

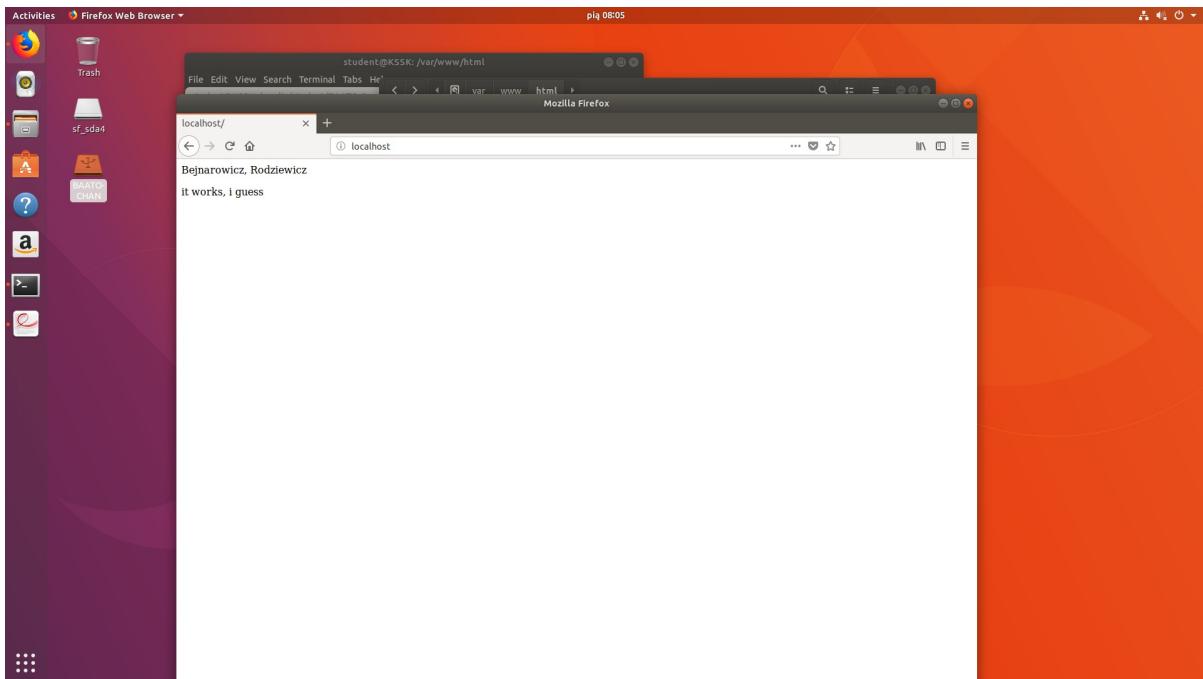
Bezpieczeństwo sieci komputerowych

Sprawozdanie z laboratorium

Data	Tytuł zajęć	Uczestnicy
16.11.2018 09:15	Bezpieczne usługi sieciowe, wirtualne sieci prywatne	Igor Bejnarowicz (218573) Bartosz Rodziewicz (226105)

1. Postawienie serwera WWW

Pierwszym zadaniem było postawienie serwera WWW na komputerze z Linuxem. Po zainstalowaniu pakietu `apache2` i przygotowaniu prostej strony sprawdziliśmy, że serwer działa.



Za pomocą drugiego komputera sprawdziliśmy połączenie z tym serwerem i jego bezpieczeństwo za pomocą Wiresharka.

