

## 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:

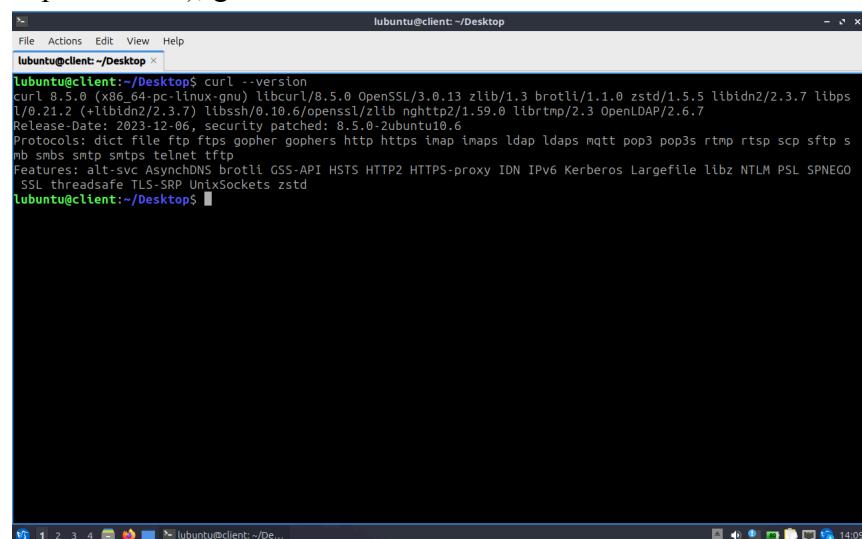
Pada progress kali ini akan difokuskan pada tahapan-tahapan tambahan yang didukung oleh screenshot yang ditangkap selama progress dilakukan. Lampiran-lampiran file capture juga akan disertakan saat pengiriman di link GitHub.

#### A. Jumlah paket tertangkap sebelum / sesudah enkripsi

##### 1. Sebelum enkripsi (HTTP)

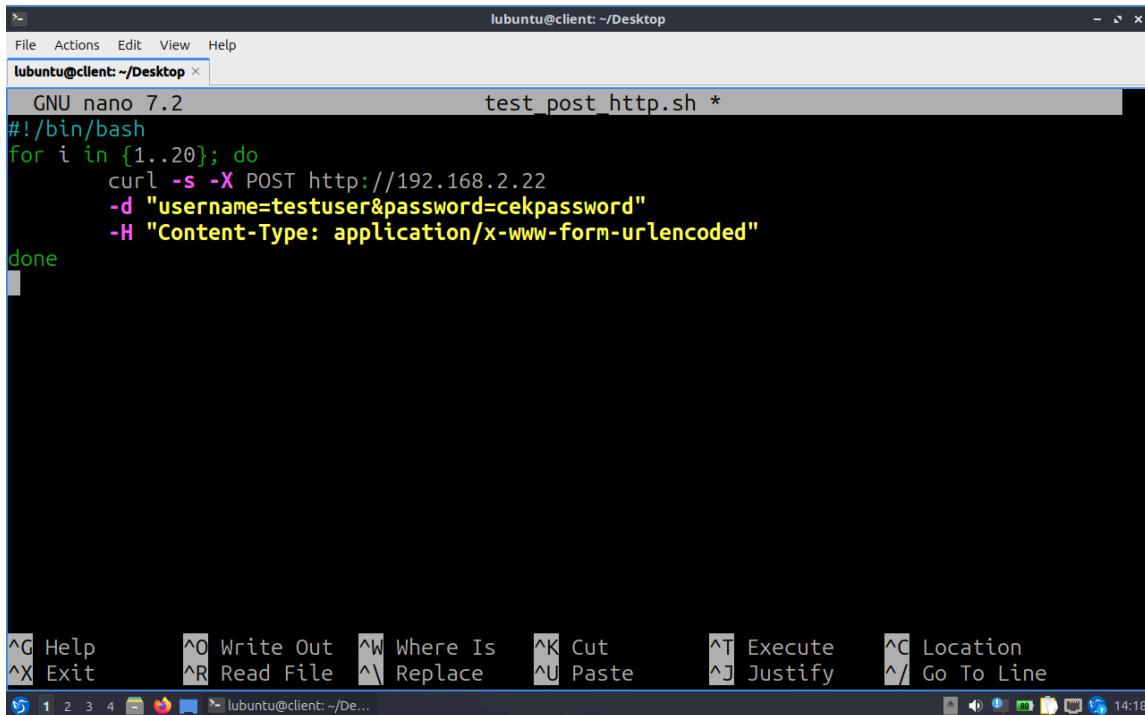
Jumlah request POST yang akan dicoba untuk progress ini ialah 20 request per tahapan, dan akan diaplikasikan pada tahapan-tahapan berikutnya demi konsistensi data. Guna memudahkan tahapan request lebih dari satu, kami memutuskan menggunakan looping script sederhana menggunakan curl.

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



```
lubuntu@client: ~/Desktop$ curl --version
curl 8.5.0 (x86_64-pc-linux-gnu) libcurl/8.5.0 OpenSSL/3.0.13 zlib/1.3 brotli/1.1.0 zstd/1.5.5 libidn2/2.3.7 libpsl/0.21.2 (+libidn2/2.3.7) libssh/0.10.6/openssl/zlib nghttp2/1.59.0 librtmp/2.3 OpenLDAP/2.6.7
Release-Date: 2023-12-06, security patched: 8.5.0-2ubuntu10.6
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smb2 smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM PSL SPNEGO
SSL threadsafe TLS-SRP UnixSockets zstd
lubuntu@client: ~/Desktop$
```

Jika sudah terinstall, maka selanjutnya kita buat script looping POST nya pada file bernama “*test\_post\_http.sh*” dengan kode sebagai berikut.

