

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, nginx atd.)

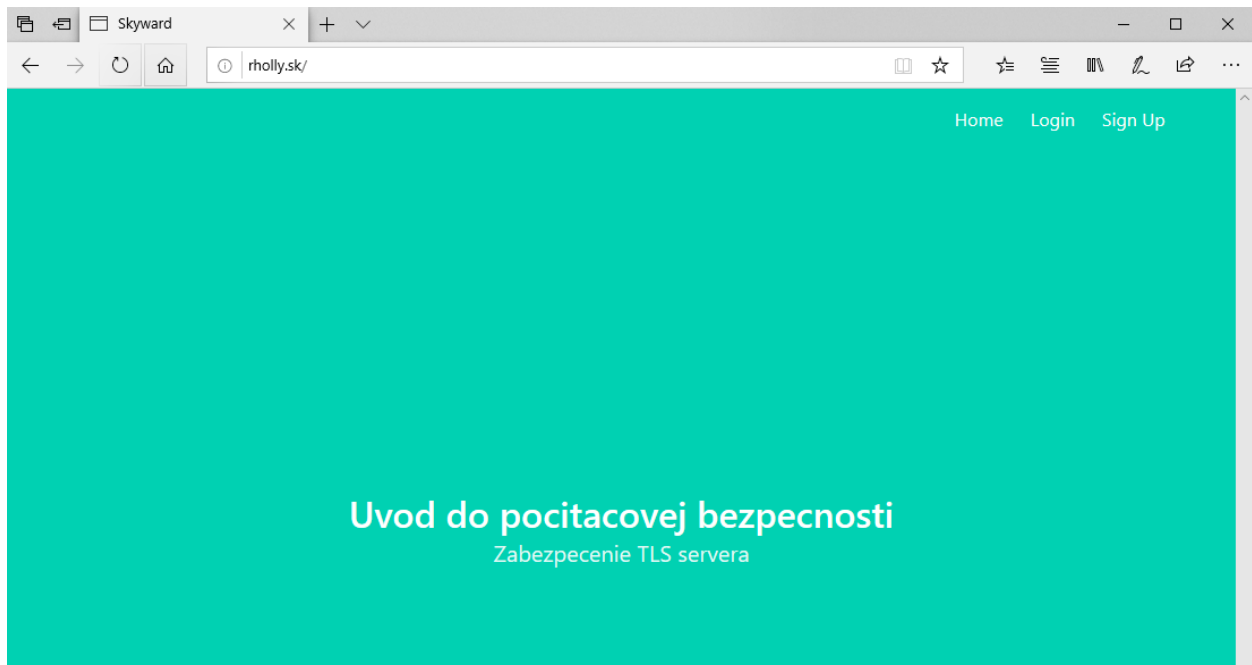
Web server som implementoval na svojom virtuálnom servery **Ubuntu 18**. Ktorý bol na účely zadania prístupný cez globálnu IP adresu a tiež cez **DNS** server s doménou rholly.sk

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

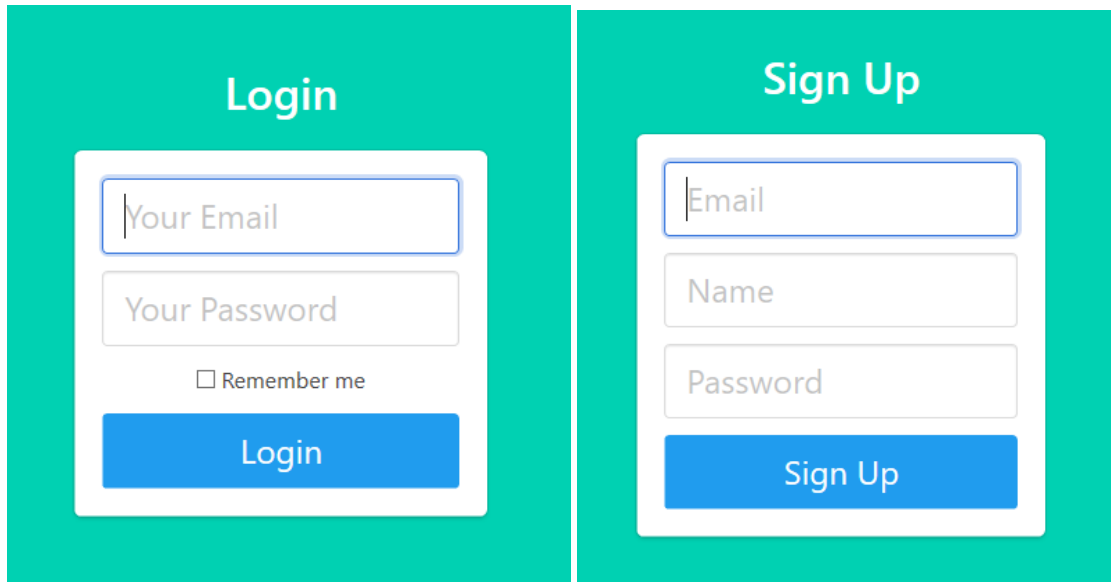
Návod: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>

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>



Screenshots z demo web aplikácia



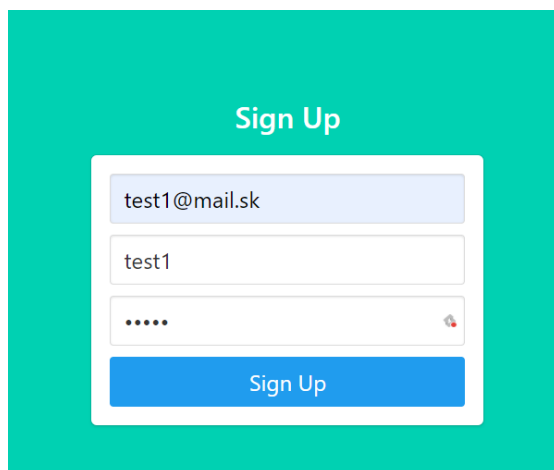
Úloha 3.) Nakonfigurujte web server aby počúval cez http protokol na štandardnom porte

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

```
GNU nano 2.9.3
<VirtualHost *:80>
    ServerName rholly.sk
    ServerAdmin root@rholly.sk
    WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi
    <Directory /var/www/FlaskApp/FlaskApp/>
        Order allow,deny
        Allow from all
    </Directory>
    Alias /static /var/www/FlaskApp/FlaskApp/static
    <Directory /var/www/FlaskApp/FlaskApp/static/>
