

WIRESHARK: **ANALISI DI UNA CATTURA**

Alberto Martino

Edoardo Coli

Indice

Introduzione	2
Riepilogo della Cattura	3
Chi è l'utente?	4
Cosa fa l'utente?	6
TLS: Handshake	8
Client Hello → us-u.openx.net	9
Domain Names	10
Stock Ticker	13
Time To Live	15
Cipher Suite	16
1-byte data packets	17
Unencrypted browsing	18
Websites visited	19
Considerazioni	21

Introduzione

In questa relazione illustriamo l'analisi di una cattura effettuata con Wireshark rispondendo alle seguenti domande:

- How many TLS **Client Hello** packets are being sent to us-u.openx.net?
- What **cipher suite** was selected by Bank of America's server?
- How many **domain names** can you associate with J.P. Morgan?
- What is the purpose of the **1-byte data packets** in this trace file?
- What is the full name of the bank that allows **unencrypted browsing** to their site?
- What is the name of the developer of the **stock ticker** application?
- What is the highest DNS **Time to Live** value in the trace file?
- In what order did the user visit the **bank web sites**?

Nel **capitolo introduttivo** abbiamo tracciato un **quadro generale** della cattura approfondendo alcuni aspetti ritenuti fondamentali per poter rispondere ai quesiti posti.

Nei **capitoli seguenti** abbiamo risposto alle **domande**, mostrando il procedimento seguito.

Infine, nell'**ultimo capitolo**, abbiamo raccolto delle **considerazioni generali** relative all'analisi.

Riepilogo della Cattura

In questo capitolo cerchiamo di tracciare un quadro generale della cattura sfruttando informazioni riepilogative di Wireshark.

Visualizzando le proprietà di cattura

- Statistiche/Proprietà file di cattura

osserviamo che questa consiste in **poco più di un minuto** di pacchetti catturati su un'interfaccia **ethernet** su sistema operativo **Windows 10**, nel 2017, per un totale di **~8.94 mb** di dati. La cattura risulta non filtrata.

Visualizzando le gerarchie dei protocolli

- Statistiche/Gerarchie dei protocolli

cerchiamo di individuare quali protocolli sono stati utilizzati maggiormente e se vi sono alcune caratteristiche che riteniamo di dover approfondire tenendo conto del fatto che le percentuali mostrate da Wireshark riguardano una suddivisione gerarchica (non concettuale) e che sono da intendersi come indicative ([riferimento](#)).

A livello rete, la maggior parte dei pacchetti è passata per **IPv4** (~78%) ma abbiamo anche una porzione passata per **IPv6** (~21%). A livello di trasporto, col protocollo **TCP** passa la maggior parte dei byte (~99.9%) rispetto ad **UDP** (<0.1%). Considerando il livello gerarchico superiore, la maggior parte dei byte passa su **TLS** (~81.3%) seguito da **HTTP** (~10.4%) che conta poco meno di 1mb di dati.

Osserviamo che vi sono anche comunicazioni **IGMP** (26 pacchetti, 504 byte), **OCSP** (118 pacchetti, ~0.07 mb), **ARP** (**99 pacchetti**, 4446 byte), **DNS** (712 pacchetti, ~0.05 mb) e **DB-LSP-DISC** (8 pacchetti, 1080 byte).

In conclusione, osserviamo che si tratta di un'analisi passiva breve e non filtrata che consiste prevalentemente di traffico criptato presumibilmente da/a molteplici host (visto l'alto numero di pacchetti DNS); crediamo valga la pena approfondire le comunicazioni ARP, IGMP e Dropbox; terremo in considerazione la presenza di comunicazioni IPv6, la data della cattura e il fatto che vi sia anche una quantità interessante di traffico in chiaro; vista la prevalenza di comunicazioni TLS, approfondiamo l'argomento così da poter analizzare la cattura e rispondere alle domande con maggior consapevolezza.

Chi è l'utente?

In questo capitolo cerchiamo di identificare **chi sia l'utente**; inoltre, vorremmo capire qualcosa in più riguardo le comunicazioni **IGMP**, **ARP** e quelle legate a **Dropbox**.

La prima cosa che notiamo è che vi sono più host mappati ad IPv4 privati, di fatti, filtrando la cattura con:

- `ip.addr in {192.168.1.0/24}`

e visualizzando a livello di *data link* solo le conversazioni inerenti notiamo che sono presenti 9 host locali e che uno, fra questi, ha generato sensibilmente più traffico degli altri (MAC: d4:3d:7e:a6:41:fd, IPv4: 192.168.1.70, [MSI](#)).

Ethernet · 12	IPv4 · 73	IPv6 · 4	TCP · 155	UDP · 87		
Address A	Address B	Packets	Bytes A → B	Bytes B → A	Bytes	
00:23:a2:ed:e1:bb	01:00:5e:00:00:16	2	124	0	124	
00:23:ee:e9:10:4a	01:00:5e:00:00:16	2	124	0	124	
01:00:5e:00:00:01	dc:7f:a4:2c:0b:15	2	0	120	120	
01:00:5e:00:00:16	bc:ee:7b:0e:0b:82	7	0	422	422	
01:00:5e:00:00:16	d4:3d:7e:a6:41:fd	7	0	386	386	
01:00:5e:00:00:16	fc:6f:b7:ce:f6:74	2	0	156	156	
01:00:5e:00:00:16	2c:ab:a4:95:18:5d	2	0	124	124	
01:00:5e:00:00:fb	16:da:43:06:c0:e8	7	0	760	760	
01:00:5e:00:00:fd	d4:3d:7e:a6:41:fd	1	0	82	82	
4c:80:93:06:e2:8f	ff:ff:ff:ff:ff:ff	8	2144	0	2144	
bc:ee:7b:0e:0b:82	d4:3d:7e:a6:41:fd	3	188	94	282	
d4:3d:7e:a6:41:fd	dc:7f:a4:2c:0b:15	9.141	623k	7062k	7685k	

La presenza di richieste ARP multiple ad intervalli brevi e le comunicazioni IGMP hanno colto la nostra attenzione; queste richieste sono fatte dall'indirizzo 191.168.1.254 (il *querier*) associato ad un'interfaccia prodotta da [2Wire](#) (MAC: dc:7f:a4:2c:0b:15). Ipotizzando che questi funga da gateway per una LAN, filtriamo approssimativamente la cattura con:

- `ip.src in { 192.168.1.0/24 } && !(ip.dst in { 192.168.1.0/24 224.0.0.0/24 255.255.255.255 }) && !(eth.dst == dc:7f:a4:2c:0b:15)`

ottenendo tutti i pacchetti che vanno da un IPv4 privato verso un indirizzo pubblico (no multicast/broadcast) senza dirigersi verso la suddetta interfaccia di 2Wire: trattasi di una lista vuota, a supporto dell'ipotesi.

Per quanto riguarda il traffico multicast, notiamo subito sfruttando gli I/O graphs di Wireshark che questi è esiguo e, più precisamente, filtrando solo i pacchetti di comunicazione inerenti

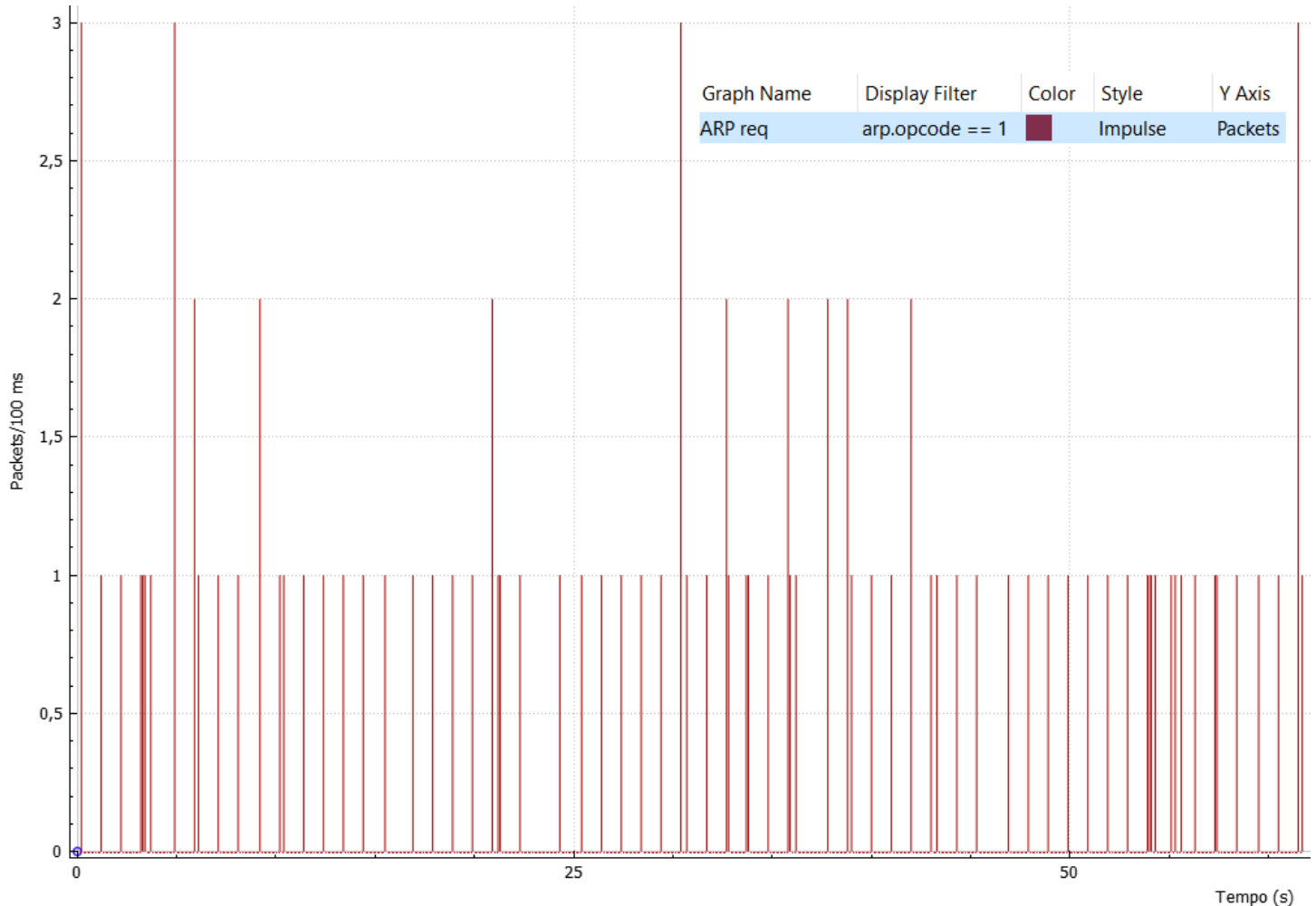
- `ip.dst in { 224.0.0.0/24 } || ipv6.addr >= ff00::`

