Programación Segura

Sebastián Castillo Saavedra

Introducción

Se ha solicitado el análisis de tres vulnerabilidades comunes presentes en aplicaciones web, de una lista de diez vulnerabilidades.

Los ataques cibernéticos pueden tener muchas manifestaciones. En este caso hemos escogido ejemplos cuidadosamente seleccionados en los que se pueda apreciar el html de interés, ya sea como resultado del ataque o para embeber código malicioso.

Objetivos

Describir tres tipos de ataques, a saber:

- SQL injection
- Control de acceso roto
- Cross site request forgery

Para cada uno de estos ataques explicaremos los siguientes puntos

- Descripción de la vulnerabilidad
- Ejemplo de la vulnerabilidad y su manifestación en HTML
- Cómo evitar la vulnerabilidad.

Consideraciones generales

Los ejemplos expuestos en este informe provienen del sitio web portswigger.com

Los ejemplos fueron realizados en laboratorios especialmente preparados para estos fines.

Mostraremos sólo las manifestaciones de estos ataques en el código HTML.

La prevención de estos ataques siempre provienen de lenguajes de programación o de una correcta configuración del entorno de producción. HTML al ser un lenguaje de marcado de etiquetas, no tiene una capacidad inherente para ser seguro o inseguro y solo se limita a dar estructura a los sitios web.

Debemos tener en cuenta que estas vulnerabilidades provienen de la arquitectura de la aplicación.

SQL Injection

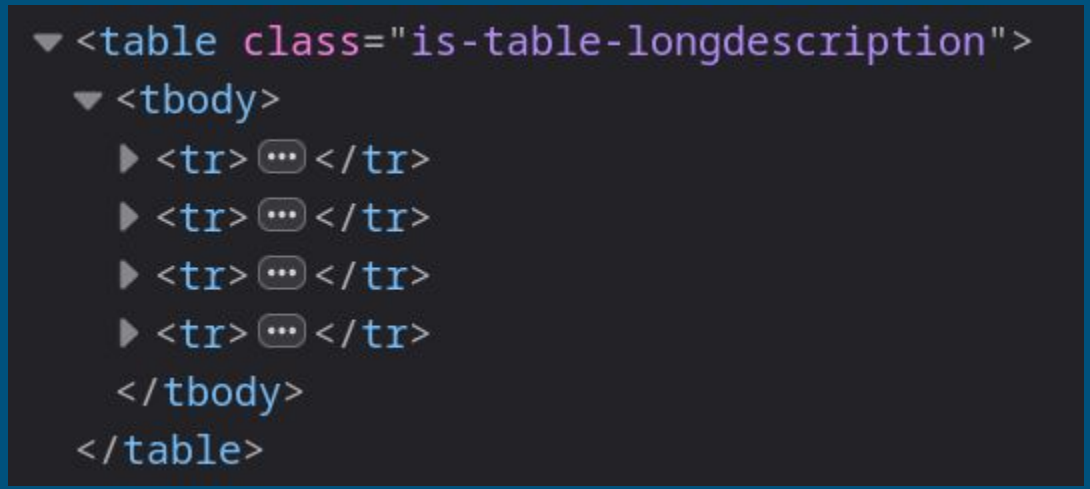
¿Qué es SQL Injection?

La inyección SQL (SQLi) es una vulnerabilidad de seguridad web que permite a un atacante interferir con las consultas que realiza una aplicación a su base de datos. Esto puede permitir que un atacante vea datos que normalmente no puede recuperar. Esto podría incluir datos que pertenecen a otros usuarios o cualquier otro dato al que pueda acceder la aplicación. En muchos casos, un atacante puede modificar o eliminar estos datos, provocando cambios persistentes en el contenido o el comportamiento de la aplicación.

En algunas situaciones, un atacante puede escalar un ataque de inyección SQL para comprometer el servidor subyacente u otra infraestructura de back-end. También puede permitirles realizar ataques de denegación de servicio.

¿Cómo se manifiesta una inyección SQL?

En un entorno especialmente preparado, podemos apreciar los resultados de una petición sin atacar su vulnerabilidad



¿Cómo se manifiesta una inyección SQL?

