Progress Tambahan

Kelompok 2
Dave Travis - 2201020008
Muhammad Noval - 2201020014
Rizsky Parsadanta R. - 2201020117
Arya Winata - 2201020001

Judul proyek	:	Analisis Serangan Sniffing & Mitigasinya di Wireshark
Tujuan proyek	:	Melengkapi data project
Target proyek	:	- Jumlah paket tertangkap sebelum / sesudah enkripsi, - Latency tambahan akibat TLS, - Persentase keberhasilan penyadapan.

Pengantar:

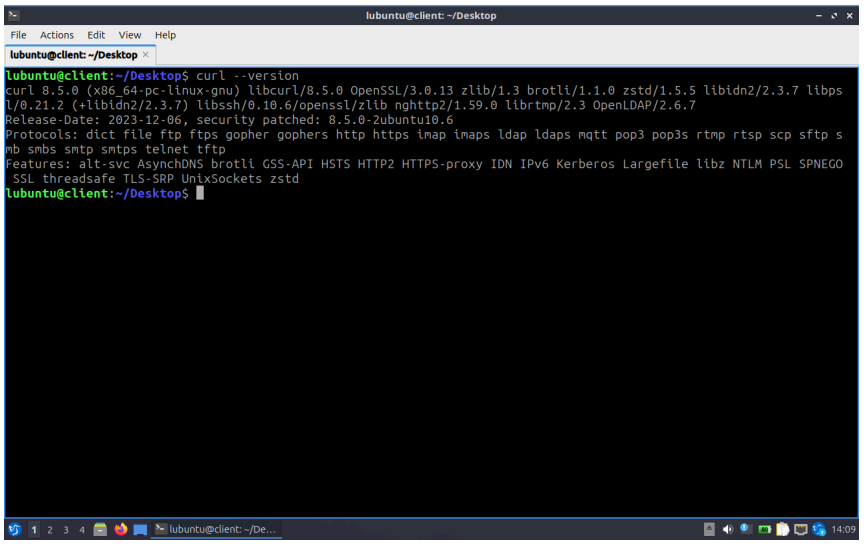
Pada progress kali ini akan difokuskan pada tahapan-tahapan tambahan yang didukung oleh screenshot yang ditangkap selama progress dilakukan. Lampiran-lampiran file capture juga akan disertakan saat pengiriman di link GitHub.

A. Jumlah paket tertangkap sebelum / sesudah enkripsi

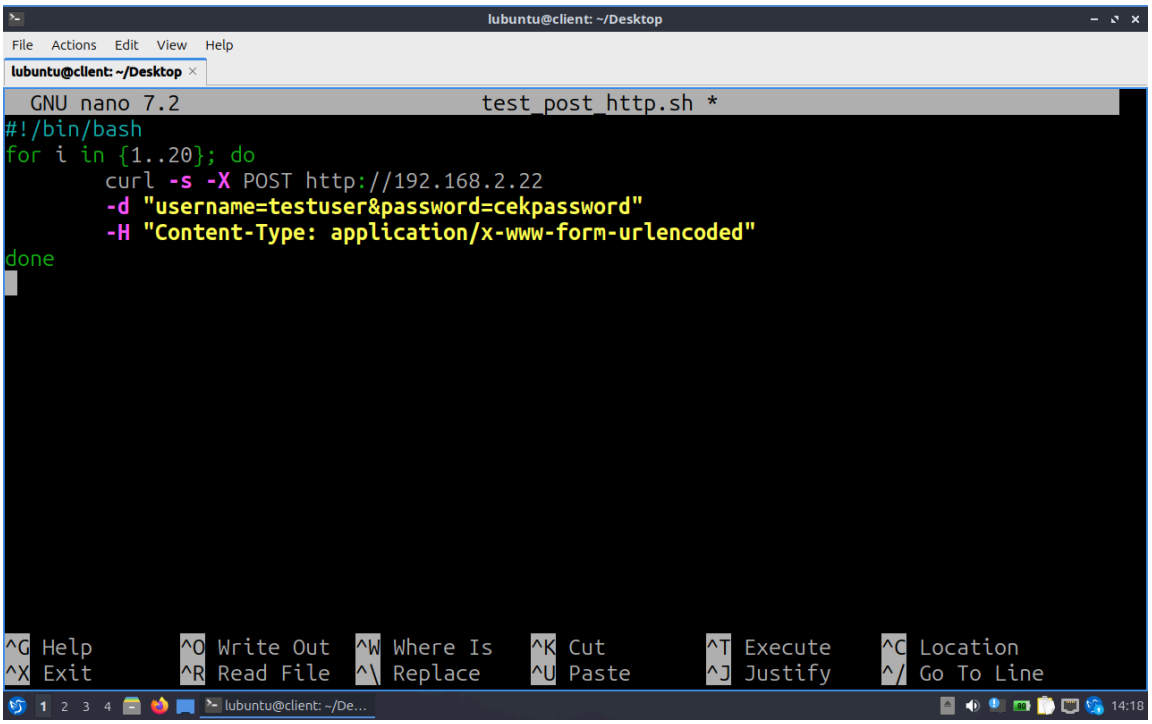
1. Sebelum enkripsi (HTTP)

   Jumlah request POST yang akan dicoba untuk progress ini ialah 20 request per tahapan, dan akan diaplikasikan pada tahapan-tahapan berikutnya demi konsistensi data. Guna memudahkan tahapan request lebih dari satu, kami memutuskan menggunakan looping script sederhana menggunakan curl.

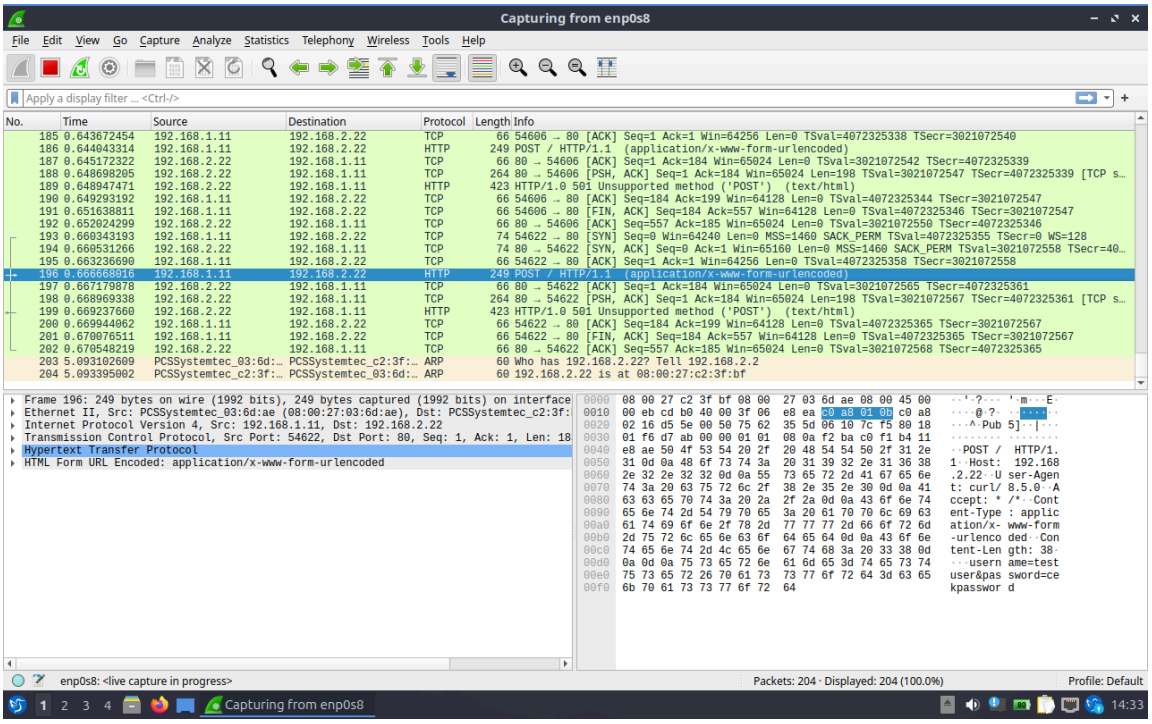
   Untuk mengecek ketersediaan curl pada vm client (karena client yang request POST), gunakan command "*curl --version*"



