

Wrocław, 17.11.2020

# Bezpieczeństwo sieci komputerowych

## Laboratorium 3

### Bezpieczne usługi sieciowe, wirtualne sieci prywatne

Prowadzący: dr inż. Marcin Markowski

Autorka: Agnieszka Płoszaj 218353

Zadanie zostało wykonane przy wykorzystaniu 2 komputerów:

- Pierwszy komputer z systemem windows 10 z ubuntu w postaci maszyny wirtualnej (Virtualbox) - pełniący rolę serwera (adres IP: 192.168.0.150).
- Drugi komputer z systemem windows 8.1.

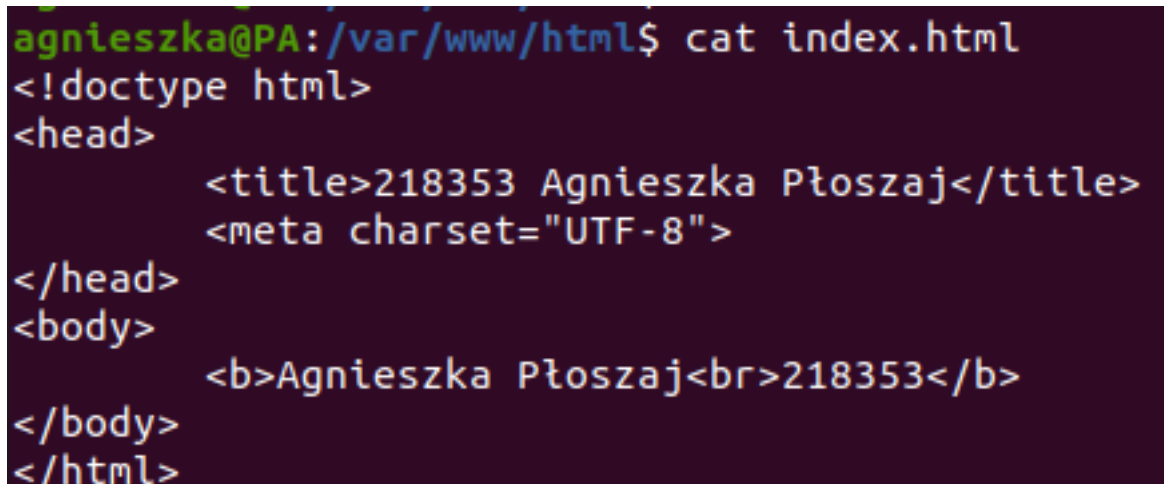
#### Zadanie 1

Na początek został zainstalowany serwer http przy pomocy polecenia

```
sudo apt install apache2
```

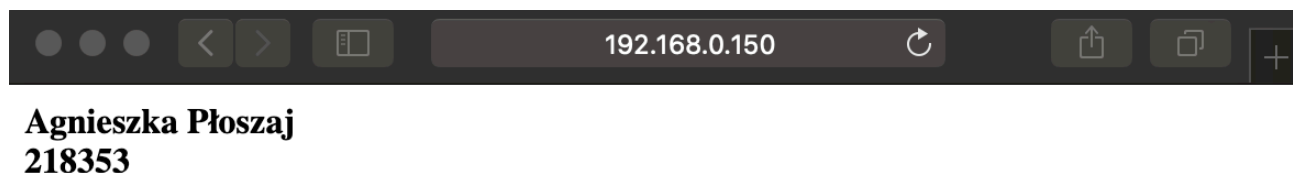
następnie została zmodyfikowana strona www, znajdująca się w katalogu /var/www/html/index.html (ilustracja 1).

Strona działa poprawnie (ilustracja 2). Następnie przy pomocy Wireshark sprawdzona została poufność transmisji. Zgodnie z oczekiwaniami, transmisja ta nie jest poufna, co widać na ilustracji 3, gdzie można zobaczyć całą zawartość strony przekazywaną jawnym tekstem.

