

UNIVERSIDAD DE CASTILLA - LA MANCHA

Seguridad en
Redes

ESCUELA SUPERIOR DE INFORMÁTICA

Práctica 1 – Alice y Bob

Autor:

Silvestre SÁNCHEZ-BERMEJO
Sánchez

25 de octubre de
2021



ÍNDICE

Introducción	2
Captura de paquetes	3
Comunicación entre Alice y Bob	4
Ejemplos de la comunicación	5
Descifrado del mensaje	6
Programa decoder.....	7
Comunicación entre Alice e Internet	9
Pruebas erróneas	9
Ataque TCP Reset.....	9
Decompilers	10
HEX TO IMG	10
BASE64, BASE32	10
MD5, SHA1	10

Introducción

En esta práctica se nos han proporcionado dos archivos ejecutables, alicie y bob, con los que tendremos que trabajar para conseguir entender su funcionamiento.

El programa principal utilizado ha sido Wireshark, aunque también se han utilizado herramientas y utilidades propias de Linux, como se desarrolla a continuación.

Ambos programas están escritos en GO, esto lo podemos comprobar analizando los binarios de la siguiente manera:

```
kali@kali:~/Seguridad/p1$ strings bob | grep -w 'go'
/home/cleto/bin/go
/home/cleto/bin/go/src/internal/cpu/cpu.go
/home/cleto/bin/go/src/internal/cpu/cpu_x86.go
/home/cleto/bin/go/src/internal/cpu/cpu_x86.s
/home/cleto/bin/go/src/runtime/internal/sys/intrinsics_common.go
/home/cleto/bin/go/src/runtime/internal/atomic/asm_amd64.s
/home/cleto/bin/go/src/internal/bytealg/bytealg.go
/home/cleto/bin/go/src/internal/bytealg/index_amd64.go
/home/cleto/bin/go/src/internal/bytealg/compare_amd64.s
/home/cleto/bin/go/src/internal/bytealg/equal_amd64.s
/home/cleto/bin/go/src/internal/bytealg/index_amd64.s
/home/cleto/bin/go/src/internal/bytealg/indexbyte_amd64.s
/home/cleto/bin/go/src/runtime/alg.go
/home/cleto/bin/go/src/runtime/stubs.go
/home/cleto/bin/go/src/runtime/typekind.go
/home/cleto/bin/go/src/runtime/type.go
/home/cleto/bin/go/src/runtime/atomic_pointer.go
/home/cleto/bin/go/src/runtime/mwbbuf.go
/home/cleto/bin/go/src/runtime/cgo.go
/home/cleto/bin/go/src/runtime/cgo_mmap.go
/home/cleto/bin/go/src/runtime/cgo_sigaction.go
/home/cleto/bin/go/src/runtime/cgocall.go
/home/cleto/bin/go/src/runtime/proc.go
/home/cleto/bin/go/src/runtime/runtime2.go
/home/cleto/bin/go/src/runtime/runtime1.go
/home/cleto/bin/go/src/runtime/mheap.go
/home/cleto/bin/go/src/runtime/mbitmap.go
/home/cleto/bin/go/src/runtime/symtab.go
/home/cleto/bin/go/src/runtime/cgocallback.go
/home/cleto/bin/go/src/runtime/string.go
/home/cleto/bin/go/src/runtime/cgocheck.go
/home/cleto/bin/go/src/runtime/malloc.go
/home/cleto/bin/go/src/runtime/chan.go
/home/cleto/bin/go/src/runtime/lock_futex.go
/home/cleto/bin/go/src/runtime/lockrank_off.go
/home/cleto/bin/go/src/runtime/cpuflags_amd64.go
/home/cleto/bin/go/src/runtime/cputime.go
/home/cleto/bin/go/src/runtime/time_fake.go
/home/cleto/bin/go/src/runtime/debugcall.go
/home/cleto/bin/go/src/runtime/env_posix.go
/home/cleto/bin/go/src/runtime/error.go
/home/cleto/bin/go/src/runtime/hash64.go
/home/cleto/bin/go/src/runtime/iface.go
/home/cleto/bin/go/src/runtime/lfstack.go
/home/cleto/bin/go/src/runtime/lfstack_64bit.go
/home/cleto/bin/go/src/runtime/lockrank.go
/home/cleto/bin/go/src/runtime/mem_linux.go
/home/cleto/bin/go/src/runtime/mfixalloc.go
/home/cleto/bin/go/src/runtime/mcache.go
/home/cleto/bin/go/src/runtime/mgc.go
/home/cleto/bin/go/src/runtime/fastlog2.go
/home/cleto/bin/go/src/runtime/float.go
/home/cleto/bin/go/src/runtime/map.go
/home/cleto/bin/go/src/runtime/msize.go
/home/cleto/bin/go/src/runtime/map_fast32.go
/home/cleto/bin/go/src/runtime/map_fast64.go
/home/cleto/bin/go/src/runtime/map_faststr.go
/home/cleto/bin/go/src/runtime/mbarrier.go
```

