



Fingerprinting TCP/IP

Renato Eugenio Maria Marziano

CyberChallenge.IT - Workshop 2024

05/07/2024



In cosa consiste?

- Il **fingerprinting** è un'analisi di traffico **passiva** che riconosce una macchina basandosi esclusivamente sui metadati che caratterizzano il suo traffico
- In particolare **non mi baso su IP** per riconoscere la macchina
- Posso vanificare gli sforzi di un attaccante che maschera il proprio IP, ricostruendo il flusso del suo traffico di rete su più sessioni separate



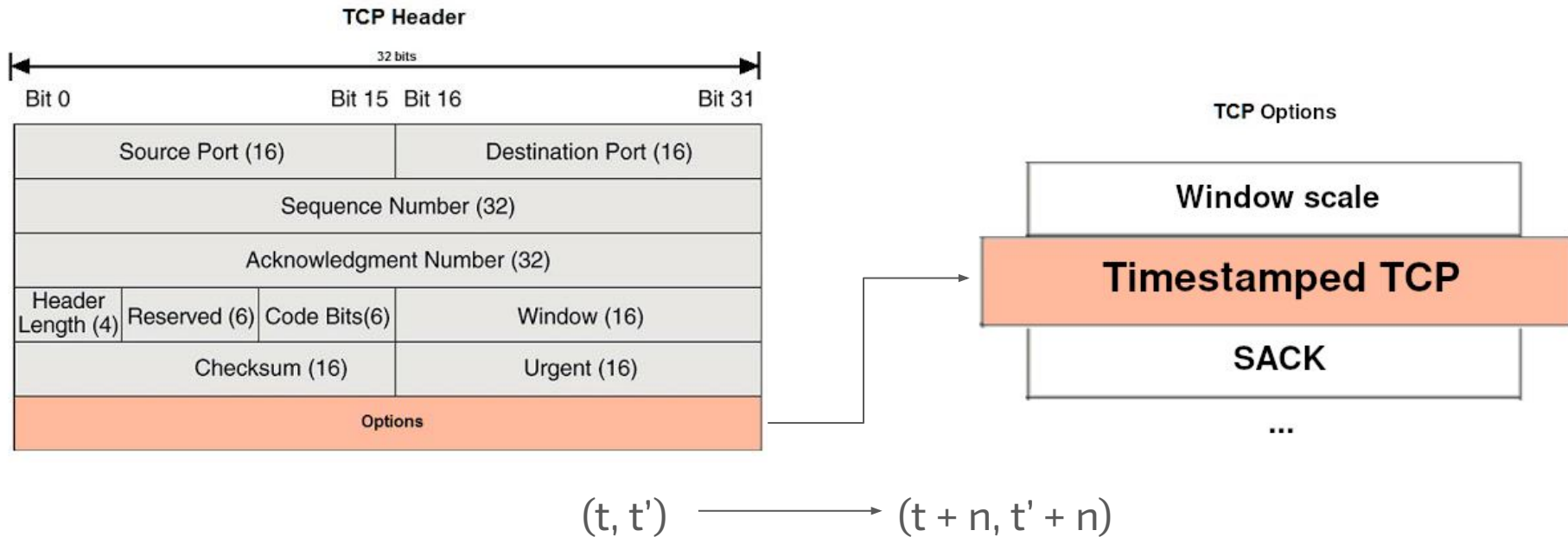
Come costruisco una fingerprint?

- Esempio **reale** : <https://amiunique.org/>
- Voglio un dato sul peer che sia il più possibile:
 - **Identificativo** : il valore del dato dipende dallo stato della macchina del peer
 - **Unico** : è difficile che due peer diversi condividano lo stesso dato
 - **Costante** : il dato non cambia nel tempo



