



---

# WIRESHARK

---

SAD



2ºASIR

I.E.S. ANTONIO MACHADO  
ANA OROZCO ASENSIO

## Contenido

Introducción.....	2
Monitoreo.....	2
MySQL.....	2
HTTP.....	3
Conclusión.....	5

# Introducción.

En esta práctica vamos a usar Wireshark para monitorear la red usando Wordpress con MySQL y Apache con PHP.

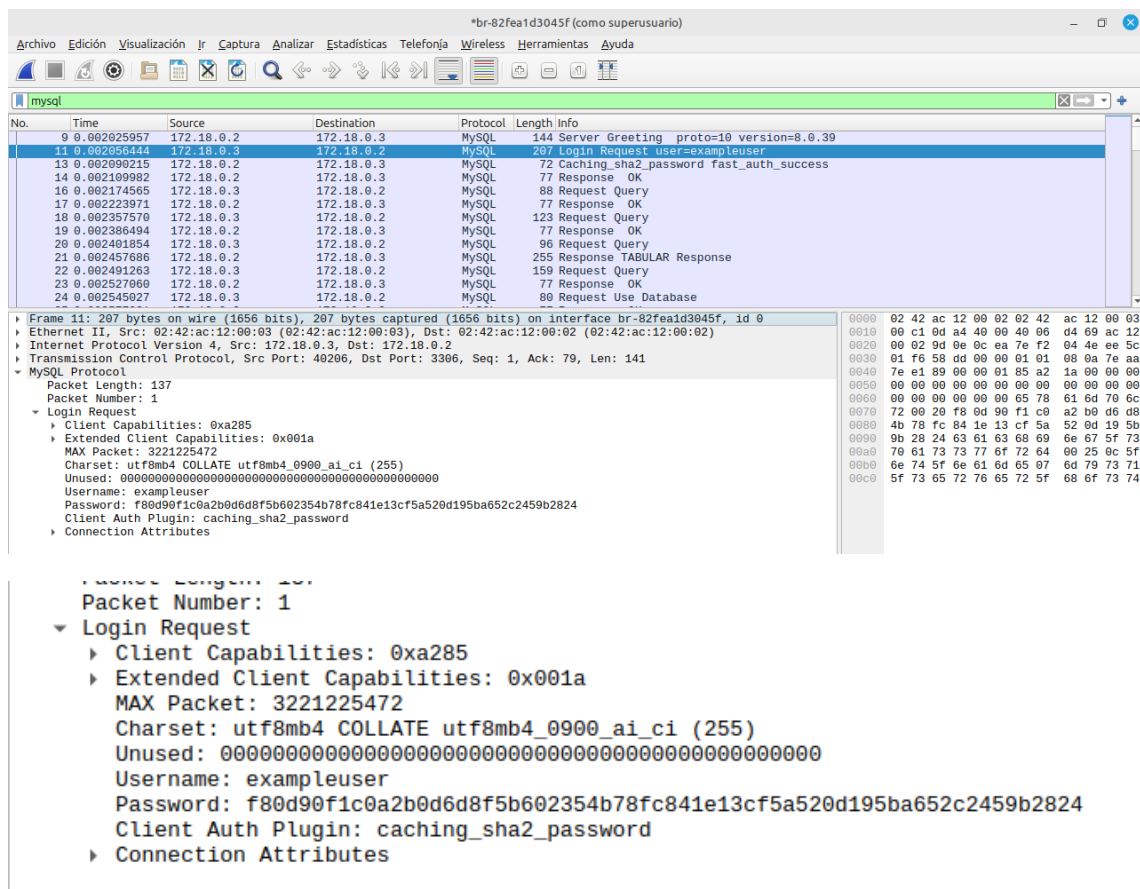
Comprobaremos lo que podemos ver al buscar en los paquetes que monitoreamos y buscaremos la contraseña y usuario de Wordpress.

# Monitoreo.

Voy a dividir en 2 este monitoreo.

# MySQL

En esta captura podemos ver el trafico de la red filtrando con el protocolo MySQL, al entrar en los login request podemos ver que la contraseña está cifrada, por lo que de estos paquetes podemos sacar poca información.



