

# Informe Laboratorio 3

## Sección 4 Laboratorio

Benjamín Pérez Ponce  
e-mail: benjamin.perez11@mail.udp.cl

Octubre de 2025

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades según criterio de rúbrica</b>	<b>2</b>
2.1. Identifica el algoritmo de hash utilizado al momento de registrarse en el sitio	3
2.2. Identifica el algoritmo de hash utilizado al momento de iniciar sesión . . . . .	4
2.3. Genera el hash de la contraseña desde la consola del navegador . . . . .	6
2.4. Intercepta el tráfico login con BurpSuite . . . . .	7
2.5. Realiza el intento de login por medio del hash . . . . .	8
2.6. Identifica las políticas de privacidad o seguridad . . . . .	11
2.7. Comente 4 conclusiones sobre la seguridad del sitio escogido . . . . .	11
2.8. Comparación: MD5 vs otros algoritmos/funciones criptográficas . . . . .	12
2.9. Comentarios . . . . .	13

## 1. Descripción de actividades

Su objetivo será auditar la implementación de algoritmos hash aplicados a contraseñas en páginas web desde el lado del cliente, así como evaluar la efectividad de estas medidas contra ataques de tipo Pass the Hash (PtH). Para llevar a cabo esta auditoría, deberá registrarse en un sitio web y crear una cuenta, ingresando una contraseña específica para realizar las pruebas.

Al concluir la tarea, es importante que modifique su contraseña por una diferente para garantizar su seguridad.

Dado que la cantidad de sitios chilenos que utilizan hash es limitada, se permite realizar esta tarea en cualquier sitio web a nivel mundial. En este sentido, realice las siguientes actividades:

- Identificación del algoritmo de hash utilizado para las contraseñas al momento del registro en el sitio.
- Identificación del algoritmo de hash utilizado para las contraseñas al momento de iniciar sesión.
- Generación del hash de la contraseña desde la consola del navegador, partiendo de la contraseña en texto plano.
- Interceptación del tráfico de login utilizando BurpSuite desde su equipo.
- Realización de un intento de login modificando la contraseña por una incorrecta haciendo uso del hash obtenido en el punto anterior. Puede interceptar el tráfico y modificar el hash por el correcto o hacer uso del servicio repeater de BurpSuite.
- Descripción de las políticas de privacidad o seguridad relacionadas con las contraseñas, incluyendo un enlace a las mismas.
- Cuatro conclusiones sobre la seguridad o vulnerabilidad de la implementación observada.

## 2. Desarrollo de actividades según criterio de rúbrica

Para cumplir con el desarrollo de las actividades se debe buscar un sitio que cifre la contraseña con algún tipo de *hash* para eso se utilizó el sitio <https://publicwww.com/> este sitio nos permite filtrar con palabras clave que estén adjuntas ya sea en el código *HTML* de la página o en el código *JavaScript*.

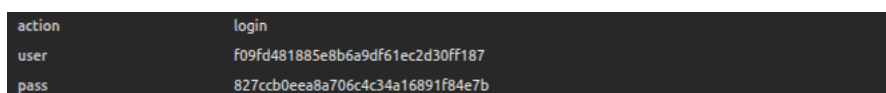
Como palabra clave en el filtro primeramente se utilizó `'md5'`. El problema de utilizar este filtro es que el resultado es muy amplio y no especifica en que parte de la página se está utilizando el hash.

Por lo que se decide cambiar el filtro a `'CryptoJS.MD5(password)'` esto se debe a que así podemos ver las paginas que involucren la función en JS que tengan MD5 e involucren el

cifrado en una password.

Con esto se procede a la búsqueda del sitio, primeramente se encontró el sitio `https://clientes.gestonova.es/`, se validó que la contraseña estaba filtrada con el tipo de hash como podemos notar en la Figura 1, no obstante este sitio presentaba una inviabilidad a la hora de registrarse que ya quera limitado solo para clientes, por lo que pedía una validación.

Se continuó la búsqueda en `https://publicwww.com/` y finalmente se llego al sitio `https://laboratoriodelformaggio.com/`. En el cual si se pudo realizar correctamente el desarrollo del laboratorio.



action	login
user	F09fd481885e8b6a9df61ec2d30ff187
pass	827ccb0eea8a706c4c34a16891f84e7b

Figura 1: Pagina1 Evidencia

### 2.1. Identifica el algoritmo de hash utilizado al momento de registrarse en el sitio

Para proceder al registro y posterior ingreso, se utiliza el sitio web `https://temp-mail.org/` para crear un email temporal para el registro en el sitio, luego de eso se procede al registro y a la identificación del algoritmo de hash.

