**Zadanie 7 - Zabezpečenie TLS web servera**

Cieľom tohto zadania je vytvorenie bezpečnej komunikácie medzi užívateľom (browserom) a serverom a testovanie možností zneužitia nezabezpečeného, alebo nedostatočne zabezpečeného spojenia.

**Úloha 1.) Vytvorte vlastný server a nainštalujte web server (apache, ngnix atd.)**

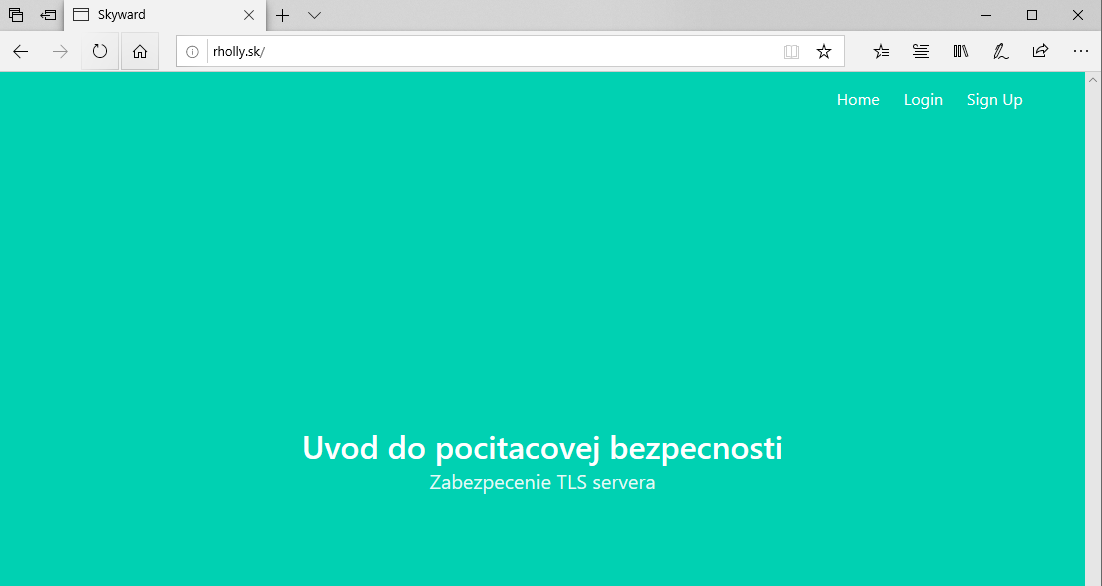
Web server som implementoval na svojom virtuálnom servery **Ubuntu** 18. Ktorý bol na účely zadania prístupný cez globálnu IP adresu a tiež cez **DNS** server s doménou rholly.sk

Backend web serveru je postavený na Python frameworku **Flask**. Ako produkčný web server som si vybral **Apache2.**

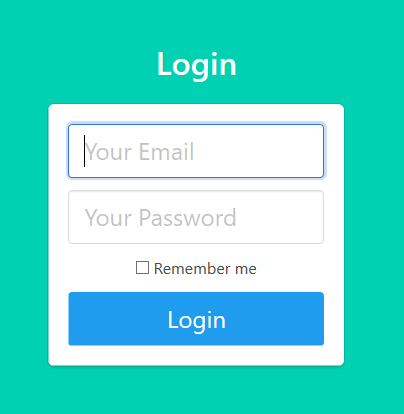
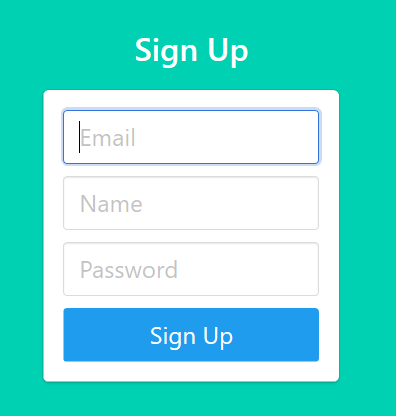
Návod: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>

**Úloha 2.) Vytvorte jednoduchú web aplikáciu na testovacie účely (jednoduchú stránku, popr. Formulár na prihlasovanie )**

