Paweł KASPROWSKI, Stanisław KOZIELSKI, Piotr KUŹNIACKI, Tadeusz PIETRASZEK
Politechnika Śląska, Instytut Informatyki

BEZPIECZEŃSTWO SYSTEMÓW BAZODANOWYCH DOSTĘPNYCH PRZEZ INTERNET

Streszczenie. W artykule przedstawiono analizę struktury, zagrożeń i zabezpieczeń wielowarstwowych systemów umożliwiających dostęp do baz danych poprzez Internet. Szczególną uwagę zwrócono na zagrożenia aplikacji internetowych i metody przeciwdziałania tym zagrożeniom, modele autoryzacji użytkowników, mechanizmy zabezpieczeń serwerów WWW, serwerów aplikacji i serwerów baz danych oraz ogólne procedury bezpieczeństwa w systemach informatycznych.

1. Wstęp

Celem pracy jest dokonanie przeglądu zagadnień związanych z bezpieczeństwem systemów informatycznych umożliwiających dostęp do baz danych poprzez Internet. W pierwszej części przedstawiono analizę struktury systemów bazodanowych, podkreślając ich rozwój od prostej architektury klient-serwer do architektury wielowarstwowej. Dokonano też krótkiego przeglądu technologii i narzędzi programowych umożliwiających budowę takich systemów.

Druga część opracowania zawiera analizę modeli i mechanizmów bezpieczeństwa odnoszących się do całego systemu i jego poszczególnych warstw. Omówiono też problem bezpiecznej komunikacji pomiędzy poszczególnymi warstwami.

Kolejna część pracy przedstawia analizę zagrożeń dla rozpatrywanych systemów, związanych ze stosowaniem różnorodnych technologii internetowych. Analizie tej towarzyszą sugestie jak unikać tych zagrożeń już na etapie projektowania aplikacji internetowych.

Pracę kończy omówienie ogólnych procedur bezpieczeństwa, których opracowanie i wdrożenie jest niezbędne w systemach informatycznych umożliwiających dostęp do baz danych przez Internet.

2. Architektura systemów bazodanowych

Typowy podział systemów bazodanowych wyróżnia:

- systemy scentralizowane,
- systemy wielowarstwowe.

Systemy scentralizowane budowane są zazwyczaj jako:

- duże systemy komputerowe (np. typu mainframe), w których zintegrowano z systemem zarządzania bazą danych aplikacje bazodanowe, obsługujące za pośrednictwem terminali wielu użytkowników,

lub też

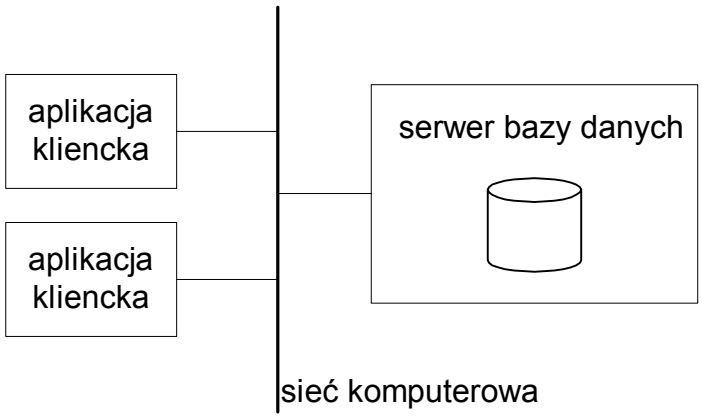
- małe (biurkowe) systemy (pojedynczy komputer PC bez dostępu do sieci), przeznaczone do współpracy z jednym użytkownikiem.

Bardziej interesujące, z punktu widzenia celów niniejszej pracy, systemy wielowarstwowe podzielić można na:

- systemy dwuwarstwowe (klient – serwer),
- systemy trójwarstwowe (klient – serwer aplikacji – serwer bazy danych),
- internetowe systemy wielowarstwowe (przeglądarka – serwer WWW – serwer aplikacji – serwer bazy danych).

2.1. Systemy klient - serwer

Typowa realizacja architektury klient-serwer (rys.1) obejmuje serwer bazy danych (zazwyczaj wykorzystujący język SQL) oraz szereg stacji roboczych, na których są wykonywane aplikacje klienckie.



Rys. 1. Architektura klient-serwer

Funkcje aplikacji w takiej architekturze obejmują zazwyczaj:

- obsługę komunikacji z użytkownikiem (interfejs użytkownika),

- wysłanie do serwera zleceń wykonania operacji na danych (w postaci instrukcji w języku SQL),
- odbiór danych z serwera i ewentualne dalsze ich przetwarzanie,
- prezentacja wyników operacji na bazie danych, wydruk raportów.

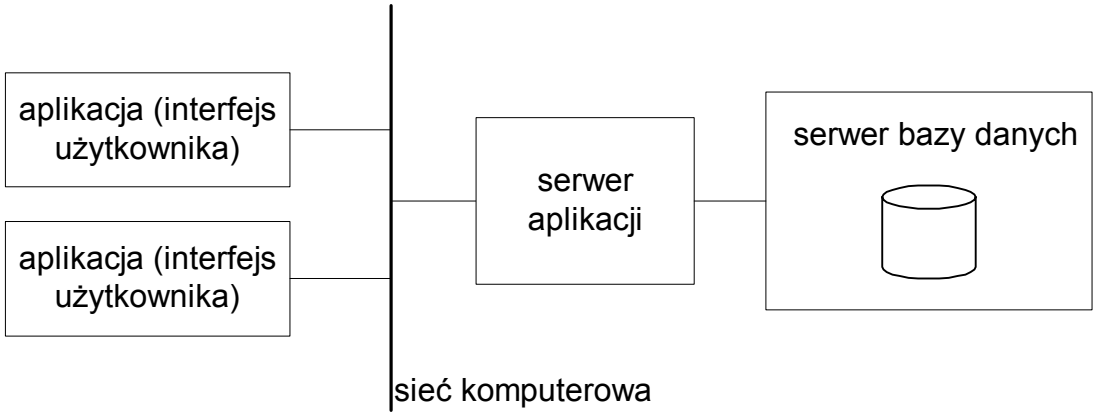
Serwer bazy danych realizuje (w dużym uproszczeniu) następujące funkcje:

- odbiór zleceń operacji na danych (w języku SQL) i kontrola uprawnień użytkownika do wykonania tych zleceń,
- analiza zleceń i ich optymalizacja,
- wykonanie operacji na bazie danych, w trybie realizacji transakcji,
- kontrola integralności bazy danych,
- wysłanie odpowiedzi do aplikacji klienckiej.

W skrajnym przypadku aplikacja kliencka nie wykonuje żadnego złożonego przetwarzania danych (tzw. „cienki" klient). W praktyce możliwości języka SQL w zakresie przetwarzania danych mogą nie wystarczać i wtedy część tego przetwarzania musi się odbywać w aplikacji klienta („gruby" klient). Może to pogorszyć efektywność pracy systemu, wymaga bowiem zwykle sprowadzenia z bazy danych, poprzez sieć, dużej liczby danych.

Alternatywą „grubego" klienta jest przeniesienie przetwarzania danych na komputer serwera i zrealizowanie tego przetwarzania w postaci tzw. procedur pamiętanych (ang. stored procedures), przechowywanych jako specjalny rodzaj obiektów w bazie danych i uruchamianych na zlecenie aplikacji klienckiej.

Etapem docelowym tego kierunku zmian w architekturze klient-serwer jest architektura trójwarstwowa: klient - serwer aplikacji – serwer bazy danych (rys. 2).



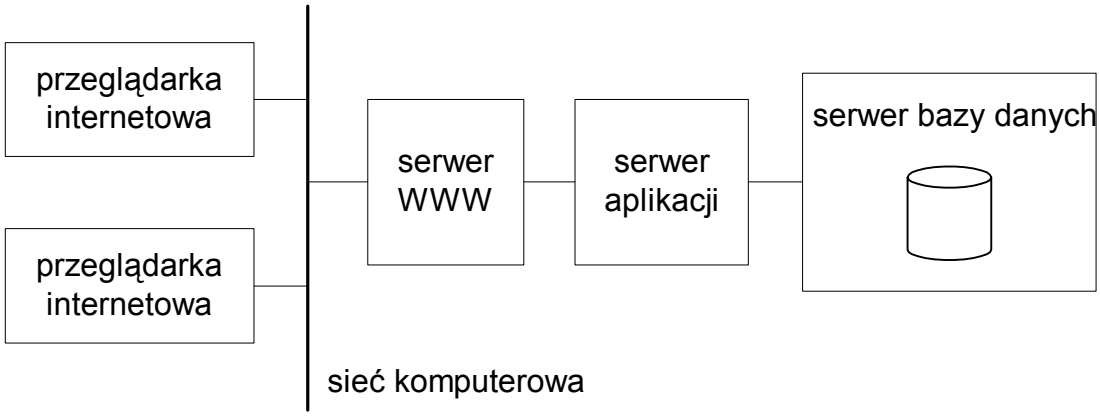
Rys. 2. Architektura trójwarstwowa

W architekturze tej serwer aplikacji przejmuje na siebie całe przetwarzanie danych (tzw. logika aplikacji). Ponadto serwer ten obsługuje również komunikację ze stacjami klienckimi, odciążając w tym zakresie serwer bazy danych. W pewnych rozwiązaniach (np. Tuxedo,

serwer aplikacji może również wspierać serwer bazy danych w zakresie zarządzania transakcjami. Zadania tej części aplikacji, która jest realizowana na stacji klienta ograniczają się do obsługi komunikacji z użytkownikiem (w tym prezentacji wyników przetwarzania danych). Zadania serwera bazy danych pozostają takie jak poprzednio, z uwagami zamieszczonymi wyżej przy opisie serwera aplikacji.

2.2. Wielowarstwowe internetowe systemy bazodanowe

Przedstawiony kierunek rozwoju architektury klient – serwer związany jest z dążeniem do poprawy efektywności przetwarzania danych. Innym problemem tej architektury, dotyczącym systemów o bardzo dużej liczbie stacji klienckich (dziesiątki, setki), jest zarządzanie wersjami aplikacji instalowanych na tych stacjach. Nadzorowanie instalowania nowych wersji aplikacji, zwłaszcza przy nieco różniących się wersjach systemu operacyjnego, jest bardzo kosztownym i uciążliwym zadaniem dla administratorów systemu. Radykalnym uproszczeniem tego problemu jest wykorzystanie technologii i metodologii internetowych do budowy wielowarstwowego systemu bazodanowego przedsiębiorstwa, czyli wykorzystanie Internetu. Architekturę takiego systemu przedstawia rys.3.



Rys. 3. Architektura wielowarstwowego bazodanowego systemu internetowego.

Zauważmy przy tym, że jest ona identyczna z architekturą typowych aplikacji internetowych korzystających z baz danych. Zadania realizowane w poszczególnych warstwach takiego systemu są następujące:

Przeglądarka: prezentuje strony internetowe (zapisane w języku HTML), umożliwia też przesłanie do serwera informacji wprowadzonych przez użytkownika.

