

1.- Análisis de la aplicación

Enviaremos una petición a los servidores de resolución de nombres de *Cloudflare* a través de comando **KDIG**, concretamente pediremos la resoluciones de nombre de www.instagram.com al servidor 1.1.1.1

Cloudflare dispone de 3 servidores primarios diferentes que podemos configurar como cliente y su contraparte el servidor secundario:

- **1.1.1.1/1.0.0.1** → Unicamente actúa como servidor **DNSoverTLS**
- **1.1.1.2/1.0.0.2** → Realiza un bloqueo de malware y de sitios sospechosos
- **1.1.1.3/1.0.0.3** → Bloqueo parental para contenido adulto entre otros..

Para esta practica usaremos el servidor 1.1.1.1 para realizar la petición de resolución, **KDIG** enviara las peticiones DNS:

```
(network-rk@network-rk)-[~]
$ kdig -d @1.1.1.1 +tls-ca +tls-host=cloudflare-dns.com www.instagram.com
;; DEBUG: Querying for owner(www.instagram.com.), class(1), type(1), server(1.1.1.1), port(853), protocol(TCP)
;; DEBUG: TLS, imported 127 system certificates
;; DEBUG: TLS, received certificate hierarchy:
;; DEBUG: #1, C=US,ST=California,L=San Francisco,O=Cloudflare\, Inc.,CN=cloudflare-dns.com
;; DEBUG: SHA-256 PIN: GP8Knf7qBae+aIfythyTMBYnL+yowaWVeD6MoLHkVRg=
;; DEBUG: #2, C=US,O=DigiCert Inc,CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1
;; DEBUG: SHA-256 PIN: e0IRz5Tio3GA1Xs4fUVWmH1xHdiH2dMbVtCBSk0IdqM=
;; DEBUG: TLS, skipping certificate PIN check
;; DEBUG: TLS, The certificate is trusted.
;; TLS session (TLS1.3)-(ECDHE-X25519)-(ECDSA-SECP256R1-SHA256)-(AES-256-GCM)
;; -->HEADER<-- opcode: QUERY; status: NOERROR; id: 34795
;; Flags: qr rd ra; QUERY: 1; ANSWER: 3; AUTHORITY: 0; ADDITIONAL: 1

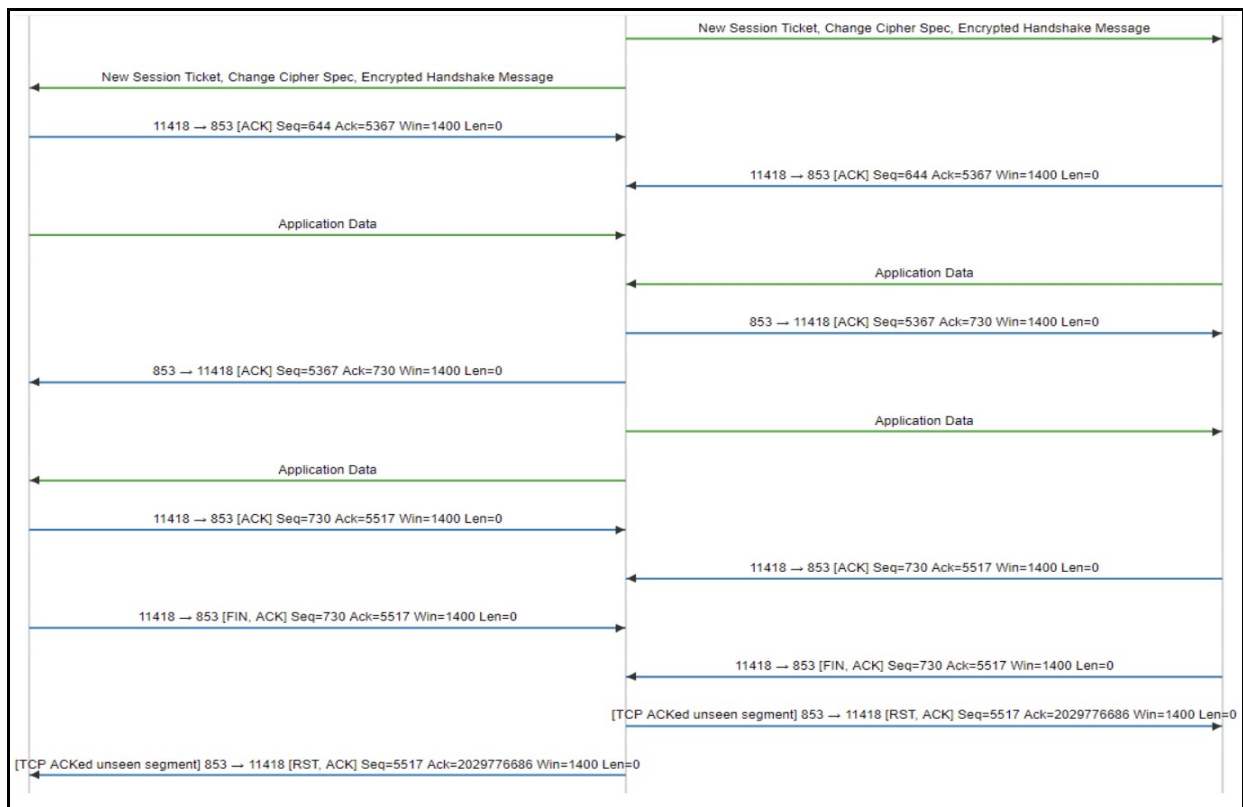
;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1232 B; ext-rcode: NOERROR
;; PADDING: 345 B

;; QUESTION SECTION:
;; www.instagram.com.          IN      A

;; ANSWER SECTION:
www.instagram.com.          1969    IN      CNAME   geo-p42.instagram.com.
geo-p42.instagram.com.     1969    IN      CNAME   z-p42-instagram.c10r.instagram.com.
z-p42-instagram.c10r.instagram.com. 48      IN      A        157.240.5.174

;; Received 468 B
;; Time 2023-03-31 10:52:05 CEST
;; From 1.1.1.1@853(TCP) in 97.6 ms
```