Si lo ejecutamos con Alice, obtenemos un resultado similar

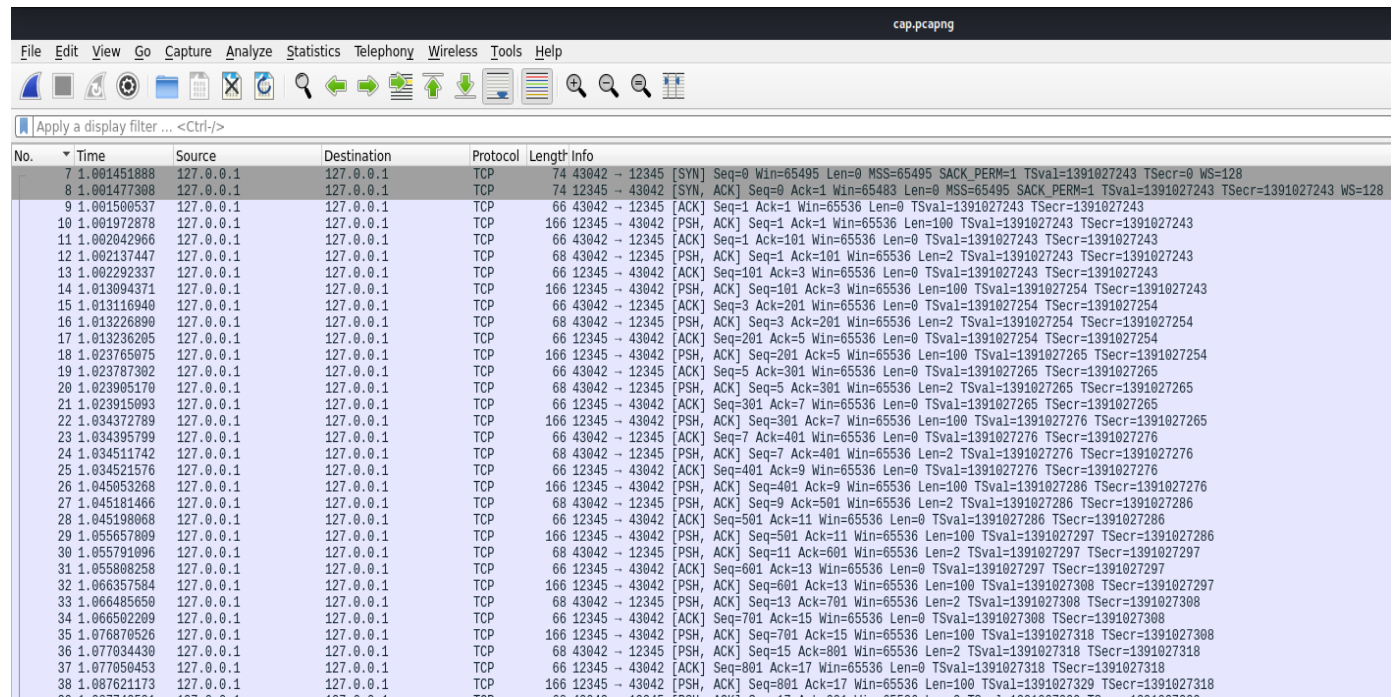
Una vez que lanzamos los dos programas, no obtenemos ningún *feedback* del terminal.

Por esto, la primera comprobación era ver si ejecutaban algún proceso demonio, con la utilidad “ps”, sin encontrar resultado alguno.

Después de esto, se comprobó el tráfico de red que generaban

Captura de paquetes

Utilizando la herramienta Wireshark, se comprueba que hay una comunicación entre ambos procesos en la interfaz *Loopback*:



