

VIII Seminarium PLOUG
Warszawa
Kwiecień 2003

Advanced Security Option i inne metody szyfrowania połączeń w Oracle 9i

Marcin Przepiórowski

marcin.przepiorowski@alkom.com.pl

Altkom Akademia S.A.

1. ZAGROŻENIA

Przetwarzanie danych w środowiskach rozproszonych może wiązać się między innymi z następującymi zagrożeniami:

- **Kradzież danych**
Wraz z wzrostem wielkości sieci rośnie zagrożenie związane z przechwyceniem przesyłanych pakietów danych. W szczególności może to dotyczyć np. haseł użytkowników.
- **Modyfikacja transakcji**
W słabo zabezpieczonych sieciach istnieje możliwość modyfikacji przepływających informacji np. zmiana kwoty w transakcji finansowej.
- **Falszowanie tożsamości**
Osoba niepowołana podszywa się pod użytkownika zdobywając dostęp do informacji lub pod serwer, do którego użytkownik próbuje uzyskać dostęp.
- **Nadmierna ilość haseł**
Użytkownicy pracujący w środowisku rozproszonym, mający dostęp do wielu usług sieciowych borykają się z problemem zapamiętania wielu haseł. W wyniku czego zapisują hasła, wybierają hasła trywialne lub jednakowe hasło do wszystkich usług. Ponadto administracja takimi systemami jest bardzo pracochłonna. Wiąże się z koniecznością zarządzania wieloma kontami dla tego samego użytkownika.

2. CECHY ORACLE ADVANCED SECURITY

2.1. Ogólne informacje o pakiecie *Oracle Advanced Security*

Oracle Advanced Security to pakiet rozszerzający możliwości konfiguracji sieci Oracle Net o opcje związane z poprawą bezpieczeństwa. Pakiet ten wzbogaca możliwości sieci Oracle Net o następujące cechy:

- **Poufność danych (*data privacy*)**
Poufność transmitowanych danych jest realizowana przy pomocy szyfrowania. *Oracle Advanced Security* pozwala na stosowanie różnych algorytmów szyfrujących w szczególności algorytmów RSA i DES.
- **Spójność danych (*data integrity*)**
Aby zapewnić spójność (niemodyfikowalność) transmitowanych danych *Oracle Advanced Security* pozwala na zastosowanie jednego z dwóch algorytmów haszujących: SHA lub MD5.
Algorytmy te chronią przed następującymi formami ataku:
 - modyfikacją danych
 - przechwyceniem pakietów
 - powtarzaniem transakcji
- **Mechanizmy uwierzytelniające (*authentication*)**
silne uwierzytelnienie użytkowników poprzez SSL lub zewnętrzne mechanizmy uwierzytelniające.

- **Single Sign-On** – jedna metoda uwierzytelniająca (np. hasło) umożliwia dostęp do wielu usług.

3. POUFNOŚĆ I SPÓJNOŚĆ – OMÓWNIENIE

3.1. Zapewnienie poufności transmisji – szyfrowanie ruchu

Szyfrowanie polega na konwersji danych z postaci czystego tekstu (*plaintext*) na postać zaszyfrowaną (*ciphertext*) przy pomocy klucza. Rozszyfrowanie wiadomości bez znajomości klucza powinno być bardzo trudne (obliczeniowo nieopłacalne). W systemie symetrycznym ten sam klucz służy zarówno do szyfrowania jak i do rozszyfrowania wiadomości. *Oracle Advanced Security* pozwala na stosowanie następujących algorytmów symetrycznych:

- DES (Data Encryption Standard)
Standard używany w USA i na całym świecie od wielu lat. Długość klucza 56 bitów.
- 3DES (Triple DES)
Potrójne szyfrowanie algorytmem DES. Wysoki stopień bezpieczeństwa kosztem spadku wydajności (kosztowny obliczeniowo). Dostępne są dwie wersje :
 - two-key właściwa długość klucza 112 bitów
 - three-key właściwa długość klucza 168 bitów
- DES40
Dostępny we wszystkich wcześniejszych wersjach Oracle Advanced Security, Oracle Advanced Networking Option i Secure Network Services. Początkowo był to algorytm eksportowy, obecnie prawo USA zastało złagodzone i można używać poza USA również silniejszych szyfrów. Bazuje na kluczu o długości 40 bitów. Zachowany dla wstecznej kompatybilności. Używanie tego algorytmu nie jest obecnie zalecane ze względu na stosunkowo niskie bezpieczeństwo i powszechną dostępność silniejszych algorytmów.
- RSA RC4
Standard opracowany przez RSA Data Security Inc. zoptymalizowany pod kątem szybkości działania. Kilkakrotnie szybszy od DES, pozwala na szyfrowanie dużych ilości danych niewielkim kosztem. Podczas działania dynamicznie zmienia długość klucza. Dostępne są wersje o następujących długościach klucza: 40, 56, 128, 256 bitów.

3.2. Zapewnienie spójności danych

Omówione powyżej szyfrowanie zapewnia poufność przesyłanych informacji, *Oracle Advanced Security* stwarza możliwość ochrony również przed innymi formami ataku na przesyłane przez sieć dane:

- modyfikacja danych – kiedy osoba postronna zmienia treść wiadomości
- powtarzanie transmisji – np. kiedy osoba postronna przechwytuje legalną transakcję i powtarza ją kilkakrotnie.

Ochrona przed tego typu atakami jest realizowana przy wykorzystaniu funkcji skrótu (*hash*)

ASO udostępnia dwa algorytmy:

- MD5 (Message Digest 5 Algorithm)
- SHA-1 (Secure Hash Algorithm)

Oba algorytmy zaopatrują pakiety danych w sumy kontrolne zabezpieczając przed modyfikacją danych i zapewniając unikalność każdej transakcji. Mechanizm ten działa niezależnie od mechanizmów szyfrujących, tzn. można stosować dowolne kombinacje algorytmów szyfrujących i haszujących.

3.3. Zarządzanie kluczami

Poufność transmitowanych danych zależy od poufności klucza używanego do szyfrowania po obu stronach. W rozproszonym środowisku, kiedy z szyfrowanych transmisji korzysta wielu użytkowników ręczne zarządzanie kluczami byłoby zadaniem trudnym i stwarzało zagrożenia. Do bezpiecznej dystrybucji kluczy Oracle Advanced Security wykorzystuje algorytm „Diffie-Hellman key negotiation algorithm”, zarówno do szyfrowania jak i haszowania transmisji. Aby podnieść bezpieczeństwo, klucze są zmieniane i unikalne dla każdej sesji. Klient rozpoczyna komunikację z serwerem używając klucza generowanego przez algorytm Diffie-Hellman. Kiedy autentkuje się do serwera ustalony zostaje wspólny klucz (*shared secret*). Z połączenia obu kluczy powstaje klucz sesji wykorzystywany podczas transmisji. Cały mechanizm zarządzania i negocjacji kluczy odbywa się bez udziału użytkownika i nie wymaga żadnej konfiguracji.

4. KONFIGURACJA SZYFROWANIA I SPÓJNOŚCI DANYCH

Zarówno klient jak i serwer mogą obsługiwać więcej niż jeden algorytm szyfrujący i haszujący. Podczas nawiązywania połączenia serwer decyduje czy i jakich algorytmów użyć. Lista wybranych algorytmów znajduje się w pliku `sqlnet.ora` na serwerze i kliencie. Konfigurację przeprowadzamy przy pomocy *Oracle Net Manager*, lub ręcznie w pliku `sqlnet.ora`.

Parametr dla klienta Oracle:

```
SQLNET.ENCRYPTION_TYPES_CLIENT=(lista algorytmów)
```

Parametr dla serwera Oracle:

```
SQLNET.ENCRYPTION_TYPES_SERVER=(lista algorytmów)
```

Serwer wybiera zawsze pierwszy algorytm ze swojej listy, który jest dostępny również u klienta, dlatego należy wybrane algorytmy podawać w kolejności zgodnej z preferencjami. Jeżeli jedna ze stron nie wyspecyfikuje wybranych algorytmów, domyślnie brane są pod uwagę wszystkie zainstalowane. Podczas konfiguracji sieci na kliencie i serwerze można wybrać dowolne lub wszystkie dostępne algorytmy, lecz podczas negocjacji połączenia wybierany jest zawsze tylko jeden algorytm szyfrujący i jeden haszujący.