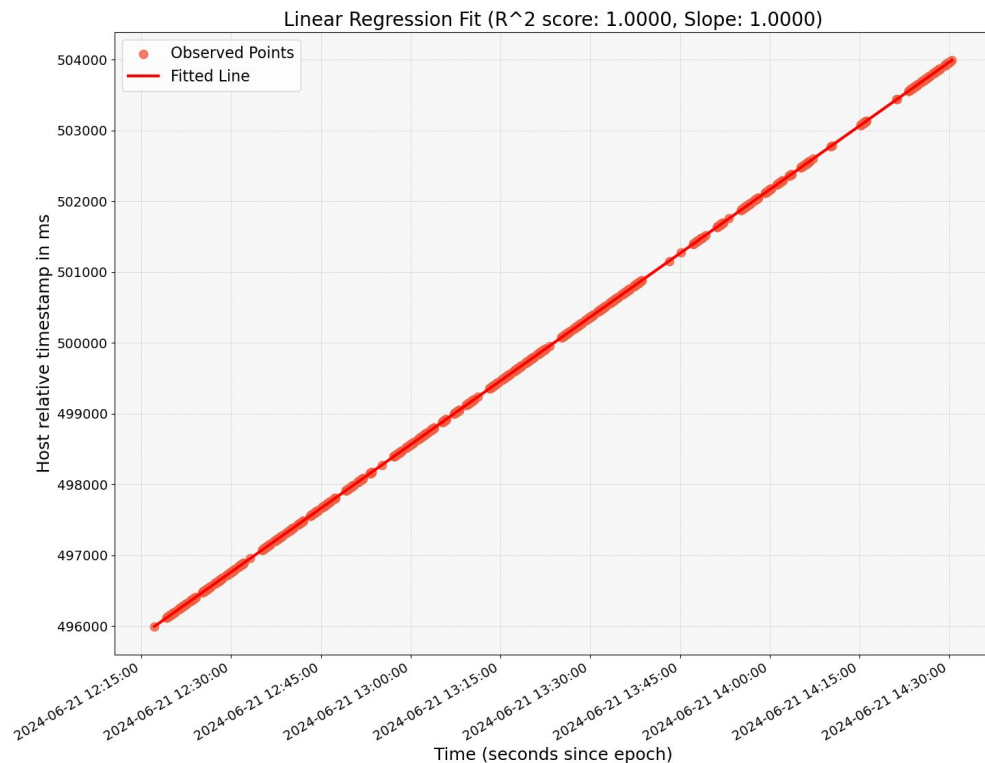
Quali dati ho a mia disposizione?

Come? Quali metadati uso?



Il **timestamp** risulta particolarmente utile

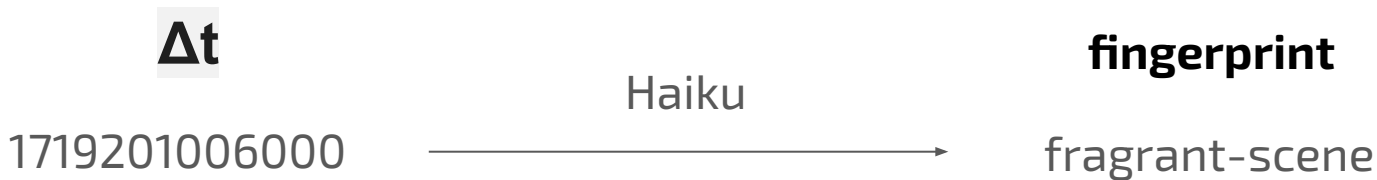
Esempio sperimentale in A/D



- Traffico loggato da una demo di A/D filtrato per una stringa molto *“particolare”* nella payload
- Logicamente il traffico proviene dallo stesso host
- Inoltre ha sempre la stessa **fingerprint**, tutto il traffico giace sulla su una retta

Abbiamo una fingerprint!

- L'orologio del peer e il nostro sono sincronizzati!
 - Possiamo confrontare il timestamp dei pacchetti del peer e il nostro clock.
- Ci basta sapere la loro differenza in un istante del tempo e possiamo **prevedere** i prossimi timestamp di quel peer