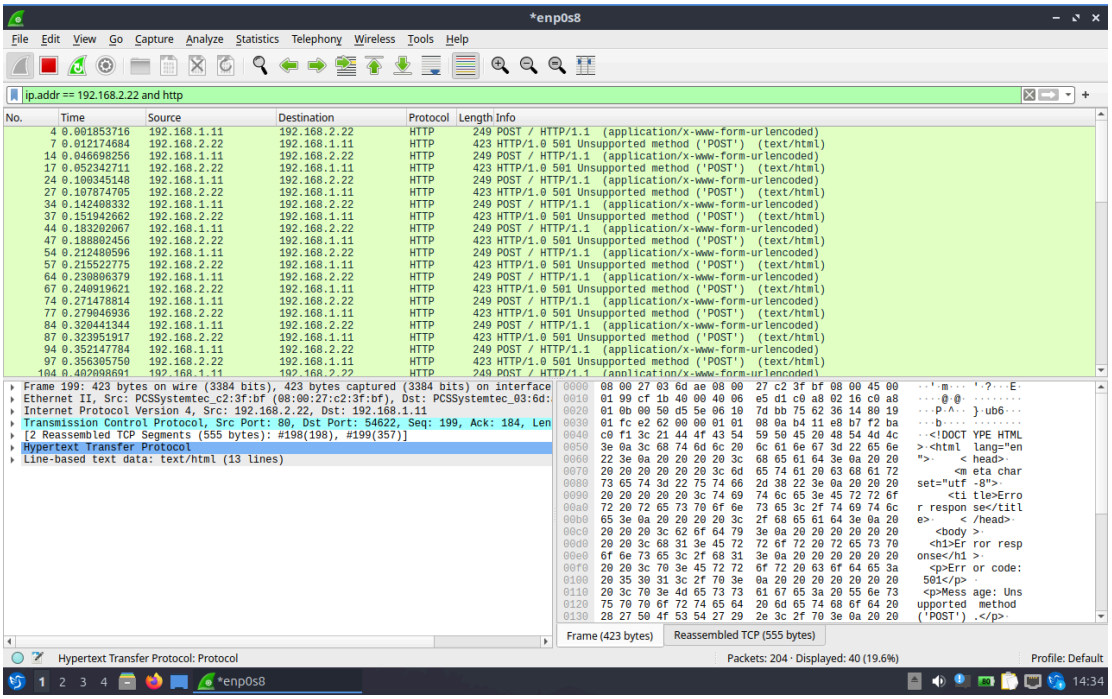
Jika sudah terinstall, maka selanjutnya kita buat script looping POST nya pada file bernama "*test_post_http.sh*" dengan kode sebagai berikut.



Lalu kita simpan dan eksekusi file tersebut, maka request akan terkirim ke vm server, dan jika kita buka vm sniffer selagi membuka WireShark selama request berlangsung, maka akan terlihat seperti ini.



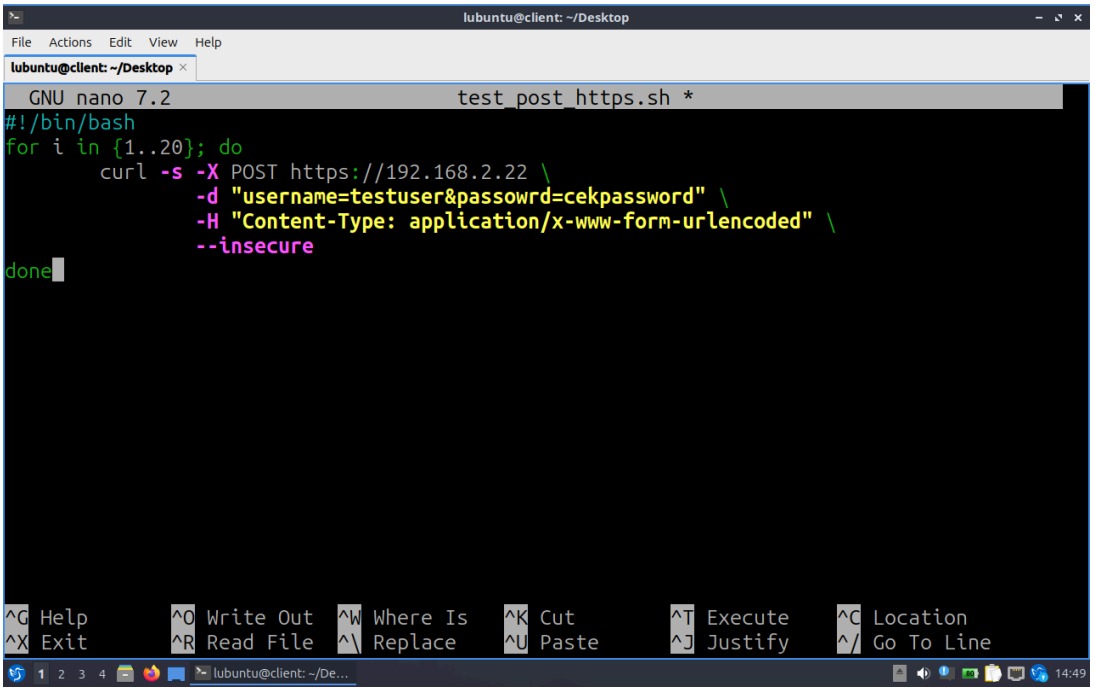
Total *packet* data sudah masuk hasil dari 20 kali request POST, dan masih bercampur-campur dengan packet lainnya, sehingga harus kita pakai filter HTTP dan IP target, yaitu vm server.