4.1. Negocjowanie algorytmów

Podczas konfiguracji klienta i serwera wybieramy algorytmy, które mają być brane pod uwagę podczas negocjowania połączenia. Osobno dla algorytmów szyfrujących i haszujących musimy również podać jeden z czterech trybów, w jakim połączenie ma być negocjowane.

Konfigurację przeprowadzamy przy pomocy *Oracle Net Manager*, lub ręcznie w pliku `sqlnet.ora`. Parametr dla klienta Oracle:

```
SQLNET.ENCRYPTION_CLIENT = wartość
```

```
SQLNET.CRYPTO_CHECKSUM_CLIENT = wartość
```

Parametr dla serwera Oracle:

```
SQLNET.ENCRYPTION_SERVER = wartość
```

SQLNET.CRYPTO_CHECKSUM_SERVER = wartość

Dozwolone wartości dla powyższych parametrów to: REJECTED, ACCEPTED, REQUESTED oraz REQUIRED.

Wartość **REJECTED** (odrzucone) odpowiada najniższemu poziomowi bezpieczeństwa. Należy ją wybrać, jeżeli chcemy odrzucić połączenie nawet, jeżeli jest wymagane (*REQUIRED*) po drugiej stronie. Inaczej mówiąc nie zezwalamy na bezpieczne połączenie.

Wartość **ACCEPTED** (przyjęte) należy wybrać, jeżeli chcemy zezwolić na bezpieczne połączenie, jeżeli druga strona żąda (*REQUESTED*) lub wymaga (*REQUIRED*) takiego połączenia. Inaczej mówiąc ta strona nie wymaga stosowania żadnych algorytmów.

Jeżeli druga strona ma wartość REQUESTED lub REQUIRED i są jakieś wspólne algorytmy na liście po obu stronach, to połączenie jest nawiązywane w sposób bezpieczny. Jeżeli druga strona jest skonfigurowana jako REQUIRED i nie zostaną znalezione wspólne algorytmy połączenie nie zostanie nawiązane.

Wartość **REQUESTED** (żądane) mówi, że bezpieczne połączenie jest pożądane, ale nie wymagane. Należy ją wybrać, jeżeli chcemy nawiązać bezpieczne połączenie w wypadku, gdy druga strona na to zezwala. Ta wartość mówi, że połączenie będzie nawiązane w sposób bezpieczny, jeżeli po drugiej stronie mamy *ACCEPTED*, *REQUESTED* lub *REQUIRED* i zostaną znalezione wspólne algorytmy.

Wartość **REQUIRED** (wymagane) informuje, że bezpieczne połączenie jest wymagane po tej stronie. Wybierz tą wartość, jeżeli chcesz żeby połączenie było nawiązane w sposób bezpieczny lub nie powiodło się.

Poniższe tabele informują czy podczas transmisji będą stosowane algorytmy bezpieczeństwa (wartość 1), czy nie (wartość 0) w zależności od parametrów ustalonych po obu stronach.

Znaleziono zgodne algorytmy	REJECTED	ACCEPTED	REQUESTED	REQUIRED
REJECTED	0	0	0	Połączenie nie powiedzie się (ORA-12660)
ACCEPTED	0	0	1	1
REQUESTED	0	1	1	1
REQUIRED	Połączenie nie powiedzie się (ORA-12660)	1	1	1

Nie znaleziono zgodnych algorytmów	REJECTED	ACCEPTED	REQUESTED	REQUIRED
REJECTED	0	0	0	Połączenie nie powiedzie się (ORA-12660)
ACCEPTED	0	0	0	Połączenie nie powiedzie się (ORA-12650)
REQUESTED	0	0	0	Połączenie nie powiedzie się (ORA-12650)
REQUIRED	Połączenie nie powiedzie się (ORA-12660)	Połączenie nie powiedzie się (ORA-12650)	Połączenie nie powiedzie się (ORA-12650)	Połączenie nie powiedzie się (ORA-12650)

Aby stosować szyfrowanie, należy również po obu stronach ustalić wartość parametru `SQLNET.CRYPTO_SEED`. Parametr ten powinien zawierać przypadkowy ciąg znaków ASCII wykorzystywany jest on jako pomoc dla generatora liczb losowych podczas ustalania wartości klucza sesji. W wersji *Oracle 8i* posiada on wartość domyślną, natomiast w wersji *9i* należy obowiązkowo podać wartość tego parametru.