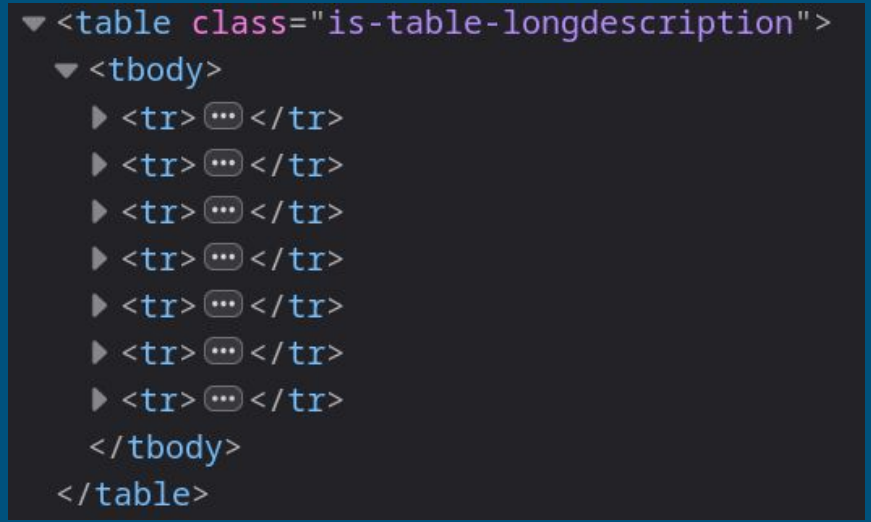
Introduciremos el siguiente fragmento de código al final de la url:

```
' UNION SELECT username, password FROM users--
```

Inspeccionando el código HTML

vemos que se han incorporado

3 nuevos elementos a la tabla.



¿Cómo se manifiesta una inyección SQL?

Al analizar más detalladamente los elementos de la tabla, podemos apreciar que una de sus filas contiene el nombre de usuario y la contraseña de administrador.

¿Cómo prevenir la vulnerabilidad SQL injection?

Podemos evitar la inyección SQL utilizando consultas parametrizadas en lugar de concatenación de cadenas dentro de la consulta. Estas consultas parametrizadas también se conocen como "prepared statements".

Para que una consulta parametrizada sea eficaz a la hora de prevenir la inyección de SQL, la cadena que se utiliza en la consulta siempre debe estar codificada. Nunca debe contener datos variables de ningún origen. No debemos decidir caso por caso si un dato es confiable y continuar usando la concatenación de cadenas dentro de la consulta para los casos que se consideren seguros. Es fácil cometer errores sobre el posible origen de los datos o que cambios en otro código contaminen los datos confiables.