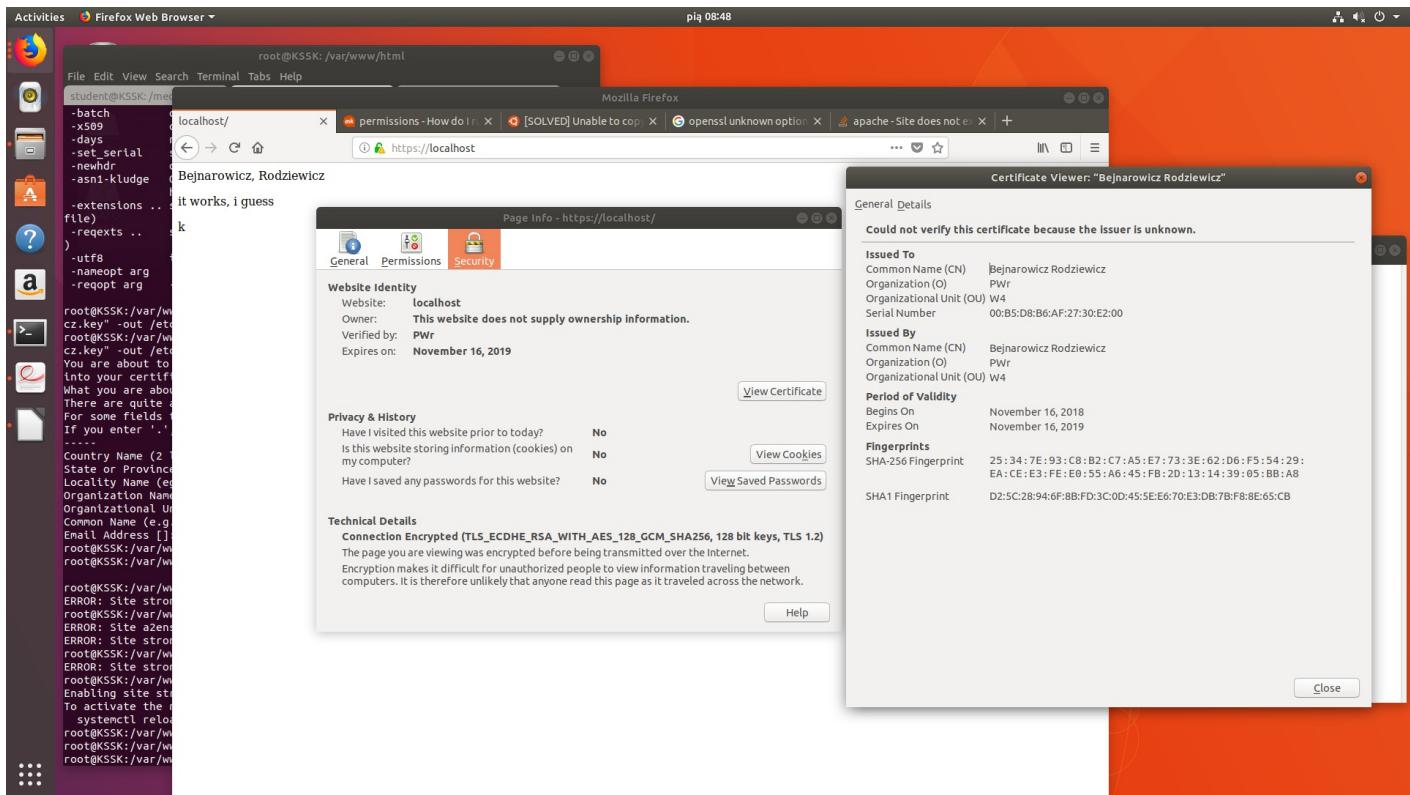
The Wireshark captures show the following sequence:

- Frame 63: 402 bytes on wire (321 bits), 22 bytes captured (176 bits) on interface Ethernet 4
Time: 0.000000 - 0.000000 192.168.255.115 → 192.168.255.142 [HTTP]
Host: 192.168.255.115
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Fri, 16 Nov 2018 07:04:21 GMT
If-None-Match: "38-57ac2cd6806c"
Cache-Control: max-age=0
HTTP/1.1 200 OK
- Frame 64: 28 bytes on wire (224 bits), 28 bytes captured (224 bits) on interface Ethernet 4
Time: 0.000000 - 0.000000 192.168.255.142 → 192.168.255.115 [HTTP]
Host: 192.168.255.115
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Fri, 16 Nov 2018 07:04:21 GMT
If-None-Match: "38-57ac2cd6806c"
Cache-Control: max-age=0
HTTP/1.1 200 OK

Na screenshocie wyżej widać, że za pomocą Wiresharka byliśmy w stanie przechwycić całą komunikację wraz z treścią dokumentu HTML.

2. Konfiguracja protokołu SSL

Zgodnie z krokami w instrukcji wykonaliśmy podstawową konfigurację pakietu `openssl`. Po wykonaniu tych kroków i dodaniu do adresu w przeglądarce przedrostka `https://` udało nam się nawiązać szyfrowane połączenie z serwerem. Fakt ten oczywiście został zgłoszony prowadzącemu.



3. Szczegóły certyfikatu

Powyższy screenshot poza nawiązaniem bezpiecznego połączenia pokazuje również szczegóły certyfikatu.

Jakie algorytmy kryptograficzne wykorzystane zostały do zestawienia tej sesji ssl?