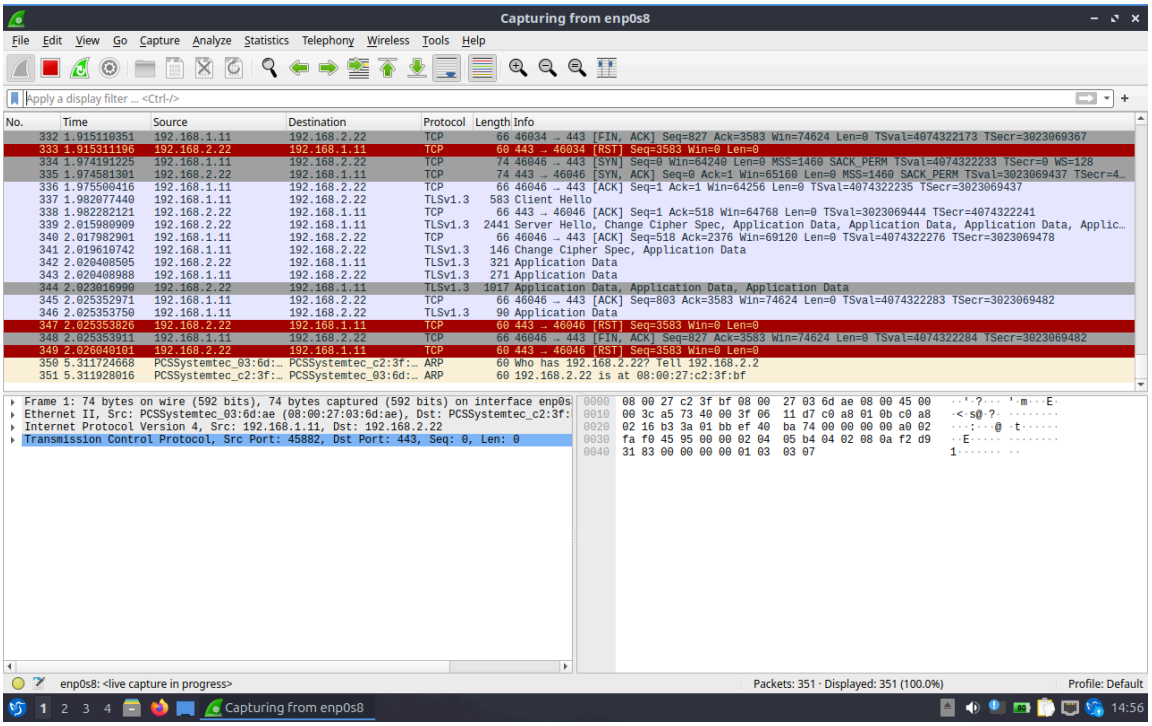
Terlihat pada tulisan di bawah kanan, **total paket** yang tercapture ialah **204 paket**, dan yang eksklusif pada **HTTP (pasca filtering) ialah 40 paket** yang terlampirkan, alias 1**9,6% dari total paket** yang ditampilkan.

2. Pasca enkripsi (HTTPS)

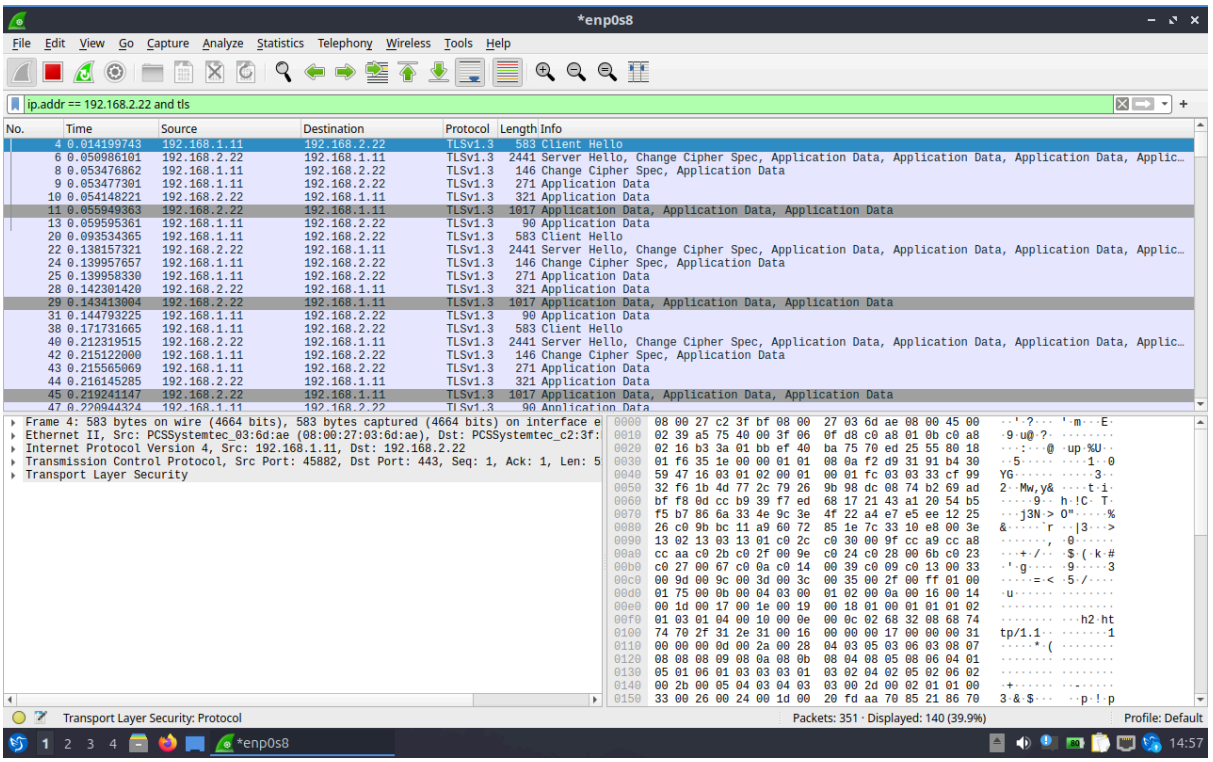
Tahapan juga sama dengan HTTP, di mana request akan dilakukan 20 kali dengan menggunakan script looping sederhana menggunakan curl sebagai berikut.



Lalu kita eksekusi file ini dan request akan kembali terkirimkan ke vm server, sehingga pada vm sniffer dapat tampilan seperti ini untuk total traffic nya.



Karena masih belum terfilter, maka kita implementasikan filter IP target (vm server) dan TLS, sehingga tampilan seperti ini.



Terlihat di bawah kanan, untuk tahapan ini **total paket** yang terkirim ialah **351 paket**, dengan **filter TLS** itu sendiri **ialah 140 paket** yang terkirim, **sekitar 39,9% dari total paket**.

Dapat disimpulkan bahwa paket yang terkirim melalui HTTPS lebih menerima banyak paket karena enkripsi yang dilakukan.