notiamo si tratti di 67 messaggi composti principalmente da traffico IGMP ([report](#)), ICMP ([sollecitazioni](#)) e query MDNS, che non riteniamo interessanti da approfondire. Una riflessione analoga può essere fatta anche per il traffico Dropbox, riguardante solamente comunicazioni di discovery che coinvolgono attivamente solo uno degli host (MAC: 4c:80:93:06:e2:8f).

Per quanto riguarda le richieste ARP osserviamo che sono mandate ripetutamente coprendo tutta la durata della cattura come mostra il grafico seguente. Queste riguardano 48 IP distinti (che occorrono non ordinati) nel range 192.168.1.0/24; anche considerando che le impostazioni dell'OS possano prevedere un [Reachable Time](#) basso per l'interfaccia, ci appare un comportamento insolito. Potremmo ad esempio considerare l'eventualità di un attacco MIM via Arp Spoofing ([RFC](#)); prendendo spunto dall'RFC citato, notiamo intanto che (quantomeno in questa cattura) non vi sono "IP duplicati": ciò è osservabile semplicemente applicando il filtro

- *arp.duplicate-address-frame*

verificando che risulta in una lista vuota oppure controllando l'assenza di un relativo warning che darebbe Wireshark (Expert Info Warning "Duplicate IP Address Configured"). Ad ogni modo, ignorando il tema, ci limitiamo a queste osservazioni e proseguiamo.



In conclusione, crediamo si tratti di una cattura fatta sulla porta di un gateway (2Wire) che interfaccia gli host di una LAN alla rete pubblica; abbiamo un host (MSI) sensibilmente più attivo rispetto agli altri e crediamo che traffico IGMP (multicast) e Dropbox non meriti ulteriori approfondimenti. Il traffico ARP ci insospettisce, ma dopo qualche controllo e ricerca dovuti decidiamo anche in questo caso di non procedere con ulteriori approfondimenti, senza prendere una posizione precisa a riguardo.

Cosa fa l'utente?

In questo capitolo tenteremo di capire **cosa ha fatto (principalmente) l'utente**, in caso vi sia un qualche tipo di consistenza evidente nella sua attività.

Modifichiamo le impostazioni di Wireshark in modo da abilitare la risoluzione di nomi per IP e trasporto

- *Modifica/Preferenze/Name Resolution.*

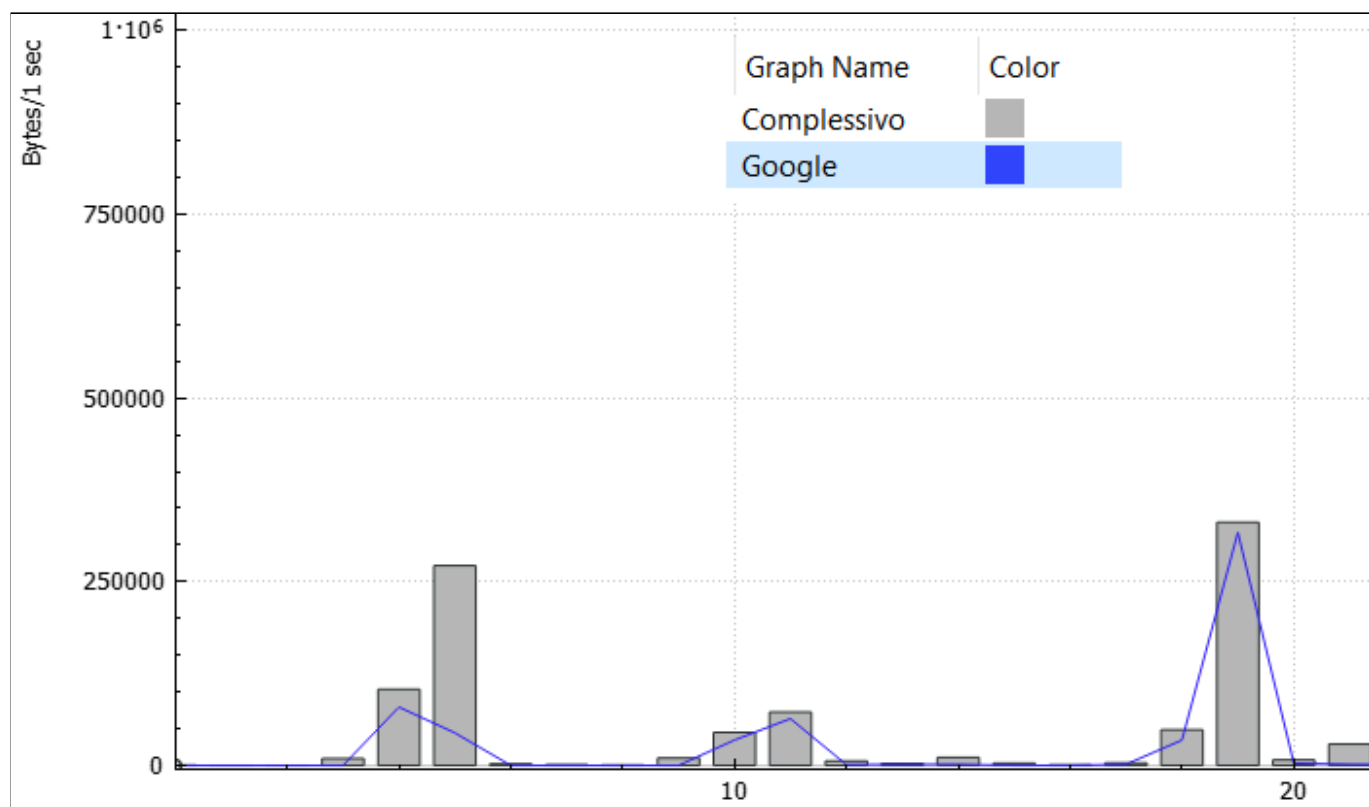
Essendoci un host sensibilmente più attivo di tutti gli altri, ci interessiamo solo ad esso: filtriamo la cattura in modo da considerare solo le sue conversazioni.

- *eth.addr == d4:3d:7e:a6:41:fd*

A questo punto, ordiniamo la cattura in base al tempo e scorriamo le entries.