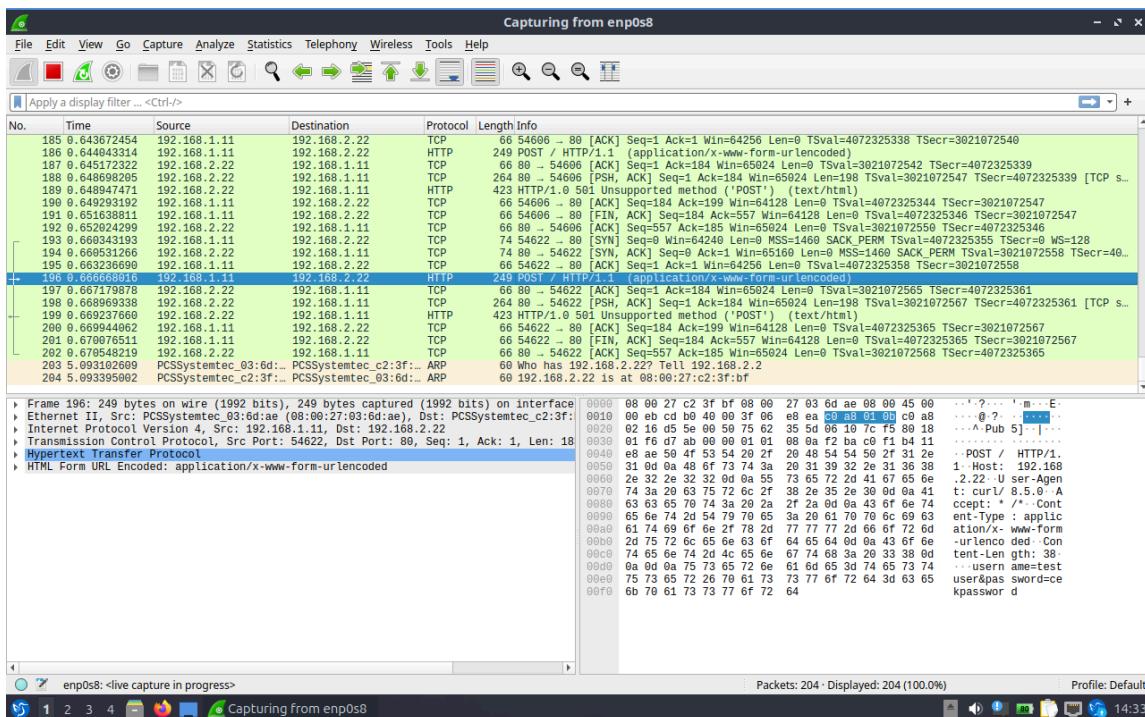


```

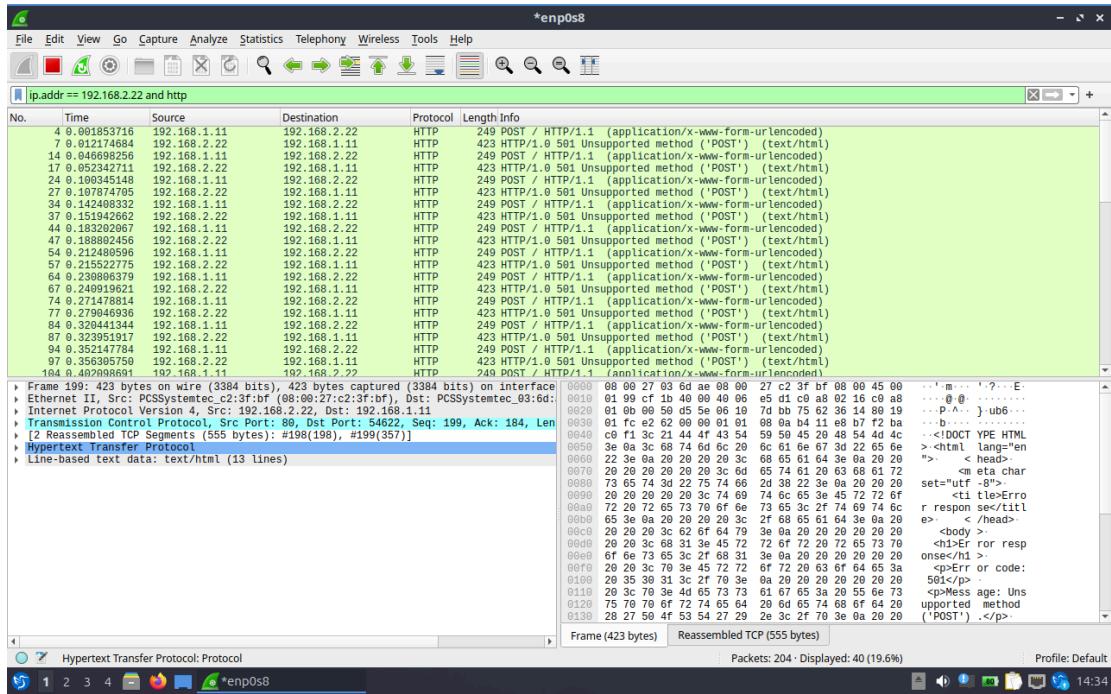
lubuntu@client: ~/Desktop
GNU nano 7.2
#!/bin/bash
for i in {1..20}; do
    curl -s -X POST http://192.168.2.22
    -d "username=testuser&password=cekpassword"
    -H "Content-Type: application/x-www-form-urlencoded"
done

```

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.