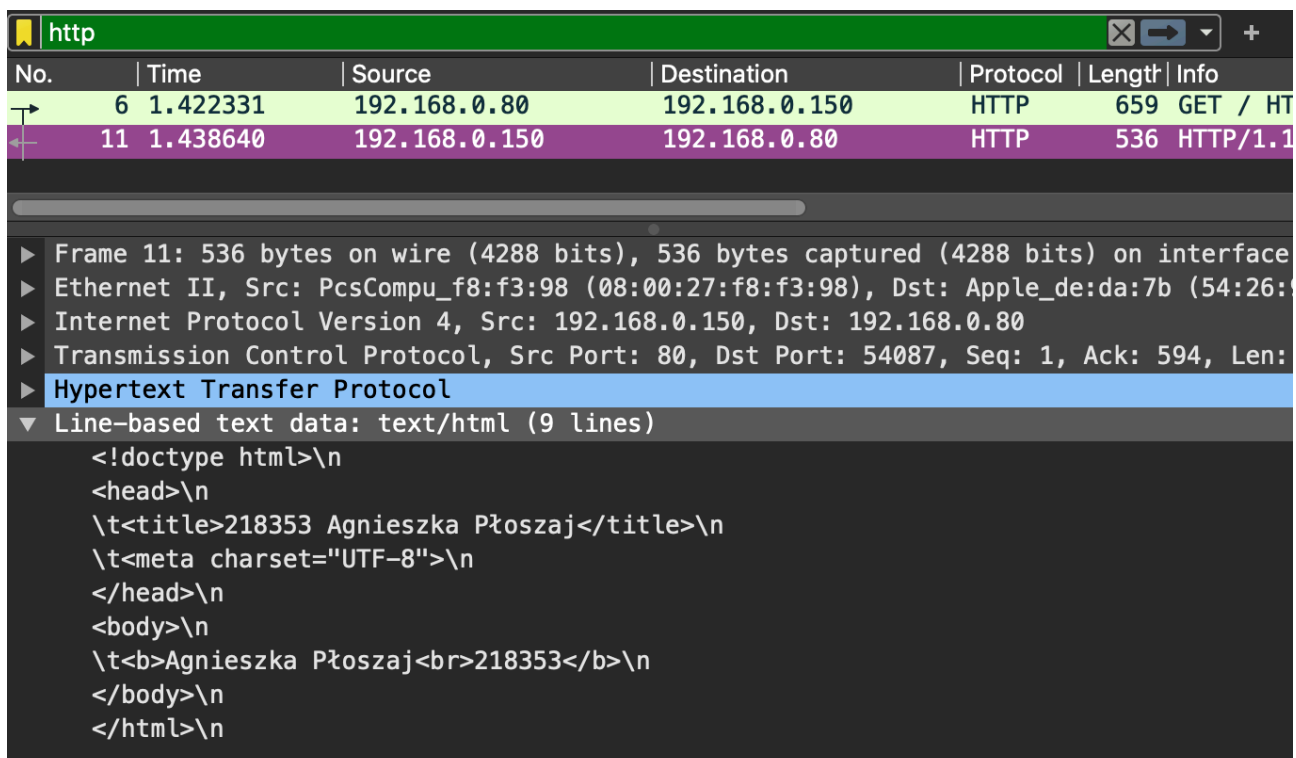
A terminal window with a dark purple background. The prompt is 'agnieszka@PA:/var/www/html\$'. The command 'cat index.html' has been executed, displaying the following HTML code:

```
<!doctype html>
<head>
    <title>218353 Agnieszka Płoszaj</title>
    <meta charset="UTF-8">
</head>
<body>
    <b>Agnieszka Płoszaj<br>218353</b>
</body>
</html>
```

Ilustracja 1 Zmodyfikowana zawartość strony www



Ilustracja 2 Sprawdzenie poprawności działania strony www



Ilustracja 3 Wireshark – http

## Zadanie 2

W następnej kolejności został zainstalowany protokół SSL. Następnie przy pomocy poleceń widocznych na ilustracji 4 zostały wygenerowane: klucz rsa oraz certyfikat ssl.

```
agnieszka@PA:/$ sudo openssl genrsa -out /etc/apache2/klucz.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
agnieszka@PA:/$ sudo openssl req -new -x509 -days 365 -key /etc/apache2/klucz.k
ey -out /etc/apache2/certyf.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PL
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Wroclaw
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:Agnieszka
Email Address []:218353@student.pwr.edu.pl
```

Ilustracja 4 Generowanie klucza rsa oraz certyfikatu ssl

W następnej kolejności sprawdzone zostało, czy usługa nasłuchuje na porcie 443 (ilustracja 5). Następnie został uruchomiony moduł ssl (ilustracja 6) oraz utworzona domyślna strona http (powstała przez modyfikację pliku default-ssl.conf). Strona została utworzona, a nowa konfiguracja aktywowana (ilustracja 7).

```
agnieszka@PA:/etc/apache2$ cat ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Ilustracja 5 Nasłuchiwanie na porcie 443

```
agnieszka@PA:/etc/apache2$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create
self-signed certificates.
To activate the new configuration, you need to run:
    systemctl restart apache2
agnieszka@PA:/etc/apache2$ systemctl restart apache2
```

Ilustracja 6 Uruchomienie modułu ssl