Server DNS: 1.1.1.1
Puerto Server: 853
Protocolo: TCP



1.1- Syscalls

Usaremos el comando STRACE para vigilar las llamadas al sistema de red que utiliza para realizar la transacción DoT:

```
(network-rk@ network-rk)-[~]
$ strace -c -e trace=network kdig -d @1.1.1.1 +tls-ca +tls-host=cloudflare-dns.com www.instagram.com
;; DEBUG: Querying for owner(www.instagram.com.), class(1), type(1), server(1.1.1.1), port(853), protocol(TCP)
;; DEBUG: TLS, imported 127 system certificates
;; DEBUG: TLS, received certificate hierarchy:
;; DEBUG: #1, C=US, ST=California, L=San Francisco, O=Cloudflare, Inc., CN=cloudflare-dns.com
;; DEBUG: SHA-256 PIN: GP8Knf7qBae+aIfythyMbYnL+yowaWVeD6MoLHkVRg=
;; DEBUG: #2, C=US, O=DigiCert Inc, CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1
;; DEBUG: SHA-256 PIN: e0IRz5Tio3GA1Xs4fUVWmH1xHDiH2dMbVtCBSk0IdqM=
;; DEBUG: TLS, skipping certificate PIN check
;; DEBUG: TLS, The certificate is trusted
;; TLS session (TLS1.3)-(ECDHE-X25519)-(ECDSA-SECP256R1-SHA256)-(AES-256-GCM)
-->HEADER<- opcode: QUERY; status: NOERROR; id: 15277
;; Flags: qr rd ra; QUERY: 1; ANSWER: 3; AUTHORITY: 0; ADDITIONAL: 1

;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1232 B; ext-rcode: NOERROR
;; PADDING: 345 B

;; QUESTION SECTION:
;; www.instagram.com.      IN      A

;; ANSWER SECTION:
www.instagram.com.      3554    IN      CNAME   geo-p42.instagram.com.
geo-p42.instagram.com. 3554    IN      CNAME   z-p42-instagram.c10r.instagram.com.
z-p42-instagram.c10r.instagram.com. 14     IN      A        157.240.5.174

;; Received 468 B
;; Time 2023-03-31 11:16:42 CEST
;; From 1.1.1.1@853(TCP) in 90.2 ms
% time    seconds    usecs/call   calls    errors syscall
-----
42.00    0.000341      68         5         0 sendmsg
35.96    0.000292      22        13         3 recvfrom
11.70    0.000095      95         1         1 connect
 8.37    0.000068      68         1         0 bind
 1.97    0.000016      16         1         0 getsockopt
 0.00    0.000000       0         1         0 socket
-----
100.00    0.000812      36        22         4 total
```

Se inicia una comunicación TLS 1.3 en la que se involucran diferentes algoritmos asimétricos y simétricos:

- **ECDHE:** Es un algoritmo asimétrico, Diffie Hellman de curva elíptica, a a partir del cual se genera un secreto compartido del que deriva una llave con la que cifrar la comunicación.
- **ECDSA:** Genera una firma única para verificar la autenticidad de los mensajes, el cual emplea operaciones sobre puntos de curvas elípticas
- **AES-256-GCM:** un algoritmo simétrico de cifrado por bloques autenticado, tiene 2 componentes AES-CTR para el cifrado y GMAC para el autenticado.

Haciendo un pequeño análisis de las llamadas al sistema que utiliza obtenemos lo siguiente:

- **sendmsg:** Las llamadas al sistema **send()**, **sendto()** y **sendmsg()** se utilizan para transmitir un mensaje a otro socket.
- **Recvfrom:** Las llamadas **recvfrom()** y **recvmsg()** se usan para recibir mensajes de un socket y se pueden usar para recibir datos en un socket, esté o no orientado a la conexión.
- **connect:** La llamada al sistema **connect()** conecta el socket al que hace referencia el descriptor de archivo *sockfd* a la dirección especificada por *addr*.
- **Bind:** asigna la dirección especificada por *addr* al socket al que hace referencia el descriptor de archivo *sockfd*. *AddrLen* especifica el tamaño, en bytes, de la estructura de dirección a la que apunta *addr*
- **getsockopt:** y **setsockopt()** manipulan las opciones para el socket al que hace referencia el descriptor de archivo *sockfd*. Las opciones pueden existir en múltiples niveles de protocolo; siempre están presentes en el nivel más alto del zócalo.
- **Socket:** crea un punto final para la comunicación y devuelve un descriptor.

