Computer Science Vol. 4. 2002 004.056:004.75

*Jędrzej Byrski**

ZAGADNIENIE BEZPIECZEŃSTWA KOMUNIKACJI W SYSTEMIE CORBA

Standard CORBA definiuje mechanizmy współdziałania systemów rozproszonych. Główną rolę odgrywa system ORB (Object Request Broker), który umożliwia zlokalizowanie odpowiedniego serwera i przezroczystą komunikację pomiędzy serwerem a klientem. W artykule przedstawiono problemy związane z bezpieczeństwem komunikacji pomiędzy systemami ORB. W systemie CORBA obiekty są identyfikowane przez IOR (Interoperable Object Reference). Dla protokołu TCP/IP zawiera on informacje o adresie IP serwera i numerze portu oraz klucz obiektu. Filtracja przez firewall może wykorzystywać informacje: typ komunikatu, adres IP klienta, klucz obiektu, do którego klient chce uzyskać dostęp, typ operacji, właściciel klienta. W zaproponowanej implementacji firewall pracuje jako serwer w systemie CORBA i współpracuje z półmostem. Jest on zarejestrowany w systemie ORB i dostarcza funkcji kontroli dla przychodzących pakietów poprzez interfejs IDL. W artykule przedstawiono budowę modułu filtrującego. Jego główną częścią jest ACL (Access Control List) zawierająca reguły dostępu. W artykule przedstawiono także analizę wydajności zaproponowanego rozwiązania.

Słowa kluczowe: *firewall, COBRA, systemy rozproszone, bezpieczeństwo komunikacji*

SECURITY PROBLEM OF COMMUNICATION IN CORBA SYSTEM

CORBA standard defines the mechanisms of shearing services. The key rule plays ORB (Object Request Broker) which enables location of suitable server and transparent communication between client and server. This paper presents problems connected with security during communication between ORB systems. In the CORBA system the objects are identified by IOR (Interoperable Object Reference). For TCP/IP it contains IP server address, port and object key. Filtration may use such information as: type of communicate, IP address of client, object key to which client wants to access, type of operation, clients principal. In proposed implementation the firewall works as CORBA server and cooperates with half-bridge. It is registered in ORB system and provides the controlling functions for entering packages by IDL interface. In the paper also the structure of filtering module is presented. Its main part is ACL (Access Control List) with rules of access. The performance evaluation results are also presented.

Keywords: *firewall, COBRA, distributed computer systems, security*

* Katedra Automatyki, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Akademia Górniczo-Hutnicza, Kraków, e-mail: jbyrski@agh.edu.pl

1. Wstęp

Ważnym zagadnieniem dla komputerowych systemów rozproszonych jest możliwość wiązania poszczególnych składowych aplikacji. Rozwiązanie tego problemu umożliwia m.in. standard tworzenia oprogramowania dla środowiska rozproszonego OMG CORBA (*Common Request Broker Architecture*). Technologia ta umożliwia integrację systemów pracujących na różnych platformach sprzętowych, w różnych systemach operacyjnych, napisanych w różnych językach programowania i korzystających z różnych protokołów komunikacyjnych. Standard CORBA określa mechanizmy dzielenia usług między obiektami scentralizowanych lub rozproszonych systemów. Kluczową rolę w tej architekturze odgrywają mechanizmy ORB (*Object Request Broker*), umożliwiające znalezienie odpowiedniego serwera i przezroczystą komunikację między klientem a serwerem. ORB składa się z jądra (*core*) i z otaczających go interfejsów [2, 25].

Niniejszy artykuł ma na celu przedstawienie zagadnień związanych z bezpieczeństwem dostępu podczas komunikacji między systemami typu ORB i przedstawienie propozycji projektu i implementacji modułu filtrującego dla systemu CORBA. Powstał on na podstawie prac prowadzonych nad implementacją programu typu moduł filtrujący (firewall) dla systemu CORBA [24].

2. Internet jako medium dla przetwarzania rozproszonego

W związku z rozwojem Internetu i rozproszonych systemów informacji pojawił się problem bezpieczeństwa pracy. Względnie łatwy dostęp do zasobów umożliwił podgląd i modyfikację składowanych i przesyłanych danych, przy jednoczesnym braku możliwości identyfikacji nieuprawnionego użytkownika sieci.

Jakość systemu ochrony danych ocenia się na podstawie analizy czterech cech:

1) poufności (*confidentiality*),
2) nienaruszalności (*integrity*),
3) zdolności oceny (*accountability*),
4) dostępności (*availability*).

Ochrona dzieli się na ochronę techniczną (komputerową i komunikacyjno-radiacyjną) oraz nietechniczną (brak fizycznego dostępu do zasobów, procedury przyznawania priorytetów dostępu, ochrona nośników, ochrona dokumentacji).

Procedury ochronne będą dotyczyły:

- usługi weryfikacji (*authentication services*) – identyfikujące i rejestrujące użytkownika sieci,
- usługi autoryzacji (*authorization services*) – dopuszczające pracę zweryfikowanego użytkownika na ściśle określonych prawach dostępu,
- usługi poufności (*confidentiality services*) – utajniające dane i gwarantujące właścicielowi wyłączność dostępu,
- usługi integralności (*integrity services*) – gwarantujące nienaruszalność i prawdziwość informacji,
- usługi niezaprzeczalności (*non-repudiation services*) – eliminujące możliwość anonimowości użytkownika.