```
agnieszka@PA:/etc/apache2/sites-available$ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
agnieszka@PA:/etc/apache2/sites-available$ systemctl reload apache2
```

Ilustracja 7 Utworzenie i aktywacja strony

Następnie została zweryfikowana poprawność działania strony, poprzez wykorzystanie drugiego komputera (ilustracja 8).



Ilustracja 8 Sprawdzenie poprawności działania strony www

### Zadanie 3

Weryfikacja poprawności działania strony rozpoczęła się od wyświetlenia komunikatu widocznego na ilustracji 9. Jednak poprzez wybranie opcji „zaawansowane” potwierdzone zostało „zaufanie” do tego certyfikatu. Przeglądarka sygnalizuje o tym, że „połączenie nie jest bezpieczne”, ponieważ wystawiony certyfikat nie znajduje się w bazie CA, która jest „zaufaną stroną trzecią”. CA – certificate authority, to urząd certyfikacji, który potwierdza, że podany przez właściciela pary kluczy, klucz publiczny, należy do danego właściciela (wymienionego w certyfikacie). Przeglądarki przechowują listy wielu zaufanych urzędów certyfikacyjnych.

Natomiast w przypadku certyfikatu samopodpisanego (self-signed), przeglądarka nie jest w stanie sama ocenić czy ten certyfikat jest autentyczny, więc to użytkownik przeglądarki musi go zatwierdzić. Dzieje się tak dlatego, że każdy może zrobić taki certyfikat, więc dopóki użytkownik nie ma pewności, że to zaufany certyfikat nie powinien korzystać z takiej witryny, gdyż umożliwiałoby to innym możliwość zdobycia np. poufnych informacji. Algorytmy kryptograficzne wykorzystane do zestawienia tej sesji ssl zostały przedstawione na ilustracji 10.

Utworzony w zadaniu 2 certyfikat można zobaczyć na ilustracji 11.

W pierwszej sekcji zostały przedstawione informacje dotyczące *nazwy podmiotu*, w drugiej sekcji została przedstawiona *nazwa wystawcy*. W obu tych przypadkach zostały podane dane takie jak: państwo, region, nazwa poсполita oraz adres e-mail.

Sekcja trzecia to *ważność*, która zawiera przedziały czasowe, w których danych certyfikat uważany jest za ważny.

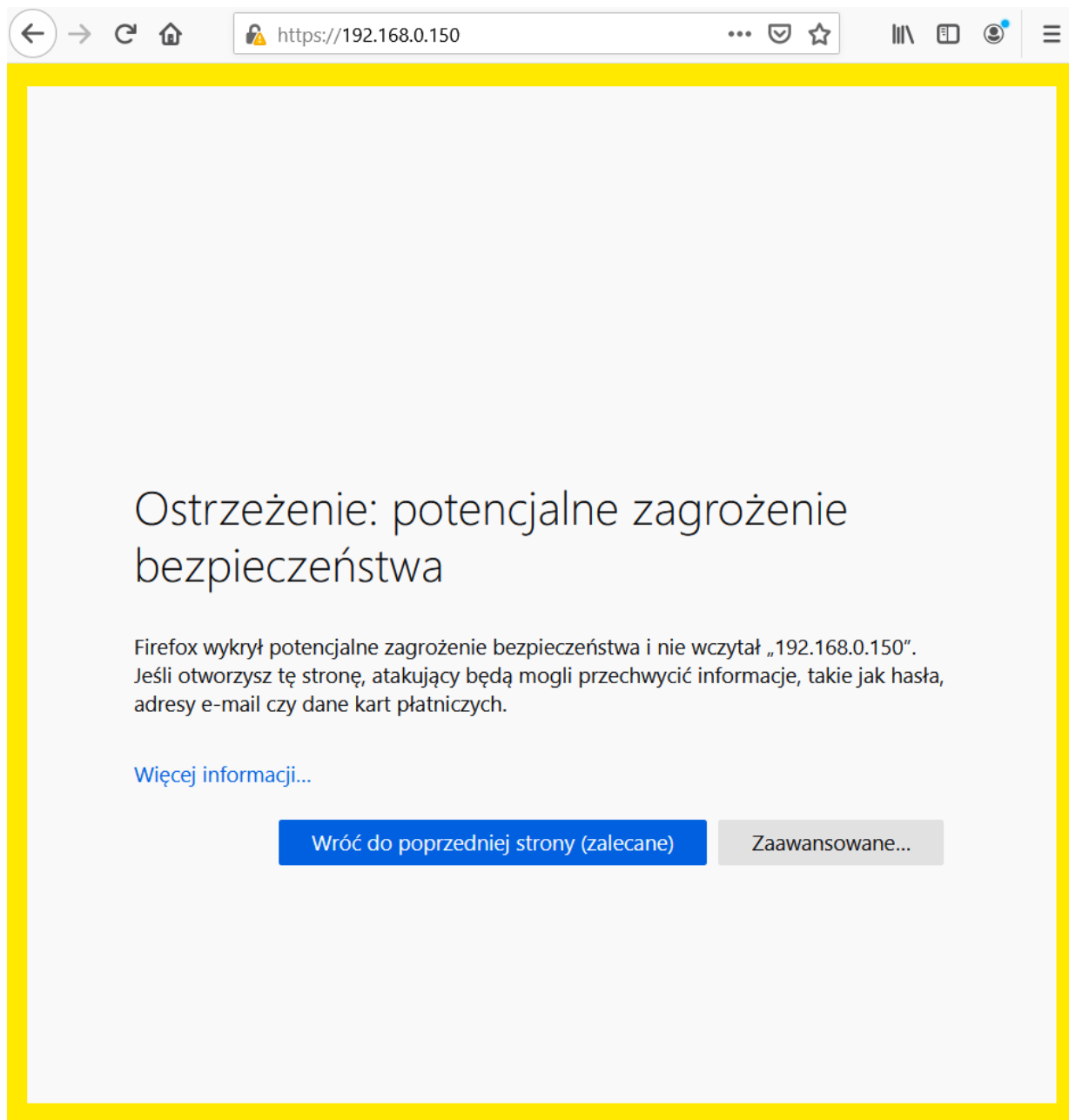
Czwarta sekcja to *informacje o kluczu publicznym*. Podane informacje to algorytm, rozmiar klucza, wykładnik oraz modulo.

Piąta sekcja to *różne*. Zawiera numer seryjny oraz algorytm podpisu.

Szósta sekcja to *odciski* (SHA-256 oraz SHA-1).

Siódma sekcja to *podstawowe ograniczenia*, gdzie można zobaczyć potwierdzenie organu certyfikacji.

Ostatnie dwie sekcje to *identyfikatory klucza podmiotu* oraz *klucza organu*.



Ilustracja 9 Informacja o potencjalnym zagrożeniu

#### Szczegóły techniczne

Połączenie szyfrowane (TLS\_AES\_128\_GCM\_SHA256, 128-bitowe klucze, TLS 1.3)

Wyświetlana strona została zaszyfrowana przed przesłaniem poprzez Internet.

Szyfrowanie utrudnia osobom nieupoważnionym odczytanie informacji przesyłanych między komputerami. Dlatego jest mało prawdopodobne, aby ktokolwiek przeczytał tę stronę, gdy podróżowała przez sieć.

Ilustracja 10 Algorytmy kryptograficzne wykorzystane do zestawienia sesji ssl

<b>Nazwa podmiotu</b>	
Państwo	PL
Region	Wrocław
Nazwa pospolita	Agnieszka
Adres e-mail	218353@student.pwr.edu.pl
<b>Nazwa wystawcy</b>	
Państwo	PL
Region	Wrocław
Nazwa pospolita	Agnieszka
Adres e-mail	218353@student.pwr.edu.pl
<b>Ważność</b>	
Nieważny przed	17.11.2020, 15:43:48 (czas środkowoeuropejski standardowy)
Nieważny po	17.11.2021, 15:43:48 (czas środkowoeuropejski standardowy)
<b>Informacje o kluczu publicznym</b>	
Algorytm	RSA
Rozmiar klucza	2048
Wykładnik	65537
Modulo	A4:83:A5:A2:E2:C1:B3:DF:E1:42:A9:E4:AF:0E:75:F2:41:C2:EC:6C:81:49:F1:F3:94:71:...
<b>Różne</b>	
Numer seryjny	5E:8E:1F:AC:13:F1:11:83:F7:05:29:88:D2:96:18:37:16:60:E6:C4
Algorytm podpisu	SHA-256 with RSA Encryption
Wersja	3
Pobierz	<a href="#">PEM (certyfikat)</a> <a href="#">PEM (łańcuch)</a>
<b>Odciski</b>	
SHA-256	71:8E:66:51:4F:00:45:DD:48:20:33:7C:3C:75:91:DE:BE:01:18:12:AC:D9:90:30:1E:82:...
SHA-1	0C:54:3B:AC:E4:DE:82:F1:43:FE:38:50:F1:44:B9:85:27:8A:7A:FC
<b>● Podstawowe ograniczenia</b>	
Organ certyfikacji	Tak
<b>Identyfikator klucza podmiotu</b>	
Identyfikator klucza	5D:B1:D9:34:F9:64:06:94:56:51:E0:FE:B3:E7:5A:B1:0E:D1:4E:FC