Se utiliza el correo '`kexec19423@bdnets.com`' y la password '`12345`'

Al momento de registrarse en el sitio con la tecla '**F12**' dirigiéndonos al apartado de '**Network**' en la inspección web, podemos notar que hay una función en JavaScript llamada '**md5.js**' esta función nos permite identificar el algoritmo de hash utilizado, el cual es md5. Como podemos ver en la Figura 2 una vez se procede al registro la pagina dirige automáticamente a otra sección pero de todas formas podemos visualizar la función.

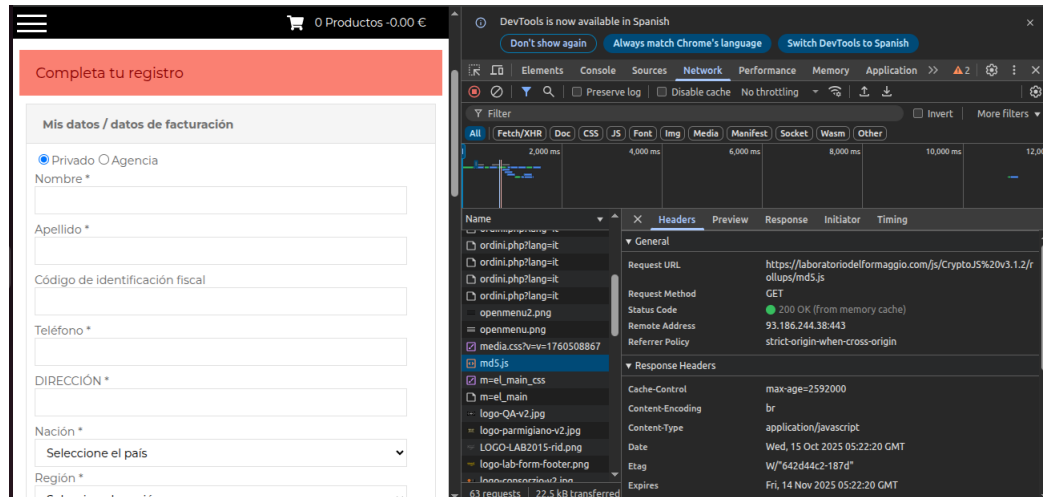


Figura 2: Función Hash registro

Al seleccionar la preview podemos acceder al código de la función el cual nos indica que se utilizó a librería '**CryptoJS v3.1.2**' en el cliente. Se adjunta la Figura 3 con la evidencia, pero no con el código completo por temas de longitud, ya que la función mantiene mas de 300 lineas (Se adjuntará el código completo en el repositorio).

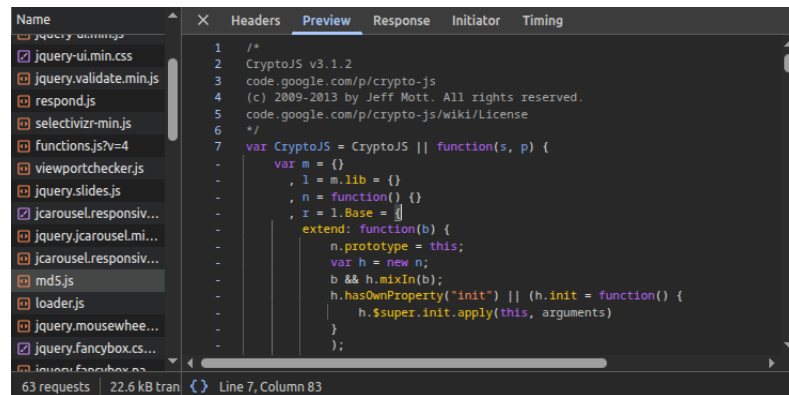


Figura 3: Función MD5 registro

## 2.2. Identifica el algoritmo de hash utilizado al momento de iniciar sesión

Al momento de iniciar sesión verificamos nuevamente con el inspector web, lo mas importante en este paso es verificar si la contraseña está siendo cifrada por la función hash md5. Como podemos ver en la Figura 4 la función presenta el siguiente hash para el campo *password* 827ccb0eea8a706c4c34a16891f84e7b.

