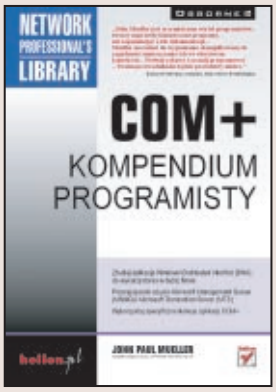
COM+. Kompendium programisty



Autor: John Paul Mueller
Tłumaczenie: Adam Balcerzak, Marcin Jędrysiak, Tomasz Wasilewski
ISBN: 83-7197-641-0
Tytuł oryginału: COM+ Developer's Guide
Format: B5, stron: 452
Przykłady na ftp: 14 kB

Dzięki tej książce odkryjemy świat modelu COM+ oraz zrozumiemy cel i zastosowanie różnych zawartych w nim technologii. Po lekturze zrozumiesz, że wiele z tej aury złożoności, która otacza COM+, bierze się jedynie z faktu konsolidacji technologii dostępnych obecnie w oddzielnych pakietach i wprowadzenia dodatkowych usprawnień, które ułatwiają tworzenie aplikacji COM w porównaniu ze stanem obecnym. Szczegółem, który odróżnia tę książkę od innych, jest zbliżenie się do rzeczywistości przy prezentowaniu modelu COM+. Oznacza to pracę z rzeczywistymi przykładami programów, które powinny pomóc Ci szybciej rozpocząć pracę z COM+. Jednym z ważniejszych rodzajów aplikacji używanych w zastosowaniach biznesowych jest baza danych – to właśnie aplikację tego typu stworzymy najpierw, a następnie będziemy ją usprawniać na wiele różnych sposobów. Przykładowa baza danych korzysta z wielu tabel. Prześledzimy dokładnie wszystkie etapy projektu wymagane do stworzenia zarówno samej bazy danych, jak i aplikacji zarządzającej nią. Część tego procesu będzie wymagała napisania i zastosowania komponentu działającego po stronie serwera, który nie tylko uprości etap tworzenia zapytań, ale również znacznie zredukuje ruch w sieci.

COM+ nie musi być puszką Pandory, pełną trudnych do zrozumienia technologii i narzędzi, która udaremni Twoją pracę. Ta książka pomoże Ci w poznaniu nowej i ekscytującej części Windows 2000. Nauczymy się wszystkiego, co COM+ może uczynić w kierunku przeskalowania Twoich aplikacji biurowych i sieciowych do zastosowania ich na poziomie sieci WAN oraz Internetu.

Książka ta ma trzy główne cele: nauczyć teorii COM+, pokazać sposoby wykorzystania modelu w środowiskach programistycznych (ogólnego przeznaczenia i firmowych) oraz zaprezentować przykładowe programy. Rozdziały 1. – 6. opisują teorię modelu COM+ oraz szczegóły użytkowe. Natomiast rozdziały 7. – 10. zawierają w pełni funkcjonalne przykłady programów.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

3.

Wyjątkowe cechy COM+

COM+ jest jednym z najbardziej oczekiwanych dodatków do systemu Windows 2000 poza Active Directory. Użytkownicy liczą, że usunie on wiele problemów związanych z technologią komponentów. Jednakże trzeba zdać sobie sprawę, że COM+ oferuje swoje funkcje jako dodatek, a nie zamiast obecnej technologii komponentów firmy Microsoft. Pomimo tego, co twierdzi Microsoft, COM+ jest kontynuacją technologii COM.

Oczywiście, ciągłość funkcji COM+ ma pewne zalety dla programisty. Po pierwsze, nakłady finansowe zainwestowane w istniejącą technologię nie są zagrożone. Zasadniczo wciąż potrzeba tych samych narzędzi, by stworzyć aplikację komponentową, a aplikacja zaprojektowana dla COM działa prawie tak samo, jak dawniej. Krótko mówiąc, COM+ jest po prostu udoskonaloną wersją COM.

W przypadku COM+ „konsolidacja" może być lepszym określeniem niż „nowość". COM+ bierze to, co najlepsze z trzech całkowicie różnych technologii. Pracując z COM+, tak naprawdę będziesz widział efekt pracy z COM, Microsoft Transaction Server (MTS), Microsoft Management Queue (MSMQ) i innymi technologiami, takimi jak DCOM. Zgadza się, DCOM jest wciąż podstawą przesyłania danych przez sieć.

Więc, po co w ogóle dodawać „+" do nazwy COM? Są pewne nowe możliwości w COM+, których istnienia powinieneś być świadomy, by móc je efektywnie wykorzystywać. Przyjrzymy się temu, co Microsoft dodał do COM w celu zrobienia z niego lepszego narzędzia do integracji. Częścią wysiłku konsolidacji jest umożliwienie pracy z różnymi elementami COM+ w sposób, który jest niewidzialny dla programisty i użytkownika. Nie warto konsolidować czegoś, jeżeli wszystkie części komponentu są wciąż całkowicie widoczne i wymagają indywidualnego dostępu.

Pierwsza część rozdziału omówi nowe funkcje automatyzacji zawarte w COM+. Najnowsza oferta COM polega na tym, że robi on pewne rzeczy automatycznie (w przeszłości musiałeś robić je ręcznie). Problem dotyczący przetrzymywania przez komponent zasobów systemowych

przestał istnieć. Komponent przetrzymuje zasoby systemu tylko tak długo, aby udało mu się zakończyć przydzielone zadanie, potem natychmiast je zwalnia. Lepsze zarządzanie zasobami oznacza, że serwer będzie pracował bardziej wydajnie i w ten sposób może spowodować opóźnienie potrzeby jego modernizacji. Dodatkowo, fakt, że COM i MTS są obecnie zintegrowane, oznacza, że istnieje większa szansa, że transakcje będą działać. Na koniec, komponenty COM+ używają tzw. kontekstu. Pomaga on określić sposób interakcji komponentu z klientem i reakcji na jego żądania.

Zasobami zarządzają dwa elementy: *rozdzielacz zasobów* oraz *menedżer wyrównywania zasobów*. Oprócz wysiłku pojedynczych komponentów, Windows 2000 ma proaktywne podejście do zarządzania zasobami. Rozdzielacz zasobów zapewnia, że każdy komponent otrzyma tylko tyle zasobów, ile potrzebuje, i tylko na czas, kiedy ich potrzebuje. Następnie, zamiast całkowicie zwalniać zasoby, rozdzielacz zasobów zwraca je do puli, która używana jest do obsługi potrzeb innych komponentów. Strategia wykorzystywania puli zasobów znacznie redukuje czas, który jest potrzebny na przydzielenie (zwolnienie) zasobów. Oznacza to, że aplikacja zużywa mniej cykli zegara procesora głównego na wywiązywanie się ze swoich podstawowych zadań.

W kolejnej części tego rozdziału przyjrzymy się *menedżerowi wyrównywania zasobów* (Compensating Resource Manager — CRM). Zwykle, byłbyś zmuszony utworzyć rozdzielacz zasobów dla swojej aplikacji. Zależnie od stopnia złożoności aplikacji, rozdzielacz zasobów nie miałby zbyt wiele do roboty. COM+ zwalnia cię z obowiązku stworzenia rozdzielacza zasobów, oferując gotowe rozwiązanie w postaci CRM. Ta cecha COM+ umożliwia aplikacjom pracę z DTC bez konieczności tworzenia rozdzielacza zasobów. Wystarczy jedynie dodać specjalny interfejs do komponentu. CRM uzyskuje dostęp do tego interfejsu i umożliwia komponentowi decydowanie o wyniku transakcji na podstawie wyników poszczególnych transakcji w komponencie. Krótko mówiąc, otrzymujesz wszystkie możliwości pełnej implementacji rozdzielacza zasobów bez najmniejszego wysiłku. CRM również odpowiada za obsługę nieudanych transakcji, opierając się na wpisach pochodzących ze swojego własnego dziennika zdarzeń. Oznacza to, że zyskujesz mechanizm automatycznej obsługi błędów bez dodatkowej pracy.

Czwarta część rozdziału omawia ujścia zdarzeń COM+. Są to odbiorniki zdarzeń. W przypadku wystąpienia zdarzenia na serwerze powiązane ujście otrzymuje wiadomość o zdarzeniu, a następnie reaguje na nie. Funkcja nie jest nowa, lecz w przypadku COM+ jest inaczej obsługiwana. Pamiętaj, że COM+ obejmuje wiele technik zdalnego uruchamiania opartych na starszych technologiach, takich jak DCOM. W rezultacie, zdarzenie, które miało miejsce, niekoniecznie wydarzyło się na lokalnym serwerze. Mógł je wygenerować klient, lub serwer obsługujący obecnie część większej aplikacji.

Koncepcja katalogu komponentów, opisana w kolejnej części rozdziału, powinna zainteresować programistów mających do czynienia z bibliotekami typów. Zamiast tworzenia opisu komponentu w oddzielnym pliku (twoja aplikacja może go nie zrozumieć), COM+ udostępnia koncepcję katalogu komponentów. Jest to w zasadzie baza danych specjalnego przeznaczenia, która służy do przechowywania opisów wszystkich komponentów zarejestrowanych na serwerze. Aplikacja może pobrać te opisy, by uzyskać więcej informacji na temat komponentu i sposobu współpracy z nim. Nie musisz się już dłużej martwić o to, gdzie znajduje się plik *TLB* (zawierający bibliotekę typów), ponieważ twoja aplikacja może pobrać niezbędne informacje z centralnego magazynu.

W kolejnej części rozdziału przyjrzymy się tematowi puli obiektów i obiektom zasobów. Pula obiektów jest następcą mechanizmu puli obecnie używanego przez Microsoft Data Access

Components (MDAC). Podczas pracy z MDAC aplikacja może stworzyć połączenie z bazą danych, następnie po udanym jego nawiązaniu zwrócić je do puli połączeń. Połączenie nie jest zrywane, więc krócej trwa wydostanie go z puli i ponowne użycie niż nawiązanie nowego połączenia. Pula obiektów działa na tej samej zasadzie, lecz odnosi się do wszystkich rodzajów obiektów, włączając w to połączenia z bazą danych. Używanie puli obiektów sprawia, że aplikacja staje się bardziej wydajna. Obiekty zasobów są specjalnymi komponentami, które obsługują mechanizm puli obiektów. Istnieje kilka wymogów dotyczących ich tworzenia, lecz najważniejsze jest to, że obsługują one interfejs *IObjectControl*.

Zabezpieczenia oparte na rolach to temat poruszony w kolejnej części rozdziału. Windows 2000 korzysta z tego mechanizmu, ponieważ jest to najbardziej wydajny sposób obsługi dostępu użytkownika w środowiskach sieciowych dużych rozmiarów. Pomyśl o rolach jako o bardziej elastycznym rozszerzeniu pojęcia grup. Tworzysz listę różnych ról w firmie (kierownik działu, informatyk itd.), a następnie przydzielasz jedną lub więcej ról każdemu użytkownikowi. Rola określa, kim jest użytkownik według kryterium wykonywanych przez niego zadań. Innymi słowy, mechanizm zabezpieczeń opartych na rolach nie zakłada, że użytkownik powinien mieć dostęp do obiektu po prostu dlatego, że pracuje w danym dziale firmy. Użytkownik musi wykonywać zadanie, które wymaga dostępu do tego obiektu, by wymagania bezpieczeństwa zostały spełnione.

Ostatnia część rozdziału omówi funkcję, która może (ale nie musi) być dostępna, gdy będziesz czytał po raz pierwszy ten rozdział — *wyrównywanie obciążenia komponentu* (CLB). Początkowo istniały plany włączenia tego mechanizmu do Windows 2000, lecz firma Microsoft ostatecznie zdecydowała się udostępnić go jako oddzielny produkt i tylko z serwerami wyższej klasy. Jednakże, mechanizm ten jest tak ważny, że pomimo wszystko zostanie omówiony. Zasadniczo, CLB umożliwia systemowi Windows 2000 automatycznie wyrównywanie obciążenia komponentu, rozkładając je na kilka serwerów. Na przykład, możesz mieć komponent bazy danych załadowany na czterech serwerach ze względu na stopień wykorzystania. Jak administrator określa serwer, do którego został przydzielony użytkownik? Dawniej administrator mógł jedynie statycznie przypisać użytkownika do serwera i mieć nadzieję, że zrobił dobrze. Niestety, oznaczało to, że niektóre serwery były bardzo obciążone, podczas gdy inne nie były wykorzystywane. CLB zmienia ten stan rzeczy. Obecnie żądanie użytkownika dotyczące dostępu do komponentu jest przypisywane serwerowi, który potrafi najlepiej wywiązać się z zadania jego obsługi. Użytkownik uzyskuje najlepszy możliwy czas dostępu, a firma nie musi przeprowadzać zbyt częstych modernizacji, ponieważ wykorzystywany jest pełen potencjał mocy każdego serwera.

COM+ i automatyzacja

Programiści COM spędzają wiele czasu na zarządzaniu częściami swoich aplikacji, których użytkownik nigdy nie widział lub uznał je za automatyczne. Problemy te wynikały częściowo z faktu, że Microsoft oferował wiele technologii, które mogły pracować z COM, lecz były one sprzedawane jako oddzielne produkty. Oczywiście, najważniejsze w tym przypadku są MSMQ i MTS, które obecnie są częścią systemu operacyjnego Windows 2000. Wcześniej wspominaliśmy, że COM+ nie jest nowy, ile raczej skonsolidowany. Konsolidacja jest bardziej przydatna do automatycznego wykonywania zadań w tle.

W tym podrozdziale przyjrzymy się elementom automatyzacji. Pierwszym z nich jest *aktywacja w czasie trwania* (JIT, *Just In Time*). Technologia ta pozwala serwerowi na bardziej wydajne używanie zasobów poprzez zachowywanie aktywności obiektów jedynie tak długo, jak są potrzebne. Kiedy klient przez długi czas przechowuje adres do nieużywanego komponentu, serwer dezaktywuje ten komponent do chwili, gdy będzie on ponownie potrzebny. Następnym razem, gdy klient go zażąda, serwer dokona automatycznej reaktywacji tego komponentu.

Drugim elementem automatyzacji są transakcje. Zapewniają one, że każde żądanie klienta jest obsługiwane przynajmniej raz i tylko ten jeden raz. Jest to niezwykle ważna cecha baz danych, gdyż wprowadzenie danych więcej niż jeden raz może spowodować ich uszkodzenie. Ponieważ będziemy pracować z transakcjami, ta część będzie jedynie przeglądem zagadnienia. Wiadomości teoretyczne na ten temat można znaleźć w rozdziale 4., zaś przykład transakcyjny w rozdziale 7.

Przegląd domen COM+ i jego kontekstu zawarty w ostatniej części rozdziału pomoże zrozumieć programistom sposób, w jaki wykorzystywany jest poszczególny egzemplarz komponentu. Przeglądanie komponentów w kontekście pozwala wykonać więcej przetwarzania mniejszym kosztem. Pomaga również w dziale automatyzacji poprzez umożliwienie COM+ dokonania ulepszeń w dziedzinie zabezpieczeń. Więcej na temat specyfiki zabezpieczeń COM+ w części zatytułowanej „Zabezpieczenia oparte na rolach".

Aktywacja w czasie trwania (JIT)

Zawsze ktoś żąda dostępu do zasobów serwera. Z tego powodu, standardowe techniki programowania COM często obejmują pojęcie przejęcia zasobu i następnie przetrzymania go do czasu, aż komponent nie będzie dłużej potrzebował tego zasobu. Oznacza to, że zasób może być zablokowany na długo i przez ten czas być bezczynny. Nowa metodologia COM+ polega na uzyskaniu dostępu do zasobu, następnie możliwie szybkim przekazaniu go do puli, by inne aplikacje miały dostęp do potrzebnych im zasobów. Oczywiście, wciąż gdzieś tam jest dużo starszych aplikacji COM, które nie używają tej nowej strategii, więc jest bardzo prawdopodobne, że wiele zasobów serwera wciąż jest marnowanych. Dlatego też potrzebny jest *mechanizm aktywacji w czasie trwania* (JIT).

Używanie aktywacji JIT oznacza, że nawet jeżeli klient przetrzymuje odwołanie do komponentu, Windows 2000 wciąż może używać fizycznych zasobów wymaganych przez komponent do chwili, gdy będą znowu potrzebne aplikacji. Windows 2000 monitoruje wszystkie komponenty, które są oznaczone jako zgodne z JIT. Kiedy minie określony czas bez wywołania jakiejkolwiek metody, komponent jest dezaktywowany, a wykorzystywane przez niego zasoby wracają do puli. Tak długo, jak mówimy o aplikacji, komponent istnieje, a odwołanie do niego wciąż pozostaje prawidłowe. Następnym razem, gdy aplikacja wywoła metodę dezaktywowanego obiektu, Windows 2000 dokona jego ponownej aktywacji i przydzielenia wymaganych zasobów. Cały ten proces rozgrywa się w tle bez żadnego udziału ze strony programisty. Użytkownik jest całkowicie nieświadomy tego, co się stało.

Zatem, jeżeli nowa technika polega na zwalnianiu zasobów, które nie są już dłużej potrzebne, jak aktywacja JIT pomaga programistom COM+? COM+ jest zaprojektowany do ułatwiania procesu tworzenia aplikacji rozproszonych. W rezultacie, komponent nie musi być aktywowany na lokalnym serwerze — właściwie może być uruchomiony na zdalnym serwerze. Oznacza to,

że wywołanie każdego klienta przejdzie przez wiele serwerów i zablokuje te zasoby na tych serwerach pośredniczących do chwili, gdy zdalny serwer umożliwi dostęp do potrzebnego komponentu. Oczywiście, nie potrzeba wiele takich żądań, by obniżyć ogólną wydajność sieci komputerowej (z powodu zwiększonej liczby żądań sieciowych) i wydajności kilku serwerów. Aktywacja JIT służy również pomocą w sytuacji, gdy chcesz zmniejszyć liczbę wywołań klientów do minimum.

Z pewnością zadajesz sobie pytanie, czy masz jakąkolwiek kontrolę nad aktywacją JIT, poza jej uruchamianiem i wyłączaniem. Microsoft udostępnia interfejs *IObjectControl*, by umożliwić ci ręczną interakcję z komponentami. Ten interfejs jest używany zarówno w przypadku puli obiektów (opisanej dalej w rozdziale), jak i aktywacji JIT. Dwie z metod `Activate()` i `Deactivate()` umożliwiają ręczne wykonanie zadań aktywacji JIT na komponencie. Wykorzystanie tych funkcji pozwoli ci poprawić wydajność całego procesu przydzielania zasobów.

Trzecią metodą interfejsu *IObjectControl* jest `CanBePooled()`. Ta metoda jest używana przez pulę obiektów, by poinformować system Windows 2000, czy dany obiekt może korzystać z puli. Obiekt musi zwrócić wartość TRUE przy wywołaniu metody `CanBePooled()`, by zostać wysłanym do puli obiektów, zamiast być zniszczony podczas dezaktywacji przez Windows 2000. Korzystanie z puli obiektów znacznie przyspiesza proces aktywacji i dezaktywacji.