Control de acceso roto

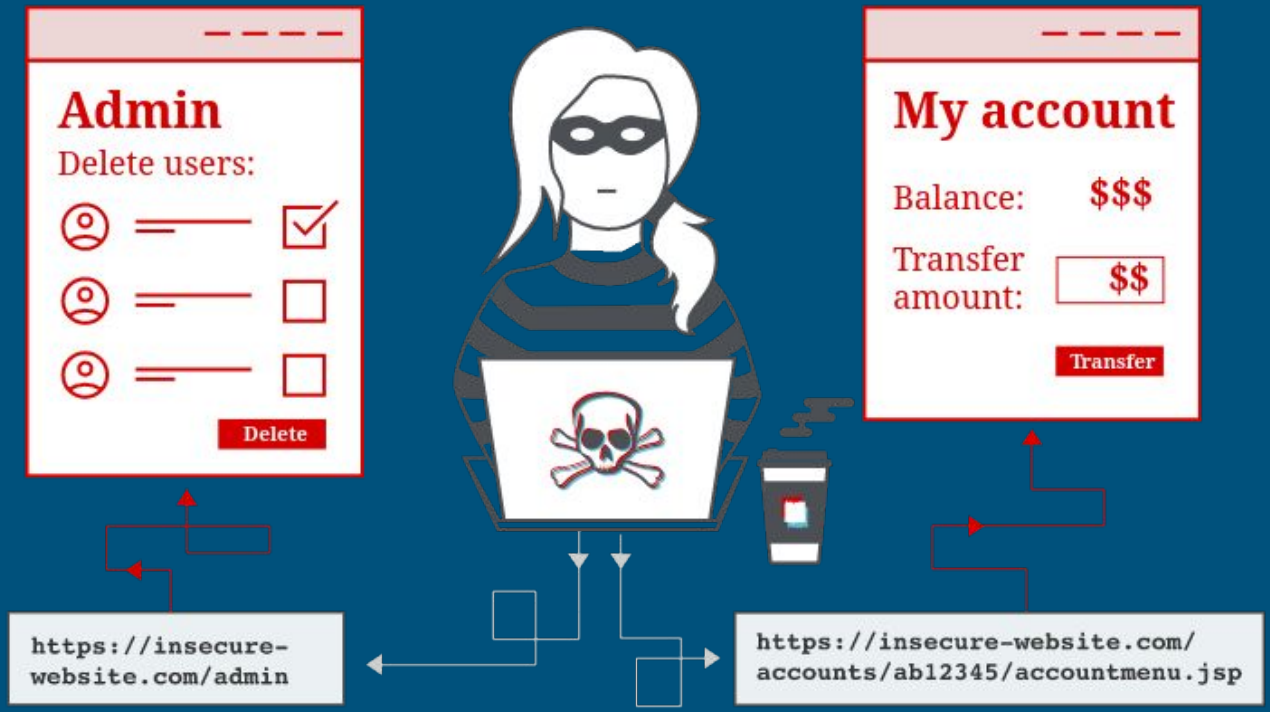
¿Qué es control de acceso roto?

El control de acceso es la aplicación de restricciones sobre quién o qué está autorizado a ejecutar acciones o a acceder a recursos. En el contexto de aplicaciones web, el control de acceso depende de la autenticación y la gestión de sesiones:

- Autenticación: Confirma que el usuario es quien dice ser.
- Gestión de sesión: Identifica cuales son las peticiones HTTP que se realizarán posteriormente por ese mismo usuario.
- Control de acceso: Determina si el usuario tiene permitido llevar a cabo la acción que intenta ejecutar.

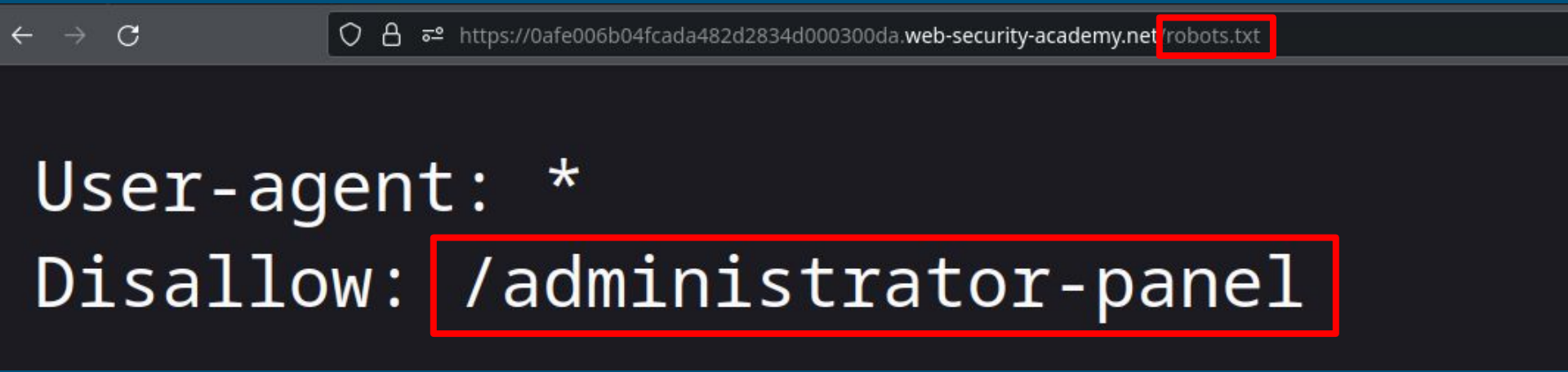
Los controles de acceso roto son comunes y a menudo presentan una vulnerabilidad de seguridad crítica. El diseño y la gestión de control de acceso es un problema complejo y dinámico que se aplica a restricciones de negocios, organizacionales y legales para una implementación técnica. Las decisiones de diseño de control de acceso están confeccionados por humanos, por lo tanto, el potencial de errores es alto.