cap.pcapng

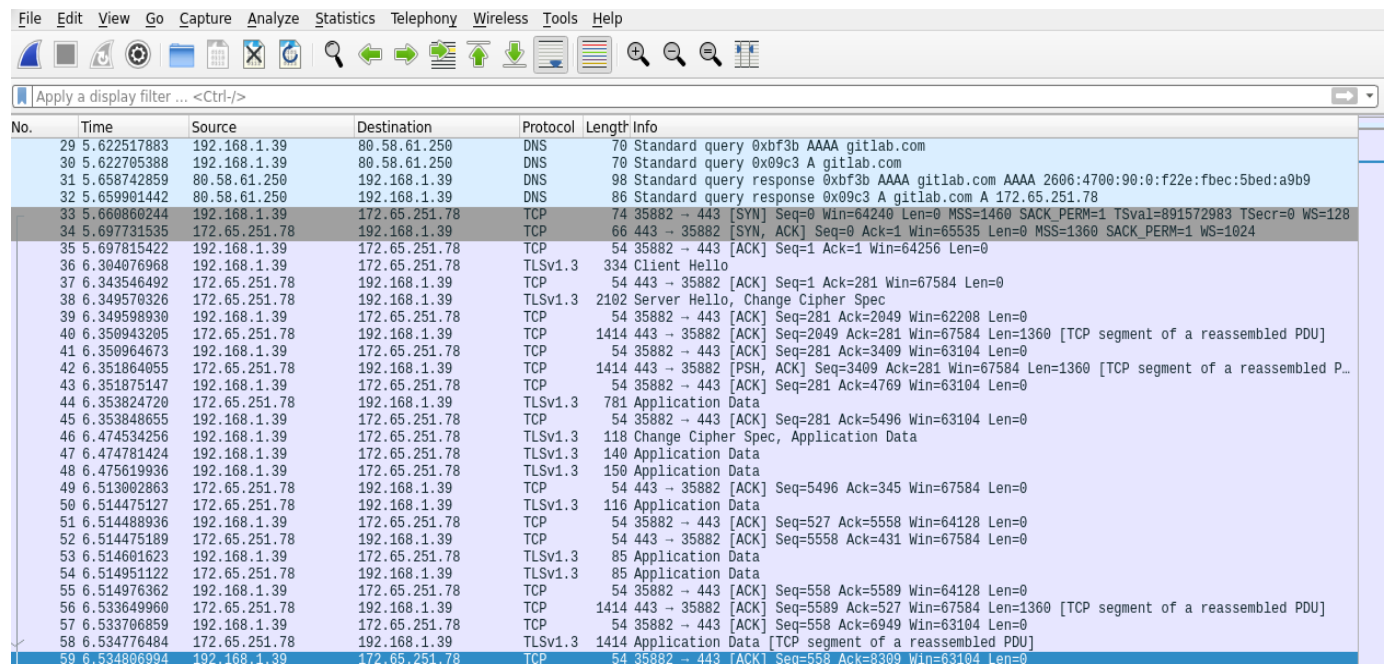
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
7	1.001451888	127.0.0.1	127.0.0.1	TCP	74	43042 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1391027243 TSecr=0 WS=128
8	1.001477398	127.0.0.1	127.0.0.1	TCP	74	12345 → 43042 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1391027243 TSecr=1391027243 WS=128
9	1.001509537	127.0.0.1	127.0.0.1	TCP	66	43042 → 12345 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1391027243 TSecr=1391027243
10	1.001972878	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=100 TSval=1391027243 TSecr=1391027243
11	1.002042966	127.0.0.1	127.0.0.1	TCP	66	43042 → 12345 [ACK] Seq=1 Ack=101 Win=65536 Len=0 TSval=1391027243 TSecr=1391027243
12	1.002137447	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=1 Ack=101 Win=65536 Len=2 TSval=1391027243 TSecr=1391027243
13	1.002292337	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=101 Ack=3 Win=65536 Len=0 TSval=1391027243 TSecr=1391027243
14	1.013094371	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=101 Ack=3 Win=65536 Len=100 TSval=1391027254 TSecr=1391027243
15	1.013116940	127.0.0.1	127.0.0.1	TCP	66	43042 → 12345 [ACK] Seq=3 Ack=201 Win=65536 Len=0 TSval=1391027254 TSecr=1391027254
16	1.013226890	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=3 Ack=201 Win=65536 Len=2 TSval=1391027254 TSecr=1391027254
17	1.013236295	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=201 Ack=5 Win=65536 Len=0 TSval=1391027254 TSecr=1391027254
18	1.023765075	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=201 Ack=5 Win=65536 Len=100 TSval=1391027265 TSecr=1391027254
19	1.023787392	127.0.0.1	127.0.0.1	TCP	66	43042 → 12345 [ACK] Seq=5 Ack=301 Win=65536 Len=0 TSval=1391027265 TSecr=1391027265
20	1.023905170	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=5 Ack=301 Win=65536 Len=2 TSval=1391027265 TSecr=1391027265
21	1.023915093	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=301 Ack=7 Win=65536 Len=0 TSval=1391027265 TSecr=1391027265
22	1.034372789	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=301 Ack=7 Win=65536 Len=100 TSval=1391027276 TSecr=1391027265
23	1.034395799	127.0.0.1	127.0.0.1	TCP	66	43042 → 12345 [ACK] Seq=7 Ack=401 Win=65536 Len=0 TSval=1391027276 TSecr=1391027276
24	1.034511742	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=7 Ack=401 Win=65536 Len=2 TSval=1391027276 TSecr=1391027276
25	1.034521576	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=401 Ack=9 Win=65536 Len=0 TSval=1391027276 TSecr=1391027276
26	1.045053268	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=401 Ack=9 Win=65536 Len=100 TSval=1391027286 TSecr=1391027276
27	1.045181466	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=9 Ack=501 Win=65536 Len=2 TSval=1391027286 TSecr=1391027286
28	1.045198068	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=501 Ack=11 Win=65536 Len=0 TSval=1391027286 TSecr=1391027286
29	1.055657899	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=501 Ack=11 Win=65536 Len=100 TSval=1391027297 TSecr=1391027286
30	1.055791096	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=11 Ack=601 Win=65536 Len=2 TSval=1391027297 TSecr=1391027297
31	1.055808258	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=601 Ack=13 Win=65536 Len=0 TSval=1391027297 TSecr=1391027297
32	1.066357584	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=601 Ack=13 Win=65536 Len=100 TSval=1391027308 TSecr=1391027297
33	1.066485650	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=13 Ack=701 Win=65536 Len=2 TSval=1391027308 TSecr=1391027308
34	1.066502209	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=701 Ack=15 Win=65536 Len=0 TSval=1391027308 TSecr=1391027308
35	1.076870526	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=701 Ack=15 Win=65536 Len=100 TSval=1391027318 TSecr=1391027308
36	1.077834430	127.0.0.1	127.0.0.1	TCP	68	43042 → 12345 [PSH, ACK] Seq=15 Ack=801 Win=65536 Len=2 TSval=1391027318 TSecr=1391027318
37	1.077850453	127.0.0.1	127.0.0.1	TCP	66	12345 → 43042 [ACK] Seq=801 Ack=17 Win=65536 Len=0 TSval=1391027318 TSecr=1391027318
38	1.087621173	127.0.0.1	127.0.0.1	TCP	166	12345 → 43042 [PSH, ACK] Seq=801 Ack=17 Win=65536 Len=100 TSval=1391027329 TSecr=1391027318