3. Zadanie firewalla

Zadaniem firewalla jest zapewnienie poufności i niedostępności danych sieci prywatnej połączonej z siecią rozległą. Najprostszym urządzeniem pełniącym tę rolę jest router filtrujący transmitowane pakiety (*screening router*) na poziomie protokołów IP, IPX. Bardziej rozbudowane możliwości posiada specjalizowany komputer-twierdza (*bastion-host*), prowadzący monitorowania pracy sieci lokalnej lub jego odmiana z dwoma interfejsami sieciowymi (*dual homed gateway*), przez którą istnieje jedyna komunikacja z siecią zewnętrzną. Stosowana jest również architektura mieszana (*screened host gateway*), gdy z komputerem-twierdzą współpracuje router filtrujący pakiety. Funkcję firewalla mogą również pełnić wydzielone podsieci (*screened subnet*). Firewall, realizowany na poziomie aplikacji na zasadzie proxy buforującego dane na trzecim komputerze widzianym z obu sieci, też zapewnia wymianę informacji do i z sieci prywatnej.

Opisy niewielkich przykładowych systemów firewall znaleźć można w literaturze przedmiotu [9, 10].

4. Konstrukcja modułu filtrującego (firewall) dla systemu CORBA

Wraz z wprowadzeniem możliwości współpracy różnych systemów ORB poprzez protokół GIOP i IIOP [14] system CORBA może być wykorzystany bezpośrednio w sieci Internet [12, 13, 14, 17]. Wprowadza to jednak nowe zagrożenia ze względu na jej ogólnoświatową dostępność, przy jednoczesnym braku gwarancji dostępności tylko przez uprawnionych użytkowników. W celu zapobieżenia takiej sytuacji muszą być podjęte odpowiednie środki bezpieczeństwa. Obecnie stosowany model bezpieczeństwa w organizacjach podłączonych do Internetu zakłada stosowanie systemu firewalla. Model ten polega na stworzeniu mocnego punktu kontrolnego, odpornego na różne formy włamania, przez który przechodzi cały ruch do i z danej sieci chronionej.

Komunikat IIOP wysyłany przez klienta, gdzieś w sieci Internet, aby dotrzeć do serwera, musi przejść dwa moduły filtrujące (firewall). Pierwszy, gdy opuszcza sieć wewnętrzną klienta, a drugi, gdy wchodzi do sieci serwera CORBY.

Zazwyczaj konstrukcja firewalla polega na wydzieleniu silnego komputera, zabezpieczonego przed włamaniem ze strony hackerów, zwanego komputerem-twierdzą (*bastion-host*). Komputer ten przy wykorzystaniu specjalistycznego oprogramowania kontroluje cały wchodzący i wychodzący ruch z sieci prywatnej. W obrębie takiego firewalla tworzone są obiekty pozorne (proxy), gdy następuje konieczność połączenia z zewnątrz sieci prywatnej z serwisem dostępnym wewnątrz tej sieci, połączenia realizowane są przez proxy, a nie bezpośrednio z serwisem wewnętrznym. Taki model nazywany jest firewallem połączeniowym. Gdy dodatkowo wyposażymy firewall w możliwość filtracji przechodzących pakietów, tak by tylko niektóre dozwolone pakiety mogły wejść lub wyjść z sieci prywatnej, mówimy o firewallu filtrującym.

Propozycja firewalla dla protokołu IIOP jest generalnie zbliżona do opisanej powyżej koncepcji, została ona tylko dopasowana do filtracji pakietów na podstawie informacji charakterystycznych dla protokołu IIOP.

Filtracja pakietów przechodzących przez moduł filtrujący opiera się na informacjach zawartych w komunikatach GIOP, które składają się z:

- nagłówka GIOP,
- nagłówka specyficznego dla danego komunikatu,
- ciała komunikatu (dla niektórych komunikatów).

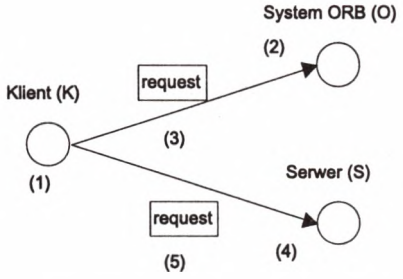
W systemie CORBA przy współdziałaniu różnych systemów ORB obiekt identyfikowany jest przez *Interoperable Object Reference* – IOR. Dla protokołu TCP/IP IOR zawiera przynajmniej:

- adres IP serwera;
- port;
- object key (łańcuch znaków identyfikujący obiekt).

Korzystając z informacji zawartych w strukturach opisanych powyżej, filtracja może się odbywać na podstawie następujących informacji:

- typu komunikatu GIOP;
- adresu IP klienta;
- object key obiektu, do którego klient chce mieć dostęp;
- wywoływanej operacji;
- principal klienta – w systemie Orbix identyfikuje użytkownika, który uruchomił klienta.

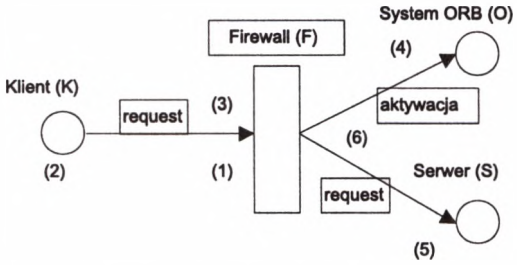
Kiedy klient łączy się z serwerem poprzez firewall, firewall kontroluje przesyłany komunikat i w przypadku gdy komunikat ma być przesłany dalej, aktywuje odpowiedni obiekt w chronionej sieci. Poniżej przedstawiono schematyczne modele połączeń pomiędzy klientem a serwerem, w przypadku łączności przechodzącej przez firewall i łączności bezpośredniej (rys. 1, 2)



Rys. 1. Normalne połączenie IIOP

OBJAŚNIENIA

1) Klient (K) pobiera IOR (*Interoperable Object Reference*) serwera, który posiada informacje konieczne do połączenia ze środowiskiem ORB i aktywowanie potrzebnego obiektu.
2) Klient otwiera połączenie TCP/IP do agenta systemu ORB (O).
3) Klient wysyła komunikat typu request do agenta systemu ORB (O), otrzymując jako odpowiedź informację o faktycznej lokalizacji poszukiwanego serwera (S).
4) Klient (K) otwiera połączenie TCP/IP do serwera (S).
5) Klient wysyła komunikat request do serwera (S).