Przykład:

```
SQLNET.CRYPTO_SEED = g43hu5yfh@435yfr$%trg2455
```

Informacje na temat bieżących sesji znajdują się w perspektywie `V$SESSION`, szczegółowe informacje dotyczące każdej sesji (na przykład o stosowanych algorytmach szyfrujących) można uzyskać wydając zapytanie na perspektywie `V$SESSION_CONNECT_INFO`.

Przykładowe zapytanie zwracające zestaw algorytmów używanych w bieżącej sesji danego użytkownika:

```
SQL> select sci.SID, ses.USERNAME, sci.NETWORK_SERVICE_BANNER
       from v$session_connect_info sci , v$session ses
       where sci.SID = ses.SID
       and USERNAME = 'SCOTT';
```

Perspektywa `v$session_connect_info` w wersji 8 nie jest prawidłowo odświeżana. Różne sesje podłączane w różnym czasie mogą otrzymać ten sam numer SID, co z kolei może prowadzić do błędnych odczytów. Problem rozwiązano w wersji 9. Informacje o użytych algorytmach można również odczytać z pliku śladu. Należy włączyć śledzenie na kliencie lub serwerze na poziomie ADMIN i przeszukać pliki śladu pod kątem wyrażenia "na_tns:".

5. AUTENTYKACJA I SZYFROWANIE Z WYKORZYSTANIEM SSL

Secure Socket Layer jest szeroko stosowanym standardem opracowanym przez *Netscape Communications Corporation*. Służy do kompleksowego zabezpieczania połączeń zapewniając autentycację, szyfrowanie, integralność. Działa w oparciu o infrastrukturę klucza publicznego (PKI).

SSL w środowisku Oracle można stosować do autentycacji:

- dowolnego klienta lub serwera do jednego lub wielu serwerów (serwer jest pewien tożsamości klienta lub innego serwera),
- dowolnego serwera do klienta (klient jest pewien że naprawdę łączy się z żądanym serwerem).

Można również skorzystać z zewnętrznych mechanizmów autentycacji w połączeniu z szyfrowaniem realizowanym przez SSL lub wykorzystać SSL jedynie jako mechanizm szyfrujący.

5.1. Omówienie architektury PKI

Szyfrowanie po SSL zakłada istnienie klucza publicznego (ogólnie dostępnego) i prywatnego (znanego tylko właścicielowi). Dane zaszyfrowane jednym z kluczy mogą być rozszyfrowane wyłącznie przy użyciu drugiego. Z założenia klucze powiązane są w taki sposób, że nie można (w rozsądnym czasie) wyliczyć wartości klucza prywatnego znając klucz publiczny i odwrotnie. Wymaga to stosowania odpowiednich długości klucza.

Przesyłanie danych z wykorzystaniem SSL wygląda następująco:

1. Aby wysłać poufną wiadomość musimy znać klucz publiczny adresata;
2. Szyfrujemy informacje kluczem publicznym;
3. Adresat używa własnego klucza prywatnego do rozszyfrowania wiadomości.

5.2. Certyfikat

Zawiera klucz publiczny właściciela oraz informacje dotyczące jego tożsamości podpisane (zaszyfrowane) przy użyciu klucza prywatnego „instytucji certyfikującej”. *Oracle Advanced Security* operuje na certyfikatach w formacie x509 v3.

Instytucja certyfikująca (*Certificate Authority*)

Zewnętrzna instytucja (*software*) sprawdzająca tożsamość i wystawiająca certyfikaty na żądanie różnych podmiotów np.: użytkowników, administratorów, serwerów, klientów.