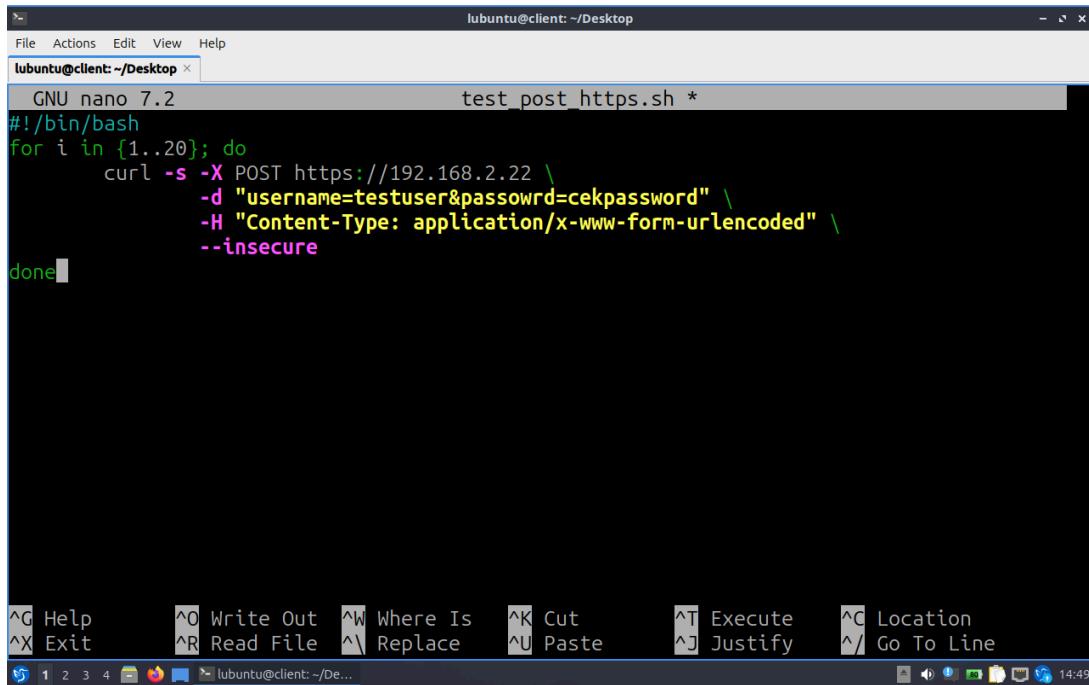
Total *packet* data sudah masuk hasil dari 20 kali request POST, dan masih bercampur-campur dengan packet lainnya, sehingga harus kita pakai filter HTTP dan IP target, yaitu vm server.



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 **19,6% dari total paket** yang ditampilkan.

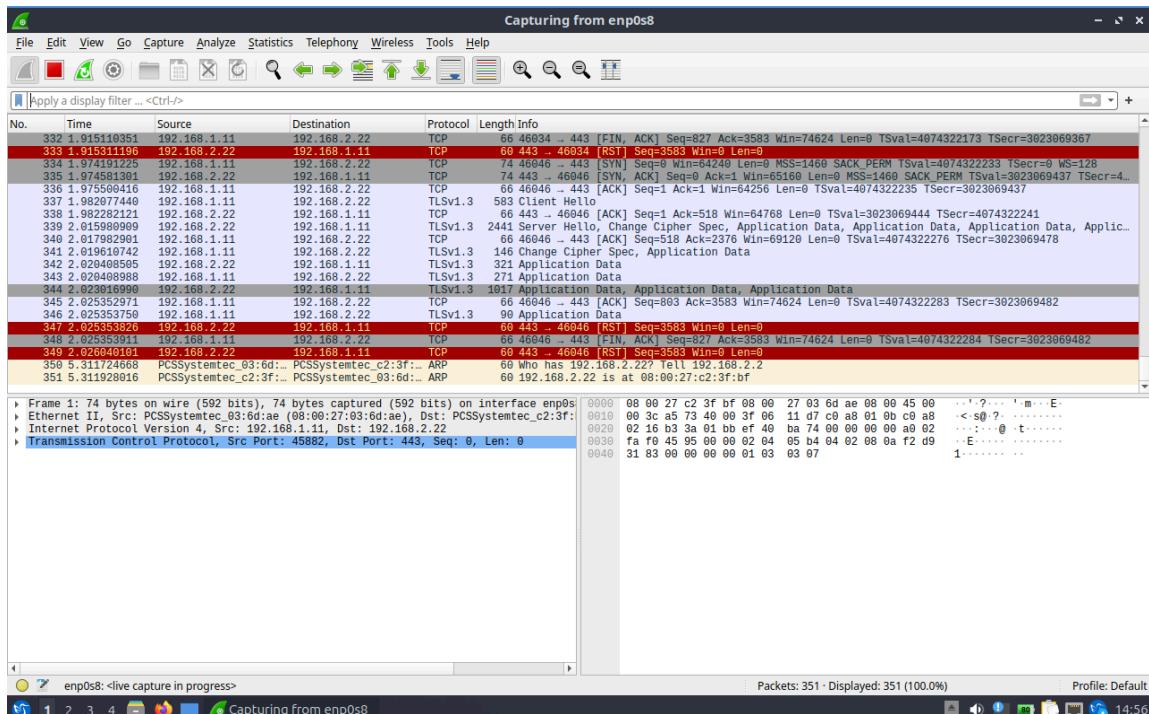
## 2. Pasca enkripsi (HTTPS)

