facing components and working inward:

1. Assume it is compromised
2. Determine what an attacker can directly reach from this point
3. Determine if this accessible data needs to be present or if it can be moved away from this component.

MAXIMIZE ISOLATION

If your webserver and database are both on one machine, this substantially drops the amount of effort required for an attacker to compromise data. As such, splitting these up can help limit attacker's access.

This is specially true in multi-tenant systems. If you have multiple independent application instances, they should each have their own database and credentials.
Additionally, consider the use of application containers like Docker to provide each tenant's application, further isolating them.

SECURE PASSWORD STORAGE

Credentials should never be stored in plain text or hashed using insecure algorithms. These should be stored using BCrypt, SCrypt, or other resilient password hashing algorithms.
Check out the Password Storage video (or transcription) for more information.

PRINCIPLE OF LEAST PRIVILEGE

The Principle of Least Privilege states that each user or service should have the lowest privilege level possible without impeding their functionality. This means that your web server should be running as a low-privilege user and your database's root user should be disabled.

AUDITABLILITY

In the event of a compromise, it is important to recognize this, prevent further compromise, and understand what has been accessed.
As such, it’s important to have centralized log services, to which each other component sends information.

Things that should be logged:

- Requests causing internal server errors
- Query failures
- Commands being run as web server or database users
- Attempted outgoing connections from the servers

It’s important to note, though, that this logging cannot be assumed to be 100% accurate. If a compromise has occurred, it’s important that you consider the possibility that the logging itself has been compromised in some way.
These logs will help you put the pieces together, though, and hopefully shut down attacks before they start.