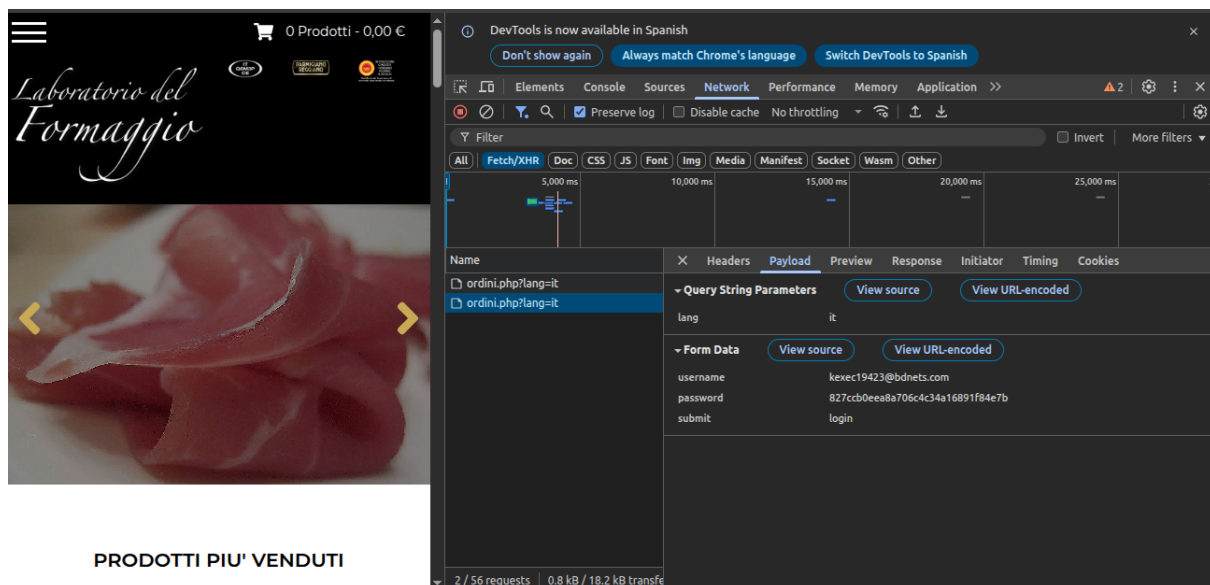


Figura 4: Password con Hash MD5 en login

Para la verificación previa de si la contraseña fue enviada con hash, solo se filtraron los paquetes post, a continuación necesitamos verificar los paquetes que contengan JS para revisar si la función hash que se envió en el login es la misma función utilizada en el register. En la siguiente Figura 5 podemos ver que la función por lo que se corrobora que el algoritmo hash fue identificado correctamente.

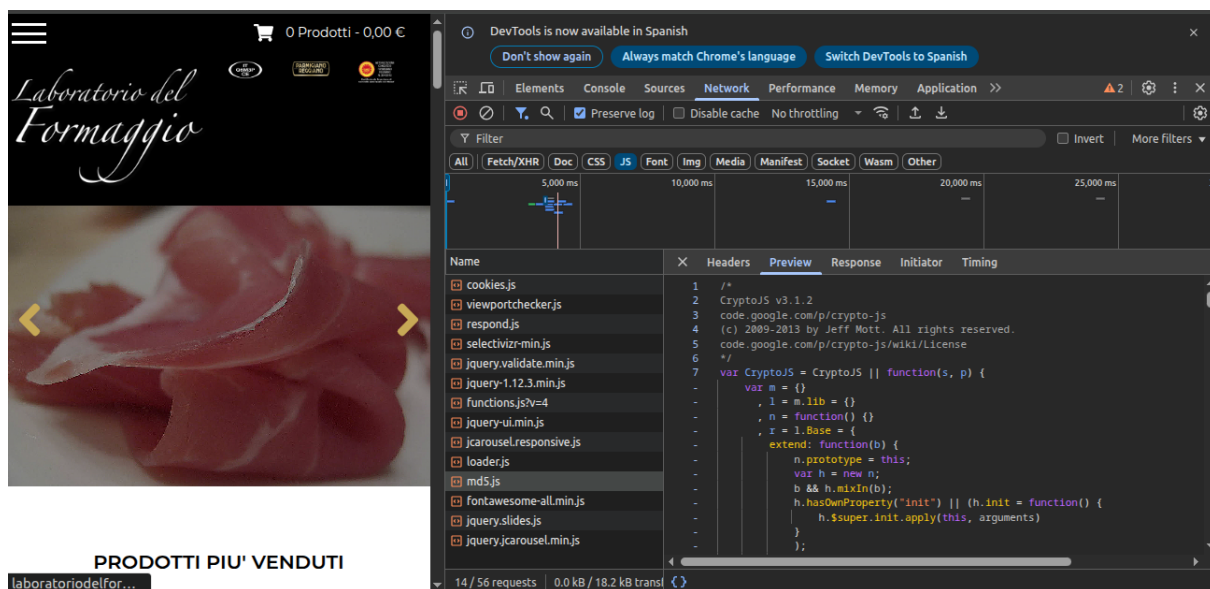


Figura 5: Funcion MD5 en login

### 2.3. Genera el hash de la contraseña desde la consola del navegador

Para el siguiente paso en la consola del navegador se utilizaron los siguientes comandos con el fin de ver si se puede generar con la función md5 el hash obtenido al ingresar la contraseña.

