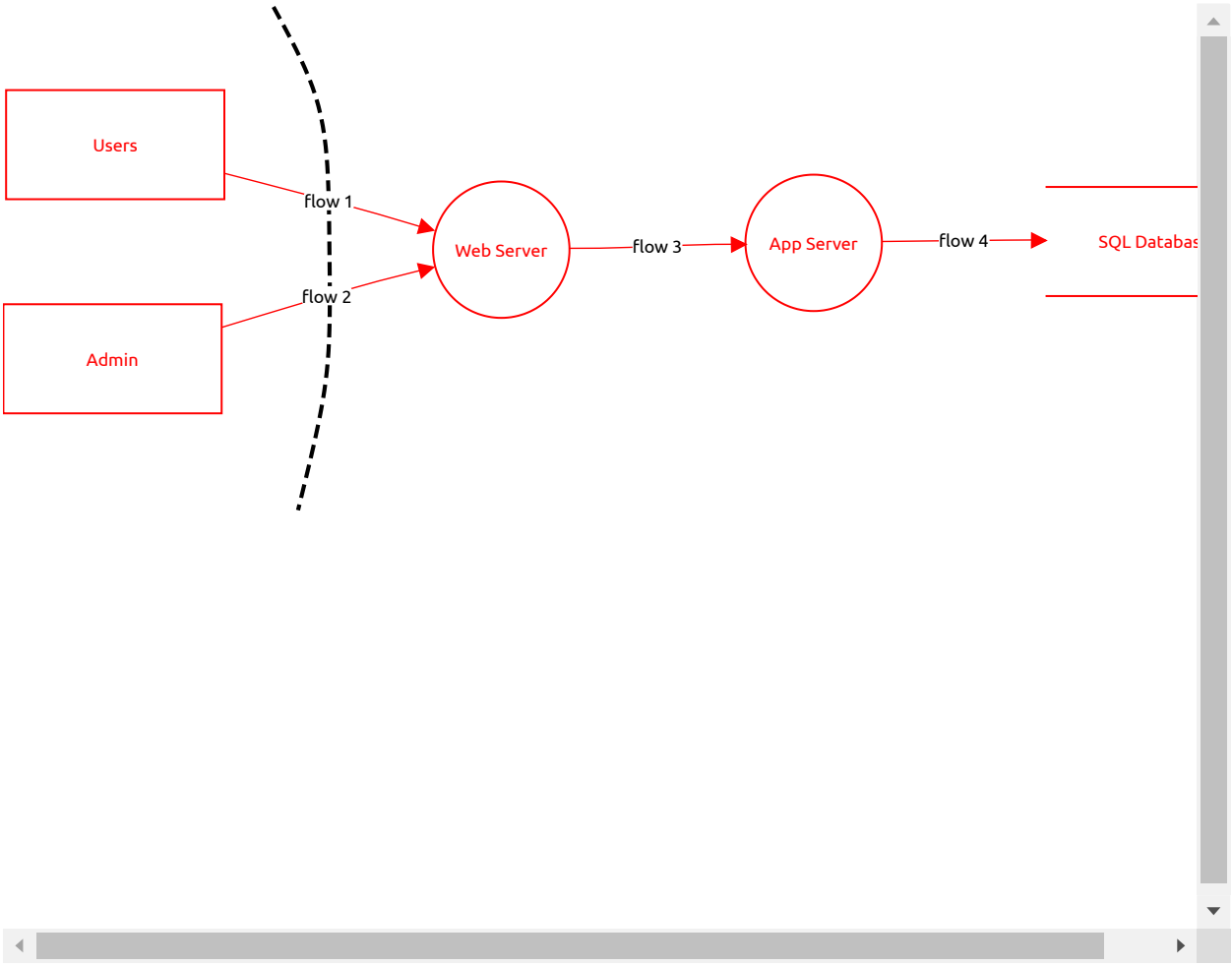
Threat model report for Web App

Owner:
Sayed Aslam
Reviewer:
Sam
Contributors:

High level system description

Website for Juice sell

WebApp


Users
Admin
flow 1
flow 2
Web Server
flow 3
App Server
flow 4
SQL Databas

Users (External Actor)

Description:

Generic spoofing threat

Spoofing, Open, High Priority

Description:
Description:
In this scenario, attackers exploit the lack of encryption in HTTP traffic to impersonate legitimate users. Without secure channels, user credentials or session tokens can be intercepted and reused by an adversary to gain unauthorized access or execute malicious actions.
Attack Vectors:
Man-in-the-Middle (MITM): Attackers intercept HTTP traffic to capture login credentials or session cookies.
Session Hijacking: Without encryption, session tokens are vulnerable to capture and reuse, allowing impersonation of a valid user.
Credential Replay: Stolen credentials can be replayed to the web server to simulate legitimate user actions.
Risks & Impacts:
Unauthorized Access: Attackers gain access to sensitive user accounts and may perform actions on behalf of the user.
Data Leakage: Exposure of sensitive personal data, leading to privacy breaches.
Loss of Trust: Users and stakeholders may lose confidence in the application's security, potentially affecting brand reputation.

Mitigation:
Mitigations:
Enforce HTTPS: Transition all user communications from HTTP to HTTPS to ensure data encryption in transit.
Secure Session Management: Utilize secure, HttpOnly cookies and short-lived tokens to reduce the risk of session hijacking.
User Education: Train users to recognize phishing attempts and secure their devices to avoid credential theft.

Generic repudiation threat

Repudiation, Open, High Priority

Description:
Description:
Repudiation threats occur when user actions can later be denied due to insufficient or insecure logging, especially over unencrypted HTTP channels. This makes it difficult to attribute actions, complicating incident investigation and accountability.
Attack Vectors:
Inadequate Logging: Lack of detailed and tamper-evident logs allows users or attackers to deny having performed certain actions.
Interception of Unsecured Traffic: Without encryption, log data or session details

can be altered or removed by an attacker, undermining evidence.
Forged Records: Attackers may inject false log entries to obfuscate their activities or create ambiguity about user actions.
Risks & Impacts:
Loss of Accountability: Inability to reliably trace user actions hinders forensic investigations and dispute resolution.
Compliance Failures: Regulatory requirements often mandate robust audit trails; failure to provide these can lead to legal or compliance issues.
Undetected Insider Threats: Malicious insiders could perform unauthorized actions and later repudiate them, complicating remediation efforts.