Serwer WWW: odbiera zgłoszenia od przeglądarki i w najprostszym przypadku odsyła do przeglądarki żądaną stronę (zapisaną w języku HTML). W bardziej złożonych przypadkach (zgłoszenie z parametrami, specyficzne typy stron) serwer WWW kieruje stosowne żądanie do serwera aplikacji, następnie odbiera od tego serwera zapis strony (kod HTML), ewentualnie ostatecznie formatuje tę stronę i wysyła ją do przeglądarki.

Serwer aplikacji: na żądanie serwera WWW generuje zapis strony w języku HTML. Często spotykanym rozwiązaniem jest wykorzystanie tzw. aktywnych stron zawierających szablon w kodzie HTML oraz wstawki w dedykowanym, skryptowym języku programowania, który jest interpretowany (przetwarzany) przez serwer aplikacji. W tym procesie serwer aplikacji komunikuje się w szczególności z bazą danych: przesyła do niej zapytania SQL, odbiera dane wynikowe, w razie potrzeby przetwarza je i formatuje w języku HTML i udostępnia serwerowi WWW. Obecnie coraz częściej stosuje się rozwiązania, w których serwer aplikacji jest podzielony na 2 części: część skryptową zajmującą się przetwarzaniem skryptów i generowaniem stron HTML oraz część aplikacyjną, w której jest implementowana logika biznesowa aplikacji i część odpowiedzialna za współpracą z bazą danych. W takiej sytuacji pierwsza część serwera aplikacji może być zintegrowana z serwerem WWW.

Serwer bazy danych: realizuje poprzednio określone funkcje, w szczególności wykonuje otrzymane zapytania w języku SQL.

Przedstawiony pobieżnie przegląd architektury systemów bazodanowych oraz zadań realizowanych przez poszczególne warstwy prawie całkowicie nie uwzględniał aspektów bezpieczeństwa takich systemów. Problemom tym poświęcona jest dalsza część niniejszej pracy.

3. Technologie i narzędzia programowe wykorzystywane do budowy aplikacji internetowych

W ostatnich latach obserwujemy gwałtowny rozwój języków, środowisk i narzędzi programowych wykorzystywanych do budowy aplikacji internetowych, w tym wielowarstwowego systemów bazodanowych. W niniejszym rozdziale przedstawiony zostanie krótki przegląd wybranych technologii.

3.1. CGI

CGI (Common Gateway Interface) to historyczny już właściwie sposób rozszerzenia funkcjonalności serwera WWW o możliwość automatycznej generacji stron na żądanie użytkownika. Idea tej metody jest bardzo prosta. Jej istotą jest przekazanie przez serwer WWW zgłoszenia użytkownika innej, niezależnej od serwera WWW, aplikacji.

Aplikacja ta na swoje standardowe wejście otrzymuje treść zgłoszenia użytkownika, a na swoim standardowym wyjściu generuje kod w języku HTML. Kod ten jest przechwytywany

przez serwer WWW i po uzupełnieniu o odpowiednie dla protokołu HTTP informacje wysyłany do użytkownika. Zaletą tego rozwiązania jest możliwość stworzenia aplikacji obsługującej zgłoszenia w dowolnym języku programowania obsługującym standardowe wejście i wyjście. Niestety poważną wadą, która wpłynęła na zmniejszenie popularności tej metody - jest konieczność uruchomienia osobnej instancji aplikacji do przetworzenia zgłoszenia każdego użytkownika. W systemach dostępnych przez Internet, gdzie liczba jednoczesnych użytkowników może być bardzo duża, jest to rozwiązanie zupełnie nieefektywne. Próbowano temu zaradzić wprowadzając standard Fast-CGI, ale w tym czasie popularność zdobyły już rozwiązania rozszerzające funkcjonalność serwera WWW przez bezpośrednie dołączenia do niego bibliotek (np.: ISAPI).

3.2. PHP

PHP (PHP Hypertext Preprocessor) jest w chwili obecnej bardzo popularnym rozwiązaniem dla prostych serwisów internetowych ze względu na swoją prostotę, relatywnie duże możliwości i dostępność na wielu różnych platformach sprzętowych. PHP to język skryptowy oparty na języku Perl, pozwalający na generację stron HTML. Instalacja polega na takim skonfigurowaniu serwera WWW, aby odwołania do plików z rozszerzeniem php serwer przekazywał do specjalnej biblioteki. Pliki z rozszerzeniem php są zwykłymi plikami HTML, wewnątrz których istnieją tzw. ‘wstawki’ w języku PHP. Wstawki te pozwalają na generowanie treści HTML dostosowanej do żądania użytkownika.

Zaletą takiego rozwiązania jest jego prostota dla prostych przypadków. Najpierw tworzymy szkielet w HTML a później w miarę potrzeby rozbudowujemy go o kolejne wstawki w języku PHP. Tak więc program generuje kod HTML tylko tam, gdzie jest to konieczne. Wszelkie stałe elementy strony są zapisywane w ‘czystym’ języku HTML. Jest to zupełnie inne podejście niż np. w CGI, gdzie całość kodu HTML musiała być generowana w programie.

3.3. ASP

ASP (Active Server Pages) jest technologią firmy Microsoft. To środowisko skryptów po stronie serwera służące do tworzenia dynamicznych i interakcyjnych stron internetowych. Dokumenty ASP są plikami tekstowymi, które zawierają zarówno treść strony w postaci znaczników HTML jak i skrypty odpowiedzialne za zmiany wyglądu tejże strony. W momencie żądania klienta skrypty ASP są interpretowane i wykonywane bezpośrednio przez serwer internetowy, który następnie generuje kod HTML przesyłany przeglądarce. W odróżnieniu od technologii CGI, skrypty ASP są interpretowane, a nie są kodem wykonywalnym, co może sugerować mniejszą wydajność. Jednak narzuty czasowe oraz

zapotrzebowanie na zasoby są większe w przypadku technologii CGI, gdzie każdorazowe odwołanie do aplikacji wiąże się z tworzeniem oddzielnego procesu CGI. W przypadku ASP wywoływana aplikacja może pracować w ramach procesu serwera internetowego, co nie jest związane ze zwiększeniem zapotrzebowania na zasoby systemu. Dodatkowo wykorzystanie technologii ASP jest dużo bardziej wygodne, a aplikacje są bardziej bezpieczne w porównaniu z CGI.