- Existencia de algoritmo hash:  
`console.log('CryptoJS.MD5:', typeof (window.CryptoJS && window.CryptoJS.MD5));`
- Verificación de si la función hashea la contraseña.  
`const plain = '12345';  
const hash = CryptoJS.MD5(plain).toString();  
console.log('MD5 de', plain, '=>', hash);`

A continuación como podemos observar en la Figura 6 el output que nos entrega la consola para verificar si el hash es generado correctamente en la consola del navegador.

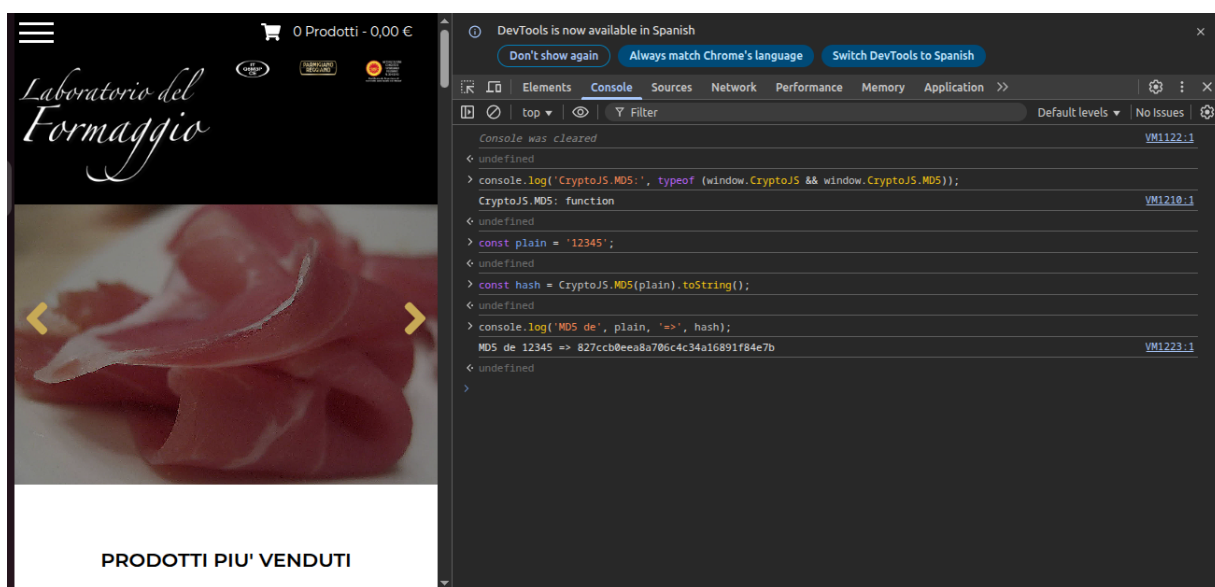


Figura 6: Generación del hash por consola

Como notamos en la figura el hash obtenido es 827ccb0eea8a706c4c34a16891f84e7b el mismo obtenido en los pasos anteriores, por lo tanto se corrobora el correcto funcionamiento de la función en el sitio web.

Como verificación adicional, se busca un sitio que genere hash md5 para verificar si el resultado otorgado por la función que cifra se mantiene en los estándares de hash md5. Se utiliza el sitio web <https://www.md5hashgenerator.com/es/> para verificar.

A continuación se adjunta la Figura 7 que muestra que el hash generado por la contraseña '12345' también es el mismo.

### Generador de hash MD5

Utilice este generador para crear un hash MD5 de una cadena:

12345

Generar →

Tu Cadena	12345
MD5 Hash	827ccb0eea8a706c4c34a16891f84e7b

Copia

Figura 7: Generación del hash por web MD5

## 2.4. Intercepta el tráfico login con BurpSuite

Para este paso se utilizó el *browser* que otorga Burpsuite, se activa la función '**Intercept = ON**' para así una vez se ingrese las credenciales correctas solo queda registrado el paquete tipo POST con la información del tráfico del login.

A continuación se adjunta la Figura 8 que muestra el paquete analizado por Burpsuite, que tiene la información del login.