Rys. 2. Połączenie IIOP przechodzące przez firewall

OBJAŚNIENIA

1) Firewall (F) musi być najpierw skonfigurowany np. z wykorzystaniem pliku konfiguracyjnego lub programu konfiguracyjnego.
2) Klient (K) pobiera IOR (*Interoperable Object Reference*) serwera, który posiada informacje konieczne do połączenia ze środowiskiem ORB i aktywowanie potrzebnego obiektu.
3) Klient (K) otwiera połączenie TCP/IP z firewallem (F) i wysyła komunikat request.
4) Firewall (F) analizuje otrzymany komunikat zgodnie ze swoja konfiguracją i jeżeli komunikat może być przesłany dalej, łączy się z agentem systemu ORB (O), pytając o lokalizację poszukiwanego serwera (S). Jako odpowiedź otrzymuje tę lokalizację.
5) Firewall (F) otwiera połączenie z serwerem (S).
6) Firewall (F) pośredniczy w przekazywaniu komunikatów od serwera do klienta i z powrotem aż do zamknięcia połączenia.

5. Budowa modułu filtrującego – firewall

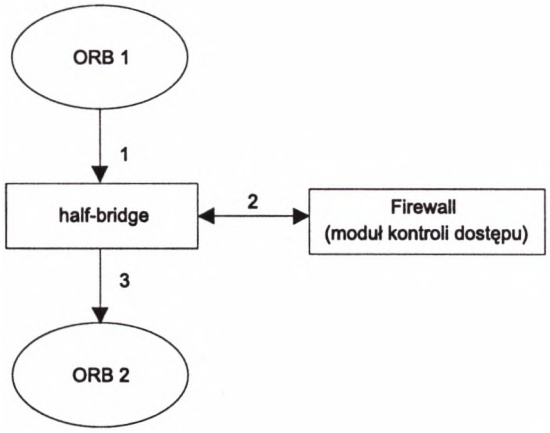
5.1. Zasada działania

W zaproponowanym rozwiązaniu firewall filtrujący jest napisany jako serwer CORBY. Jest on rejestrowany w danym systemie ORB i udostępnia poprzez interfejs IDL funkcję umożliwiająca sprawdzanie nadchodzących pakietów. Funkcja filtracji pakietów na poziomie półmostu (half-bridge) sprowadzi się więc do wywołania poprzez ORB funkcji sprawdzającej firewalla filtrującego. Funkcja ta jako parametr otrzymuje informacje o pakiecie, który półmost (half-bridge) ma przesłać dalej. Moduł filtrujący na podstawie informacji od firewalla i skonfigurowanej listy dostępu oceni, czy dany pakiet może być przesłany dalej. Informację tę zwróci do półmostu (half-bridge), który na tej podstawie albo prześle dany pakiet dalej, albo go odrzuci. Przy komunikacji pomiędzy modułem filtrującym a półmostem (half-bridge), przy przesyłaniu informacji od modułu filtrującego do półmostu (half-bridge), musi być zapewniona autentykacja, tzn. półmost (half-bridge) musi mieć pewność, że uzyskana odpowiedź pochodzi od „prawdziwego" modułu filtrującego. Autentykacja przy komunikacji w drugim kierunku nie jest konieczna, ponieważ moduł filtrujący bez obniżenia poziomu bezpieczeństwa może udzielać informacji, czy dany pakiet może być przesłany każdemu zainteresowanemu taką informacją. Konfiguracja modułu filtrującego zapisana jest w pliku konfiguracyjnym. Przy starcie modułu filtrującego konfiguracja jest wczytywana. Zmiana konfiguracji będzie mogła być dokonywana poprzez edycję/utworzenie pliku konfiguracyjnego i powtórną inicjalizację modułu filtrującego.

Możliwe jest stworzenie interfejsu GUI służącego do konfigurowania firewalla. Do konfiguracji przy użyciu interfejsu GUI służy applet napisany w Javie i OrbixWebie. Applet ten jest klientem CORBY i łączył się z modułem filtrującym, który jest dla niego serwerem CORBY. Wykonując operacje w interfejsie graficznym, można skonfigurować moduł filtrujący. Ostatecznie wywoływana będzie funkcja zapisu tak ustawionej konfiguracji w pliku przez moduł filtrujący w taki sposób, że przy kolejnym uruchomieniu konfiguracja ta zostaje wczytana z tego pliku. Wykorzystując interfejs graficzny, można także dokonać modyfikacji aktualnie ustawionej konfiguracji modułu filtrującego, a także przeglądnąć plik, w którym dokonuje się logowania sprawdzanych pakietów i wygenerować na podstawie tego pliku pewne informacje ogólne, typu: liczbę odrzuconych pakietów w stosunku do wszystkich pakietów, rozkład typu komunikatów, czy też sprawdzić, do których obiektów bywa najczęstszy dostęp. Przedstawiony powyżej sposób konfigurowania modułu filtrującego wymaga zastosowania mechanizmów autentykacji przy komunikacji pomiędzy klientem napisanym w Javie, służącym do ustawienia konfiguracji, a modułem filtrującym, tak żeby tylko uprawniony klient mógł zmienić konfigurację modułu filtrującego. Najważniejsza jest autentykacja w kierunku od appletu konfigurującego do modułu filtrującego. Moduł filtrujący musi być pewny, że klient konfigurujący ma prawo do zmiany konfiguracji. Autentykacja w przeciwnym kierunku nie jest tak istotna, ponieważ bez obniżania poziomu bezpieczeństwa moduł filtrujący może przekazywać informacje o swojej konfiguracji każdemu zainteresowanemu.