Widać to na powyższym screenshocie - jest to algorytm `TSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256`.

Czy wystawca certyfikatu Państwa witryny został dodany do listy zaufanych wystawców w przeglądarce?

Wystawcą naszego certyfikatu jesteśmy my sami (`--self-signed`), więc ten wystawca nie mógł być na liście zaufanych wystawców. Konieczne było dodanie wyjątku, aby przeglądarka pozwoliła na połączenie.

Dlaczego przeglądarka sygnalizuje że „Połączenie nie jest bezpieczne” (proszę szczegółowo wyjaśnić na czym polega problem)?

Tak jak napisałem wyżej - nasz certyfikat został wystawiony bezpośrednio przez nas. W związku z tym przeglądarka nie jest pewna, kto ten nasz certyfikat wystawił i nie wie, czy można ufać co do jego autentyczności. Inaczej mówiąc, taki certyfikat zapewnia poufność połączenia, ale nie zapewnia uwierzytelnienia serwera.

Aby to było możliwe przeglądarka musi wiedzieć, że klucz prywatny użyty do podpisania certyfikatu należy do kogoś, komu można ufać i nie podpisze on certyfikatu dla naszej domeny tak po prostu, byle komu. Do ustalenia tego służy łańcuch zaufania (chain of trust) – potrzebny nam certyfikat jest podpisywany za pomocą jakiegoś klucza, dla którego też został wystawiony certyfikat i też został on podpisany za pomocą jakiegoś klucza. Takich poziomów w góre może być wiele, ale ważne jest, że w końcu na samej górze znajdzie się certyfikat jakiegoś urzędu certyfikacji (Certificate Authority) podpisany samodzielnie. Każda przeglądarka ma bazę takich certyfikatów najwyższego poziomu (tzw. root CA), których wystawcom ufa. Jeżeli używany przez nas certyfikat odwołuje się w końcu do jednego z nich, przeglądarka uzna połączenie za bezpieczne. W naszym przypadku łańcuch zaufania składa się oczywiście tylko z jednego poziomu (`self-signed`).

Za pomocą Wireshark ocenić bezpieczeństwo połączenia ssl (czy przesyłane treści są szyfrowane, czy certyfikat jest szyfrowany, itp.).

Po nawiązaniu połączenia przesyłany jest certyfikat (w formie jawniej, jednak to nie jest problem ponieważ zawiera tylko klucz publiczny) i od tego momentu nawiązane jest szyfrowane połączenie. Sama treść przesyłana jest w formie szyfrowanej.

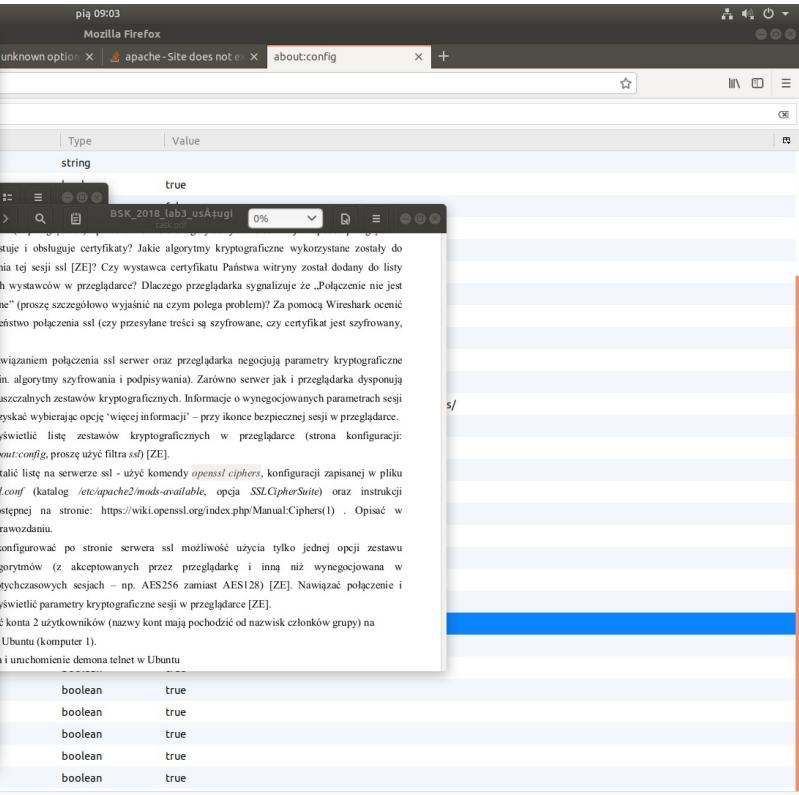
The screenshot shows a Wireshark session titled "Follow SSL Stream (tcp.stream eq 0) - wireshark_D847557B-E0A5-4972-A46F-567C6127FC0D_201811160857". The interface includes a top bar with file, edit, view, history, bookmarks, tools, and help menus. Below the menu is a toolbar with various icons. The main window is divided into several panes: a left pane for frame numbers, a middle-left pane for detailed frame information, a middle-right pane for bytes, and a bottom pane for ASCII representation. The bytes pane shows the raw data of the captured frames, which include the SSL/TLS handshake process, such as Client Hello, Server Hello, and Change Cipher Spec messages.