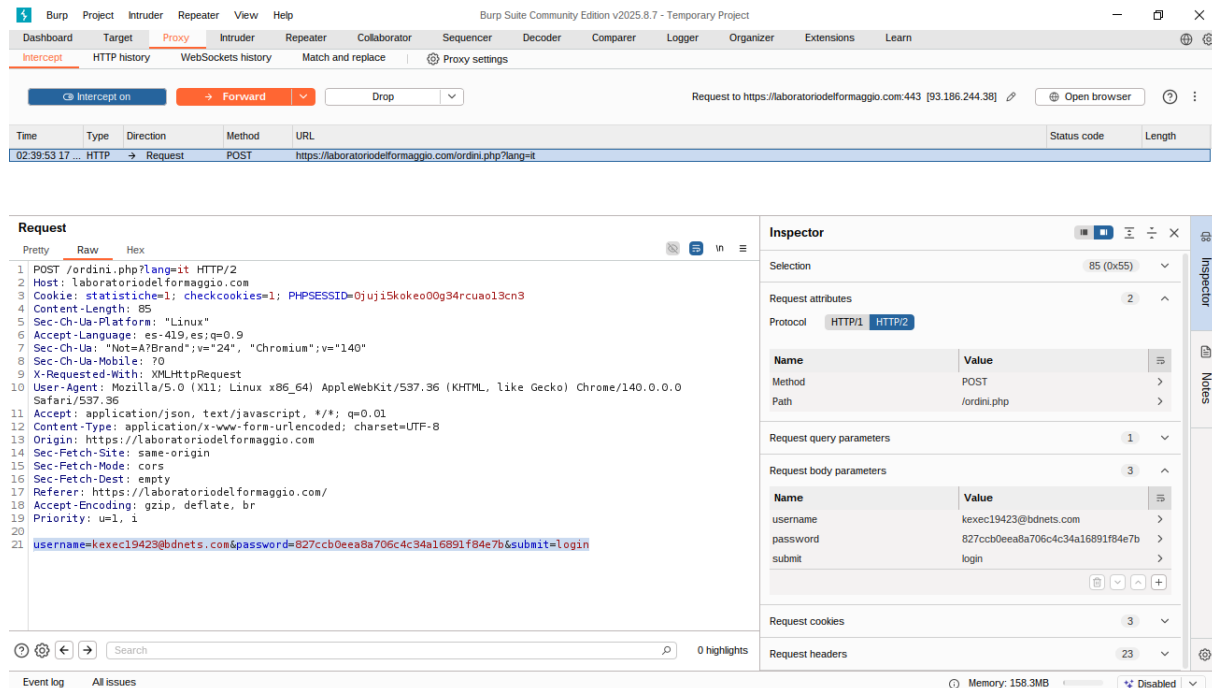


Figura 8: Captura tráfico BurpSuite

## 2.5. Realiza el intento de login por medio del hash

Para esta sección es necesario destacar que en BurpSuite en el apartado de Repeater, se tiene una herramienta que nos permite modificar y enviar un mensaje HTTP o WebSocket en repetidas ocasiones, es decir en este caso nos permitiría modificar los parámetros vistos antes en la Figura 8, ya sea **username** o **password** para así ver cual sería la respuesta de la pagina ante esto.

Esto nos permite hacernos una idea de como el parámetro de contraseña aunque este con un tipo de hash puede ser modificado o involucrado en intentos de acceso en herramientas tipo BurpSuite, en los siguientes pasos se demostrará esto mas en detalle, por ahora primero es necesario mostrar cual es la respuesta al paquete capturado con la herramienta Repeater.

En la Figura 9 podemos observar como ingresando correctamente las credenciales la pagina devuelve un redireccionamiento a la sección de '**MyAccount**'



## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

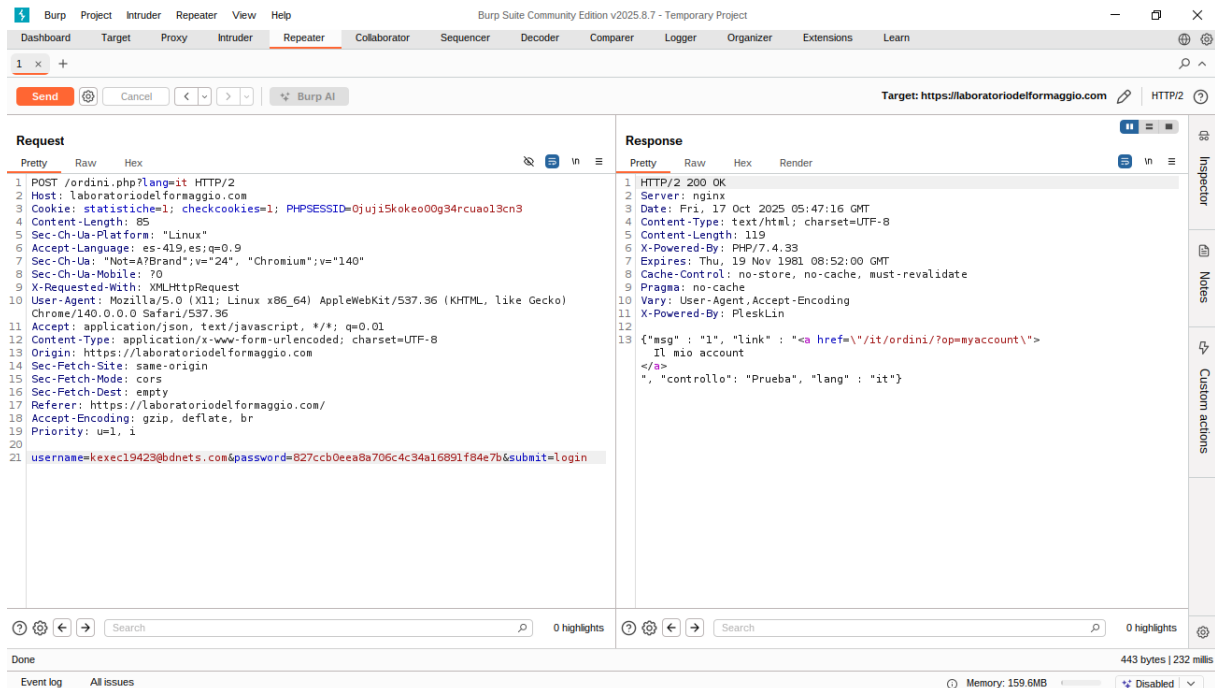


Figura 9: Repeater BurpSuite Exito Login

Se verifica la contraparte del uso de la herramienta modificando un poco el hash para así ver cual es la respuesta de la pagina, el hash de la password '12345' era '827ccb0eea8a706c4c34a16891f' por lo que se decidió modificar a '827ccb0eea8a706c4c34a16891f84e' eliminando los dos últimos caracteres , al realizar esto como se puede ver en la Figura 10 se recibe la siguiente respuesta de la página 'Username o password errati' lo que significa que modificando el hash, o directamente la contraseña la pagina bloquea el acceso. Esto no define la relación a tratar en el siguiente punto que es el intento de acceso al login por medio del hash.

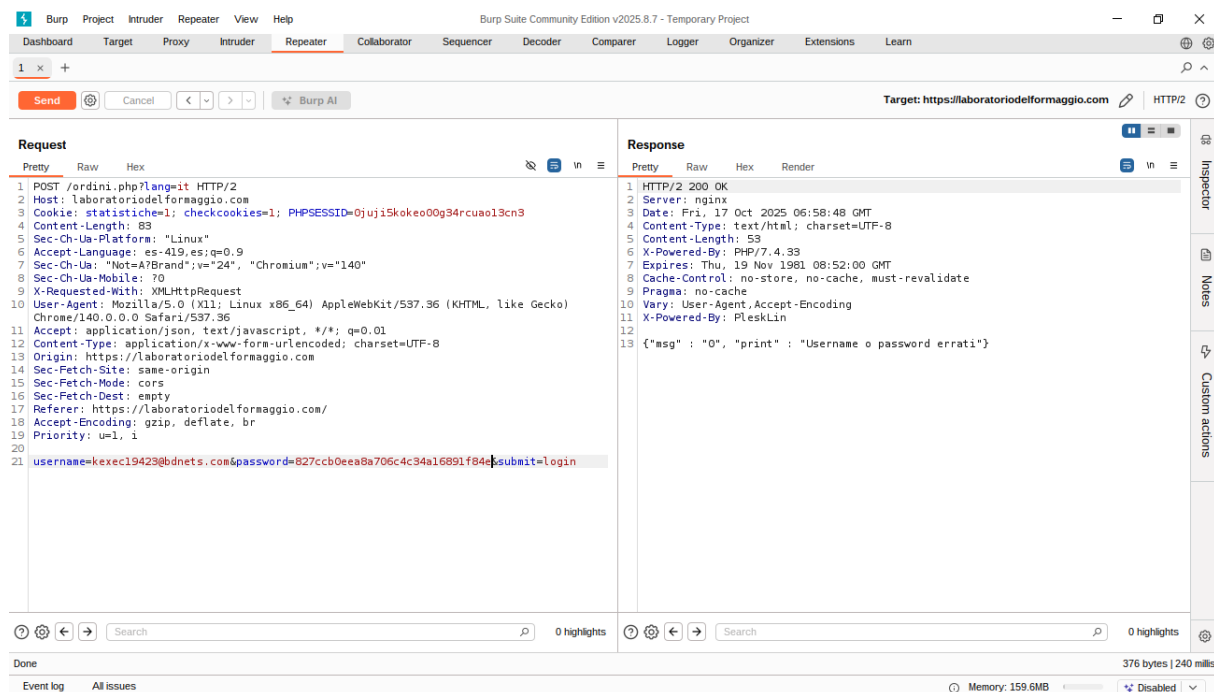


Figura 10: Repeater BurpSuite Fallo Login

Otra validación extra que se hizo, es que se pasó el hash directamente por la pagina, y se intercepta en BurpSuite para así ver la respuesta del paquete.