Mitigation:
Mitigations:
Implement Secure Logging: Use tamper-evident logging mechanisms with digital signatures and secure timestamps to ensure the integrity of audit trails.
Enforce HTTPS: Encrypt all data transmissions, including log data, to prevent interception or manipulation.
Regular Log Audits: Periodically review logs for inconsistencies or anomalies that might indicate tampering or unauthorized activity.
Re-Authentication for Critical Actions: Require additional verification (e.g., multi-factor authentication) before performing high-risk operations to strengthen accountability.

Admin (External Actor)

Description:

Generic spoofing threat

Spoofing, Open, High Priority

Description:
Description:
This threat occurs when an attacker impersonates an administrator by exploiting the insecure nature of HTTP. Without encryption, admin session data, credentials, or tokens can be intercepted and replayed to gain unauthorized admin privileges.
Attack Vectors:
Man-in-the-Middle (MITM): Interception of unencrypted HTTP traffic to capture session cookies or credentials.
Session Hijacking: Stealing active session tokens due to lack of secure transmission.
Credential Replay: Reusing intercepted admin credentials to authenticate as an admin.
Risks & Impacts:
Unauthorized Access: Attackers gain admin-level access, enabling them to alter configurations, view sensitive data, or control critical functions.
System Compromise: The integrity and availability of the web server may be fully undermined.
Reputational and Financial Damage: Compromised admin accounts can lead to regulatory breaches and loss of stakeholder trust.

Mitigation:
Mitigations:
Enforce HTTPS: Migrate all admin communications to TLS-encrypted channels to prevent eavesdropping.
Secure Session Management: Use secure cookies (with HttpOnly and Secure flags) and implement short-lived tokens.
Multi-Factor Authentication (MFA): Require an additional authentication factor for admin access.
Intrusion Detection: Monitor for unusual login patterns or session anomalies that may indicate spoofing attempts.

Generic repudiation threat

Repudiation, Open, High Priority

Description:
Description:
Repudiation threats arise when admin actions can be denied due to weak or absent auditing mechanisms. Using HTTP without encryption exposes both the communication and the logging process to manipulation, making it difficult to attribute actions accurately.
Attack Vectors:
Weak or Tampered Logging: Without tamper-proof logs, an admin or attacker can

deny having executed certain actions.
Interception of Unsecured Traffic: HTTP’s lack of encryption allows attackers to modify or forge log entries, obscuring evidence of admin actions.
Lack of Non-Repudiation Controls: Insufficient measures to sign or verify critical admin transactions, leading to denial of responsibility.
Risks & Impacts:
Loss of Accountability: Inability to trace and verify admin actions complicates forensic analysis and incident response.
Legal/Compliance Issues: Failure to produce reliable audit trails may result in non-compliance with security standards or legal requirements.
Insider Threat: Malicious insiders might perform critical operations and later repudiate them, hindering remediation efforts.

Mitigation:
Mitigations:
Secure Communications (HTTPS): Encrypt admin interactions to protect the integrity of transmitted data and logs.
Tamper-Evident Logging: Implement centralized logging with secure timestamps and digital signatures to ensure logs cannot be altered without detection.
Periodic Log Audits: Regularly review audit trails to identify anomalies or evidence of tampering.
Re-Authentication for Sensitive Actions: Require administrators to verify their identity (e.g., through MFA) before performing high-risk operations.

App Server (Process)

Description:

Generic spoofing threat

Spoofing, Open, High Priority

Description:
Threat Description:
Spoofing attacks occur when an attacker impersonates a legitimate entity to gain unauthorized access or manipulate data. This could involve IP spoofing, email spoofing, ARP spoofing, DNS spoofing, or user identity spoofing. The attacker exploits weaknesses in authentication mechanisms to deceive systems or users, potentially leading to unauthorized access, data theft, or session hijacking.
Attack Vectors:
Forged credentials or digital certificates
DNS spoofing, ARP poisoning, or IP address spoofing
Man-in-the-middle techniques that impersonate the legitimate server
Risks & Impacts:
Attackers may impersonate the App Server, leading to unauthorized access
Interception and manipulation of sensitive data
Loss of user trust and potential financial or reputational damage

Mitigation:
Mitigations:
Implement strong mutual authentication (e.g., using mutual TLS and certificate pinning)
Enforce secure DNS practices and ARP monitoring
Regularly update and validate digital certificates and credentials

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Exploiting vulnerabilities to modify configuration files or code
Intercepting and altering data in transit (e.g., via injection attacks)
Unauthorized changes to critical application parameters
Risks & Impacts:
Compromised data integrity, leading to corrupted or maliciously altered

transactions
Service disruptions if configuration settings are modified
Increased risk of further vulnerabilities being introduced

Mitigation:
Mitigations:
Use cryptographic hashes or digital signatures to verify the integrity of files and data
Implement secure channels (e.g., TLS) for all communications
Employ strict configuration management and regular security audits

Generic repudiation threat

Repudiation, Open, High Priority

Description:
Threat Description:
A repudiation attack occurs when an entity (user or system) denies performing an action within an application, making it difficult to track activities or hold users accountable. This happens when proper logging, authentication, and non-repudiation mechanisms are not in place. Attackers can erase, modify, or manipulate logs to hide malicious activities or falsely claim they didn't perform an action.
Attack Vectors:
Lack of comprehensive audit logging, allowing attackers to deny actions
Exploiting weak or missing authentication records to obfuscate malicious activities
Risks & Impacts:
Inability to trace and attribute actions in the event of a breach
Challenges during forensic investigations and potential legal or compliance issues
Erosion of accountability within the system

Mitigation:
Mitigations:
Implement robust, tamper-proof logging mechanisms (with digital signing and time stamps)
Ensure audit trails are stored securely and are accessible for incident analysis
Regularly review logs for anomalies and enforce non-repudiation policies

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:
Threat Description:
An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.
Attack Vectors:

Exploiting misconfigured endpoints or verbose error messages that leak internal details
Unencrypted transmissions or poorly secured storage exposing sensitive data
Overly permissive access controls on APIs or log files
Risks & Impacts:
Exposure of sensitive user data, internal configurations, or proprietary information
Increased vulnerability to targeted attacks or social engineering
Regulatory penalties, reputational damage, and potential financial loss

Mitigation:
Mitigations:
Enforce strict access control and authentication on all data endpoints
Encrypt sensitive data both in transit and at rest
Sanitize error messages to prevent leakage of internal system details
Regularly review and harden data permissions and API security

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
Flooding the App Server with a high volume of requests (network-level or application-level attacks)
Exploiting resource-intensive operations or known vulnerabilities to exhaust system resources
Distributed DoS (DDoS) attacks using botnets
Risks & Impacts:
Service unavailability for legitimate users, leading to business disruption
Increased operational costs and potential financial loss
Damage to the organization’s reputation and loss of customer confidence