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

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 testovací účet s nasledovnými parametrami, heslo je rovnaké ako meno: „test1“



Po registrácii 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.

Capturing from Ethernet

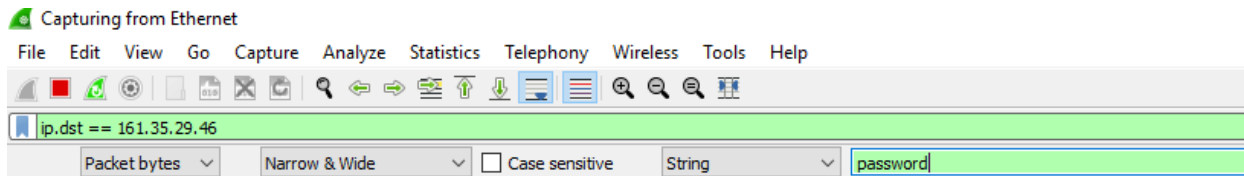
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.dst == 161.35.29.46

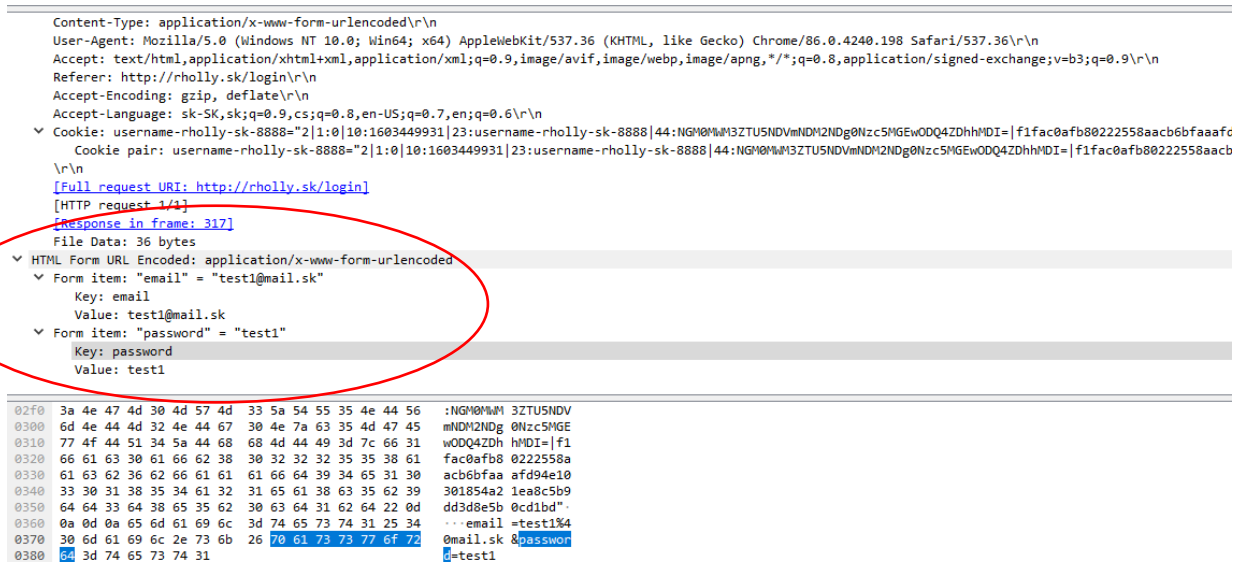
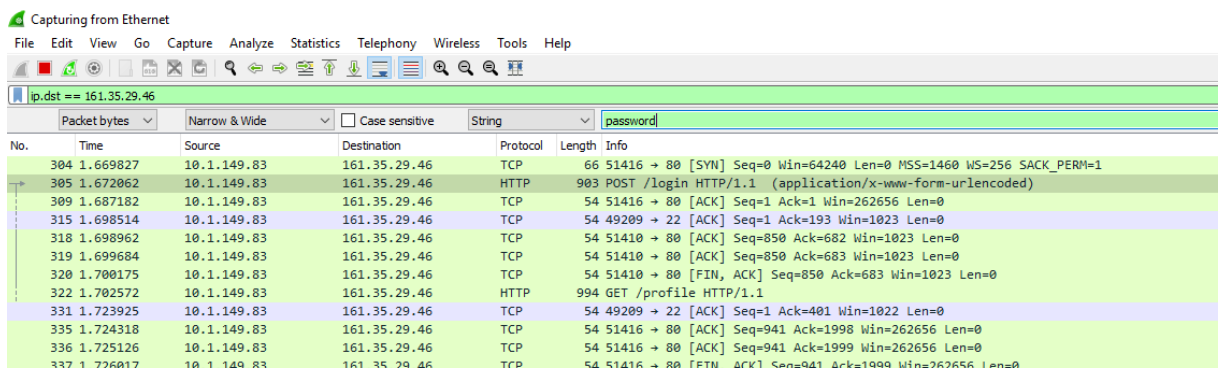
No.	Time	Source	Destination	Protocol	Length	Info
362	1.773978	10.1.149.83	161.35.29.46	TCP	66	50874 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
374	1.791043	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
375	1.791745	10.1.149.83	161.35.29.46	HTTP	466	GET /login HTTP/1.1
383	1.813510	10.1.149.83	161.35.29.46	TCP	54	49209 → 22 [ACK] Seq=1 Ack=193 Win=1021 Len=0
385	1.813900	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [ACK] Seq=413 Ack=18 Win=261888 Len=0
387	1.814134	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [ACK] Seq=413 Ack=1478 Win=262144 Len=0