V pythone som implementoval správu používateľských hesiel zahashovaných v SQLAlchemy databáze. Inšpiroval som sa návodom, ktorý využíva aj milý froontend prihlasovací a registrovací formulár <https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login>

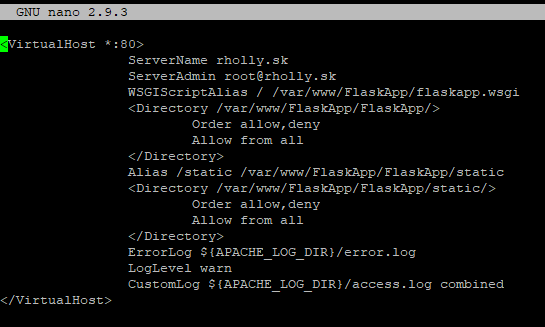


Screenshoty z dema web aplikácia

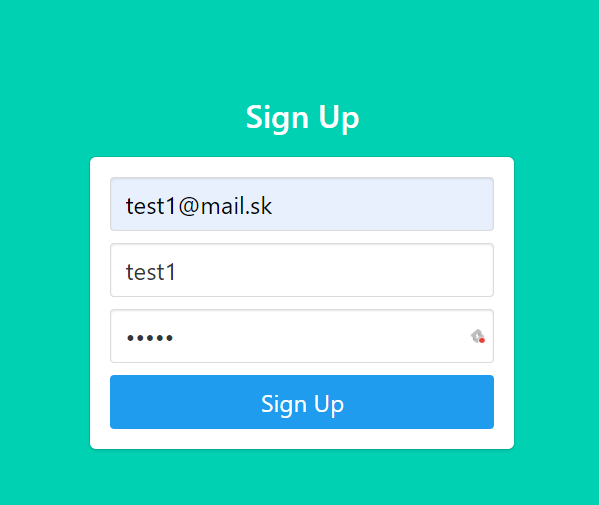
**Úloha 3.) Nakonfigurujte web server aby počúval cez http protokol na štandardnom porte**

Štandardný TCP port pre webovú http komunikáciu je port 80, tento port sa nastavuje pri konfigurácií virtuálneho spojenia na Apache2 server nasledovne:

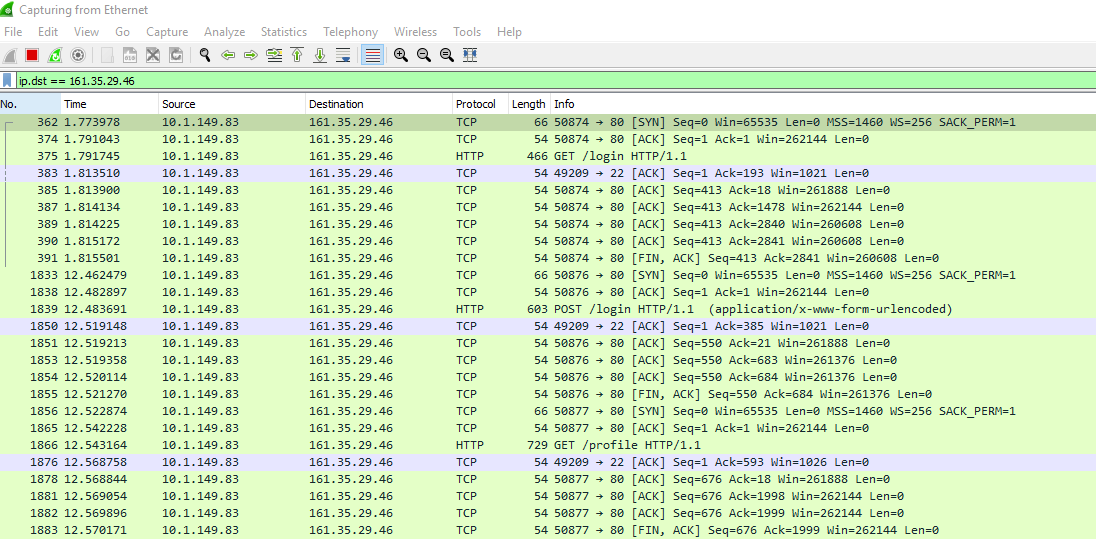


**Úloha 4.) Pomocou nástroja na sniffovanie komunikácie overte možnosť sledovania komunikácie medzi browserom a vašim serverom. Zistenia popíšte a zdôvodnite vo vašej dokumentácii k zadaniu**