<b>Identyfikator klucza organu</b>	
Identyfikator klucza	5D:B1:D9:34:F9:64:06:94:56:51:E0:FE:B3:E7:5A:B1:0E:D1:4E:FC

Ilustracja 11 Certyfikat

Następnie przy pomocy programu Wireshark zostało ocenione bezpieczeństwo połączenia ssl (ilustracja 12 i 13). Przesyłane treści są szyfrowane. Certyfikat ma jawne informacje dotyczące SHA-1 oraz klucz publiczny.

No.	Time	Source	Destination	Protocol	Length	Info
27	1.860664	192.168.0.129	34.107.221.82	HTTP	360	GET /success.txt HTTP/1
31	1.898843	34.107.221.82	192.168.0.129	HTTP	274	HTTP/1.1 200 OK (text/
32	1.902250	192.168.0.129	34.107.221.82	HTTP	365	GET /success.txt?ipv4 t
34	1.924723	34.107.221.82	192.168.0.129	HTTP	274	HTTP/1.1 200 OK (text/
184	28.768901	192.168.0.129	54.230.228.8	OCSP	452	Request
191	28.891440	54.230.228.8	192.168.0.129	OCSP	1060	Response
642	31.221178	192.168.0.129	216.58.215.99	OCSP	449	Request
673	31.265728	216.58.215.99	192.168.0.129	OCSP	755	Response
1415	86.200640	192.168.0.129	216.58.215.99	OCSP	450	Request
1417	86.245396	216.58.215.99	192.168.0.129	OCSP	756	Response

Transmission Control Protocol, Src Port: 18333, Dst Port: 80, Seq: 1, Ack: 1, Len: 398

Hypertext Transfer Protocol

Online Certificate Status Protocol

- tbsRequest
  - requestList: 1 item
    - Request
      - reqCert
        - hashAlgorithm (SHA-1)
          - Algorithm Id: 1.3.14.3.2.26 (SHA-1)
          - issuerNameHash: 33f5aac61d66e7055d03173a4d1f3e187138850d
          - issuerKeyHash: 59a4660652a07b95923ca394072796745bf93dd0
          - serialNumber: 0x0d7225a84e0c459ea70e77d602aed0a1

Ilustracja 12 Wireshark – SSL

No.	Time	Source	Destination	Protocol	Length	Info
27	1.860664	192.168.0.129	34.107.221.82	HTTP	360	GET /success.txt HTTP/1.1
31	1.898843	34.107.221.82	192.168.0.129	HTTP	274	HTTP/1.1 200 OK (text/plain)
32	1.902250	192.168.0.129	34.107.221.82	HTTP	365	GET /success.txt?ipv4 HTTP/1.1
34	1.924723	34.107.221.82	192.168.0.129	HTTP	274	HTTP/1.1 200 OK (text/plain)
184	28.768901	192.168.0.129	54.230.228.8	OCSP	452	Request
191	28.891440	54.230.228.8	192.168.0.129	OCSP	1060	Response
642	31.221178	192.168.0.129	216.58.215.99	OCSP	449	Request
673	31.265728	216.58.215.99	192.168.0.129	OCSP	755	Response
1415	86.200640	192.168.0.129	216.58.215.99	OCSP	450	Request
1417	86.245396	216.58.215.99	192.168.0.129	OCSP	756	Response

Content-Length: 8\r\n  
 Via: 1.1 google\r\n  
 Age: 35198\r\n  
 Cache-Control: public, must-revalidate, max-age=0, s-maxage=86400\r\n  
 \r\n  
 [HTTP response 1/1]  
 [Time since request: 0.022473000 seconds]  
[\[Request in frame: 32\]](#)  
 [Request URI: http://detectportal.firefox.com/success.txt?ipv4]  
 File Data: 8 bytes

Line-based text data: text/plain (1 lines)

success\n

Ilustracja 13 Wireshark – SSL



Na początek została wyświetlona lista zestawów kryptograficznych w przeglądarce (ilustracja 14). Następnie na serwerze została użyta komenda

`openssl ciphers -s`

(Ilustracja 15). Korzystając z list przedstawionych na ilustracjach 14 i 15, wybrany został jeden zestaw algorytmów, tak jak zostało to pokazane na ilustracji 16. Następnie zostało nawiązane połączenie. Parametry kryptograficzne sesji można zobaczyć na ilustracji 17.

security.ssl.treat_unsafe_negotiation_as_broken	false	1
security.ssl3.dhe_rsa_aes_128_sha	false	1
security.ssl3.dhe_rsa_aes_256_sha	false	1
security.ssl3.ecdhe_ecdsa_aes_128_gcm_sha256	true	1
security.ssl3.ecdhe_ecdsa_aes_128_sha	true	1
security.ssl3.ecdhe_ecdsa_aes_256_gcm_sha384	true	1
security.ssl3.ecdhe_ecdsa_aes_256_sha	true	1
security.ssl3.ecdhe_ecdsa_chacha20_poly1305_sha256	true	1
security.ssl3.ecdhe_rsa_aes_128_gcm_sha256	true	1
security.ssl3.ecdhe_rsa_aes_128_sha	true	1
security.ssl3.ecdhe_rsa_aes_256_gcm_sha384	true	1
security.ssl3.ecdhe_rsa_aes_256_sha	true	1
security.ssl3.ecdhe_rsa_chacha20_poly1305_sha256	true	1
security.ssl3.rsa_aes_128_gcm_sha256	true	1
security.ssl3.rsa_aes_128_sha	true	1
security.ssl3.rsa_aes_256_gcm_sha384	true	1
security.ssl3.rsa_aes_256_sha	true	1
security.ssl3.rsa_des_ede3_sha	true	1

Ilustracja 14 Lista zestawów kryptograficznych przeglądarki