389	1.814225	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [ACK] Seq=413 Ack=2840 Win=260608 Len=0
390	1.815172	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [ACK] Seq=413 Ack=2841 Win=260608 Len=0
391	1.815501	10.1.149.83	161.35.29.46	TCP	54	50874 → 80 [FIN, ACK] Seq=413 Ack=2841 Win=260608 Len=0
1833	12.462479	10.1.149.83	161.35.29.46	TCP	66	50876 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
1838	12.482897	10.1.149.83	161.35.29.46	TCP	54	50876 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
1839	12.483691	10.1.149.83	161.35.29.46	HTTP	603	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
1850	12.519148	10.1.149.83	161.35.29.46	TCP	54	49209 → 22 [ACK] Seq=1 Ack=385 Win=1021 Len=0
1851	12.519213	10.1.149.83	161.35.29.46	TCP	54	50876 → 80 [ACK] Seq=550 Ack=21 Win=261888 Len=0
1853	12.519358	10.1.149.83	161.35.29.46	TCP	54	50876 → 80 [ACK] Seq=550 Ack=683 Win=261376 Len=0
1854	12.520114	10.1.149.83	161.35.29.46	TCP	54	50876 → 80 [ACK] Seq=550 Ack=684 Win=261376 Len=0
1855	12.521270	10.1.149.83	161.35.29.46	TCP	54	50876 → 80 [FIN, ACK] Seq=550 Ack=684 Win=261376 Len=0