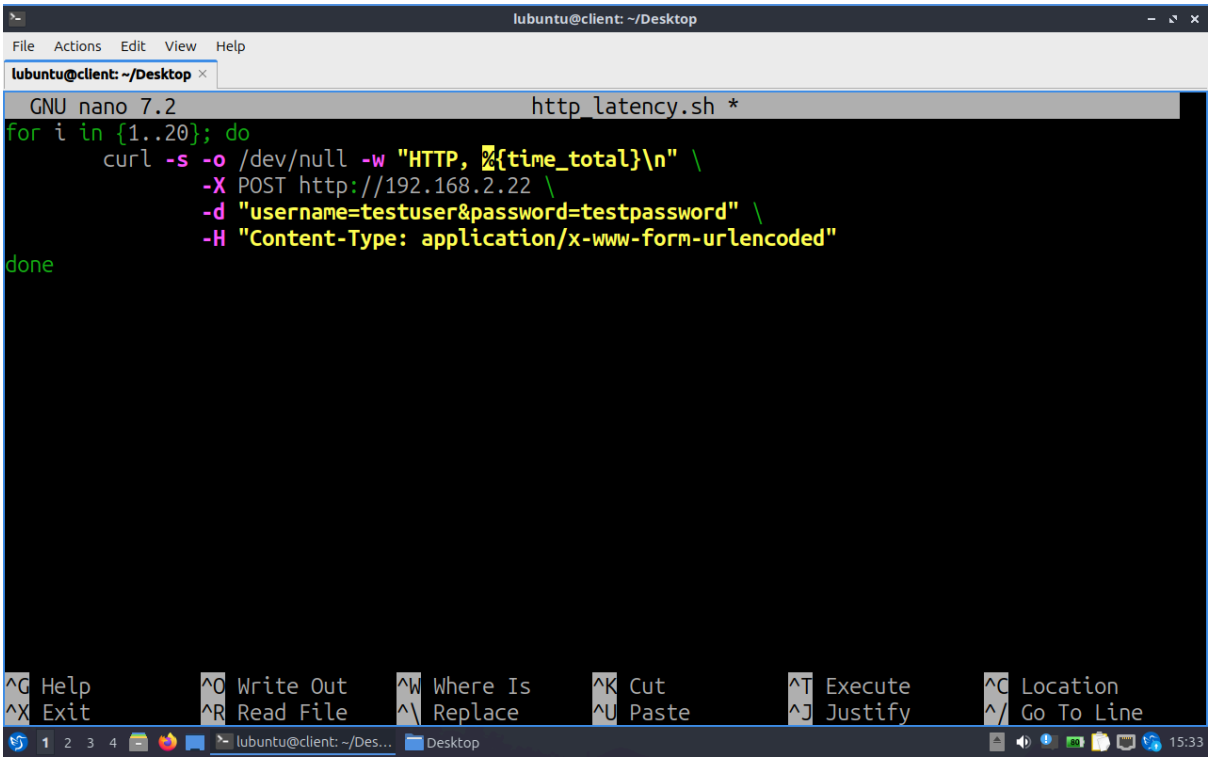
B. Latency tambahan akibat TLS

Pada tahapan ini akan kita uji waktu respons (*latency*) dari HTTP dan HTTPS, dengan membandingkan lama respons tanpa enkripsi dan dengan enkripsi TLS untuk menghitung *overhead* performa yang ditimbulkan oleh enkripsi.

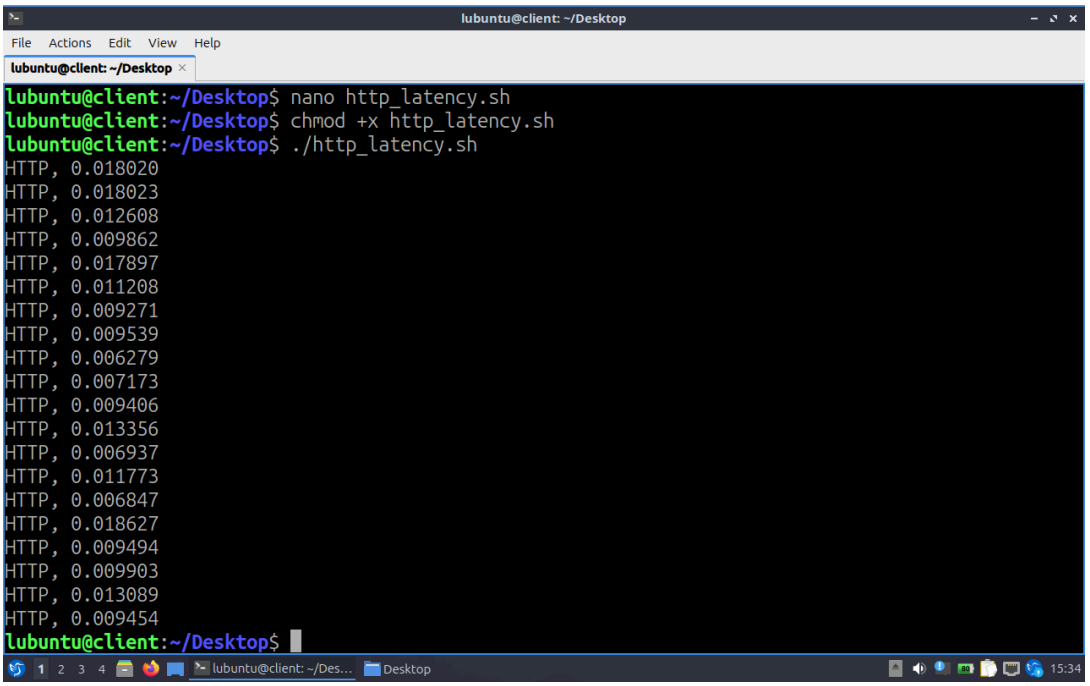
Perhitungan yang akan diukur ialah e*nd-to-end latency*, yaitu waktu kirim request hingga terima respons lengkap dengan satuan milidetik (ms) yang akan dilakukan dengan menggunakan "*curl -w*" (*built-in timing* di curl) sebanyak 20 kali untuk masing-masing HTTP dan HTTPS.

1. Latency HTTP

Percobaan ini juga akan menggunakan script looping otomatis dengan kode sebagai berikut.