Mitigation:
Mitigations:
Implement rate limiting and request throttling to manage traffic surges
Use load balancers and auto-scaling infrastructure to distribute and handle peak loads
Deploy network-level defenses (e.g., firewalls, DDoS protection services) and monitor for abnormal traffic patterns

Generic elevation threat

Elevation of privilege, Open, High Priority

Description:
Threat Description:
Elevation of Privilege (EoP) occurs when an attacker gains unauthorized access or increases their level of access within a system by exploiting vulnerabilities, misconfigurations, or weak security controls. This can lead to unauthorized data access, system control, or privilege abuse, compromising the security and integrity of the application.
Attack Vectors:
Exploiting vulnerabilities (e.g., injection flaws, misconfigured permissions) to bypass access controls
Taking advantage of insecure defaults or unpatched software components
Using crafted requests to trigger logic errors that grant unauthorized access levels
Risks & Impacts:
Attackers gaining administrative or higher-level access, leading to full system compromise
Unauthorized execution of sensitive operations or data manipulation
Severe operational, financial, and reputational impacts due to widespread control of the server

Mitigation:
Mitigations:
Enforce the principle of least privilege, ensuring users and processes have only the access they need
Regularly update and patch software to remediate known vulnerabilities
Implement rigorous access control checks and multi-factor authentication
Conduct regular security testing and code reviews to identify and remediate potential elevation paths

Web Server (Process)

Description:

Generic spoofing threat

Spoofing, Open, High Priority

Description:
Threat Description: Spoofing attacks occur when an attacker impersonates a legitimate entity to gain unauthorized access or manipulate data. This could involve IP spoofing, email spoofing, ARP spoofing, DNS spoofing, or user identity spoofing. The attacker exploits weaknesses in authentication mechanisms to deceive systems or users, potentially leading to unauthorized access, data theft, or session hijacking.
Attack Vectors:
Forged SSL/TLS certificates or manipulated DNS records (e.g., via DNS cache poisoning)
IP address spoofing to mislead clients into connecting to a malicious server
Using compromised domain registrar settings to redirect traffic
Risks & Impacts:
Attackers may impersonate the legitimate web server, capturing sensitive user credentials or data
Clients may be deceived into divulging confidential information
Overall erosion of trust and potential for widespread phishing or man-in-the-middle attacks

Mitigation:
Mitigations:
Enforce robust authentication using mutual TLS and certificate pinning
Use DNSSEC to protect against DNS spoofing
Regularly validate and renew digital certificates through secure channels

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Exploiting vulnerabilities (e.g., injection flaws) to alter web server code or configurations
Intercepting and modifying data in transit if encryption isn't enforced
Unauthorized access to configuration files that govern the server's behavior
Risks & Impacts:
Altered configurations or code can lead to data corruption or injection of malicious

functionality
Unauthorized changes may open new vulnerabilities, further compromising system integrity
Potential for cascading failures affecting multiple services or applications

Mitigation:
Mitigations:
Implement integrity checks (e.g., cryptographic hashing or digital signatures) on code and configuration files
Secure all communication channels with up-to-date TLS encryption
Enforce strict file system permissions and configuration management controls

Generic repudiation threat

Repudiation, Open, High Priority

Description:
Threat Description:
A repudiation attack occurs when an entity (user or system) denies performing an action within an application, making it difficult to track activities or hold users accountable. This happens when proper logging, authentication, and non-repudiation mechanisms are not in place. Attackers can erase, modify, or manipulate logs to hide malicious activities or falsely claim they didn't perform an action.
Attack Vectors:
Lack of detailed audit logs that allow an attacker to hide their activities
Weak authentication and logging practices that make it possible to deny actions
Tampering with logs or using transient sessions without proper traceability
Risks & Impacts:
Inability to accurately trace malicious activity during an incident
Challenges in performing forensic analysis or establishing accountability
Legal and compliance issues due to insufficient non-repudiation measures

Mitigation:
Mitigations:
Deploy robust, tamper-evident logging mechanisms that record all critical actions
Use digital signatures and secure time stamps on log entries to ensure non-repudiation
Regularly audit log files for anomalies and enforce comprehensive log retention policies

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:
Threat Description:
An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential

information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.
Attack Vectors:
Misconfigured web server settings leading to open directory listings or exposed backups
Detailed error messages or stack traces that reveal internal system architecture
Insecure API endpoints or unencrypted data transmissions
Risks & Impacts:
Exposure of sensitive data, including user credentials, internal configurations, and proprietary information
Increased risk of targeted attacks by providing attackers with detailed system insights
Regulatory and compliance repercussions if sensitive information is leaked

Mitigation:
Mitigations:
Harden server configurations by disabling unnecessary directory listings and securing backup data
Sanitize error messages to prevent leakage of internal details
Encrypt sensitive data in transit and at rest; enforce strict access control on APIs

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
High-volume HTTP request floods (or distributed DoS via botnets) targeting server resources
Exploiting specific application endpoints that are resource-intensive
Using slowloris-type attacks that open numerous connections without completing them
Risks & Impacts:
Overwhelming the server’s resources, resulting in service unavailability for legitimate users
Financial losses due to downtime and degradation of service performance
Damage to brand reputation and customer trust

Mitigation:
Mitigations:
Implement rate limiting and request throttling to control traffic volumes
Use load balancers and auto-scaling architectures to distribute incoming traffic
Deploy a Web Application Firewall (WAF) and other DoS/DDoS mitigation solutions to detect and block malicious traffic

Generic elevation threat

Elevation of privilege, Open, High Priority

Description:

Threat Description:
Elevation of Privilege (EoP) occurs when an attacker gains unauthorized access or increases their level of access within a system by exploiting vulnerabilities, misconfigurations, or weak security controls. This can lead to unauthorized data access, system control, or privilege abuse, compromising the security and integrity of the application.
Attack Vectors:
Exploiting vulnerabilities in the web server software or third-party components to gain higher access
Bypassing access controls via injection or misconfiguration errors
Using crafted requests to trigger flaws that grant unauthorized administrative rights
Risks & Impacts:
Attackers may obtain administrative privileges, enabling full control over the web server
Unauthorized modifications to sensitive data and critical configurations
Severe compromise of system integrity, potentially affecting other connected systems

Mitigation:

Mitigations:
Enforce the principle of least privilege, ensuring minimal access rights for all processes and users
Regularly update and patch the server and all associated software to remediate known vulnerabilities
Implement rigorous access control and multi-factor authentication measures
Conduct periodic security testing and code reviews to identify and remediate potential escalation paths