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

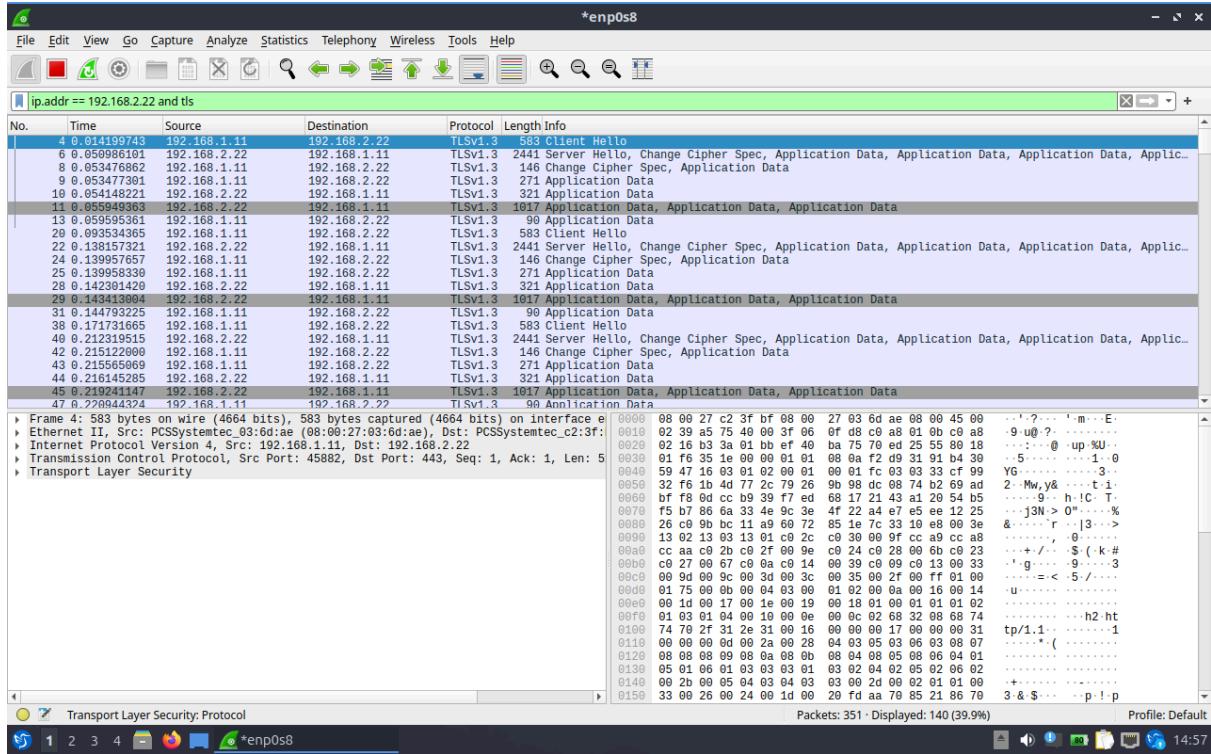


```
lubuntu@client: ~/Desktop
File Actions Edit View Help
lubuntu@client: ~/Desktop x
GNU nano 7.2 test_post_https.sh *
#!/bin/bash
for i in {1..20}; do
    curl -s -X POST https://192.168.2.22 \
        -d "username=testuser&password=cekpassword" \
        -H "Content-Type: application/x-www-form-urlencoded" \
        --insecure
done
```

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.

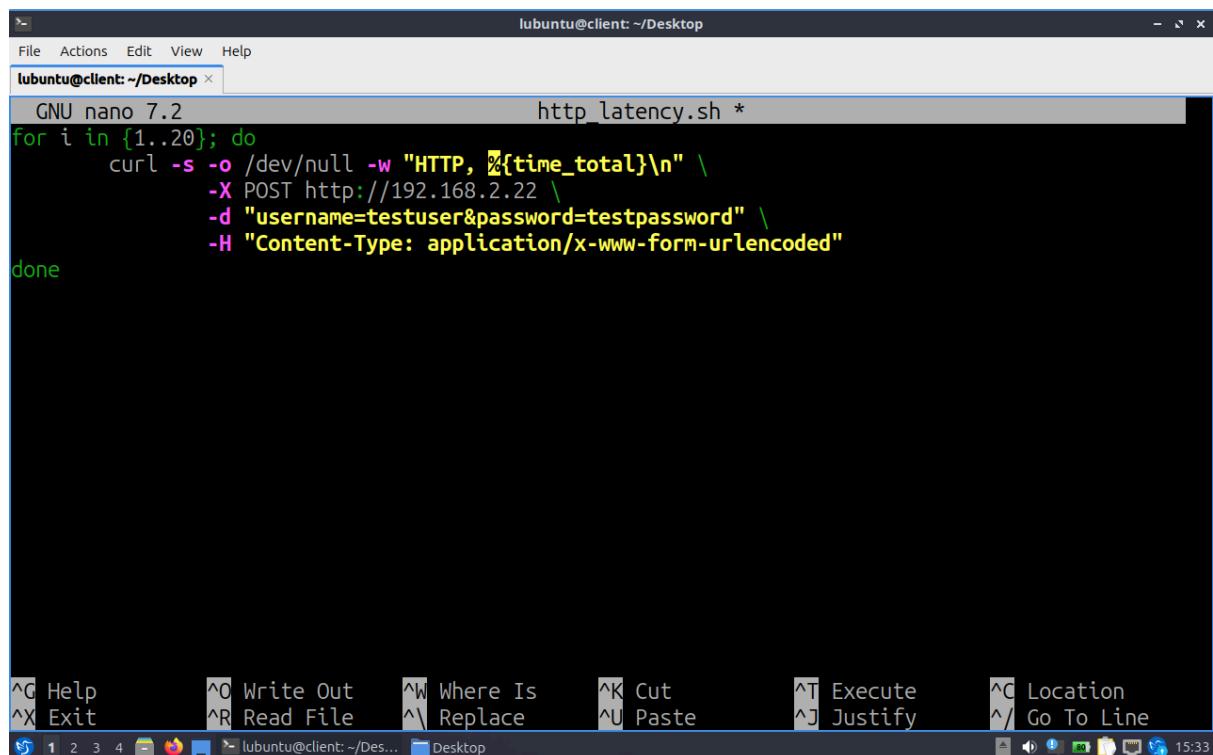
## B. Latency tambahan akibat TLS

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 *end-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.