Przy zastosowaniu ASP można wykonać wszystkie typowe zadania aplikacji internetowej: odczytywanie i przetwarzanie danych z formularzy, operacje na bazach danych, prowadzenie sesji użytkownika i śledzenie jego poczynań. Programista ma do dyspozycji szereg wbudowanych obiektów ASP, przy pomocy których można wygodnie przeprowadzać komunikację z użytkownikiem, utrzymywać i kontrolować sesję użytkownika, czy pobierać informacje na temat stanu pracy serwera, przeglądarki itp.

Połączenia aplikacji internetowej z bazą danych jest możliwe dzięki technologii ADO (ActiveX Data Objects), dzięki której dostęp do danych jest wyjątkowo łatwy i efektywny poprzez obiektową strukturę biblioteki. Ponadto ADO umożliwia pracę z różnego typu źródłami danych, a nie tylko z informacjami zgromadzonymi w bazach danych. I tak przykładowo informacje można otrzymywać z serwera indeksów lub z serwera pocztowego.

Programista projektując aplikację może oprzeć się na programowaniu obiektowym tworząc własne klasy z metodami i właściwościami. Dokumenty ASP mogą także posiadać wywołania składników COM, pozwalające wykonywać rozmaite złożone zadania. Zgodnie z obowiązującymi trendami, w przypadku większych projektów programista może rozdzielić poszczególne części aplikacji na oddzielne warstwy. Wydzielone warstwy prezentacji, logiki aplikacji (mogącej opisywać np. procesy ekonomiczne) oraz warstwy bazy danych, zaprezentowane w poprzednich rozdziałach, pozwalają zwiększyć bezpieczeństwo, zapewniają skalowalność oraz ułatwiają modernizację i pielęgnację systemu. Całość logiki aplikacji może zostać zamknięta w składniku COM wielokrotnego użytku, który można wykorzystywać w skryptach lub innych programach. Odizolowanie logiki aplikacji od warstwy związanej z wyglądem interfejsu zapewnia łatwe modyfikowanie systemu. Ukrycie struktury bazy danych i jej szczegółów implementacyjnych w wyizolowanym składniku ActiveX zwiększa bezpieczeństwo oraz zapewnia łatwe zmiany dotyczące bazy danych.

Środowisko ASP dzięki przetwarzaniu rozproszonemu i swojej skalowalności jest doskonałą platformą zarówno do tworzenia prostych aplikacji internetowych jak i rozbudowanych serwisów internetowych obsługujących setki a nawet tysiące użytkowników.

Na koniec warto wspomnieć, że ASP nie jest językiem, lecz terminem określającym technologię. Językami standardowo dostępnymi są VBScript oraz Jscript, lecz można

doinstalować dowolny język skryptowy zgodny z COM, przykładowo JavaScript, PERL czy Python.

3.4. Java Server Pages i technologie Javowe

Wśród technologii internetowych związanych z językiem Java, działających po stronie serwera, można wyróżnić następujące grupy: servlety, Java Server Pages oraz Enterprise Java Beans. Technologie te cechują się różnymi własnościami i działają na innej zasadzie, jednak wspólnie tworzą środowisko do pisania aplikacji działających po stronie serwera WWW. W dalszej części dokumentu zostaną one skrótowo scharakteryzowane. Kompletny opis środowiska Java 2 Enterprise Edition oraz specyfikacja Servletów znajdują się w [6] i [7].

3.4.1. Servlety

Z technologicznego punktu widzenia, servlety przypominają skrypty CGI, czyli programy wykonywane po stronie serwera, przetwarzające zapytania i generujące strony WWW. Programy te z reguły wykonywane są jako odpowiedź na wysłanie formularza ze strony i zajmują się weryfikowaniem poprawności i przetwarzaniem przesłanych danych. Cechą odróżniającą servlety od skryptów CGI jest to, że są one napisane w języku Java i skompilowane do postaci bytecode a następnie wykonywane w środowisku maszyny wirtualnej Javy działającej w ramach serwera WWW.

Programista projektujący servlety otrzymuje standardowe środowisko wraz z odpowiednimi klasami i metodami, umożliwiające realizowanie podstawowej interakcji z serwerem WWW (obsługę autoryzacji, nagłówki serwera WWW, pliki cookies i przetwarzanie przekazywanych parametrów, wsparcie dla sesji), dzięki czemu minimalizowane jest ryzyko popełnienia pomyłek a sama aplikacja tworzona jest szybko i efektywnie. Ze względu na wykorzystanie standardowego środowiska Javy programista może korzystać zarówno ze standardowego API, jak i dodatkowych bibliotek, a same programy są niezależne od platformy, na której są uruchamiane. W przypadku aplikacji bazodanowych z dostępem do WWW, szczególnie interesujący jest interfejs JDBC zapewniający jednolity dostęp do różnych baz danych.

W porównaniu do skryptów CGI napisanych w językach niższego poziomu, servlety cechują się na pewno niższą wydajnością, natomiast są dość dobrze skalowalne się w przypadku wysokowydajnych systemów z dużą liczbą klientów.