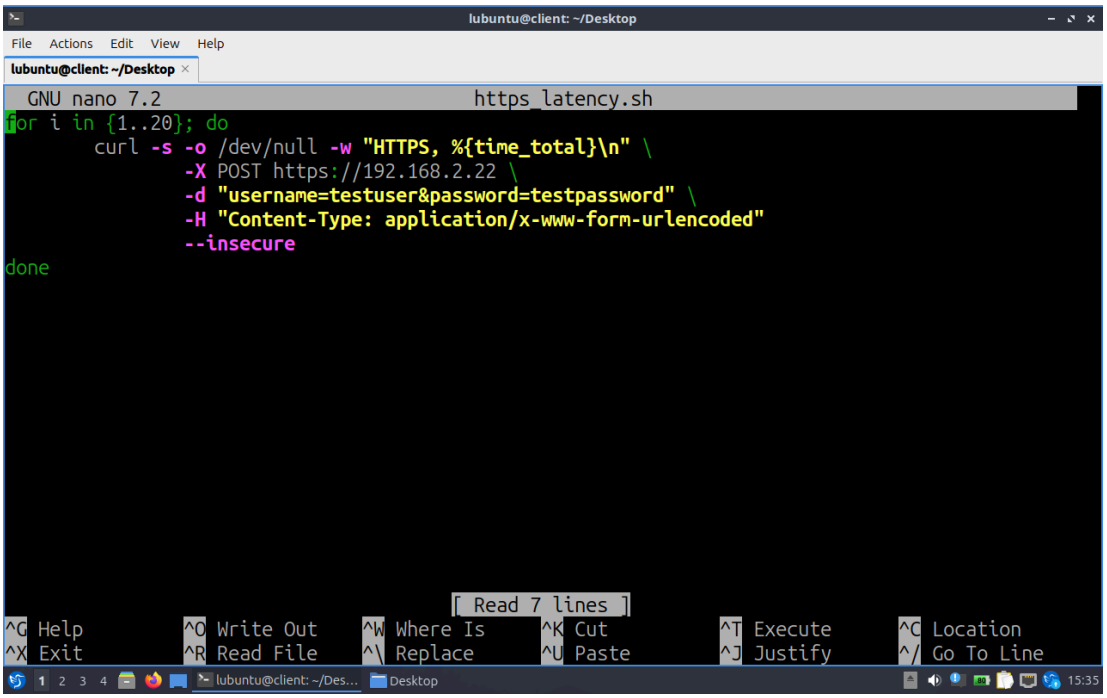
Setelah selesai dikonfigurasi, maka ketika dijalankan akan menampilkan hasil seperti ini.



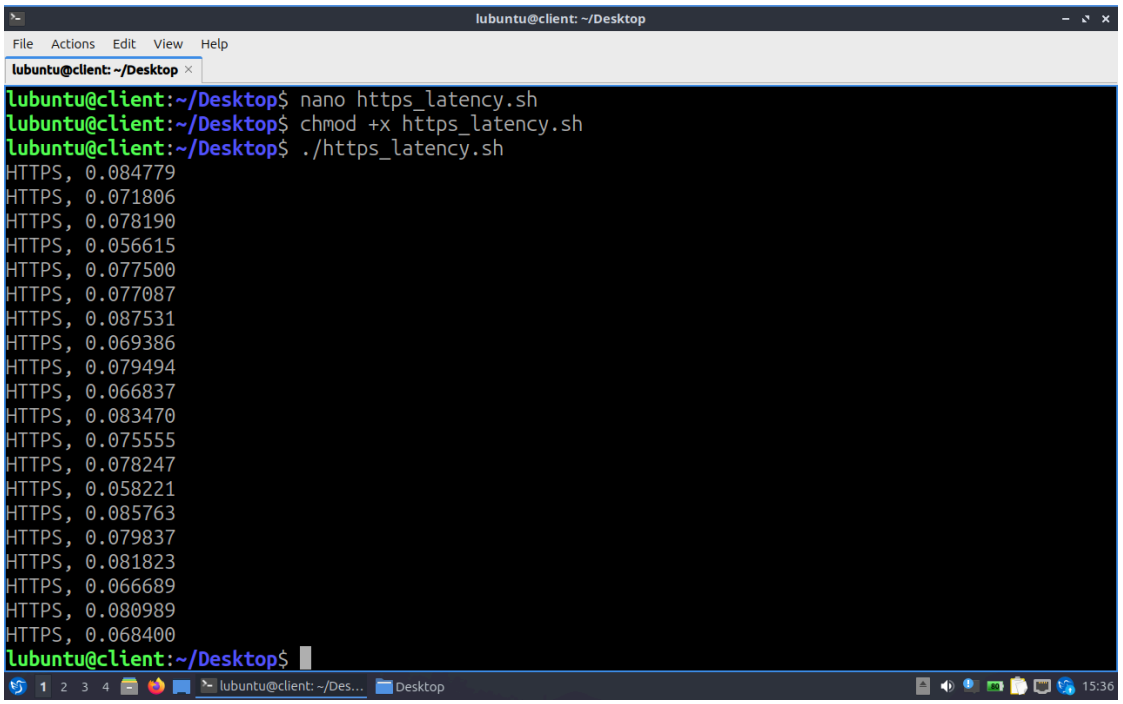
Hasil dari perhitungan latensi ini akan kita simpan agar mempermudah perbandingannya dengan *latency* dari HTTPS.

2. Latency HTTPS

Tahapan yang sama dilakukan untuk HTTPS, dengan script sebagai berikut.

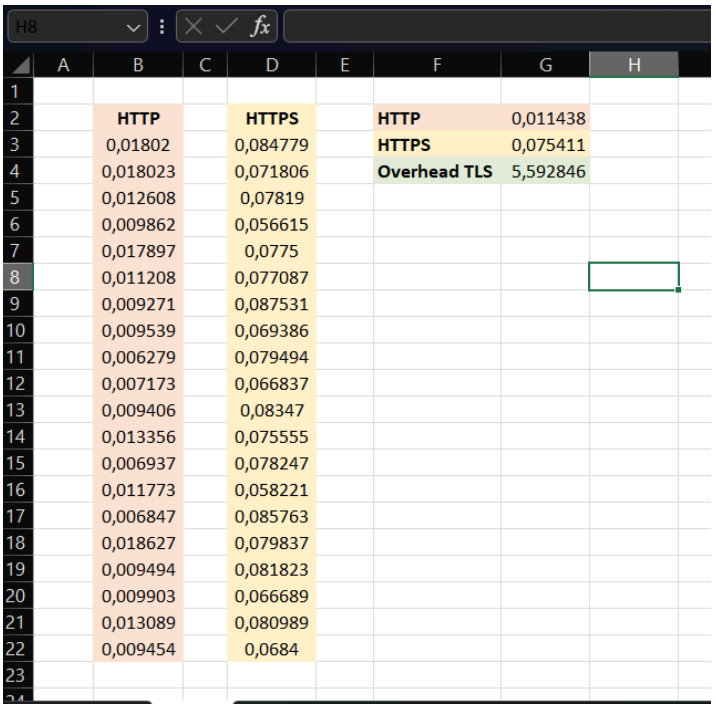


Setelah selesai dikonfigurasi, maka ketika dijalankan akan menampilkan hasil seperti ini.