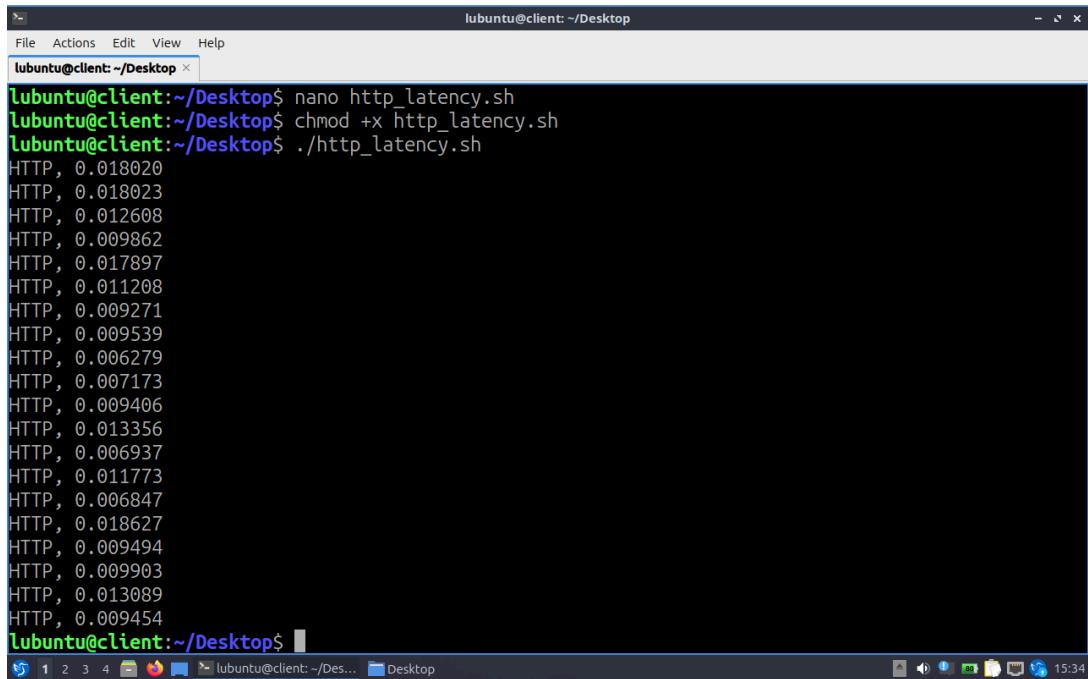


The screenshot shows a terminal window titled "lubuntu@client: ~/Desktop". The window contains a nano editor session with the following content:

```
GNU nano 7.2                               http_latency.sh *
for i in {1..20}; do
    curl -s -o /dev/null -w "%{time_total}\n" \
        -X POST http://192.168.2.22 \
        -d "username=testuser&password=testpassword" \
        -H "Content-Type: application/x-www-form-urlencoded"
done
```

The terminal window has a standard Linux desktop interface at the bottom, including icons for file operations, a terminal icon, and system status indicators like battery level and time (15:33).

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

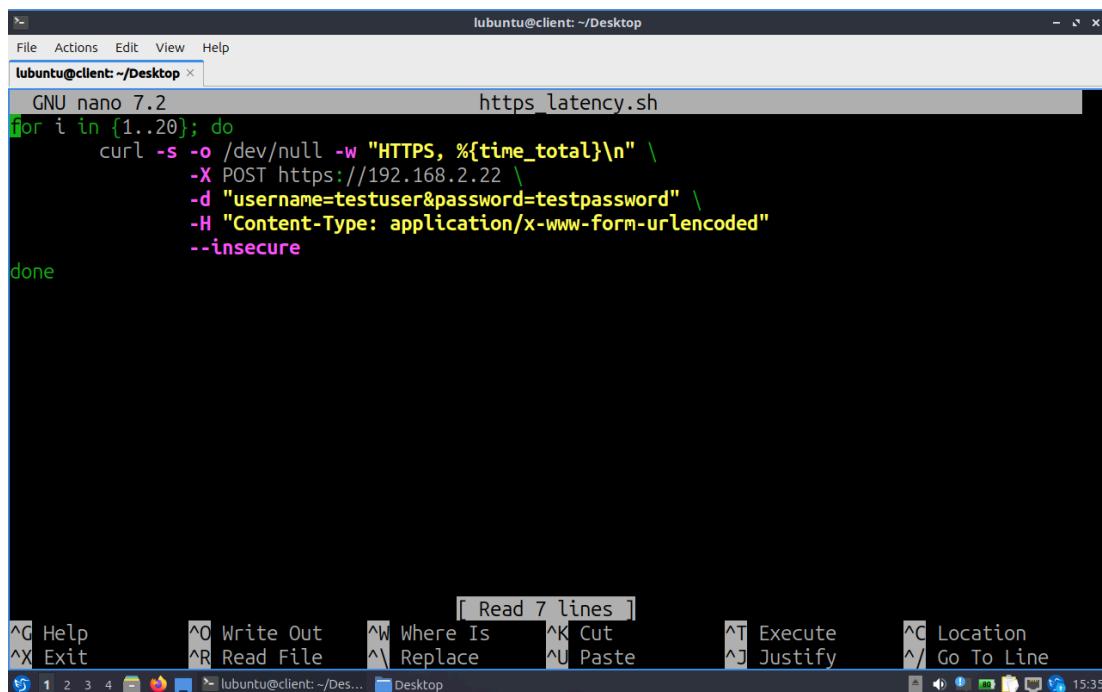


```
lubuntu@client: ~/Desktop$ nano http_latency.sh
lubuntu@client: ~/Desktop$ chmod +x http_latency.sh
lubuntu@client: ~/Desktop$ ./http_latency.sh
HTTP, 0.018020
HTTP, 0.018023
HTTP, 0.012608
HTTP, 0.009862
HTTP, 0.017897
HTTP, 0.011208
HTTP, 0.009271
HTTP, 0.009539
HTTP, 0.006279
HTTP, 0.007173
HTTP, 0.009406
HTTP, 0.013356
HTTP, 0.006937
HTTP, 0.011773
HTTP, 0.006847
HTTP, 0.018627
HTTP, 0.009494
HTTP, 0.009903
HTTP, 0.013089
HTTP, 0.009454
lubuntu@client: ~/Desktop$
```

