Broken access control:

Permission and access issues in a system. For example, one user might not have correct access to the items they need access to or they might have too much accesses which allows them to do potential damage. In some cases there might a possibility for an attacker to gain access through a broken access control system without the correct credentials.

Example: https://www.capitalone.com/digital/facts2019/

Prevention: The first line of defense here is to implement a "least privilege" principle. This means only giving users access to what they absolutely need to do their jobs. Also, regular checks and audits of access control configurations are a must. Also test your system in the first place for obvious attack strategies.

Cryptographic Failures:

Cryptographic failures happen when encryption algorithms are weak or aren't implemented correctly (OWASP, 2021).

Example: https://wolfesystems.com.au/insights-from-the-2016-adult-friend-finder-breach/ (used weak SHA-1 encryption)

Prevention: Use strong encryption methods and implement them everywhere containing sensitive data.

Injection:

Injection vulnerabilities happen when user-supplied data is not sanitized and is used to modify the intended behavior of the application (OWASP, 2021). Essentially, a hacker is injecting malicious code into the system through a seemingly harmless input field.

Example: https://www.twingate.com/blog/tips/Heartland%20Payment%20Systems-data-breach (SQL Injection)

Prevention: Input validation and sanitization. User inputs should always be treated with suspicion. Parameterized queries and stored procedures can help prevent SQL injection, while other forms of injection should be handled through appropriate input validation techniques.

Insecure Design:

There are fundamental flaws in the architecture of the application/system from the start. It is much harder to fix vulnerabilities like this when they are found on the implementation phase of the system.

Example: https://en.wikipedia.org/wiki/2017_Equifax_data_breach (simply poor system design and not fixing them once found)

Prevention: Apply security at every stage of the development process, especially at the beginning. Threat modeling, audits and security reviews etc.

Security Misconfiguration:

Security misconfigurations are when security settings are not properly configured or are left at default settings.

Example: https://www.healthcareitnews.com/news/accenture-latest-breach-client-data-due-misconfigured-aws-server (Accenture left s3 bucket security settings at default settings)

Prevention: Use best practices in security configuration, e.g. custom strong passwords, keys, updating and patching the systems, turning off unnecessary features etc.

Vulnerable and Outdated Components:

Using software components, like libraries and frameworks, that are outdated or have known security vulnerabilities leaves the system open to attacks.

Example: https://en.wikipedia.org/wiki/2017_Equifax_data_breach (it was partly due to their failure to update an Apache Struts web server component with a well-known vulnerability)

Prevention: Regular vulnerability scanning, patching, and updates are very important. It's essential to keep track of the components you are using and their security status.

Identification and Authentication Failures:

These failures happen when authentication and session management are poorly implemented allowing users to gain access to system without proper verification of their identity.

Example: https://www.strongdm.com/what-is/snowflake-data-breach (Third party companies didn’t have MFA enabled)

Prevention: Implementing multi-factor authentication (MFA), having strong password policies, and good session management are essential. Also important is ensuring that password reset functionality is secure and that brute force attacks are blocked.

Software and Data Integrity Failures:

Software and data integrity failures happen when software updates, critical data, or critical processes are compromised. This is basically when a malicious piece of code is painted as authentic and secure.

Example: https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know (Hackers targeted a third-party app which was trusted by the SolarWinds Orion platform)

Prevention: Can be hard, but there are ways. Having rigorous verification processes for software updates and data, along with using digital signatures to ensure the integrity of the software and data are important. The use of hashing for data integrity checks is also important. Security audits through the supply chain of the code.

Security logging and monitoring failures:

Security logging and monitoring failures happen when there are insufficient logs to see what's going on in the system and to notice breaches. This makes it hard to detect and respond to security events.

Example: https://www.syteca.com/en/blog/real-life-examples-insider-threat-caused-breaches

Prevention: Implementing comprehensive security logging and monitoring is key, along with analysis of these logs, so that the security team can notice anomalies. Security event management systems and analysis tools are very helpful with this work.

Server-Side Request Forgery:

SSRF is when an application can be tricked into making requests to resources that it should not be able to access.

Example: https://blog.appsecco.com/an-ssrf-privileged-aws-keys-and-the-capital-one-breach-4c3c2cded3af (Gained access to a credential which had more priviledges than needed and was able to do malicious stuff in other AWS areas)

Prevention: Input validation for URLs and restricting requests to only approved domains. Separating the server from the network it needs to protect, also helps to avoid this vulnerability.