Observemos mas de cerca las llamadas las sistema con los parámetros que maneja durante la transacción:

```

(ang)-(network-rk@ network-rk)-[~]
$ strace -e trace=network -f kdig -d @1.1.1.1 +tls-ca +tls-host=cloudflare-dns.com www.instagram.com
;; DEBUG: Querying for owner(www.instagram.com.), class(1), type(1), server(1.1.1.1), port(853), protocol(TCP)
;; DEBUG: TLS, imported 127 system certificates
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 3
bind(3, {sa_family=AF_INET, sin_port=htons(0), sin_addr=inet_addr("0.0.0.0")}, 16) = 0
connect(3, {sa_family=AF_INET, sin_port=htons(853), sin_addr=inet_addr("1.1.1.1")}, 16) = -1 EINPROGRESS (Operaci
ón en curso)
getsockopt(3, SOL_SOCKET, SO_ERROR, [0], [4]) = 0
sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov={iov_base="\26\3\1\1\225\1\0\1\221\3\3\233q\335\262\353\33>\32
\343\7:\243\222K\33\32\256\236)[\"..., iov_len=410}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 410
recvfrom(3, 0x55b16ee0dec0, 5, 0, NULL, NULL) = -1 EAGAIN (Recurso no disponible temporalmente)
recvfrom(3, "\26\3\3\0z", 5, 0, NULL, NULL) = 5
recvfrom(3, "\2\0\0v\3\3\246\213\30\17Xd\336\n\16=X\33\34\3152\200\23\341\223aCB|\20\0:...., 122, 0, NULL, NULL)
= 122
sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov={iov_base="\24\3\3\0\1\1", iov_len=6}], msg_iovlen=1, msg_cont
rollen=0, msg_flags=0}, 0) = 6
recvfrom(3, "\24\3\3\0\1", 5, 0, NULL, NULL) = 5
recvfrom(3, "\1", 1, 0, NULL, NULL) = 1
recvfrom(3, "\27\3\3\n\303", 5, 0, NULL, NULL) = 5
recvfrom(3, "\23\316\370\235\330vNe\361Z\273vq\230V*\v\271\3055c\336\221\212M\255k\3048o\327@\"..., 2755, 0, NULL,
NULL) = 2755
;; DEBUG: TLS, received certificate hierarchy:
;; DEBUG: #1, C=US, ST=California, L=San Francisco, O=Cloudflare, Inc., CN=cloudflare-dns.com
;; DEBUG: SHA-256 PIN: GP8Knf7qBae+aIfythyTmBYnL+yowawVeD6MoLHkVRg=
;; DEBUG: #2, C=US, O=DigiCert Inc, CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1
;; DEBUG: SHA-256 PIN: e0IRz5Tio3GA1Xs4fUVWmH1xHdIH2dMbVtCBSk0IdqM=
;; DEBUG: TLS, skipping certificate PIN check
;; DEBUG: TLS, The certificate is trusted.
sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov={iov_base="\27\3\3\0E\237;f\227\217k+\323\350g\30\213F\234P\23
6\236\263\212\325\347Qs\300Y+\"1\"..., iov_len=74}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 74
sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov={iov_base="\27\3\3\0\223\254*\32\232le\17\353\201B\177\#j\245
_\212>\212h\1\342r?@\"..., iov_len=152}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 152
recvfrom(3, 0x55b16ee1baf3, 5, 0, NULL, NULL) = -1 EAGAIN (Recurso no disponible temporalmente)
recvfrom(3, "\27\3\3\1\275", 5, 0, NULL, NULL) = 5
recvfrom(3, "Q\253Mq\32\37\216\177\30u\353U\365\355\230k\361\323\272`ksC\265\22re\332\337A\"..., 445, 0, NULL, NU
LL) = 445
recvfrom(3, "\27\3\3\1\347", 5, 0, NULL, NULL) = 5
recvfrom(3, "IB\201w#e\302\240TT\256\356\0\366\305*\303\3075cb\2746>\377e\0314\332\350\344\371\"..., 487, 0, NULL,
NULL) = 487
;; TLS session (TLS1.3)-(ECDHE-X25519)-(ECDSA-SECP256R1-SHA256)-(AES-256-GCM)
;; ->>HEADER<- opcode: QUERY; status: NOERROR; id: 34349
;; Flags: qr rd ra; QUERY: 1; ANSWER: 3; AUTHORITY: 0; ADDITIONAL: 1

;; EDNS PSEUDOSECTION:
;; Version: 0; flags: ; UDP size: 1232 B; ext-rcode: NOERROR
;; PADDING: 345 B

;; QUESTION SECTION:
;; www.instagram.com.          IN      A

;; ANSWER SECTION:
www.instagram.com.          3538   IN      CNAME   geo-p42.instagram.com.
geo-p42.instagram.com.     3538   IN      CNAME   z-p42-instagram.c10r.instagram.com.
z-p42-instagram.c10r.instagram.com. 51     IN      A       157.240.5.174

;; Received 468 B
;; Time 2023-03-31 11:23:57 CEST
;; From 1.1.1.1@853(TCP) in 102.1 ms
sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov={iov_base="\27\3\3\0\23\fa\371\326tb_0\267k\26\305v\244\337\fp
\347\276\", iov_len=24}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 24
recvfrom(3, 0x55b16ee1baf3, 5, 0, NULL, NULL) = -1 EAGAIN (Recurso no disponible temporalmente)

```

Realicemos un pequeño análisis de las **syscall** y sus parámetros:

- **socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 3**
 - Crea un socket con estos parámetros:
 - **AF_INET** → Protocolos de Internet Ipv4
 - **SOCK_STREAM** → Los protocolos de comunicación que lo implementan aseguran que los datos no se pierdan ni se dupliquen
- **bind(3, {sa_family=AF_INET, sin_port=htons(0), sin_addr=inet_addr("0.0.0.0")}, 16) = 0**
 - Vincula el socket a la IP local 0.0.0.0
- **connect(3, {sa_family=AF_INET, sin_port=htons(853), sin_addr=inet_addr("1.1.1.1")}, 16) = -1 EINPROGRESS**
 - Se inicia la conexión a la dirección 1.1.1.1 y puerto 853 del servidor DNS de cloudflare
- **getsockopt(3, SOL_SOCKET, SO_ERROR, [0], [4]) = 0**
 - En caso de éxito al bindear el socket nos devuelve 0
- **sendmsg(3, {msg_name=NULL, msg_namelen=0, msg_iov=[.....], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 410**
 - Cuando el socket está en estado conectado se procede a enviar los elementos de la matriz a la que apunta *msg_iov*

Realizamos un seguimiento de la comunicación DoT con Wireshark:

No.	Time	Source	Destination	Protocol	Length	Info
5	14.684839565	192.168.1.191	1.1.1.1	TCP	74	48955 → 853 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1730973158 TSecr=333333333
6	14.703986002	1.1.1.1	192.168.1.191	TCP	74	853 → 48955 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1452 SACK_PERM TSval=1730973158 TSecr=333333333
7	14.704029073	192.168.1.191	1.1.1.1	TCP	66	48955 → 853 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1730973158 TSecr=333333333
8	14.704804697	192.168.1.191	1.1.1.1	TLSv1.3	476	Client Hello
9	14.716944350	1.1.1.1	192.168.1.191	TCP	66	853 → 48955 [ACK] Seq=1 Ack=411 Win=65536 Len=0 TSval=3338371614 TSecr=333333333
10	14.718073606	1.1.1.1	192.168.1.191	TLSv1.3	2961	Server Hello, Change Cipher Spec, Application Data
11	14.718098580	192.168.1.191	1.1.1.1	TCP	66	48955 → 853 [ACK] Seq=411 Ack=2896 Win=63104 Len=0 TSval=1730973172 TSecr=333333333
12	14.719080542	192.168.1.191	1.1.1.1	TLSv1.3	72	Change Cipher Spec
13	14.772838198	1.1.1.1	192.168.1.191	TCP	66	853 → 48955 [ACK] Seq=2896 Ack=417 Win=65536 Len=0 TSval=3338371670 TSecr=333333333
14	14.772860555	192.168.1.191	1.1.1.1	TLSv1.3	292	Application Data, Application Data
15	14.783950975	1.1.1.1	192.168.1.191	TCP	66	853 → 48955 [ACK] Seq=2896 Ack=643 Win=65536 Len=0 TSval=3338371681 TSecr=333333333
16	14.784558745	1.1.1.1	192.168.1.191	TLSv1.3	1008	Application Data, Application Data
17	14.791182083	192.168.1.191	1.1.1.1	TLSv1.3	90	Application Data
18	14.791602570	192.168.1.191	1.1.1.1	TCP	66	48955 → 853 [FIN, ACK] Seq=667 Ack=3838 Win=64128 Len=0 TSval=1730973240 TSecr=333333333
19	14.802956558	1.1.1.1	192.168.1.191	TCP	66	853 → 48955 [FIN, ACK] Seq=3838 Ack=667 Win=65536 Len=0 TSval=3338371700 TSecr=333333333
20	14.802972573	192.168.1.191	1.1.1.1	TCP	66	48955 → 853 [ACK] Seq=668 Ack=3839 Win=64128 Len=0 TSval=1730973257 TSecr=333333333
21	14.803909669	1.1.1.1	192.168.1.191	TCP	66	853 → 48955 [ACK] Seq=3839 Ack=668 Win=65536 Len=0 TSval=3338371701 TSecr=333333333

Primero se establece una comunicación TCP desde 192.168.1.191 sobre el servidor DNS de cloudflare 1.1.1.1, se realiza mediante la comunicación por 3 pasos:

- SYN → seq = x
- SYN-ACK → ack = x + 1 syn = y
- ACK → ack = y + 1 seq = x + 1

A partir de aquí se establece la comunicación TLS 1.3 a través de un intercambio de paquetes entre el cliente y servidor:

8	14.704804697	192.168.1.191	1.1.1.1	TLSv1.3	476	Client Hello
10	14.718073606	1.1.1.1	192.168.1.191	TLSv1.3	2961	Server Hello, Change Cipher Spec, Application Data
12	14.719080542	192.168.1.191	1.1.1.1	TLSv1.3	72	Change Cipher Spec
14	14.772860555	192.168.1.191	1.1.1.1	TLSv1.3	292	Application Data, Application Data
16	14.784558745	1.1.1.1	192.168.1.191	TLSv1.3	1008	Application Data, Application Data
17	14.791182083	192.168.1.191	1.1.1.1	TLSv1.3	90	Application Data

- Primero con el Client Hello se envía al servidor una lista de algoritmos con los que puede trabajar:

```

- Cipher Suites (29 suites)
  Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
  Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
  Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
  Cipher Suite: TLS_AES_128_CCM_SHA256 (0x1304)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CCM (0xc0ad)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CCM (0xc0ac)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
  Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
  Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
  Cipher Suite: TLS_RSA_WITH_AES_256_CCM (0xc09d)
  Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
  Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
  Cipher Suite: TLS_RSA_WITH_AES_128_CCM (0xc09c)
  Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
  Cipher Suite: TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CCM (0xc09f)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CCM (0xc09e)
  Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)

```

- En este punto el servidor compara con su propia lista y negocian los algoritmos que usaran para el resto de la

transmisión: (usando **nmap** vemos también la lista de algoritmo que posee el servidor dependiendo de la versión de TLS)

```
(ang)-(network-rk@ network-rk)-[~]
$ nmap --script ssl-enum-ciphers -p 443 1.1.1.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-05 22:54 CEST
Nmap scan report for one.one.one.one (1.1.1.1)
Host is up (0.019s latency).

PORT      STATE SERVICE
443/tcp   open  https
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|     compressors:
|       NULL
|     cipher preference: server
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|     compressors:
|       NULL
|     cipher preference: server
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256-draft (ecdh_x25519) - A
|     compressors:
|       NULL
|     cipher preference: client
|   TLSv1.3:
|     ciphers:
|       TLS_AKE_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_AKE_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_AKE_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|     cipher preference: client
|_  least strength: A

Nmap done: 1 IP address (1 host up) scanned in 7.33 seconds
```

◦ En nuestro caso se usara **TLS_AKE_WITH_256_GCM_SHA384** que tiene el siguiente desglose:

- **TLS** → protocolo
- **AKE** → *authenticated key exchange*
- **256** → longitud en bits de la clave simétrica
- **GCM** → modo del cifrado simétrico
- **SHA384** → Hash

- Una vez acordado y compartido se intercambian los paquetes cifrados entre C/S:

8	14.704804697	192.168.1.191	1.1.1.1	TLSv1.3	476 Client Hello
10	14.718073606	1.1.1.1	192.168.1.191	TLSv1.3	2961 Server Hello, Change Cipher Spec, Application Data
12	14.719080542	192.168.1.191	1.1.1.1	TLSv1.3	72 Change Cipher Spec
14	14.772860555	192.168.1.191	1.1.1.1	TLSv1.3	292 Application Data, Application Data
16	14.784558745	1.1.1.1	192.168.1.191	TLSv1.3	1008 Application Data, Application Data
17	14.791182083	192.168.1.191	1.1.1.1	TLSv1.3	90 Application Data

```

.....;1.u.I...v.6..^..iR..."|.F..>...Rc.po
.Y.$..Mx.....9....I.:.....
+. ...0.... / .....5.... / .....9....3.....
.....
".....
.....dot.....#...
3.k.i...A.....s.O.M.J.....B.K@....V_P...u.,.C.Re....6b.}.`}.K.-.a..l....
....._"...b....w..[pz.M.j.H.d...+.....cloudflare-
dns.com.-.....@.....E!D}Z9T6.L0.../^..Ig....0.J....."5S....@..Zn*....+7#.....
[...i.....o.T....Q{..s.....|.. a;..l...._eK....ANK^/-..8
..">WX\HI. .D..BSy....i...l...0....w&;.,.0..A1.*..{.....c
~...
....nI .t9
r.MT+....U...X.....b..fm.y5.b.T...K.