¿Qué es control de acceso roto?



¿Cómo se manifiesta un control de acceso roto?

En un entorno especialmente preparado podemos ver que se expone el archivo robots.txt, el cual me indica que hay una ruta que no debe ser indexada por los motores de búsqueda.



¿Cómo se manifiesta un control de acceso roto?

Al entrar a la ruta indicada /administrator-panel podemos apreciar el siguiente código html. Nos interesa el atributo href de las etiquetas “a”

```
<div>
  <span>wiener -</span>
  <a href="/administrator-panel/delete?username=wiener">Delete</a>
</div>
<div>
  <span>carlos -</span>
  <a href="/administrator-panel/delete?username=carlos">Delete</a>
</div>
```

Si observamos los cuadros resaltados en rojo podemos apreciar una petición al servidor que nos permite eliminar cuentas de usuario. Si entramos a esa ruta, entonces podremos apreciar su eliminación

¿Cómo prevenir la vulnerabilidad control de acceso roto?

Las vulnerabilidades del control de acceso se pueden prevenir adoptando un enfoque de defensa en profundidad y aplicando los siguientes principios:

- Nunca confíe únicamente en la ofuscación para el control de acceso.
- A menos que un recurso esté destinado a ser accesible públicamente, deniegue el acceso de forma predeterminada.
- Siempre que sea posible, utilice un único mecanismo para toda la aplicación para hacer cumplir los controles de acceso.
- A nivel de código, haga obligatorio que los desarrolladores declaren el acceso permitido para cada recurso y deniegue el acceso de forma predeterminada.
- Audite y pruebe minuciosamente los controles de acceso para garantizar que funcionen según lo diseñado.

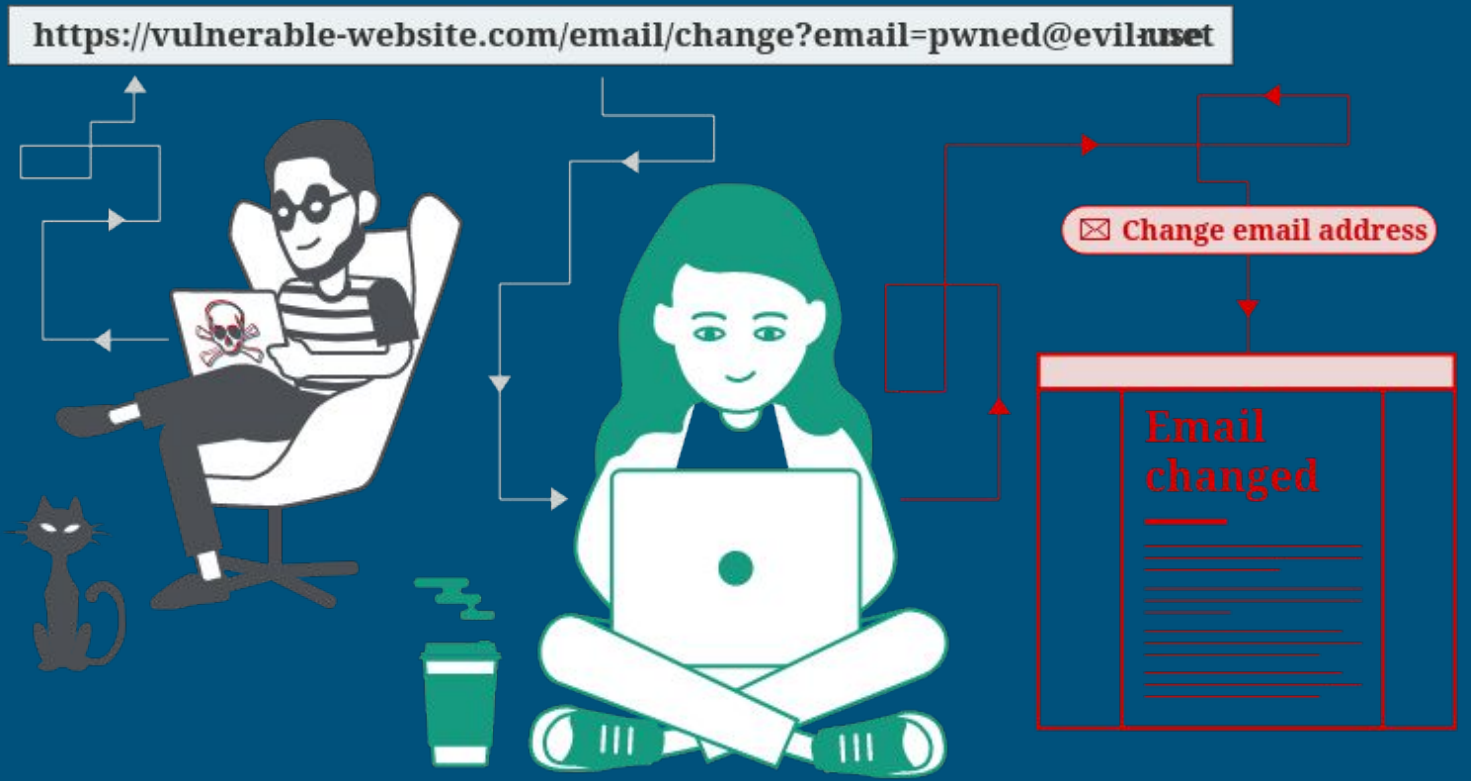
Cross site request forgery (CSRF)