Przetwarzanie transakcji

Przetwarzanie transakcji zwykle rodzi wizje zarządzania bazami danych oraz różnego rodzaju danych finansowych. Jest to faktyczna przyczyna zaistnienia mechanizmu przetwarzania transakcji. Pracownicy bankowi muszą mieć pewność, że informacje wprowadzane przez nich do bazy danych są w niej zapisywane raz i tylko jeden raz. Pomyśl, jakie spustoszenie wywołałby kilkukrotny zapis tych samych transakcji!

Jednakże, nie tylko instytucje finansowe potrzebują korzystania z transakcji. Szpital lub podobna instytucja również musi mieć pewność, że dane są zapisywane prawidłowo. Krótko mówiąc, MTS był początkowo projektowany do spełnienia wymagań całej grupy ludzi, którzy potrzebowali bezwzględnej integralności danych. Są po prostu takie miejsca, gdzie utrata integralności danych grozi katastrofą.

Oczywiście, dane nie są jedynym rodzajem obiektu, który wymaga ściślej kontroli, zwłaszcza w środowisku aplikacji rozproszonych Windows 2000. Co się stanie, jeżeli potrzebny komponent zostanie uszkodzony w trakcie przesłania z serwera lub jeżeli dwie kopie tego komponentu są przesyłane w różnym czasie? W pierwszym przypadku, aplikacja klienta może po prostu zaprzestać pracy. W drugim przypadku, możesz doświadczyć naprawdę dziwnych i nieprzewidywalnych efektów, ponieważ niezainicjowany komponent jest w stanie zamazać ten, który aplikacja zdążyła zainicjować.

Jak widać, transakcje są bardzo ważną częścią COM+. Zarówno dane, jak i kod wymagają nadzoru, by mieć pewność, że obiekt wysyłany do klienta jest tym, który tam właściwie dociera. Więcej na temat sposobu działania transakcji w dalszej części książki.

Uwaga

By odświeżyć nieco Windows 2000, Microsoft nadał nowe nazwy kilku istniejącym już produktom. Przeznaczenie ich jest takie same, jedynie ich nazwy uległy zmianie. Większość z zestawu funkcji udostępnianych przez MTS obecna jest w DTC, który właściwie zaczynał jako część składowa pakietu SQL Server. Obecnie MSMQ jest określany mianem Queued Components (Składniki w kolejce) lub Message Queuing (Kolejkowanie wiadomości), zależnie od tego, z którą wersją MSMQ pracujesz. W wielu przypadkach, podczas przeglądania dokumentacji Microsoftu określenia te będą występowały razem. Dla porządku, odwołując się do MTS i MSMQ, będziemy używać ich starych nazw. Z wyjątkiem sytuacji, gdy będzie mowa o oknach dialogowych lub innych cechach Windows 2000, które korzystają z nowego nazewnictwa lub jeżeli nowe określenie jest z jakiegoś powodu bardziej odpowiednie.
Na przykład, jeżeli będę mówił o tym, jak skonfigurować MTS do użycia z komponentem, najprawdopodobniej użyję określenia DTC w trakcie opisywania sposobu wykonania tego zadania, ponieważ jest to nowe zastosowanie istniejącego produktu.

Kontekst COM+

Każdy komponent COM+ jest tworzony ze specjalnym zestawem właściwości, znanym pod nazwą *kontekst*. Identyfikuje on dany egzemplarz komponentu i jest związany z tylko jednym apartamentem COM. Wiele obiektów może być związanych z pojedynczym kontekstem — zwykle, gdy tych obiektów żąda aplikacja. Dodatkowo, jeden apartament może zawierać wiele kontekstów. Zamysł skrywający się za kontekstem polega na daniu komponentowi punktu odwołania, jakiegoś pomysłu na to, gdzie jest jego miejsce w aplikacji. Poprzez sprawdzenie kontekstu, w którym jest uruchomiony, komponent może, zależnie od potrzeb, wykonywać dodatkowe przetwarzanie.

Wskazówka

Może spotkałeś się z sytuacją, gdy konteksty używane przez COM+ były określane mianem interfejsów obiektowych kontekstu MTS („MTS context wrapper"). W systemie Windows 2000 pojęcie to zostało rozszerzone, by objąć całą działalność COM. W rezultacie, wcześniejsze określenie nie jest używane, lecz nie zmienia to faktu, że jest ono wciąż obecne w innej formie.

COM+ również radzi sobie z kontekstem komponentu. Używa właściwości zawartych w kontekście, by udostępniać usługi podczas fazy wykonania. Dodatkowo, kontekst ma wpływ na środowisko uruchamiania komponentu. Podczas pracy z COM+, będziesz bezpośrednio pracował z właściwościami kontekstu, by wskazać systemowi operacyjnemu pewne zmiany w środowisku

i rodzajach aplikacji. Na przykład, podczas pracy z komponentem transakcyjnym będziesz używał kontekstu, by decydować o wyniku transakcji.

Każdy obiekt COM+ ma dostęp do interfejsu *IObjectContext*. Przyjrzymy mu się dokładniej w rozdziale 4. podczas omawiania MTS. Istnieje sześć metod bezpośrednio związanych z MTS, które pozwalają wpływać na wynik transakcji: `CreateInstance()`, `IsInTransaction()`, `SetAbort()`, `SetComplete()`, `EnableCommit()`, `DisableCommit()`. Metoda `IsCallerInRole()` jest używana wraz z zabezpieczeniami opartymi na rolach, w celu sprawdzenia obecnej roli programów żądających — ich uprawnień do używania różnych metod wewnątrz twojego komponentu. Metoda `IsSecurityEnabled()` pozwala określić, czy mechanizm zabezpieczeń MTS jest włączony. Zabezpieczenia MTS nie będą włączone jedynie wtedy, gdy obiekt jest uruchamiany w procesie klienta. Należy sprawdzić tę konkretną wartość, aby upewnić się, czy zabezpieczenia są włączone. Sprawdzenie tej wartości pozwala również określić miejsce uruchomienia komponentu (klient czy serwer). Dodatkowo, oprócz interfejsu *IObjectContext*, istnieje kilka dodatkowych interfejsów związanych z kontekstem komponentu. Oto ich lista:

- *IObjectContextInfo* — pozwala uzyskać kilka konkretnych informacji na temat bieżącego kontekstu, włączając w to identyfikatory transakcji, aktywności i kontekstu. Z pewnością użyjesz również metody tego interfejsu — `IsInTransaction()`, by dowiedzieć się, czy obecnie twój komponent jest częścią transakcji. Metoda `GetTransaction()` pozwoli uzyskać wskaźnik `ITransaction` do bieżącej transakcji (jeżeli jest jakaś).
- *IContextState* — udostępnia bieżące informacje o obecnym stanie kontekstu i pozwala określać czy twój komponent zgadza się na wykonanie transakcji. W praktyce właściwie nie zachodzi potrzeba skorzystania z tego interfejsu. Jednakże, metoda `GetMyTransactionVote()` pozwoli określić stan swojego głosu w głosowaniu w sprawie zakończenia transakcji. Metoda `GetDeactivateOnReturn()` pozwoli na określenie bieżącego stanu bitu „realizacji" twojego komponentu — ten bit określa moment zakończenia procesu przetwarzania przez komponent.
- *IObjectContextActivity* — jedyna metoda tego interfejsu uzyskuje identyfikator aktywności dla bieżącego kontekstu komponentu.
- *ISecurityCallContext* — omówimy dokładnie ten interfejs w części dotyczącej zabezpieczeń opartych na rolach. Ten interfejs jest odpowiedzialny przede wszystkim za umożliwianie komponentowi określania bieżącego stanu zabezpieczeń. Można użyć metod tego interfejsu do określenia, jakie role zostały przydzielone bieżącemu użytkownikowi, co pozwoliłoby ustalić, czy powinien mieć dostęp do konkretnych metod w twoim komponencie.

Zwykle, by określić stan swojego komponentu — jak został aktywowany i przez kogo — będziesz używał właściwości kontekstu, by w ten sposób wybrać najlepszy sposób działania podczas obsługi wywołań jego metod. Jednakże istnieją dwa szczególne zdarzenia, które wymagają dokładniejszego omówienia: aktywacja i przechwycenie. Aktywacja pojawia się, gdy komponent jest po raz pierwszy tworzony lub reaktywowany z puli obiektów. Podczas aktywacji tworzony jest nowy kontekst dla komponentu lub używany jest już istniejący — zwykle podczas wywołania obsługi zamykanego klienta przez kolejny obiekt. Przechwycenie pojawia się podczas wywołania krzyżującego kontekst. Kontekst programu żądającego jest tak dostosowywany, by pasował do tego, który posiada wywoływany komponent. Dwa obiekty mają różne wymagania odnośnie do środowiska działania, które są zawarte w ich kontekstach. To sedno kwestii używania kontekstów

— pozwalają obiektom działać w środowisku najlepiej dostosowanym do ich potrzeb. Używanie przechwytywania pozwala dwóm obiektom współdziałać w sposób nie naruszający kontekstu, w jakim pracują.

Rozdzielacze zasobów

Rozdzielacze zasobów robią dokładnie to, co sugeruje ich nazwa — rozdzielają zasoby komponentowe, których wymagają aplikacje. Każdy komponent COM+ potrafi korzystać z rozdzielaczy zasobów, by móc zarządzać różnymi obiektami, których wymaga komponent. Windows 2000 dostarcza domyślny zestaw rozdzielaczy zasobów, lecz istnieje również możliwość stworzenia własnego. Na przykład, wiele menedżerów baz danych zawiera Menedżera Zasobów oraz jeden lub więcej niestandardowych rozdzielaczy zasobów.

Wskazówka

W momencie wywołania rozdzielacza zasobów komponent staje się transakcyjny. Komponent nie musi robić nic więcej w celu uruchomienia MTS lub pracy z nim. W systemie Windows 2000 wszystkie transakcje komponentu są deklaracyjne, co oznacza, że uczestniczące komponenty nie wymagają dodatkowego kodu. Wszystkie transakcje odpowiadają wymaganiom specyfikacji transakcji OLE wdrażanej przez MS DTC(Microsoft Distributed Transaction Coordinator). Jeszcze do tego wrócimy.

Rozdzielacz zasobów pracuje z nietrwałymi danymi, które są przechowywane w pamięci operacyjnej i są tracone w momencie wyłączenia komputera lub jego ponownego uruchamiania. Żaden z komponentów obsługiwanych przez rozdzielacz zasobów nie jest trwały. Krótko mówiąc, rozdzielacz zasobów pozwala systemowi Windows 2000 zarządzać dużą pulą tymczasowych obiektów, która jest wykorzystywana do wielu różnych celów przez pewną liczbę aplikacji.

Istnieją różne rodzaje obiektów, z którymi może pracować rozdzielacz zasobów. Nie pracuje on tylko ze standardowymi komponentami. Obsługuje wszystkie rodzaje obiektów. Nawet takie, które na pierwszy rzut oka nie potrzebują jakiegokolwiek zarządzania. Zwykle zobaczysz rozdzielacz zasobów podczas pracy z takimi obiektami, jak połączenia (z bazami danych, sieciowe, z gniazdkami), bloki i struktury pamięci, kolejki, wątki oraz z innymi formami nietrwałych obiektów.

Zatem, jak pracuje rozdzielacz zasobów? Klient wysyła żądanie odnośnie do zasobu do menedżera zasobów (np. SQL Server). Wraz ze wzrostem liczby żądań, SQL Server tworzy jeden lub więcej rozdzielaczy zasobów. One zaś po kolei tworzą pulę zasobów przeznaczoną do tymczasowego przechowywania obiektów, które nie są już dłużej potrzebne aplikacji. W momencie gdy aplikacja kończy używanie obiektu w rodzaju połączenia z bazą danych, jest on umieszczany w puli zasobów, zamiast zostać zniszczony. Używanie puli redukuje czas potrzebny zarówno na pozbycie się zbędnego obiektu, jak i na stworzenie nowego, gdy tylko aplikacje tego zażądają.

Menedżer wyrównywania zasobów (CRM)

Menedżer wyrównywania zasobów (Compensating Resource Manager — CRM) jest częścią COM+, która obsługuje mechanizm zarządzania wyrównywaniem zasobów w połączeniu z Distributed Transaction Coordinator (DTC — dla tych, którzy nie lubią nowej terminologii). Pamiętaj, że Windows 2000 pozwala zarządzać obiektami innymi niż połączenia z bazą danych oraz innymi obiektami powiązanymi również z bazami danych początkowo zarządzanymi przez MTS. Teraz możesz zarządzać wszystkimi rodzajami obiektów różnego typu, zarówno zawierającymi kod, jak i dane.

Zasadniczo, menedżer zasobów obsługuje trwałe obiekty w komponencie transakcyjnym, podczas gdy rozdzielacz zasobów zajmuje się tymi nietrwałymi (istniejącymi wyłącznie w pamięci). SQL Server jest przykładem menedżera zasobów, a sterownik ODBC (zapewniający dostęp do niego) jest przykładem rozdzielacza zasobów.

Ogólnie, obiekty zaangażowane w transakcję nie są świadome, czy transakcja się udała, czy też nie. Autor komponentu zakłada, że albo transakcja się powiodła i system Windows 2000 nie będzie zmuszony robić nic więcej, albo zakończyła się niepowodzeniem. W przypadku niepowodzenia, przywracany jest stan sprzed transakcji i rozpoczynana jest nowa transakcja. W żadnym przypadku, nie istnieje żaden powód, by komponent musiał znać wynik transakcji.

Jednakże, istnieją sytuacje, gdy komponent będzie musiał wykonać pewną pracę w sytuacji niepowodzenia transakcji. Na przykład, pracując z aplikacją bazodanową, komponent może zażądać przywrócenia stanu sprzed błędnej transakcji. Ponieważ komponent jest zwykle dezaktywowany na granicach transakcji w celu zachowania ochrony transakcji, potrzebuje jakiegoś sposobu, by wykonać tę potransakcyjną czynność. I to jest zadanie dla wyrównawczej części CRM. Menedżer wyrównywania zasobów jest specjalnym rodzajem menedżera zasobów, który pozwala komponentowi na wykonanie pewnej pracy naprawczej w sytuacji niepowodzenia transakcji.

CRM jest obiektem nietransakcyjnym. Innymi słowy, nie może brać udziału w transakcji. Transakcja została zakończona i nie ma nic, na co CRM mógłby mieć wpływ — transakcja zakończyła się niepowodzeniem i czas, by wykonać naprawę.

Stworzysz CRM jako część kodu startowego swojego komponentu i zarejestrujesz za pomocą udostępnianego przez system programu *CRM Clerk*, który zapamiętuje informacje o CRM i zapisuje przejawy wszelkiej aktywności ze strony komponentu. Ten dziennik stanowi podstawę działań naprawczych, które podejmuje CRM — pozwala on CRM dowiedzieć się, gdzie transakcja doznała niepowodzenia, i wykonać każdą czynność wymaganą do przywrócenie stanu sprzed transakcji.

Podstawowy CRM jest złożony z dwóch części: *CRM Worker* i *CRM Compensator*. Ten pierwszy jest częścią, która jest aktywna podczas przygotowania do etapu dwufazowego zatwierdzania transakcji. Jest to część twojego CRM, która współpracuje programem *CRM Clerk* w celu utworzenia trwałych wpisów w dzienniku, które posłużą do przywrócenia stanu sprzed błędnej

transakcji. *CRM Compensator* jest uruchamiany podczas etapu zatwierdzania transakcji. To jest jedyna część twojego komponentu, która właściwie może zobaczyć wynik transakcji. Jeżeli transakcja się powiedzie, *CRM Compensator* zwykle zwalnia wszystkie zasoby, które przechowuje wraz z tymi przetrzymywanymi przez komponent. W przeciwnym razie *CRM Compensator* korzysta z zawartości dziennika, w celu zapewnienia, że zostanie przywrócony stan rzeczy sprzed nieudanej transakcji, bez wywierania wpływu na zadania wykonywane przez inne transakcje.

Wszystkie menedżery wyrównywania zasobów (CRMs) korzystają z usług interfejsu *ICrmCompensator*. Interfejs ten udostępnia metody umożliwiające CRM tworzenie wpisów w dzienniku, zatwierdzanie transakcji lub jej przerwanie. Istnieją również metody, które określają moment rozpoczęcia i zakończenia każdego etapu transakcji — dwie dla fazy przygotowań i kolejne dwie dla fazy zatwierdzania.

Ujścia zdarzeń COM+

W tej kwestii naprawdę nie ma nic nowego poza nazwą i sposobem, w jaki ta technologia może być zastosowana do twoich komponentów. W przypadku każdego komponentu COM, ujście zdarzeń odbiera powiadomienia o zdarzeniu. Po odebraniu komponent zwykle przeprowadza jakąś akcję w odpowiedzi na zdarzenie. Ujścia zdarzeń COM+ początkowo odnosiły się do zdarzeń generowanych przez MTS. Nowa nazwa odzwierciedla po prostu zmianę obecnych obowiązków MTS: jego zdolność do wykonywania szerszego zakresu zadań i że jest bardziej zorientowany obiektowo.

Katalog COM+

Jeżeli chcesz coś kupić w sprzedaży wysyłkowej, przeglądasz katalog, zapisujesz numer katalogowy produktu, dzwonisz do firmy i zamawiasz potrzebną rzecz. To samo dzieje się w przypadku COM+ na poziomie aplikacji. Katalog COM+ udostępnia pełną listę wszystkich komponentów zarejestrowanych na serwerze oraz ich dane konfiguracyjne. W większości przypadków zawierają one atrybuty aplikacji COM+, atrybuty z poziomów klasy i komputera. Katalog COM+ jest fizycznie przechowywany w REGDB — która jest rejestrową bazą danych (jak sugeruje nazwa). Zwykle będziesz miał bezpośredni dostęp do katalogu COM+ za pośrednictwem przystawki konsoli MMC — *Usługi składowe*.

Oczywiście, możesz również mieć dostęp do katalogu COM+ poprzez swoją aplikację. W tej sytuacji celem katalogu jest udostępnienie warstwy uniezależnienia, która ukrywa pewne kaprysy związane z uzyskiwaniem dostępu do informacji konfiguracyjnych komponentu. Dostęp do katalogu COM+ uzyskiwany jest poprzez interfejs *ICOMAdminCatalog*, który udostępnia metody dostępu do kolekcji jako całości lub pojedynczych komponentów (zwanych w tej sytuacji aplikacjami). Dostęp do pojedynczego komponentu obejmuje m.in. numer wersji kom-

ponentu. Możesz również instalować, importować, eksportować, uruchamiać oraz zamykać pojedyncze aplikacje.

Interfejs *ICOMAdminCatalog* udostępnia również inne „makropoleceniowe" możliwości. Na przykład, są takie metody, które umożliwiają ci pracę z więcej niż jedną aplikacją naraz. Ta „makropoleceniowa" zdolność jest rozszerzana na rejestrowanie więcej niż jednej klasy zdarzeń komponentu naraz, co naprawdę oszczędza dużo czasu w trakcie pracy z bardziej złożonymi komponentami.