```

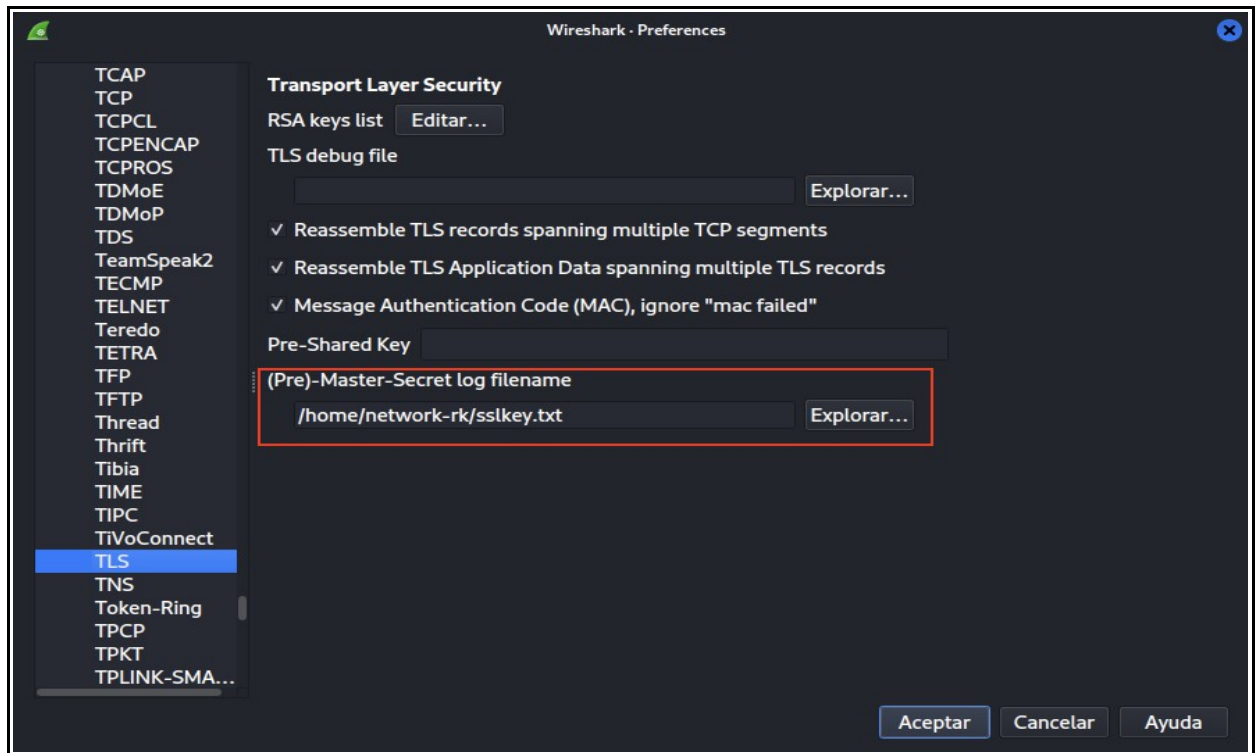
client pkt(s), 0 server pkt(s), 0 turn(s).

192.168.1.191:48955 → 1.1.1.1:853 (666 bytes) Mostrar datos como ASCII Secuencia 0

2.- Descifrando con las claves de sesión TLS

Para descifrar el tráfico TLS con Wireshark tendremos que apuntar la variable de entorno **SSLKEYLOGFILE** a un archivo de texto en el que se almacenaran los secretos de C/S y en *wireshark* en el apartado PREFERENCIAS>PROTOCOLOS>TLS seleccionar el mismo log para que pueda descifrar el tráfico:

```
(ang)-(network-rk@ network-rk)-[~]  
$ export SSLKEYLOGFILE=/home/network-rk/sslkey.txt
```



Una vez configurado y con Wireshark capturando el tráfico enviaremos peticiones al servidor **DNS** de cloudflare con el comando `kgid` para que nos resuelva el nombre de dominio www.instagram.com, filtraremos en wireshark por paquetes TLS:

No.	Time	Source	Destination	Protocol	Length	Info
16	1.046683371	192.168.1.191	one.one.one.one	TLSv1.3	476	Client Hello
18	1.060353778	one.one.one.one	192.168.1.191	TLSv1.3	2961	Server Hello, Change Cipher Spec, Encrypted Extensions, Certifica
20	1.060620644	192.168.1.191	one.one.one.one	TLSv1.3	72	Change Cipher Spec
32	1.114052413	192.168.1.191	one.one.one.one	DNS	292	Standard query 0xdaf8 A www.instagram.com OPT
34	1.127653761	one.one.one.one	192.168.1.191	DNS	1008	Standard query response 0xdaf8 A www.instagram.com CNAME geo-p42.
36	1.128268046	192.168.1.191	one.one.one.one	TLSv1.3	90	Alert (Level: Warning, Description: Close Notify)