4.1. Zestawy algorytmów obsługiwane przez Firefoxa i [openssl](#)

Firefox obsługuje następujące algorytmy:

The screenshot shows the Mozilla Firefox browser window with the address bar set to "localhost/" and the title bar showing "Mozilla Firefox". The main content area displays the "about:config" page. A search bar at the top is set to "ssl". Below the search bar is a table with the following columns: "Preference Name", "Status", "Type", and "Value". The table lists numerous SSL/TLS-related preferences, such as "network.negotiate-auth.gsslib", "network.negotiate-auth.using-native-gsslib", "network.predictor.enable-hover-on-ssl", "network.proxy.ssl", "network.proxy.ssl_port", "security.ssl.enable_alpn", "security.ssl.enable_false_start", "security.ssl.enable_ocsp_must_staple", "security.ssl.errorReporting.automatic", "security.ssl.errorReporting.enabled", "security.ssl.errorReporting.url", "security.ssl.require_safe_negotiation", "security.ssl.treat_unsafe_negotiation_as_broken", "security.ssl3.dhe_rsa_aes_128_sha", "security.ssl3.dhe_rsa_aes_256_sha", "security.ssl3.ecdh_ecdsa_aes_128_gcm_sha256", "security.ssl3.ecdh_ecdsa_aes_128_sha", "security.ssl3.ecdh_ecdsa_aes_256_gcm_sha384", "security.ssl3.ecdh_ecdsa_aes_256_sha", "security.ssl3.ecdh_ecdsa_chacha20_poly1305_sha256", "security.ssl3.ecdh_ecrsa_aes_128_gcm_sha256", "security.ssl3.ecdh_ecrsa_aes_128_sha", "security.ssl3.ecdh_ecrsa_aes_256_gcm_sha384", "security.ssl3.ecdh_ecrsa_aes_256_sha", "security.ssl3.ecdh_ecrsa_chacha20_poly1305_sha256", "security.ssl3.rsa_aes_128_sha", "security.ssl3.rsa_aes_256_sha", and "security.ssl3.rsa_desede3_sha". Each preference has a status (e.g., default, boolean), type (e.g., string, boolean, integer), and value (e.g., true, false, 0).

OpenSSL obsługuje następujące algorytmy:



4.2. Zmiana algorytmu połączenia z serwerem

Za pomocą edycji pliku `ssl.conf` zmieniliśmy wartość pola `SSLCipherSuite` z `HIGH:!aNULL` na `ECDHE-RSA-AES256-GCM-SHA384:!aNULL` co wymusiło na przeglądarce używanie algorytmu `AES256`.

localhost/ permissions

File Edit View Search Terminal Tabs Help

student@KSSK:/media/s... x student@KSSK:/var/www/html x student@KSSK:/etc/apac... x