SQL Database (Data Store)

Description:
SQL Database

Generic tampering threat
Tampering, Open, High Priority

Description:
Threat Description:
Tampering with an SQL database involves an attacker modifying, deleting, or corrupting data by exploiting weak security controls. This could be done through SQL injection, privilege abuse, or direct unauthorized access to the database. If successful, it can lead to data integrity issues, unauthorized modifications, or even complete database destruction.
Attack Vectors:
SQL Injection (SQLi): Exploiting unvalidated user inputs to modify database queries (e.g., DROP TABLE users;).
Privilege Misuse: Exploiting overly permissive user roles to execute unauthorized SQL commands.
Unsecured Database Configuration: Using default credentials or weak authentication to gain direct access.
API or Backend Data Manipulation: Sending malicious API requests that modify stored SQL data.
Database Log Tampering: Deleting or modifying audit logs to erase traces of malicious activities.
Malware or Insider Threats: Using malicious scripts or compromised accounts to modify database records.
Risks & Potential Impact:
Data Integrity Loss: Attackers modify critical records, affecting business operations.
Unauthorized Access & Control: Attackers alter permissions to escalate privileges or lock out legitimate users.
Data Deletion or Corruption: Permanent loss or damage to important information.
Compliance Violations: Violating data integrity laws (GDPR, HIPAA, PCI-DSS) due to unauthorized modifications.
Financial & Reputational Damage: Loss of customer trust due to tampered financial records, transactions, or logs.

Mitigation:
Mitigation Strategies:
Use Parameterized Queries & ORM: Prevent SQL Injection by using prepared statements instead of raw SQL queries.
Enforce Role-Based Access Control (RBAC): Restrict database privileges based on user roles (least privilege principle).
Enable Database Auditing & Logging: Monitor changes with tamper-proof audit logs (e.g., immutable logging).
Encrypt Sensitive Data: Protect stored information using AES encryption to prevent unauthorized modifications.
Apply Strong Authentication & MFA: Require multi-factor authentication for

database access.
Use Web Application Firewall (WAF): Detect and block SQLi attempts before they reach the database.
Regular Security Patching: Keep database software, frameworks, and dependencies updated to prevent known exploits.

Generic repudiation threat

Repudiation, Open, High Priority

Description:
Threat Description:
A repudiation attack on an SQL database occurs when a user (legitimate or malicious) denies performing an action, such as modifying, deleting, or inserting records. This happens when proper logging, authentication, and integrity mechanisms are missing, making it difficult to track and verify who made changes to the database. Attackers may erase logs, manipulate transactions, or exploit weak auditing controls to cover their tracks.
Attack Vectors:
Lack of Secure Logging: Attackers delete or modify audit logs to erase traces of malicious activity.
Insufficient Authentication: Weak identity verification allows unauthorized users to perform actions without accountability.
SQL Injection (SQLi): Attackers inject SQL queries to modify data and remove traces of changes.
Privilege Escalation & Misuse: Unauthorized users gain elevated access and perform actions under another user's identity.
Timestamp & Data Manipulation: Altering transaction timestamps, user records, or financial data to create false records.
Malicious API Requests: Sending forged API requests to modify database entries without proper verification.
Risks & Potential Impact:
Denial of Responsibility: Attackers or users can claim they never performed an action, making investigations difficult.
Data Integrity Issues: Unauthorized modifications lead to data inconsistencies and business logic failures.
Financial & Legal Consequences: Altered or deleted transactions may result in fraud, financial loss, or regulatory violations (e.g., GDPR, HIPAA, PCI-DSS).
Security Investigation Failure: If logs are missing or altered, security teams cannot trace unauthorized activities.
Loss of Trust & Compliance Violations: Without accountability, organizations fail audits and risk legal penalties.

Mitigation:
Mitigation Strategies:
Implement Tamper-Proof Logging: Use immutable logs (e.g., WORM storage, SIEM solutions) to prevent log modifications.
Use Cryptographic Signatures for Transactions: Digitally sign records to ensure data integrity and non-repudiation.
Enforce Strong Authentication & MFA: Require multi-factor authentication (MFA) for database access and administrative actions.

Role-Based Access Control (RBAC): Ensure least privilege access to prevent unauthorized changes.
Enable SQL Auditing & Alerts: Configure database audit trails to detect and notify security teams of suspicious modifications.
Prevent SQL Injection & API Exploits: Use prepared statements, input validation, and API authentication to prevent tampering.
Monitor & Review Database Logs: Implement automated monitoring (e.g., SIEM, database integrity scanners) to detect anomalies.

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:
Threat Description:
An information disclosure attack on an SQL database occurs when sensitive data is exposed to unauthorized users due to misconfigurations, weak access controls, or SQL injection vulnerabilities. Attackers can gain access to confidential data such as user credentials, financial records, personally identifiable information (PII), or system configurations by exploiting weaknesses in how the database stores and transmits data.
Attack Vectors:
SQL Injection (SQLi): Attackers manipulate queries to extract sensitive information (e.g., ' OR 1=1 -- to dump user credentials).
Misconfigured Database Permissions: Excessive privileges allow unauthorized users to view confidential tables.
Unencrypted Data Storage: Sensitive records (passwords, credit card details, etc.) are stored in plaintext instead of being encrypted.
Exposed Debugging & Error Messages: Detailed SQL errors reveal database structure, table names, and internal queries.
Backup File Exposure: Database backup files are left publicly accessible, allowing attackers to download and analyze them.
Leaked API Keys & Credentials: Hardcoded credentials in configuration files, scripts, or logs expose database access.
Side-Channel Attacks: Timing-based attacks allow attackers to infer if a record exists without direct access.
Risks & Potential Impact:
Data Breach: Unauthorized users access sensitive business and user data.
Identity Theft & Fraud: Attackers steal PII, login credentials, or payment details for malicious purposes.
Regulatory Non-Compliance: Violates GDPR, HIPAA, PCI-DSS due to improper data protection.
Financial & Reputational Damage: Loss of customer trust and legal penalties due to leaked confidential data.
Targeted Attacks & Exploitation: Exposed database structures make the system vulnerable to further attacks.

Mitigation:
Mitigation Strategies:
Use Strong Encryption: Encrypt sensitive data at rest (AES-256) and in transit (TLS/SSL) to prevent unauthorized access.