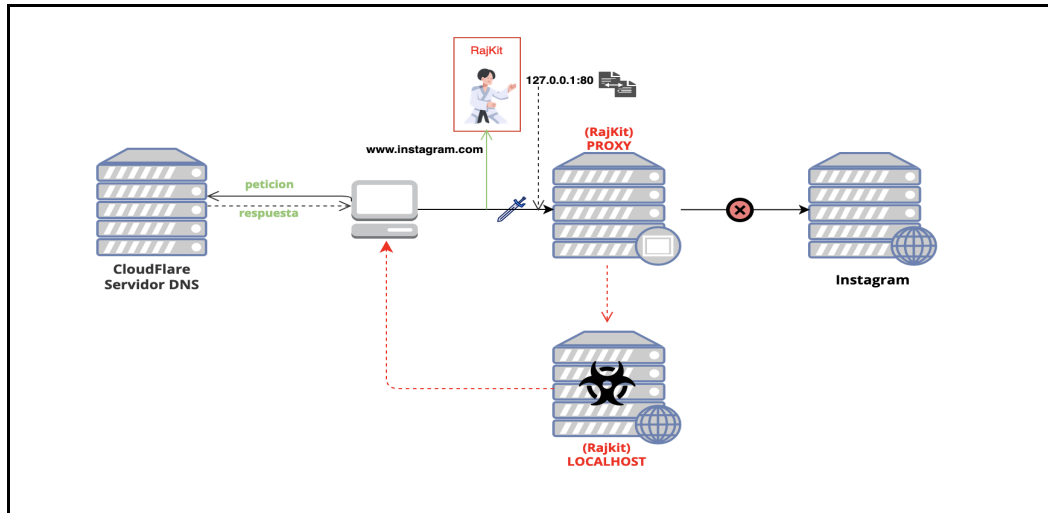
Comprobamos el archivo log en el que se almacenaron los **Master-Secret** para verificar:

```
(ang)-(network-rk@ network-rk)-[~]  
$ cat sslkey.txt  
CLIENT_HANDSHAKE_TRAFFIC_SECRET 212309c119bd27b193b1ace2926226ac48fa0b7099bffd1ea8031774fe1d9e2 685ff8bde384958bb729d31b663e79e948f4464b659508c26af5ee39000502753nee896827fba06a2c64ac642a83cdf  
SERVER_HANDSHAKE_TRAFFIC_SECRET 212309c119bd27b193b1ace2926226ac48fa0b7099bffd1ea8031774fe1d9e2 777e8a4dbdab42ff6a7acd8e5c3b970d4a15b4c99da41c5b2c6987925b1934452b71b88904916a27722c20a1ce204c3  
EXPORTER_SECRET 212309c119bd27b193b1ace2926226ac48fa0b7099bffd1ea8031774fe1d9e2 f9834d40ac786211d6047257b9d837bca553286d1d8860d21d5265d57b07e05b2ff6a13a708c8f3055d80ea8640508c  
CLIENT_TRAFFIC_SECRET_0 212309c119bd27b193b1ace2926226ac48fa0b7099bffd1ea8031774fe1d9e2 debd1313d17c55aeff07abdb01656cfd1852f5c015f9db15f2c58a10d1b9a36cc66fb992ae926fc19f522e516ee422  
SERVER_TRAFFIC_SECRET_0 212309c119bd27b193b1ace2926226ac48fa0b7099bffd1ea8031774fe1d9e2 8755f0376bda538997dbac44310fd060d82cfe799d4979b2cde2bfa351f082e03bbae7379addec9f5640c5f3cf7734
```

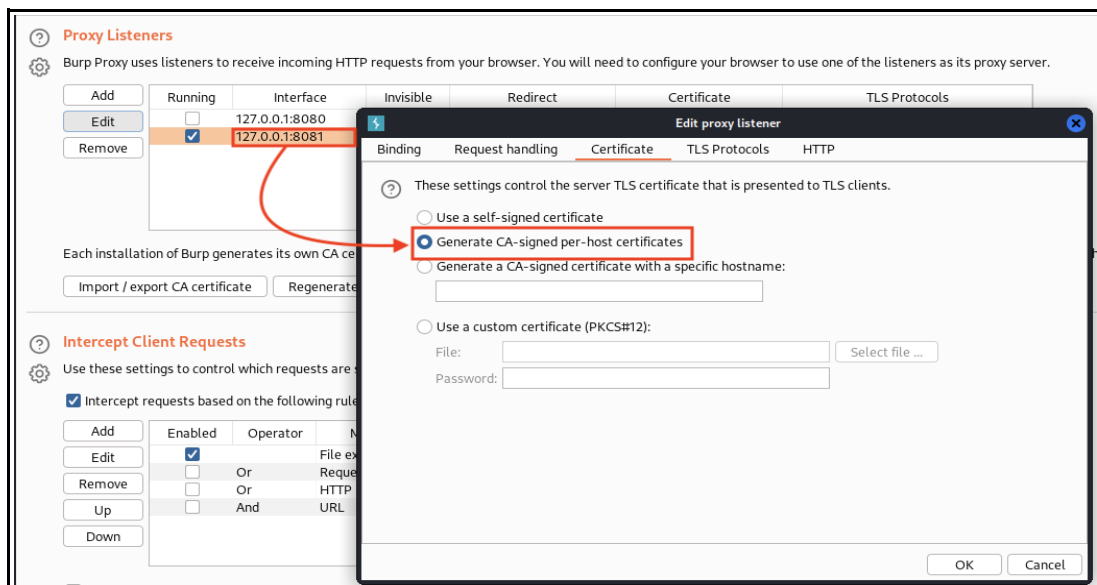

2.1- Mitm a TLS

Para realizar un ataque Mitm a TLS necesitaremos generar nuestros propios certificados **CA** firmados para tratar de que el cliente confíe en que es el servidor quien se los entrega, para ello usaremos la herramienta Burp Suite Community que nos va a facilitar el generar automáticamente esos certificados en función del host al que quiere acceder el cliente, además podremos configurar un servidor proxy en la **IP:PUERTO** que queramos para poder interceptar los paquetes y modificarlos como queramos.

Por otra parte montare también un servidor http básico en el puerto **80**, el cual arrancare en una carpeta donde guardo un índice web que solicitara automáticamente al hacer un *"request"*, para esta prueba e realizado una simple copia de un login de Instagram.



- Configuramos el proxy a la escucha en el puerto **8081** y generamos certificados **CA** firmados per-host:

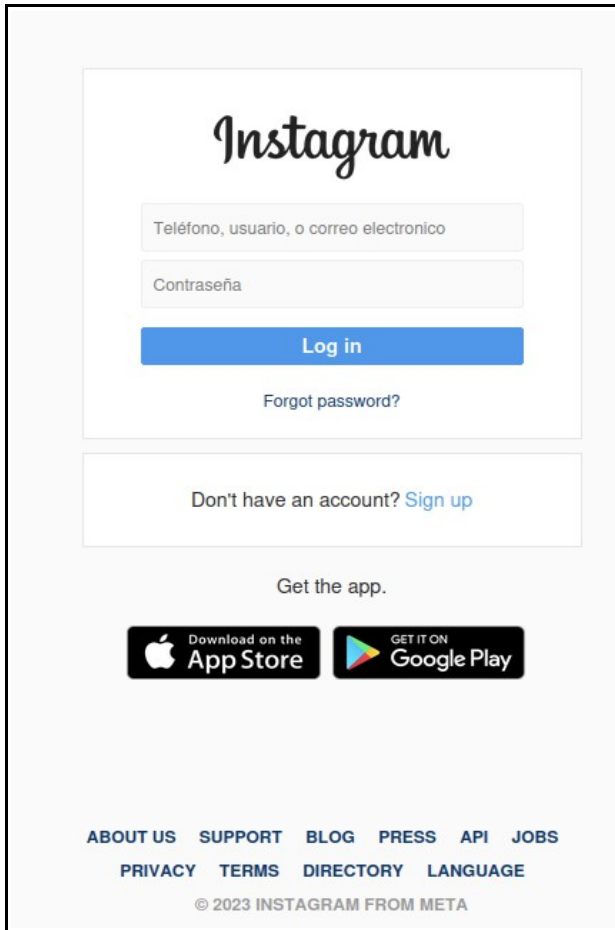


- Ponemos en el puerto **80** un servidor **HTTP** a la escucha para redireccionar las peticiones:

```
(network-rk@ network-rk)-[~/instagram]
$ ls
cacerts  index.html  logo.png  server.pem  server_ssl.py  server_tls_twisted.py  style.css
(network-rk@ network-rk)-[~/instagram]
$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/)...

```

- Una copia falsa del login de Instagram "*index.html*" bajo nuestro control atenderá al cliente desde el servidor:



```
<!-- network-k&o network-rk -> /-instagram%
<html>
<head>
<title>Instagram</title>
<link rel="stylesheet" href="http://127.0.0.1:80/style.css">
</head>
<body>
    <span id="root">
        <section class="section-all">

            <!-- 1-Role Main -->
            <main class="main" role="main">
                <div class="wrapper">
                    <article class="article">
                        <div class="content">
                            <div class="login-box">
                                <div class="header">
                                    
                                </div><!-- Header end -->
                                <div class="form-wrap">
                                    <form class="form">

                                        <div class="input-box">
                                            <input type="text" id="name" aria-describedby="" placeholder="Tel&eacuteforname">
                                        </div>

                                        <div class="input-box">
                                            <input type="password" name="password" id="password" placeholder="Contrase&nt;">
                                        </div>

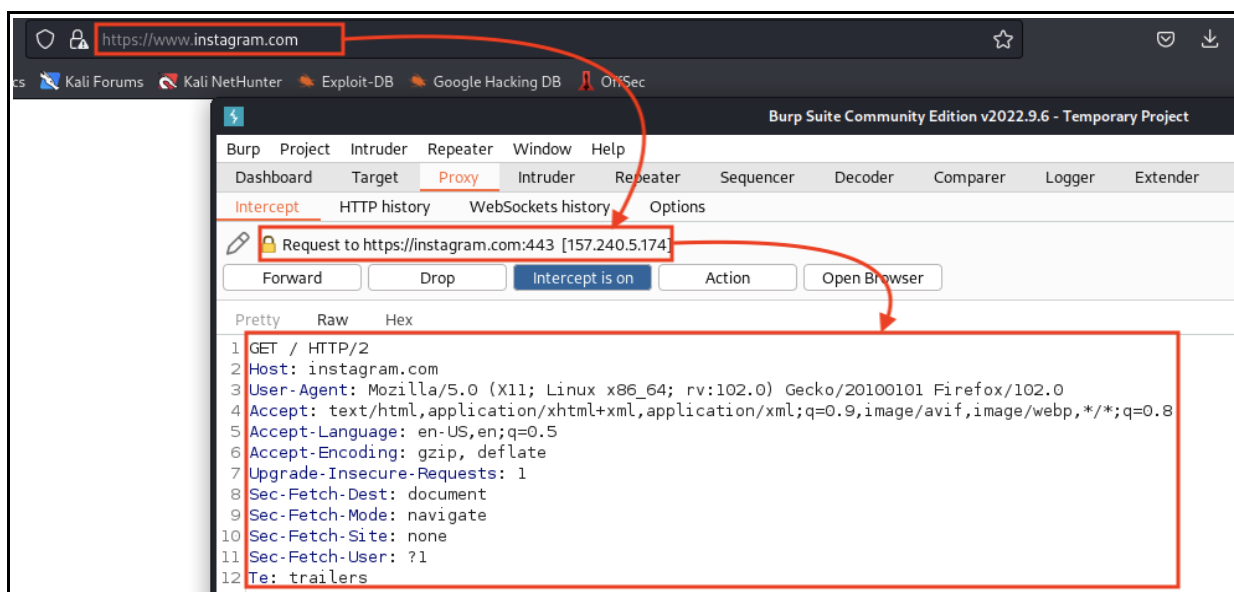
                                        <span class="button-box">
                                            <button class="btn" type="submit" name="submit">Log in</button>
                                        </span>

                                        <a class="forgot" href="#">Forgot password?</a>
                                    </form>
                                </div> <!-- Form-wrap end -->
                            </div> <!-- Login-box end -->
                        </div>
                    <div class="login-box">
                        <p class="text">Don't have an account?<a href="#">Sign up</a></p>
                    </div> <!-- Signup-box end -->
                </div>
            <div class="app">
                <p>Get the app.</p>
                <div class="app-img">
                    <a href="https://itunes.apple.com/app/instagram/id389801262?pt=428156&amp;ct=i&ing src="https://www.instagram.com/static/images/appstore-install-badges/badg
                    </a>
                    <a href="https://play.google.com/store/apps/details?id=com.instagram.android&amp;ing src="https://www.instagram.com/static/images/appstore-install-badges/badg
                    </a>
                </div> <!-- App-img end-->
            </div> <!-- App end -->
        </div> <!-- Content end -->
    </article>
</div> <!-- Wrapper end -->
</main>

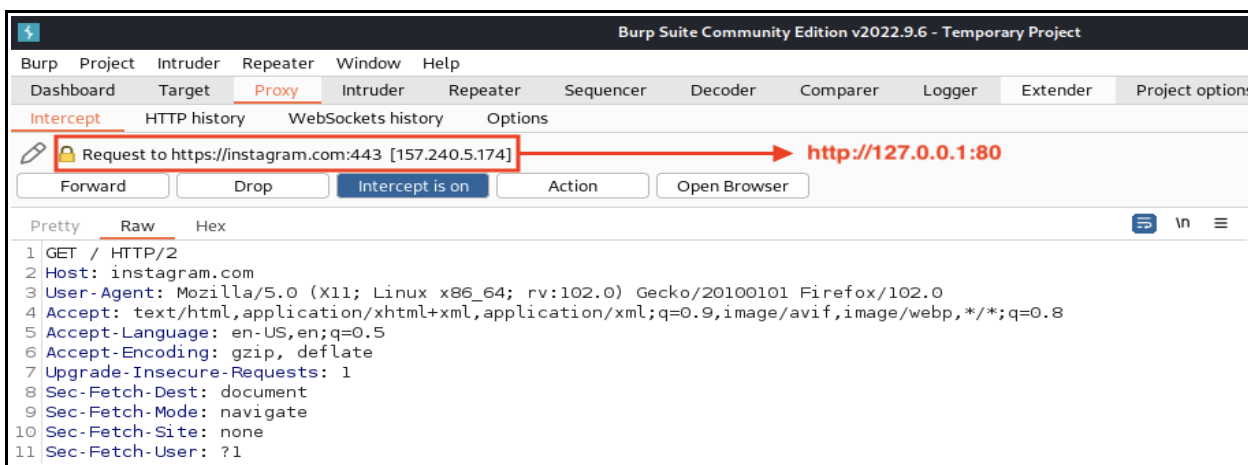
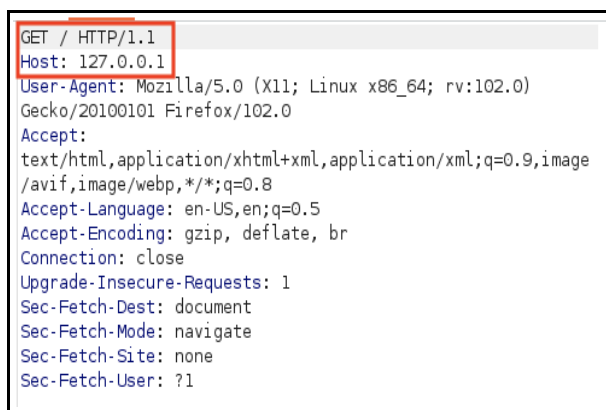
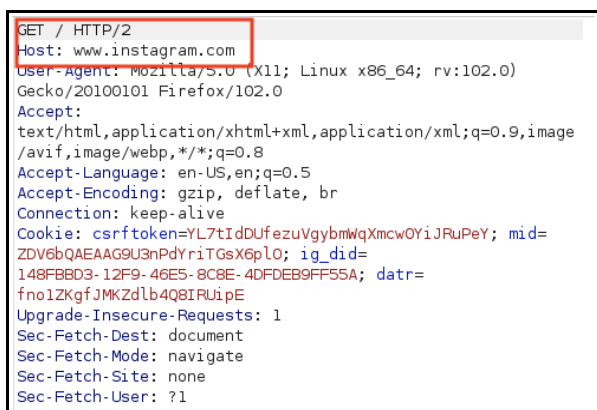
    <!-- 2-Role Footer -->
    <footer class="footer" role="contentinfo">
        <div class="footer-container">

            <nnav class="footer-nav" role="navigation">
                <ul>
                    <li><a href="#">About Us</a></li>
                    <li><a href="#">Support</a></li>
                    <li><a href="#">Blog</a></li>
                    <li><a href="#">Press</a></li>
                    <li><a href="#">Api</a></li>
                    <li><a href="#">Jobs</a></li>
                </ul>
            </div>
        </div>
    </body>
</html>
```