Certyfikat jest zaszyfrowany przy użyciu klucza prywatnego CA. Aby rozszyfrować dane zawarte w certyfikacie podpisanym przez CA trzeba znać klucz publiczny CA.

CA udostępnia swój własny certyfikat zawierający dane identyfikacyjne oraz klucz publiczny niezbędny do rozszyfrowania certyfikatów wystawionych przez tą CA. Każda jednostka przechowuje zbiór certyfikatów CA, którym ufa (*trusted certificate* lub *root certificate*) zanim rozpocznie komunikację z inną jednostką, sprawdza czy jej certyfikat jest podpisany przez znaną instytucję certyfikującą.

W środowisku Oracle proces autentycacji przebiega następująco:

- Użytkownik inicjuje połączenie z serwerem używając protokołu *TCP with SSL*.
- SSL wykonuje „*handshake*”:
- Następuje negocjacja algorytmów szyfrujących.

- Serwer wysyła swój certyfikat do klienta i klient sprawdza czy został on podpisany przez odpowiednią instytucję CA.
- Jeżeli wymagana jest autentykacja klienta, klient wysyła swój certyfikat i analogicznie serwer sprawdza czy został on podpisany przez zaufaną CA.
- Zarówno serwer jak i klient generują klucze sesji i wymieniają się nimi. Wartości kluczy są szyfrowane przy pomocy kluczy zawartych w certyfikatach.
- Jeżeli „handshake” zakończył się sukcesem serwer sprawdza uprawnienia użytkownika w bazie danych. Może do tego wykorzystać dane klienta zawarte w certyfikacie x509.

Dalsza komunikacja między serwerem a klientem jest szyfrowana i deszyfrowana przy pomocy kluczy sesji z zastosowaniem wynegocjowanych symetrycznych algorytmów szyfrujących i haszujących.

5.3. Zarządzanie portfelami - Wallet Manager

Zbiór informacji (klucze, certyfikaty, certyfikaty CA) niezbędnych do nawiązania połączenia przy użyciu SSL. W środowisku Oracle każda jednostka używająca SSL musi posiadać portfel zawierający:

- Certyfikat (format x509 v3),
- Parę kluczy prywatny i publiczny,
- Listę certyfikatów zaufanych instytucji certyfikujących (CA).

Do zarządzania portfelami zarówno w warstwie klienta i serwera służy *Oracle Wallet Manager*. Jest to samodzielna aplikacja napisana w języku *Java*. Pozwala na wykonanie następujących zadań związanych z zapewnieniem komunikacji poprzez SSL:

- Generowanie pary kluczy prywatnego i publicznego i żądania certyfikatu (*certificate request*) do podpisania przez CA;
- Instalacja certyfikatu dla jednostki (*user certificate*);
- Instalacja certyfikatów zaufanych instytucji CA (*trusted certificates*);
- Otwarcie portfela – umożliwia dostęp do usług wymagających PKI;
- Stworzenie i zapisanie portfela, który może zostać otwarty przez *Oracle Wallet Manager* lub *Oracle Enterprise Login Assistant*.

6. ZEWNĘTRZE SYSTEMY AUTENTYFIKACJI

Za pomocą opcji ASO istnieje możliwość połączenia systemu autentyfikacji użytkowników w środowisku rozproszonym z autentyfikacją w bazie danych Oracle. Opcja ASO wspiera między innymi następujące systemy autentyfikacji: Kerberos, Radius, SecurID. Posiadanie jednego, wspólnego systemu uwierzytelniania użytkowników, upraszcza zarządzanie całym środowiskiem. Użytkownik otrzymuje dostęp do różnych usług, za pomocą pojedynczego logowania.

Poniższy przykład pokazuje połączenie systemu Kerberos5 z bazą danych Oracle 9iR2 na platformie Linux-a. W systemie Kerberos oprócz pojedynczego logowania się do systemu, dodatkową zaletą jest nie przesyłanie hasła przez sieć, co utrudnia potencjalne włamanie do systemu.