Implement Role-Based Access Control (RBAC): Restrict database access based on user roles and follow the principle of least privilege (PoLP).
Prevent SQL Injection: Use parameterized queries and input validation to prevent unauthorized data extraction.
Secure Database Configuration: Disable public access, restrict IP whitelisting, and enforce strong authentication.
Mask & Hash Sensitive Data: Use hashing (bcrypt, PBKDF2, Argon2) for passwords and data masking for PII fields.
Disable Detailed Error Messages: Display generic errors instead of exposing database details.
Regular Security Audits: Periodically scan for data leaks, misconfigurations, and exposed credentials.

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack on an SQL database occurs when an attacker disrupts database availability by overwhelming it with excessive queries, locking resources, or exploiting vulnerabilities. This prevents legitimate users from accessing the database, leading to service downtime, degraded performance, or even system crashes.
Attack Vectors:
SQL Query Flooding: Repeatedly sending expensive queries (e.g., SELECT * FROM large_table) to exhaust resources.
Slowloris-Type Query Attacks: Keeping multiple connections open for long periods to exhaust database connections.
Locking Database Transactions: Initiating transactions but never committing them, preventing other users from executing queries.
Abusing Full-Table Scans: Forcing inefficient queries that scan large datasets and slow down performance.
Triggering Recursive Queries: Exploiting stored procedures or recursion to cause infinite loops.
Exploiting Disk or Memory Exhaustion: Generating huge temporary tables, excessive logs, or large file uploads to consume system storage.
Exploiting Unpatched Vulnerabilities: Using buffer overflow or memory corruption exploits to crash the database server.
Risks & Potential Impact:
Service Downtime: Database crashes, preventing applications from functioning.
Slow Performance: Application latency increases, affecting business operations.
Resource Exhaustion: Excessive CPU, memory, or disk usage leads to a denial of service for legitimate users.
Financial & Reputational Damage: Downtime results in lost transactions, revenue, and customer trust.
Data Corruption or Loss: Uncontrolled resource exhaustion may result in data integrity issues.

Mitigation:

Mitigation Strategies:
Rate Limiting & Connection Throttling: Restrict the number of allowed queries per user/IP to prevent abuse.
Optimize Query Performance: Use indexing, caching, and query optimization to reduce the impact of heavy queries.
Enforce Timeout & Connection Limits: Set query execution time limits and max concurrent connections to avoid slow query attacks.
Implement Web Application Firewall (WAF): Use a WAF to block automated query flooding attempts.
Use Load Balancing & Replication: Distribute database load across multiple replicated instances to mitigate DoS effects.
Monitor & Alert on Anomalous Activity: Set up Intrusion Detection Systems (IDS) or database monitoring tools to detect unusual spikes in resource usage.
Regularly Patch & Update Database Software: Prevent exploitation of known vulnerabilities that can cause crashes.

flow 1 (Data Flow)

Description:

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Client-Side Tampering: Modifying cookies, local storage, or browser scripts.
API Tampering: Altering API requests, parameters, or responses.
Database Tampering: Manipulating stored data via SQL injection or unauthorized access.
Software Tampering: Reverse-engineering or injecting malicious code into the application.
Network Tampering: Intercepting and modifying data in transit (e.g., MITM attacks).
File Manipulation: Altering configuration files, logs, or executables.
Risks & Potential Impact:
Unauthorized privilege escalation or access to restricted functions.
Financial fraud (e.g., modifying transaction values in an e-commerce app).
Data integrity loss (e.g., altering database records, logs, or stored credentials).
Malware injection leading to remote code execution (RCE).
Bypassing authentication by tampering with session tokens or authorization parameters.

Mitigation:
Mitigation Strategies:
Implement Data Integrity Controls: Use cryptographic hashing (SHA-256, HMAC) to detect unauthorized changes.
Secure APIs & Inputs: Enforce input validation and parameterized queries to prevent API tampering.
Use Digital Signatures: Ensure software authenticity using code-signing techniques.
Enforce HTTPS/TLS Encryption: Prevent MITM attacks and data tampering in transit.
Apply Secure Logging Mechanisms: Implement tamper-proof logs using WORM (Write Once Read Many) storage.
Implement File Integrity Monitoring (FIM): Detect unauthorized modifications to configuration files, scripts, or binaries.
Restrict Client-Side Manipulation: Use HTTP-only, Secure, and SameSite cookies to prevent tampering.

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:

Threat Description:

An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.

Attack Vectors:

Insufficient Encryption: Data transmitted over HTTP instead of HTTPS, allowing attackers to intercept traffic via Man-in-the-Middle (MitM) attacks.

Misconfigured Access Controls: Unauthorized users can access sensitive files, logs, or APIs.

Error Messages & Stack Traces: Exposing detailed error messages that reveal system details, database structures, or API keys.

Directory Listing Enabled: Web servers reveal sensitive directories or files, allowing attackers to explore system structures.

Insecure Data Storage: Sensitive data stored unencrypted in databases, logs, or configuration files.

Hardcoded Credentials & API Keys: Source code or configuration files contain plaintext passwords or API keys.

Side-Channel Attacks: Attackers gather unintended data leaks through system behaviors, such as response times or error messages.

Risks & Potential Impact:

Data Breach: Unauthorized users access confidential business or user data.

Identity Theft & Fraud: Exposed PII (e.g., usernames, emails, payment details) can be exploited for fraud.

Credential Leaks: Attackers obtain plaintext passwords or session tokens, leading to account takeovers.

Regulatory Non-Compliance: Violating data protection laws (e.g., GDPR, HIPAA, PCI-DSS) due to improper data handling.

Reputational Damage: Exposed internal data (e.g., source code, business strategies) harms the company’s reputation.

Mitigation:

Mitigation Strategies:

Enforce Strong Encryption: Use TLS (HTTPS), AES-256 encryption for data storage, and hash passwords using bcrypt, PBKDF2, or Argon2.

Apply Strict Access Controls: Implement role-based access control (RBAC) and principle of least privilege (PoLP).

Mask Sensitive Data in Logs & Responses: Avoid exposing confidential information in error messages, logs, or API responses.

Disable Directory Listing & Restrict File Access: Ensure web servers do not reveal directory structures or unnecessary files.

Secure Error Handling: Configure the system to return generic error messages instead of detailed stack traces.

Conduct Regular Security Audits: Scan code repositories for hardcoded credentials, API keys, and sensitive information.

Implement Content Security Policy (CSP): Prevent data leaks via malicious scripts

(XSS attacks).
Use Web Application Firewalls (WAF): Protect against automated attacks and data harvesting attempts.