Setelah semua nya muncul, maka selanjutnya kita akan transfer angka-angka ini ke Excel, sehingga bisa kita hitung dengan rumus excel, yang mana:

- Rata-rata HTTP: =AVERAGE() semua nilai HTTP
- Rata-rata HTTPS: =AVERAGE() semua nilai HTTPS
- Overhead TLS: ((HTTPS_avg - HTTP_avg) / HTTP_avg) * 100%



	A	B	C	D	E	F	G	H
1								
2		**HTTP**		**HTTPS**		**HTTP**	0,011438	
3		0,01802		0,084779		**HTTPS**	0,075411	
4		0,018023		0,071806		**Overhead TLS**	5,592846	
5		0,012608		0,07819				
6		0,009862		0,056615				
7		0,017897		0,0775				
8		0,011208		0,077087				
9		0,009271		0,087531				
10		0,009539		0,069386				
11		0,006279		0,079494				
12		0,007173		0,066837				
13		0,009406		0,08347				
14		0,013356		0,075555				
15		0,006937		0,078247				
16		0,011773		0,058221				
17		0,006847		0,085763				
18		0,018627		0,079837				
19		0,009494		0,081823				
20		0,009903		0,066689				
21		0,013089		0,080989				
22		0,009454		0,0684				
23								

Seperti yang bisa dilihat, bahwa **rata-rata latency HTTP ialah 11 ms** dan r**ata-rata latency HTTPS ialah 75 ms**, dengan **overhead TLS** nya sebesar **~559,38%**, yang artinya **latency HTTPS hampir 6,6 kali lebih lambat** daripada HTTP.

C. Persentase keberhasilan penyadapan

Pada bagian ini kita akan menguji seberapa sering penyerang berhasil mengekstrak password dari lalu lintas yang ditangkap, namun dalam logika simpel, dapat disimpulkan bahwa:

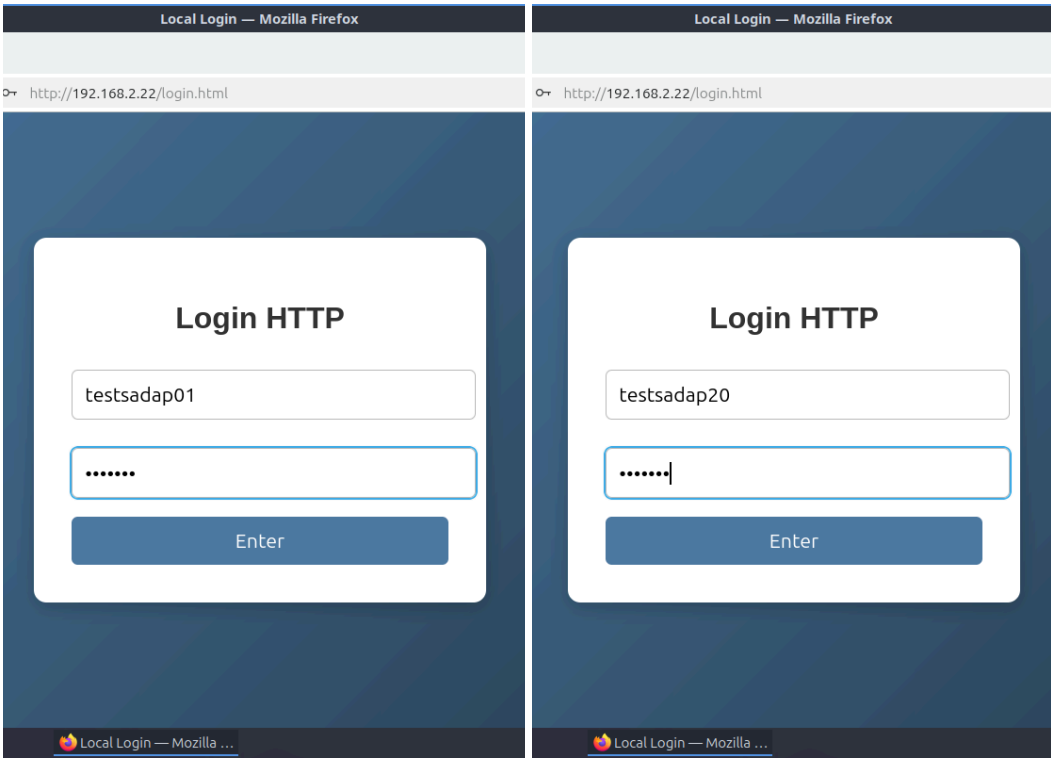
- HTTP yang mana password dalam plain text, seharusnya 100% berhasil.
- HTTPS yang mana password terenkripsi, seharusnya 0% berhasil.

Namun kita akan tetap melakukan eksperimen sehingga kita melihat secara langsung dan memiliki bukti atas kesimpulan kita.

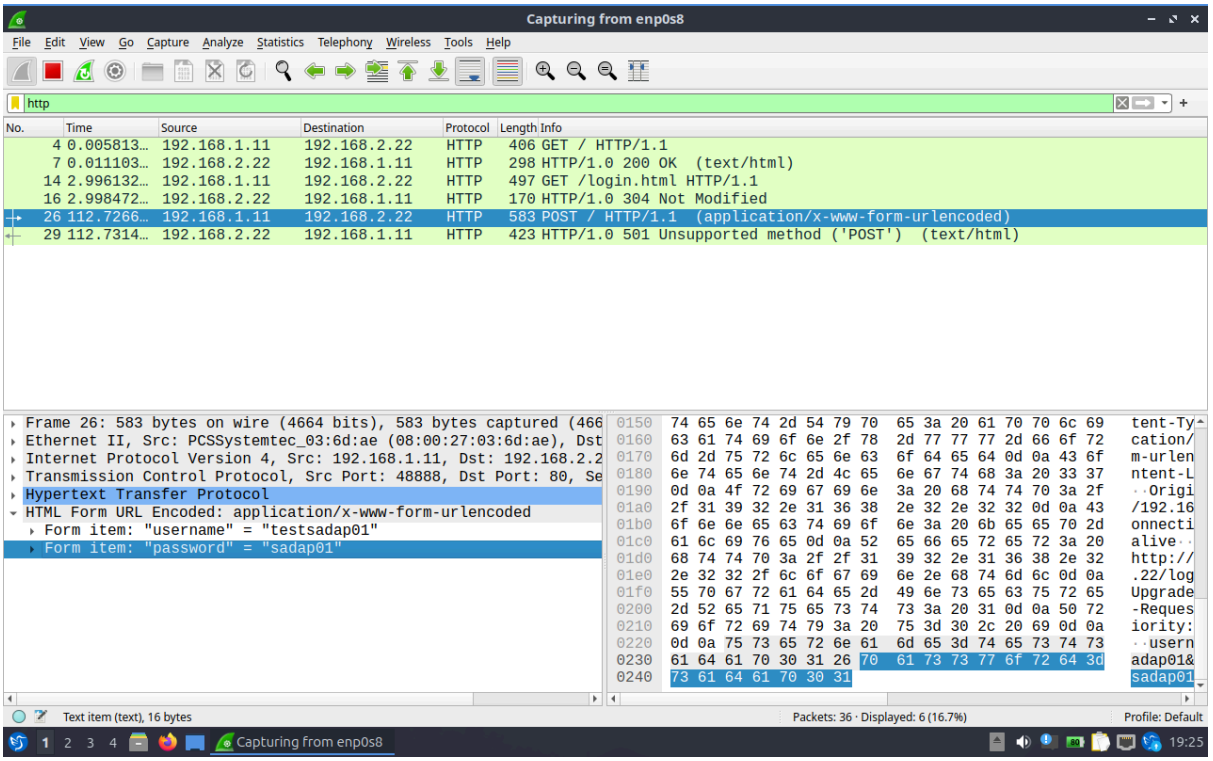
Sama seperti sebelumnya, percobaan ini akan kita loop sebanyak 20 kali untuk tiap protokol, dan untuk memudahkan pengecekan, tahapan ini akan dilakukan secara manual, yaitu request POST melalui portal login untuk tiap-tiap protokol.

