1. ¿Qué significa `maxlength="13"` en un `<input>`?

Cuando ves este atributo:

```
<input type="text" name="uname" placeholder="email" maxlength="13">
```

Significa que **el navegador no permitirá al usuario escribir más de 13 caracteres** en ese campo. Sin embargo:

> **Esto es una validación del lado cliente (frontend), no del servidor.**
> Si el atacante modifica el HTML o usa herramientas como Burp Suite, curl o DevTools, puede enviar más de 13 caracteres igual.

2. ¿Qué sugiere esto sobre la base de datos?

Si el formulario impone un `maxlength="13"` para el campo de usuario, probablemente **la columna en la base de datos también está definida con una longitud fija o restringida**, como:

```
CREATE TABLE users (
  uname VARCHAR(13),
  ...
)
```

Esto hace sospechar de una posible vulnerabilidad por **truncamiento**, especialmente si:

- La aplicación no valida la longitud en el backend.
- La base de datos **no rechaza** entradas más largas, sino que **las trunca silenciosamente**.

3. ¿Cómo se explota esto?

Supongamos que en la base de datos existe este usuario:

```
username: jacob@tornado
password: (hash de algo)
```

Y tú haces lo siguiente:

- Modificas el campo en el navegador o con una herramienta para que acepte **15 caracteres** (`maxlength="15"`).
- En el formulario de **registro**, envías nuevamente `jacob@tornado` con una contraseña tuya:

```
POST /signup.php
uname=jacob@tornado
password=mipass
```

Si la base de datos **trunca automáticamente el nombre de usuario a 13 caracteres**, el sistema cree que estás intentando registrar `jacob@tornado`, es decir **el mismo nombre** que ya existía.

Pero si **no hay una restricción de unicidad** bien aplicada o **la lógica del sistema reemplaza registros existentes**, puedes terminar **sobrescribiendo el usuario original** con tu propia contraseña.

> O incluso, si se permite registro duplicado, al truncarse ambos usuarios a los mismos primeros 13 caracteres, cuando inicies sesión con `jacob@tornado`, podrías autenticarte con tu nueva cuenta en vez de la original.

Conclusión

Esto es un ejemplo clásico de un **SQL Truncation Attack**:

- El atacante **envía un valor más largo que el permitido**.
- La base de datos **lo trunca silenciosamente** a la longitud máxima.
- El atacante logra **colisión con un valor ya existente**, sobrescribiendo información o autenticándose sin conocer la contraseña real.