```

agnieszka@PA:/etc/apache2/mods-available$ openssl ciphers -s
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:ECDH
E-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384
:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POL
Y1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-
GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES256-SHA
256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:ECD
HE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:ECDHE-ECDSA-AES128-
SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:AES256-GCM-SHA384:AES128-GCM-SHA256
:AES256-SHA256:AES128-SHA256:AES256-SHA:AES128-SHA

```

Ilustracja 15 Lista zestawów kryptograficznych serwera

```

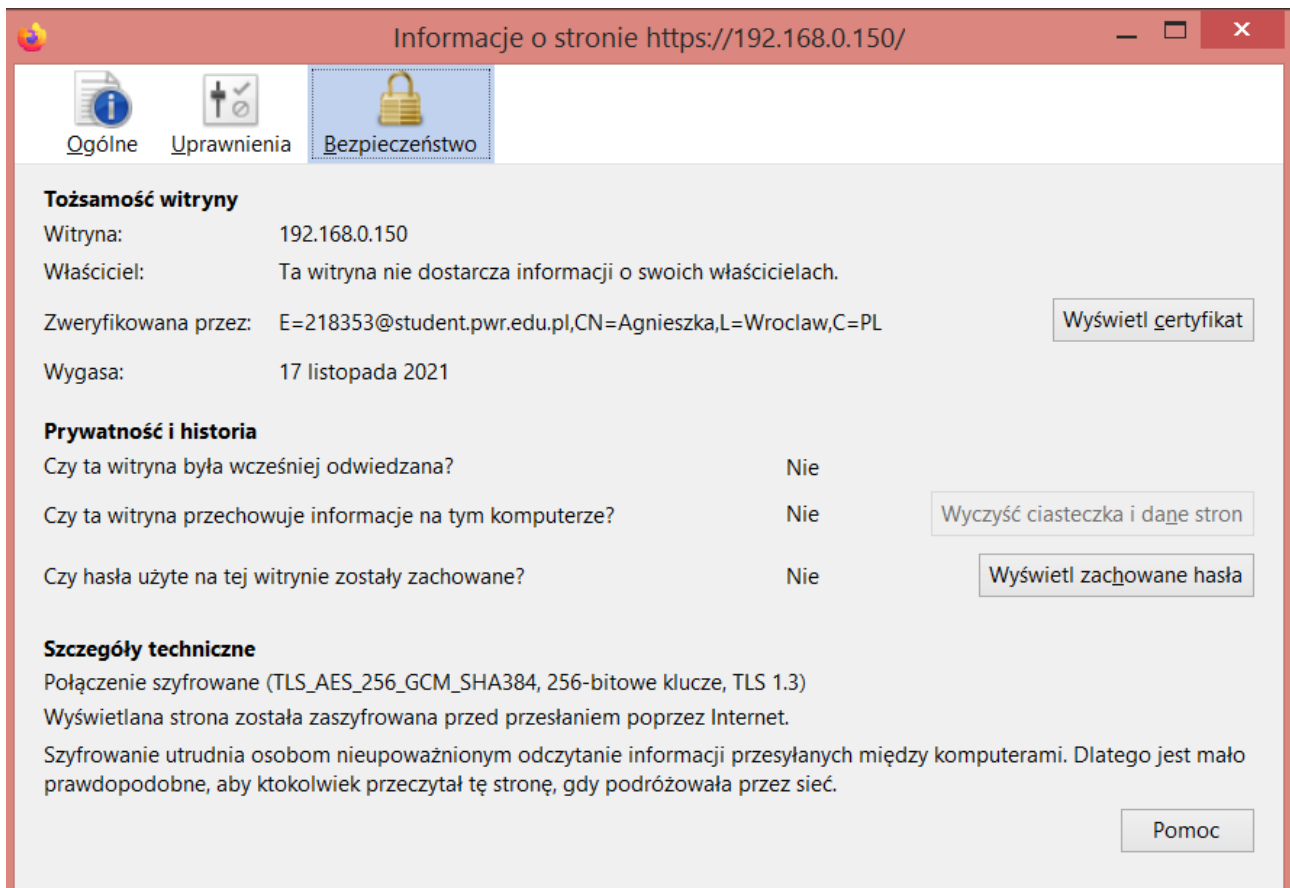
GNU nano 4.8                                ssl.conf
# Configure the SSL Session Cache: First the mechanism
# to use and second the expiring timeout (in seconds).
# (The mechanism dbm has known memory leaks and should not be used).
#SSLSessionCache                                dbm:${APACHE_RUN_DIR}/ssl_scache
SSLSessionCache                                shmcb:${APACHE_RUN_DIR}/ssl_scache(512000)
SSLSessionCacheTimeout 300