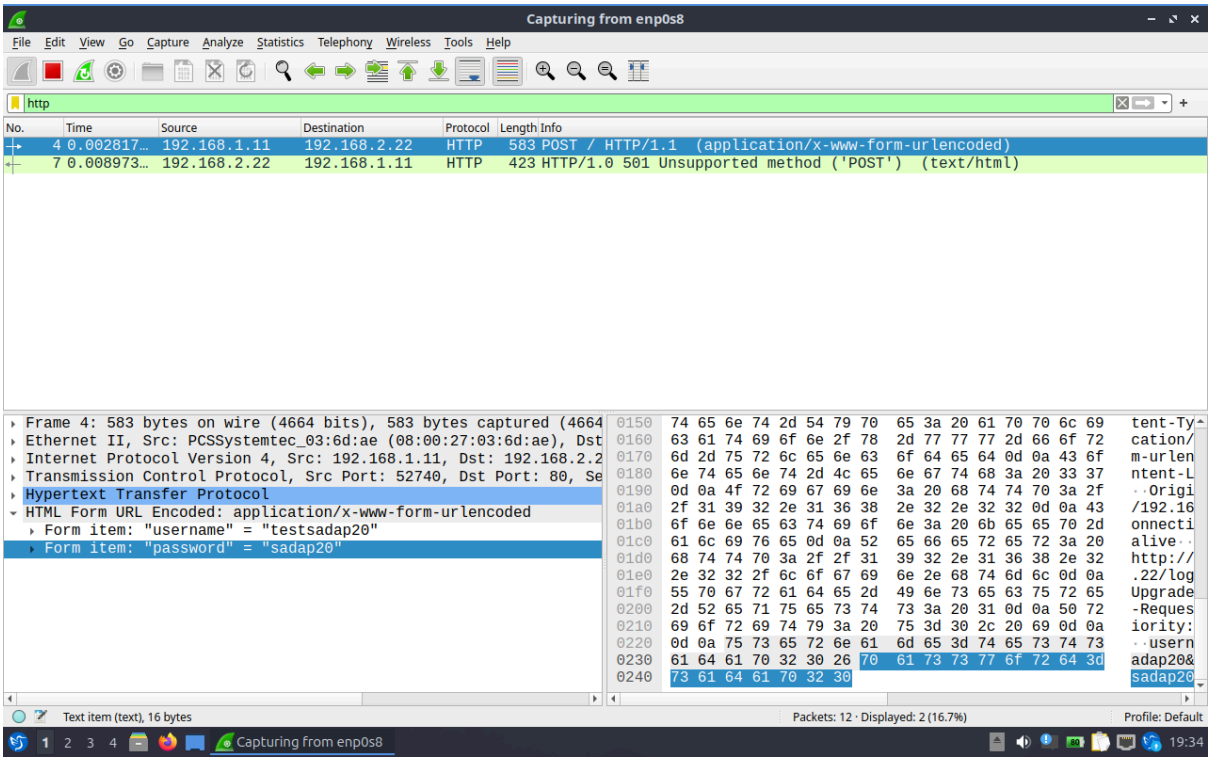
1. HTTP

Data username dan password yang akan digunakan disini akan berubah secara minim guna membandingkan apakah ada perubahan dari tiap tahapan atau tidak, dengan contoh seperti berikut.



Ketika request dikirim, maka dapat terlihat untuk masing-masing request POST HTTP dapat terlihat *plain text* passwordnya.

