1. **Security at the Edge**: By validating API keys at the WAF, which usually sits at the edge, you can reject unauthorized requests earlier in the processing pipeline. This can provide an additional layer of security.
2. **Reduced Load on Backend Servers**: Since unauthorized requests are blocked at the WAF level, there is a reduction in the number of requests that hit your backend servers. This can reduce the load on your servers and save bandwidth.
3. **DDoS Protection**: WAFs are typically also equipped to handle DDoS attacks. By integrating API key checking with the WAF, you can make sure that your API is even more secure against such attacks.
4. **Centralized Security Policies**: Having API key checks at the WAF can help in centralizing security policies and rules which can be more efficient and easier to manage especially if you have multiple APIs or services.

Disadvantages:

1. **Limited Flexibility and Logic**: WAFs are not as flexible as application code. Complex authorization logic (e.g., rate limiting, advanced role-based access control) might be difficult or impossible to implement at the WAF level.
2. **Increased Latency**: While WAFs can reduce the load on your servers, they can also introduce an additional layer that every request must pass through, which might slightly increase latency.
3. **Maintenance Overhead**: Managing security rules at both the WAF and application levels can lead to an increased maintenance overhead, especially if the rules are not kept in sync.
4. **Cost**: WAF services usually come with an associated cost. Depending on the traffic your application is handling, this cost can be significant, especially if you are paying for a third-