# Semaphore:
# 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 c
# this)
#Mutex file:${APACHE_LOCK_DIR}/ssl_mutex ssl-cache

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate. See th
# ciphers(1) man page from the openssl package for list of all avail
# options.
# Enable only secure ciphers:
#SSLCipherSuite HIGH:!aNULL
SSLCipherSuite ECDHE-RSA-AES256-SHA384
# SSL server cipher order preference:
# Use server priorities for cipher algorithm choice.
# Clients may prefer lower grade encryption. You should enable this

```

Ilustracja 16 Wybrany zestaw kryptograficzny



Ilustracja 17 Wybrany zestaw kryptograficzny

## Zadanie 5

Do utworzenia kont 2 nowych użytkowników zostało użyte polecenie `adduser` (ilustracja 18). Użytkownicy zostali nazwani kolejno: `ploszaj1` i `ploszaj2`.

```
agnieszka@PA:/$ sudo adduser ploszaj1
Dodawanie użytkownika "ploszaj1"...
Dodawanie nowej grupy "ploszaj1" (1003)...
Dodawanie nowego użytkownika "ploszaj1" (1003) w grupie "ploszaj1"...
Tworzenie katalogu domowego "/home/ploszaj1"...
Kopiowanie plików z "/etc/skel" ...
Nowe hasło :
Proszę ponownie wpisać nowe hasło :
passwd: hasło zostało zmienione
Zmieniam informację o użytkowniku ploszaj1
Wpisz nową wartość lub wciśnij ENTER by przyjąć wartość domyślną
    Imię i nazwisko []:
    Numer pokoju []:
    Telefon do pracy []:
    Telefon domowy []:
    Inne []:
Czy informacja jest poprawna? [T/n] t
```

Ilustracja 18 Dodanie nowego użytkownika

Następne zadanie polegało na zainstalowaniu oraz uruchomieniu demona telnet. Zgodnie z instrukcją zostały na początek użyte 2 polecenia:

```
sudo apt-get install telnetd
```

oraz

```
sudo /etc/init.d/openbsd-inetd restart
```

(Ilustracja 19). Zmiana numeru portu została pokazana na ilustracji 21. Nowy numer portu telnet to w tym przypadku 24. Następnie za pomocą programu *Putty* zostało nawiązane połączenie z serwerem (ilustracja 20) i przy pomocy programu *Wireshark* została sprawdzona poufność danych. Zgodnie z oczekiwaniami zarówno login, jak i użyte komendy możemy zobaczyć w formie jawnego tekstu (ilustracja 20). Podsumowując, telnet nie zapewnia poufności danych.