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
Volumetric Attacks: Flooding the target with massive amounts of network traffic (SYN Flood, UDP Flood, ICMP Flood, HTTP Flood).
Resource Exhaustion Attacks: Consuming CPU, memory, or disk space by sending malformed requests, large payloads, or excessive database queries.
Application Layer DoS: Overloading web servers or APIs with slow requests, frequent login attempts, or malicious bot traffic.
Exploiting Vulnerabilities: Triggering infinite loops, memory leaks, or unhandled exceptions to crash the system.
Distributed DoS (DDoS): Using a botnet (compromised devices) to launch a coordinated attack, making it harder to mitigate.
Risks & Potential Impact:
Service Downtime: Disrupting operations, leading to loss of availability.
Financial Loss: Downtime results in lost revenue and productivity.
Reputational Damage: Customers lose trust due to frequent service outages.
Legal & Compliance Issues: Failure to provide reliable services may violate SLA (Service Level Agreements).
System Instability: Unhandled DoS attempts may cause server crashes, database lockups, or resource exhaustion.

Mitigation:
Mitigation Strategies:
Rate Limiting & Traffic Filtering: Implement rate limits on API requests and use firewalls to filter malicious traffic.
Web Application Firewall (WAF): Deploy a WAF to block automated and volumetric attacks at the application level.
DDoS Protection Services: Use cloud-based DDoS mitigation services like Cloudflare, AWS Shield, or Akamai.
Load Balancing & Redundancy: Distribute traffic across multiple servers and implement auto-scaling to absorb attacks.
Application Hardening: Optimize code efficiency, database queries, and exception handling to prevent resource exhaustion.
Anomaly Detection & Monitoring: Use Intrusion Detection Systems (IDS) to detect unusual traffic patterns and respond proactively.
Timeout & Connection Limits: Set timeouts for incoming requests and limit the number of open connections per user.

flow 2 (Data Flow)

Description:

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Client-Side Tampering: Modifying cookies, local storage, or browser scripts.
API Tampering: Altering API requests, parameters, or responses.
Database Tampering: Manipulating stored data via SQL injection or unauthorized access.
Software Tampering: Reverse-engineering or injecting malicious code into the application.
Network Tampering: Intercepting and modifying data in transit (e.g., MITM attacks).
File Manipulation: Altering configuration files, logs, or executables.
Risks & Potential Impact:
Unauthorized privilege escalation or access to restricted functions.
Financial fraud (e.g., modifying transaction values in an e-commerce app).
Data integrity loss (e.g., altering database records, logs, or stored credentials).
Malware injection leading to remote code execution (RCE).
Bypassing authentication by tampering with session tokens or authorization parameters.

Mitigation:
Mitigation Strategies:
Implement Data Integrity Controls: Use cryptographic hashing (SHA-256, HMAC) to detect unauthorized changes.
Secure APIs & Inputs: Enforce input validation and parameterized queries to prevent API tampering.
Use Digital Signatures: Ensure software authenticity using code-signing techniques.
Enforce HTTPS/TLS Encryption: Prevent MITM attacks and data tampering in transit.
Apply Secure Logging Mechanisms: Implement tamper-proof logs using WORM (Write Once Read Many) storage.
Implement File Integrity Monitoring (FIM): Detect unauthorized modifications to configuration files, scripts, or binaries.
Restrict Client-Side Manipulation: Use HTTP-only, Secure, and SameSite cookies to prevent tampering.

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:

Threat Description:

An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.

Attack Vectors:

Insufficient Encryption: Data transmitted over HTTP instead of HTTPS, allowing attackers to intercept traffic via Man-in-the-Middle (MitM) attacks.

Misconfigured Access Controls: Unauthorized users can access sensitive files, logs, or APIs.

Error Messages & Stack Traces: Exposing detailed error messages that reveal system details, database structures, or API keys.

Directory Listing Enabled: Web servers reveal sensitive directories or files, allowing attackers to explore system structures.

Insecure Data Storage: Sensitive data stored unencrypted in databases, logs, or configuration files.

Hardcoded Credentials & API Keys: Source code or configuration files contain plaintext passwords or API keys.

Side-Channel Attacks: Attackers gather unintended data leaks through system behaviors, such as response times or error messages.

Risks & Potential Impact:

Data Breach: Unauthorized users access confidential business or user data.

Identity Theft & Fraud: Exposed PII (e.g., usernames, emails, payment details) can be exploited for fraud.

Credential Leaks: Attackers obtain plaintext passwords or session tokens, leading to account takeovers.

Regulatory Non-Compliance: Violating data protection laws (e.g., GDPR, HIPAA, PCI-DSS) due to improper data handling.

Reputational Damage: Exposed internal data (e.g., source code, business strategies) harms the company’s reputation.

Mitigation:

Mitigation Strategies:

Enforce Strong Encryption: Use TLS (HTTPS), AES-256 encryption for data storage, and hash passwords using bcrypt, PBKDF2, or Argon2.

Apply Strict Access Controls: Implement role-based access control (RBAC) and principle of least privilege (PoLP).

Mask Sensitive Data in Logs & Responses: Avoid exposing confidential information in error messages, logs, or API responses.

Disable Directory Listing & Restrict File Access: Ensure web servers do not reveal directory structures or unnecessary files.

Secure Error Handling: Configure the system to return generic error messages instead of detailed stack traces.

Conduct Regular Security Audits: Scan code repositories for hardcoded credentials, API keys, and sensitive information.

Implement Content Security Policy (CSP): Prevent data leaks via malicious scripts

(XSS attacks).
Use Web Application Firewalls (WAF): Protect against automated attacks and data harvesting attempts.

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
Volumetric Attacks: Flooding the target with massive amounts of network traffic (SYN Flood, UDP Flood, ICMP Flood, HTTP Flood).
Resource Exhaustion Attacks: Consuming CPU, memory, or disk space by sending malformed requests, large payloads, or excessive database queries.
Application Layer DoS: Overloading web servers or APIs with slow requests, frequent login attempts, or malicious bot traffic.
Exploiting Vulnerabilities: Triggering infinite loops, memory leaks, or unhandled exceptions to crash the system.
Distributed DoS (DDoS): Using a botnet (compromised devices) to launch a coordinated attack, making it harder to mitigate.
Risks & Potential Impact:
Service Downtime: Disrupting operations, leading to loss of availability.
Financial Loss: Downtime results in lost revenue and productivity.
Reputational Damage: Customers lose trust due to frequent service outages.
Legal & Compliance Issues: Failure to provide reliable services may violate SLA (Service Level Agreements).
System Instability: Unhandled DoS attempts may cause server crashes, database lockups, or resource exhaustion.