5.2. Projekt modułu filtrującego

W zaproponowanym rozwiązaniu przyjęto, że moduł filtrujący nie będzie wbudowany wewnątrz półmostu, ale będzie stanowił osobny program, z którym półmost będzie się komunikował, wykorzystując mechanizmy systemu CORBY. Taka konstrukcja rozwiązania z jednej strony zwiększa jego przejrzystość, elastyczność i uniwersalność, z drugiej strony obniża wydajność działania. Schematycznie jest to przedstawione na rysunku 3.



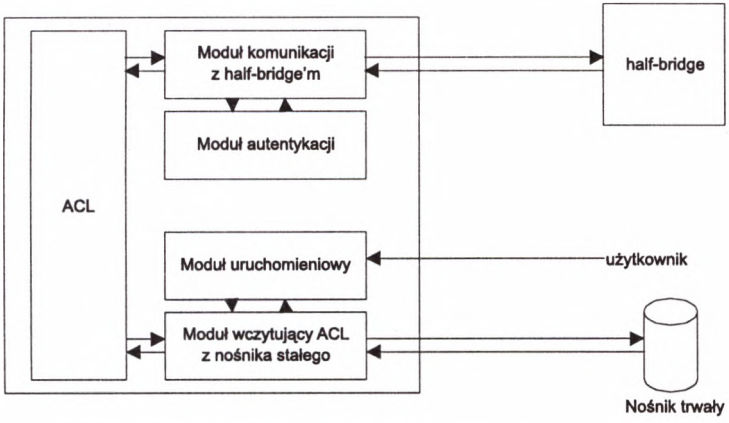
Rys. 3. Przyjęte rozwiązanie połączenia półmostu (half-bridge) i modułu filtrującego

Gdy w środowisku ORB 1 wywoływany jest obiekt ze środowiska ORB 2, wywołanie takie trafia do półmostu (half-bridge) (1). Gdy półmost (half-bridge) nie jest wyposażony w moduł kontroli dostępu, wywołanie to w każdym przypadku jest przekazywane dalej (3).

W przypadku gdy półmost (half-bridge) jest wyposażony w moduł kontroli dostępu, najpierw przesyła on informacje o pakiecie, który otrzymał ze środowiska ORB 1, do modułu kontroli (2).

Moduł kontroli na podstawie tych informacji i informacji zawartych w swojej konfiguracji ocenia, czy dany pakiet może być przesłany dalej, czy też powinien być odrzucony i informację tę zwraca do półmostu (half-bridge), który zgodnie z tym albo dany pakiet przesyła dalej do środowiska ORB 2 (3) bądź odrzuca. Komunikacja pomiędzy półmostem (half-bridge) a modułem filtrującym bazuje na standardowych mechanizmach systemu CORBA. Taka konstrukcja modułu filtrującego pozwala stosunkowo łatwo wyposażyć istniejące półmosty (half-bridge) w funkcje kontroli dostępu.

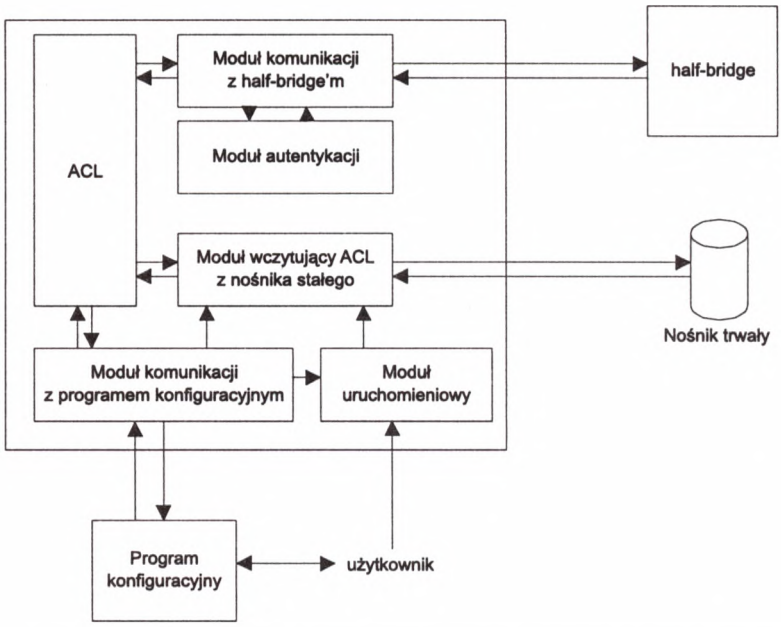
Wewnętrznie można w module filtrującym wyróżnić kilka modułów funkcjonalnych. Schematycznie przedstawiono to na rysunku 4.



Rys. 4. Wewnętrzna konstrukcja modułu filtrującego

Na rysunku 5 przedstawiono schematyczną budowę wewnętrzną modułu filtrujacego rozbudowanego o możliwość korzystania z interfejsu użytkownika typu GUI.