```
agnieszka@PA:/$ sudo /etc/init.d/openbsd-inetd restart
Restarting openbsd-inetd (via systemctl): openbsd-inetd.service.
```

Ilustracja 19 Zainstalowanie oraz uruchomienie demona telnet

The left screenshot shows a Wireshark packet capture of a telnet session. The packet list on the left shows a TELNET packet (No. 121) from 192.168.0.150 to 192.168.0.129. The packet details pane shows the TELNET data structure, including the 'Data' field which contains the login prompt and the user's input 'ploszaj1@PA:~\$'. The packet bytes pane shows the raw data, including the telnet command sequence and the user's input.

The right screenshot shows the terminal output of the telnet client on a Ubuntu 20.04.1 LTS system. The output shows the login prompt 'PA login: ploszaj1', the password prompt 'Password:', and the user's input 'ploszaj1@PA:~\$'. The terminal also shows the system's update status and the user's hardware enablement stack (HWE) information.

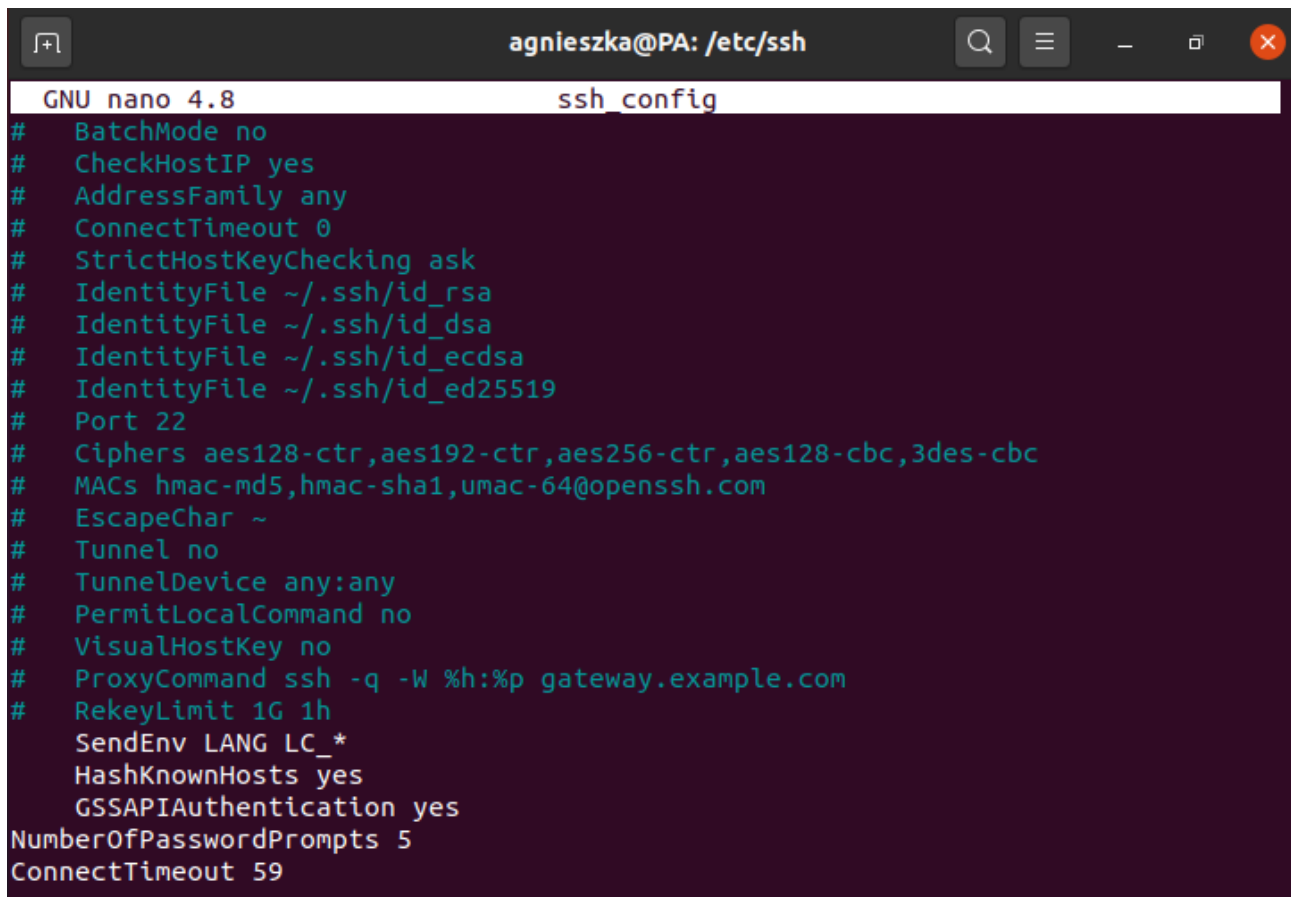
Ilustracja 20 Połączenie z serwerem – telnet

Ostatnie zadanie polegało na zainstalowaniu ssh na serwerze (polecenie: `sudo apt install openssh-server`), zmianie domyślnego numeru portu oraz modyfikacji 2 parametrów ssh. Numer portu spoza puli well-known-ports (czyli od 0 do 1023), który został wybrany to 2005 (ilustracja 21). Następnie dokonana została modyfikacja 2 parametrów. Pierwszym z nich był **ConnectTimeout 59**, który określa limit czasu w sekundach podczas łączenia się z serwerem, natomiast drugim parametrem było **NumberOfPasswordPrompts 5**, które określa liczbę możliwych nieudanych prób podania hasła (ilustracja 22).

```

GNU nano 4.8          services          Zmodyfikowany
discard      9/udp      sink null
sysstat      11/tcp     users
daytime      13/tcp
daytime      13/udp
netstat      15/tcp
qotd         17/tcp     quote
chargen      19/tcp     ttytst source
chargen      19/udp     ttytst source
ftp-data     20/tcp
ftp          21/tcp
fsp         21/udp     fspd
#ssh         22/tcp          # SSH Remote Login Protocol
ssh         2005/tcp
#telnet      23/tcp
telnet      24/tcp
smtp        25/tcp     mail
time        37/tcp     timserver
time        37/udp     timserver
whois       43/tcp     nicname
tacacs      49/tcp          # Login Host Protocol (TACACS)
tacacs      49/udp
domain      53/tcp          # Domain Name Server
domain      53/udp
bootps      67/udp
bootpc      68/udp
  
```

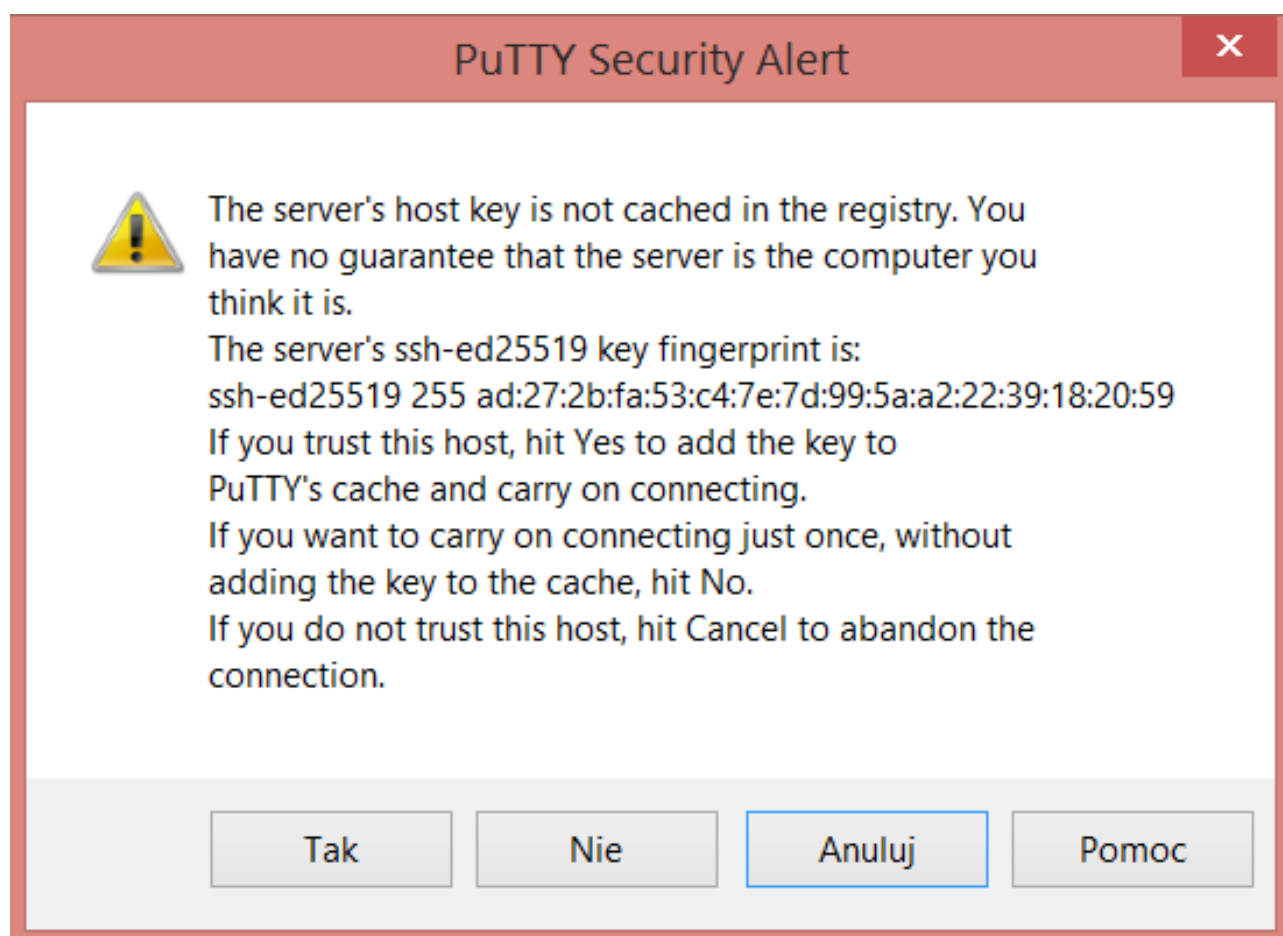
Ilustracja 21 Zmiana numerów portów ssh oraz telnet