Mitigation:
Mitigation Strategies:
Rate Limiting & Traffic Filtering: Implement rate limits on API requests and use firewalls to filter malicious traffic.
Web Application Firewall (WAF): Deploy a WAF to block automated and volumetric attacks at the application level.
DDoS Protection Services: Use cloud-based DDoS mitigation services like Cloudflare, AWS Shield, or Akamai.
Load Balancing & Redundancy: Distribute traffic across multiple servers and implement auto-scaling to absorb attacks.
Application Hardening: Optimize code efficiency, database queries, and exception handling to prevent resource exhaustion.
Anomaly Detection & Monitoring: Use Intrusion Detection Systems (IDS) to detect unusual traffic patterns and respond proactively.
Timeout & Connection Limits: Set timeouts for incoming requests and limit the number of open connections per user.

flow 3 (Data Flow)

Description:

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Client-Side Tampering: Modifying cookies, local storage, or browser scripts.
API Tampering: Altering API requests, parameters, or responses.
Database Tampering: Manipulating stored data via SQL injection or unauthorized access.
Software Tampering: Reverse-engineering or injecting malicious code into the application.
Network Tampering: Intercepting and modifying data in transit (e.g., MITM attacks).
File Manipulation: Altering configuration files, logs, or executables.
Risks & Potential Impact:
Unauthorized privilege escalation or access to restricted functions.
Financial fraud (e.g., modifying transaction values in an e-commerce app).
Data integrity loss (e.g., altering database records, logs, or stored credentials).
Malware injection leading to remote code execution (RCE).
Bypassing authentication by tampering with session tokens or authorization parameters.

Mitigation:
Mitigation Strategies:
Implement Data Integrity Controls: Use cryptographic hashing (SHA-256, HMAC) to detect unauthorized changes.
Secure APIs & Inputs: Enforce input validation and parameterized queries to prevent API tampering.
Use Digital Signatures: Ensure software authenticity using code-signing techniques.
Enforce HTTPS/TLS Encryption: Prevent MITM attacks and data tampering in transit.
Apply Secure Logging Mechanisms: Implement tamper-proof logs using WORM (Write Once Read Many) storage.
Implement File Integrity Monitoring (FIM): Detect unauthorized modifications to configuration files, scripts, or binaries.
Restrict Client-Side Manipulation: Use HTTP-only, Secure, and SameSite cookies to prevent tampering.

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:

Threat Description:

An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.

Attack Vectors:

Insufficient Encryption: Data transmitted over HTTP instead of HTTPS, allowing attackers to intercept traffic via Man-in-the-Middle (MitM) attacks.

Misconfigured Access Controls: Unauthorized users can access sensitive files, logs, or APIs.

Error Messages & Stack Traces: Exposing detailed error messages that reveal system details, database structures, or API keys.

Directory Listing Enabled: Web servers reveal sensitive directories or files, allowing attackers to explore system structures.

Insecure Data Storage: Sensitive data stored unencrypted in databases, logs, or configuration files.

Hardcoded Credentials & API Keys: Source code or configuration files contain plaintext passwords or API keys.

Side-Channel Attacks: Attackers gather unintended data leaks through system behaviors, such as response times or error messages.

Risks & Potential Impact:

Data Breach: Unauthorized users access confidential business or user data.

Identity Theft & Fraud: Exposed PII (e.g., usernames, emails, payment details) can be exploited for fraud.

Credential Leaks: Attackers obtain plaintext passwords or session tokens, leading to account takeovers.

Regulatory Non-Compliance: Violating data protection laws (e.g., GDPR, HIPAA, PCI-DSS) due to improper data handling.

Reputational Damage: Exposed internal data (e.g., source code, business strategies) harms the company’s reputation.

Mitigation:

Mitigation Strategies:

Enforce Strong Encryption: Use TLS (HTTPS), AES-256 encryption for data storage, and hash passwords using bcrypt, PBKDF2, or Argon2.

Apply Strict Access Controls: Implement role-based access control (RBAC) and principle of least privilege (PoLP).

Mask Sensitive Data in Logs & Responses: Avoid exposing confidential information in error messages, logs, or API responses.

Disable Directory Listing & Restrict File Access: Ensure web servers do not reveal directory structures or unnecessary files.

Secure Error Handling: Configure the system to return generic error messages instead of detailed stack traces.

Conduct Regular Security Audits: Scan code repositories for hardcoded credentials, API keys, and sensitive information.

Implement Content Security Policy (CSP): Prevent data leaks via malicious scripts

(XSS attacks).
Use Web Application Firewalls (WAF): Protect against automated attacks and data harvesting attempts.

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
Volumetric Attacks: Flooding the target with massive amounts of network traffic (SYN Flood, UDP Flood, ICMP Flood, HTTP Flood).
Resource Exhaustion Attacks: Consuming CPU, memory, or disk space by sending malformed requests, large payloads, or excessive database queries.
Application Layer DoS: Overloading web servers or APIs with slow requests, frequent login attempts, or malicious bot traffic.
Exploiting Vulnerabilities: Triggering infinite loops, memory leaks, or unhandled exceptions to crash the system.
Distributed DoS (DDoS): Using a botnet (compromised devices) to launch a coordinated attack, making it harder to mitigate.
Risks & Potential Impact:
Service Downtime: Disrupting operations, leading to loss of availability.
Financial Loss: Downtime results in lost revenue and productivity.
Reputational Damage: Customers lose trust due to frequent service outages.
Legal & Compliance Issues: Failure to provide reliable services may violate SLA (Service Level Agreements).
System Instability: Unhandled DoS attempts may cause server crashes, database lockups, or resource exhaustion.