3.4.2. Java Server Pages

W przypadku servletów cała zawartość strony WWW musi być ręcznie tworzona przez programistę wewnątrz aplikacji, co powoduje, że każda zmiana wiąże się z koniecznością rekompilacji całego programu. Java Server Pages (JSP) umożliwiają połączenie statycznego

języka HTML z kodem dynamicznie generowanym za pomocą jezyka Java. JSP działa w sposób bardzo zbliżony do technologii Active Server Pages oraz PHP. Z technicznego punktu widzenia, strony JSP sprowadzają się do servletów, ponieważ przy pierwszym uruchomieniu strony JSP są przetwarzane do postaci kodu Javy i kompilowane do postaci bytecode.

Dzięki wykorzystaniu technologii JSP, część odpowiedzialna za prezentację danych i wygląd strony pozostaje w pliku tekstowym, z niewielkimi wstawkami kodu Javy, natomiast pozostała logika aplikacji może być zaszyta w niezależnych klasach Javy. Połączenie tych dwóch elementów realizowane jest za pomocą tzw. JavaBeans. Dodatkowo możliwe jest wywoływanie servletów z poziomu strony JSP w celu realizacji przetwarzania danych. Kolejnym ułatwieniem przy współpracy z aplikacjami bazodanowymi jest wykorzystanie języka XML i XSL do prezentacji danych.

Ze względu na swoje właściwości, servlety najlepiej nadają się do stron, gdzie przetwarzanie danych jest intensywne, a danych wynikowych jest mało (np. przetwarzanie formularzy i zapisywanie ich w bazie danych). Technologia JSP natomiast jest korzystniejsza przy stronach z dużą liczbą stałych elementów na stronie i mało skomplikowanym przetwarzaniu. Możliwe jest jednak połączenie tych technologii, co wykorzystuje się do tworzenia stron, w których przetwarzanie danych jest intensywne a zawartość złożona, szczególnie, jeżeli korzystają z bazy danych. W tej sytuacji właściwa strona obsługiwana jest za pomocą servletu, który realizuje całe przetwarzanie, a wyniki umieszcza w JavaBeans. Następnie wywoływana jest strona JSP, która korzystając z elementów umieszczonych w beans prezentuje wyniki użytkownikowi.

3.4.3. Enterprise Java Beans

Technologie servletów i JSP doskonale nadają się do prostych aplikacji webowych, tworzonych przez nieliczną grupę programistów, odpowiedzialnych również za wygląd strony. Technologie te nie zapewniają jednak odpowiedniego oddzielenia warstwy prezentacji od warstwy biznesowej, ani też nie zapewniają odpowiedniej skalowalności aplikacji.

Dlatego też, w przypadku bardziej złożonych systemów i aplikacji, stosuje się rozwiązania wielowarstwowe.

Warstwa klienta, podobnie jak w poprzednich technologiach jest oparta na przeglądarce WWW odpowiedzialnej za prezentację danych dla klienta. W zależności od rozwiązania, po stronie klienta mogą być także stosowane applety, formanty ActiveX i skrypty realizujące część przetwarzania.

Kolejną warstwą jest warstwa serwera WWW zajmującą się udostępnianiem stron w wykorzystaniem technologii JSP i servletów. Serwer ten korzysta również z Enterprise

Java Beans (EJB), w których jest zakodowana logika biznesowa aplikacji. EJB działają na serwerze aplikacji i komunikują się z platformą serwera WWW za pomocą RMI-IIOP.

Serwer aplikacji jest serwerem, w ramach którego działają Enterprise Java Beans w środowisku pojemnika EJB (EJB Containter). Zapewniają one mapowanie relacyjnej bazy danych na obiekty reprezentujące pojęcia biznesowe oraz rozwiązują kwestie transakcyjne. Uniezależnia to dalszą część aplikacji od fizycznej struktury bazy danych i umożliwia łatwe projektowanie aplikacji. Ostatnią warstwą jest warstwa relacyjnego serwera bazy danych.

Podsumowując, technologia Enterprise Java Beans umożliwia szybkie tworzenie złożonych i skalowanych aplikacji z dostępem poprzez WWW. Wykorzystanie warstwy serwera aplikacji umożliwia oddzielenie warstwy logiki biznesowej z jednej strony od sposobu prezentacji danych, z drugiej strony od implementacji serwera bazy danych.

4. Mechanizmy bezpieczeństwa w architekturze systemów bazodanowych

Zapewnienie bezpieczeństwa w wielowarstwowym internetowym systemie bazodanowym wymaga stosowania m.in. następujących działań:

- autentyfikację i autoryzację użytkowników w poszczególnych warstwach systemu,
- przesył danych oraz wrażliwych informacji bezpiecznymi kanałami systemu,
- unikanie w projektowaniu i implementacji aplikacji rozwiązań wrażliwych na atak,
- przestrzeganie ustalonych procedur bezpieczeństwa przez wszystkich użytkowników systemu.

W niniejszym rozdziale zajmiemy się pierwszym i drugim z wymienionych wyżej zagadnień. Pozostałe dwa zostaną omówione w kolejnych punktach pracy.

Autentyfikacja jest procesem identyfikacji użytkownika w systemie komputerowym, a następnie weryfikacji jego tożsamości. **Autoryzacja** dotyczy nadania użytkownikowi określonych uprawnień. W niniejszej pracy interesować nas będą przede wszystkim uprawnienia dotyczące dostępu do bazy danych i operacji na bazie.