Screenshot de la captura de wireshark “cap_Alice-Bob.pcapng”

También existe una comunicación entre uno de ellos con internet, capturada mediante la interfaz wlan0:



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
29	5.622517883	192.168.1.39	80.58.61.250	DNS	70	Standard query 0xbfb3b AAAA gitlab.com
30	5.622705388	192.168.1.39	80.58.61.250	DNS	70	Standard query 0xbfb3b AAAA gitlab.com
31	5.658742859	80.58.61.250	192.168.1.39	DNS	98	Standard query response 0xbfb3b AAAA gitlab.com AAAA 2006:4700:90:0:f22e:fbec:5bed:a9b9
32	5.659901442	80.58.61.250	192.168.1.39	DNS	86	Standard query response 0xb9c3b A gitlab.com A 172.65.251.78
33	5.660860244	192.168.1.39	172.65.251.78	TCP	74	35882 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=891572983 TSecr=0 WS=128
34	5.697731535	172.65.251.78	192.168.1.39	TCP	66	443 → 35882 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 WS=1024
35	5.697815422	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
36	6.304076968	192.168.1.39	172.65.251.78	TLSv1.3	334	Client Hello
37	6.343546492	172.65.251.78	192.168.1.39	TCP	54	443 → 35882 [ACK] Seq=1 Ack=281 Win=67584 Len=0
38	6.349570326	172.65.251.78	192.168.1.39	TLSv1.3	2102	Server Hello, Change Cipher Spec
39	6.349598930	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=281 Ack=2049 Win=62208 Len=0
40	6.350943205	172.65.251.78	192.168.1.39	TCP	1414	443 → 35882 [ACK] Seq=2049 Ack=281 Win=67584 Len=1360 [TCP segment of a reassembled PDU]
41	6.350964673	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=281 Ack=3409 Win=63104 Len=0
42	6.351864055	172.65.251.78	192.168.1.39	TCP	1414	443 → 35882 [PSH, ACK] Seq=3409 Ack=281 Win=67584 Len=1360 [TCP segment of a reassembled PDU]
43	6.351875147	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=281 Ack=4769 Win=63104 Len=0
44	6.353824720	172.65.251.78	192.168.1.39	TLSv1.3	781	Application Data
45	6.353848655	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=281 Ack=5496 Win=63104 Len=0
46	6.474534256	192.168.1.39	172.65.251.78	TLSv1.3	118	Change Cipher Spec, Application Data
47	6.474781424	192.168.1.39	172.65.251.78	TLSv1.3	140	Application Data
48	6.475619936	192.168.1.39	172.65.251.78	TLSv1.3	150	Application Data
49	6.513002863	172.65.251.78	192.168.1.39	TCP	54	443 → 35882 [ACK] Seq=5496 Ack=345 Win=67584 Len=0
50	6.514475127	172.65.251.78	192.168.1.39	TLSv1.3	116	Application Data
51	6.514488936	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=527 Ack=5558 Win=64128 Len=0
52	6.514475189	172.65.251.78	192.168.1.39	TCP	54	443 → 35882 [ACK] Seq=5558 Ack=431 Win=67584 Len=0
53	6.514601623	192.168.1.39	172.65.251.78	TLSv1.3	85	Application Data
54	6.514951122	172.65.251.78	192.168.1.39	TLSv1.3	85	Application Data
55	6.514976362	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=558 Ack=5589 Win=64128 Len=0
56	6.533649960	172.65.251.78	192.168.1.39	TCP	1414	443 → 35882 [ACK] Seq=5589 Ack=527 Win=67584 Len=1360 [TCP segment of a reassembled PDU]
57	6.533706859	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=558 Ack=6949 Win=63104 Len=0
58	6.534776484	172.65.251.78	192.168.1.39	TLSv1.3	1414	Application Data [TCP segment of a reassembled PDU]
59	6.534806994	192.168.1.39	172.65.251.78	TCP	54	35882 → 443 [ACK] Seq=558 Ack=8309 Win=63104 Len=0