6.1. Kerberos5 – zewnętrzny system autentyfikacji

W celu zainstalowania i używania systemu Kerberos5 w systemie operacyjnym Linux, należy zainstalować następujące pakiety:

- krb5-libs
- krb5-server
- krb5-workstation

Następnie należy skonfigurować system Kerberos (Key Distribution Center), a później wykonać następujące czynności:

- dodanie jednostki odpowiadającej serwerowi gdzie będzie zainstalowana baza danych
`addprinc host/laptop.altkom.com.pl`
- eksport kluczy do pliku
`ktadd -k /etc/krb5.keytab host/laptop.altkom.com.pl`
- dodanie jednostki odpowiedzialnej za bazę danych – nazwa usługi dowolna
`addprinc ora920/laptop.altkom.com.pl`
- eksport jednostki do pliku kluczy
`ktadd -k /etc/krb5.keytab ora920/laptop.altkom.com.pl`
- dodanie jednostki odpowiadającej użytkownikowi
`addprinc ouser`

Zmiany w pliku `init.ora`

```
remote_os_authent = false
os_authent_prefix = ""
```

Zmiany w bazie danych:

W bazie danych należy stworzyć użytkownika odpowiadającemu nazwie użytkownika utworzonego w systemie Kerberos. Do nazwy użytkownika należy dziedzinę naszej instalacji Kerberosa (REALM). Uwaga użytkownika musi być utworzony DUŻYMI literami.

```
create user "OUSER@ALTKOM.COM.PL" identified externally;
```

po czym należy nadać mu uprawnienia:

```
grant connect to OUSER@ALTKOM.COM.PL;
```

Zmiany w konfiguracji sieci:

```
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE = "ora920"
SQLNET.KERBEROS5_CONF = /etc/krb5.conf
SQLNET.KERBEROS5_CONF_MIT = TRUE
SQLNET.KERBEROS5_KEYTAB = /etc/krb5.keytab
SQLNET.KERBEROS5_REALMS = /etc/krb.realms
SQLNET.AUTHENTICATION_SERVICES= (BEQ, TCP, KERBEROS5)
SQLNET.CRYPTO_SEED = 12h38ge76g3dhg436gd3hudg73hdy3fdyuwgsagsw
```

W celu zalogowania się do bazy należy wydać następujące polecenia:

```
okinit -f ouser
```

powoduje otrzymanie od serwera Kerberos tokenu, który może być wykorzystany do otrzymania kolejnych tokenów.

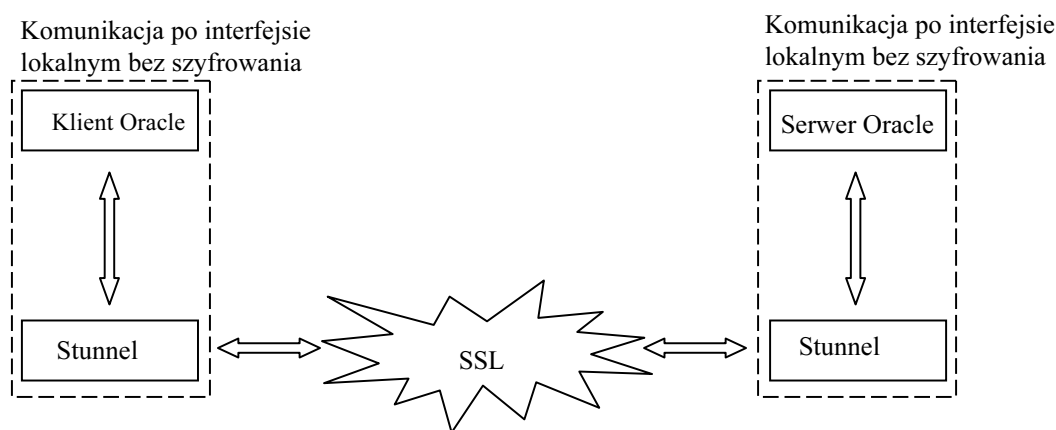
```
sqlplus /@secur
```

7. SZYFROWANIE POŁĄCZENIA ZA POMOCĄ PAKIETU STUNNEL

Za pomocą pakietu Stunnel możliwe jest zaszyfrowanie połączenia pomiędzy serwerem a klientem. Po obu stronach połączenia należy uruchomić program stunnel, oraz odpowiednio skonfigurować połączenia sieciowe.