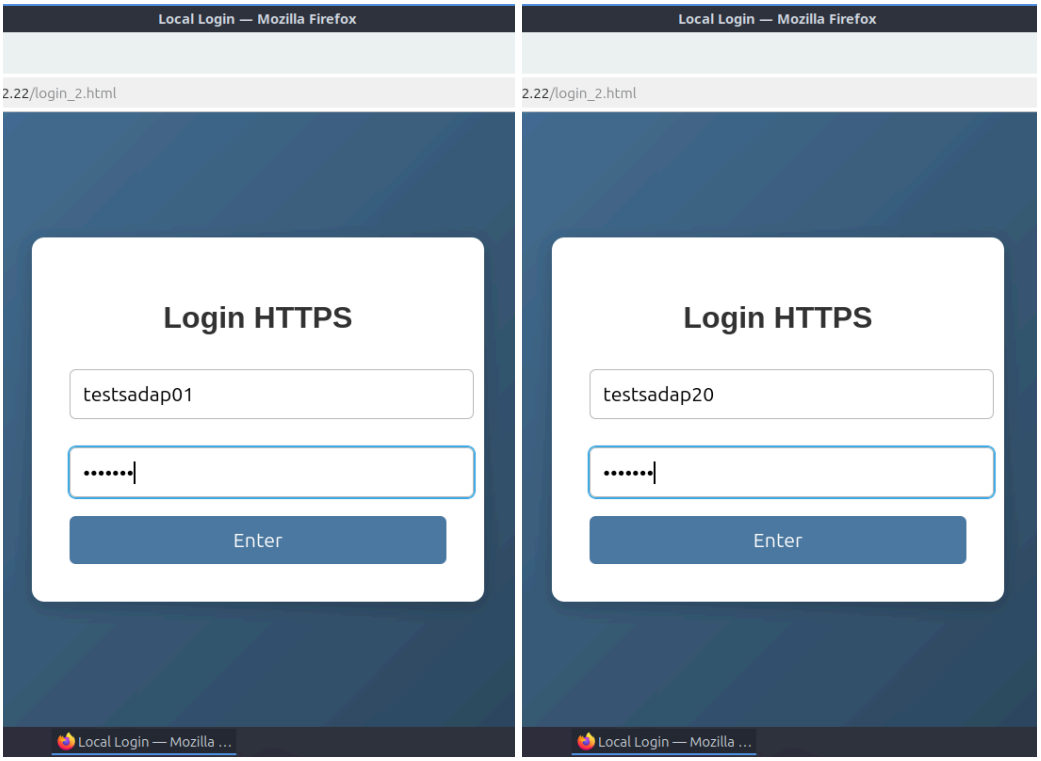




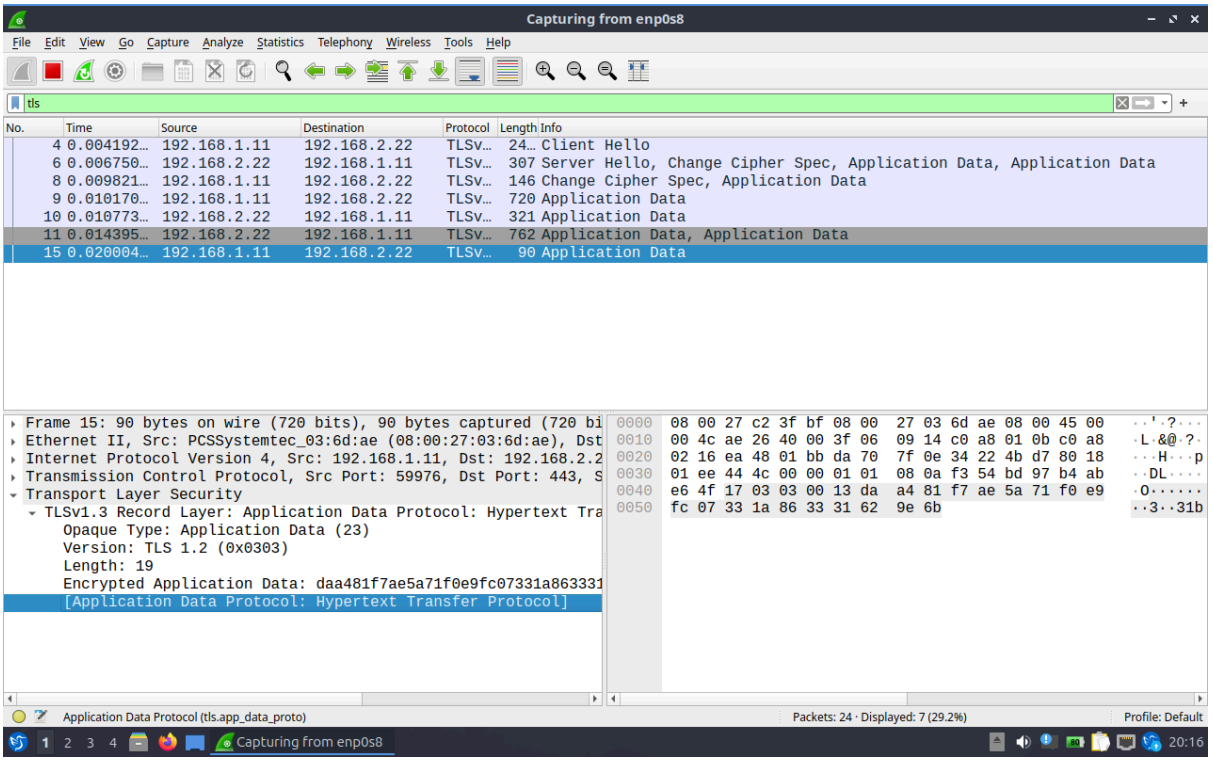
Terbukti bahwa persentase keberhasilan penyadapan pada protokol HTTP ialah 100%, mengingat dikirimnya menggunakan plain text.

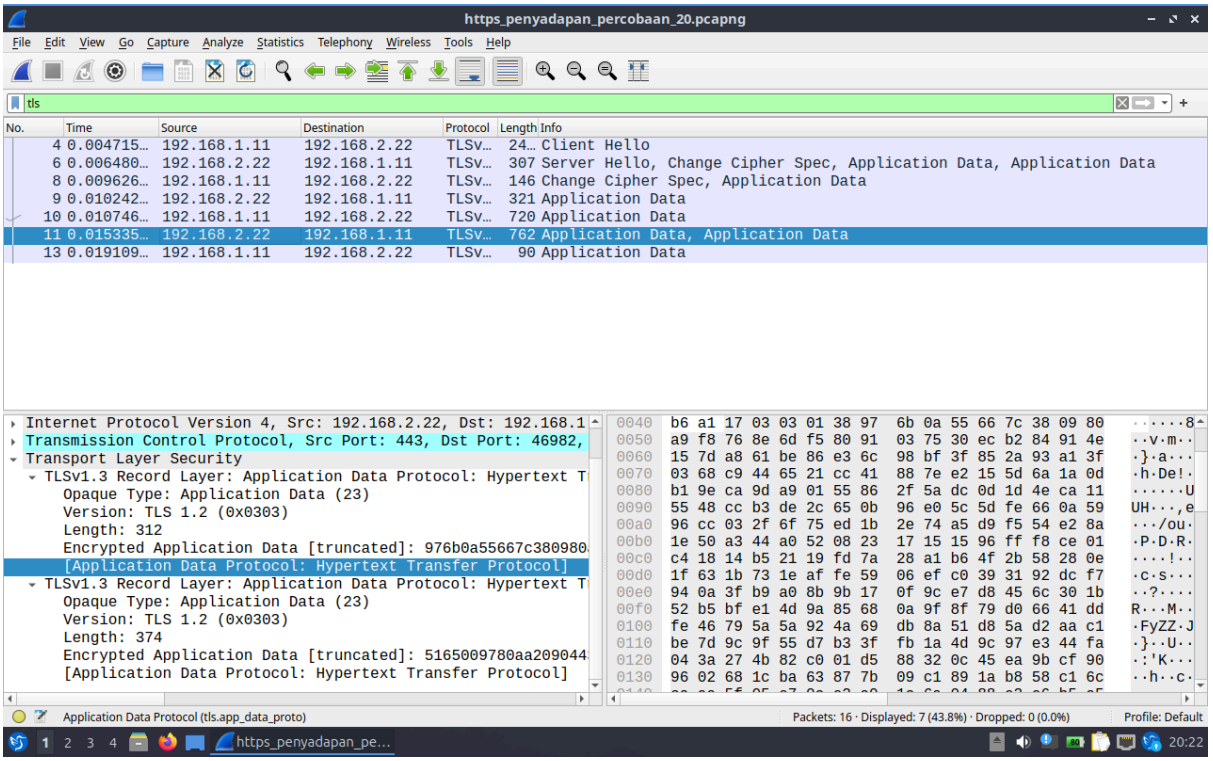
2. HTTPS

Sama seperti sebelumnya, data username dan password yang akan digunakan disini akan berubah secara minim guna membandingkan apakah ada perubahan dari tiap tahapan atau tidak, dengan contoh seperti berikut.



Ketika request dikirim, maka dapat terlihat untuk masing-masing request POST HTTPS dapat terlihat bahwa semua data telah terenkripsi, sehingga tidak memungkinkan untuk melihat data yang terkirim pada tiap paket POST.





Sehingga kesimpulan sebelumnya benar, bahwa terdapat persentase 0% untuk data bocor pada protokol HTTPS.