Screenshot de la captura de Wireshark “cap_Alice-WLAN.pcapng”

Por lo tanto, debemos ver qué roles tienen cada uno de los programas; utilizando la herramienta netstat (y con ayuda de grep) vemos que puertos utiliza cada proceso:

```
kali@kali:~/Seguridad/p1$ netstat pid -p | grep -w 'alice\|bob'
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 192.168.1.46:44472    172.65.251.78:https  ESTABLISHED 4528/./alice
tcp        0      2 localhost:46986       localhost:12345      ESTABLISHED 4524/./bob
tcp        0      0 localhost:12345       localhost:46986      ESTABLISHED 4528/./alice
```

(Nota: en las capturas, el puerto de Bob puede variar ya que son distintas ejecuciones, pero el de Alice se mantiene)

Con esta información tenemos que:

- En la comunicación loopback, Alice se ejecuta en el puerto 12345
- Bob se ejecuta en el 46986
- Es Alice la que se comunica con internet

Sabiendo esto, podemos comenzar a analizar el tráfico entre ambos procesos

Comunicación entre Alice y Bob

En toda la comunicación tenemos esa estructura:

- Alice envía un mensaje de 166 bytes (exceptuando el último) que contiene datos cifrados
- Bob responde con mensajes PSH, ACK que contienen el texto en claro "OK"

El primer objetivo es obtener todos los datos encriptados en un mismo archivo, para esto, hacemos click derecho en un mensaje de la conversación-> Follow-> TCP Stream.

De esta manera, podemos seleccionar únicamente los datos que ha enviado Alice, y exportarlos.

Para una mayor eficiencia, he exportado los datos en ASCII y en hexadecimal (dump.txt y dump_raw.txt respectivamente), para utilizar uno u otro en función de las pruebas.

Ejemplos de la comunicación

```

▶ Frame 131: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits) on interface lo, id 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 12345, Dst Port: 43042, Seq: 3901, Ack: 79, Len: 100
▼ Data (100 bytes)
  Data: 797d6d287b6d28746d2878697b696a69762874697b151276...
  [Length: 100]

```

0000	00 00 00 00 00 00 00 00	00 00 00 00 08 00 45 00E.
0010	00 98 c9 28 40 00 40 06	73 35 7f 00 00 01 7f 00	...(@.@. s5.....
0020	00 01 30 39 a8 22 a1 36	d6 a3 51 67 b6 8a 80 18	..09."6 ..Qg....
0030	02 00 fe 8c 00 00 01 01	08 0a 52 e9 65 d4 52 e9R.e.R.
0040	65 c9 79 7d 6d 28 7b 6d	28 74 6d 28 78 69 7b 69	e.y}m({m (tm(xi{i
0050	6a 69 76 28 74 69 7b 15	12 76 77 6b 70 6d 7b 28	jiv(ti{. .vwkpm{(
0060	74 6d 81 6d 76 6c 77 28	6c 6d 28 6b 74 69 7a 77	tm.mvlw(lm(ktizw
0070	28 6d 76 28 6b 74 69 7a	77 34 28 81 28 74 77 7b	(mv(ktiz w4(. (tw{
0080	28 6c cb b5 69 7b 28 6c	6d 28 7c 7d 7a 6a 71 77	(l..i{(l m(}zjqw
0090	28 6d 76 28 7c 7d 7a 6a	71 77 43 28 81 28 69 7b	(mv(}zj qwC(. (i{
00a0	cb b5 34 15 12 6c		..4..]