Po stronie klienta, aplikacja łączy się za pomocą interfejsu lokalnego, do programu Stunnel, który nawiązuje połączenie z serwerem, uzgadnia protokół szyfrowania i następnie przekazuje zaszyfrowany pakiet otrzymany od aplikacji do serwera. Na serwerze dokonywana jest czynność odwrotna, pakiet jest deszyfrowany, a następnie przekazywany, albo do procesu listenera (nawiązywanie połączenia), albo do procesu serwera, odpowiedzialnego za daną sesję.

Aby wdrożyć to rozwiązanie, należy odpowiednio skonfigurować pakiet Stunnel, wygenerować odpowiednie certyfikaty a następnie zmienić konfigurację sieci Net8. Konfiguracja ta ma zapewniać łączenie się klienta, po interfejsie lokalnym z programem stunnel, a po stronie serwera, nasłuch procesu listenera na interfejsie lokalnym.



Przykładowo:

Plik TNSNAMES.ORA na stacji klienckiej

```
SECUR.KATOWICE.ALTKOM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = 127.0.0.1) (PORT = 20000))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = secur)
    )
  )
)
```

Oprócz tego należy uruchomić program stunnel po stronie klienta z następującymi parametrami:

```
Stunnel -c -d 20000 -r serwer.oracle:port_stunnel
```

Natomiast po stronie serwera, konfiguracja listenera wygląda następująco:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT = 1521))
    )
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = secur)
      (ORACLE_HOME = /vol/oracle)
      (SID_NAME = secur)
    )
  )
)
```

Oprócz tego należy uruchomić program stunnel z następującymi parametrami:

```
Stunnel -d port_stunnel -r localhost:1521
```

Taka konfiguracja zapewni szyfrowanie połączeń pomiędzy serwerem a klientem Oracle. Dzięki możliwości programu Stunnel, oprócz szyfrowania połączeń można również dokonywać weryfikacji stacji klienckich na podstawie ich certyfikatów. Wszelkie opcje konfiguracji można znaleźć na stronie www.stunnel.org.

8. SZYFROWANIE POŁĄCZENIA ZA POMOCĄ PAKIETU SSH

Za pomocą pakietu SSH można dokonać przekierowania ruchu pomiędzy bazą danych a klientem, na połączenie bezpieczne – szyfrowane. Zasada działania tego rozwiązania jest taka sama jak w przypadku szyfrowania za pomocą pakietu Stunnel. Ruch pomiędzy klientem Oracle a klientem SSH odbywa się poprzez interfejs lokalny, a następnie ruch pomiędzy stacją a serwerem odbywa się poprzez kanał szyfrowany. Ruch na serwerze pomiędzy Oraclem a serwerem SSH odbywa się za pomocą interfejsu lokalnego. Użytkownik powinien posiadać możliwość zalogowania się do systemu operacyjnego serwera.

Przykład konfiguracji:

Plik TNSNAMES.ORA na stacji klienckiej

```
SECUR.KATOWICE.ALTKOM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = 127.0.0.1) (PORT = 20000))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = secur)
    )
  )
)
```

Oprócz tego należy uruchomić program SSH po stronie klienta z następującymi parametrami:

```
ssh -L 10000:adres_serwera:port_listener user@adres_serwera
```

Natomiast po stronie serwera, konfiguracja listenera wygląda następująco:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT = 1521))
    )
  )
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = secur)
      (ORACLE_HOME = /vol/oracle)
      (SID_NAME = secur)
    )
  )
)
```

UWAGA: Dla serwerów baz danych zainstalowanych w środowisku Windows, w celu zapewnienia działania dwóch w/w metod należy ustawić parametr `USE_SHARED_SOCKET`, na wartość `TRUE`. Parametr ten wyłącza przekierowanie portów połączenia pomiędzy klientem a serwerem, które występuje w środowisku Windows. Podobna sytuacja ma miejsce w przypadku serwerów pracujących w trybie MTS, gdzie należy określić zakres portów, na których następują przekierowania.