## HTTP.

Con el protocolo HTTP vemos varios tipos de paquetes como GET que solicita los recursos al servidor o POST que envía datos al servidor (se suele usar para los formularios)

The image shows a Wireshark network traffic capture. The top pane displays a list of captured packets, with the selected packet (No. 4) being an HTTP POST request from 172.18.0.1 to 172.18.0.3. The bottom pane shows the details of this packet, including the Ethernet II header, Internet Protocol Version 4 header, and the Hypertext Transfer Protocol section. The POST request is for the URL /wp-admin/install.php?step=1 and has a content type of application/x-www-form-urlencoded. The response (No. 5) is an HTTP 200 OK from 172.18.0.3 to 172.18.0.1, with a content type of text/html. The packet list shows a total of 4064 packets, with 20 (0.5%) displayed.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.008875095	172.18.0.1	172.18.0.3	HTTP	650	POST /wp-admin/install.php?step=1 HTTP/1.1 (application/x-www-form-urlencoded)
5	0.009075095	172.18.0.3	172.18.0.1	HTTP	3446	HTTP/1.1 200 OK (text/html)
1790	1.799424886	172.18.0.1	172.18.0.3	HTTP	407	GET /wp-includes/js/zxcvbn-async.min.js?ver=1.0 HTTP/1.1
1792	1.821935810	172.18.0.1	172.18.0.3	HTTP	664	HTTP/1.1 200 OK (text/javascript)
1794	1.822175217	172.18.0.1	172.18.0.3	HTTP	422	GET /wp-includes/js/dist/hooks.min.js?ver=2810c76e705dd1a53b18 HTTP/1.1
1795	1.822308279	172.18.0.1	172.18.0.3	HTTP	421	GET /wp-includes/js/dist/l10n.min.js?ver=5e580eb46a90c2b997e6 HTTP/1.1
1799	1.822486417	172.18.0.1	172.18.0.3	HTTP	1951	HTTP/1.1 200 OK (text/javascript)
1806	1.822507575	172.18.0.1	172.18.0.3	HTTP	417	GET /wp-admin/js/password-strength-meter.min.js?ver=6.6.2 HTTP/1.1
1803	1.822730296	172.18.0.1	172.18.0.3	HTTP	1029	HTTP/1.1 200 OK (text/javascript)
1804	1.82298943	172.18.0.1	172.18.0.3	HTTP	408	GET /wp-includes/js/underscore.min.js?ver=1.13.4 HTTP/1.1
1805	1.823059243	172.18.0.1	172.18.0.3	HTTP	4078	HTTP/1.1 200 OK (text/javascript)
1806	1.823102088	172.18.0.1	172.18.0.3	HTTP	404	GET /wp-includes/js/wp-util.min.js?ver=6.6.2 HTTP/1.1
1811	1.823277007	172.18.0.1	172.18.0.3	HTTP	1164	HTTP/1.1 200 OK (text/javascript)
1812	1.823458537	172.18.0.1	172.18.0.3	HTTP	406	GET /wp-admin/js/user-profile.min.js?ver=6.6.2 HTTP/1.1
1813	1.823590878	172.18.0.1	172.18.0.3	HTTP	481	HTTP/1.1 200 OK (text/javascript)
1817	1.823685961	172.18.0.1	172.18.0.3	HTTP	2891	HTTP/1.1 200 OK (text/javascript)
1818	1.824131800	172.18.0.1	172.18.0.3	HTTP	393	GET /wp-includes/js/zxcvbn.min.js HTTP/1.1
1820	1.854213802	172.18.0.1	172.18.0.3	HTTP	27314	HTTP/1.1 200 OK (text/javascript)
1913	1.905440804	172.18.0.1	172.18.0.3	HTTP	827	POST /wp-admin/install.php?step=2 HTTP/1.1 (application/x-www-form-urlencoded)
1929	263.426158024	172.18.0.1	172.18.0.3	HTTP	2360	HTTP/1.1 200 OK (text/html)
4058	266.125180578	172.18.0.1	172.18.0.3	HTTP		

Frame 4: 650 bytes on wire (5200 bits), 650 bytes captured (5200 bits) on interface  
Ethernet II, Src: 02:42:7d:14:69:27 (02:42:7d:14:69:27), Dst: 02:42:ac:12:00:03  
Internet Protocol Version 4, Src: 172.18.0.1, Dst: 172.18.0.3  
Transmission Control Protocol, Src Port: 53648, Dst Port: 80, Seq: 1, Ack: 1, Len: 650  
Hypertext Transfer Protocol  
HTML Form URL Encoded: application/x-www-form-urlencoded

Paquetes: 4064 - Mostrado: 20 (0.5%)

En esta captura vemos que solo tenemos códigos del estado del HTTP como 200 OK que significa que la solicitud se ha procesado correctamente, también tenemos el 500 Internal Server Error que dice que ha ocurrido un error en el servidor y el famoso 404 Not Found, que lo que nos dice es que no se encuentra el recurso que hemos solicitado. Además de estos hay más errores pero esto son los más comunes.