Ejemplo de los paquetes enviados por Alice

```

▶ Frame 132: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface lo, id 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 43042, Dst Port: 12345, Seq: 79, Ack: 4001, Len: 2
▼ Data (2 bytes)
  Data: 4f4b
  [Length: 2]

```

0000	00 00 00 00 00 00 00 00	00 00 00 00 08 00 45 00E.
0010	00 36 b4 84 40 00 40 06	88 3b 7f 00 00 01 7f 00	.6..@.@. .;.....
0020	00 01 a8 22 30 39 51 67	b6 8a a1 36 d7 07 80 18	... "09Qg ...6....
0030	02 00 fe 2a 00 00 01 01	08 0a 52 e9 65 d4 52 e9	...*.....R.e.R.
0040	65 d4 4f 4b		e.OK

Ejemplo de los paquetes enviados por Bob

Descifrado del mensaje

He utilizado la siguiente página web: <https://www.dcode.fr/ascii-shift-cipher>, la cual, permite decodificar un cifrado César en ASCII probando varios *shifts*.

Es un cifrado César con desplazamiento de 8, ya que podemos ver texto en claro, más concretamente, el Quijote:

Results

Primera parte del ingenioso hidalgo don Quijote de la ManchaCaptulo primero. Que trata de la condicin 0 ejercicio del famoso hidalgodon Quijote de la ManchaEn un lugar de la Mancha, de cuo nombre no quiero ac0ordarme, no ha muchotiempo que viva un hidalgo de los de lana en astillero, adarga antigua,rocn flaco 0galgo corredor. Una olla de algo ms vaca que carnero,salpicn las ms noches, duelos quebrantos los sba0dos, lantejas losviernes, algn palomino de aadidura los domingo

+8

Tvmqive\$tevxiship\$mrkirmsw\$lmhepks\$hsr \$Uymnsxi\$hi\$pe\$QergleGetxyps\$tmqivs2\$U yi\$vxex\$hi\$pe\$grhmgmr\$0\$inivgmgs\$hip \$jeqsws\$lmhepkshsr\$Uymnsxi\$hi\$pe\$Qergle Ir\$yr\$pykev\$hi\$pe\$Qergle0\$hi\$gys\$rsqfvi \$rs\$uymivs\$eg0\$vhvq10\$rs\$le\$yglxmiqt s\$uyi\$zme\$yr\$lmhepks\$hi\$psw\$hi\$pere\$ir \$ewxmpiv\$0\$ehvke\$erxmkye0\$vsgr\$jpjeps

+4

0kepk\$gsvvihsv2\$Yre\$sppe\$hi\$epks \$qw\$zege\$uyi\$gevriv\$0weptmgr\$pew \$qw\$rsqliw0\$hyipsu

uyifverxsw\$psw\$wfe0hsw0\$perxinew\$pswzmi vriw0\$epkr\$stepsqmr\$hi\$eehmhyve\$psw\$hsq mrks

Rtkogtc"rctvg"fgn"kpigpkquq"jkfcniq"fqp "Swklqvg"fngc"0cpejcEcrvwnq"rtkogtq0"S wg"vtcvc"fngc"eqpfkekp"0glgtekekq"fgn "hcoquq"jkfcniqfqp"Swklqvg"fngc"0cpejc Gp"wp"nwict"fngc"0cpejc."fg"ewq"pqodtg "pq"swkgqtq"ce0qtftctog."pq"jc"owejqvkgor q"swg"xxxc"wp"jkfcniq"fngnqu"fngncpc"gp "cuvknngtq."cfctic"cpvkiwc.tqep"hnceq"" 0icniq"eqttgfat0"Wpc"qnnc"fngcniq"ou"xc

+6

ASCII SHIFT DECODER

★ ASCII SHIFTED CIPHERTEXT

ASCII Printable Characters (Automatic Detection)

000Xzqumzi(xiz|m(lmt(qvomvqw{w(pqlitow(lwv(Y}qrw|m(lm(ti(UivkpiKix-})tw(xzqumzw6(Y)m(|zi|i(lm(ti(kwvlqkq.v(0(mrmzkqkqw(lmt(niuw{w(pqlitowlwv(Y}qrw|m(lm(ti(Uivkpi Mv{)v(t)0iz(lm(ti(Uivkpi4(lm(k)0w(vwuujzm(vw(y)qmw(ik wzlizum4(vw(pi(u)kpw|qmuxw(y)m(~q~l{)v(pqlitow(lm(tw{(l m(tiv0i(mv(i{|qttmzw4(ilizoi(iv|qo)i4zkw~v(ntikw0(