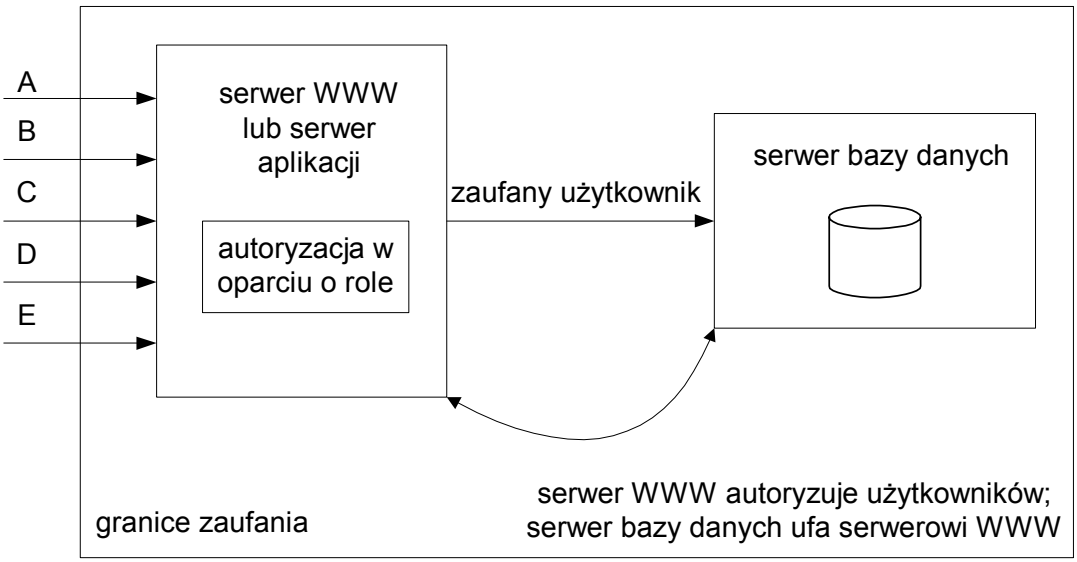
Autentyfikacja i autoryzacja mogą się odbywać w każdej warstwie wielowarstwowego internetowego systemu bazodanowego. W związku z tym w pierwszej kolejności rozpatrzymy ogólne modele bezpieczeństwa systemu wielowarstwowego. Ponieważ szczególne miejsce w takim systemie zajmuje baza danych, więc mechanizmy bezpieczeństwa stosowane w serwerach baz danych zostaną omówione z większym stopniem szczegółowości.

4.1. Modele bezpieczeństwa systemu wielowarstwowego

W tym punkcie zostaną omówione dwa różne podejścia do procesu autoryzacji użytkowników w wielowarstwowym systemie bazodanowym. Krótko wymienione zostaną zalety i wady obu rozwiązań.

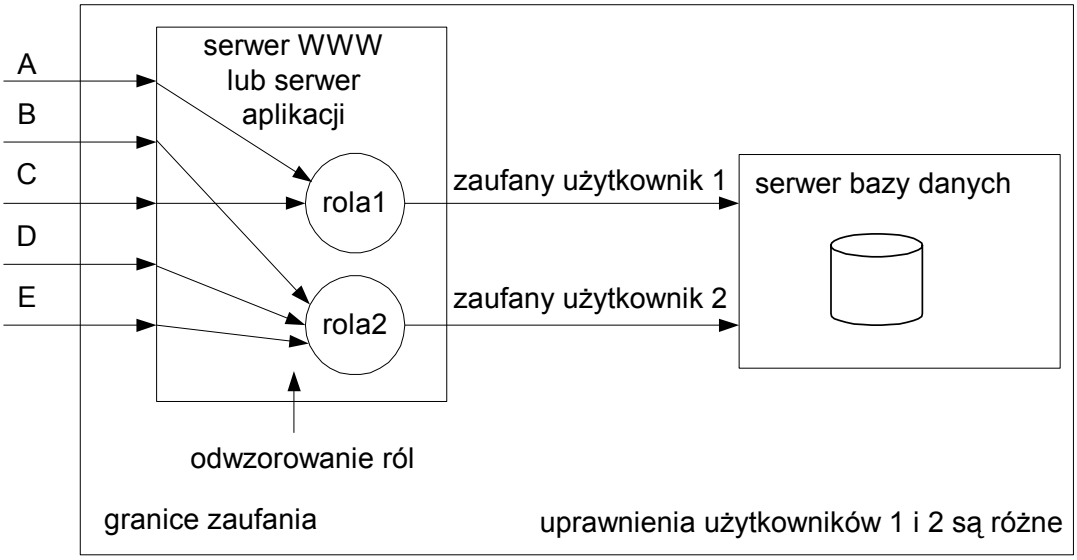
4.1.1. Model zaufanego podsystemu

W tym rozwiązaniu [1] autoryzacja użytkownika jest przeprowadzana w warstwie pośredniej (zwykle w serwerze WWW lub aplikacji), tworzącej pewnego rodzaju zaufany podsystem. Serwer bazy danych nie widzi wtedy indywidualnych użytkowników, realizuje tylko polecenia podsystemu autoryzującego, przedstawiającego się ustalonym identyfikatorem (rys. 4). Identyfikator taki ma przypisaną w bazie danych określoną rolę, związaną z pewnym zbiorem uprawnień.



Rys. 4. Model zaufanego podsystemu.

W systemach o wielu użytkownikach i złożonej funkcjonalności możliwe jest wyróżnienie kilku ról, przypisanych identyfikatorom reprezentującym różne grupy użytkowników (rys.5).



Rys. 5. Model zaufanego podsystemu z większą ziarnistością autoryzacji.

Zaletą omawianego modelu jest duża skalowalność. Określona pula połączeń z bazą danych, utrzymywana przez zaufany podsystem, może bowiem być wykorzystywana przez wielu użytkowników. Model ten upraszcza administrowanie bazą danych. Autoryzacja użytkowników w warstwie pośredniej, co zapewnia omawiany model, uniemożliwia też użytkownikom bezpośredni dostęp do bazy danych. Pośrednictwo zaufanego podsystemu w dostępie do bazy danych umożliwia kontrolę żądań użytkowników i zapewnia większe bezpieczeństwo.