Osserviamo che una prima parte della cattura (circa 20 secondi) consiste principalmente nell'attività web tramite il motore di ricerca Google. A sostegno di ciò, sfruttiamo gli I/O graphs comparando il traffico non filtrato con parte di quello associabile a Google:

- Google: *ipv6.host == www.google.com*

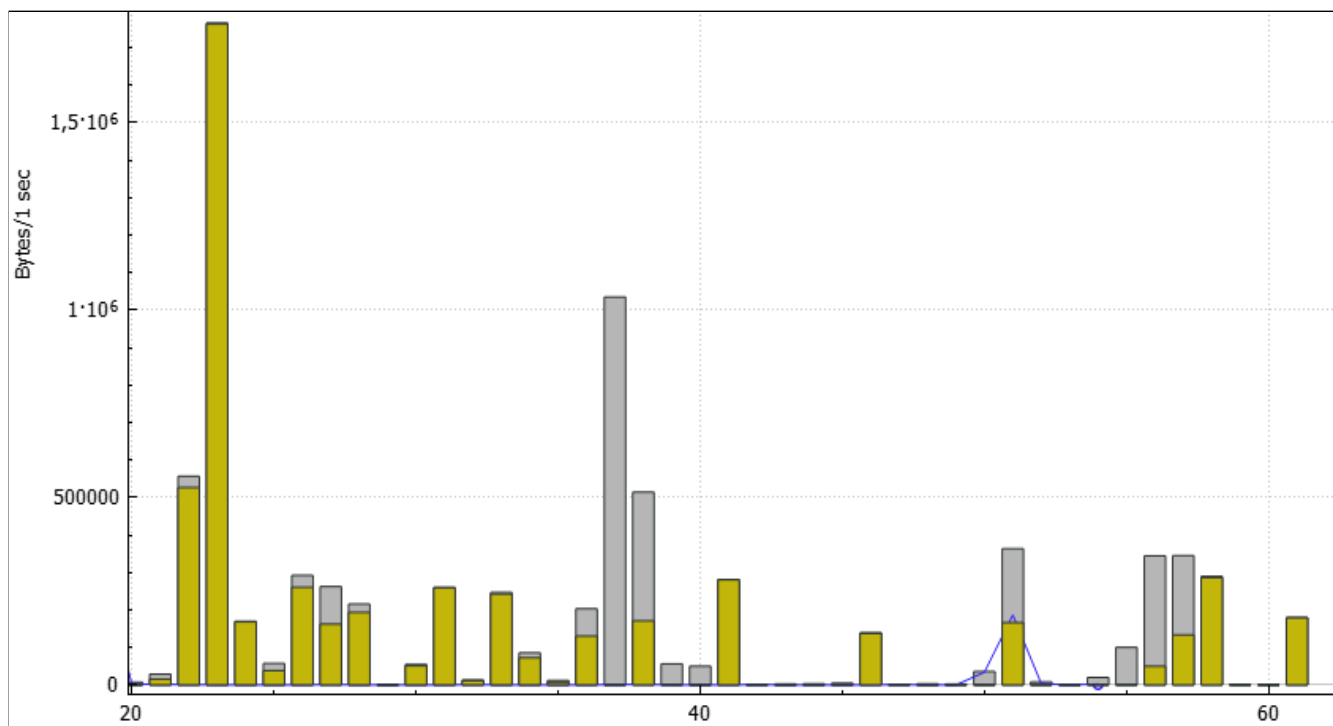


Crediamo che il resto della cattura riguardi principalmente la visita di siti bancari.

Prepariamo un altro filtro approssimativo per individuare tutti quei pacchetti che riteniamo associabili a banche; il filtro risultante è:

- *ip.host contains "bank" || ip.host contains "chase" || ip.host contains "morgan" || ip.host contains "ecgib" || ip.host contains "cbb"*

Aggiungiamo quindi una **stacked bar giallastra** al grafico realizzato poco fa per vedere l'incidenza del traffico bancario rispetto a quello complessivo dal ventesimo secondo in poi; anche il riepilogo grafico suggerisce una certa consistenza riguardo l'interazione dell'utente con host relativi a banche.



Scorrendo i pacchetti catturati, notiamo inoltre la presenza di comunicazioni non sicure che potrebbero spiegare la quantità di dati HTTP (in chiaro) indicata nel riepilogo.

56.511099	192.168.1.70	www.ccb.com	HTTP	GET /en/commen_new/images/ico_li_01.gif HTTP/1.1
56.516003	www.ccb.com	192.168.1.70	HTTP	HTTP/1.1 200 OK (GIF89a)
56.516424	192.168.1.70	www.ccb.com	HTTP	GET /en/commen_new/images/product_service_content.jpg HTTP/1.1
56.570560	ir.finet.com.cn	192.168.1.70	HTTP	HTTP/1.1 200 OK (text/html)
56.570839	192.168.1.70	ir.finet.com.cn	HTTP	GET /ccb/js/lib/jquery.query-2.1.7.js HTTP/1.1
56.571378	ir.finet.com.cn	192.168.1.70	HTTP	HTTP/1.1 200 OK (text/html)
56.571583	192.168.1.70	ir.finet.com.cn	HTTP	GET /ccb/lang/zh-CN.js HTTP/1.1
56.605050	192.168.1.70	ir.finet.com.cn	HTTP	GET /ccb/lang/zh-HK.js HTTP/1.1
56.609280	192.168.1.70	ir.finet.com.cn	HTTP	GET /ccb/lang/en-US.js HTTP/1.1
56.668573	www.ccb.com	192.168.1.70	HTTP	HTTP/1.1 200 OK (GIF89a)
56.669097	192.168.1.70	www.ccb.com	HTTP	GET /en/commen_new/images/block_438px_top.gif HTTP/1.1

In conclusione, crediamo che l'utente abbia visitato principalmente siti bancari. Notiamo, inoltre, che alcuni host hanno fornito contenuti in chiaro.

TLS: Handshake

Particolare attenzione è stata data alla fase di handshake prevista da comunicazioni che sfruttano il protocollo TLS. In questo capitolo cerchiamo di evidenziare i passaggi più interessanti rispetto al nostro caso.

Studiando in particolare l'[RFC](#) relativo alla **versione 1.2** (utilizzata per tutte le comunicazioni TLS rilevanti alle domande poste), si osservano vari dettagli utili a comprendere le comunicazioni oggetto di studio.

La struttura di un handshake TLS segue indicativamente queste fasi:

- **Client Hello:** il client inizia la conversazione mandando un messaggio contenente le informazioni utili alla negoziazione dei parametri di sicurezza (cosa il client supporta); in particolare vengono elencate per preferenza le **cipher suite supportate**, la versione massima TLS supportata e un numero random (adoperato in tale versione anche per rendere più difficili attacchi di tipo [replay](#); [riferimento](#)) che verrà combinato con altri valori per comporre la chiave di cifratura.
- **Server Hello:** supponendo che il server possa instaurare una conversazione con il client (in caso contrario viene mandato un TLS alert di errore rifiutando al client la possibilità di instaurare una conversazione; [riferimento](#)), questi risponde con un ulteriore numero random, un messaggio di hello informando il client dell'esito della negoziazione (**cipher suite utilizzata**, metodo di compressione, ...) e tentando di autenticarsi includendo una gerarchia di certificati firmati e una chiave pubblica; per verificare la validità di un certificato, il client potrebbe non interrogare direttamente una CA ma ricevere dal Server l'esito di una richiesta OCSP (Online Certificate Status Protocol) datato e firmato da una Certificate Authority fidata ad entrambe le parti ([OCSP Stapling](#)).
- **Client Key Exchange:** ora che il server ha fatto la sua parte (Server Hello Done), il client invia la sua chiave pubblica e indicherà al server che ha tutto il necessario per iniziare una conversazione criptata (Client Change Cipher Spec). Una combinazione delle chiavi pubbliche di entrambi gli interlocutori costituisce la *pre-master key*; combinando quest'ultima con i numeri random generati da entrambi si otterrà la [master key](#), utilizzata per cifrare i messaggi. Ora che entrambe le parti sono in grado di iniziare una comunicazione criptata, il client manda un messaggio di fine handshake che riassume il risultato della negoziazione criptato secondo il metodo accordato ([riferimento](#)).
- **Server Key Exchange:** analogamente al passo precedente, il server invia un Change Cipher Spec e un messaggio di fine criptato. I messaggi di fine sono stati introdotti anche al fine di rendere più difficili alcuni attacchi (i.e. [MIM](#)).

è importante precisare anche che:

- Nel nostro caso, l'**OCSP stapling** avviene ma non per tutti gli handshake: alcuni messaggi TLS incapsulano responsi OCSP ed altri no; l'utente effettua comunicazioni OCSP per verificare alcuni certificati.
- il server potrebbe voler **autenticare il client**: in tal caso l'autenticazione avviene analogamente a quanto succede per l'autenticazione del server; il client, in questa cattura e per ciò che concerne l'attività sui siti bancari, crediamo non si autentichi mai durante l'handshake perché non risulta che questi invii verso l'esterno alcun certificato
- la connessione rimane valida secondo le politiche e la struttura del server; in particolare, potrebbe avvenire un nuovo handshake se passa un certo lasso di tempo dall'ultimo handshake concluso con successo ([Session ticket](#))
- **il fatto che una pagina web acceduta sia considerata "sicura" dal browser** (https, tls) non significa che tutte le risorse acquisite siano in effetti criptate (ricordando che la cattura è stata fatta nel 2017, ecco dei riferimenti: [chrome](#), [firefox](#)); nel nostro caso non si presentano situazioni di questo tipo

Client Hello → us-u.openx.net

Cerchiamo di rispondere ora alla prima richiesta: **individuare quanti pacchetti “Client Hello” sono stati inviati dal client al server us-u.openx.net.**

L'estensione TLS “[Server Name Indication](#)” consente di associare al messaggio di apertura dell'handshake il nome del server che si sta tentando di contattare. Utilizzando il filtro

- `tls.handshake.extensions_server_name == "us-u.openx.net" && tls.handshake.type == 1`

è possibile, intanto, individuare tutti e soli i messaggi di Client Hello che includono la suddetta estensione con campo “us-u.openx.net”.