GNU nano 2.8.6 File: /etc/apache2/mods-available/ssl.conf

```
# Configure the path to the mutual exclusion semaphore the
# SSL engine uses internally for inter-process synchronization.
# (Disabled by default, the global Mutex directive consolidates by de...
# this)
#Mutex file:${APACHE_LOCK_DIR}/ssl_mutex ssl-cache

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate. See the
# ciphers(1) man page from the openssl package for list of all available
# options.
# Enable only secure ciphers:
SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384:!aNULL

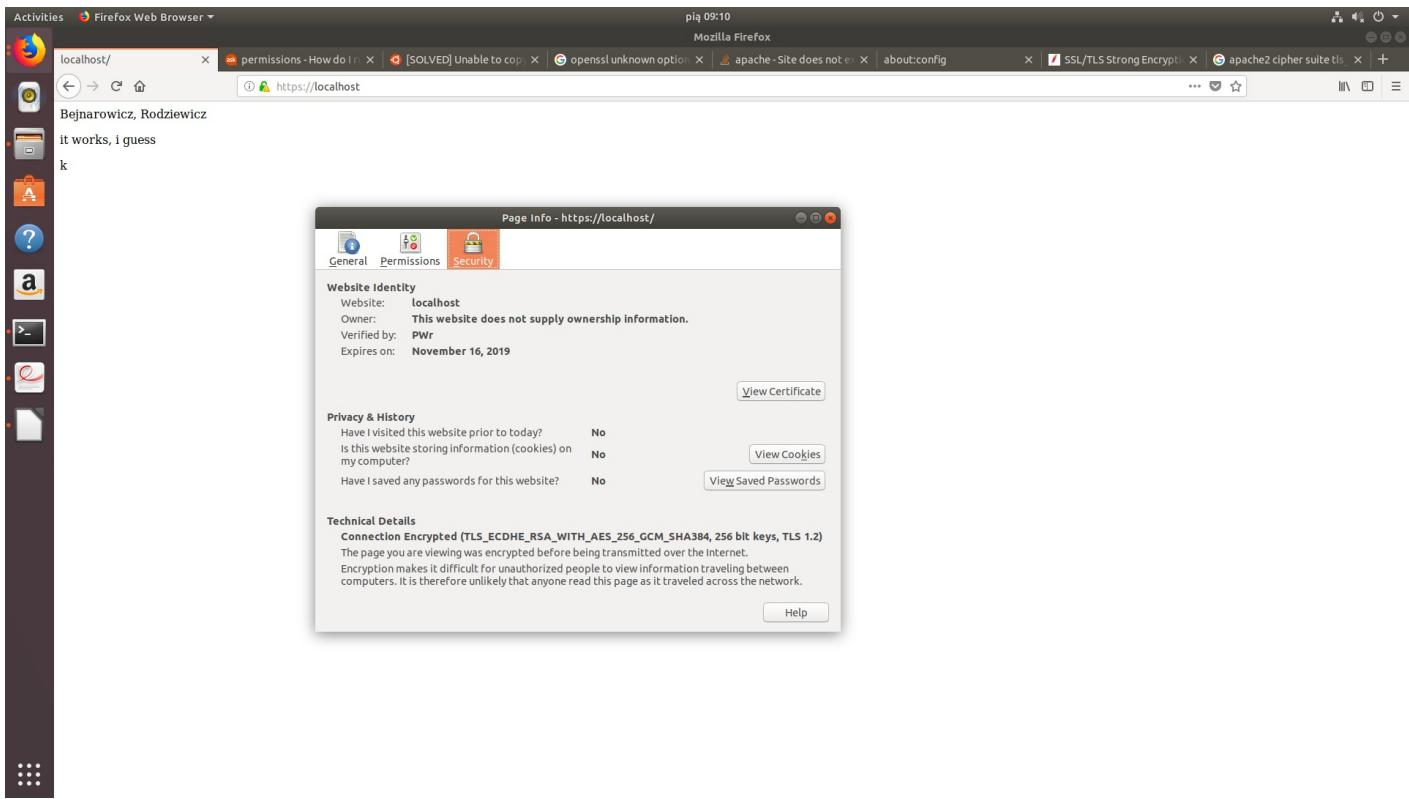
# SSL server cipher order preference:
# Use server priorities for cipher algorithm choice.
# Clients may prefer lower grade encryption. You should enable this
# option if you want to enforce stronger encryption, and can afford
# the CPU cost, and did not override SSLCipherSuite in a way that puts
# insecure ciphers first.
# Default: Off
#SSLHonorCipherOrder on

# The protocols to enable.
# Available values: all, SSLv3, TLSv1, TLSv1.1, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all -SSLv3

# Allow insecure renegotiation with clients which do not yet support
# secure renegotiation protocol. Default: Off
#SSLInsecureRenegotiation on

# Whether to forbid non-SNI clients to access name based virtual hosts.
# Default: Off
#SSLStrictSNIVHostCheck On

security.ssl.require_safe_negotiation
security.ssl.treat_unsafe_negotiation_as_broken
security.ssl.dhe_rsa_aes_128_sha
security.ssl.dhe_rsa_aes_256_sha
security.ssl.ecdhe_ecdsa_aes_128_gcm_sha256
security.ssl.ecdhe_ecdsa_aes_128_sha
security.ssl.ecdhe_ecdsa_aes_256_gcm_sha384
security.ssl.ecdhe_ecdsa_aes_256_sha
security.ssl.ecdhe_ecdsa_chachao20_poly1305_sha</IfModule>
security.ssl.ecdhe_ecdsa_aes_128_gcm_sha256
security.ssl.ecdhe_rsa_aes_128_sha
security.ssl.ecdhe_rsa_aes_256_gcm_sha384
security.ssl.ecdhe_rsa_aes_256_sha
security.ssl.ecdhe_rsa_chachao20_poly1305_sha256
security.ssl.rsa_aes_128_sha
security.ssl.rsa_aes_256_sha
security.ssl.rsa_des_ed3_sha
```