Najważniejszą częścią modułu filtrującego jest lista dostępu ACL (*Access Control List*). Zawiera ona reguły dostępu zgodnie z którymi działa moduł filtrujący. Po otrzymaniu od półmostu (half-bridge) informacji o pakiecie, moduł filtrujący stara się dopasować tę informację do którejś z pozycji listy dostępu. Na podstawie tego dopasowania moduł filtrujący może zdecydować, czy dany pakiet może być przesłany dalej, czy też powinien być odrzucony. Z listą dostępu współpracuje moduł wczytujący ACL z nośnika trwałego (zazwyczaj dysk twardy), interpretuje on zapisaną w pliku listę reguł dostępu i zgodnie z nią inicjuje wewnętrzną listę ACL. Moduł ten jest też odpowiedzialny za zapisanie na nośniku trwałym ustawionej aktualnie listy dostępu. Moduł ten jest pewnego typu prostym skanerem i parserem interpretującym język charakterystyczny dla konfiguracji modułu filtrującego. Z modułem wczytującym współpracuje moduł uruchomieniowy, interpretuje on parametry wywołania podane przez użytkownika, uruchamia i rejestruje cały moduł filtrujący jako serwer systemu CORBA, a następnie uruchamia moduł wczytujący ACL.



Rys. 5. Wewnętrzna konstrukcja modułu filtrującego rozbudowanego o moduł konfiguracji

Po uruchomieniu i inicjalizacji modułu filtrującego oczekuje on na zapytania od półmostu (half-bridge). Osobny moduł jest odpowiedzialny za komunikację pomiędzy firewallem a półmostem (half-bridge). Odbiera on przesłane przez półmost (half-bridge) informacje o otrzymanych pakietach i na podstawie listy dostępu ACL ocenia, czy pakiety te mogą być dalej przesłane przez półmost (half-bridge) czy nie. Informację tę zwraca do półmostu (half-bridge). Moduł ten współpracuje z modułem zapewniającym autentykację przy komunikacji pomiędzy firewallem a półmostem (half-bridge). Autentykacja jest konieczna przy przekazywaniu informacji w kierunku od modułu filtrującego do półmostu (half-bridge). Dzieje się tak, ponieważ półmost (half-bridge) musi mieć pewność, że otrzymana odpowiedź jest prawdziwa, a nie została podana przez podstawiony firewall w celu uzyskania nieuprawnionego dostępu. Autentykacja przy komunikacji w kierunku przeciwnym nie jest konieczna, ponieważ moduł filtrujący może udzielać informacji dowolnemu półmostowi (half-bridge) bez obniżania poziomu bezpieczeństwa. W celu autentykacji wykorzystany został algorytm szyfrowania IDEA. Do pary 128-bitowych kluczy dostęp posiada zarówno półmost (half-bridge), jak i firewall. Najpierw półmost (half-bridge) generuje 800-bajtowy losowy ciąg i koduje go przy użyciu jednego klucza. Tak zakodowany ciąg jest następnie przekazywany jako jeden z parametrów poprzez ORB do firewallu. Firewall dekoduje ten ciąg, uzyskując oryginał. Następnie firewall koduje tak uzyskany ciąg przy użyciu drugiego klucza i wraz z odpowiedzią zwraca do półmostu (half-bridge). Półmost (half-bridge) dekoduje ten ciąg i porównuje z oryginałem, który wygenerował. Gdy ciągi są identyczne oznacza to, że odpowiedź pochodzi od prawdziwego modułu filtrującego.

Zastosowana metoda autentykacji zabezpiecza system przed próbą „podsłuchania" przesyłanego ciągu, ponieważ sam przesyłany ciąg nie daje nic, istotne są dwa klucze IDEA, które nie są przesyłane przez sieć, a są wykorzystywane tylko lokalnie. Dystrybucja kluczy w obecnej wersji jest rozwiązana w sposób najprostszy, tzn. administrator systemu „ręcznie" umieszcza wygenerowaną przez moduł filtrujący parę kluczy w lokacji półmostu (half-bridge) i firewalla. W obecnej wersji programu firewall może on współpracować jednocześnie tylko z jednym półmostem (half-bridge), więc nie występuje konieczność częstego zmieniania kluczy, a zaproponowana metoda dystrybucji kluczy jest wystarczająca. W ramach rozbudowy, poprzez wprowadzenie możliwości jednoczesnej współpracy jednego modułu filtrującego z wieloma półmostami (half-bridge), można wprowadzić inne mechanizmy dystrybucji kluczy, np. uwzględniające automatyczną wymianę kluczy na nowsze, co pewien określony czas. Przy autentykacji wybrano symetryczny algorytm szyfrowania IDEA. Ponieważ nie występuje konieczność przesyłania przez sieć klucza, więc nie ma podstaw do stosowania bardziej czasochłonnych algorytmów niesymetrycznych (np. RSA), które mają bardziej skomplikowaną budowę i implementację. Wnoszą przez to większe obciążenia czasowe podczas działania. Ponadto nie istnieją dowody formalne na ich nierozwiązalność matematyczną. Szyfry niesymetryczne są bezpieczne, dopóki nie powstanie algorytm i maszyna matematyczna umożliwiająca ich złamanie. W przypadku szyfrów symetrycznych, takich jak IDEA czy też DES, zostało udowodnione, że ich złamanie jest nierozwiązywalne matematycznie ze względu na wykorzystanie funkcji jednokierunkowych i architektury symetrycznej. Przy wyborze szyfru symetrycznego brano więc głównie pod uwagę szyfry IDEA i DES. Ponieważ szyfr IDEA jest algorytmem nowszym i o wyższym poziomie bezpieczeństwa niż algorytm DES, zdecydowano się ostatecznie na ten algorytm. Dodatkowo implementacja algorytmu IDEA nie wymaga operacji na dużych liczbach całkowitych, ponieważ 128-bitowy klucz dzieli się na osiem 16-bitowych podkluczy i szyfrowanie odbywa się z wykorzystaniem tych podkluczy. Cechy takiej nie posiada algorytm DES, w którym wymagane są operacje na 64-bitowych liczbach całkowitych. Sam algorytm IDEA powstał na bazie doświadczeń uzyskanych podczas 10-letnich badań algorytmu DES, zebranych przez specjalistów od kryptografii. IDEA to algorytm symetryczny, działający w oparciu o 64-bitowe bloki danych, wykorzystujący 128-bitowe klucze. Projektanci IDEA usunęli wiele słabych punktów zauważonych w algorytmie DES.