!tls.handshake.extensions_server_name == "us-u.openx.net" && tls.handshake.type == 1						
No.	Time	Source	Destination	Protocol	Info	SNI
9045	39.124324	192.168.1.70	us-u.openx.net	TLSv1.2	Client Hello	us-u.openx.net
9048	39.125149	192.168.1.70	us-u.openx.net	TLSv1.2	Client Hello	us-u.openx.net

Adesso però, dovremmo controllare le richieste che non includono tale estensione

- `!tls.handshake.extensions_server_name && tls.handshake.type == 1`

e, volendo, tenere anche in considerazione i pacchetti che Wireshark non è riuscito ad analizzare (“Ignored Unknown Packet”): potrebbero essere anch'essi dei Client Hello (o parte di essi).

Inoltre, come documentato nell'[RFC](#), il server_name potrebbe non essere affidabile per identificare il server contattato e quindi andrebbe trattato con cautela; per questa analisi assumeremo che il client abbia specificato coerentemente il server_name.

È presente un solo pacchetto di Client Hello senza estensione SNI; riguardo all'IP di destinazione sembrerebbe che si tratti di McAfee ([WhatIsMyIPAddress](#)), tuttavia non riusciamo ad accertarci della bontà delle informazioni, neanche cercando conferma sfruttando nslookup, Whois o Dig. Interrogiamo allora il servizio [Whois di IANA](#) individuando l'organizzazione che amministra tale indirizzo: [Arin](#). A questo punto proviamo a recuperare informazioni dalla base di dati corrispondente, sfruttando il servizio di ricerca online ([ARIN db lookup](#)). L'esito di questa ricerca supporta il fatto che si tratti di McAfee: nonostante nell'RFC inerente non si evidenziano problemi di sicurezza legati all'utilizzo dell'estensione SNI, sospettiamo che l'assenza di questa nel suddetto pacchetto di hello sia dovuta ad un qualche tipo di misura cautelare del servizio di antivirus ma non approfondiamo oltre.

!tls.handshake.extensions_server_name && tls.handshake.type == 1					
No.	Time	Source	Destination	Protocol	Info
885	9.377522	192.168.1.70	8.21.161.29	TLSv1	Client Hello

Per gli **Ignored Unknown Packet** notiamo che hanno tutti come MAC mittente lo stesso del querier IGMP che avevamo identificato in precedenza (gateway 2Wire) e come indirizzo MAC di destinazione lo stesso associato all'IPv4 192.168.1.70 di conseguenza escludiamo che si tratti di client hello rivolti ad us-u.openx.net. Analogamente avremmo potuto osservare che i nomi risolti degli host sorgente sono sempre relativi ad host esterni.

In entrambi i casi, per filtrare solo questi pacchetti, abbiamo usato il filtro:

- `tls.ignored_unknown_record`

How many TLS Client Hello packets are being sent to us-u.openx.net?

In light of the considerations above, we conclude that 2 Client Hello packets have been sent to us-u.openx.net host (173.241.250.143).

Domain Names

In questo capitolo cerchiamo di capire **quanti nomi di dominio sono associabili a J.P. Morgan**.
Vista la quantità di risultati trovati questi saranno indicati una volta sola in conclusione.

Dopo aver aggiunto una colonna al layout di Wireshark in modo da mostrare ordinatamente i valori per

- `dns.qry.name`

filtriamo la cattura con

- `dns.flags.response == 0 &&`
`(dns.qry.name contains "chase" || dns.qry.name contains "morgan")`

in modo da trovare le **richieste dns** atte a risolvere nomi che contengono una parola chiave riconducibile a J.P.Morgan.

Da queste individuiamo due nomi di dominio; adesso vorremmo cercare anche risposte DNS che abbiano come dato risolto un nome, quindi filtriamo la cattura con

- `dns.flags.response == 1 &&`
`(dns.qry.name contains "chase" || dns.qry.name contains "morgan") &&`
`(dns.resp.type == 5 || dns.resp.type == 39)`

ottenendo risposte DNS con un dato CNAME o [DNAME](#) contenenti parole chiave relative a J.P.Morgan dai quali possiamo prendere alias o nomi canonici associati da aggiungere alla lista.

Oltre alle informazioni relative al DNS potremmo osservare gli SNI relativi a Client Hello in comunicazioni TLS. Dopo aver aggiunto un'ulteriore colonna al layout di Wireshark col fine di mostrare i campi SNI per i tls

- `tls.handshake.extensions_server_name`

filtriamo la cattura con:

- `tls.handshake.extensions_server_name contains "morgan" || tls.handshake.extensions_server_name contains "chase"`

trovando, però, domini già individuati al passo precedente.

Infine, cercando (separatamente) la stringa "chase" e "morgan" nei dettagli dei pacchetti si trovano informazioni interessanti relativamente ad una particolare estensione del TLS ([RFC](#)). Di fatti, filtrando con

- `x509ce.dNSName contains "morgan" || x509ce.dNSName contains "chase"`

è possibile individuare ulteriori nomi di dominio associabili a J.P. Morgan.

Mettendo insieme tutti i risultati trovati eliminando duplicati, possiamo finalmente elencare i nomi di domino (*fully qualified domain names*) correlabili a JPMorgan che abbiamo individuato:

1. `cws-psaas.amer-c2.gslbjpmchase.com`
2. `chaseonline.chase.com`
3. `jpmorganchase.112.2o7.net`
 - a. una nota va spesa in particolare per questo che è l'unico dominio che non appartiene a J.P. Morgan (o JPM & Chase) ma ad Adobe.
4. `www.chase.com`
5. `www.jpmorgan.com`
6. `wwwbcchase.gslb.bankone.com`
7. `chaseonline.amer.gslbjpmchase.com`
8. `app.xign.net`
9. `app-emea.xign.net`
10. `careers.jpmorgan.com`
11. `careersatchase.com`
12. `commercial.jpmorganchase.com`
13. `corporatechallenge.jpmorgan.com`
14. `csr-emea.xign.net`
15. `csr-ro.xign.net`
16. `cws-psaas.jpmorgan.com`
17. `emea.xign.net jpmorgan.ca`
18. `jpmorgan.co.id`
19. `jpmorgan.co.jp`

20. jpmorgan.co.kr
21. jpmorgan.co.uk
22. jpmorgan.com
23. jpmorgan.com.au
24. jpmorgan.com.br
25. jpmorgan.com.hk
26. jpmorgan.com.mx
27. jpmorgan.com.pk
28. jpmorgan.com.tw
29. jpmorgan.com.vn
30. jpmorgan.in
31. jpmorgan.ru
32. jpmorgan.sg
33. jpmorgancc.com
34. jpmorganchasecc.com
35. jpmorganchina.com.cn
36. jpmorgansecurities.com
37. jpmasia.com
38. media.chase.com
39. merchantservices.chase.com
40. ordertopay.jpmorgan.com
41. ordertopay.jpmorganchase.com
42. otp.jpmorgan.com
43. otp-app.jpmorganchase.com
44. otp-app-emea.jpmorganchase.com
45. otp-csc.jpmorganchase.com
46. otp-csc-https.jpmorganchase.com
47. otp-csr-emea.jpmorganchase.com
48. otp-csr-ro.jpmorganchase.com
49. otp-csr-ro.xign.net
50. otp-emea.jpmorganchase.com
51. otp-enrollment.jpmorganchase.com
52. otp-jpm.jpmorganchase.com
53. otp-nocsc.jpmorganchase.com
54. otp-www.jpmorganchase.com
55. query.jpmorgan.com
56. query.jpmorganchase.com
57. www.careersatchase.com
58. www.corporatechallenge.jpmorgan.com
59. www.jpmmc.com.br
60. www.jpmchase.com.br
61. www.jpmorgan.ca
62. www.jpmorgan.co.id
63. www.jpmorgan.co.jp
64. www.jpmorgan.co.kr
65. www.jpmorgan.co.th
66. www.jpmorgan.co.uk
67. www.jpmorgan.com.br
68. www.jpmorgan.com.hk
69. www.jpmorgan.com.mx
70. www.jpmorgan.com.pk
71. www.jpmorgan.com.tw
72. www.jpmorgan.com.vn
73. www.jpmorgan.hk
74. www.jpmorgan.in
75. www.jpmorgan.jp
76. www.jpmorgan.pk

77. www.jpmorgan.ru
78. www.jpmorgan.sg
79. www.jpmorgancc.com
80. www.jpmorganchasecc.com
81. www.jpmorganchina.com.cn
82. www.jpmorgansecurities.com
83. www.merchantservices.chase.com
84. www.xign.net
85. www-2.jpmorgan.com
86. xign.net
87. apply.chase.com
88. banking.chase.com
89. cards.chase.com
90. deposits.chase.com
91. investments.chase.com
92. jpmorgan.chase.com
93. payments.chase.com
94. servicing.chase.com
95. stmts.chase.com
96. ultimaterewards.chase.com

Inoltre, utilizzando il servizio Whois di IANA si ottengono ulteriori informazioni riguardo ai name server dai quali si risolvono le richieste dns relative al dominio JPMORGAN (<https://www.iana.org/whois?q=jpmorgan>):

nserver:	NS1.DNS.NIC.JPMORGAN	156.154.144.92	2610:a1:1071:0:0:0:0:5c
nserver:	NS2.DNS.NIC.JPMORGAN	156.154.145.92	2610:a1:1072:0:0:0:0:5c
nserver:	NS3.DNS.NIC.JPMORGAN	156.154.159.92	2610:a1:1073:0:0:0:0:5c
nserver:	NS4.DNS.NIC.JPMORGAN	156.154.156.92	2610:a1:1074:0:0:0:0:5c
nserver:	NS5.DNS.NIC.JPMORGAN	156.154.157.92	2610:a1:1075:0:0:0:0:5c
nserver:	NS6.DNS.NIC.JPMORGAN	156.154.158.92	2610:a1:1076:0:0:0:0:5c

Volendo, potremmo sfruttare questa ed altre informazioni per tentare di identificare ulteriori domini associati a JPMORGAN; potremmo anche compiere altre ricerche più esaustive sulla cattura per accertarci che non vi siano altri nomi di dominio ma ci limitiamo a ciò che siamo riusciti a trovare.