Na sledovanie internetovej komunikácie som využil nástroj **WireShark**. Na web servery som si vytvoril testovaci účet s nasledovnými parametrami, heslo je rovnaké ako meno: **„test1“**

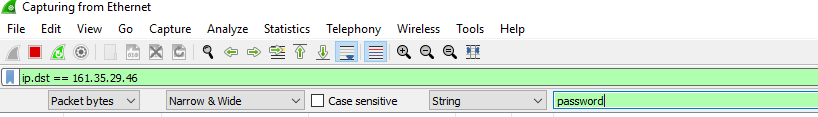


Po registrácií som prešiel na kartu Log In a začal sledovať cez WireShark komunikáciu, na čo som použil ip.dst filter, aby som videl len komunikáciu, ktorá smeruje na môj web server.

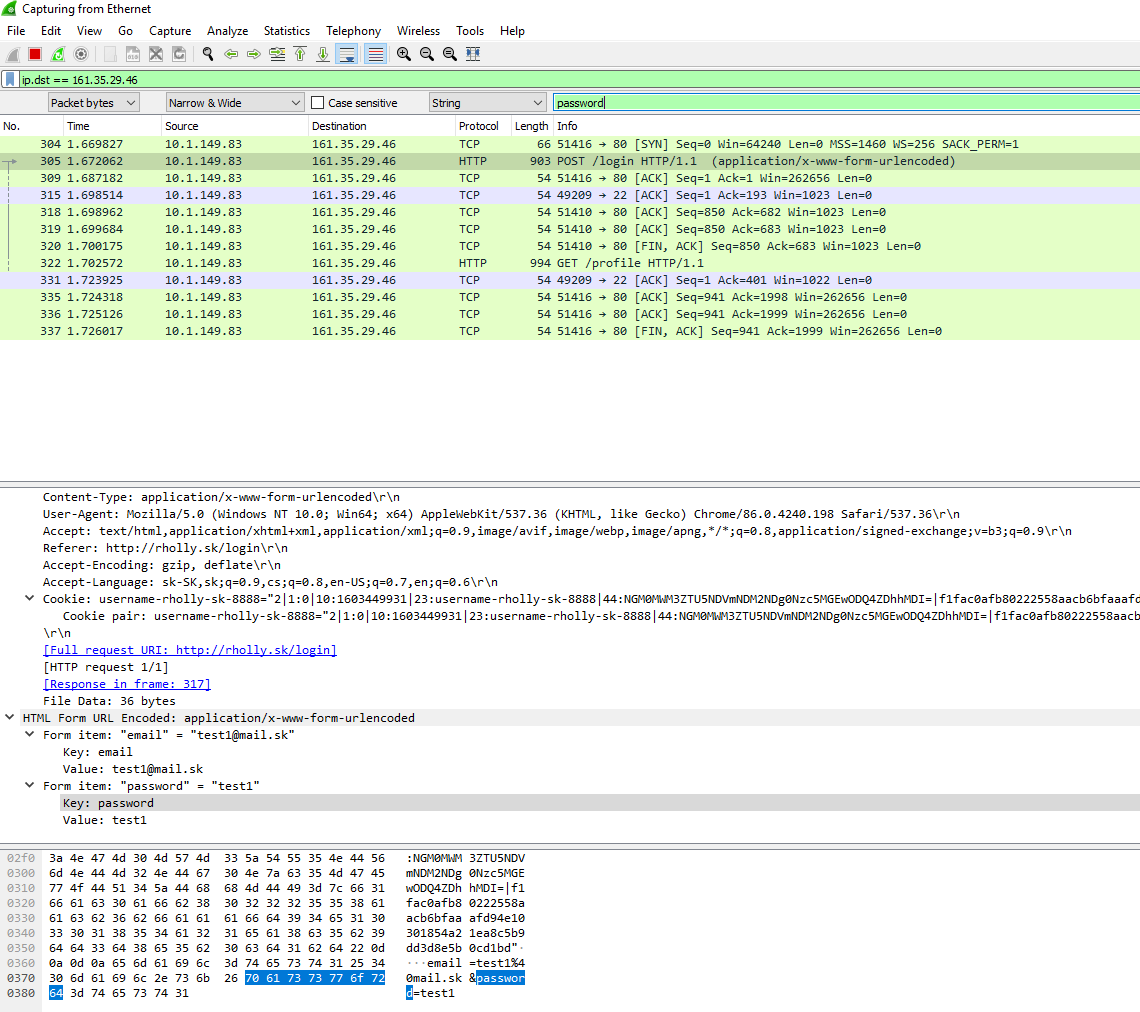


Vidíme, že napriek tomu, že je komunikácia filtrovaná, zachytávame viacej packetov, nás zaujíma http packet s metódou POST /login, tento by mal