Mitigation:
Mitigation Strategies:
Rate Limiting & Traffic Filtering: Implement rate limits on API requests and use firewalls to filter malicious traffic.
Web Application Firewall (WAF): Deploy a WAF to block automated and volumetric attacks at the application level.
DDoS Protection Services: Use cloud-based DDoS mitigation services like Cloudflare, AWS Shield, or Akamai.
Load Balancing & Redundancy: Distribute traffic across multiple servers and implement auto-scaling to absorb attacks.
Application Hardening: Optimize code efficiency, database queries, and exception handling to prevent resource exhaustion.
Anomaly Detection & Monitoring: Use Intrusion Detection Systems (IDS) to detect unusual traffic patterns and respond proactively.
Timeout & Connection Limits: Set timeouts for incoming requests and limit the number of open connections per user.

flow 4 (Data Flow)

Description:

Generic tampering threat

Tampering, Open, High Priority

Description:
Threat Description:
Tampering attacks involve unauthorized modification of data, files, or application code to manipulate system behavior, bypass security controls, or cause system failures. Attackers exploit weak data integrity protections, inadequate access controls, or insecure software configurations to alter stored data, transmitted data, or application logic.
Attack Vectors:
Client-Side Tampering: Modifying cookies, local storage, or browser scripts.
API Tampering: Altering API requests, parameters, or responses.
Database Tampering: Manipulating stored data via SQL injection or unauthorized access.
Software Tampering: Reverse-engineering or injecting malicious code into the application.
Network Tampering: Intercepting and modifying data in transit (e.g., MITM attacks).
File Manipulation: Altering configuration files, logs, or executables.
Risks & Potential Impact:
Unauthorized privilege escalation or access to restricted functions.
Financial fraud (e.g., modifying transaction values in an e-commerce app).
Data integrity loss (e.g., altering database records, logs, or stored credentials).
Malware injection leading to remote code execution (RCE).
Bypassing authentication by tampering with session tokens or authorization parameters.

Mitigation:
Mitigation Strategies:
Implement Data Integrity Controls: Use cryptographic hashing (SHA-256, HMAC) to detect unauthorized changes.
Secure APIs & Inputs: Enforce input validation and parameterized queries to prevent API tampering.
Use Digital Signatures: Ensure software authenticity using code-signing techniques.
Enforce HTTPS/TLS Encryption: Prevent MITM attacks and data tampering in transit.
Apply Secure Logging Mechanisms: Implement tamper-proof logs using WORM (Write Once Read Many) storage.
Implement File Integrity Monitoring (FIM): Detect unauthorized modifications to configuration files, scripts, or binaries.
Restrict Client-Side Manipulation: Use HTTP-only, Secure, and SameSite cookies to prevent tampering

Generic information disclosure threat

Information disclosure, Open, High Priority

Description:

Threat Description:

An information disclosure attack occurs when sensitive data is exposed to unauthorized users due to insecure storage, transmission, or improper access controls. Attackers exploit vulnerabilities to gain access to confidential information, such as user credentials, personally identifiable information (PII), financial data, or system configuration details.

Attack Vectors:

Insufficient Encryption: Data transmitted over HTTP instead of HTTPS, allowing attackers to intercept traffic via Man-in-the-Middle (MitM) attacks.

Misconfigured Access Controls: Unauthorized users can access sensitive files, logs, or APIs.

Error Messages & Stack Traces: Exposing detailed error messages that reveal system details, database structures, or API keys.

Directory Listing Enabled: Web servers reveal sensitive directories or files, allowing attackers to explore system structures.

Insecure Data Storage: Sensitive data stored unencrypted in databases, logs, or configuration files.

Hardcoded Credentials & API Keys: Source code or configuration files contain plaintext passwords or API keys.

Side-Channel Attacks: Attackers gather unintended data leaks through system behaviors, such as response times or error messages.

Risks & Potential Impact:

Data Breach: Unauthorized users access confidential business or user data.

Identity Theft & Fraud: Exposed PII (e.g., usernames, emails, payment details) can be exploited for fraud.

Credential Leaks: Attackers obtain plaintext passwords or session tokens, leading to account takeovers.

Regulatory Non-Compliance: Violating data protection laws (e.g., GDPR, HIPAA, PCI-DSS) due to improper data handling.

Reputational Damage: Exposed internal data (e.g., source code, business strategies) harms the company’s reputation.

Mitigation:

Mitigation Strategies:

Enforce Strong Encryption: Use TLS (HTTPS), AES-256 encryption for data storage, and hash passwords using bcrypt, PBKDF2, or Argon2.

Apply Strict Access Controls: Implement role-based access control (RBAC) and principle of least privilege (PoLP).

Mask Sensitive Data in Logs & Responses: Avoid exposing confidential information in error messages, logs, or API responses.

Disable Directory Listing & Restrict File Access: Ensure web servers do not reveal directory structures or unnecessary files.

Secure Error Handling: Configure the system to return generic error messages instead of detailed stack traces.

Conduct Regular Security Audits: Scan code repositories for hardcoded credentials, API keys, and sensitive information.

Implement Content Security Policy (CSP): Prevent data leaks via malicious scripts

(XSS attacks).
Use Web Application Firewalls (WAF): Protect against automated attacks and data harvesting attempts.

Generic DoS threat

Denial of service, Open, High Priority

Description:
Threat Description:
A Denial of Service (DoS) attack is an attempt to disrupt the normal functionality of an application, system, or network by overwhelming it with excessive requests, consuming resources, or exploiting vulnerabilities. This attack prevents legitimate users from accessing services, leading to downtime, degraded performance, or even complete system failure.
Attack Vectors:
Volumetric Attacks: Flooding the target with massive amounts of network traffic (SYN Flood, UDP Flood, ICMP Flood, HTTP Flood).
Resource Exhaustion Attacks: Consuming CPU, memory, or disk space by sending malformed requests, large payloads, or excessive database queries.
Application Layer DoS: Overloading web servers or APIs with slow requests, frequent login attempts, or malicious bot traffic.
Exploiting Vulnerabilities: Triggering infinite loops, memory leaks, or unhandled exceptions to crash the system.
Distributed DoS (DDoS): Using a botnet (compromised devices) to launch a coordinated attack, making it harder to mitigate.
Risks & Potential Impact:
Service Downtime: Disrupting operations, leading to loss of availability.
Financial Loss: Downtime results in lost revenue and productivity.
Reputational Damage: Customers lose trust due to frequent service outages.
Legal & Compliance Issues: Failure to provide reliable services may violate SLA (Service Level Agreements).
System Instability: Unhandled DoS attempts may cause server crashes, database lockups, or resource exhaustion.

Mitigation:
Mitigation Strategies:
Rate Limiting & Traffic Filtering: Implement rate limits on API requests and use firewalls to filter malicious traffic.
Web Application Firewall (WAF): Deploy a WAF to block automated and volumetric attacks at the application level.
DDoS Protection Services: Use cloud-based DDoS mitigation services like Cloudflare, AWS Shield, or Akamai.
Load Balancing & Redundancy: Distribute traffic across multiple servers and implement auto-scaling to absorb attacks.
Application Hardening: Optimize code efficiency, database queries, and exception handling to prevent resource exhaustion.
Anomaly Detection & Monitoring: Use Intrusion Detection Systems (IDS) to detect unusual traffic patterns and respond proactively.
Timeout & Connection Limits: Set timeouts for incoming requests and limit the number of open connections per user.