How many domain names can you associate with J.P. Morgan?

In conclusion, we have identified 96 *fully qualified domain names* (distinct) which can be associated with J.P.Morgan.

Stock Ticker

In questo capitolo cerchiamo di capire **quale sia l'organizzazione che ha sviluppato l'applicazione di monitoraggio di stock.**

Cercando la stringa "stock" fra i dettagli dei pacchetti oppure filtrando con

- *frame contains "stock"*

si visualizzano risultati che contengono la stringa "stock".

Alcuni pacchetti in particolare catturano la nostra attenzione:

- sono delle GET HTTP
- presumibilmente recuperano delle integrazioni Javascript
- sono associate ad una certa *Finet*

121.201.39.215	192.168.1.70	HTTP	928 HTTP/1.1 200 OK (application/javascript)
192.168.1.70	121.201.39.215	HTTP	379 GET /ccb/js/finet.core.js HTTP/1.1
192.168.1.70	121.201.39.215	HTTP	385 GET /ccb/js/finet.app.common.js HTTP/1.1
192.168.1.70	121.201.39.215	HTTP	403 GET /ccb/js/finet.app.stocksticker2.js?v=20110829 HTTP/1.1
121.201.39.215	192.168.1.70	HTTP	908 HTTP/1.1 200 OK (application/javascript)
192.168.1.70	121.201.39.215	HTTP	376 GET /ccb/lang/zh-CN.js HTTP/1.1
121.201.39.215	192.168.1.70	HTTP	889 HTTP/1.1 200 OK (application/javascript)
192.168.1.70	121.201.39.215	HTTP	391 GET /ccb/js/lib/jquery.query-2.1.7.js HTTP/1.1
121.201.39.215	192.168.1.70	HTTP	885 HTTP/1.1 200 OK (application/javascript)

Questi dettagli ci fanno pensare che possa trattarsi dell'applicazione di monitoraggio di stock.

A questo punto, osservando le richieste dns contenenti la stringa "finet"

- *dns.qry.name contains finet*

troviamo risoluzioni per il nome **ir.finet.com.cn** (IPv4: 121.201.39.215)

10629 55.809364	2602:306:cee5:f4d0:91c5:de6c:41...	2602:306:cee5:f4d0::1	DNS	95 Standard query 0xcec4 A ir.finet.com.cn
10630 55.809649	2602:306:cee5:f4d0:91c5:de6c:41...	2602:306:cee5:f4d0::1	DNS	95 Standard query 0x4797 AAAA ir.finet.com.cn
10658 55.833832	192.168.1.70	192.168.1.254	DNS	75 Standard query 0xcec4 A ir.finet.com.cn
10659 55.834726	192.168.1.70	192.168.1.254	DNS	75 Standard query 0x4797 AAAA ir.finet.com.cn
10722 56.038465	2602:306:cee5:f4d0::1	2602:306:cee5:f4d0:91c5:de6c...	DNS	111 Standard query response 0xcec4 A ir.finet.com.cn A 121.201.39.215
10723 56.041725	2602:306:cee5:f4d0::1	2602:306:cee5:f4d0:91c5:de6c...	DNS	140 Standard query response 0x4797 AAAA ir.finet.com.cn SOA ns1.finet.com
10724 56.042848	2602:306:cee5:f4d0:91c5:de6c:41...	2602:306:cee5:f4d0::1	DNS	95 Standard query 0x2a63 A ir.finet.com.cn
10729 56.058095	192.168.1.254	192.168.1.70	DNS	120 Standard query response 0x4797 AAAA ir.finet.com.cn SOA ns1.finet.com
10730 56.063432	192.168.1.254	192.168.1.70	DNS	91 Standard query response 0xcec4 A ir.finet.com.cn A 121.201.39.215
10731 56.067609	192.168.1.70	192.168.1.254	DNS	75 Standard query 0x2a63 A ir.finet.com.cn
10814 56.282970	2602:306:cee5:f4d0::1	2602:306:cee5:f4d0:91c5:de6c...	DNS	111 Standard query response 0x2a63 A ir.finet.com.cn A 121.201.39.215
10815 56.284461	2602:306:cee5:f4d0:91c5:de6c:41...	2602:306:cee5:f4d0::1	DNS	95 Standard query 0x3517 AAAA ir.finet.com.cn
10816 56.309040	192.168.1.70	192.168.1.254	DNS	75 Standard query 0x3517 AAAA ir.finet.com.cn

Tuttavia, anche facendo qualche ricerca sul web, non arriviamo ad esiti conclusivi.

Torniamo allora sui pacchetti HTTP prima individuati ed osservando meglio il loro contenuto crediamo che l'utente abbia acceduto ad un servizio offerto da *Finet* tramite il sito di una banca, la CCB (China Construction Bank).

```
Host: ir.finet.com.cn\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Referer: http://www.ccb.com/en/home/index.html\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
[Full request URI: http://ir.finet.com.cn/ccb/stocksticker_2.html?type=a&lang=en-US]
```

Forse riusciamo a capire quale sia la compagnia riferita sul sito della banca cinese visitandone il sito non protetto:

<http://www.ccb.com/en/newinvestor/index.html>

troviamo (in basso a destra nell'immagine) un'estensione offerta proprio da Finet riguardante stock (Stock Chart).

[Corporate Profile](#)
[Corporate Governance](#)
[Announcements](#)
[Shareholders' Meetings](#)
[Periodic Results Report](#)
[Presentations & Webcasts](#)
[Financial Highlights](#)
[Dividend History](#)
[Analysts Coverage](#)
[FAQ](#)
[Service](#)
[Credit Ratings and Awards](#)
[Investor Calendar](#)
[Green, Social and Sustainability Bond](#)
[Regulatory Regulations on
 Minority Management](#)



About CCB >>

We are a leading commercial bank in China providing a comprehensive range of commercial banking products and services.

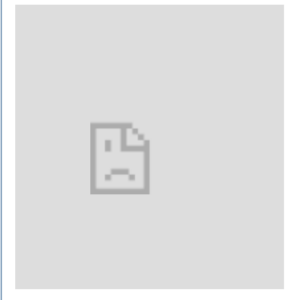
Announcement >>

- > 25 Jun 2021 [Announcement on Retirement of Independent No...](#)
- > 25 Jun 2021 [List of Directors and Their Roles and Functions](#)
- > 25 Jun 2021 [Announcement on the Resolutions of the Meeting...](#)

Stock Chart

A Share

H Share



Information Is Provided by:
[Finet Disclaimer](#) [DISCLAIMER](#)

Riteniamo che questa informazione sia sufficiente a confermare il presunto sviluppatore dell'applicazione - che riteniamo essere Finet - tuttavia, vorremmo ottenere informazioni più precise riguardo a questi. Interagendo con il sito della banca e con il plugin di monitoraggio, non si riesce ad accedere ad alcun contenuto riguardante Finet: il sito non è raggiungibile.

Effettuando una ricerca sull'IP destinatario delle richieste HTTP presentate sopra (sfruttando [IANA](#) e il servizio di [Whois](#) offerto da APNIC) giungiamo all'organizzazione cinese **Guangdong RuiJiang Science and Tech Ltd** ma questa informazione non ci dà nuovi dettagli utili.

Sul sito della HKEX ([wiki](#)) troviamo un report riguardante gli stock della Finet ([link](#)), scoprendo qualche dettaglio in più (aggiornati, fortunatamente, al 2017):

- la Finet aveva come maggiore shareholder (100% della quota) la stessa CCB
- il nome completo dell'organizzazione è "Finet Group Ltd."

Su un presunto report annuale (2013) della CCB ([link](#)), inoltre, troviamo una sorta di immagine "auto-celebrativa" riportante il logo della Finet che ritroviamo al sito <https://ir.finnet.hk/en>.