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

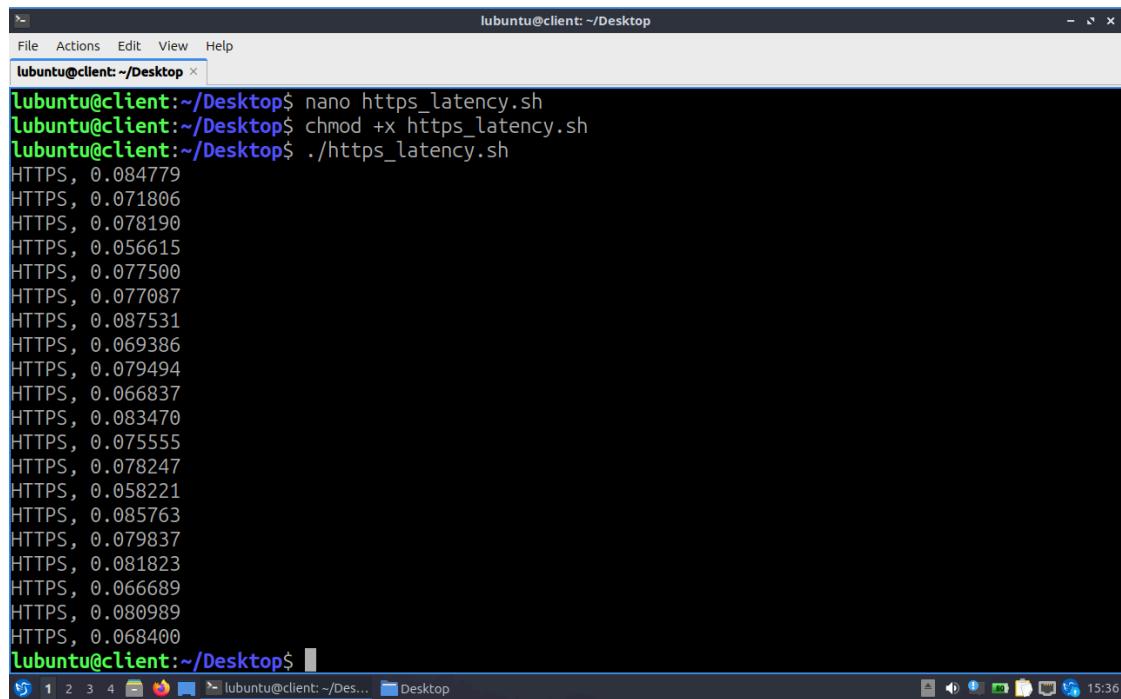
## 2. Latency HTTPS

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



```
lubuntu@client: ~/Desktop$ nano https_latency.sh
GNU nano 7.2
for i in {1..20}; do
    curl -s -o /dev/null -w "HTTPS, %{time_total}\n" \
        -X POST https://192.168.2.22 \
        -d "username=testuser&password=testpassword" \
        -H "Content-Type: application/x-www-form-urlencoded"
--insecure
done
```

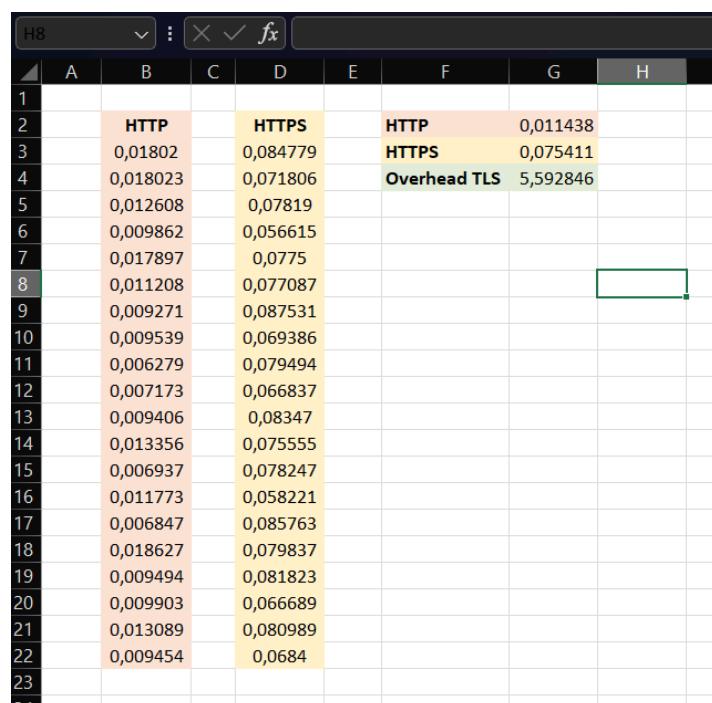
Setelah selesai dikonfigurasi, maka ketika dijalankan akan menampilkan hasil seperti ini.



```
lubuntu@client: ~/Desktop$ nano https_latency.sh
lubuntu@client: ~/Desktop$ chmod +x https_latency.sh
lubuntu@client: ~/Desktop$ ./https_latency.sh
HTTPS, 0.084779
HTTPS, 0.071806
HTTPS, 0.078190
HTTPS, 0.056615
HTTPS, 0.077500
HTTPS, 0.077087
HTTPS, 0.087531
HTTPS, 0.069386
HTTPS, 0.079494
HTTPS, 0.066837
HTTPS, 0.083470
HTTPS, 0.075555
HTTPS, 0.078247
HTTPS, 0.058221
HTTPS, 0.085763
HTTPS, 0.079837
HTTPS, 0.081823
HTTPS, 0.066689
HTTPS, 0.080989
HTTPS, 0.068400
lubuntu@client: ~/Desktop$
```

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 **rata-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:

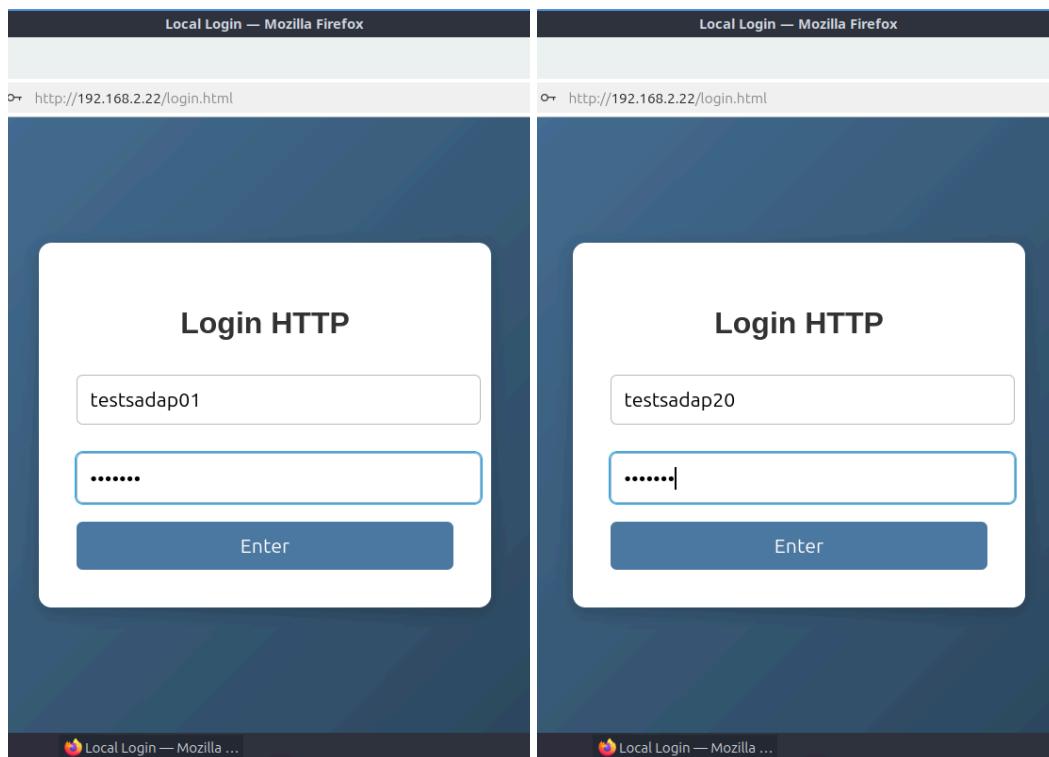
- HTTP yang mana password dalam plain text, seharusnya 100% berhasil.
- HTTPS yang mana password terenkripsi, seharusnya 0% berhasil.