5.3. Funkcjonalność modułu filtrującego

Zaproponowana budowa modułu filtrującego, sposób analizy pliku konfiguracyjnego, metoda komunikacji z półmostem (half-bridge) i zaproponowana budowa programu konfiguracyjnego zapewniają dużą elastyczność przy ewentualnych rozbudowach i modyfikacjach modułu filtrującego. Stosunkowo łatwo jest wprowadzić nowe słowa kluczowe do pliku konfiguracyjnego, zrozumiałe dla analizatora leksykalnego modułu filtrującego i tym samym wyposażyć moduł kontroli dostępu w filtrację przesyłanych pakietów na podstawie nowych nie uwzględnionych obecnie parametrów, które mogą pojawić się w miarę rozwoju systemu CORBA lub rozbudowy modułu filtrującego na bazie obecnego standardu. Również sposób implementacji modułu filtrującego w formie osobnego programu pozwala na stosunkowo łatwe wbudowanie go do różnych półmostów bez konieczności istotnych zmian w ich budowie wewnętrznej.

Program konfiguracyjny będący osobnym programem niezależnym od samego modułu filtrującego, a komunikującym się z nim poprzez system CORBA, umożliwia w razie potrzeby łatwe zastąpienie go innym, bardziej odpowiadającym potrzebom użytkownika praktycznie bez zagłębiania się w konstrukcję wewnętrzną samego modułu filtrującego. Tak duża modularność zastosowana w projekcie wpływa korzystnie na jasną strukturę budowy modułu filtrującego i łatwość jego rozbudowy i modyfikacji.

Stroną ujemną takiego rozwiązania są z pewnością większe obciążenia czasowe niż w przypadku zabudowania wszystkiego w postaci jednego programu. Zastosowanie jednak szybkich mechanizmów komunikacji i zoptymalizowanej pod względem szybkości budowy wewnętrznej (algorytmów działania) ogranicza tę niekorzystną stronę.

5.4. Testowanie modułu filtrującego

Moduł filtrujący był dokładnie testowany zarówno pod względem prawidłowości działania, tzn. przepuszczalności tylko pakietów, które są dozwolone przez plik konfiguracyjny, jak i pod względem obciążeń czasowych wnoszonych podczas połączeń przechodzących przez półmost (half-bridge). Testy były przeprowadzane na różnych plikach konfiguracyjnych przygotowywanych ręcznie, jak również losowo – automatycznie. Pliki generowane automatycznie były wykorzystywane głównie podczas testowania obciążeń czasowych. W tym celu generowane były duże pliki konfiguracyjne (aż do 10 000 reguł dostępu), a następnie testowano działanie modułu filtrującego. Podczas wszystkich testów moduł filtrujący działał poprawnie. W dalszej części artykułu zamieszczono wyniki testów szybkości działania dla plików konfiguracyjnych o wielkości 10, 100, 1000 i 10 000 reguł dostępu. Testy dotyczyły czasu uruchomienia modułu filtrującego, czyli czasu potrzebnego do interpretacji reguł zawartych w pliku konfiguracyjnym i czasu potrzebnego do obsługi pojedynczego zapytania ze strony półmostu (half-bridge). Podczas testów czasu uruchomienia mierzono czas startu modułu filtrującego dla list dostępu różnej długości. Dla każdej takiej listy pomiar powtórzono 10 razy, a następnie wyciągnięto z niego średnią. Testy dotyczące czasu obsługi jednego zapytania były przeprowadzone dla dwóch przypadków.

1) Dopasowywania do pierwszej pasującej reguły (instrukcja modułu filtrującego *match first*), – w tym przypadku moduł filtrujący podczas dopasowywania przegląda listę dostępu tylko tak długo, aż znajdzie pasującą regułę.
2) Następnie przeprowadzono testy dla obciążeń czasowych przy konfiguracji modułu filtrującego tak, by dopasowywał pakiet przesłany przez półmost (half-bridge) do ostatniej pasującej reguły (instrukcja modułu filtrującego *match last*). W takim ustawieniu moduł filtrujący musi, dla każdego dopasowywania danego pakietu do reguły, przeglądać cały plik konfiguracyjny.

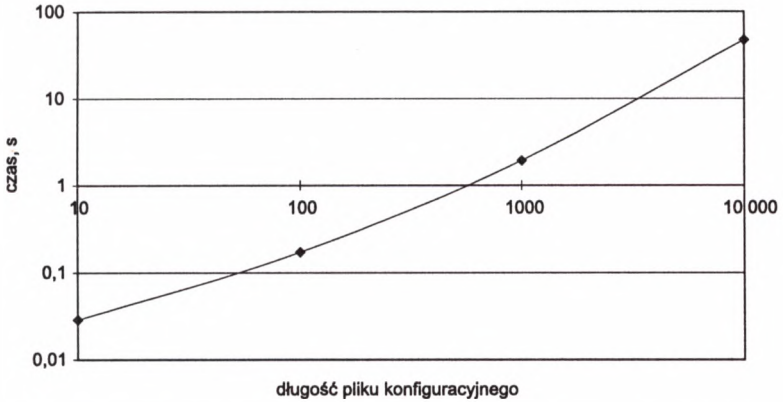
Pomiary wykonano dla grupy pakietów, z których część miała być przepuszczana, część zatrzymana przez moduł dostępu, a część nie pasowała do żadnej reguły listy dostępu. Takie pomiary powtórzono dla każdej analizowanej listy dostępu 10 razy, wyciągając na końcu wartość średnią. Testy wykonano na komputerze Sun Sparc Station. Zarówno półmost (half-bridge), jak i moduł filtrujący działały na jednym komputerze. Komputer ten był cały czas wykorzystywany jako normalny serwer sieciowy. W tabelach (tab. 1, 2, 3) i na wykresach (rys. 6, 7) zamieszczono uzyskane wyniki.

