Web security

1. **General methods:**
   1. **Encryption**: the method by which information is converted into secret code that hides the information’s true meaning.

* Funtions:
  + Confidentiality: Encodes the data to prevent it from being understood if it is intercrepted.
  + Authentication: Verifies the origin of the data that has been encrypted.
  + Integrity: Validates that the data has not been altered since it was encrypted.
  + Nonrepudiation: Prevents senders from denying they sent the encrypted data.
* Benefits:
  + Protect the confidentiality of digital data stored on computer systems or transmitted over the internet or other computer networks.
* Disadvantages:
  + Prevent the data’s owners from being able to access their own information.
  + The data owners might be permanently locked out of their data if the encryption keys get lost or destroyed.
  + Cybercriminals might also go after the encryption keys, rather than the data itself. Once they’ve accquired the keys, they can easily decipher the data.
* How does encryption work
  + Plaintext => encryption algorithms => encrypted data.
* Two types of encryption:
  + Symmetric ciphers: use a single key for both encrypting and decrypting data.
  + Asymmetric ciphers: use two different (but logically linked) keys for encrypting and decrypting data.
* Encryption algorithms: AES, DES, Diffie-Hellman key exchange, Elliptical curve cryptography (ECC), Quantum key distribution (QKD), RSA, Twofish.
  1. **Authentication:** the method of verifying the identity of a consumer or system to ensure they’re who they claim to be.
     + - Functions:
         * Indentification
         * Verification
         * Access control
         * Session management
         * Audit and monitoring
         * Multi-factor authentication (MFA)
         * Credential management
         * Interoperability
         * Encryption
         * Scalability
       - Types of authentication methods:
         * Password-based authentication
         * Certificate-based authentication
         * Certificate-based authentication
         * Biometric authentication
         * Token-based authentication
         * One-time password
         * Push notification
         * Voice authentication
         * Multifactor authentication
         * Two-factor authentication
  2. **Authorization**: the process of determining what actions, resources, or services a user or system has permission to access after authentication.
     + - Authorization methods
         * Role-based access control (RBAC)
         * Attributi-Based access control (ABAC)
         * Access control lists (ACLs)
         * Policy-based access Control (PBAC)
         * Open policy agent (OPA)
       - Authorization techniques
         * Token-based authorization
         * Session-based authorization
         * Oauth 2.0
         * Multi-level authorization
         * Contextual authorization
  3. **API Rate Limitting**: a technique for controlling the amount of incoming and outgoing traffic on an API by setting a predefined limit on how many requests an API user can make within a given timeframe.
     + - Algorithms:
         * Token bucket
         * Leaky bucket
         * Fixed window
         * Sliding log
       - Rate limiting alternatives:
         * Throttling
         * Spike control
         * Circuit breaking
  4. **Input validation**: the process of analyzing inputs and disallowing those which are considered unsuitable.
     + - Methods of input validation:
         * Client-side validation
         * Server-side validation
       - Techniques for input validation:
         * Data type validation
         * Length validation
         * Range validation
         * Format validation
         * Whitelist validation (allowlist)
         * Blacklist validation (denylist)
         * Cross-site scripting (XSS) prevention
         * SQL injection prevention
         * File validation
         * Custom validation