FINET & QQ.COM

Ranked 2nd in the "Top 100 Hong Kong Listed Companies"

Alla luce di queste informazioni concludiamo che lo sviluppatore dell'applicazione di monitoraggio fosse la Finet Group che, per inciso, non ha più come maggiore shareholder la CCB ([riferimento](#)).

What is the name of the developer of the stock ticker application?

The stock ticker developer's name is **Finet Group Ltd.**

Time To Live

In questo capitolo cerchiamo di capire **quale sia il Time To Live DNS più lungo**.

Per individuare il Time To Live più alto designato nei pacchetti DNS catturati, considerando che in una risposta DNS possono essere indicati più responsi e conseguentemente più TTL (e che quindi un ordinamento di colonna *dns.resp.ttl* potrebbe non essere informativo) procediamo semplicemente con filtri TTL via via più stringenti per escludere risposte dns con tempi più piccoli fino a rimanere con il (o i) pacchetti di risposta DNS con TTL più lungo. Filtrando con:

- *dns.resp.ttl > 100000*

otteniamo un solo pacchetto DNS che presenta la risposta con Time To Live più lungo (149927 secondi).

dns.resp.ttl > 100000	
Protocol	Info
DNS	Standard query response 0x4014 A apis.google.com CNAME plus.l.google.com A 172.217.5.78

▼ **apis.google.com:** type CNAME, class IN, cname plus.l.google.com
Name: apis.google.com
Type: CNAME (Canonical NAME for an alias) (5)
Class: IN (0x0001)
Time to live: 149927 (1 day, 17 hours, 38 minutes, 47 seconds)
Data length: 9
CNAME: plus.l.google.com

What is the highest DNS Time to Live value in the trace file?

The highest DNS Time To Live value in the trace file is 149927 seconds (just over a day and a half)

Cipher Suite

In questo capitolo cerchiamo di capire quale Cipher Suite venga scelta dai server della Bank of America.

Come visto sopra ([REC](#)) nella struttura del TLS viene impiegata una Cipher Suite che viene concordata quando si contatta il Server. Per scoprire qual è stata scelta dal Server della Bank Of America vediamo nel pacchetto di risposta Server Hello relativo cosa viene indicato.

Partiamo dal restringere la nostra ricerca con il filtro

- `ssl.handshake.type==2`

per isolare i dettagli relativi ai Server Hello.

Per trovare i pacchetti relativi alla banca è stato aggiunto un controllo sul contenuto del pacchetto

- `frame contains "bank" && frame contains "america"`

La cattura filtrata risulta con 5 risposte di Server con IP differenti.

No.	Time	Length	Source	SrcPort
4691	26.289155	1514	171.161.206.100	443
4948	26.874066	1514	171.161.198.200	443
5114	27.122387	1514	52.203.103.33	443
5117	27.123199	1514	52.86.105.41	443
5182	27.158927	1514	34.201.177.47	443
5187	27.160644	1514	34.201.177.47	443
5191	27.162195	1514	52.86.105.41	443
5197	27.166229	1514	52.203.103.33	443

Per controllare i vari indirizzi è possibile utilizzare il servizio WhoIs di IANA oppure abilitare direttamente da Wireshark la traduzione degli IP tramite *Edit>Preferences>Name Resolution > Resolve network addresses*

No.	Time	Length	Source	SrcPort
4691	26.289155	1514	wwwui.ecg1b.bac.com	443
4948	26.874066	1514	secure.ecg1b.bac.com	443
5114	27.122387	1514	boss-bankofamerica-295288654.us-east-1.elb.amazonaws.com	443
5117	27.123199	1514	aero-bankofamerica-1025035548.us-east-1.elb.amazonaws.com	443
5182	27.158927	1514	dull-bankofamerica-1921450868.us-east-1.elb.amazonaws.com	443
5187	27.160644	1514	dull-bankofamerica-1921450868.us-east-1.elb.amazonaws.com	443
5191	27.162195	1514	aero-bankofamerica-1025035548.us-east-1.elb.amazonaws.com	443
5197	27.166229	1514	boss-bankofamerica-295288654.us-east-1.elb.amazonaws.com	443

qui osserviamo che gli indirizzi 52.203.103.33, 52.86.105.41 e 34.201.177.47 hanno a che fare con Amazon infatti possiamo verificare sul [sito](#) che amazonaws corrisponde a Amazon Web Service (AWS), un servizio di cloud computing; per gli altri utilizzando [IANA](#) individuiamo l'organizzazione che li amministra, Apnic, dalla quale si evince che entrambi appartengono all'organizzazione Bank of America. All'interno del Secure Sockets Layer troviamo la cipher suite scelta dal server.

- Secure Sockets Layer
 - TLSv1.2 Record Layer: Handshake Protocol: Server Hello
 - Content Type: Handshake (22)
 - Version: TLS 1.2 (0x0303)
 - Length: 85
 - Handshake Protocol: Server Hello
 - Handshake Type: Server Hello (2)
 - Length: 81
 - Version: TLS 1.2 (0x0303)
 - Random: 4b89a8dbdcefe276ff83b4836b1b327839308c5d483648e1...
 - Session ID Length: 32
 - Session ID: 8f27bd2ab82112d78aaa4e8e36e2678a52e42bac8d244624...
 - Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
 - Compression Method: null (0)
 - Extensions Length: 9
 - Extension: renegotiation_info (len=1)
 - Extension: server_name (len=0)

What cipher suite was selected by Bank of America's server?

Bank of America's server selected, for the communication with this client, the Cipher Suite:
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)

1-byte data packets

In questo capitolo cerchiamo di capire **quale sia lo scopo dei pacchetti da 1 byte**

Per vedere a cosa corrispondono i pacchetti di dati lunghi 1 byte iniziamo utilizzando il filtro

- `data.len==1`

Osserviamo che i risultati riguardano esclusivamente pacchetti TCP Keep-Alive quindi, volendo, possiamo anche cambiare il filtro con

- `tcp.analysis.keep_alive`

I Keep-Alive servono a mantenere aperta una connessione TCP, senza che questa venga chiusa per inattività ([RFC](#)). Tali pacchetti sono inviati dall'utente tramite l'indirizzo IPv4 `192.168.1.70` e IPv6 `2602:306:cee5:f4d0:fc22:cbce:cf61:9dc2` per controllare lo stato delle connessioni con i server.

Una volta stabilita la connessione il client si aspetta di avere una conversazione duratura fino alla chiusura del canale, per questo, nel caso in cui non veda più arrivare pacchetti da parte del server, una volta passato un determinato lasso di tempo, invia dei pacchetti (TCP Keep-Alive) di un solo byte per verificare che il server sia ancora in ascolto.

Con Wireshark dopo aver utilizzato il filtro è possibile selezionare un qualsiasi pacchetto e con tasto destro del mouse, *Follow > TCP stream* possiamo vedere la comunicazione relativa. Preso in esempio il primo pacchetto TCP Keep-Alive che troviamo nel file .pcap (No. 1177), notiamo come allo scadere di un timeout interno di circa 10" dall'ultimo pacchetto ricevuto (compreso TCP Keep-Alive ACK) ne viene inviato un altro con lo stesso intento.

No.	Time	Length	Source	SrcPort	Destination	DstPort	Protocol	Info
54	4.090169	66	192.168.1.70	60526	23.72.181.66	80	TCP	60526 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=
55	4.099974	66	23.72.181.66	80	192.168.1.70	60526	TCP	80 → 60526 [SYN, ACK] Seq=0 Ack=1 Win=292
56	4.100102	54	192.168.1.70	60526	23.72.181.66	80	TCP	60526 → 80 [ACK] Seq=1 Ack=1 Win=65536 Le
57	4.100238	347	192.168.1.70	60526	23.72.181.66	80	HTTP	GET /success.txt HTTP/1.1
60	4.111241	60	23.72.181.66	80	192.168.1.70	60526	TCP	80 → 60526 [ACK] Seq=1 Ack=294 Win=30272
61	4.111242	438	23.72.181.66	80	192.168.1.70	60526	HTTP	HTTP/1.1 200 OK (text/plain)
63	4.162843	54	192.168.1.70	60526	23.72.181.66	80	TCP	60526 → 80 [ACK] Seq=294 Ack=385 Win=6528
1177	14.124654	55	192.168.1.70	60526	23.72.181.66	80	TCP	[TCP Keep-Alive] 60526 → 80 [ACK] Seq=293
1178	14.133551	66	23.72.181.66	80	192.168.1.70	60526	TCP	[TCP Keep-Alive ACK] 80 → 60526 [ACK] Seq
4400	24.134347	55	192.168.1.70	60526	23.72.181.66	80	TCP	[TCP Keep-Alive] 60526 → 80 [ACK] Seq=293
4401	24.144435	66	23.72.181.66	80	192.168.1.70	60526	TCP	[TCP Keep-Alive ACK] 80 → 60526 [ACK] Seq
6634	34.144846	55	192.168.1.70	60526	23.72.181.66	80	TCP	[TCP Keep-Alive] 60526 → 80 [ACK] Seq=293
6635	34.153975	66	23.72.181.66	80	192.168.1.70	60526	TCP	[TCP Keep-Alive ACK] 80 → 60526 [ACK] Seq
9604	44.154552	55	192.168.1.70	60526	23.72.181.66	80	TCP	[TCP Keep-Alive] 60526 → 80 [ACK] Seq=293
9606	44.163579	66	23.72.181.66	80	192.168.1.70	60526	TCP	[TCP Keep-Alive ACK] 80 → 60526 [ACK] Seq
10478	54.164185	55	192.168.1.70	60526	23.72.181.66	80	TCP	[TCP Keep-Alive] 60526 → 80 [ACK] Seq=293
10480	54.173381	66	23.72.181.66	80	192.168.1.70	60526	TCP	[TCP Keep-Alive ACK] 80 → 60526 [ACK] Seq

What is the purpose of the 1-byte data packets in this trace file?