Istnieją również pewne metody obsługiwane przez interfejs *ICOMAdminCatalog*, które mogą nadawać się, lub nie, do pracy na twoim serwerze. Na przykład, ze względu na to, że Microsoft sprzedaje CLB jako oddzielny produkt, nie istnieje żaden sposób, by dowiedzieć się, czy będziesz miał tę usługę zainstalowaną na swojej maszynie. Jednakże, istnieją metody służące do uruchamiania i zatrzymywania tej usługi, jeżeli tylko jest zainstalowana. Jednym z zadań związanych z usługą, które zawsze będzie dostępne, jest zdolność do tworzenia kopii bezpieczeństwa REGDB (centralnego magazynu danych konfiguracyjnych COM+) i jej późniejszego odtwarzania.

Pula obiektów

Właściwie jest to pojęcie, które bardzo łatwo zrozumieć. Pomyśl o ilości kodu wymaganego do utworzenia egzemplarza obiektu i do zmiany właściwości obiektu. Nie trzeba być naukowcem, by zrozumieć, że zmiana właściwości jest daleko mniej wymagająca w kwestii kodowania niż tworzenie nowego obiektu. To jest właśnie to, o co chodzi w puli obiektów. Zamiast tworzenia nowych obiektów za każdym razem, gdy pojawia się taka potrzeba, a następnie niszczenia ich po zakończeniu pracy z nimi, pula obiektów umożliwia zachowanie obiektu do ponownego wykorzystania w przyszłości.

Zabezpieczenia oparte na rolach

Ochrona to bardzo ważna kwestia, zwłaszcza że komputerowi włamywacze (crackerzy) zaczęli pojawiać się już na prawie każdym polu zastosowań komputerów. Nowości, takie jak rozpowszechnienie kontrolek ActiveX, skryptów języka Java, różnego rodzaju innych skryptów, używanie makropoleceń w aplikacjach oraz nieprzewidywalne błędy programistyczne w najważniejszych produktach — chociaż brzmi to paranoicznie — przyczyniły się do tego, że administratorzy mają mniejszą możliwość ochrony zasobów, zaś crackerzy większą łatwość ich uszkodzenia.

Windows 2000 udostępnia różnorodne sposoby zmniejszania takiego zagrożenia. Jeżeli piszesz aplikację, która będzie korzystać ze wszystkich mechanizmów zabezpieczających systemu Windows 2000 i jeżeli administrator sieci prawidłowo skonfiguruje uprawnienia obiektów w sieci, wtedy ryzyko dokonania włamania jest znacznie mniejsze — ale wciąż istnieje. Krótko mówiąc, ochrona jest nieustannie zmieniającym się procesem zarządzania konfiguracją, uważnego śledzenia zdarzeń i aktualizacji oprogramowania.

Na szczęście, Windows 2000 posiada wiele narzędzi zabezpieczających. Kolejne podrozdziały opiszą dwa elementy składowe tego układu ochrony. Pierwszym jest standardowy mechanizm zabezpieczeń obiektów, który istnieje do momentu powstania systemu Windows NT i, który został znacząco usprawniony w Windows 2000. Zabezpieczenia oparte na obiektach określają wymogi odnośnie do konkretnego poziomu dostępu, które muszą być spełnione przez użytkownika, jeśli ten chce uzyskać dostęp do tego obiektu poprzez użycie żetonu. Drugim elementem jest nowszy mechanizm zabezpieczeń opartych na rolach, którego będziesz używał z wieloma komponentami. W tej sytuacji, obiekty są przydzielane konkretnym rolom, tak jak użytkownicy, którzy potrzebują dostępu do nich. Rola jest ukierunkowana na zadanie. By uzyskać dostęp do obiektu, użytkownik musi wykonywać zadanie, którego wymaga ten obiekt. Te dwie technologie pracują razem, by zapewnić twojej sieci bezpieczeństwo.

Uwaga

W tym rozdziale będą używane dwa terminy na określenie ludzi łamiących zabezpieczenia serwerów WWW lub sieci komputerowych. Pierwsze z nich — *hacker* — odnosi się do kogoś, kto wykonuje niskopoziomowe zadanie, np. sprawdza zabezpieczenia stosowane przez firmę, za jej przyzwoleniem. Hackerzy nie robią nic, co mogłoby uszkodzić sieć lub przechowywane w niej dane. Drugie określenie — *cracker* — określa osobę, która włamuje się na serwery WWW lub do sieci komputerowych, chcąc ukraść dane, uszkodzić dane, sieć lub zainstalować wrogie oprogramowanie (wirusy, konie trojańskie).

Standardowe mechanizmy zabezpieczeń Windows

Windows 2000 pozwala konfigurować mechanizmy ochronne na kilka różnych sposobów, włączając w to dobrze znane zabezpieczenia na poziomie użytkownika i pliku. Możesz również tworzyć grupy, przydzielając prawa dostępu właśnie im, zamiast poszczególnym użytkownikom. Dodatkowo, istnieje możliwość śledzenia każdego aspektu działania systemu zabezpieczeń przy korzystaniu z różnych ostrzeżeń i plików zawierających zapis wszelkiej aktywności. Żadne zdarzenie nie prześlizgnie się bez kontroli i bliższego przyjrzenia się mu. Faktycznie, prosty moment przekazywania informacji z jednego procesu do innego jest przyczyną przeprowadzania przez system Windows 2000 pewnego rodzaju kontroli.

Dla aplikacji poziom zabezpieczeń udostępniany przez Windows 2000 ma zarówno wady, jak i zalety. Z jednej strony, nie ma zbyt wielu słabych punktów., które aplikacje mogą wykorzystać, by złamać zabezpieczenia. W większości przypadków, Windows 2000 po prostu przerwie pracę błędnie działającej aplikacji, zanim zdąży dojść do jakiegokolwiek naruszenia zabezpieczeń. Z drugiej strony, taka surowa ochrona właściwie zbyt mocno ingeruje w pracę niektórych starszych aplikacji, utrudniając im w pełni prawidłową pracę, która jest możliwa w warunkach mniej surowych zabezpieczeń panujących w Windows 95/98. Zasadniczo, zabezpieczenia oferowane przez Windows 2000 mogą właściwie mieć wpływ na poziom zgodności, który będzie mogła

udostępnić twoja maszyna. Na szczęście, dla programisty COM+ rodzaje aplikacji, na które zwykle mają wpływ surowe zabezpieczenia systemu Windows 2000, nie stanowią problemu, więc nie będziemy więcej wracać do tego tematu.

Duża liczba mechanizmów zabezpieczających, które udostępnia Windows 2000, to duży plus dla użytkowników Internetu i witryn WWW. Faktycznie, wielu twórców serwisów WWW chciałoby, aby system Windows 2000 zapewniał jeszcze bardziej surowe reguły zabezpieczeń, niż oferuje obecnie. To poziom własnych zabezpieczeń sprawia, że Windows 2000 jest świetną platformą systemową do uruchomienia witryny WWW. W dodatku, w większości przypadków możesz poprawić standardowe możliwości serwisu WWW, jeżeli napiszesz kod korzystający z mechanizmów oferowanych przez system Windows 2000. W większości przypadków, oznacza to tworzenie komponentów, do których możesz uzyskać dostęp poprzez skrypty ASP, jako dodatek do zwykłych metod konfiguracji zabezpieczeń. Innym sposobem na osiągnięcie tego samego celu jest stworzenie filtra ISAPI, który będzie monitorował zabezpieczenia twojego systemu (tej techniki nie będziemy omawiać). Nawet jeżeli nie uda ci się zapobiec złamaniu twoich zabezpieczeń przez crackera, będziesz mógł przynajmniej śledzić, co on robi, by zminimalizować zakres wyrządzonych szkód. W dodatku, znając miejsce włamania, można w przyszłości wzmocnić je, uszczelniając tym samym zabezpieczenia.

Wskazówka

Pewien sławny hacker powiedział, że nie ma znaczenia czy, lecz kiedy ktoś włamie się na twoją witrynę WWW lub do twojej sieci — śledzenie zdarzeń dotyczących zabezpieczeń pozwoli ci wykryć usterki w zabezpieczeniach, zanim staną się problemem. W pewnych przypadkach, możesz nawet przeprowadzić dokładną kontrolę swojej sieci, korzystając z usług firmy doradczej zatrudniającej hackerów. Należy pamiętać, że zarówno crackerzy, jak i hackerzy często używają tych samych narzędzi i metod do przełamywania zabezpieczeń twojej sieci — różnica polega na tym, że hacker jest tutaj po to, by pomóc ci wzmocnić ochronę twojej sieci — wykryć słabe miejsca w zabezpieczeniach i je naprawić.

Dziury w zabezpieczeniach

Zatem, czy zdobyć się na wysiłek wbudowywania dodatkowych — charakterystycznych dla systemu Windows 2000 — zabezpieczeń w swój komponent lub aplikację? Po pierwsze, to niemożliwe, by konfiguracja zabezpieczeń twojej sieci LAN była perfekcyjna. Musisz się przekonać, że wszystkie części każdej aplikacji zapewniają wymagany poziom zabezpieczeń — inaczej będziesz miał dziury w zabezpieczeniach swojej sieci lokalnej. Obecnie jednym z najbardziej powszechnych źródeł problemów z ochroną w firmach jest użytkownik, który nie rozumie reguł biznesu dotyczących zabezpieczeń. Dodanie do aplikacji i związanych z nimi komponentów, charakterystycznych dla Windows 2000, metod ochrony, pomoże ci pokonać pewnego rodzaju problemy związane z wiedzą użytkownika, lub przynajmniej ostrzec go, że robi coś, co wykracza poza wytyczne firmy.

Nie możesz martwić się jedynie o swoją sieć LAN, trzeba pomyśleć także o Internecie. Nawet jeżeli nie udostępnisz swojej witryny WWW w Internecie, zawsze będą sposoby, które umożliwią włamanie i dokonanie spustoszeń w sieci. Internet otwiera crackerowi wszystkie rodzaje potencjalnych możliwości. Nie ma rady na to, że w obecnych specyfikacjach API dotyczących zabezpieczeń w Internecie, jak również w ich implementacjach używanych przez różne aplikacje znajdują się dziury. W pewnych przypadkach, te dziury w zabezpieczeniach nie znajdują się w samym interfejsie API, ale są wynikiem twórczych rozwiązań ludzi, które były używane w przeszłości do uruchamiania Internetu. W zasadzie można powiedzieć, że rozwiązanie problemu w przeszłości stworzyło dodatkowe wejścia w teraźniejszości.

Bez względu na to, jakie jest źródło dziury w zabezpieczeniach, możesz być pewien, że jakiś cracker czeka na to, by się o niej dowiedzieć. Wykorzystywanie dobrze znanych wyrw w zabezpieczeniach to jedno z najważniejszych narzędzi w świecie crackerów. To nie są tylko konta użytkowników, które powinieneś albo monitorować, albo zablokować (np. *Guest*), to jest sposób, w jaki aplikacja została zaprojektowana. Nawet Java, która dla zapewnienia bezpieczeństwa wykorzystuje mechanizm sandbox, miała w nim dziury. Ostatnio, crackerzy byli w stanie przekonać wirtualną maszynę Javy (JVM), że ma skasować plik na maszynie użytkownika. Podczas gdy niektóre dziury w zabezpieczeniach są niewielkie i trudno uzyskać do nich dostęp, faktem jest, że masz wyrwy w zabezpieczeniach, których cracker może użyć do uszkodzenia twojej sieci.

W nieodległej przeszłości Microsoft był obiektem szykan ze strony prasy branżowej. Oczywiście, po części wynikało to z faktu, że jest jedną z największych korporacji. Crackerzy znaleźli dziury we wszystkich produktach firmy, zaczynając od przeglądarki WWW Internet Explorer i kończąc na pakiecie programów biurowych Microsoft Office. Istnienie dziur wykryto nawet w rzekomo bezpiecznej infrastrukturze usługi ICD (Internet Component Download) — metody służącej do pobierania komponentów ActiveX z sieci Internet. Poniższa lista przedstawia informacje dotyczące trzech dziur zawartych w usłudze ICD (chociaż to prawie pewne, że jest ich więcej):

- *Znacznik HTML <A HREF>* — istnieją sposoby na pobranie i uruchomienie pliku *EXE* za pomocą znacznika języka HTML `<A HREF>`. Metodą, z której korzysta przeglądarka Internet Explorer 3.x/4.x/5.x do utrzymywania tego problemu pod kontrolą jest obecnie to, by analizator składni HTML wykorzystywał bezpośrednio przydomek URL do pobierania kodu. Wtedy następuje wywołanie *WinVerifyTrust*, będące częścią usługi ICD mające na celu weryfikację wiarygodności kodu. Czy ta metoda jest w 100% bezpieczna? Nie, ponieważ używasz czegoś innego niż standardowa procedura weryfikująca zawartość pliku. W tej sytuacji polegasz na analizatorze składni HTML. Proponowanym rozwiązaniem problemu, sugerowanym przez Microsoft, jest odmowa pozwolenia na pobieranie kontrolek ActiveX (gdy padnie takie pytanie ze strony przeglądarki) oraz to, że powinieneś sprawdzić każdy pobrany z sieci plik EXE przed jego uruchomieniem. Jednakże, możliwość obejścia tych dwóch sugestii wciąż istnieje i niedoświadczony użytkownik może zainfekować wirusem twoją sieć, zanim zdążysz się zorientować.
- *Skrypty* — obecnie skrypty nie są w ogóle poddawane kontroli pod kątem wykrywania dziur w zabezpieczeniach. Nie istnieje sposób na sprawdzenie, kto jest autorem danego skryptu lub co mógłby on zrobić na twojej maszynie. Nie ma także sposobu na to, by sprawdzić, jakie informacje skrypt może pobierać z twojej maszyny. Obecnie Microsoft pracuje nad stworzeniem pewnego rodzaju certyfikatu dla skryptów. Jednakże, nawet w przypadku Internet Explorera, najlepiej po prostu wyłączyć ich przetwarzanie, jeżeli

wirus dokonuje ataku za pośrednictwem złośliwego skryptu. W momencie gdy certyfikaty skryptów ujrzą światło dzienne, przeglądarka WWW będzie mogła dokonać wywołania *WinVerifyTrust* w celu przeprowadzenia kontroli skryptu, zanim zostanie on uruchomiony.

- *Całe aplikacje lub inne złożone sytuacje pobierania* — Internet Explorer potrafi kontrolować określone rodzaje pobieranych danych. Na przykład, pobieranie pliku OCX rozpoczyna sekwencję *WinVerifyTrust*. Co się dzieje, jeżeli parametry pobierania nie mieszczą w ograniczonym zakresie spraw sprawdzanych przez Internet Explorera? Użytkownik chciałby pobrać i zainstalować, dajmy na to, Dooma, Quake'a lub inną grę. Proces instalacji może zawierać nieprzewidywalne akcje, takie jak: dodawanie wpisów w rejestrze systemowym czy ponowne uruchomienie komputera. Obecnie Internet Explorer nie potrafi sobie poradzić z taką sytuacją. Firma Microsoft planuje uczynić przyszłą wersję usługi ICD bardziej solidną i poszerzyć jej możliwości o obsługę takich zdarzeń, jak te opisane wcześniej. Krótko mówiąc, nie ma znaczenia, jak mocno jesteś dzisiaj pewien bezpieczeństwa swojej sieci, ponieważ istnieje duża liczba dziur w zabezpieczeniach, których crackerzy mogą użyć w celu uzyskania dostępu do twojej sieci.

Częścią twojej strategii ochrony powinno być przeprowadzenie testu kilku przeglądarek, by poznać ich reakcje — zwłaszcza wtedy, gdy tworzysz witrynę WWW przeznaczoną do ogólnego dostępu. Jako programista pracujący z wieloma produktami powinieneś regularnie śledzić zmiany na wszystkich odwiedzanych przez siebie serwisach WWW w poszukiwaniu nowych pomysłów. Nowsze wersje przeglądarek Internet Explorer i Netscape Communicator uwzględniają import-eksport pewnej ilości danych aplikacji. Jednakże, jeżeli używasz starszej wersji którejś z tych przeglądarek, produkt o nazwie NavEx pozwoli ci stworzyć kopie folderu Ulubione używanego przez Internet Explorera i zapisanie go w postaci zakładek Netscape Communicatora i na odwrót. Znajdziesz go w następujących miejscach w sieci, skąd będziesz mógł go pobrać i przetestować: *http://www.pcworld.com/fileworld/file_description/frameset/0,1458,4008,00*, lub *http://www.zdnet.com/pcmag/pctech/content/17/01/ ut1701.001.html*.

Uwaga

W tym rozdziale przyjrzymy się również dziurom w innych rodzajach technologii zabezpieczeń (zwłaszcza tym, które pomagają w ochronie danych w Internecie). Tabele 3.1 i 3.2 są pełne nowych specyfikacji zaprojektowanych do zatkania dziur w obecnie używanych technologiach. SHTTP, S/WAN oraz inne technologie nie byłyby potrzebne, gdyby ochrona nie sprawiała problemów. Istnieje nawet nowa wersja MIME o nazwie S/MIME, która daje pewność, że nikt nie przeczyta twojej poczty.

Oczywiście, ta lista wskazuje jedynie trzy dziury w jednej usłudze systemu Windows. Teraz, zastanów się, ile usług masz uruchomionych na swojej stacji roboczej i związanym serwerze.

Nawet jeżeli każda usługa ma tylko jedną dziurę, możliwości do naruszenia systemu zabezpieczeń będzie naprawdę dużo. Możesz jednak dodać zabezpieczenia do swoich komponentów i aplikacji, by mieć możliwość śledzenia takiego dostępu i (lub) zablokowania go. Zamykanie dziur i wypełnianie szczelin w konfiguracji zabezpieczeń jest sprawą ważną, jeżeli zależy ci na zachowaniu danych, nad zdobyciem których twoja firma tak ciężko pracowała. Krótko mówiąc, będziesz musiał wiedzieć, jakie mechanizmy zabezpieczeń oferuje system Windows 2000 — potraktuj to jako swój pierwszy krok w kierunku zabezpieczenia swojego systemu.

Aby zrozumieć, w jaki system Windows 2000 może pomóc ci wzmocnić ochronę twojej sieci jako całości, należy przyjrzeć się podstawowym funkcjom zabezpieczeń udostępnianym przez sam system operacyjny. Do każdego mechanizmu zabezpieczeń (np. odwzorowanie napędów dyskowych), który ma do zaoferowania system Windows 2000, można uzyskać dostęp z poziomu aplikacji. Wielu programistów nie pamięta o tym, że dostęp do funkcji mechanizmów zabezpieczeń Windows 2000 można również uzyskać za pomocą kontrolek ActiveX. Dla programisty COM+ oznacza to konieczność implementacji mechanizmów zabezpieczających komponenty. Takie podejście zapobiegnie różnym rodzajom nieautoryzowanego dostępu. Oczywiście, należy przede wszystkim określić tożsamość użytkownika żądającego dostępu do komponentu, a następnie ograniczyć dostępności jego funkcji tylko do tych, których naprawdę potrzebuje.