Come la sfrutto? →

Tools e come applicarli in A/D

euriclea




- Scritto in casa, si concentra nell'identificare gli **stessi** host nel tempo

```
root@marziano:~/fingerprinter# ./nfqueue -queue 0 -black "fragrant-scene" -host "195.246.230.201"
[3] 79.35.53.70 -> 195.246.230.201 (fragrant-gitter): nsecure-Requests: 1..User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8..Sec-GPC: 1..Accept-Language: en-US,en;q=0.9..Accept-Encoding: gzip, deflate...
FLAG-IN OR SECRET DETECTED : fragrant-scene
[8] 79.35.53.70 -> 195.246.230.201 (fragrant-scene): 1; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8..Accept-Language: en-US,en;q=0.5..Accept-Encoding: gzip, deflate..Connection: keep-alive..Upgrade-Insecure-Requests: 1...
[13] 79.35.53.70 -> 195.246.230.201 (dry-sun): nsecure-Requests: 1..User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8..Sec-GPC: 1..Accept-Language: en-US,en;q=0.7..Accept-Encoding: gzip, deflate...
[18] 79.35.53.70 -> 195.246.230.201 (dry-sun): nsecure-Requests: 1..User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8..Sec-GPC: 1..Accept-Language: en-US,en;q=0.7..Accept-Encoding: gzip, deflate...
```

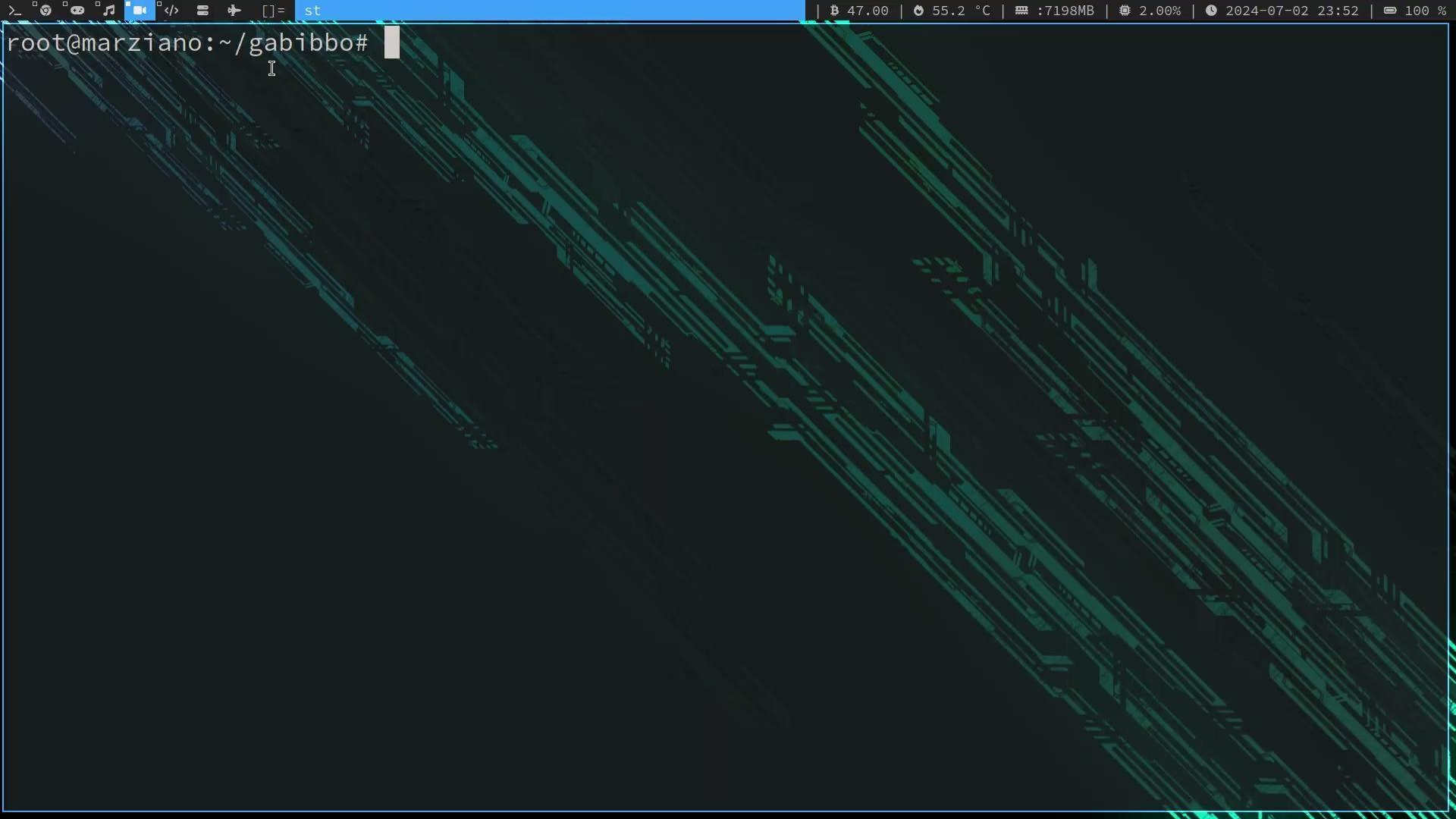
- ❑ Analizza fingerprint, in tempo reale o a posteriori da PCAP
- ❑ Accetta pipe, ad esempio da **PCAP-over-IP** (stay tuned)
- ❑ Permette di **filtrare** il traffico per fingerprint