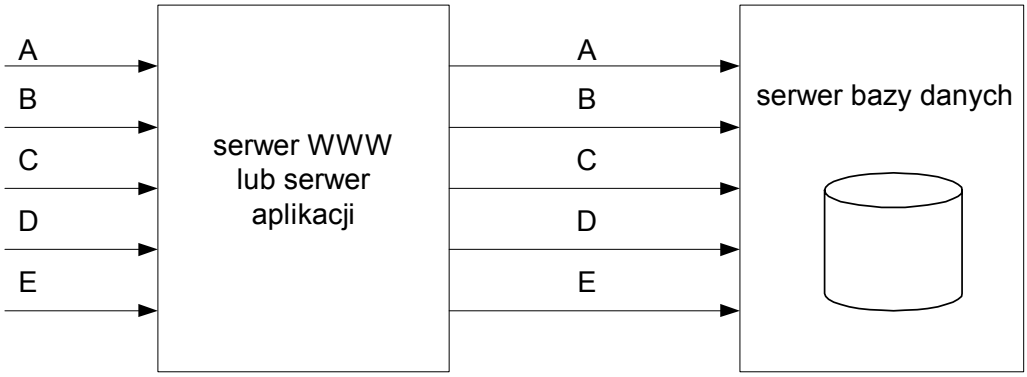
Wadą modelu zaufanego podsystemu jest ograniczenie możliwości prowadzenie monitorowania przez serwer bazy danych, bowiem nie dysponuje on danymi identyfikującymi klienta generującego zadania kierowane do bazy. Pośrednim wyjściem jest prowadzenie równolegle monitoringu w warstwie pośredniej (zaufanym podsystemie) oraz w bazie danych, a następnie skorelowanie wyników obu procesów monitorowania.

Inną wadą omawianego modelu jest wzrost ryzyka narażenia na szwank serwera bazy danych. Ponieważ warstwa pośrednia w tym modelu ma szerokie uprawnienia dostępu do bazy danych, więc jej ewentualne przejęcie przez nieuprawnionego użytkownika czyni łatwym późniejszy atak na bazę danych.

4.1.2. Model personifikacji/delegowania

Alternatywą zaufanego podsystemu, przejmującego odpowiedzialność za działania użytkowników, jest model [1], w którym żądania klientów zgłaszane do serwera WWW lub serwera aplikacji, są kierowane dalej do bazy danych wraz z identyfikatorami (kontekstem bezpieczeństwa) tych klientów (rys. 6). Warstwa pośrednia (serwer WWW lub serwer aplikacji) personifikuje tu klienta lub też deleguje jego żądania na odległy komputer serwera

bazy danych. Ostateczna autentyfikacja i autoryzacja użytkowników następuje w tym modelu w samym serwerze bazy danych



Rys. 6. Model personifikacji i delegowania użytkowników do bazy danych.

Zaletą zastosowania tego modelu autoryzacji użytkowników jest możliwość prowadzenia w serwerze bazy danych monitorowania działań użytkowników. Jest to możliwe, ponieważ serwer bazy danych potrafi zidentyfikować poszczególnych użytkowników. Podstawową wadą tego modelu jest natomiast ograniczona skalowalność, bowiem serwer bazy danych może być przeciążony obsługą indywidualnych połączeń wielu użytkowników. Większość technologii internetowych nie wspiera tego modelu.

4.2. Mechanizmy bezpieczeństwa w serwerach WWW i serwerach aplikacji

Warstwa serwera WWW i warstwa serwera aplikacji (często łączone w jedną całość) budowane są zazwyczaj przy wykorzystaniu istniejących narzędzi programowych. Narzędzia takie mogą udostępniać szereg mechanizmów bezpieczeństwa. Omówimy je na przykładzie wybranego systemu, o bogatej funkcjonalności, jakim jest serwer firmy Microsoft IIS (opisywany w [1]) w wersji 5.0.

Serwer internetowy IIS 5.0 posiada wiele mechanizmów mających za zadanie zwiększenia bezpieczeństwa aplikacji. Usługi serwera internetowego oferują trzy poziomy ochrony procesu aplikacji. Aplikacja może zostać uruchomiona w tym samym procesie co usługi sieci Web lub w oddzielnym procesie. Dodatkowo aplikacje mogą zostać uruchamiane w procesie buforowanym, czyli w innym oddzielnym procesie. Uruchamianie aplikacji w procesie usług sieci zwiększa wydajność kosztem potencjalnej możliwości zawieszenia usługi na skutek błędów aplikacji. Uruchamianie aplikacji izolowanych lub buforowanych jest bezpieczniejsze, jednak obniża wydajność.

Serwer internetowy IIS 5.0 posiada własną kontrolę uprawnień dotyczących aplikacji internetowej. Uprawnienia serwera mogą zezwalać na odczyt, zapis, przeglądanie katalogu

i dostęp do źródła skryptów. Dodatkowo kontrolę uprawnień można połączyć z uprawnieniami systemu plików NTFS, które mogą odnosić się do kont uwierzytelnionych użytkowników w ramach których pracuje aplikacja. Przy wywołaniu strony serwer internetowy identyfikuje użytkownika, sprawdza ważność jego konta oraz uprawnienia serwera a następnie uprawnienia do skryptów związane z NTFS.

Metody uwierzytelnienia, czyli potwierdzenie identyfikacji użytkownika są następujące:

- Uwierzytelnianie anonimowe – publiczny dostęp dla wszystkich, mapowanie na konto anonimowego użytkownika, uprawnienia wynikające z NTFS.
- Uwierzytelnianie podstawowe – część specyfikacji HTTP, logowanie na konto systemu Windows, hasło i użytkownik przesyłane przez sieć czystym tekstem.
- Uwierzytelnianie skrócone – część specyfikacji HTTP 1.1, logowanie na konto systemu Windows, hasło i użytkownik przed przesłaniem podlega mieszaniu.
- Zintegrowane uwierzytelnienie Windows – protokół uwierzytelniania Kerberos v5, identyfikacja użytkownika na podstawie logowania lokalnego.
- Uwierzytelnienie certyfikatów - mapowanie certyfikatów klientów na konta systemu Windows, używane przy połączeniach wykorzystujących protokół SSL.