1. **Specific cases**
   1. **REST API:** a standardized way for systems to communicate over the internet using HTTP. It adheres to the principles of REST, a lightweight architectural style introduced by Roy Fielding in his dissertation.
      * + Key features of REST APIs:
          - Statelessness
          - Client-server architecture
          - Uniform interface
          - Resource-based
          - Cacheable
          - Layered system
          - Code on demand (Optional)
        + HTTP methods in REST API
          - GET
          - POST
          - PUT
          - DELETE
          - PATCH
        + Components of a REST API
          - Endpoints (URIs)
          - Headers
          - Resquest body
          - Response codes
          - Response body
        + Advantages of RESP API
          - Scalability
          - Flexibility
          - Simplicity
          - Cacheability
   2. **SOAP API:** a protocol-based approach for exchanging structured information in web services. It operates over various transport protocols (HTTP, SMTP, TCP,..) and is particularly useful in enterprise application requiring strong security, reliability, and formal contracts.
      * + Key features of SOAP APIs
          - Protocol-based
          - XML-based
          - Platform and Language Independence
          - Built-in error handling
          - Stateful or stateless
          - Transport flexibility
        + Structure of a SOAP Message
          - Envelope
          - Header (optional)
          - Body
          - Fault (optional)
        + Advantages of SOAP APIs
          - Standardized
          - Security
          - Built-in error handling
          - Extensibility
          - Transport protocol flexibility
          - Enterprise-grade
        + Disadvantages of SOAP APIs
          - Complexity
          - Performance Overhead
          - Limited human readability
          - Less flexibility
   3. **GraphQL API:** a query language and runtime for APIs that allows clients to request exactly the data they need, and nothing more.
      * + Key features of GraphQL API
          - Flexible querying
          - Single endpoint
          - Strongly typed schema
          - Real-time data
          - Hierachical structure
          - Introspection
        + Advantages of GraphQL
          - Efficient data fetching
          - Single endpoint
          - Real-time support
          - Strongly typed
          - Self-documenting
        + Disadvantages of GraphQL
          - Complexity
          - Performance overhead
          - Caching challenges
          - Overly flexible queries
   4. **Microservices architecture**: a software development style that structures an application as a collection of loosely coupled, independently deployable services. It promotes modularization and scalability, making it easier to build, maintain, and scale complex application.
      * + Key features of microservices architecture
          - modularity
          - independence
          - technology diversity
          - fault isolation
          - decentralized data management
          - scalability
          - API-based communication
        + Advantages of microservices
          - Scalability
          - Flexibility
          - Faster development
          - Fault tolerance
          - Easier maintenance
          - Continuous deployment
        + Disadvantages of microservices
          - Complexity
          - Data mangement challenges
          - Increased network traffic
          - Deployment overhead
          - Debugging and monitoring
   5. **Service mesh:** a dedicated infrastructure layer for managing service-to-service communication in a microservices architecture. It abstracts the complexity of handling communication between microservices and provides features such as observability, security, and traffic management.
      * + Key features of a service mesh
          - Service discovery
          - Load balancing
          - Traffic management
          - Security
          - Observability
          - Resilience
        + Key components of a service mesh
          - Sidecar proxy
          - Control plane
          - Ingress/egress gateways
          - Policy engine
          - Telemetry and monitoring
        + Popular service mesh solutions
          - Istio
          - Linkerd
          - Consul
          - AWS App mesh
          - Traefik mesh
          - Open service mesh (OSM)
        + Benefits of a service mesh
          - Enhanced security
          - Simplified traffic control
          - Improved observability
          - Fault tolerance
          - Decoupling application logic
   6. **Zero trust security:** a cybersecurity framework that operates on the principle of “nerver trust, always verify”. Unlike traditional security models that assume everything inside the network perimeter is safe, Zero Trust assumes potential threats can exist both inside and outside the network. It enforces strict identity verification and access control at every level to minimize risks.
      * + Core principles of zero trust security
          - Verify explicitly
          - Least privilege access
          - Assume breach
          - Segment and Isolate
          - Continuous monitoring
        + How zero trust works
          - Identity and access management (IAM)
          - Micro-segmentation
          - Device posture assessment
          - Continuous verification
          - Secure access to resources
        + Key components of Zero trust security
          - User authentication
          - Endpoint security
          - Data protection
          - Network security
          - Application security
          - Visibility and analytics
        + Benefits of Zero trust security
          - Enhanced security
          - Minimized attack surface
          - Improved compliance
          - Resilience against insider threats
          - Better adaptation to remote work
          - Reduced impact of breaches
   7. **API Gateway:** a server or software layer that acts as a single entry point for managing and routing API requests between clients and backend services. It simplifies interactions by handling request routing, authentication, rate limiting, and other cross-cutting concerns, enabling microservices or other backend architectures to operate more efficiently and securely.
      * + Core functiong of an API Gateway
          - Request routing
          - Protocol translation
          - Load balancing
          - Authentication and authorization
          - Rate limiting and throttling
          - Caching
          - Request/response transformation
          - Monitoring and analytics
        + How an API Gateway works
          - Client interaction
          - Request processing
          - Backend communication
          - Response to client
        + Benefits of using an API Gateway
          - Simplified client interaction
          - Improved security
          - Centralized control
          - Enhanced performance
          - Support for modern Architectures
          - Improved developer productivity
        + Key features of an API Gateway
          - Authentication and authorization
          - Traffic management
          - Monitoring and logging
          - API transformation
          - Caching
          - Security