TRY ALL POSSIBLE SHIFTS (FROM 1 TO 127)

USE A SHIFT OF 8

★ RESULTS FORMAT

ASCII (PRINTABLE) CHARACTERS

HEXADECIMAL 00-7F-FF

DECIMAL 0-127-255

OCTAL 000-177-377

BINARY 00000000-11111111

INTEGER NUMBER

DECRYPT

See also: ASCII Code — ROT-47 Cipher

ASCII SHIFT ENCODER

★ ASCII SHIFTED PLAINTEXT

ASCII Printable Characters (Automatic Detection)

USE A SHIFT OF 64

★ RESULTS FORMAT

ASCII (PRINTABLE) CHARACTERS

HEXADECIMAL 00-7F-FF

DECIMAL 0-127-255

OCTAL 000-177-377

BINARY 00000000-11111111

INTEGER NUMBER

ENCRYPT

See also: Shift Cipher — Caesar Cipher

Y además,

Smartphones desde solo 16

Programa decoder

Para hacer más cómodo el descryptado, he creado un programa en python el cual, dado un archivo en ASCII, el shift y una ruta de destino, decodifica el encriptado cesar.

Este programa lee un archivo en formato ASCII, e itera cada carácter restando el shift correspondiente, para obtener el carácter descryptado.

```
1#!/usr/bin/python3 -u
2# -*- coding: utf-8 -*-
3import os,sys,io
4
5decrp=""
6
7try:
8    file,shift,d_file=sys.argv[1:]
9    shift=int(shift)
10except ValueError:
11    print('Command arguments: {} <encoded file> <shift> <decoded file>'.format(
12        os.path.basename(sys.argv[0])))
13    )
14    sys.exit(1)
15
16
17f_in = io.open(file,'r',encoding='ascii',errors='ignore')
18txt= f_in.read()
19
20for char in txt:
21    value= ord(char) - shift
22    decrp+= chr(value%128)
23
24with open(d_file,'w') as f_out:
25    f_out.write(decrp)
26
27print("Succesfully decoded, saved into: " + d_file)
28sys.exit(0)
```

Código en python del decoder


```
kali@kali:~/Seguridad/p1$ ./caesar_decoder.py dump.txt 8 Quijote_P1.txt
Succesfully decoded, saved into: Quijote_P1.txt
kali@kali:~/Seguridad/p1$ more Quijote_P1.txt
Primera parte del ingenioso hidalgo don Quijote de la Mancha

Captulo primero. Que trata de la condicin ejercicio del famoso hidalgo
don Quijote de la Mancha

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho
tiempo que viva un hidalgo de los de lana en astillero, adarga antigua,
rocn flaco galgo corredor. Una olla de algo ms vaca que carnero,
salpicn las ms noches, duelos quebrantos los sbados, lantejas los
viernes, algn palomino de aadidura los domingos, consuman las tres
partes de su hacienda. El resto della concluan sao de velarte, calas de
velludo para las fiestas, con sus pantuflos de lo mismo, los das de
entresemana se honraba con su vellor de lo ms fino. Tena en su casa una
ama que pasaba de los cuarenta, una sobrina que no llegaba a los veinte,
un moo de campo plaa, que as ensillaba el rocn como tomaba la
podadera. Frisaba la edad de nuestro hidalgo con los cincuenta aos; era de
complein recia, seco de carnes, enjuto de rostro, gran madrugador amigo
de la caa. Quieren decir que tena el sobrenombre de Quijada, o Quesada,
que en esto ha alguna diferencia en los autores que deste caso escriben;
aunque, por conjeturas verosmiles, se deja entender que se llamaba
Quejana. Pero esto importa poco a nuestro cuento; basta que en la narracin
dl no se salga un punto de la verdad.

Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba
ocioso, que eran los ms del ao, se daba a leer libros de caballeras, con
tanta afinidad gusto, que olvid casi de todo punto el ejercicio de la
caa, aun la administracin de su hacienda. Y lleg a tanto su curiosidad
desatino en esto, que vendi muchas hanegas de tierra de sembradura para
comprar libros de caballeras en que leer, as, llev a su casa todos
cuantos pudo haber dellos; de todos, ningunos le parecan tan bien como
--More-- (0%)
```

Funcionamiento de caesar_decoder.py

Comunicación entre Alice e Internet

Como se puede ver en la captura adjunta anteriormente, Alice hace una serie de peticiones a gitlab.com, datos que se ven en las peticiones DNS.

Después, el envío de datos está completamente cifrado ya que es una conexión TLS, es decir, utilizando https.