¿Qué es CSRF?

Cross-site request forgery (también conocido como CSRF) es una vulnerabilidad de seguridad web que permite a un atacante inducir al usuario ejecutar acciones que él no pretende realizar. Esto permite a un atacante eludir parcialmente la misma política de origen, que está diseñada para evitar que diferentes sitios web interfieran entre sí.

En un ataque CSRF exitoso, el atacante provoca al usuario víctima llevar a cabo una acción de forma no intencional. Por ejemplo, esto podría ser cambiar la dirección de email de su cuenta, cambiar su contraseña o realizar transferencias bancarias.

¿Qué es CSRF?


https://vulnerable-website.com/email/change?email=pwned@evil-user.net
Change email address
Email changed

¿Qué es CSRF?

Si un usuario víctima visita la página web del atacante, sucederá lo siguiente:

- La página del atacante activará una solicitud HTTP al sitio web vulnerable.
- Si el usuario inicia sesión en el sitio web vulnerable, su navegador incluirá automáticamente su cookie de sesión en la solicitud
- El sitio web vulnerable procesa la solicitud de la forma habitual, la tratará como si hubiera sido realizada por el usuario víctima y cambiará su dirección de correo electrónico.

Nota: Aunque CSRF normalmente se describe en relación con el manejo de sesiones basado en cookies, también surge en otros contextos donde la aplicación agrega automáticamente algunas credenciales de usuario a las solicitudes, como la autenticación básica HTTP y la autenticación basada en certificados.

¿Cómo se manifiesta un ataque CSRF?

Por ejemplo, suponga que una aplicación contiene una función que permite al usuario cambiar la dirección de email en su cuenta. Cuando un usuario lleva a cabo esta acción, realiza una petición HTTP como la siguiente:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE

email=wiener@normal-user.com
```

¿Cómo se manifiesta un ataque CSRF?

Podemos utilizar un fragmento de código como el de la primera imagen.

Este formulario lleva a cabo una petición POST a la url donde se cambia la dirección de correo.

Creamos un input oculto con el valor de correo que deseamos

Ejecutamos un fragmento de código javascript para llevar a cabo la operación de envío del formulario, con los valores modificados.

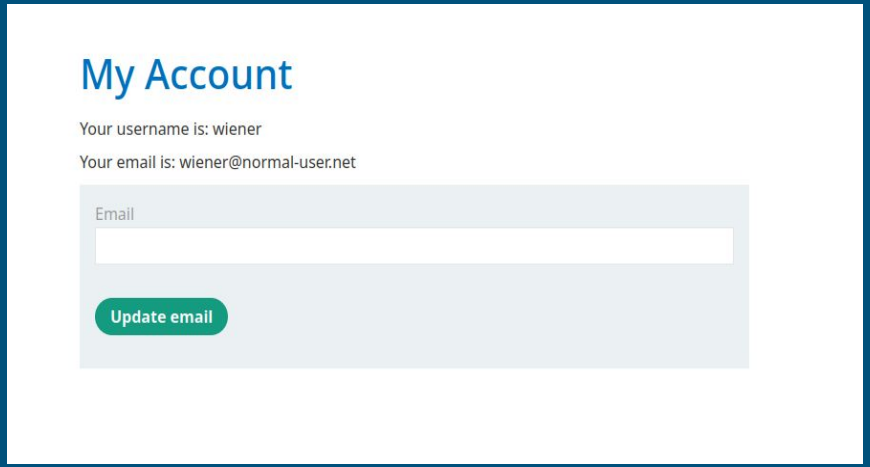
```
<h1>Falsificación de solicitudes del lado del cliente</h1>
<iframe id="csrf-iframe" name="csrf-iframe" title="csrf-iframe"></iframe>
<form
    action="https://0aa300190431f42d8147d47f004200d4.web-security-academy.
    method="POST"
    target="csrf-iframe"
    id="csrf-form"
    >
    <input type="hidden" name="email" value="test@test.tst">
</form>
<script>
    document.getElementById('csrf-form').submit()
</script>
```

```
<style>
    #csrf-iframe {
        display: none;
    }
</style>
```

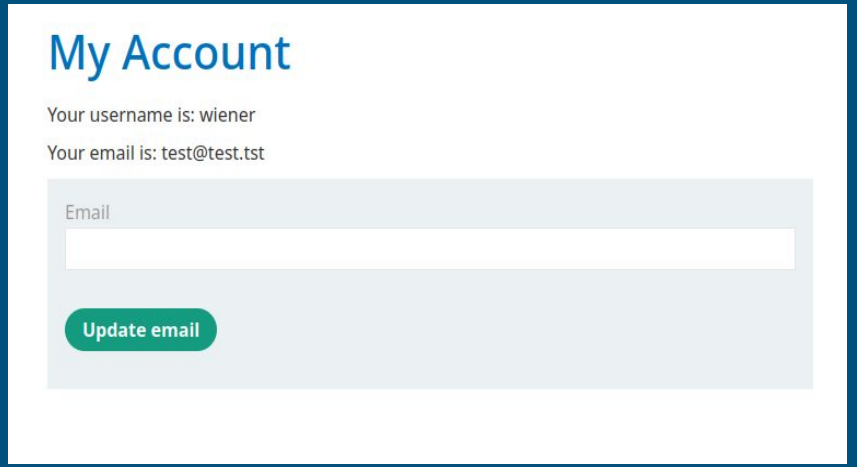
Con esto, ocultamos el formulario, para que pase desapercibido a la víctima

¿Cómo se manifiesta un ataque CSRF?

Cuenta sin vulnerar



Cuenta vulnerada



¿Cómo prevenir un ataque CSRF?

La forma más sólida de defenderse contra los ataques CSRF es incluir un token CSRF en las solicitudes relevantes. El token debe cumplir los siguientes criterios:

- Impredecible con alta entropía, como ocurre con los tokens de sesión en general.
- Vinculado a la sesión del usuario.
- Estrictamente validado en todos los casos antes de ejecutar la acción correspondiente.

Conclusión

Hemos podido revisar las vulnerabilidades, SQL injection, Control de acceso roto y CSRF.

Para cada una de ellas hemos explicado en qué consiste la vulnerabilidad, de qué forma se manifiesta en un código html y de qué manera las podemos prevenir.

Cabe señalar que estas vulnerabilidades no siempre tienen una manifestación directa en el frontend, por lo cual los ejemplos mostrados, han sido seleccionados para exponer en código html el efecto de explotarlas.