Durante esta prueba de pass-the-hash se introdujo en el campo de password en la pagina el MD5 de '12345' (valor observado: 827ccb0eea8a706c4c34a16891f84e7b). Al interceptar la petición con BurpSuite, el valor que realmente se percibe en el paquete como podemos notarlo en la Figura 11 es (cab47add236cbd300fd86e668e51e08f) y aparte el servidor denegó el acceso.

Esto indica que no basta con enviar el MD5 calculado localmente ya que, el cliente está aplicando alguna transformación adicional sobre lo que se introduce o puede ser que el que el servidor vuelve a procesar el valor recibido antes de comparar.

En la práctica esto significa que colocar el hash directamente en la pagina mitiga un ataque simple de pass-the-hash, sin embargo como se demostró anteriormente si se mantiene el hash en el paquete de Burpsuit, si se puede acceder, por lo que **NO** quita que el sitio si se mantenga aun vulnerable a un ataque pass-the-hash.

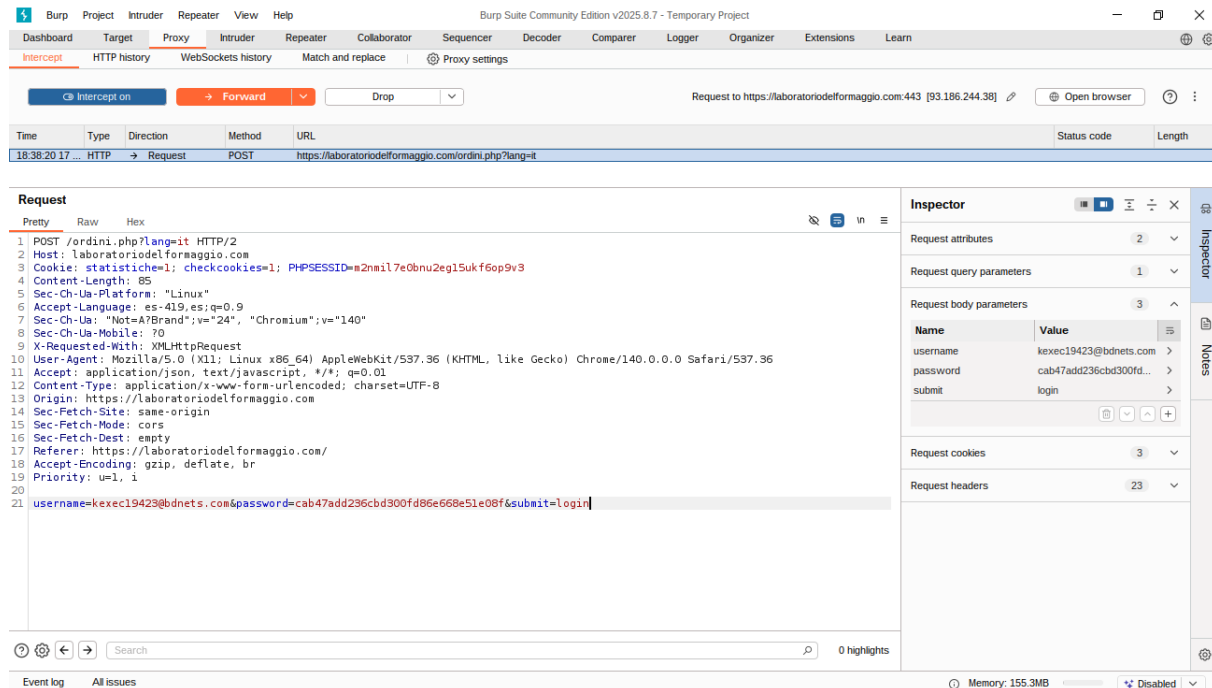


Figura 11: Repeater BurpSuite Fallo Login

## 2.6. Identifica las políticas de privacidad o seguridad

En el apartado de <https://laboratoriodelformaggio.com/it/p/privacy/86/> se especifica todos los aspectos de privacidad y gestión del sitio.