Analisi a posteriori


- Utile per caratterizzare e poter riconoscere i **pattern** di un attaccante
- Mi permette di filtrare per payload malevole in modo efficiente
 - **WORKFLOW** 
 - Filtro per una payload nota, ne ricavo la fingerprint e poi filtro per la fingerprint!
 - Eventuale machine learning?
- Risulta fondamentale per **testare** il fingerprinting contro un bersaglio

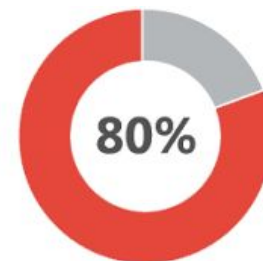
Vediamo alcuni risultati 

root@marziano:~/gabibbo#



Ma funziona veramente in gara?

- È stato necessario del **tuning** per adattarsi al network in locale
- Ma alla fine  ...



% di traffico filtrato (su MB)

```
[renny@archlap ~/doc/euriclea/cmd/extractv2]$ ./extractv2 -F -black "nameless-waterfa
roud-mouse,falling-block,morning-dew,odd-bar,crimson-cherry,snowy-frog,nameless-base,
nt-glade,shy-fire,shiny-unit,wispy-butterfly,nameless-bush,black-cake,green-mouse,emp
ght-cloud,broken-wave,royal-shape,old-salad,shrill-boat,aged-base,lucky-hall,aged-ban
uddy-darkness,gentle-mud,steep-breeze,weathered-cloud,winter-breeze,soft-king,cold-mu
cold-bread,weathered-smoke,icy-limit,young-shadow" -o filtered.pcap log.pcap
[renny@archlap ~/doc/euriclea/cmd/extractv2]$ ls -lh
total 132M
-rwxr-xr-x 1 renny wheel 4.5M Jul  3 17:36 extractv2
-rw-r--r-- 1 renny wheel 7.0K Jul  3 17:36 extractv2.go
-rw-r--r-- 1 renny wheel 19M Jul  6 02:57 filtered.pcap
-rw-r--r-- 1 renny wheel 90M Jul  4 11:56 log.pcap
```

Perché ha funzionato bene in gara?

- **identifico** gli attaccanti (blocco su **tutti** i servizi!) senza far capire che lo sto facendo, né come
 - Altri sistemi sono meno stealth (UserAgent)
 - Si può migliorare, invece di droppare posso ingannare l'attaccante manomettendo le flag (**THE BAD ENDING** 🐈)
- Originalità : gli attaccanti **non** erano pronti
 - **NOBUS** : Nobody but Us
 - Drawback : non abbiamo lavorato ad una difesa pratica, ma c'è sempre un fix (**nfqueue in uscita**)! ⏳ 🛠️
 - Non per molto...

We love free and open source tools

codice sorgente **WIP** ⚠

<https://github.com/drank40/euriclea>



(nuovi loghi disegnati da
umani sono i benvenuti)



- **competizione**, ma anche **collaborazione**



Fonti

- NetResSec :
<https://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting>
- Documentazione di p0f :
 - github.com/p0f/p0f/blob/master/docs/README
- Silence on the Wire: A Field Guide to Passive Reconnaissance and Indirect Attacks, **Michal Zalewski**

Grazie a tutti!

Domande? Idee?