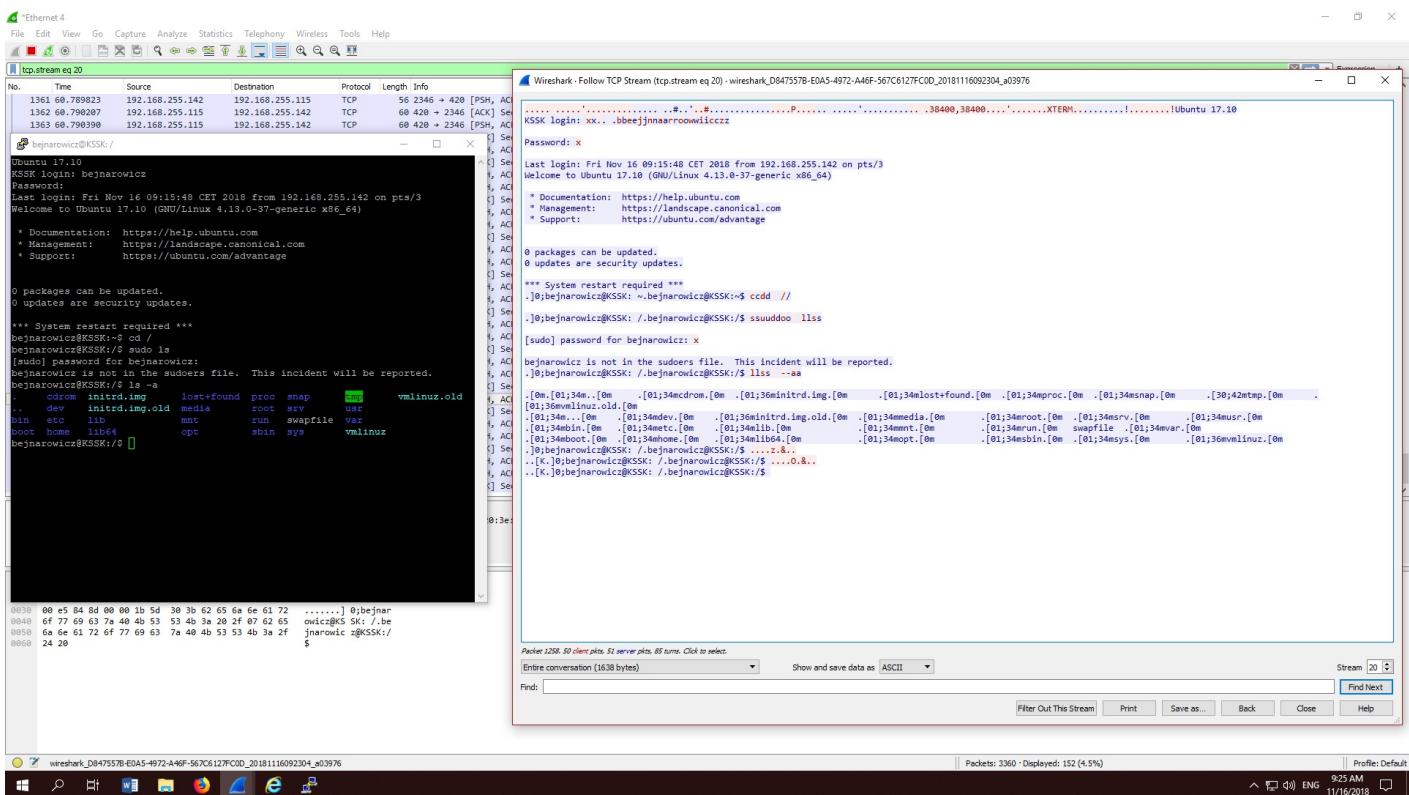


5. Stworzenie kont użytkowników w Ubuntu

Na Komputerze 1 stworzyliśmy dwa konta użytkowników.