- Ponemos la interceptación activada y realizamos una petición a **https://www.instagram:443** desde la capa de aplicación:



- Una vez interceptada la petición la modificamos completamente para redirigirla a nuestro servidor, nuestro servidor local no admite peticiones **HTTP/2** por lo tanto modificaremos la cabecera a **HTTP/1.1**, el puerto de escucha del **80** al **443** y el host a nuestro servidor:



- *Forward request* y veamos como hemos obtenido las credenciales del usuario a través del ataque:
 - **username:** rajkit@rajkit.com
 - **password:** *rajkit*

The screenshot shows a web browser window with the Instagram login page. The username field contains 'rajkit@rajkit.com' and the password field contains 'rajkit'. A green box highlights the login form. A red box highlights the browser's address bar showing 'https://www.instagram.com'. A red arrow points from the address bar to the Burp Suite interface. The Burp Suite interface shows an intercepted HTTP request to 'https://www.instagram.com:443 [157.240.243.174]'. The request is a POST method with the following body:

```
POST /?username=rajkit%40rajkit.com&password=rajkit&submit= HTTP/2
Host: www.instagram.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: https://www.instagram.com/
```

Below the Burp Suite interface, there is a network traffic capture window showing a list of network packets. The table below represents the data shown in the capture window:

No.	Time	Source	Destination	Protocol	Length	Info
70	196.754837296	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TCP	66	57288 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
71	196.757240894	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	888	Client Hello
72	196.768824944	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TCP	66	443 → 57288 [ACK] Seq=1 Ack=820 Win=67328 Len=0
73	196.769337866	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TLSv1.3	240	Server Hello, Application Data
74	196.769365251	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TCP	66	57288 → 443 [ACK] Seq=820 Ack=175 Win=64128 Len=0
75	196.771505411	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	140	Application Data
76	196.771835985	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	128	Application Data
77	196.771929525	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	137	Application Data
78	196.772065892	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	117	Application Data
79	196.772295782	192.168.1.191	instagram-p42-shv-01-mad2.fbcdn.net	TLSv1.3	445	Application Data
80	196.783832991	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TCP	66	443 → 57288 [ACK] Seq=175 Ack=1078 Win=67328 Len=0
81	196.785212214	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TLSv1.3	242	Application Data
82	196.785540075	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TLSv1.3	140	Application Data
83	196.785540239	instagram-p42-shv-01-mad2.fbcdn.net	192.168.1.191	TLSv1.3	110	Application Data