Możesz w bardzo prosty sposób dokonać implementacji mechanizmów zabezpieczających systemu Windows 2000. Na przykład, tworzenie kontrolki ActiveX do wyświetlania okna dialogowego służącego do przeprowadzania logowania właściwie służy dwóm sprawom w każdej aplikacji, zwłaszcza tym, do których dostęp można uzyskać spoza firmy za pomocą Internetu. Po pierwsze, pozwala ci zweryfikować tożsamość tego konkretnego użytkownika i dostosować poziom dostępu do aplikacji. Cracker, nie znając hasła, będzie miał trudności w zalogowaniu się. Po drugie, możesz tak skonfigurować serwer, by zapisywał każdy bezpieczny dostęp — pamiętaj, że Windows 2000 daje ci możliwość nadzoru nad wszystkim. Jeżeli ktoś poradzi sobie z przełamaniem zabezpieczeń twojego systemu, będziesz przynajmniej znał konto, które zostało użyte do włamania. Znajomość nazwy konta pozwoli ci oszacować poziom uszkodzeń i strat, które cracker mógł spowodować, korzystając uprawnień przyznanych osobie, której konto zostało użyte do włamania.

Przegląd standardów zabezpieczeń

Standardy zabezpieczeń są ważną częścią ochrony bezpieczeństwa sieci. W poprzedniej części ostrzegaliśmy cię o istnieniu dziur w zabezpieczeniach sieci, które biorą się z różnych źródeł — włączając w to system operacyjny i oprogramowanie „z półki", którego używasz do prowadzenia swojej firmy. Aby mieć pewność bezpieczeństwa swoich danych i całego systemu, na którym pracujesz, skorzystaj ze standardów zabezpieczeń

Użytkowanie zestandaryzowanych zabezpieczeń daje oczywiste korzyści. Twoje metody zabezpieczeń zazębią się z tymi używanymi przez inne lokalizacje, redukując krzywą nauki użytkownika i umożliwiając korzystanie z narzędzi stworzonych przez innych programistów. Ponadto, wiedząc, gdzie mogą pojawić się naruszenia bezpieczeństwa, będziesz mógł stworzyć komponenty, których zadaniem jest monitorowanie w celu ich wyszukiwania do chwili, gdy odpowiednie komisje standaryzacyjne nie określą sposobu rozwiązania problemu.

Istnieją właściwie dwa poziomy standardów zabezpieczeń, o których należy pomyśleć. Pierwszy z nich zawiera wewnętrzne zabezpieczenia udostępniane przez system Windows 2000. Będziesz z nich korzystać przed lokalnymi zagrożeniami bezpieczeństwa, które istnieją w twojej sieci LAN lub WAN. Te środki ochrony są równie ważne w przypadku korzystania z Internetu. Ich przegląd zawiera tabela 3.1.

WWW **Jeżeli chcesz poznać najnowsze informacje na temat najnowszych standardów bezpieczeństwa (zwłaszcza w Internecie), zajrzyj pod adres: *http://www.w3.org/Security/*. Co prawda, znajdziesz tam jedynie ogólne informacje, ale są tam również łącza do innych miejsc, gdzie są umieszczone bardziej szczegółowe materiały. Programiści, chcący spojrzeć na kwestię zabezpieczeń komercyjnie, powinni zajrzeć pod adres: *http://www.rsasecurity.com/*. Witryna RSA zawiera różnorodne tematy, włączając obecny stan wysiłków firm MasterCard i VISA, by umożliwić bezpieczne transakcje za pomocą kart kredytowych. Z tego miejsca powinieneś również zaczynać poszukiwania informacji dotyczących dodawania obsługi nowych technologii fizycznego zabezpieczania (np. smartcard) do swoich aplikacji. Możesz również zapoznać się z osiągnięciami IETF, przeglądając dokument znajdujący się pod adresem: *ftp://ftp.isi.edu/internet-drafts/1id-abstracts.txt*.**

Tabela 3.1. *Standardy wewnętrznych zabezpieczeń systemu Windows 2000*

Standard	Opis
Inicjatywa cyfrowych podpisów (DSI — Digital Signature Initiative)	Standard zapoczątkowany przez W3C w celu przezwyciężenia pewnych ograniczeń zabezpieczeń na poziomie kanału transmisji. Zabezpieczenia na tym poziomie nie radzą sobie z semantyką nazw dokumentów i aplikacji. Występują również problemy z wydajnym wykorzystaniem szerokości pasma sieci Internet, ponieważ całość procesu przetwarzania odbywa się w sieci, zamiast po stronie klienta lub serwera. Ten standard określa matematyczną metodę przesyłania podpisów — będących zasadniczo unikalną reprezentacją pojedynczej osoby lub firmy. DSI również zapewnia nową metodę oznaczania właściwości zabezpieczeń (PICS2) i nowy format zapewnień (PEP). Ten standard jest również oparty na standardach PKCS7 i X509.v3. Więcej informacji znajdziesz pod adresem: *http://www.w3.org/Dsig/Overview.html*.
Protokół zabezpieczeń IP (IPSec — Internet Protocol Security Protocol)	IETF stworzył specjalną grupę roboczą (IP Security Protocol Working Group), by przyjrzała się problemom zabezpieczeń protokołu IP — chociażby jego niemożności szyfrowania danych na poziomie protokołu. Obecnie grupa pracuje nad specyfikacjami, które ostatecznie przyczynią się do zwiększenia bezpieczeństwa transakcji IP. Więcej informacji o tej grupie roboczej znajdziesz pod adresem: *http://www.ietf.cnri.reston .va.us/html.charters/ipsec-charter.html*.
Usługa uwierzytelniania sieciowego Kerberos (Kerberos Network Authentication Service)	Jest to zatwierdzona specyfikacja IETF, która określa niezależny protokół uwierzytelniania. Model Kerberos jest częściowo oparty na zaufanym, niezależnym protokole uwierzytelniania Needhama i Schroedera oraz modyfikacjach zaproponowanych przez Denninga i Sacco. Tak jak

IETF RFC1510	w przypadku wielu innych protokołów uwierzytelniania w Internecie, Kerberos działa jako zaufana, niezależna usługa uwierzytelniania. Korzysta z konwencjonalnej kryptografii, która posługuje się kombinacją ogólnie dostępnego klucza publicznego i nie rozpowszechnianego klucza prywatnego. Kerberos kładzie nacisk na uwierzytelnianie klienta z opcjonalną weryfikacją serwera.
Technologia prywatnej komunikacji (PCT — Private Communication Technology)	IETF i firma Microsoft pracują razem nad tym konkretnym protokołem. Podobnie jak SSL, PCT jest zaprojektowany, by zapewnić bezpieczny sposób komunikacji pomiędzy serwerem i klientem na niskim poziomie protokołu. Może współpracować z każdym protokołem wysokiego poziomu, np. HTTP, FTP czy TELNET. Aktualne informacje na temat tej nowej technologii zabezpieczeń znajdziesz pod adresem:. *http://www.graphcomp .com/info/specs/ms/pct.htm*.
Protokół SSL (Secure Socket Layer)	Jest to standard W3C, początkowo proponowany przez Netscape, służący do przesyłania zaszyfrowanych danych między klientem a serwerem na poziomie protokołu. Umożliwia niskopoziomowe szyfrowanie transakcji w protokołach wyższego poziomu, np. HTTP, NNTP i FTP. Standard również określa metody uwierzytelniania serwera i klienta (chociaż weryfikacja lokalizacji klienta jest opcjonalna). Więcej szczegółów na temat protokołu SSL znajdziesz pod adresem: *http://home.netscape.com/security/index.html*.

Po zapoznaniu się z wymogami wewnętrznego bezpieczeństwa, będziesz musiał poznać te, które zawsze pojawiają się poza firmą — mianowicie w Internecie. Przyjrzyjmy się różnym rodzajom standardów zabezpieczeń internetowych, które albo już się nimi stały, albo będą nimi już wkrótce. Tabela 3.2 pokazuje listę standardów lub ich projektów, które były dostępne w trakcie powstawania tej książki. Tak więc, w momencie gdy czytasz te słowa może być dostępna już większa liczba tych standardów.

Tabela 3.2. *Standardy zabezpieczeń w Internecie*

Standard	Opis
Distributed Authentication Security Service (DASS) IETF RFC1507	DASS jest dziełem IETF. Definiuje eksperymentalną metodę udostępniania usług uwierzytelniania w Internecie. Celem weryfikacji w tym wypadku jest określenie, kto wysłał komunikat albo żądanie. Obecne schematy haseł mają pewne problemy, które DASS próbuje rozwiązać. Na przykład, nie ma sposobu, by dokonać weryfikacji, że nadawca hasła nie podszywa się pod kogoś innego. DASS udostępnia usługi uwierzytelniania w środowisku rozproszonym, które jest źródłem wyzwań specjalnego rodzaju — użytkownicy nie logują się po prostu na jedną maszynę; istnieje potencjalna możliwość, że mogliby zalogować się na każdy komputer w sieci. Więcej informacji na temat tego standardu znajdziesz pod następującym adresem: *http://www.wu-wien.ac.at:8082/rfc/rfc1507.hyx/$$root*. Warto również spojrzeć na listę innych standardów zabezpieczeń znajdującą się pod adresem: *http://afs.wu-wien.ac.at/usr/edvz/gonter/rfc-list.html*
Generic Security Service Application Program Interface (GSS-API)	Jest to zatwierdzona specyfikacja IETF, która określa metody ogólnej obsługi wywołań usług zabezpieczeń. Używając standardowego interfejsu, pozwala na większą mobilność kodu źródłowego pomiędzy szerszym zakresem platform systemowych. IETF nie postrzega tej specyfikacji jako końca procesu, lecz

IETF RFC1508	raczej punkt wyjścia do innych, bardziej dokładnych standardów w przyszłości. Jednakże, świadomość istnienia tego standardu może pomóc ci odnaleźć podobieństwa między różnymi metodami implementacji zabezpieczeń. Więcej informacji odnośnie tego standardu znajdziesz pod adresem: *http://www.wu-wien.ac.at:8082/rfc/rfc1508.hyx/$$root*. Warto spojrzeć także na listę innych standardów zabezpieczeń. Można ją znaleźć pod adresem: *http://afs.wu-wien.ac.at/usr/edvz/gonter/rfc-list.html*
Generic Security Service Application Program Interface (GSS-API) C — bindings IETF RFC1508	Jest to zatwierdzona specyfikacja IETF, która określa metody obsługi wywołań usług używających języka C. To jedna z pierwszych określonych implementacji standardów opartych na RFC1508.
Joint Electronic Payment Initiative (JEPI)	Standard zapoczątkowany przez W3C. JEPI udostępnia metodę służącą do tworzenia usług handlu elektronicznego. Do przeprowadzenia transakcji będzie używana jakaś forma elektronicznej gotówki lub karty kredytowe. Dane przesyłane między klientem i serwerem będą używać szyfrowania, cyfrowych podpisów oraz uwierzytelniania (wymiana kluczy), by w ten sposób zapewnić bezpieczeństwo wymiany informacji. Niektóre elementy, takie jak zabezpieczenia na poziomie transportu (również zwane prywatnością), dopiero są definiowane przez IETF. Więcej o tym standardzie znajdziesz pod adresem: *http://www.w3.org/Ecommerce/Overview-JEPI.html*.

Tabela 3.2. *Standardy zabezpieczeń w Internecie — ciąg dalszy*

Standard	Opis
Privacy Enhanced Mail Part I (PEM1) Message Encryption and Authentication Procedures IETF RFC1421	Jest to zatwierdzona specyfikacja IETF, która ma zapewnić, że twoja prywatna korespondencja elektroniczna taka pozostanie. Zarysowuje szkic procedury szyfrowania poczty w taki sposób, że poczta użytkownika jest bezpieczna, a proces odszyfrowywania jest dla niego niewidoczny. Obejmuje to obsługę kluczy i zarządzania innymi formami certyfikatów. Część specyfikacji jest oparta na CCITT X.400 — zwłaszcza w obszarach usługi obsługi poczty (Mail Handling Service — MHS) i agenta przesyłania wiadomości (Mail Transfer System — MTS). Więcej wiadomości na temat tego standardu znajdziesz pod adresem: *http://www.ietf.org/rfc/rfc1421.txt.*
Privacy Enhanced Mail Part II (PEM2) Certificate-Based Key Management IETF RFC1422	Jest to zatwierdzona specyfikacja IETF, która służy do zarządzania kluczami zabezpieczeń. Udostępnia zarówno infrastrukturę, jak i architekturę zarządzania opartą na technice certyfikacji za pomocą klucza publicznego. Standard IETF RFC1422 jest rozszerzeniem specyfikacji CCITT X.509. Wykracza poza granice określone przez CCITT w X.509, udostępniając procedury i konwencje dla infrastruktury zarządzania kluczami przeznaczone do użytku z PEM. Więcej o tym standardzie znajdziesz pod adresem: *http://www.si.hhs.nl/~henks/comp/crypt.html*.
Privacy Enhanced Mail Part III (PEM3) Algorithms, Mode, Identifiers IETF RFC1423	Jest to zatwierdzona specyfikacja IETF, która określa algorytmy kryptograficzne, tryby używania oraz identyfikatory do używania z PEM. Specyfikacja obejmuje cztery główne obszary informacji związanych z szyfrowaniem. Są to następujące algorytmy: szyfrowania wiadomości, kontroli ich integralności, zarządzania kluczami symetrycznymi i asymetrycznymi (włączając w to zarówno algorytmy symetrycznego szyfrowania, jak i asymetrycznych podpisów). Więcej o tym standardzie

	dowiesz się pod adresem: *http://www.si.hhs.nl/ ~henks/comp/crypt.html*.
Privacy Enhanced Mail Part IV (PEM4) Certification and Related Services IETF RFC1424	Jest to zatwierdzona specyfikacja IETF, która określa metodę certyfikacji kluczy. Również udostępnia listę usług związanych z kryptografią, które witryna internetowa potrzebowałaby udostępnić użytkownikowi. Więcej informacji dotyczących tego standardu znajdziesz pod adresem: *http://www.si.hhs.nl/~henks/comp/crypt.html*.
Secure/Multipurpose Internet Mail Extensions (S/MIME)	Ta specyfikacja jest promowana przez konsorcjum producentów, m.in. Microsoft, Banyan, VeriSign, QUALCOMM, Frontier Technologies, Network Computing Devices, FTP Software, Wollongong, SecureWare i Lotus. Początkowo była rozwijana przez RSA Data Security, Inc., jako metoda dla innych programistów do tworzenia agentów przesyłania wiadomości (MTA), które używają zgodnej technologii szyfrowania. Oznacza to, że jeżeli ktoś wysyła do ciebie wiadomość, korzystając z produktu Lotusa, ty możesz ją odczytać za pomocą programu firmy Banyan. S/MIME jest oparty na popularnym standardzie internetowym MIME (RFC1521). Informacje o standardzie MIME znajdziesz pod adresem: *http://www.nacs.uci.edu/indiv/ehood/MIME/MIME.html.* Całą listę zasobów charakterystycznych dla S/MIME można pobrać pod adresem: *http://www.rsasecurity.com/standards/smime/resources.html*.

Tabela 3.2. *Standardy zabezpieczeń w Internecie — ciąg dalszy*

Standard	Opis
Bezpieczna sieć rozległa (S/WAN — Secure Wide Area Network)	W tej chwili S/WAN znajduje się w początkowej fazie swojego rozwoju. Jest to inicjatywa wspierana przez RSA Data Security, Inc. IETF również posiada komisję, która pracuje nad tym standardem. RSA chciałaby włączyć standard IETF — IPSec do specyfikacji S/WAN. Głównym celem S/WAN jest umożliwienie firmom wyszukania najlepszego firewall'a oraz produktów stosu TCP/IP i wykorzystanie do budowy wirtualnych sieci prywatnych (VPN) opartych na Internecie. Obecne rozwiązania często zmuszają do korzystania z obydwu produktów pochodzących z jednego źródła (od tego samego producenta). Więcej o S/WAN znajdziesz pod adresem: *http://www.rsasecurity.com/rsalabs/faq/5-1-3.html*.
Bezpieczny protokół transmisji hipertekstu (SHTTP — Secure Hypertext Transfer Protocol)	Bieżąca technologia szyfrowania przesyłanych danych używana przez Open Marketplace Server, który ma podobne możliwości, jak protokół SSL. Różnica polega na tym, że działa on z protokołem HTTP. Grupa WTS (Web Transaction Security) instytucji IETF powstała ostatnio właśnie w celu przygotowania takiej specyfikacji jak ta. Więcej informacji o tym standardzie znajdziesz pod adresem: *http://www.ietf. org/html.charters/wts-charter.html*.
Uniwersalne identyfikatory zasobów (URI — Universal Resource Identifiers) IETF RFC2396 oraz inne np. RFC1630	URI to bieżące zadanie IETF. Obecnie nazwy zasobów i adresy są udostępniane w postaci zwykłego tekstu. URL jest właściwie formą URI, która zawiera adres odwzorowywujący konkretne miejsce w Internecie. URI udostępniłby środki szyfrowania nazw i adresów obiektów w Internecie. W zasadzie, by odwiedzić prywatną witrynę WWW, musiałbyś znać zaszyfrowaną nazwę zamiast nazwy wyrażonej zwykłym tekstem. Jeżeli chciałbyś dowiedzieć się czegoś więcej na temat URI i jak wypada on w porównaniu do URL, zajrzyj na witrynę W3C znajdującą się pod adresem: *http://www.w3.org/ Addressing/*.

WWW **Tabele 3.1 oraz 3.2 dają przejrzysty obraz standardów związanych z**

zabezpieczeniami stworzonymi przez IETF. Większość dokumentów IETF RFC możesz znaleźć pod adresem: *http://www.ietf.org/rfc/*, zaś listę obecnie działających grup roboczych IETF pod adresem: *http://ietf.cnri.reston.va.us/ html.charters/*. Te wszystkie grupy robocze pomagają w tworzeniu standardów używanych w Internecie.

To zadziwiające, że standardy producenta są prawdopodobnie najszybciej rozwijającym się obecnie obszarem w Internecie poza technologią przeglądarek (która wydaje rozwijać się tak szybko, że nawet betatesterzy mają duży problem, by dotrzymać kroku zmianom). Zauważysz również, że większość standardów znajdujących się tabeli 3.1 oraz 3.2 pochodzi od dwóch grup: IETF (Internet Engineering Task Force) lub W3C (World Wide Web Consortium). IETF istnieje od dawna. Jest to jedna z pierwszych grup pracujących nad sprawami związanymi z Internetem. Chcąc zagłębiać się w zagadnienia związane z bezpieczeństwem w sieci Internet (i w mniejszym wymiarze — o innych obszarach standardów, takich jak znaczniki języka HTML), będziesz dużo czytał o grupie W3C. Na przykład, firma Microsoft obecnie stara się o zatwierdzenie przez W3C znacznika <OBJECT> oraz innych rozszerzeń języka HTML związanych z ActiveX.