6. Instalacja i połączenie telnet

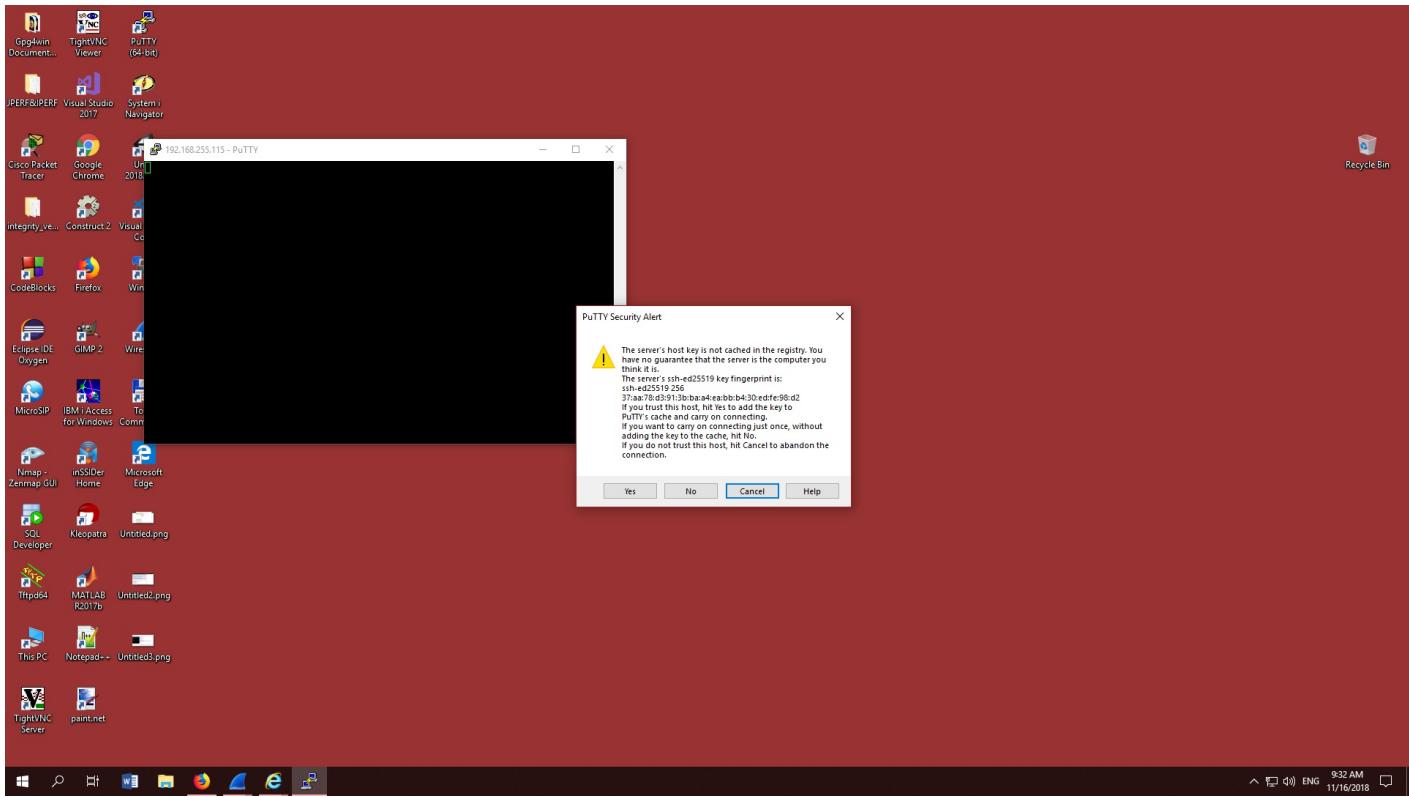
Deamon `telnet` został zainstalowany na komputerze 1. Połączylismy się na standardowy port (23) i połączenie przeszło. Zmieniliśmy port na własny (420) i również połączenie się udało. Analizując pakiety w Wiresharku widać, że połączenie w żaden sposób nie jest zabezpieczone i łatwe do przechwycenia, wraz z hasłami.