Those 1-byte data packets are TCP Keep-Alive packets. Their purpose is to check for dead peers if they were not able to notify in time.

Unencrypted browsing

In questo capitolo cerchiamo di capire **quali browser delle banche permettono una navigazione non criptata**

Per individuare quali browser concedono la navigazione non criptata iniziamo restringendo il campo sui pacchetti inviati dall'indirizzo fisico dell'utente con il filtro

- `eth.src == d4:3d:7e:a6:41:fd`

di questi vogliamo analizzare i pacchetti HTTP escludendo quelli che presentano OCSP ([RFC](#)).

- `http && !ocsp`

Dopo aver abilitato il Name Resolution degli IP, osserviamo che l'utilizzo di HTTP in chiaro è al fine di reindirizzare la navigazione su un sito sicuro HTTPS; tuttavia fa eccezione CCB con finet, servizio ostato dalla CCB, che consente la navigazione in chiaro.

```
GIF89a.....f..q..r.....R.;GET /en/home/uploadfile/productservice/gr.jpg HTTP/1.1
Host: www.ccb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.ccb.com/en/home/index.html
Cookie: BIGipServerccvcc_jt_10.3.198.1_tcp80_web_pool=1344471818.20480.0000
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Tue, 13 Jun 2017 02:04:01 GMT
Server: Apache
Last-Modified: Sat, 26 Mar 2016 17:54:08 GMT
ETag: "8811e8-750b-52ef75dbcb000"
Accept-Ranges: bytes
Content-Length: 29963
Cache-Control: max-age=604800
Expires: Tue, 20 Jun 2017 02:04:01 GMT
Keep-Alive: timeout=5, max=61
Connection: Keep-Alive
Content-Type: image/jpeg
```

```
.....Exif..II*.....Ducky.....d.....+http://ns.adobe.com/xap/1.0/.<?xpacket begin="..."
id="W5M0MpCehiHzreSzNTczkc9d"?> <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.3-c011
66.145661, 2012/02/06-14:56:27" > <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"> <rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/" xmlns:xmpMM="http://
ns.adobe.com/xap/1.0/mm/" xmlns:stRef="http://ns.adobe.com/xap/1.0/sType/ResourceRef#"
xmp:CreatorTool="Adobe Photoshop CS6 (Windows)" xmpMM:InstanceID="xmp.iid:
8464BA85760811E5B8309039DCDE50AC" xmpMM:DocumentID="xmp.did:8464BA86760811E5B8309039DCDE50AC">
<xmpMM:DerivedFrom stRef:instanceID="xmp.iid:8464BA83760811E5B8309039DCDE50AC"
stRef:documentID="xmp.did:8464BA84760811E5B8309039DCDE50AC"/> </rdf:Description> </rdf:RDF> </x:xmpme
<?xpacket end="r"?>
>....Adobe.d.....
```

```
.....!..1..AQ". a2.q...R#.
..B.3$.brCs...%5&...S...E.6.(.....!1..AQ..aq"...2.....BR#.b3..r....C$....Sc
46.....?...B..
#f.{sK...E.....4.....w.\:s.A.....F.#B.....hB...Y...{.....y...t.....>8..2.s.....
6.D1e.....J:RF...N...}/...R..W..H.bv...^Bq...G.....i.x...B... (.....
```

esempio con 'tcp.stream eq 189' (pacchetti No. 11069 e 11110)

What is the full name of the bank that allows unencrypted browsing to their site?

The full name of the bank that allows unencrypted browsing is the China Construction Bank (CCB)

Websites visited

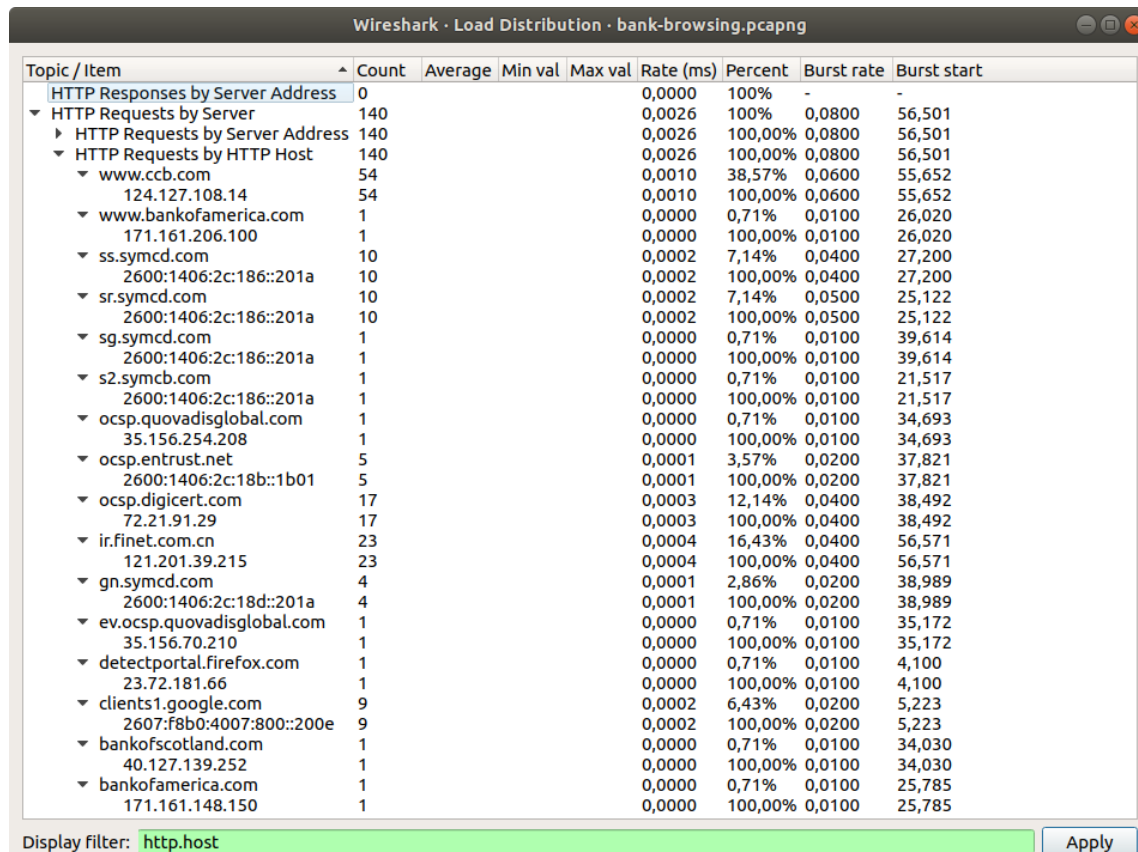
In questo capitolo cerchiamo di capire **in quale ordine sono stati visitati i siti nella cattura**

Per vedere quali sono i siti visitati durante la cattura, possiamo intanto controllare nelle richieste HTTP il campo host che contiene il nome del sito che stiamo richiedendo.

Utilizzando Wireshark è possibile andare su *Statistics>HTTP>Load Distribution* e, usando come filtro

- *http.host*

sotto *HTTP Request by HTTP host* è possibile vedere il traffico dati relativo ai siti visitati.