Standardy zabezpieczeń znajdujące się w tabeli 3.2 prezentują spojrzenie na bezpieczeństwo ze strony Internetu. To ważne, by zapamiętać ten fakt. Standardy te określają połączenie między klientem i serwerem. Wciąż możesz dodawać kolejne środki zabezpieczające na kliencie, serwerze lub obydwu naraz (tak jak pokazano to w tabeli 3.1). Większość tych standardów nie obejmuje swoim zasięgiem dodatkowych produktów internetowych, takich jak firewall'e. Twoja firma może oczywiście skorzystać z tych dodatkowych mechanizmów zabezpieczających.

Konfiguracja systemu zabezpieczeń Windows 2000

Odkryliśmy już niektóre z wyzwań dotyczących ochrony w Windows 2000 (w postaci dziur w zabezpieczeniach) i standardów, które starają się naprawić te problemy. Jednakże, wciąż nie przyjrzeliśmy się interfejsowi API zabezpieczeń systemu Windows 2000. Jest to część systemu operacyjnego Windows 2000, która umożliwia programiście pełną kontrolę nad sposobem, w jaki aplikacje i komponenty będą obsługiwać zabezpieczenia. Przyjrzyjmy się, jak można zaimplementować te mechanizmy zabezpieczeń w świecie realnym. Poniższa lista opisuje dziewięć poziomów zabezpieczeń systemu Windows 2000:

- *Wbudowane zabezpieczenia* — Windows 2000 posiada pewien poziom zabezpieczeń, który stanowi część systemu operacyjnego. Pod Windows 2000 każdy obiekt posiada pewien rodzaj zabezpieczeń ze sobą związany i warto przyjrzeć się sposobowi, w jaki są one implementowane. Korzystanie z zabezpieczeń na poziomie obiektu oznacza, że istnieje niewielka szansa, by każdy mógł uzyskać dostęp do każdej części systemu operacyjnego lub jego danych bez odpowiedniej autoryzacji. Oczywiście, niewielka szansa nie oznacza, że taka szansa w ogóle nie istnieje.
- *Podpis Authenticode* — jest to część programu Internet Explorer. Dokonuje weryfikacji, czy pobierane z Internetu komponenty nie zostały w jakikolwiek sposób sfałszowane, i zapewnia określony poziom odpowiedzialności za komponent (poprzez wyświetlanie użytkownikowi nazwiska jego autora oraz innych informacji). W zasadzie jest to równoznaczne z podpisywaniem komponentów. Podpis Authenticode obsługuje trzy różne rodzaje cyfrowych certyfikatów: 128-bitowe cyfrowe podpisy generowane

lokalnie, przemysłowy standard kryptografii opartej na kluczu publicznym (PKCS) #7 i #10 oraz cyfrowe certyfikaty X.509 w wersji 3.

- *Interfejs API obsługujący kryptografię (CryptoAPI)* — niektóre rodzaje zabezpieczeń korzystają podczas pracy z warstw ochrony. Innymi słowy, jeżeli cracker przełamałby jedną warstwę zabezpieczeń, inna istniałaby nadal, by zapobiec dalszej ingerencji w system. CryptoAPI to warstwa ochrony dodana do wszystkich innych istniejących już warstw zabezpieczeń. Służy do powstrzymania innych od dalszego penetrowania twojego systemu zabezpieczeń.
- *Cyfrowe podpisy-certyfikaty* — cyfrowy podpis działa tak, jak sugeruje jego nazwa. Nadawca dokumentu lub pliku wykonywalnego podpisuje go. Sprawdzając ten podpis, upewniasz się, że „dzieło" jest autentyczne. By wdrożyć ten poziom ochrony, użyjesz serii kluczy prywatnych i publicznych.
- *Uwierzytelnianie rozproszonych haseł (DPA)* — udostępniona tajna metoda uwierzytelniania początkowo stosowana przez niektóre z większych serwisów sieciowych (CompuServe i MSN). Pozwala użytkownikowi korzystać z tego samego hasła członkowskiego, by uzyskać dostęp do pewnej liczby witryn w Internecie, które są powiązane razem w całość jako jedna organizacja członkowska. Innymi słowy, metoda ta polega na replikacji niektórych identycznych funkcji wielu serwerów w sieci lokalnej, do których użytkownicy mogą uzyskać dostęp, posługując się tym samym hasłem. DPA korzysta z usługi MMS (Microsoft Membership Service) zapewniającej uwierzytelnianie członkostwa oraz informacji dostępowych specyficznych dla serwera.
- *Kerberos* — jest to główny zamiennik dostawcy zabezpieczeń o nazwie *Windows NT LAN Manager (NTLM)*, który jest używany do zapewnienia bezpieczeństwa twoim danym podczas pracy z systemem Windows. Kerberos wersja 5 jest relatywnie nowym standardem przemysłowym protokołu zabezpieczeń stworzonym przez specjalistów z MIT (Massachussets Institute of Technology) i oferuje pierwszorzędną obsługę zabezpieczeń uzyskiwaną poprzez wykorzystanie architektury klucza prywatnego. Protokół ten obsługuje obustronne uwierzytelnianie klienta i serwera, redukuje obciążenie serwera podczas ustanawiania połączenia i umożliwia klientowi delegację (przekazanie) uwierzytelniania serwerowi poprzez wykorzystanie mechanizmów pośredniczących (proxy). Kerberos łączy się bezpośrednio z *centrum dystrybucji kluczy* (KDC, *Key Distribution Center*) i kontem *usługi katalogowej* (DS, *Directory Service*) w celu uzyskania kluczy sesyjnych niezbędnych do prawidłowego uwierzytelniania.
- *Technologia komunikacji prywatnej (PCT — Private Communication Technology)* — jest to specjalny poziom zabezpieczeń w Internecie, nad którym wspólnie pracują Microsoft i IETF. PCT umożliwi klientowi i serwerowi nawiązanie bezpiecznego połączenia z niewielką szansą na podsłuchanie. Skuteczność działania zabezpieczeń na tym poziomie zależy od takich technologii, jak: podpisy cyfrowe i szyfrowanie danych.
- *Infrastruktura klucza publicznego (PKI)* — jest to protokół umożliwiający dwóm lokalizacjom wymianę zaszyfrowanych danych (nie wymagający żadnych wcześniejszych przygotowań). Domyślną metodą rozpoczęcia wymiany danych jest nawiązanie bezpiecznego połączenia za pomocą protokołu *zabezpieczeń warstwy łączy* (SSL, *Secure Socket Layer*). Główną różnicą między tą technologią a pozostałymi obecnymi na rynku jest to, że korzysta ona z systemu certyfikatów klucza publicznego służącego

do przeprowadzania bezpiecznego transferu danych. Najnowsza specyfikacja protokołu SSL to SSL3, przez organizację IETF nazywana również protokołem *zabezpieczenia warstwy transportu* (TLS, *Transport Layer Security*). Nowszym dodatkiem jest technologia PCT. PCT wciąż korzysta z metody szyfrowania kluczem publicznym, lecz istnieją pewne wyraźne zalety wynikające z jego użytkowania, które omówimy w tym podrozdziale. Jedną z korzyści wynikających z używania technologii PKI jest to, że nie istnieje wymóg korzystania z usług żadnego serwera autoryzującego, ponieważ certyfikat jest wydawany przez dobrze znaną instytucję certyfikującą (np. firmy Thawte czy VeriSign — w przypadku ogólnodostępnego wykorzystywania tej technologii).

- ✧ *Uwierzytelnianie Windows 2000 poprzez HTTP* — wielu ludzi przyjmuje błędne założenie, że zabezpieczenia muszą być trudne do zrozumienia lub bardzo złożone. Ale wcale tak być nie musi. Pod Windows 2000 uwierzytelnianie może przybierać dwie postacie, gdy przyglądamy się tej kwestii z perspektywy Internetu. Po pierwsze, poproś o nazwę użytkownika i hasło, a następnie sprawdź te informacje z listą dostępu na serwerze. Istnieją dwa sposoby, by to zrobić: podstawowa metoda, która jest używana podczas standardowej procedury logowania i w mechanizmie wyzwanie-odpowiedź systemu Windows 2000. Druga metoda korzysta z informacji od klienta, który dostarcza wymaganą nazwę użytkownika i hasło na podstawie bieżącego ustawienia sesji. Klient dostarcza informacji, z których korzysta użytkownik, by zalogować się do maszyny. Inna metoda polega na istniejącej technologii, zwanej SSL (*Secure Socket Layer*). SSL korzysta w swojej pracy z algorytmów szyfrowania oraz cyfrowych certyfikatów. Nic nie stoi na przeszkodzie, aby wykorzystać równocześnie obydwa mechanizmy zabezpieczeń — nie wykluczają się nawzajem.

Wbudowane mechanizmy zabezpieczeń

Windows 2000 jest chyba najbardziej przeprojektowanym systemem operacyjnym dostępnym obecnie na rynku, jeżeli chodzi o kwestie zabezpieczeń. Jeżeli masz jakieś wątpliwości na temat znaczenia zabezpieczeń, przyjrzyj się możliwościom prezentowanym przez Windows 2000. Możesz ich używać według własnego uznania. Na przykład, możesz zrobić użytek z możliwości zaawansowanych, wbudowując ich obsługę w komponent lub przyznać dostęp do określonych funkcji programu poprzez Internet. Windows 2000 udostępnia różne rodzaje ochrony stosownie do potrzeb użytkownika. Niektóre z tych mechanizmów są oparte na bieżących standardach zabezpieczeń.

Oczywiście, sposób używania mechanizmów zabezpieczających systemu Windows 2000 zależy od użytkownika oraz rozmieszczenia aplikacji. Internet nakłada ograniczenia, ponieważ niektóre elementy ochrony znajdują się poza twoją kontrolą. Należy wykazać dużo rozsądku podczas przydzielania dostępu do funkcji aplikacji w Internecie, aby nie uszkodzić systemu zabezpieczeń swojej sieci. W dodatku, korzystanie z pewnych funkcji poprzez Internet jest po prostu niepraktyczne, bo po prostu nie są one ważne. (Inne mechanizmy zabezpieczeń zajmują miejsce systemów lokalnej ochrony — patrz tabele 3.1 oraz 3.2).

Czy tworzysz komponenty dla Internetu, sieci WAN, LAN lub lokalnego użytku, zauważysz, że zrozumienie wewnętrznej architektury systemu zabezpieczeń jest sprawą zasadniczą. Na przykład, musisz zdecydować, czy komponent kliencki lub serwerowy jest lepiej dostosowany

do architektury maszyny klienta. System Windows 95/98 nie udostępnia takiego samego poziomu zabezpieczeń, jak Windows 2000, więc będziesz pracował bez dodatkowych zabezpieczeń pod Windows 95/98. Jednakże, gdy Windows 95/98 udostępnia mechanizm zabezpieczeń, używa tego samego zestawu parametrów, co Windows 2000, więc jeden moduł zabezpieczeń będzie pracował pod kontrolą obydwu systemów operacyjnych. (W innych przypadkach, będziesz definitywnie dążył do korzystania z oddzielnego modułu przeznaczonego wyłącznie dla Windows 2000, by lepiej wykorzystać zaawansowane mechanizmy zabezpieczające). Problemem dla programisty jest zapewnienie spójnego poziomu zabezpieczeń wraz z redukcją ruchu w sieci. Komponenty klienckie redukują ruch w sieci, ponieważ to klient wykonuje większą część przydzielonej pracy. Komponenty serwerowe są zaś bardziej bezpieczne, gdyż to zwykle serwer dysponuje lepszymi i bardziej wydajnymi mechanizmami zabezpieczającymi.

Windows 2000 naturalnie korzysta z zabezpieczeń obiektów. Co wynika z takiego schematu zabezpieczeń? W pierwszej kolejności należy zdefiniować określenie „obiekt". Windows 2000 oraz Windows 95/98 używają tego pojęcia raczej luźno. To prawda, że duża ilość obiektów skrywa się w systemie, lecz możesz zauważyć, że nie pasują one dokładnie do sposobu użycia tego określenia przez C++. W kolejnych częściach tego rozdziału, przyjrzymy się obiektom jako hermetycznym tworom zawierającymi kod i dane potrzebne do wykonania konkretnego zadania związanego z bezpieczeństwem. Innymi słowy, każdy obiekt zabezpieczeń jest zamkniętym

Uwaga

Windows 2000 obsługuje o wiele więcej wywołań funkcji zabezpieczeń interfejsu Windows API niż Windows 95/98, ponieważ jego mechanizmy zabezpieczeń są lepsze i bardziej wydajne. Prawdopodobnie przekonasz się, że zarządzanie przez ciebie zabezpieczeniami podczas korzystania z Windows 95/98 jest poważnie ograniczone, ze względu na brak obsługi mechanizmów zabezpieczeń systemu Windows 2000. Na przykład, nie możesz używać funkcji `GetUserObjectSecurity()` **pod Windows 95/98. Większość wywołań żetonów dostępu, którym przyjrzymy się w następnej części rozdziału, również nie będzie działać. Najlepszym sposobem, by przekonać się, czy dane wywołanie jest obsługiwane, jest próba jego użycia. Otrzymanie błędu *ERROR_CALL_NOT_IMPLEMENTED (value 120)*, będzie sygnałem, że możesz z niego korzystać jedynie pod Windows 2000.**

elementem zaprojektowanym do spełniania określonej roli. (W wielu miejscach w Windows 95/98 i Windows 2000, Microsoft wybrał używanie pełnej wersji obiektu C++, ponieważ wdraża on potrzebny zestaw funkcji jako część MFC. Jednakże, podczas lektury dokumentacji Microsoftu lub tego rozdziału nie powinieneś polegać na tym, że definicja obiektu oznacza ścisły obiekt C++ — myśl o obiektach bardziej w kategoriach COM tego słowa).

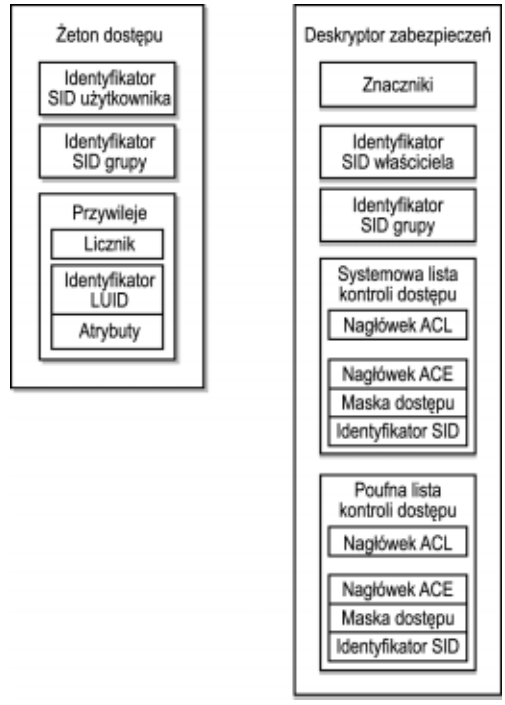
Świadomość, że wszystko jest obiektem, ułatwia zrozumienie pojęcia zabezpieczeń — przynajmniej na początku. Jednakże, same obiekty to tylko początek zagadnienia. Druga strona równania zabezpieczeń to użytkownicy. Użytkownik uzyskuje dostęp do obiektu, więc zabezpieczenia w Windows to kwestia porównywania ochrony obiektów z prawami dostępu użytkownika. Jeżeli użytkownik posiada wystarczające prawa (prawa, które spełniają wymagania obiektu lub je przewyższają), wtedy może on używać tego obiektu. Dokumentacja Windows

określa poziom ochrony obiektu jako *deskryptor zabezpieczeń*. Jest to struktura, która informuje system zabezpieczeń o tym, jakich praw potrzebuje użytkownik, by uzyskać dostęp do obiektu. Podobnie, użytkownik posiada *żeton dostępu*, który jest inną strukturą, informującą system zabezpieczeń o tym, jakie prawa posiada użytkownik w konkretnej sytuacji. „Żeton" jest właściwym określeniem, ponieważ użytkownik przekazuje go systemowi Windows 2000 w zamian za dostęp do obiektu. (Potraktuj obiekt jako autobus, a system Windows 2000 jako jego kierowcę i użytkownika, który przedstawia wymagany żeton do kontroli). Rysunek 3.1 pokazuje obydwie te struktury.

To jest najkrótszy przegląd zabezpieczeń zarówno pod Windows 95/98, jak i Windows 2000. Świadomość tego, że istnieją obiekty zabezpieczeń i żetony użytkownika, ułatwi ci zrozumienie funkcji zabezpieczeń interfejsu Windows API. Kolejne podrozdziały pozwolą przyjrzeć się bardziej szczegółowo, czym dokładnie jest żeton i jak działa. Przyjrzymy się również deskryptorowi zabezpieczeń. Nie musisz znać tych informacji, by wdrożyć system zabezpieczeń, korzystając z technologii ActiveX, jeżeli twoim jedynym celem jest Internet, lecz ta wiedza może pomóc zaprojektować kontrolki ActiveX bardziej ogólnej natury i o większych możliwościach.

Żetony dostępu

Istnieją dwa sposoby patrzenia na prawa użytkownika pod Windows; obydwa są związane z obiektami w takiej czy innej formie. Żeton dostępu użytkownika posiada identyfikator zabezpieczeń (SID) służący do jego identyfikacji w sieci — coś jak posiadanie numeru konta. Żeton



Rysunek 3.1. *Żetony dostępu określają prawa użytkowników, podczas gdy deskryptory zabezpieczeń definiują poziom zabezpieczeń procesu*

użytkownika, który jest identyfikowany przez SID, mówi o tym, do jakich grup należy użytkownik i jakie uprawnienia posiada. Każda grupa posiada również identyfikator SID, więc SID użytkownika zawiera odwołania do identyfikatorów SID grup, do których należy użytkownik, a nie do pełnego zestawu praw dostępu grupy. Zwykle używałbyś narzędzia *Użytkownicy i hasła* w Windows 2000, by zmienić zawartość tego żetonu dostępu.

Przypadek używania zabezpieczeń opartych na obiektach

Istnieją właściwie dwa rodzaje zabezpieczeń, które możesz wdrożyć, pracując z COM+: schemat zabezpieczeń oparty na obiektach (tradycyjny) oraz opartych na rolach (charakterystyczny dla Windows 2000). Wybór między tymi dwiema technologiami wywrze znaczący wpływ na sposób tworzenia projektu własnego komponentu — przynajmniej z perspektywy zabezpieczeń. Zabezpieczenia oparte na rolach pozwalają zdefiniować zasady dostępu w dół do poziomu metody na podstawie ról, jakie wypełnia użytkownik w obrębie firmy. Zwykle wystarczy określenie zabezpieczeń do poziomu metody. Może się jednak zdarzyć, że będziesz musiał zabezpieczyć komponent w sposób niemożliwy do zrealizowania przy wykorzystaniu mechanizmu zabezpieczeń opartych na rolach.

