**Why Does a Rejected Ticket Trigger a Full TLS Handshake?**

**Invalid or Rejected TLS Ticket**

The server cannot reuse previously derived keys.

It must therefore initiate a new, full TLS handshake to establish the security context.

**Additional Cost for the Server**

A full handshake requires more cryptographic computation and message exchanges.

Under high load (many simultaneous requests), this can stress the server’s resources.

**Mitigations and Defense Mechanisms**

Despite this possibility, most servers (and CDNs, load balancers, etc.) implement mechanisms to mitigate attacks targeting the TLS handshake:

**Rate Limiting**

The server may limit the number of new connections or full handshakes allowed within a certain time frame.

**IP Reputation**

Servers can monitor IP addresses (or address ranges) that issue an abnormal amount of tickets/handshakes and apply blocking or throttling rules.

**Cryptographic Resilience**

Optimizing cryptographic algorithms and using hardware acceleration (such as TLS accelerator cards) enable faster handling of numerous handshakes.

**Secure Session Resumption**

Even if a TLS ticket is rejected, servers might leverage caching methods (session IDs) or other mechanisms (0-RTT) to reduce handshake costs for legitimate cases.

**Conclusion**

While it is technically possible to push a server into performing numerous full TLS handshakes by sending invalid tickets, most properly configured and protected servers use various techniques (rate limiting, filtering, hardware acceleration, etc.) to limit the risk of denial-of-service. Consequently, a script that floods a server with fake TLS tickets usually has limited practical impact on a well-hardened server.