```
agnieszka@PA: /etc/ssh
GNU nano 4.8 ssh_config
# BatchMode no
# CheckHostIP yes
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# IdentityFile ~/.ssh/id_ecdsa
# IdentityFile ~/.ssh/id_ed25519
# Port 22
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com
# EscapeChar ~
# Tunnel no
# TunnelDevice any:any
# PermitLocalCommand no
# VisualHostKey no
# ProxyCommand ssh -q -W %h:%p gateway.example.com
# RekeyLimit 1G 1h
# SendEnv LANG LC_*
# HashKnownHosts yes
# GSSAPIAuthentication yes
NumberOfPasswordPrompts 5
ConnectTimeout 59
```

Ilustracja 22 Zmiana dwóch parametrów

Z komputera drugiego przy pomocy programu Putty zostało nawiązane połączenie. Autentyczność serwera została potwierdzona przez użytkownika, klikając odpowiednią opcję w wyświetlonym alercie bezpieczeństwa (ilustracja 23). Następnie korzystając z programu Wireshark została sprawdzona poufność przesyłanych danych (ilustracja 24). W przeciwieństwie do protokołu telnet, ssh nie podaje jawnym tekstem loginu, hasła ani komend. Podsumowując, protokół ssh zapewnia poufność.



Ilustracja 23 Alert bezpieczeństwa



No.	Time	Source	Destination	Protocol	Length	Info
58	8.854977	192.168.0.129	192.168.0.150	SSHv2	82	Client: Protocol (S
61	8.910356	192.168.0.150	192.168.0.129	SSHv2	95	Server: Protocol (S
62	8.960169	192.168.0.129	192.168.0.150	SSHv2	1222	Client: Key Exchange
63	8.979152	192.168.0.150	192.168.0.129	SSHv2	1110	Server: Key Exchange
64	8.987943	192.168.0.129	192.168.0.150	SSHv2	102	Client: Elliptic Cu
67	9.023496	192.168.0.150	192.168.0.129	SSHv2	262	Server: Elliptic Cu
118	29.436044	192.168.0.129	192.168.0.150	SSHv2	134	Client: New Keys, E
119	29.448459	192.168.0.150	192.168.0.129	SSHv2	118	Server: Encrypted p
134	35.367826	192.168.0.129	192.168.0.150	SSHv2	134	Client: Encrypted p
135	35.407368	192.168.0.150	192.168.0.129	SSHv2	134	Server: Encrypted p
149	40.486054	192.168.0.129	192.168.0.150	SSHv2	326	Client: Encrypted p
150	40.526160	192.168.0.150	192.168.0.129	SSHv2	102	Server: Encrypted p
151	40.527964	192.168.0.129	192.168.0.150	SSHv2	134	Client: Encrypted p
156	42.410891	192.168.0.150	192.168.0.129	SSHv2	710	Server: Encrypted p

▶ Transmission Control Protocol, Src Port: 18701, Dst Port: 22, Seq: 1245, Ack: 1306, Len: 80  
 ▲ SSH Protocol
 

- ▶ SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
  - Packet Length: 12
  - Padding Length: 10
  - ▶ Key Exchange
    - Padding String: a6a2b6ee6b9dde4c9484
- ▶ SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
  - Packet Length (encrypted): 7f6b1ad8
  - Encrypted Packet: 37fbb444e24f524a3ae193e3b953e31f51e0f21b213d3fed...
  - MAC: 022033c22ec5ee59a74ac877bd374afd0bcf0d6ae36eddab...
  - [Direction: client-to-server]

Ilustracja 24 Wireshark – SSH