Podczas pracy z zabezpieczeniami obiektów, w czasie projektowania musisz zadecydować o tym, w jaki sposób twój komponent będzie używany, aby dodać konieczne funkcje bezpieczeństwa. Wielu programistów obecnie myśli jedynie o Internecie, jednak firmy mają znacznie szerszy zasięg użytkowania komponentów, niż tylko zastosowanie ich w Internecie. Nie ma takiej reguły, która zabraniałaby tworzenia komponentów projektowanych specjalnie do wykorzystania jako fragmentu aplikacji. W rozdziale 7. przekonamy się, że komponenty będące częścią n-warstwowej aplikacji mogą być używane w sieci LAN lub WAN do uzyskiwania dostępu do bazy danych. Komponenty mogą być odpowiedzią na aplikacje zwykle używane przez administratorów sieci (lub inne wykwalifikowane osoby). *Konsola zarządzania firmy Microsoft* (MMC, *Microsoft Management Console*) może obecnie przysparzać wielu problemów, ponieważ administratorzy nie przywykli do korzystania z niej. Jednak komponentowa natura przystawek, których używa konsola MMC, sprawi, że korzystanie z narzędzi administracyjnych stanie się bezpieczniejsze i łatwiejsze. Faktycznie, prawie każdy komponent może przyjmować wiele tożsamości. Mógłbyś nawet dodać zdolność do wykrywania bieżącego położenia kontrolki lub dodać specjalne pole lokalizacji jako część konfiguracji arkusza właściwości. Ktoś mógłby wybrać podzestaw funkcji do użycia w Internecie, inny zestaw funkcji do wykorzystania w sieci LAN i jeszcze inny — do użytku lokalnego.

Do teraz wielu z was pyta, jakiego rodzaju narzędzi potrzebuje administrator, by uzyskać dostęp do mechanizmów zabezpieczeń, lecz nie korzystając w tym celu z programów dostarczonych wraz z sieciowym systemem operacyjnym (NOS). Windows 2000 właściwie udostępnia wiele łatwych w użyciu narzędzi, więc dodawanie nowych funkcji, w opinii niektórych programistów nie jest warte zachodu — przynajmniej nie wtedy, gdy komponent jest używany lokalnie. W podrozdziale zatytułowanym „Zabezpieczenia oparte na rolach" przekonamy się, że nie zawsze konieczne jest wbudowywanie specjalnych mechanizmów zabezpieczających w twój komponent, nawet jeżeli istnieją pewne szczególne potrzeby dotyczące ochrony, lecz zdarza się, że mechanizm zabezpieczeń opartych na rolach będzie niewystarczający. Mowa o mechanizmie zabezpieczeń opartych na rolach, ponieważ ta nowa funkcja systemu Windows 2000 znacząco zmniejszyła ilość kodu pisanego ręcznie potrzebnego do

wdrożenia ochrony, lecz nie wyeliminowała całkowicie tej konieczności.

Przyjrzyjmy się sytuacji, w której potrzebowałbyś implementacji pewnego rodzaju wewnętrznych zabezpieczeń w swoim komponencie. Zastanów się, co by się stało, gdyby osobą, która zarządza aplikacją, nie był administrator sieci — ktoś, kto jest przeszkolony do samodzielnej pracy z sieciowym systemem operacyjnym. Powiedzmy, że tą osobą jest kierownik zespołu lub inna osoba, która nie musi mieć pełnego wglądu w sprawy sieci, ale potrzebuje wystarczających informacji, by móc skutecznie zarządzać powierzoną aplikacją. Znajdziesz się w takiej sytuacji o wiele razy częściej, niż mógłbyś sądzić. Duże firmy mające wiele małych zespołów pracowników bardzo często zaliczają się do tej kategorii. Administrator sieci nie musi mieć wiedzy potrzebnej do prawidłowego zarządzania aplikacją, ale również nie chce, by kierownik zespołu miał wgląd do sieci. Zabezpieczenia oparte na rolach mogą być odpowiedzią na tę potrzebę, ale wcale nie musi tak być. Problemem jest to, że tak naprawdę nie możesz zdefiniować pojedynczej roli, która zaspokoiłaby potrzeby wszystkich zespołów w firmie — przynajmniej nie możesz tego zrobić ze stuprocentową pewnością. Potrzeba implementacji własnych wewnętrznych metod ochrony lub używania mechanizmu zabezpieczeń opartych na rolach zależy od tego, jak dobrze potrafisz zdefiniować rolę dla osoby używającej komponentu. Czasami dodanie oddzielnych zabezpieczeń jest zasadniczą częścią procesu tworzenia komponentu.

Więc, o co chodzi w części żetonu dostępu dotyczącej przywilejów? Zaczyna się od liczby przywilejów, które posiada użytkownik — nie grup, do których on należy, ale liczby wpisów o specjalnych przywilejach w żetonie dostępu. Ta część również zawiera tablicę wpisów dotyczących przywilejów. Każdy wpis dotyczący przywileju zawiera *lokalnie unikalny identyfikator* (LUID) — zasadniczo wskaźnik do obiektu — i maskę atrybutów. Maska ta informuje o prawach, jakie użytkownik ma do obiektu. Grupowe wpisy SID są zasadniczo takie same. Zawierają liczbę przywilejów i tablicę wpisów ich dotyczących.

Wskazówka

Warto przejrzeć plik pomocy interfejsu Windows API udostępniony wraz z twoim egzemplarzem pakietu Visual C++, by dowiedzieć się, jakie funkcje API związane z identyfikatorami SID i żetonami tam znajdziesz. Przykłady funkcji związanych z SID obejmują: `CopySID()` **i** `AllocateAndInitializeSID()`**. Dowiesz się również, że funkcje** `OpenProcessToken()` **i** `GetTokenInformation()` **mają zasadnicze znaczenie dla prawidłowej pracy zabezpieczeń z każdym językiem, którego używasz.**

Korzystanie z żetonów dostępu

W zagadnienie zabezpieczeń doskonale wprowadzają wywołania żetonów, które udostępnia interfejs Windows API. By zrobić cokolwiek z kontem użytkownika — nawet jeżeli chcesz się dowiedzieć, kto posiada dostęp do konkretnej stacji roboczej — powinieneś wiedzieć o żetonach. Żetony są centralną częścią równania zabezpieczeń po stronie użytkownika. Prawie zawsze będziesz rozpoczynał dostęp do konta użytkownika, wywołując funkcję `OpenProcessToken()`. Przyjrzyj się nazwie tej funkcji — radzi sobie z każdym rodzajem procesu. Jedynym celem tej funkcji

jest pobranie uchwytu żetonu wraz z określonymi prawami przypisanymi do niego. Na przykład, jeżeli chcesz sprawdzić konto użytkownika, musisz mieć przywilej *TOKEN_QUERY*. (Twój żeton dostępu musi zawierać prawa, których żądasz od systemu — dlatego administrator może uzyskać dostęp do żetonu, a pozostali użytkownicy — nie). Wszystkie zmiany dotyczące kont użytkowników wymagają posiadania przywileju *TOKEN_ADJUST_PRIVILEGES*. Istnieje spora liczba tych praw dostępu, więc nie będziemy omawiać ich wszystkich.

Kiedy masz już uchwyt do żetonu dostępu, musisz zdecydować, co z nim zrobić. Jeżeli zdecydowałeś, że chcesz zmienić poziom uprzywilejowania użytkownika w celu wykonania jakiegoś zadania, potrzebujesz identyfikatora LUID tego przywileju, który chcesz zmienić. Wszystkie pojawiają się w pliku *WINNT.H* z przedrostkiem SE_. Na przykład, przywilej *SE_SYSTEM_PROFILE_NAME* pozwala użytkownikowi zebrać informacje profilujące odnośnie do całego systemu. Niektóre z wartości SE nie są w najmniejszym stopniu związane z użytkownikami (np. *SE_LOCK_MEMORY_NAME* pozwala procesowi zablokować dostęp do pamięci systemu). Uzyskujesz identyfikator LUID dla przywileju, używając funkcji `LookupPrivilegeValue()`. Teraz możesz połączyć informacje uzyskane do tej pory, by dokonać zmiany przywileju. Ogólnie, użyjesz funkcji `AdjustTokenPrivileges()`, by przeprowadzić potrzebną zmianę.

Obserwacja przepływu praw dostępu

Trzeba wiedzieć, że uprawnienia do obiektu spływają w dół do możliwie najniższego węzła, chyba że inny identyfikator `SID` to zmienia. Na przykład, jeżeli dasz użytkownikowi prawo do odczytu i zapisu w katalogu *\Temp* na twardym dysku, te uprawnienia będą również dotyczyły katalogu *\Temp\Stuff* (chyba że przyznałeś użytkownikowi konkretne uprawnienia do tego podkatalogu).

To samo pojęcie przepływu praw dostępu zachowuje prawdziwość również w odniesieniu do kontenerów. Przyznanie użytkownikowi praw dostępu do obiektu kontenera, takiego jak dokument programu Word, daje mu prawo do przeglądania wszystkiego, co znajduje się wewnątrz tego kontenera, nawet innych plików, które pojawiają się w jego wnętrzu (w większości przypadków). Jak widzisz, ważne jest śledzenie dokładnych praw użytkownika do różnych obiektów na twoim serwerze poprzez wykonywanie przeglądów zabezpieczeń, ponieważ mógłbyś przypadkowo przyznać użytkownikowi większe uprawnienia, niż potrzebuje on do wykonania określonego zadania.

Sprawdzenie konta użytkownika (lub inny dostęp do informacji żetonu) jest zadaniem całkiem prostym. By odebrać potrzebne informacje, używasz funkcji `GetTokenInformation()`. Ta funkcja wymaga klasy żetonu jako parametru informującego system Windows, jakiego rodzaju informacji potrzebujesz. Na przykład, użyłbyś klasy *TokenUser*, gdybyś chciał zdobyć informacje o konkretnym użytkowniku. Musisz przekazać funkcji właściwą strukturę danych, której system Windows może użyć do przechowywania zwracanych przez tą funkcję informacji, których potrzebujesz — różnią się one, w zależności od klasy żetonu, którego żądasz.

Deskryptory zabezpieczeń

Przyjrzyjmy się deskryptorowi zabezpieczeń. Składa się on z pięciu głównych części. Pierwsza z nich to lista znaczników, które informują o numerze modyfikacji deskryptora, jego formacie i stanie ACL (listy kontroli dostępu).

Następne dwie części zawierają identyfikatory SID. Identyfikatory SID właściciela informują o tym, kto jest właścicielem obiektu. To nie musi być konkretny użytkownik; Windows pozwala używać również grupowych identyfikatorów SID. Jedynym czynnikiem ograniczającym jest to, że identyfikator grupowy SID musi pojawić się w żetonie dostępu osoby zmieniającej wpis. Grupowy identyfikator SID umożliwia grupie ludzi posiadanie obiektu na własność. Z dwóch identyfikatorów SID, jedynie SID właściciela jest ważny w Windows. Grupowy SID jest używany jako część środowiska zabezpieczeń komputerów Macintosh i POSIX.

Dwie ostatnie części zawierają listy ACL. *Systemowa lista kontroli dostępu* (SACL) steruje mechanizmem inspekcji systemu Windows. Za każdym razem, gdy użytkownik lub grupa uzyskuje dostęp do obiektu, a inspekcja dla tego obiektu jest włączona, Windows dokonuje wpisu w dzienniku zdarzeń. *Poufna lista kontroli dostępu* (DACL) kontroluje dostęp do obiektu (kto ma prawo z niego korzystać). Możesz do określonego obiektu przypisać zarówno pojedynczych użytkowników, jak i całe grupy.

Uwaga

Istnieją właściwie dwa rodzaje deskryptorów zabezpieczeń: bezwzględny i względny. Bezwzględny deskryptor zabezpieczeń zawiera właściwą kopię każdego ACL znajdującego się w jego strukturze. Ten rodzaj deskryptora zabezpieczeń jest przeznaczony do użycia z obiektami, które wymagają specjalnej obsługi. Względne deskryptory zabezpieczeń zawierają jedynie wskaźnik do SACL i DACL. Ten rodzaj deskryptora oszczędza pamięć i zmniejsza czas potrzebny do zmiany praw grupy obiektów. Użyłbyś go w sytuacji, gdy w konkretnej grupie wszystkie obiekty wymagałyby tego samego poziomu zabezpieczeń. Na przykład, możesz użyć tej metody do zabezpieczenia wszystkich wątków w pojedynczej aplikacji. System Windows wymaga zamiany względnych deskryptorów zabezpieczeń na bezwzględne, zanim będziesz mógł je zapisać lub przenieść do innego procesu. Każdy deskryptor, który pobierasz, używając funkcji API, jest typu względnego — musisz przekonwertować, zanim będziesz mógł go zapisać. Konwersji możesz dokonać za pomocą dwóch funkcji API: `MakeAbsoluteSD` **i** `MakeSelfRelativeSD`**.**

ACL składa się z dwóch rodzajów wpisów. Pierwszym z nich jest nagłówek, który zawiera liczbę rekordów kontroli dostępu (ACE) zawartych w ACL. Windows korzysta z tej liczby jako metody określania, kiedy osiągany jest koniec listy ACE, ponieważ nie istnieje żaden rekord struktury kończącej lub jakikolwiek sposób dokładnego określenia rozmiaru każdego ACE w strukturze. Drugim wpisem jest tablica wpisów ACE.

Ostrzeżenie

Nigdy nie dokonuj bezpośrednich zmian zawartości ACL lub SID, ponieważ Microsoft może zmienić ich strukturę w przyszłych wersjach systemu Windows. Interfejs Windows API oferuje bogactwo funkcji umożliwiających zmiany w tych strukturach. Zawsze używaj funkcji API do wykonywania dowolnego zadania dotyczącego każdego rodzaju struktury, by zmniejszyć liczbę zmian w strukturze twojej aplikacji.

Więc, co to jest ACE? ACE określa prawa do obiektu dla pojedynczego użytkownika lub grupy. Każdy ACE posiada nagłówek określający rodzaj, rozmiar i znaczniki dla ACE. Dalej mamy maskę dostępu, która określa prawa posiadane przez użytkownika lub grupę do danego obiektu. Na końcu znajduje się wpis przeznaczony na identyfikator SID użytkownika lub grupy.

Istnieją cztery różne rodzaje nagłówków ACE (trzy z nich są używane w bieżącej wersji systemu Windows). Rodzaj *access-allowed* pojawia się w DACL i przyznaje prawa użytkownikowi. Możesz użyć go, by dodać do praw, które użytkownik już posiada do danego obiektu, kolejne uprawnienie według wzorca jeden po drugim. Powiedzmy, że nie chcesz, aby użytkownik zmieniał czas systemowy, żeby można było zachować synchronizację czasu na wszystkich maszynach znajdujących się w sieci. Jednakże, może zdarzyć się sytuacja — taka jak zmiana czasu — kiedy użytkownik potrzebowałby tego prawa. Możesz użyć *access-allowed* ACE, by w tej konkretnej sytuacji przyznać użytkownikowi to prawo. *Access-denied* ACE odbiera uprawnienia, które użytkownik posiada do obiektu. Możesz go użyć do odmowy dostępu do obiektu podczas szczególnych zdarzeń w systemie. Na przykład, możesz odebrać prawa dostępu do zdalnego terminala podczas przeprowadzania jego modernizacji. Trzeci rodzaj ACE — *system audit* współpracuje z SACL. Określa, jakie zdarzenia należy kontrolować w przypadku konkretnego użytkownika lub grupy. Ostatni rodzaj ACE *currently-unused* to ACE alarmu systemowego. Pozwala albo SACL albo DACL wszcząć alarm, gdy pojawi się określone zdarzenie.

Wskazówka

Warto przejrzeć plik pomocy interfejsu Windows API, by dowiedzieć się, jakie prawa dostępu udostępnia system Windows. Powinieneś również przyjrzeć się różnym strukturom używanym do uzyskiwania tych informacji. Szczególnie ważne są struktury ACL i ACE. Poszukaj znaczników ACE określających sposób reakcji obiektów zawartych w kontenerze. Na przykład, przyjrzyj się stałej *CONTAINER_INHERIT_ACE*, która umożliwia podkatalogom dziedziczenie ustawień zabezpieczeń od katalogu nadrzędnego.

Używanie deskryptorów zabezpieczeń

Zrozumienie, czym jest deskryptor zabezpieczeń i jak współdziałają różne struktury, które on zawiera, nie wystarczy. Powinieneś również wiedzieć, jak rozpocząć proces właściwego uzyskiwania dostępu i korzystania z deskryptorów zabezpieczeń, by móc napisać program. Pierwszą sprawą, którą powinieneś zrozumieć, jest to, że w odróżnieniu od żetonów, deskryptory zabez-

pieczeń nie są uogólnione. Nie możesz użyć standardowego zestawu funkcji, by uzyskać do nich dostęp. Faktycznie, jest pięć klas deskryptorów zabezpieczeń, z których każda używa innego zestawu wywołań deskryptora do uzyskania początkowego dostępu do obiektu. (Musisz posiadać przywilej *SE_SECURITY_NAME*, by móc używać którejś z tych funkcji).

- *Pliki, katalogi, potoki, złącza poczty* — użyj funkcji `GetFileSecurity` oraz `SetFileSecurity`, by uzyskać dostęp do obiektu tego rodzaju.

Uwaga

Jedynie system plików NTFS pod Windows 2000 zapewnia ochronę. System plików VFAT udostępnia je w mniejszym stopniu pod Windows 95/98. Nie możesz przydzielić lub uzyskać deskryptorów zabezpieczeń dla systemów plików, takich jak FAT czy HPFS pod odpowiednimi systemami operacyjnymi. System plików FAT nie udostępnia żadnej rozszerzonej przestrzeni atrybutów, której obecność jest warunkiem koniecznym do implementacji mechanizmów zabezpieczeń.
System plików HPFS udostępnia rozszerzoną przestrzeń atrybutów, lecz nie obejmują one żadnych funkcji zabezpieczających. Więc ze wszystkich opisanych systemów plików, NTFS jest najbardziej bezpieczny. Jednakże, nigdy nie zakładaj, że system plików jest całkowicie bezpieczny. Istnieją programy narzędziowe (dostępne w Internecie), które umożliwiają odczyt zawartości plików znajdujących się na partycji NTFS, nawet wtedy, gdy użytkownik nie jest zalogowany do systemu Windows 2000.

- *Procesy, wątki, żetony dostępu i obiekty synchronizacji* — potrzebujesz funkcji `GetKernelObjectSecurity` i `SetKernelObjectSecurity`, by uzyskać dostęp do tych obiektów. Wszystkie te obiekty, nawet żetony dostępu, są właściwie obiektami jądra systemu i jako takie, w celach ochronnych, posiadają również swój własny deskryptor zabezpieczeń.
- *Terminale, pulpity, okna i menu* — funkcje `GetUserObjectSecurity` i `SetUserObjectSecurity` umożliwiają dostęp do tych obiektów. *Terminal* to kombinacja klawiatury, myszy i monitora — sprzętu używanego do uzyskiwania dostępu do systemu. *Pulpit* zawiera *okna* i *menu* — wyświetla elementy, które widać na ekranie. Te cztery obiekty dziedziczą uprawnienia od siebie w pokazany sposób. Innymi słowy, pulpit odziedziczy prawa terminala.
- *Klucze rejestru systemowego* — ten rodzaj obiektów wymaga użycia `RegGetKeySecurity` i `RegSetKeySecurity`. Zauważ, że nazwy tych dwóch funkcji zaczynają się od Reg, tak jak inne funkcje charakterystyczne dla rejestru, które obsługuje system Windows.
- *Obiekty usługi wykonawczej* — funkcje `QueryServiceObjectSecurity` oraz `SetServiceObjectSecurity` pracują z tym obiektem. Z pewnych powodów, żadna z tych funkcji nie pojawia się wśród pozostałych wywołań funkcji zabezpieczeń w pliku pomocy Windows API. Musisz więc znać nazwy tych funkcji by odszukać

je w pliku pomocy. Usługa wykonawcza jest zadaniem działającym w tle udostępnianym przez system Windows — takim samym, jak na przykład funkcja nadzorowania zasilacza UPS. Listę usług obsługiwanych przez twój system znajdziesz, klikając dwukrotnie ikonę apletu *Usługi* w *Panelu sterowania* (Windows NT) lub *Narzędzia administracyjne* i dalej *Usługi* (Windows 2000).