1856	12.522874	10.1.149.83	161.35.29.46	TCP	66	50877 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
1865	12.542228	10.1.149.83	161.35.29.46	TCP	54	50877 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
1866	12.543164	10.1.149.83	161.35.29.46	HTTP	729	GET /profile HTTP/1.1
1876	12.568758	10.1.149.83	161.35.29.46	TCP	54	49209 → 22 [ACK] Seq=1 Ack=593 Win=1026 Len=0
1878	12.568844	10.1.149.83	161.35.29.46	TCP	54	50877 → 80 [ACK] Seq=676 Ack=18 Win=261888 Len=0
1881	12.569054	10.1.149.83	161.35.29.46	TCP	54	50877 → 80 [ACK] Seq=676 Ack=1998 Win=262144 Len=0
1882	12.569096	10.1.149.83	161.35.29.46	TCP	54	50877 → 80 [ACK] Seq=676 Ack=1999 Win=262144 Len=0
1883	12.570171	10.1.149.83	161.35.29.46	TCP	54	50877 → 80 [FIN, ACK] Seq=676 Ack=1999 Win=262144 Len=0

Vidíme, že napriek tomu, že je komunikácia filtrovaná, zachytávame viacero packetov, nás zaujíma http packet s metódou POST /login, tento by mal obsahovať credentials, teda prihlasovacie údaje 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.

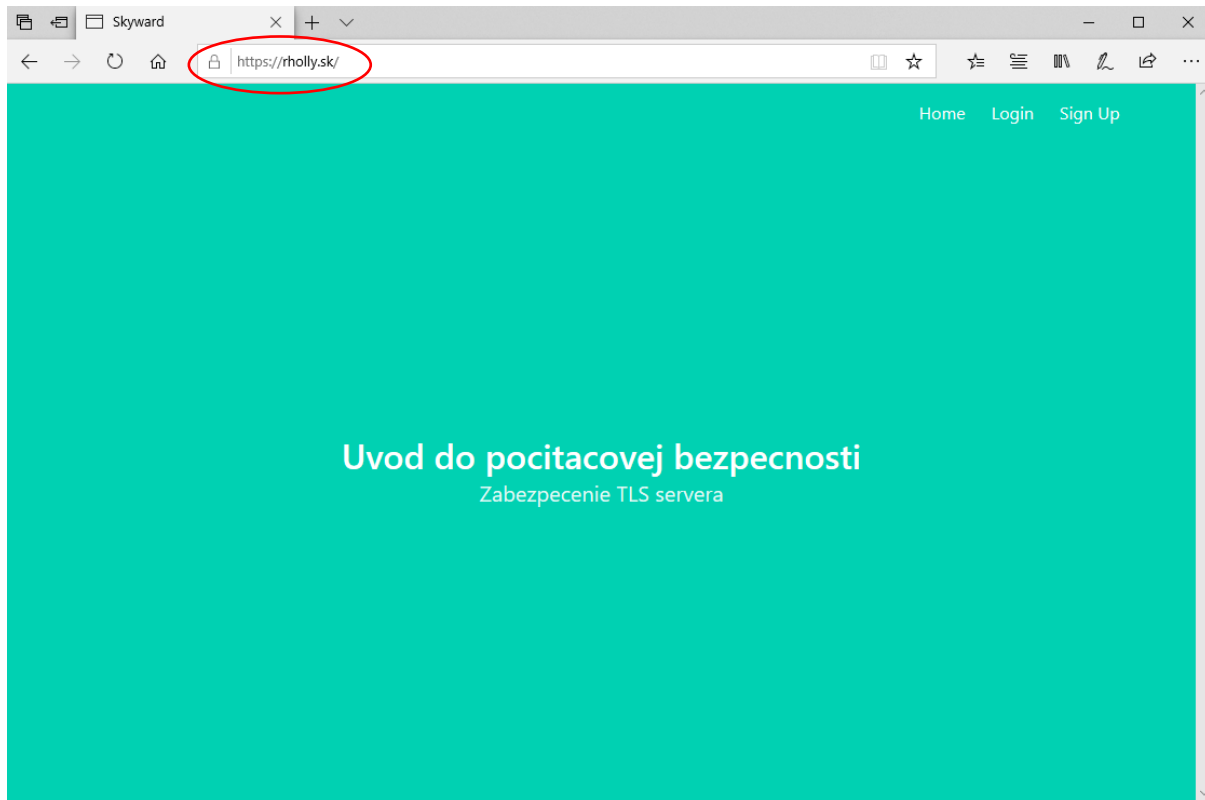


Naozaj vidíme, že medzi údajmi je posielane heslo „test1“ ako plain text.