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
HTTP Responses by Server Address	0				0,0000	100%	-	-
HTTP Requests by Server	140	0,0026			100%	0,0800	56,501	
HTTP Requests by Server Address	140	0,0026			100,00%	0,0800	56,501	
HTTP Requests by HTTP Host	140	0,0026			100,00%	0,0800	56,501	
www.ccb.com	54	0,0010			38,57%	0,0600	55,652	
124.127.108.14	54	0,0010			100,00%	0,0600	55,652	
www.bankofamerica.com	1	0,0000			0,71%	0,0100	26,020	
171.161.206.100	1	0,0000			100,00%	0,0100	26,020	
ss.symcd.com	10	0,0002			7,14%	0,0400	27,200	
2600:1406:2c:186::201a	10	0,0002			100,00%	0,0400	27,200	
sr.symcd.com	10	0,0002			7,14%	0,0500	25,122	
2600:1406:2c:186::201a	10	0,0002			100,00%	0,0500	25,122	
sg.symcd.com	1	0,0000			0,71%	0,0100	39,614	
2600:1406:2c:186::201a	1	0,0000			100,00%	0,0100	39,614	
s2.symcb.com	1	0,0000			0,71%	0,0100	21,517	
2600:1406:2c:186::201a	1	0,0000			100,00%	0,0100	21,517	
ocsp.quovadisglobal.com	1	0,0000			0,71%	0,0100	34,693	
35.156.254.208	1	0,0000			100,00%	0,0100	34,693	
ocsp.entrust.net	5	0,0001			3,57%	0,0200	37,821	
2600:1406:2c:18b::1b01	5	0,0001			100,00%	0,0200	37,821	
ocsp.digicert.com	17	0,0003			12,14%	0,0400	38,492	
72.21.91.29	17	0,0003			100,00%	0,0400	38,492	
ir.finet.com.cn	23	0,0004			16,43%	0,0400	56,571	
121.201.39.215	23	0,0004			100,00%	0,0400	56,571	
gn.symcd.com	4	0,0001			2,86%	0,0200	38,989	
2600:1406:2c:18d::201a	4	0,0001			100,00%	0,0200	38,989	
ev.ocsp.quovadisglobal.com	1	0,0000			0,71%	0,0100	35,172	
35.156.70.210	1	0,0000			100,00%	0,0100	35,172	
detectportal.firefox.com	1	0,0000			0,71%	0,0100	4,100	
23.72.181.66	1	0,0000			100,00%	0,0100	4,100	
clients1.google.com	9	0,0002			6,43%	0,0200	5,223	
2607:f8b0:4007:800::200e	9	0,0002			100,00%	0,0200	5,223	
bankofscotland.com	1	0,0000			0,71%	0,0100	34,030	
40.127.139.252	1	0,0000			100,00%	0,0100	34,030	
bankofamerica.com	1	0,0000			0,71%	0,0100	25,785	
171.161.148.150	1	0,0000			100,00%	0,0100	25,785	

Per controllare la cronologia delle visite è possibile utilizzare lo stesso filtro in wireshark e ordinare temporalmente i pacchetti, i dati relativi al sito si troveranno nel campo *Hypertext Transfer Protocol > Host*. Tuttavia ciò ci permette di controllare solo il traffico HTTP (porta 80) senza considerare quello HTTPS (porta 443). **Una soluzione parziale** può essere quella di seguire le richieste DNS isolando opportunamente quelle relative a siti bancari; per fare ciò, utilizziamo il filtro

- *dns.qry.name*

isoliamo le richieste d'interesse costruendo incrementalmente un filtro che includa solo quelle inerenti

- *(dns.qry.name matches "word1" || dns.qry.name matches "word2" ||)*

ed ordiniamo cronologicamente i pacchetti filtrati.

Uno dei problemi di questa soluzione è che non si tiene conto di eventuali casistiche che potrebbero portarci ad un'interpretazione errata (i.e. caching DNS ci porterebbe all'esclusione di siti/indirizzi).

Per ora in base al filtro *http.host* abbiamo una lista dei siti bancari visitati del tipo:

- Bank of America
- Bank of Scotland
- China Construction Bank Corporation

mentre per quanto riguarda la lista in base alle richieste DNS abbiamo:

- JPMorgan Chase & Co
- Bank of America
- Bank of Scotland
- JPMorgan Chase & Co
- China Construction Bank Corporation

Considerando che la cattura comprende un lasso di tempo di un minuto supponiamo che la cache non abbia influito sul nostro controllo nelle richieste DNS. Le liste HTTP e DNS sembrano essere concordi, inoltre durante l'analisi generale non sono stati trovati pacchetti verso altre banche non nominate. Per determinare l'ordine di visita viene preso in considerazione l'ordine della prima visita avvenuta verso uno specifico sito; non ogni comunicazione con lo stesso dominio è necessariamente intenzionale dell'utente, infatti potrebbero risultare scambio di dati volto all'utilizzo del sito.

In what order did the user visit the bank web sites?

User, **mostly** according with DNS request, visited bank websites in this order:

- JPMorgan Chase & Co
- Bank of America
- Bank of Scotland
- JPMorgan Chase & Co
- China Construction Bank Corporation

Considerazioni

Dopo aver finalizzato l'analisi e la stesura della relazione, abbiamo deciso di mettere in luce alcune considerazioni emerse durante questa esperienza pratica.

Gli aspetti che hanno colto maggiormente la nostra attenzione riguardano la sicurezza.

Non ci aspettavamo che una banca (CCB) potesse consentire in effetti navigazione in chiaro sul proprio sito trafficando non solo immagini ma anche codice Javascript di terzi e, inoltre, non ci siamo spiegati le continue richieste ARP che pensiamo siano interessanti da approfondire. Studiando questioni legate al protocollo TLS, anche alla luce di quanto discusso durante il corso, abbiamo riflettuto sul concetto di "sicurezza": il Transport Layer *Security* è stato designato e adoperato proprio al fine di rendere *sicure* comunicazioni sulla rete; per accertarsi della sicurezza si validano certificati "fidandosi" di enti autorevoli. Quando navighiamo in rete su siti HTTPS, l'interfaccia dei browser più popolari ci mostra icone di lucchetti chiusi, spesso contornate da un rassicurante colore verde e accompagnate dalla scritta: "la tua connessione è sicura". Anche ignorando i cosiddetti [Mixed Contents](#), se fosse possibile mettere in circolazione certificati falsi dei quali organizzazioni molto popolari si fidano, allora questi sarebbero facilmente accettati da client ignari. In effetti, non solo crediamo che la questione sia lecita ma anche inerente al traffico studiato: la [CNNIC](#), responsabile *anche* dell'amministrazione dei siti navigati della CCB è stata diffidata da Google proprio perché avrebbe rilasciato certificati falsi ([articolo](#)); ma questo è solo uno di tanti casi: fra gli altri, abbiamo anche il caso analogo dell'olandese [DigiNotar](#) e, uscendo un dal contesto, troviamo anche casi molto più celebri come quello della [Facebook-Cambridge Analytica](#) che, senza neanche il bisogno di falsificare alcunché, dati personali che consideriamo sicuri sarebbero stati trafficati a terzi al fine di fare campagna politica. Approfondendo varie tematiche riguardanti la sicurezza in rete abbiamo avuto modo di ricordare che un'azione malevola non per forza debba lasciare segnali evidenti, magari addirittura automaticamente rilevabili da un applicativo come Wireshark e che un'innocente noncuranza può essere molto pericolosa: questa è certamente una banalità ma riteniamo che sia una banalità che si presta molto bene ad essere ignorata a giudicare dagli innumerevoli esempi che ieri, come oggi, mettono in imbarazzo non solo privati ma anche compagnie, organizzazioni ed enti considerati popolarmente istituzioni di sicurezza e inviolabilità.

Se avessimo approfondito esaustivamente tutti i concetti inerenti alla cattura, ci sarebbero voluti diversi mesi.

Ci siamo trovati a dover gestire, per quanto possibile, la nostra ignoranza in merito a certi argomenti: questo del resto potrebbe capitare anche ad esperti nel confrontarsi (magari in un tempo molto limitato) con nuove tecnologie che specialmente nel nostro campo sono molte e in continua evoluzione. Una lettura attenta degli standard, magari anticipata da qualche articolo divulgativo in rete di semplice comprensione, è stata più che efficace per poter quantomeno capire cosa stavamo trattando ma non abbiamo mai preso posizione su argomenti più avanzati perché avremmo potuto dare interpretazioni sulla base di sensazioni e non di conoscenze. Un esempio a tal riguardo sono le continue richieste ARP: ci è sembrato strano e ribadiamo quanto sarebbe interessante approfondire l'argomento ma non possiamo dire con ragionevole certezza che si tratti di un'azione malevola o di qualche tipo di funzionamento imprevisto perché conosciamo troppo poco sull'argomento.

I [confirmation bias](#) emergono in particolare quando si crea un filtro.

Nel nostro caso, ci siamo resi conto di quanto ciò che pensavamo di trovare ha condizionato inconsapevolmente il modo in cui lo abbiamo cercato, in particolare il *filtro* costruito per cercarlo; questo problema è emerso svariate volte portandoci a conclusioni iniziali affrettate ed errate (i.e.: inizialmente avevamo individuato un solo nome di dominio associabile a JP Morgan). Ci siamo accorti di quanto i confirmation bias siano potenti e possano addirittura inficiare un'intera analisi; come per gli altri casi, si è trattato di risolvere con uno sforzo di consapevolezza in più ricordandosi sempre che un nome, ma anche un indirizzo (i.e. duplicate IP address, ipv4 && ipv6, ...), così come altre caratteristiche non è detto che siano sempre univoci e sufficienti a discriminare una certa entità. In conclusione, senza screditare la potenza di un approccio top-down e l'utilizzo di filtri approssimativi (quindi veloci da realizzare) che hanno certamente il loro ruolo in analisi, abbiamo compreso che quando si scrive un filtro è importante procedere con cautela senza mai dare per scontato, in particolare, che una certa caratteristica sia sufficiente a discriminare una certa entità d'interesse.