Interview questions - **Network & Infrastructure Security**

1. What is the difference between a firewall and a reverse proxy?

Answer:

- **Firewall:** A firewall is a network security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. It typically works at the network layer and can be used to block or allow traffic based on factors like IP addresses, ports, and protocols.

  Key Functions:

  - Filters traffic based on rules.
  - Protects networks from unauthorized access.
  - Can be hardware or software-based.
  - May be configured for different levels of security (e.g., packet filtering, stateful inspection, or application-level inspection).

- **Reverse Proxy:** A reverse proxy is a server that sits between clients and web servers, routing requests from clients to one or more backend servers. Unlike a forward proxy (which forwards requests from clients to servers), a reverse proxy hides the identity of backend servers and adds an additional layer of security.

  Key Functions:

  - Distributes incoming traffic to multiple backend servers.
  - Masks the identity of backend servers.
  - Can provide caching, load balancing, SSL termination, and security features like protection from DDoS attacks.

---

2. How does rate limiting protect your backend services?

Answer:

- **Rate Limiting** is a technique used to control the amount of incoming requests to a server in a specified period. It prevents **overload**, **abuse**, and ensures fair use of resources. In the context of backend services, rate limiting helps protect them from:
    - **DDoS attacks** (Distributed Denial of Service) where an attacker floods your service with a large number of requests.
    - **API abuse** where users (or malicious actors) send more requests than they are allowed to, either accidentally or intentionally, overloading the service.
    - **Ensuring fair usage** by limiting the number of requests a user can make within a time period (e.g., 1000 requests per hour).
- **Common Techniques:**
    - **Fixed Window Rate Limiting:** Limits the number of requests in a fixed time period (e.g., 100 requests per hour).
    - **Leaky Bucket or Token Bucket:** Requests are allowed through at a steady rate, and excess requests are delayed or rejected.
    - **Exponential Backoff:** In case of overloading, the system gradually increases the delay between requests.

---

3. Explain the Zero Trust security model and its importance.

Answer:

- **Zero Trust** is a security model based on the concept of **"Never Trust, Always Verify."** In a Zero Trust model, **trust is never assumed**, regardless of whether the request comes from inside or outside the network. All access requests, even from trusted internal networks, must be authenticated and authorized.

  Key Principles:
    - **Identity and Access Management (IAM):** Every user or device is treated as untrusted and must prove its identity before accessing resources.
    - **Least Privilege Access:** Users and devices are only granted the minimum level of access they need to perform their functions.
    - **Micro-Segmentation:** Network traffic is segmented to limit access to critical resources.

    - **Continuous Monitoring:** All activity is continuously monitored, and behavior anomalies are flagged.

- **Importance:**
    - Reduces the attack surface by ensuring no one has unrestricted access to the network.
    - Mitigates **lateral movement** by attackers in case they gain access to one part of the network.
    - Essential for modern cloud and hybrid infrastructures where perimeter-based security is less effective.

---

4. How would you secure a containerized workload in Kubernetes?

Answer: To secure a containerized workload in **Kubernetes**, you should consider multiple layers of security:

- **Container Image Security:**
    - Use trusted, minimal base images (e.g., Alpine Linux).
    - **Scan container images** for vulnerabilities (using tools like **Clair** or **Trivy**).
    - **Sign container images** to ensure integrity and authenticity.
- **Kubernetes Role-Based Access Control (RBAC):**
    - Use **RBAC** to enforce the principle of least privilege by defining roles and permissions for users and service accounts.
    - Ensure only authorized entities can deploy or modify workloads.
- **Pod Security Policies (PSPs):**
    - Use **PSPs** to restrict the capabilities of containers, such as limiting root access, disabling privilege escalation, or preventing containers from running as root.
- **Network Policies:**
    - Implement **Network Policies** to control the communication between pods, limiting the attack surface.

- **Service Mesh (e.g., Istio):**
    - Secure service-to-service communication with **mTLS** (mutual TLS) and encryption.
- **Secrets Management:**
    - Use Kubernetes **Secrets** to securely store sensitive data (e.g., passwords, API keys) and avoid hardcoding them in the application code.

---

5. What are some common OWASP Top 10 vulnerabilities and how do you mitigate them?

Answer:

- **Injection:** (e.g., SQL Injection)
    - **Mitigation:** Use **prepared statements** and **parameterized queries**.
- **Broken Authentication:**
    - **Mitigation:** Implement **strong authentication mechanisms**, use multi-factor authentication (MFA), and securely store passwords (e.g., hash + salt).
- **Sensitive Data Exposure:**
    - **Mitigation:** Use **encryption** for data in transit (e.g., HTTPS, TLS) and at rest (e.g., AES). Store passwords securely using hashing and salting techniques.
- **XML External Entities (XXE):**
    - **Mitigation:** Disable external entities in XML parsers and validate inputs.
- **Broken Access Control:**
    - **Mitigation:** Implement role-based access control (RBAC) and always check user permissions before granting access.
- **Security Misconfiguration:**
    - **Mitigation:** Follow the **principle of least privilege** and review your system configuration regularly. Use tools for configuration management (e.g., Chef, Puppet).

- **Cross-Site Scripting (XSS):**
    - **Mitigation:** Sanitize and validate all user inputs and use content security policies (CSP).
- **Insecure Deserialization:**
    - **Mitigation:** Avoid deserialization of untrusted data and use signed objects to prevent manipulation.
- **Using Components with Known Vulnerabilities:**
    - **Mitigation:** Regularly update libraries and frameworks. Use dependency management tools to track known vulnerabilities.
- **Insufficient Logging & Monitoring:**
    - **Mitigation:** Enable proper **logging** and **monitoring** for suspicious activity and ensure logs are tamper-proof.

---

6. How does a service mesh help enforce security in microservices?

Answer:

- A **service mesh** is a dedicated infrastructure layer that manages service-to-service communication in microservices architectures. It enforces security by providing:
    - **Mutual TLS (mTLS):** All traffic between services is encrypted, and each service authenticates the other using certificates.
    - **Access Control:** Service mesh can enforce strict **access policies**, ensuring that only authorized services can communicate with each other.
    - **Traffic Monitoring and Logging:** It can monitor traffic and log service interactions, helping to detect anomalies or attacks.
    - **Rate Limiting & Circuit Breaking:** These features can protect services from being overwhelmed or misused.
- Popular service meshes like **Istio** and **Linkerd** provide built-in security features to protect microservices environments.

---

7. What's the role of IAM in cloud security?

Answer:

- **IAM (Identity and Access Management)** is crucial in **cloud security** as it governs how users, systems, and applications access resources in the cloud.

  Key Functions:

  - **Authentication:** Confirms the identity of a user or system (using passwords, MFA, or other authentication mechanisms).
  - **Authorization:** Ensures the authenticated entity has permission to access specific cloud resources (e.g., using roles and policies).
  - **Granular Permissions:** IAM allows fine-grained control over who can access what resources and what actions they can perform, adhering to the **least privilege** principle.
  - **Audit Logging:** Logs all access requests and activities for monitoring and compliance purposes.

- In cloud environments (AWS, Azure, GCP), IAM plays a critical role in ensuring only the right users and services have the appropriate permissions to interact with cloud resources, reducing the risk of unauthorized access.