Tabela 1

Czasy startu modułu filtrującego, w s

	1	2	3	4	5	6	7	8	9	10	średnio
10	0,030	0,031	0,029	0,029	0,028	0,029	0,029	0,029	0,030	0,029	0,029
100	0,173	0,172	0,170	0,176	0,171	0,174	0,174	0,172	0,168	0,170	0,172
1000	1,939	1,926	1,935	1,921	1,940	1,925	1,930	1,957	1,919	1,923	1,931
10 000	48,07	47,05	47,27	47,52	47,68	47,41	47,69	47,19	48,11	47,17	47,52

Na rysunku 6 zamieszczono wykres logarytmiczny wartości średnich czasu startu modułu filtrującego w funkcji wielkości pliku konfiguracyjnego.



Rys. 6. Zależność czasu uruchomienia modułu filtrującego od długości pliku konfiguracyjnego

Z wykresu na rysunku 6 widać, że podczas startu modułu filtrującego decydującą rolę odgrywa długość pliku konfiguracyjnego. Wraz ze wzrostem długości pliku konfiguracyjnego wzrasta czas uruchomienia modułu filtrującego (na skali logarytmicznej – proporcjonalnie).

Czasy obsługi pakietu, w sekundach, przesłanego przez półmost (half-bridge) do analizy w przypadku konfiguracji modułu filtrującego komendą `match first` przedstawiono w tabeli 2.

Tabela 2

Czas autoryzacji pakietu przez moduł filtrujący przy konfiguracji `match first`, w s

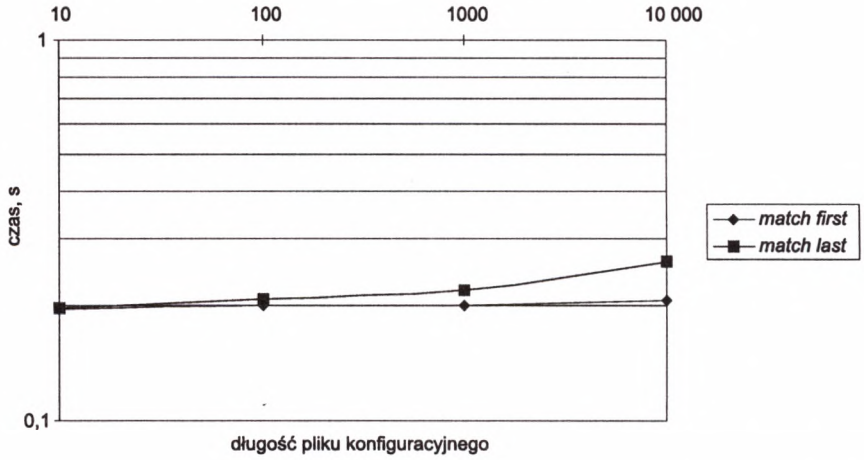
	1	2	3	4	5	6	7	8	9	10	średnio
10	0,198	0,196	0,2	0,19	0,189	0,193	0,193	0,202	0,191	0,204	0,196
100	0,209	0,2	0,199	0,2	0,199	0,199	0,202	0,196	0,198	0,198	0,2
1000	0,202	0,203	0,196	0,222	0,194	0,193	0,202	0,198	0,196	0,201	0,201
10 000	0,204	0,219	0,235	0,202	0,203	0,199	0,2	0,196	0,203	0,201	0,206

Czasy obsługi pakietu, w sekundach, przesłanego przez półmost (half-bridge) do analizy w przypadku konfiguracji modułu filtrującego komendą `match last` przedstawiono w tabeli 3.

Tabela 3

Czas autoryzacji pakietu przez moduł filtrujący przy konfiguracji `match last`, w s

	1	2	3	4	5	6	7	8	9	10	średnio
10	0,2	0,19	0,204	0,193	0,193	0,189	0,19	0,21	0,195	0,202	0,197
100	0,21	0,212	0,211	0,217	0,208	0,21	0,209	0,205	0,21	0,2	0,209
1000	0,222	0,219	0,2	0,225	0,226	0,223	0,222	0,224	0,223	0,22	0,22
10 000	0,281	0,256	0,257	0,252	0,253	0,256	0,257	0,261	0,27	0,259	0,26



Rys. 7. Czas autoryzacji pakietu przez moduł filtrujący

Na rysunku 7 zamieszczono wykres logarytmiczny wartości średnich czasu obsługi wywołań przez moduł filtrujący w funkcji wielkości pliku konfiguracyjnego.

Z powyższego wykresu wynika, że czas obsługi wywołań przez moduł filtrujący jest słabo zależny od długości pliku konfiguracyjnego. W przypadku konfiguracji modułu filtrującego komendą `match first` taka zależność właściwie nie występuje, a większość czasu obsługi wywołania zabiera komunikacja pomiędzy półmostem (half-bridge) a firewallem. Przy skonfigurowaniu modułu filtrującego komendą `match last` (w takim przypadku przy obsłudze wywołania moduł filtrujący musi za każdym razem przeglądnąć całą listę dostępu) zależność taka jest już widoczna, jednakże czas obsługi nadal rośnie znacznie wolniej niż długość pliku konfiguracyjnego. Z powyższych obserwacji można wyciągnąć wniosek, że algorytmy dopasowywania zapytania do listy konfiguracyjnej są zaimplementowane w sposób optymalny i nie wnoszą dużych obciążeń czasowych. Obciążenia czaso-

we związane z komunikacją wynikają z założeń przyjętych przy projektowaniu modułu filtrującego, w których postawiono sobie za cel stworzenie rozwiązania otwartego kosztem trochę większych obciążeń czasowych.