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.

Capturing from enp0s8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length Info
4	0.005813...	192.168.1.11	192.168.2.22	HTTP	406 GET / HTTP/1.1
7	0.011163...	192.168.2.22	192.168.1.11	HTTP	298 HTTP/1.0 200 OK (text/html)
14	2.996132...	192.168.1.11	192.168.2.22	HTTP	497 GET /login.html HTTP/1.1
16	2.998472...	192.168.2.22	192.168.1.11	HTTP	170 HTTP/1.0 304 Not Modified
26	112.7266...	192.168.1.11	192.168.2.22	HTTP	583 POST / HTTP/1.1 (application/x-www-form-urlencoded)
29	112.7314...	192.168.2.22	192.168.1.11	HTTP	423 HTTP/1.0 501 Unsupported method ('POST') (text/html)

```

> Frame 26: 583 bytes on wire (4664 bits), 583 bytes captured (466
> Ethernet II, Src: PCSSystemtec_03:6d:ae (08:00:27:03:6d:ae), Dst
> Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.2.2
> Transmission Control Protocol, Src Port: 48888, Dst Port: 80, Se
> Hypertext Transfer Protocol
-> HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "testsadap01"
    > Form item: "password" = "sadap01"

```

Hex	Text
0150	74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69
0160	63 61 74 69 6f 6e 2f 78 2d 77 77 77 2d 66 6f 72
0170	6d 2d 75 72 6c 65 6e 63 6f 64 65 64 0d 0a 43 6f
0180	6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 33 37
0190	0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70 3a 2f
01a0	2f 31 39 32 2e 31 36 38 2e 32 2e 32 32 0d 0a 43
01b0	6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d
01c0	61 6c 69 76 65 0d 0a 52 65 66 65 72 65 72 3a 20
01d0	68 74 74 70 3a 2f 2f 31 39 32 2e 31 36 38 2e 32
01e0	2e 32 32 2f 6c 6f 67 69 6e 2e 68 74 6d 6c 0d 0a
01f0	55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65
0200	2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 50 72
0210	69 6f 72 69 74 79 3a 20 75 3d 30 2c 20 69 0d 0a
0220	0d 0a 75 73 65 72 6e 61 6d 65 3d 74 65 73 74 73
0230	61 64 61 70 30 31 26 70 61 73 73 77 6f 72 64 3d
0240	73 61 64 61 70 30 31

Packets: 36 · Displayed: 6 (16.7%)

Profile: Default

1 2 3 4 Capturing from enp0s8 19:25

Capturing from enp0s8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length Info
4	0.002817...	192.168.1.11	192.168.2.22	HTTP	583 POST / HTTP/1.1 (application/x-www-form-urlencoded)
7	0.008973...	192.168.2.22	192.168.1.11	HTTP	423 HTTP/1.0 501 Unsupported method ('POST') (text/html)

```

> Frame 4: 583 bytes on wire (4664 bits), 583 bytes captured (466
> Ethernet II, Src: PCSSystemtec_03:6d:ae (08:00:27:03:6d:ae), Dst
> Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.2.2
> Transmission Control Protocol, Src Port: 52740, Dst Port: 80, Se
> Hypertext Transfer Protocol
-> HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "testsadap20"
    > Form item: "password" = "sadap20"

```

Hex	Text
0150	74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69
0160	63 61 74 69 6f 6e 2f 78 2d 77 77 77 2d 66 6f 72
0170	6d 2d 75 72 6c 65 6e 63 6f 64 65 64 0d 0a 43 6f
0180	6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 33 37
0190	0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70 3a 2f
01a0	2f 31 39 32 2e 31 36 38 2e 32 2e 32 32 0d 0a 43
01b0	6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d
01c0	61 6c 69 76 65 0d 0a 52 65 66 65 72 65 72 3a 20
01d0	68 74 74 70 3a 2f 2f 31 39 32 2e 31 36 38 2e 32
01e0	2e 32 32 2f 6c 6f 67 69 6e 2e 68 74 6d 6c 0d 0a
01f0	55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65
0200	2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 50 72
0210	69 6f 72 69 74 79 3a 20 75 3d 30 2c 20 69 0d 0a
0220	0d 0a 75 73 65 72 6e 61 6d 65 3d 74 65 73 74 73
0230	61 64 61 70 32 30 26 70 61 73 73 77 6f 72 64 3d
0240	73 61 64 61 70 32 30

Packets: 12 · Displayed: 2 (16.7%)