La política de privacidad del sitio Laboratorio del Formaggio declara cumplimiento del Reglamento (UE) 2016/679 (GDPR) y describe las categorías de datos que recogen (datos voluntariamente facilitados por el usuario, datos de navegación, cookies) y fines del tratamiento. Además indica que los datos se conservan en servidores “protegidos” y que para la transmisión se utiliza (SSL) (un protocolo de seguridad que cifra las comunicaciones entre un navegador y un servidor web para proteger los datos transmitidos en Internet). No se especifican, sin embargo, los algoritmos o mecanismos concretos empleados para el almacenamiento de contraseña. Pero como vimos anteriormente se hace uso de la biblioteca ‘CryptoJS.MD5’ lo que implica el uso de un algoritmo obsoleto.

## 2.7. Comente 4 conclusiones sobre la seguridad del sitio escogido

1. **Uso de MD5 en cliente inseguro.** Aunque el cliente calcula un MD5 (CryptoJS.MD5) y lo envía como campo `pass`, MD5 es criptográficamente débil y no debe usarse para protección de contraseñas; además, el hecho de enviar la contraseña con su hash visible aumenta el riesgo de exposición si TLS no se aplica correctamente. La política de privacidad menciona cifrado en tránsito (SSL/TLS), lo que mitiga parcialmente el riesgo

de intercepción pero no garantiza totalmente el que no hayan ataques.

2. **Pass-The-Hash.** Al realizar el test en BurpSuite notamos que se mostró que al introducir localmente el MD5 conocido se produjo un valor distinto el acceso fue denegado; eso indica que el cliente o el servidor aplican una transformación adicional antes de la comprobación final. Esto reduce la viabilidad de un ataque básico *pass-the-hash*, pero luego en Burpsuit a enviar el paquete con el Hash en la password si permite el acceso por lo que el fallo se mantiene y la validacion solo pertenece a que aplican nuevamente hash al campo, pero esto no quita toda la posibilidad de ataque *pass-the-hash*.
3. **Falta de transparencia sobre almacenamiento de contraseñas en la política de privacidad.** La política consultada declara cumplimiento normativo y uso de SSL para la transmisión, pero **no especifica el método de almacenamiento** (algoritmo, uso de *salt*, parámetros). Desde el punto de vista de seguridad y cumplimiento, esa ausencia es significativa: sin esa información no es posible verificar si las contraseñas están protegidas correctamente.
4. **Conclusiones finales y generales.** Debe eliminarse la dependencia de MD5 en el cliente como mecanismo de seguridad: el servidor debe ser la única fuente de verdad y debe aplicar hashing con *salt* y algoritmos adaptativos (bcrypt o Argon2) para almacenar contraseñas. Además, asegurar TLS moderno en todo el sitio (HSTS), evitar aceptar un hash del cliente como credencial directa, implementar limitación de intentos/rate-limiting y ofrecer MFA. Finalmente, la política de privacidad debería actualizarse para declarar explícitamente las prácticas de almacenamiento y protección de contraseñas.

## 2.8. Comparación: MD5 vs otros algoritmos/funciones criptográficas

Como en la ultima conclusión se menciono algoritmos adaptativos que han sido mencionados en clases, a continuación se presenta una tabla comparativa que nos otorga un mayor contexto de como funcionan estos algoritmos y porque para este tipo de ataque *pass-the-hash* son vitales que existan en la seguridad para evitar lo visto y desarrollado que demuestra multiples vulnerabilidades.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Algoritmo	Tipo / Propósito	Velocidad	Resistencia a fuerza bruta / GPU	Sal	Uso recomendado
MD5	Hash criptográfico (obsoleto)	Muy rápido	Muy baja (fácil de romper con GPU / rainbow tables)	No incorporada	<b>No recomendado</b>
SHA-256	Hash criptográfico (SHA-2)	Rápido	Baja-moderada (mejor que MD5, pero aún rápido — vulnerable a GPU)	No incorporada	<b>No recomendado por sí solo; usar en KDF con sal e iteraciones</b>
PBKDF2	KDF (derivación de claves)	Configurable (iteraciones)	Moderada (según iteraciones)	Sí (requerido)	<b>Recomendado si se usan parámetros adecuados</b>
bcrypt	Función adaptativa para contraseñas	Intencionalmente más lento (cost configurable)	Buena resistencia (diseñado contra GPU)	Sí (incorporado)	<b>Recomendado</b>
Argon2	KDF memory-hard (Password Hashing winner)	Configurable (tiempo + memoria)	Muy buena (resistente a GPU/ASIC)	Sí (incorporado)	<b>Recomendado</b>

Tabla 1: Comparativa breve entre MD5 y algoritmos/funciones para protección de contraseñas.

### 2.9. Comentarios

Repositorio GitHub:  
<https://github.com/hikenN17/Laboratorio3Criptografia>