Seguimos buscando podemos encontrar información como el navegador que se utiliza en el tipo POST, el sistema operativo e incluso el idioma.

En los paquetes GET suele estar la información de la versión de HTTP utilizada, la url del recurso, la versión del PHP, el tipo de contenido, etc.

The screenshot shows a Wireshark capture of network traffic. The packet list on the left highlights a POST request at 1929. The packet details pane on the right shows the request body with form items: weblog\_title, user\_name, admin\_password, admin\_password2, pw\_weak, admin\_email, Submit, and language.

```
1929 263 426158624 172.18.0.1 172.18.0.3 HTTP 827 POST /wp-admin/install.php?step=2 HTTP/1.1 (application/x-www-form-urlencoded)
4058 266 125180578 172.18.0.3 172.18.0.1 HTTP 2300 HTTP/1.1 200 OK (text/html)
```

Frame 1929: 827 bytes on wire (6616 bits), 827 bytes captured (6616 bits) on interface br-82fe1d3045f, id 0  
Ethernet II, Src: 02:42:7d:14:69:27 (02:42:7d:14:69:27), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)  
Internet Protocol Version 4, Src: 172.18.0.1, Dst: 172.18.0.3  
Transmission Control Protocol, Src Port: 43072, Dst Port: 80, Seq: 1, Ack: 1, Len: 761  
Hypertext Transfer Protocol  
POST /wp-admin/install.php?step=2 HTTP/1.1\r\n  
Host: 172.18.0.3\r\n  
User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:131.0) Gecko/20100101 Firefox/131.0\r\n  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8\r\n  
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n  
Accept-Encoding: gzip, deflate\r\n  
Content-Type: application/x-www-form-urlencoded\r\n  
Content-Length: 183\r\n  
Origin: http://172.18.0.3\r\n  
Connection: keep-alive\r\n  
Referer: http://172.18.0.3/wp-admin/install.php?step=1\r\n  
Upgrade-Insecure-Requests: 1\r\n  
Priority: u=0, i=1\r\n  
Full request URI: http://172.18.0.3/wp-admin/install.php?step=2  
[HTTP request 1/1]  
[Response in frame: 4058]  
File Data: 183 bytes  
HTML Form URL Encoded: application/x-www-form-urlencoded  
Form item: "weblog\_title" = "anaorozco"  
Form item: "user\_name" = "anaorozco"  
Form item: "admin\_password" = "Admin1."  
Form item: "admin\_password2" = "Admin1."  
Form item: "pw\_weak" = "on"  
Form item: "admin\_email" = "anaorozco99@iesamachado.org"  
Form item: "Submit" = "Instalar WordPress"  
Form item: "language" = "es\_ES"

Además, podemos ver el usuario y la contraseña usados en el formulario de creación, por lo que no hay cifrado y por lo tanto no es seguro.

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "weblog_title" = "anaorozco"
    Key: weblog_title
    Value: anaorozco
  Form item: "user_name" = "anaorozco"
    Key: user_name
    Value: anaorozco
  Form item: "admin_password" = "Admin1."
    Key: admin_password
    Value: Admin1.
  Form item: "admin_password2" = "Admin1."
    Key: admin_password2
    Value: Admin1.
  Form item: "pw_weak" = "on"
  Form item: "admin_email" = "anaorozco99@iesamachado.org"
    Key: admin_email
    Value: anaorozco99@iesamachado.org
  Form item: "Submit" = "Instalar WordPress"
  Form item: "language" = "es_ES"
```

## Conclusión

Después de esta práctica me he dado cuenta de por qué no es seguro el protocolo HTTP y deberíamos usar el HTTPS (aunque no todo el contenido de HTTPS está cifrado).

La diferencia que veríamos en wireshark sería que al hacer el monitoreo veríamos los paquetes, pero estos tendrían la información encriptada y autenticada