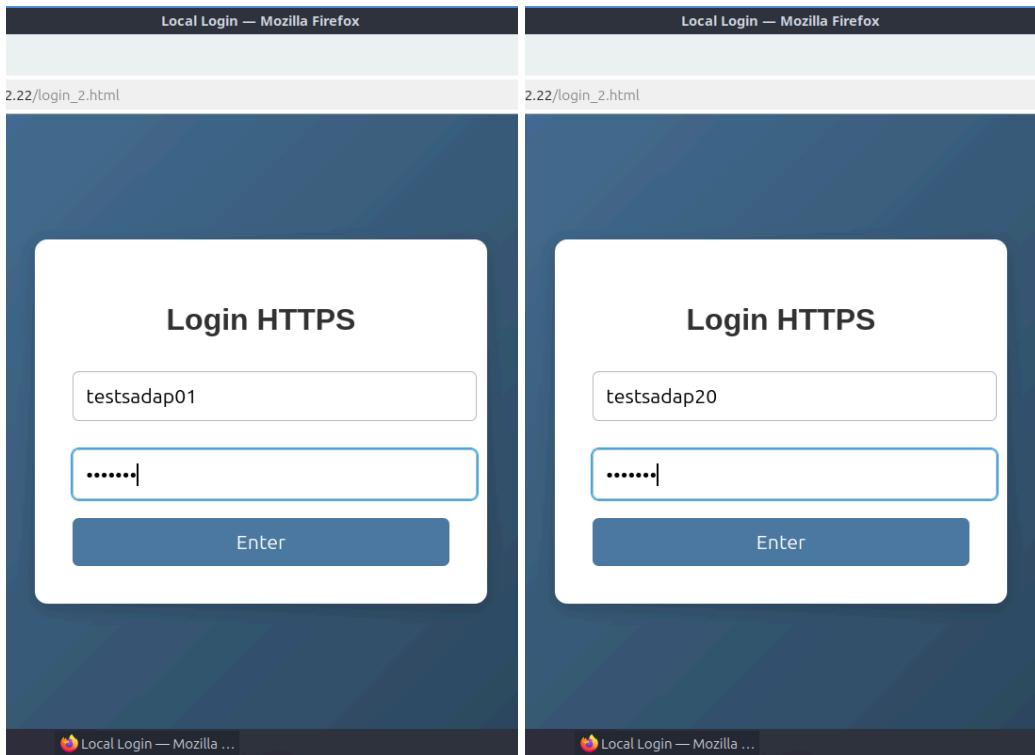
Profile: Default

1 2 3 4 Capturing from enp0s8 19:34

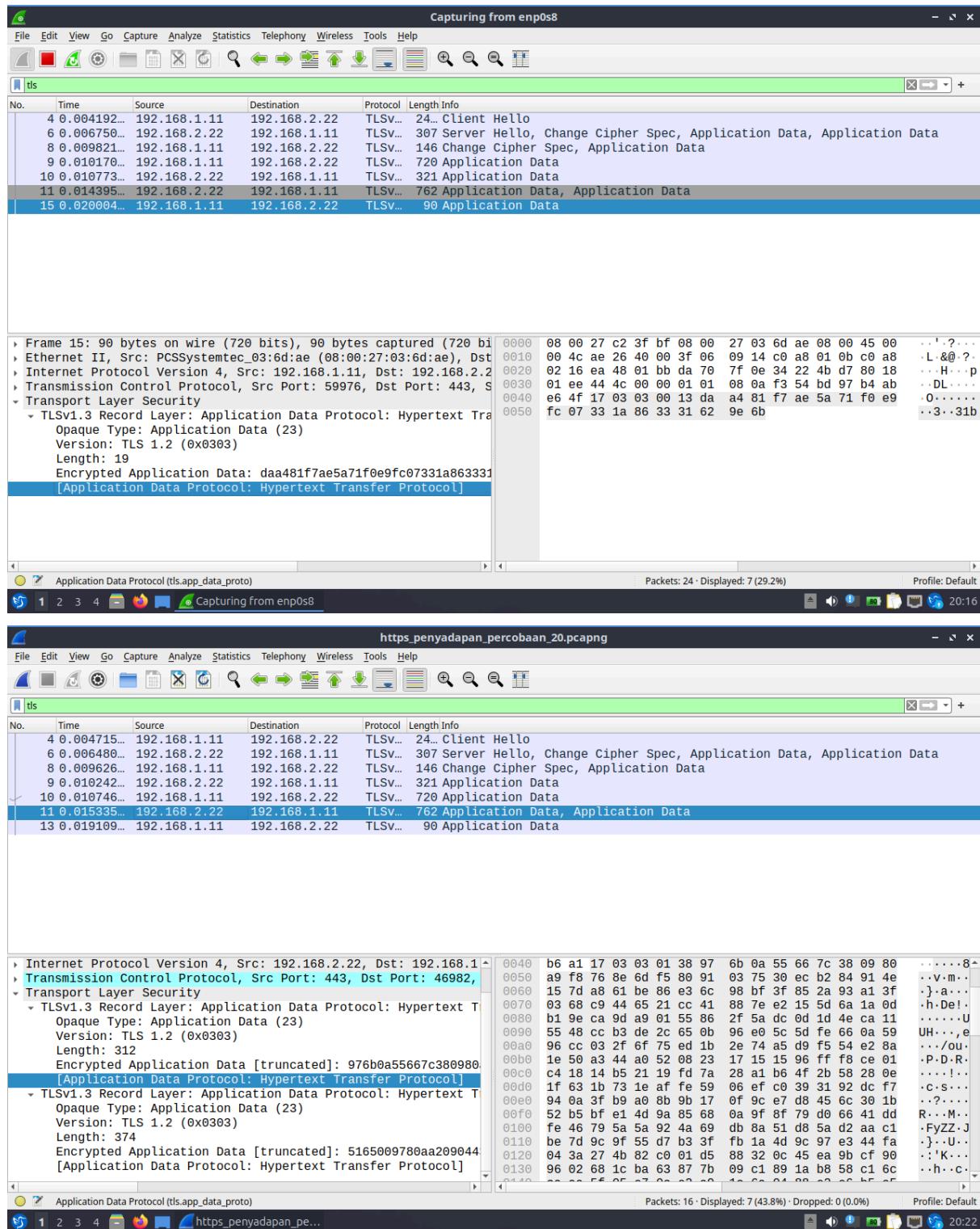
Terbukti bahwa persentase keberhasilan penyadapan pada protokol HTTP ialah 100%, mengingat dikirimnya menggunakan plain text.

## 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.