6. Wnioski końcowe

Zrealizowany projekt jest pełnowartościowym oprogramowaniem typu firewall, możliwym do zastosowań w dowolnej sieci UNIX-owej działającej pod systemem CORBA. Zgodnie z założeniami projektu oprogramowanie spełnia warunek otwartości, uniwersalności i niezależności od półmostu (half-bridge). Przy wielokrotnej komunikacji rzutuje to oczywiście na czasy obsługi (testowe czasy obsługi w granicach 0,2 – 0,26 s, przy liczbie reguł dostępu – 10 000). W konkretnych przypadkach istnieje możliwość rezygnacji z otwartości rozwiązania poprzez wbudowanie oprogramowania modułu filtrującego w półmost (half-bridge) i uzyskanie znacznie lepszych czasów obsługi wywołań. Obecna implementacja modułu pozwala na łatwe jego wbudowanie do istniejących półmostów (half-bridge) bez konieczności znajomości ich struktury wewnętrznej. Również fakt implementacji w języku C++ w postaci klas, umożliwia łatwą późniejszą rozbudowę lub modyfikację projektu zgodnie z aktualnymi potrzebami.

Projekt zrealizowany w aktualnej postaci stanowi pewną propozycję realizacji firewalla filtrującego dla standardu CORBA. Dalsza rozbudowa samego modułu filtrującego może przebiegać w dwóch kierunkach:

1) można modyfikować moduł filtrujący, dopasowując go do rozwijającej się cały czas specyfikacji CORBA;
2) można równocześnie zwiększać samą funkcjonalność modułu filtrującego.

Jako możliwe kierunki rozbudowy modułu filtrującego można zaproponować zwiększenie integralności połączenia pomiędzy półmostem (half-bridge) a firewallem, co z jednej strony podniesie sprawność działania, z drugiej jednak ograniczy otwartość zaproponowanego rozwiązania. Sam moduł filtrujący może zostać dopasowany do jednoczesnej współpracy z więcej niż jednym półmostem (half-bridge), przez wyposażenie go w możliwość analizy i działania w oparciu o więcej niż jedną listę dostępu.

Literatura

[1] Zieliński K.: *Środowiska programowania rozproszonego w sieciach komputerowych.* Kraków 1994

[2] *The Common Object Request Broker: Architecture and Specification.* OMG Document Number 91.12.1. Revision 1.1.

[3] Stawowski M.: *OMG CORBA.* Software. Narzędzia, Programy, Sieci, 8, 1996, 86

[4] Kowalczyk N.: *Specyfikacja CORBA – perspektywy rozwoju.* UNIX FORUM, Systemy Otwarte, Sieci, 1, 1997, 14

[5] Betz M.: *Obiekty sieciowe i CORBA.* Software. Narzędzia, Programy, Sieci, 1, 1996, 14

[6] Berg C.: *CORBA w Javie.* Software. Narzędzia, Programy, Sieci, 3, 1997, 40

[7] Hoffman P.: *Internet. Poradnik.* Warszawa, Wydawnictwo PLJ 1995

[8] Vdell J.: *Your Business Needs the Web.* Byte 8, 1996, 68

[9] Kostick C.: *Firewall w systemie Linux.* Software. Narzędzia, Programy, Sieci, 6, 1996, 82

[10] Stawowski M.: *Budujemy Firewall.* Software. Narzędzia, Programy, Sieci, 1, 1997, 76

[11] Denning D.: *Kryptografia i ochrona danych.* Warszawa 1992
[12] *Inter-ORB Gateway Infrastructure Design and Implementation. Part II, Design.* Kraków, Instytut Informatyki AGH 1996
[13] *Inter-ORB Gateway Infrastructure Design and Implementation. Part III. Implementation.* Kraków, Instytut Informatyki AGH 1996
[14] CORBA 2.0 Interoperability Universal Networked Objects (UNO). OMG TC Document 95.3.xx
[15] IDL C++ Language Mapping Specification OMG Document 94-9-14
[16] The Orbix® Architecture, Iona Technologies 1996
[17] IIOP on the Internet White Paper, Iona Technologies 1997
[18] Orbix ® Programming Guide, Iona Technologies
[19] Orbix ® Reference Guide, Iona Technologies
[20] Młynarski K.: *Selektywna kontrola dostępu.* Software. Narzędzia, Programy, Sieci, 11, 1995, 76
[21] Włodarz J.: *Filtrujący firewall dla Linuxa.* LinuxPlus. 2, 1997, 16
[22] *Komputer sieciowy a bezpieczeństwo.* Software. Narzędzia, Programy, Sieci, 7, 1997, 60
[23] Główczyński R.: *Wszechstronna ocena bezpieczeństwa sieci.* Software. Narzędzia, Programy, Sieci, 5, 1997, 76
[24] Byrski J.: *Moduł filtrujący w systemie CORBA.* Kraków, Katedra Informatyki AGH 1997 (praca dyplomowa)
[25] Anisimowicz J. *Technologia CORBA.* Software 2.0, 12, 1999, 56 i 02, 2000, 63
[26] Byrski J.: *Organizacja komunikacji w systemach obiektowych opartych na OMG CORBA.* Linux & Unix, 5, 2000, 8
[27] Software 2.0, nr 09, 2000. Bezpieczeństwo
[28] Software 2.0, nr 10, 2000. CORBA
[29] Software 2.0, nr 10, 2001. CORBA