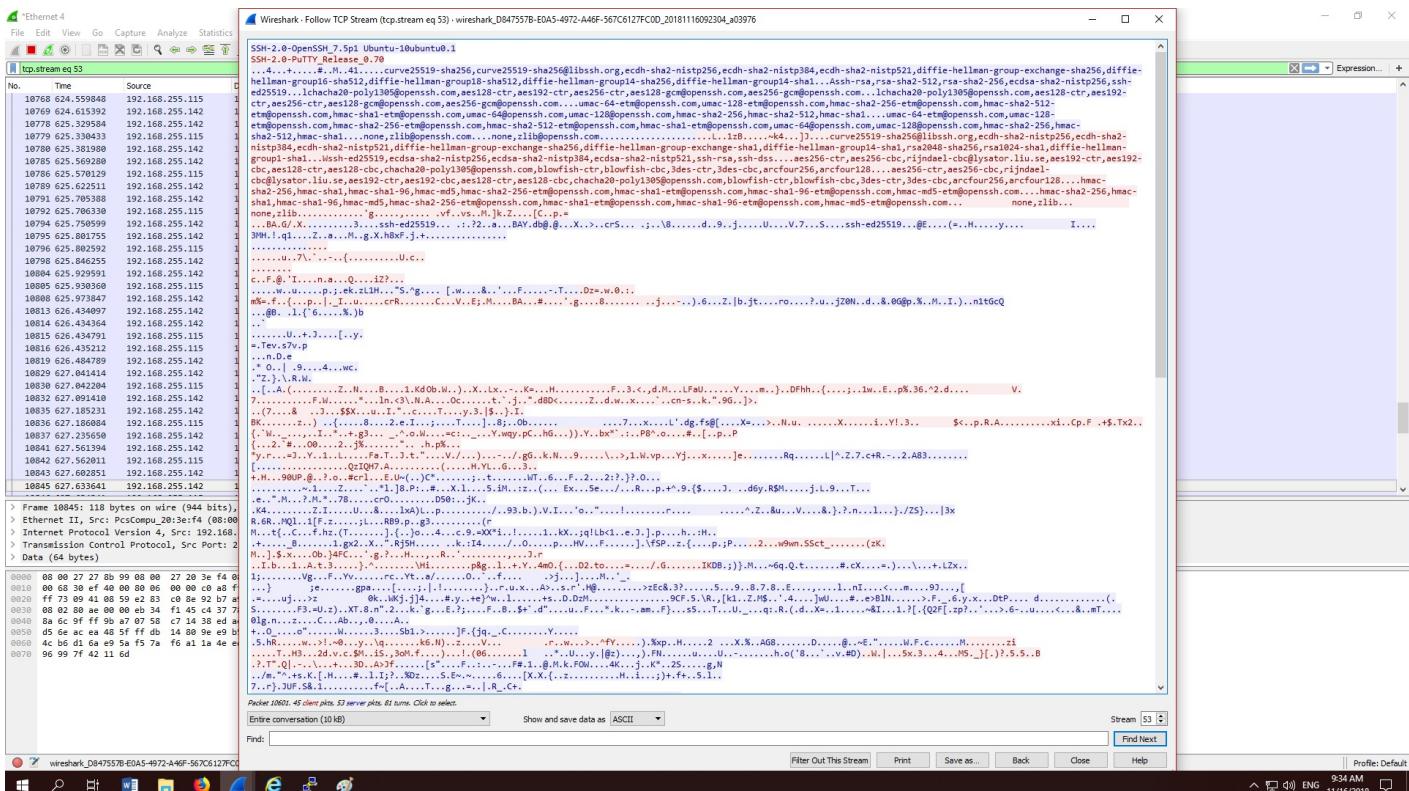
7. Połączenie ssh

Na komputerze 1 zainstalowaliśmy ssh. Port został zmieniony na 2137. Zmieniliśmy również baner powitalny i ilość dopuszczalnych błędnych logowań.

Połączenie z komputera 2 się powiodło. Potwierdzić autentyczność można weryfikując fingerprint klucza. Przy połączeniu wyświetla się on w komunikacie na komputerze z którego się łączy. Na komputerze do którego się łączy należy użyć komendy `ssh-keygen -l -E md5 -f SCIEZKA_DO_KLUCZA`.



Sprawdzając połączenie w Wiresharku widać, że jest ono szyfrowane:



8. Tunelowanie ssh

Na wykonanie tego punktu zabrakło nam czasu.