Po uzyskaniu dostępu do obiektu, możesz wykonać różnorodne zadania, korzystając ze standardowego zestawu funkcji API. Na przykład, funkcja `GetSecurityDescriptorDACL` pobiera kopię DACL z dowolnego rodzaju deskryptora. Innymi słowy, deskryptory dla tych wszystkich obiektów mają prawie taki sam format — nawet jeżeli rozmiar większości komponentów będzie się różnił (z powodu tego, że każdy obiekt zawiera inną liczbę list ACE). Identyfikatory SID także mają różne rozmiary.

W celu sprawdzenia lub zmodyfikowania zawartości deskryptora zabezpieczeń, należy następnie deassemblować komponenty. Na przykład, możesz zobaczyć konkretne ACE w DACL lub SACL, korzystając z funkcji API — `GetACE`. Można również użyć identyfikatorów SID właściciela i grupy do wywołania funkcji związanych z tym identyfikatorem (omówiliśmy te funkcje w części tego rozdziału dotyczącej żetonów dostępu). Możesz użyć standardowego zestawu funkcji, by mieć możliwość manipulacji deskryptorem zabezpieczeń po zdobyciu konkretnej procedury działania. Aby uzyskać dostęp do deskryptora zabezpieczeń:

1. Pobierz deskryptor
2. Usuń konkretny komponent
3. Zmodyfikuj zawartość tego komponentu

W celu zmiany deskryptora zabezpieczeń, możesz odwrócić ten proces. Innymi słowy, używasz funkcji takiej jak `AddACE`, by dodać nowy ACE do ACL, następnie korzystasz z `SetSecurityDescriptorSACL`, by dokonać zmiany SACL w deskryptorze i na koniec zapisać sam deskryptor, używając funkcji, takiej jak `SetFileSecurity` (zakładając, że chcesz zmodyfikować obiekt typu plik).

Wzmacnianie zabezpieczeń w Windows

Jeśli zacząłeś zastanawiać się, jak system Windows ocenia wpisy ACE w DACL, prawdopodobnie odkryjesz kilka obszarów, gdzie mogą potencjalnie wystąpić problemy, z którymi narzędzia Windows radzą sobie automatycznie, ale ty będziesz musiał to przewidzieć w swojej aplikacji, by uzyskać ten sam efekt. (SACL ma ten sam problem, lecz dotyczy on jedynie mechanizmu inspekcji, więc efekt jest mniej dotkliwy od strony systemu zabezpieczeń).

System Windows ocenia wpisy ACE w DACL w kolejności ich pojawiania się. Na początku wygląda to niewinnie. Jednak, w pewnych sytuacjach może stać się to problemem. Na przykład, jeżeli chcesz odebrać wszystkie prawa użytkownika w jednym obszarze, lecz jego lista wpisów ACE zawiera członkostwo w grupie, które umożliwia mu dostęp do tego obszaru. Jeżeli najpierw na liście umieściłbyś wpis *access-allowed* ACE, użytkownik uzyskałby dostęp do tego obszaru — Windows kończy przeszukiwanie listy natychmiast po znalezieniu pierwszego wpisu ACE, który przyznaje wszystkie potrzebne prawa użytkownika (lub ACE, który odmawia jednego prawa). Przyznane prawa są kumulacyjne. Jeżeli jeden wpis ACE przyznaje prawo do odczytu

pliku, a inny — do jego zapisu, a użytkownik prosi o te dwa prawa (do zapisu i odczytu pliku), Windows będzie widział obydwa wpisy ACE jako przyznające żądane uprawnienia.

Wskazówka

System Windows przestanie czytać wpisy ACE po spełnianiu żądania użytkownika odnośnie do dostępu. Wszystkie swoje wpisy *access-denied* ACE zawsze umieszczaj na początku listy, by uniknąć powstania jakiejkolwiek możliwości naruszenia zabezpieczeń.

Musisz również dbać o porządkowanie grupowych identyfikatorów SID. Prawa, które nabywa użytkownik z tytułu przynależności do różnych grup, również są kumulacyjne. Oznacza to, że użytkownik należący do dwóch grup, z których jedna posiada dostęp do danego pliku, druga — takiego prawa nie ma, będzie miał dostęp do tego pliku — pod warunkiem że grupa mająca prawo dostępu znajduje się wcześniej na liście.

Oczywiście, możesz próbować znaleźć najlepszy układ grup. Wraz ze wzrostem liczby grup i poszczególnych praw, które posiadają konkretni użytkownicy, możliwość niezamierzonego naruszenia zabezpieczeń również wzrasta. To właśnie dlatego należy ostrożnie tworzyć nowe grupy i ograniczać ich użytkownikom prawa dostępu.

Inne kwestie dotyczące zabezpieczeń

Przyglądając się zabezpieczeniom w systemach Windows 95/98 lub Windows 2000, trzeba jeszcze zwrócić uwagę na dwie kwestie: ochronę danych i serwera. Pierwsza z nich dotyczy zdolności klienta do uzyskiwania dostępu do danych, do których nie powinien go mieć podczas uzyskiwania dojścia do danych poprzez serwer (nie mówię tu o serwerze plików, lecz o pewnym rodzaju mechanizmu DDE lub innym serwerze aplika cji). Pomyśl o tym w ten sposób: co się dzieje, jeżeli klient nie miał praw do określonego rodzaju danych, lecz uzyskał do nich dostęp poprzez wywołanie DDE do serwera, który miał wymagane uprawnienia? Jak można zabezpieczyć sam serwer, aby nieświadomie nie naruszał zabezpieczeń?

Windows udostępnia kilka funkcji API, które umożliwiają przedstawienie klienta serwerowi. W zasadzie, funkcje te umożliwiają serwerowi przyjęcie ograniczeń zabezpieczeń klienta w celu określenia, czy klient posiada wystarczające prawa, by mieć dostęp do danych lub procesu. Załóżmy, że użytkownik programu MS Word potrzebuje dostępu do pliku danych programu Excel. Użytkownik może uzyskać dostęp do tego pliku za pośrednictwem mechanizmu DDE. W tej sytuacji, serwer musiałby zweryfikować poziom uprawnień użytkownika programu Word (czy są one wystarczające, by uzyskać dostęp do tego pliku), zanim wyśle potrzebne dane. Serwer może nawet „zauważyć", że klient posiada wyższe uprawnienia, gdy korzysta z tej techniki. Ostatecznym kryterium jest to, że najważniejszym zadaniem serwera jest ochrona danych, zasobów i środowiska, którymi zarządza.

Ten zestaw funkcji API obsługuje trzy różne rodzaje komunikacji: DDE, nazwane potoki oraz RPC. Potrzebujesz różnych funkcji API dla każdego rodzaju komunikacji. Na przykład, by przedstawić klienta DDE, użyłbyś funkcji `DDEImpersonateClient`. Istnieją pewne ograniczenia odnośnie do poziomu przedstawiania udostępnionego obecnie przez system Windows. Na przykład,

nie są obsługiwane połączenia TCP/IP, więc będziesz zmuszony skorzystać z innych metod weryfikacji tego, czy użytkownik posiada odpowiedni poziom praw dostępu.

Innym zagadnieniem związanym z ochroną jest zabezpieczenie samego serwera. Co powstrzyma użytkownika, który wywołuje program Excel z poziomu programu Word przed zrobieniem czegoś z Excelem, co uszkodzi sam serwer? Zapewnienie, że kwestie zabezpieczeń są realizowane, nie jest trudne w przypadku plików oraz innych rodzajów nazwanych struktur, ponieważ serwer automatycznie przydziela tym obiektom deskryptor zabezpieczeń. (Serwer DDE, taki jak Excel, nie musiałby nic robić w tej sytuacji, ponieważ plik znajduje się pod kontrolą serwera plików). Jednakże, wiele prywatnych obiektów DDE lub serwerów aplikacji nie jest nazwanych i wymaga specjalnej ochrony. Windows udostępnia funkcje API, by wspomóc ochronę samego serwera. Na przykład, funkcja `CreatePrivateObjectSecurity` pozwala serwerowi przydzielić deskryptor zabezpieczeń każdemu ze swoich prywatnych obiektów — powiedzmy, wątkowi lub innemu procesowi. Deskryptor zabezpieczeń ma za zadanie powstrzymanie wszystkich innych (poza serwerem, oczywiście) przed uzyskaniem dostępu do obiektów prywatnych.

Zabezpieczenia oparte na rolach

Jak dotąd, omawialiśmy tradycyjne zabezpieczenia obiektów udostępnianym przez Windows 2000. Jeżeli tworzysz standardowe aplikacje lub komponenty, które będą przeznaczone do wykorzystania w Windows NT/2000, ochrona obiektów byłaby wystarczająca. Nowością podczas pracy z COM+ pod Windows 2000 są zabezpieczenia oparte na rolach.

Przyjrzyjmy się dokładniej temu nowemu mechanizmowi w systemie Windows 2000. Kolejne części tego rozdziału pomogą zrozumieć, jak zabezpieczenia oparte na rolach dopasowują się do scenariusza programowania komponentu COM+. Warto znać korzyści stosowania mechanizmu zabezpieczeń opartego na rolach. Zabezpieczenia oparte na rolach mogą być używane razem z uwierzytelnianiem lub zamiast niego. Po poznaniu tego, czego dotyczą wymogi korzystania z zabezpieczeń opartych na rolach, przyjrzymy się głównemu interfejsowi — *ISecurityCallContext*.

Zalety zabezpieczeń opartych na rolach

Podczas gdy mechanizm zabezpieczeń obiektowych nieźle się sprawdza, w wielu sytuacjach ma również surowe ograniczenia, które zmuszają programistę albo do zignorowania kwestii zabezpieczeń, albo do ręcznego napisania ogromnej ilości kodu. Najważniejszym ograniczeniem tego mechanizmu zabezpieczeń jest rozdrobnienie. Ustalasz zabezpieczenia dla całego obiektu, uprawnienia użytkownika do tego obiektu są oparte na indywidualnym lub grupowym żetonie dostępu. Mechanizm zabezpieczeń opartych na rolach w pewnym stopniu radzi sobie z problemem rozdrobnienia poprzez umożliwienie ci ustawiania zabezpieczeń na poziomie metody. Dodatkowo, oprócz większej elastyczności w ustalaniu poziomu dostępu użytkownika do komponentu, zabezpieczenia oparte na rolach dają ci następujące korzyści:

- *Konfiguracja* — zwykle będziesz używał narzędzi administracyjnych przystawki konsoli MMC o nazwie *Usługi składowe* lub skryptów, by konfigurować komponent do korzystania z zabezpieczeń opartych na rolach. Jednakże możesz również ustalić

pewne rodzaje zabezpieczeń jako część procesu inicjalizacji twojego komponentu. (Zwykle, nie dodasz żadnego kodu do komponentu, by pozwolić administratorowi na pełną elastyczność tworzenia zasad bezpieczeństwa, które spełniają wymagania firmy).

- *Żadnego pisania dodatkowego kodu* — COM+ automatycznie rozwiązuje szczegóły związane z bezpieczeństwem, jeżeli potrafisz sobie poradzić z zabezpieczeniami na poziomie metody. Będziesz musiał dodatkowo zawrzeć tylko obsługę interfejsu *ISecurityCallContext* w swoim komponencie. Oczywiście, wymóg mniejszej ilości kodu przekłada się na mniejszą specyfikację projektu — nie ma żadnych wymagań odnośnie do zabezpieczeń na poziomach interfejsu lub projektu komponentu.

Wskazówka

COM+ nie uchroni cię przed dodawaniem do komponentu kodu przeznaczonego do obsługi mechanizmu zabezpieczeń opartego na rolach, on tylko umożliwia tworzenie komponentu bez dodawania kodu. Jeżeli dodajesz kod do swojego komponentu, by wymusić korzystanie z zabezpieczeń opartych na rolach, komponent używa zaprogramowanej kontroli zabezpieczeń. Z drugiej strony, jeżeli dasz administratorowi pełną kontrolę nad zabezpieczeniami i nie dodasz żadnego kodu, komponent użyje zabezpieczeń deklaracyjnych.

- *Łatwiejsze do zrozumienia* — opieranie zabezpieczeń na rolach pozwala administratorowi lepiej wykonywać pracę związaną z konfiguracją zabezpieczeń komponentu. Wciąż możesz dodawać grupy lub pojedynczych użytkowników do roli, różnica tkwi w sposobie postrzegania. Łatwiej jest przyznać użytkownikowi lub grupie dostęp do pojedynczych metod zawartych w komponencie przy wykorzystaniu zadań, które ten użytkownik lub grupa mają realizować.
- *Bardziej elastyczne* — w odróżnieniu od niezmiennych funkcji ochrony, mechanizm zabezpieczeń opartych na rolach jest konfigurowany spoza komponentu. Oznacza to, że nie są konieczne zmiany w kodzie, gdy potrzeby firmy ulegają zmianie. Zamiast zmieniać kod komponentu (wraz z usuwaniem błędów i nieodłącznym testowaniem), administrator może dokonać zmiany, której wykonanie będzie trwało jedynie sekundy.
- *Dokładna inspekcja* — w tym rozdziale zdążyliśmy już przyjrzeć się sprawie inspekcji. Z mechanizmu tego korzysta się, aby śledzić poczynania każdego, komu uda się włamać do twojego systemu. Zabezpieczenia oparte na rolach pozwalają dokonywać inspekcji zabezpieczeń na poziomie metody, zamiast na poziomie obiektu. Uzyskasz lepszy obraz tego, co cracker próbował zrobić z twoim komponentem, i będziesz miał większą możliwość reakcji.

Wskazówka

W tym rozdziale przyglądamy się zabezpieczeniom opartym na rolach na poziomie teorii. W wielu sytuacjach administrator może nie mieć pojęcia o sposobie działania tego mechanizmu lub o jego wpływie na bezpieczeństwo komponentów, co oznacza, że być może będziesz musiał udostępnić

instrukcję konfiguracji swojego komponentu. Mechanice pracy z zabezpieczeniami opartymi na rolach przyjrzymy się w rozdziale 7., podczas omawiania instalacji komponentów.

Mechanizm zabezpieczeń opartych na rolach może również zapewnić pewną ochronę, której zabezpieczenia na poziomie obiektu tak naprawdę nie obejmują. Zwykle, listy uwierzytelniające użytkownika są sprawdzane na poziomie aplikacji. Oznacza to, że dostęp użytkownikowi jest przyznawany tylko raz. Powstały w wyniku tego żeton dostępu jest wykorzystywany do uzyskiwania dostępu do pozostałych zasobów. W świecie monolitycznych aplikacji, gdzie pojedynczy serwer udostępnia użytkownikowi wszystko to, czego potrzebuje, ten rodzaj weryfikacji zabezpieczeń nieźle się sprawdza. Jednakże, co się stanie, gdy początkowa aplikacja odwoła się do komponentu na jednym serwerze, a następnie okaże się, że ten komponent musi odwołać się do innego komponentu znajdującego się na drugim serwerze? Dostęp użytkownika do pierwszego serwera jest weryfikowany podczas procesu uruchamiania aplikacji (stanowi jeden z jego etapów), lecz co z dostępem do drugiego serwera? Niestety, może wtedy dojść do naruszenia integralności zabezpieczeń pomiędzy dwoma serwerami — kolejna dziura w zabezpieczeniach, o których była mowa wcześniej.

Windows 2000 oferuje tzw. *granicę zabezpieczeń (security boundary)* — możesz wymusić sprawdzanie zabezpieczeń albo na poziomie procesu (aplikacji), albo na poziomie komponentu. Oznacza to, że jesteś w stanie sprawdzać listy uwierzytelniające użytkownika za każdym razem, gdy ma miejsce odwołanie do kolejnego komponentu. To zapewnia, że nie pojawi się możliwość naruszenia zabezpieczeń między dwoma serwerami. Dodatkowo, oprócz lepszej kontroli zabezpieczeń, dostęp na poziomie komponentu oznacza, że bieżące informacje o zabezpieczeniach są włączane jako część informacji dotyczących kontekstu wysyłanych do twojego komponentu podczas wywołania. W sumie, gdy wymuszasz sprawdzanie zabezpieczeń na poziomie komponentu, może on użyć zabezpieczeń jako jednej z metod określania sposobu odpowiedzi na żądanie użytkownika, czyniąc komponent bardziej elastycznym i lepiej przystosowanym do reakcji na zmienne warunki w sieci. Użycie kontroli zabezpieczeń na poziomie komponentu pociąga za sobą jednak redukcję wydajności aplikacji i oznacza, że użytkownik będzie spędzał więcej czasu w oczekiwaniu na reakcję ze strony aplikacji. W dodatku, rośnie nieznacznie ruch w sieci, co może być problemem w sieciach, i tak już zatłoczonych, żeby zapewnić odpowiednią przepustowość potrzebną do wykonywania niezbędnych dla firmy zadań.

Uwaga

Aplikacje bibliotek COM+ zawsze korzystają z zabezpieczeń na poziomie komponentu. Nie możesz włączyć mechanizmu zabezpieczeń na poziomie procesu dla aplikacji bibliotek, ponieważ zawsze polegają one na rolach — co oznacza weryfikację dostępu na poziomie komponentu.