Se ha intentado realizar un ataque MITM (Man in the middle), el cual consiste en ponerse en medio de la conexión para capturar los datos, pero sin resultados.

Pruebas erróneas

Todas estas pruebas fueron realizadas antes de llegar al resultado final

Ataque TCP Reset

Un ataque TCP reset, o TCP Reset Attack, se da cuando un atacante envía paquetes pequeños con el flag RST a ambos extremos de una comunicación, para que ambos piensen que ha finalizado.

En un primer momento pensé que era el caso de un ataque TCP Reset, ya que, si ejecutamos únicamente a Bob, y capturamos el tráfico de loopback, nos encontramos con lo siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	:::1	:::1	TCP	94	38436 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737065911 TSecr=0 W... [
2	0.000019140	:::1	:::1	TCP	74	12345 → 38436 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.000199440	127.0.0.1	127.0.0.1	TCP	74	42906 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007382202 TSecr=0 W...
4	0.000224952	127.0.0.1	127.0.0.1	TCP	54	12345 → 42906 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	1.009707595	:::1	:::1	TCP	94	38448 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737066921 TSecr=0 W...
6	1.009726010	:::1	:::1	TCP	74	12345 → 38448 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	1.009898738	127.0.0.1	127.0.0.1	TCP	74	42910 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007383211 TSecr=0 W...
8	1.009923105	127.0.0.1	127.0.0.1	TCP	54	12345 → 42910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	2.010791744	:::1	:::1	TCP	94	38444 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737067922 TSecr=0 W...
10	2.010810286	:::1	:::1	TCP	74	12345 → 38444 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	2.010977573	127.0.0.1	127.0.0.1	TCP	74	42914 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007384212 TSecr=0 W...
12	2.011002361	127.0.0.1	127.0.0.1	TCP	54	12345 → 42914 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	3.011025854	:::1	:::1	TCP	94	38448 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737068923 TSecr=0 W...
14	3.011044486	:::1	:::1	TCP	74	12345 → 38448 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	3.0110812912	127.0.0.1	127.0.0.1	TCP	74	42918 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007385213 TSecr=0 W...
16	3.0110837422	127.0.0.1	127.0.0.1	TCP	54	12345 → 42918 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	4.012823173	:::1	:::1	TCP	94	38452 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737069924 TSecr=0 W...
18	4.012841816	:::1	:::1	TCP	74	12345 → 38452 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	4.013013593	127.0.0.1	127.0.0.1	TCP	74	42922 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007386214 TSecr=0 W...
20	4.013039472	127.0.0.1	127.0.0.1	TCP	54	12345 → 42922 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	5.014472338	:::1	:::1	TCP	94	38456 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737070926 TSecr=0 W...
22	5.014490693	:::1	:::1	TCP	74	12345 → 38456 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	5.014657875	127.0.0.1	127.0.0.1	TCP	74	42926 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007387216 TSecr=0 W...
24	5.014682560	127.0.0.1	127.0.0.1	TCP	54	12345 → 42926 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	6.015467488	:::1	:::1	TCP	94	38460 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737071927 TSecr=0 W...
26	6.015485566	:::1	:::1	TCP	74	12345 → 38460 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	6.015657083	127.0.0.1	127.0.0.1	TCP	74	42930 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007388217 TSecr=0 W...
28	6.015682097	127.0.0.1	127.0.0.1	TCP	54	12345 → 42930 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	7.016076197	:::1	:::1	TCP	94	38464 → 12345 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=1 TSval=1737072928 TSecr=0 W...
30	7.016093452	:::1	:::1	TCP	74	12345 → 38464 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	7.016863833	127.0.0.1	127.0.0.1	TCP	74	42934 → 12345 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1007389218 TSecr=0 W...
32	7.016890154	127.0.0.1	127.0.0.1	TCP	54	12345 → 42934 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Screenshot de la captura de Wireshark "capLoopBack_Bob.pcapng"

La idea era que Bob estaba intentando dificultar la comunicación entre Alice e Internet, pero realmente se estaba intentando conectar con Alice.

Decompilers

Se intentó obtener el código fuente de los binarios, pero dicha operación es muy compleja, y en la mayoría de lenguajes es excesivamente complicado obtener código veraz.

HEX TO IMG

Al tener tan cantidad de datos en hexadecimal, pensé que podría ser una imagen, para comprobarlo, utilicé una pagina (<https://codepen.io/abdhass/full/jdRNdj>) que hace esta conversión, sin dar resultado.

BASE64, BASE32

Se intentó también convertir todo el texto con ambos formatos

MD5, SHA1

Ya que en clase se comentó que estos hashes estaban rotos, se hicieron pruebas para intentar decodificarlos sin resultado alguno