obsahovať credentials, teda prihlasovacie udaje meno a heslo v nezašifrovanom stave, nakoľko náš web server nepodporuje zatiaľ TLS ochranu.

V prípade, že by komunikácia bola neprehľadná, WireShark umožňuje vyhľadávať stringy priamo z obsahu packetov. Nasledovne: na obrázku je ako sa snažím vyhľadať reťazec „**password**“



Na tomto obrázku je vidieť, že sa mi naozaj podarilo nájsť v obsahu packetov reťazec password a to priamo v http POST packete, ako sme predpokladali.

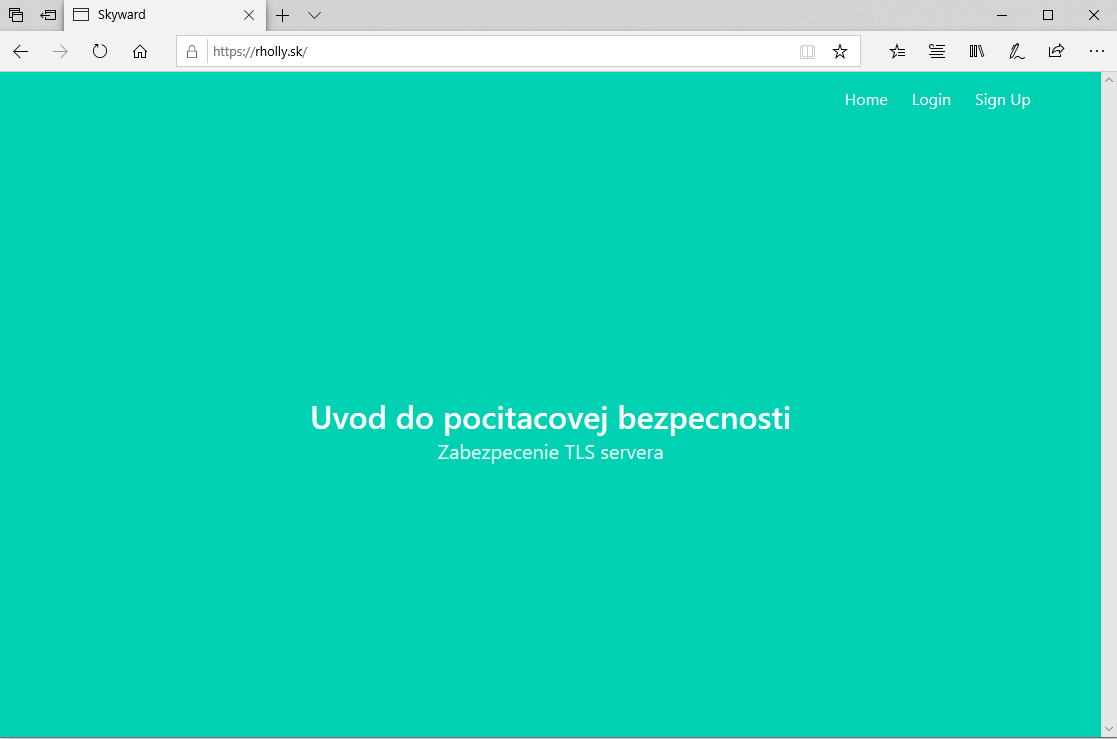


Naozaj vidíme, že medzi údajmi je posielane heslo **„test1“ ako** plain text.