Uwierzytelnianie i role

Uwierzytelnianie i zabezpieczenia oparte na rolach nie wykluczają się wzajemnie. Możesz tworzyć komponenty oraz instalować je w taki sposób, by można było używać jednego lub obydwu sposobów weryfikacji użytkownika oraz poziomu dostępu do komponentu, jaki posiadają. Trzeba jednak pamiętać, że proces weryfikacji przebiega inaczej w zależności od sposobu konfiguracji

komponentu. Poniższa lista pomoże ci zrozumieć, w jaki sposób uwierzytelnianie i zabezpieczenia oparte na rolach mogą współpracować:

- *Uwierzytelnianie włączone, zabezpieczenia oparte na rolach są używane* — uwierzytelnianie pojawia się na poziomie procesu. Każdy użytkownik, który nie może zostać uwierzytelniony, nie zostanie zweryfikowany na poziomie komponentu. Gdy już znajduje się na poziomie komponentu, mechanizm zabezpieczeń opartych na rolach pozwala uzyskać im dostęp do zerowej lub większej liczby metod komponentu, w zależności od przyznanej danemu użytkownikowi roli. Jeżeli ma prawo korzystać z komponentu, wciąż nie mając przypisanej roli, wtedy cały proces zakończy się niepowodzeniem, a żądanie użytkownika będzie odrzucone. Jest to domyślne ustawienie konfiguracyjne COM+.
- *Uwierzytelnianie włączone, zabezpieczenia oparte na rolach nie są używane* — użytkownik jest uwierzytelniany jedynie na poziomie procesu, na poziomie komponentu — nie. Jeżeli uwierzytelnianie powiedzie się, użytkownik zyskuje dostęp do wszystkich możliwości oferowanych przez komponent. Jest to domyślne ustawienie systemu Windows NT 4.0. To są również ustawienia, jakich będzie używał każdy komponent COM, który przeniesiesz do Windows 2000 do chwili, gdy administrator konkretnie określi ustawienia mechanizmu zabezpieczeń opartego na rolach lub programista doda obsługę tych zabezpieczeń do swojego komponentu.
- *Uwierzytelnianie wyłączone, zabezpieczenia oparte na rolach są używane* — nawet jeżeli Windows 2000 przechodzi przez procedury uwierzytelniania użytkownika, to wynik tego procesu jest zasadniczo pominięty. Jedyny przeprowadzany sposób weryfikacji dostępu użytkownika to zabezpieczenia oparte na rolach. Można użyć tego sposobu, gdy standardowe uwierzytelnianie systemu Windows 2000 nie jest wystarczająco elastyczne, by poradzić sobie z różnymi rolami, które użytkownik może spełniać, lub gdy większość zasobów komponentu jest ogólnie dostępnych. Na przykład, możesz pozwolić każdemu dokonywać przeszukiwania twoich fragmentów bazy danych katalogu, lecz ograniczyć dostęp do funkcji usuwania oraz edycji.
- *Uwierzytelnianie wyłączone, zabezpieczenia oparte na rolach nie są używane* — jest to ustawienie zabezpieczeń, które można określić jako „nie obchodzi mnie ta kwestia". Poprzez wyłączenie obydwu poziomów kontroli zabezpieczeń, umożliwiasz wszystkim dostęp do wszystkich zasobów, które komponent ma do zaoferowania. W większości sytuacji, zobaczysz takie ustawienie w przypadku komponentów przeznaczonych całkowicie do użytku publicznego. Na przykład, możesz użyć tego ustawienia dla komponentów, które obsługują wykonywanie jakichś przyziemnych zadań dla twojej witryny WWW. Oczywiście, taki komponent powinien przejść rygorystyczne testy pod kątem obecności jakichkolwiek dziur w zabezpieczeniach i dodatkowo musi mieć ograniczony dostęp do zasobów sieciowych.

Interfejs ISecurityCallContext

Interfejs *ISecurityCallContext* umożliwia dostęp do danych o zabezpieczeniach dla konkretnego komponentu w danym kontekście. Dla ciebie jako programisty oznacza to, że dowiadujesz się, w jakiej roli znajduje się użytkownik i jakie są jego prawa. Fakt, że interfejs ten jest przeznaczony dla określonego kontekstu, oznacza, że możesz wyłącznie pracować z żądaniem powiązanym

z tym konkretnym egzemplarzem komponentu. W dodatku, nie będziesz w stanie uzyskać informacji o komponencie jako całości.

Ten interfejs jest zwykle dostępny dla twojego komponentu COM+, jeżeli administrator włączył mechanizm zabezpieczeń opartych na rolach, lecz nie zawsze będziesz zmuszony korzystać z niego. Będzie to potrzebne tylko wtedy, gdy zdecydujesz się na obsługę zabezpieczeń wewnątrz komponentu, zamiast umożliwiać administratorowi obsługę tych zabezpieczeń jako części konfiguracji komponentu. Otrzymasz komunikat o błędzie *E_NOINTERFACE*, jeżeli podejmiesz próbę uzyskania wskaźnika do interfejsu, w momencie gdy mechanizm zabezpieczeń opartych na rolach jest wyłączony, więc można łatwo dowiedzieć się, kiedy możesz korzystać z tego interfejsu wewnątrz komponentu.

Zwykle będziesz używał interfejsu *ISecurityCallContext*, by zdobyć określone rodzaje informacji o bieżącym kontekście komponentu. Wszystkie te informacje są zawarte w kolekcji kontekstu wywołania zabezpieczeń — jest to zasadniczo tablica informacji na temat obecnie uruchomionego egzemplarza twojego komponentu. Poniżej znajduje się lista informacji, które możesz uzyskać z kolekcji kontekstu wywołania zabezpieczeń:

- ✧ Liczba programów żądających
- ✧ Najniższy poziom uwierzytelnienia
- ✧ Obiekty wywołujące
- ✧ Bezpośredni obiekt wywołujący
- ✧ Początkowy obiekt wywołujący

Dodatkowo, oprócz informacji z kolekcji kontekstu wywołania zabezpieczeń, interfejs *ISecurityCallContext* pozwala określić, czy program wywołujący lub użytkownik znajdują się w konkretnej roli. To jest rodzaj informacji, którego używałbyś do uznawania lub odrzucania żądań dostępu do określonych metod komponentu. Możesz również określić, czy zabezpieczenia oparte na rolach są włączone dla tego komponentu (czy są udostępniane przez serwer).

Teraz, gdy znasz już korzyści stosowania interfejsu *ISecurityCallContext*, przyjrzyjmy się dostępnym metodom. Tabela 3.3 udostępnia listę metod, z których będziesz najczęściej korzystał.

Tabela 3.3. *Podsumowanie metod interfejsu ISecurityCallContext*

Metoda	Opis
`get_Count()`	Zwraca liczbę dostępnych właściwości w kolekcji kontekstu wywołania zabezpieczeń.
`Get_Item()`	Pobiera wartość dla określonego elementu danych wewnątrz kolekcji kontekstu wywołania zabezpieczeń. Oczywiście, element danych, który podasz, musi znajdować się w zakresie dostępnych właściwości, więc powinieneś najpierw użyć metody `get_Count()`, by określić liczbę dostępnych właściwości.
`Get_NewEnum()`	Uzyskuje wskaźnik iteracji dla kolekcji kontekstu wywołania zabezpieczeń.
`IsCallerInRole()`	Określa, czy bezpośredni program żądający znajduje się w określonej roli.

	Ta metoda nie wyświetli wszystkich ról, w jakich się znajduje; pozwala jedynie sprawdzić, czy znajduje się on w podanej przez ciebie roli. Możesz używać tej metody do określania, czy program żądający powinien mieć dostęp do konkretnych metod lub zasobów wewnątrz komponentu.
`IsSecurityEnabled()`	Określa, czy mechanizm zabezpieczeń opartych na rolach jest włączony dla tego egzemplarza komponentu. Ta metoda nie określi, czy mechanizm zabezpieczeń jest aktywny dla innych egzemplarzy komponentu. Już wiesz, że zabezpieczenia oparte na rolach są dostępne na serwerze, ponieważ każda próba uzyskania wskaźnika do interfejsu *ISecurityCallContext* nie powiedzie się, jeżeli serwer nie obsługuje zabezpieczeń opartych na rolach
`IsUserInRole()`	Zasadniczo wykonuje to samo zadanie, co metoda `IsCallerInRole()`, lecz dla określonego użytkownika. Różnica pomiędzy programem wywołującym a użytkownikiem jest taka, że ten pierwszy jest tym, który obecnie używa komponentu. Żądanie użytkownika może odnosić się do każdego użytkownika, który ma dostęp do serwera — niekoniecznie tego, który odpowiada za bieżące wywołanie.

Mechanizm wyrównywania obciążenia komponentu (CLB)

Niestety, obecna strategia komponentowa Microsoftu nie potrafi dostosować się do wymogów i potrzeb dużych korporacji lub dużych obciążeń, tak charakterystycznych dla Internetu. Co się stanie, jeżeli aplikacja, którą tworzysz, nie uruchomi się na pojedynczym serwerze, nie zadziała dla każdego w firmie, bez względu na to, jak bardzo wydajną ją uczynisz? Pomyśl, co mogłoby się stać, gdyby twoja elektroniczna witryna handlowa nagle przestała działać z powodu obciążenia wywołanej przez szał świątecznych zakupów. Coś takiego przydarzyło się już niektórym firmom. Krótko mówiąc, komponenty niezbyt dobrze dostosowują się do obecnych wymagań związanych z możliwymi obciążeniami, jak również z dużą ich zmiennością. W tym momencie, powinieneś się martwić nie tylko o dostrojenie samej aplikacji, ale również o sposób jej pracy na wielu serwerach. Okazuje się, że możliwość skalowania jest jedną z najważniejszych przyczyn używania COM+. Być może, że twoja aplikacja będzie musiała działać na wielu serwerach, lecz użycie COM+ pozwala użytkownikowi sądzić, że jego wersja aplikacji jest uruchamiana na tej samej maszynie. Technologia ta to wyrównywanie obciążenia komponentu (CLB) — możliwość przydziału użytkowników do serwera, który najlepiej potrafi obsłużyć żądania ich komponentów.

Uwaga

CLB nie stał się częścią Windows 2000, lecz jest ważną częścią COM+. Microsoft zdecydował się na usunięcie tego mechanizmu ze swojego systemu operacyjnego i udostępnienie go w postaci oddzielnego produktu (CLB wchodzi w skład pakietu Microsoft Application Center 2000). Ta część rozdziału opiera się na wersji mechanizmu CLB, który początkowo firma Microsoft udostępniała wraz z Windows 2000, a która później

została udostępniona jako oddzielny pakiet do pobrania dla beta-testerów. To jest część wyłącznie teoretyczna, która pomoże ci zrozumieć sposób, w jaki CLB jest powiązany z COM+ jako całością. Nie powinieneś używać tych informacji do pisania aplikacji — upewnij się, że posiadasz obecnie wydaną wersję CLB i dopiero tej wersji używaj do pisania aplikacji.

W kolejnych podrozdziałach przyjrzymy się pewnym zagadnieniom pracy z aplikacjami wieloserwerowymi. W pierwszej części zostaną omówione cele, jakie CLB próbuje osiągnąć. W kolejnej przyjrzymy się teorii kryjącej się za wewnętrznymi metodami pracy mechanizmu wyrównywania obciążenia. Na koniec przyjrzymy się niesprawnym serwerom. COM+ udostępnia pewne techniki służące do rozwiązywania tego problemu.

Cele mechanizmu wyrównywania obciążenia

Aplikacje przedsiębiorstwa muszą obsługiwać dużą liczbę użytkowników, którzy mogą mieć, lub nie, łatwy dostęp do serwera. Ta sama aplikacja, która umożliwia pracownikowi działu sprzedaży przyjmowanie zamówień przez telefon, musi poradzić sobie z pracownikami tego działu znajdującymi się w podróży lub nawet przebywającymi za granicą. Faktycznie, wraz z pojawieniem się Internetu, sprzedawca nie musi nawet być człowiekiem — może to być program komputerowy znajdujący się na drugim końcu linii obsługi klienta. Więc, pierwszym celem mechanizmu wyrównywania obciążenia jest danie aplikacji możliwości rozrastania się, by mogła pracować równie dobrze zarówno na jednym, jak i na kilku serwerach.

Sprzedawcy w podróży oraz użytkownicy, którzy nawiązują kontakt z twoją firmą przez Internet, mają jedną cechę wspólną: korzystają z połączeń telefonicznych, które w każdej chwili mogą ulec zerwaniu. Ostatnie artykuły zamieszczane we wszystkich wydawnictwach branżowych pokazały, że firmy potrzebują mechanizmu „odporności na błędy" dla aplikacji pracujących w trybie online. Niektórzy użytkownicy rozważają możliwość zaprzestania używania Internetu z powodu niewystarczającego poziomu obsługi, którego doświadczyli w przeszłości. Pomyśl, jakie spowodowałoby to straty, gdyby pracownik działu sprzedaży nie mógł śledzić na bieżąco zawartości firmowej bazy danych. Drugim celem mechanizmu wyrównywania obciążenia jest tworzenie aplikacji, które potrafią nie tylko tolerować awarie serwera, ale również sprawiają, że są one niewidoczne dla klienta (przynajmniej w takim stopniu, w jakim jest to możliwe).

Ostatecznie, strategia wyrównywania obciążenia COM+ jest zaprojektowana tak, by ułatwić pracę z uzupełniającymi się technologiami. Dlatego ważne jest, aby tworzone przez ciebie aplikacje działające na poziomie przedsiębiorstwa potrafiły współpracować z rozwiązaniami dostarczanymi przez inne firmy.

Jak działa mechanizm wyrównywania obciążenia?

W przeszłości klient kontaktowałby się bezpośrednio z serwerem, by uzyskać dostęp do zawartości bazy danych (sprawdzić coś, dopisać nowy rekord lub go zmodyfikować). Takie podejście problem. Jest nim to, że klient musi znać konkretną nazwę serwera oraz, bardzo często, również jego lokalizację. Oczywiście, jeżeli uruchamiasz aplikację na wielu serwerach, może to stać się przyczyną problemu zarówno dla administratora sieci, jak i autora aplikacji.

COM+ ma inne podejście do problemu dostępu. Zamiast łączyć się z serwerem, klient uzyskuje dostęp do routera. Router, na podstawie własnych informacji na temat dostępności poszczególnych serwerów, łączy użytkownika z najbardziej dyspozycyjnym i najmniej obciążonym serwerem. Router cały czas otrzymuje informacje kontrolne na temat dostępności poszczególnych serwerów. W tym celu korzysta z mechanizmu śledzenia czasu reakcji, który ułatwia mu dokonanie wyboru najlepszego serwera. Oczywiście, ponieważ każdy klient korzysta z tego samego routera, nie muszą znać lokalizacji i nazwy serwera, którego używają. Wszystko rozgrywa się w dynamicznym środowisku.

Uwaga

Na razie typowa konfiguracja mechanizmu wyrównywania obciążenia może składać się z jednego routera oraz z maksymalnie ośmiu serwerów (zwanych klastrem aplikacji). Kombinacja COM i MTS pozwala tym maszynom pracować razem w taki sposób, że użytkownik zyskuje możliwie najszybszą odpowiedź od aplikacji rozproszonej. Oczywiście, pojawią się takie aplikacje, które będą wymagać większej liczby serwerów (niż wspomniane osiem) i więcej niż jednego routera, by móc prawidłowo zarządzać całością. W związku z tym, firma Microsoft na pewno rozszerzy możliwości mechanizmu wyrównywania obciążenia.

Nie należy sądzić, że wyrównywanie obciążenia będzie miało wpływ na całą aplikację. Część twojej aplikacji może być zainstalowana na serwerze, inna — na maszynie użytkownika i jeszcze jedna na specjalizowanym serwerze w celu uzyskania możliwości dostępu do bazy danych. Wyrównywanie obciążenia działa na poziomie klasy. Musisz jasno określić klasę, która ma podlegać wyrównywaniu obciążenia, i następnie zainstalować ją na serwerze. W momencie gdy klient odwoła się do komponentu zawierającego taką klasę, system Windows 2000 „widząc", że ma ona podlegać działaniu wyrównywania obciążenia, będzie ją odpowiednio obsługiwał.

Podczas tworzenia klas COM+ obsługujących wyrównywanie obciążenia trzeba jednak pokonać kilka przeszkód. Ponieważ zasoby serwera są jednym z czynników wpływających na efektywność, powinieneś używać tej techniki jedynie w odniesieniu do „krótko żyjących" klas. Na przykład, klasa, która po uzyskaniu odpowiedzi serwera na zapytanie skierowane do bazy danych przekazuje jego wyniki użytkownikowi, a następnie ulega autodestrukcji — pasuje do tej kategorii. W dodatku, klasa nie może zakładać, że zostanie wywołana przez określoną maszynę. Oznacza to, że nie może polegać na zasobach (takich jak pliki) znalezionych na tej konkretnej maszynie. Klienty

będą potrzebowały również obserwować stan „krótko żyjących" klas podlegających działaniu mechanizmu wyrównywania obciążenia. Innymi słowy, kliencka część twojej aplikacji powinna tworzyć egzemplarz obiektu, użyć go, a następnie zniszczyć możliwie szybko.

Router wyrównywania obciążenia właściwie składa się z dwóch oddzielnych usług związanych ze sobą poprzez współużytkowanie obszaru pamięci. Pierwszą z nich *usługa wyrównywania obciążenia* (LBS), drugą zaś — *menedżer sterowania usługami* (SCM).

Usługa LBS zawiera kompletną listę komponentów podlegających wyrównywaniu obciążenia przechowywaną w bazie danych znajdującej się w pamięci. Celem tej usługi jest obsługa tablic routera wyrównywania obciążenia na potrzeby SCM. Wraz ze zmianami ilości komponentów (instalacja-usunięcie) dokonywanymi przez administratora sieci lub programistę, następuje odzwierciedlenie ich w tablicach routera wyrównywania obciążenia.

W momencie gdy klient formułuje wywołanie DCOM i wysyła je do routera wyrównywania obciążenia, to właściwie SCM odbiera to żądanie. SCM poszukuje komponentu w tablicy routera wyrównywania obciążenia, następnie sam przesyła wywołanie DCOM do jednego z serwerów klastra aplikacji, wymagając realizacji żądania. Jeden z serwerów znajdujących się w klastrze tworzy egzemplarz żądanego obiektu, następnie przekazuje pośrednika bezpośrednio do klienta. W tym momencie, serwer i klient mają ustanowione bezpośrednie połączenie — router nie jest już dłużej potrzebny.

W tym miejscu potrzebna jest usługa LBS. Formułuje żądanie odnośnie do działających serwerów, by uzyskać zestawienie czasu ich reakcji. Zestawienia te są następnie wprowadzane do tabeli routera wyrównywania obciążenia, która umożliwi mu prawidłowy wybór sprawnego serwera w klastrze aplikacji do obsługi kolejnego żądania komponentu.

Radzenie sobie z niesprawnymi serwerami i routerami

Router jest podstawowym sposobem obsługi niesprawnych serwerów. Część informacji w tablicy routera wyrównywania obciążenia oznacza stan serwera (sprawny-niesprawny). Router „wie", kiedy serwer uległ awarii, ponieważ w związku z tym nie odpowiada on na żądania pochodzące z części routera obsługującej wyrównywanie obciążenia. Kiedy takie zdarzenie ma miejsce, serwer jest oznaczany w tej tablicy jako niesprawny. Router również przydzieli innemu serwerowi obsługę żądań obsługiwanych przez nieczynny serwer. W ten sposób użytkownik nawet nie zauważy, że jeden serwer uległ awarii, a drugi go zastąpił.

Nieczynny router jest prawdziwym problemem, ponieważ stanowi główny punkt awarii. Możesz go skonfigurować jako zasób Microsoft Cluster Service (MSCS). W zasadzie stworzyłbyś wirtualną maszynę złożoną z kilku serwerów i w ten sposób znacząco zredukowałbyś szansę na wyłączenie routera z pracy. Nieprawdopodobne jest, by w tym samym czasie wszystkie serwery w klastrze zostały odłączone ze względu na awarię.