Úloha 5.) Nakonfigurujte web server na komunikáciu pomocou https protokolu. Pri konfigurácii sa riad'te 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 zašifrovaná komunikácia.



Úloha 6. Vykonajte základné testovanie bezpečnosti https konfigurácie a odhalené nedostatky odstráňte)

Znovu som odchyťoval 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.

Capturing from Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.dst == 161.35.29.46

Packet bytes Narrow & Wide Case sensitive String password

No.	Time	Source	Destination	Protocol	Length	Info
214	2.199934	10.1.149.83	161.35.29.46	TCP	54	61136 → 443 [ACK] Seq=1 Ack=26 Win=1021 Len=0
678	6.381386	10.1.149.83	161.35.29.46	TCP	54	61136 → 443 [RST, ACK] Seq=1 Ack=26 Win=0 Len=0
679	6.381638	10.1.149.83	161.35.29.46	TCP	66	61146 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
680	6.384460	10.1.149.83	161.35.29.46	TLSv1.2	838	Application Data
682	6.398342	10.1.149.83	161.35.29.46	TCP	54	61146 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0
683	6.399129	10.1.149.83	161.35.29.46	TLSv1.3	324	Client Hello
687	6.412188	10.1.149.83	161.35.29.46	TLSv1.2	1015	Application Data
692	6.417274	10.1.149.83	161.35.29.46	TCP	54	61146 → 443 [ACK] Seq=271 Ack=3128 Win=262656 Len=0
694	6.417982	10.1.149.83	161.35.29.46	TLSv1.3	134	Change Cipher Spec, Application Data
697	6.433509	10.1.149.83	161.35.29.46	TCP	54	61137 → 443 [ACK] Seq=1746 Ack=1789 Win=1026 Len=0
712	6.480922	10.1.149.83	161.35.29.46	TCP	66	61146 → 443 [ACK] Seq=351 Ack=3734 Win=262144 Len=0 SLE=3431 SRE=3734
1451	11.412533	10.1.149.83	161.35.29.46	TCP	54	61137 → 443 [ACK] Seq=1746 Ack=1814 Win=1026 Len=0
3003	26.455941	10.1.149.83	161.35.29.46	TCP	54	61146 → 443 [ACK] Seq=351 Ack=3759 Win=262144 Len=0

Pri implantácii 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 databáze. 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.

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > rholly.sk

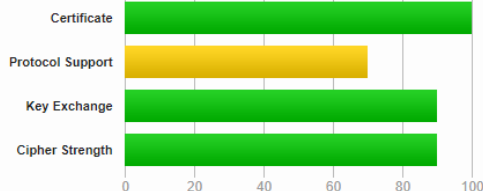
SSL Report: rholly.sk (161.35.29.46)

Assessed on: Sun, 22 Nov 2020 17:40:56 UTC | [Hide](#) | [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server supports TLS 1.0 and TLS 1.1. Grade capped to B. [MORE INFO »](#)

This server supports TLS 1.3.

Certificate #1: RSA 2048 bits (SHA256withRSA)



Server Key and Certificate #1



Subject	rholly.sk Fingerprint SHA256: 818deffe5bc9fda2ddffac31b7f7b9a8da45fcf823040a227284058ef9da4a23 Pin SHA256: mLzLKY6ocWXwmwDQ0iUPJ2y2d98ZkPA5L2e2ex4Od+E=
Common names	rholly.sk
Alternative names	rholly.sk www.rholly.sk
Serial Number	03f0e080e7762f2942c29dfcbeb27cbf38ec
Valid from	Thu, 24 Sep 2020 09:28:21 UTC
Valid until	Wed, 23 Dec 2020 09:28:21 UTC (expires in 1 month)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	Let's Encrypt Authority X3 AIA: http://cert.int-x3.letsencrypt.org/
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation information	OCSP OCSP: http://ocsp.int-x3.letsencrypt.org
Revocation status	Good (not revoked)
DNS CAA	No (more info)
Trusted	Yes Mozilla Apple Android Java Windows