**Úloha 5.) Nakonfigurujte web server na komunikáciu pomocou https protokolu. Pri konfigurácii sa riaďte odporúčaniami pre vytvorenie správnych bezpečnostných nastavení (odporúčané verzie TLS, odporúčané veľkosti kľúčov a použitie šifrovacích algoritmov, atď)**

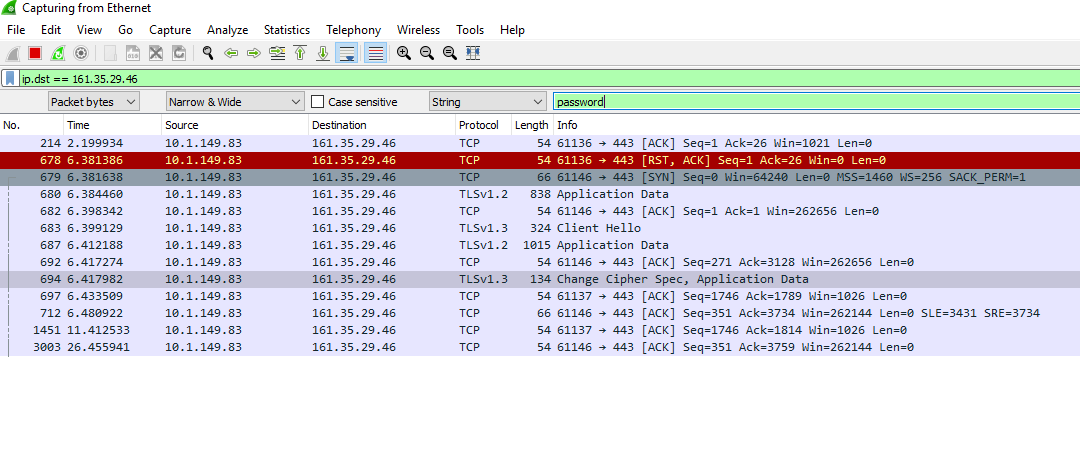
Aj pri tejto úlohe som sa inspiroval návodom, ktorý integruje LetsEncrypt TLS certifikát do Apache2 web servera. <https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-18-04>

Pomohlo my, že môj server má registrovanú DNS rholly.sk , takže LetsEncrypt nemal problém vygenerovať certifikát. Trvalo mi než som správne zostavil Flask aplikáciu, nastavil firewall a ďalšie záležitosti. Ale nakoniec sa mi podarilo správne a bezpečne nakonfigurovať web server Flask + Apache + TLS. Dôkaz úspešnej implementácia je screen s URL podporovanou HTTPS prenos a neskôr odchytená zašifrované komunikácia.



**Úloha 6. Vykonajte základne testovanie bezpečnosti https konfigurácie a odhalené nedostatky odstránte)**

Znovu som odchytával komunikáciu pomocou WireSharku, avšak teraz, ako je vidno packety sú zašifrované TLS protokolom. Teraz už neviem nájsť plain text „password“ ani žiadny podobný v obsahu packetov.



Pri implantácií TLS protokolu som využil možnosť presmerovanie http komunikácie na bezpečné HTTPS na strane Apache serveru.

Považujem za bezpečnostné plus, že web server beží na najaktuálnejších verziách Ubuntu, Apache, Flask. Ďalej, užívateľské mená sú bezpečne uložené v SQLALchemy databaze. TLS protokol je nevyhnutná vlastnosť každého verejného servera.

Pokúšal som sa nájsť slabinu na svoj systém útokmi , avšak nepodarilo sa my takú nájsť.

Overenie bezpečnosti TLS na  <https://www.ssllabs.com/ssltest/> vyšlo pozitívne a technické požiadavky ako veľkosti kľúčov a algoritmy, spĺňajú štandardy.