Zabezpieczenia kanału informacyjnego może zostać przeprowadzone przy użyciu protokołu SSL, który jest wspierany w serwerze internetowym.

Niezbędnym elementem bezpiecznego systemu jest audyt i monitorowanie. Serwer internetowy IIS 5.0 posiada własne mechanizmy przeprowadzania audytu oraz logi. Można do tego także wykorzystać możliwości systemu operacyjnego, które pozwalają na audyt systemu plików oraz całości serwera.

4.3. Mechanizmy bezpieczeństwa w serwerach baz danych

Bazy danych są centralnym elementem rozpatrywanych w tej pracy systemów informatycznych. Przechowywane w nich dane mogą być narażone na wiele rodzajów zagrożeń. Do najważniejszych zaliczamy:

- nielegalny odczyt danych przez nieuprawnionych użytkowników,
- niepoprawne operacje modyfikacji danych, które mogą być skutkiem:
    - umyślnego działania nieuprawnionych użytkowników,
    - przypadkowych omyłek użytkowników,
    - braku właściwej kontroli przy współbieżnym dostępie do danych wielu użytkowników,
    - błędów oprogramowania lub awarii systemów komputerowych,
- zniszczenie danych w przypadku poważnych awarii sprzętu komputerowego.

Środki ochrony baz danych rozpatruje się zazwyczaj w następujących obszarach:

1. Autentyfikacja i autoryzacja użytkowników, określane też jako kontrola dostępu. Kontrola dostępu realizowana jest według określonych reguł, nazywanych polityką bezpieczeństwa.
2. Ochrona integralności danych.
   Mechanizmy zapewniające integralność baz danych można podzielić na mechanizmy integralności semantycznej oraz mechanizmy integralności transakcyjnej [3, 4]. Semantyczne więzy integralności definiują poprawny stan baz danych pomiędzy kolejnymi operacjami na bazie, stąd też umożliwiają ochronę, w pewnym zakresie, przed niepoprawną (umyślną lub przypadkową) modyfikacją danych. Mechanizmy integralności transakcyjnej chronią spójność bazy danych w warunkach współbieżnie realizowanych operacji na bazie przez wielu użytkowników, a także w przypadku błędów oprogramowania i awarii sprzętowych.
   Zapewnienie ochrony integralności należy do klasycznych zadań każdego systemu zarządzania bazą danych i nie będzie rozpatrywane w niniejszej pracy.
3. Monitorowanie operacji na bazie danych
   Wszystkie działania użytkowników istotne z punktu widzenia bezpieczeństwa systemu winny być monitorowane i rejestrowane. Analiza wyników takiej rejestracji może służyć do oceny poprawności przyjętej polityki bezpieczeństwa.
4. Szyfrowanie zawartości bazy danych.

W niniejszym podrozdziale przedyskutujemy punkt 1 powyższego wykazu, nawiązując też krótko do punktów 3 i 4.

W systemach zarządzania bazami danych (w skrócie nazywanych dalej serwerami baz danych) stosowane są dwa różne podejścia w zakresie polityki bezpieczeństwa [2, 3]:

- Uznaniowy model bezpieczeństwa (ang. discretionary security)
  W modelu tym dla każdego podmiotu (użytkownika) definiuje się reguły dostępu określające uprawnienia podmiotu do obiektów systemu. Ważną właściwością modelu uznaniowego jest możliwość nadawania przez określonego użytkownika posiadanych przez niego uprawnień innym użytkownikom.
- Obowiązkowy model bezpieczeństwa (ang. mandatory security)
  Model ten wymusza obowiązkową kontrolę dostępu do danych, realizowaną według ścisłej, hierarchicznej klasyfikacji podmiotów (użytkowników) i obiektów bazy danych. Każdemu obiektowi przypisuje się pewien poziom tajności, zaś podmiotowi – przepustkę. Porównanie rodzaju przepustki z poziomem tajności określa możliwości odczytu bądź zapisu danych.

W komercyjnych systemach baz danych jest powszechnie implementowany uznaniowy model bezpieczeństwa, stąd też tylko tym modelem będziemy się dalej zajmowali.

Mechanizmy tego modelu [3, 4] przedstawić można w kategoriach autentyfikacji i autoryzacji użytkowników.

Każdy użytkownik zdefiniowany w systemie zarządzania bazą danych musi mieć określony identyfikator (nazwa użytkownika/konto) oraz charakterystyczne cechy tożsamości Autentykacja użytkownika rozpoczyna się od jego identyfikacji (login), a następnie obejmuje ona weryfikację jego tożsamości (hasło, podpis cyfrowy lub t.p.). Autoryzacja obejmuje nadawanie użytkownikom uprawnień, które są zwykle dzielone na dwie kategorie:

1. uprawnienia systemowe, o charakterze ogólnym, odnoszące się do całej bazy danych (wszystkich jej obiektów),
2. uprawnienie szczegółowe (obiektowe), definiujące prawo do wykonywania określonej operacji na określonym obiekcie bazy danych.

Nadawanie uprawnień systemowych jest różnie realizowane w różnych Systemach Zarządzania Bazami Danych:

- W niektórych systemach (np. Oracle, SQLServer) określony jest duży zbiór uprawnień systemowych (kilkadziesiąt) nadawanych indywidualnie lub grupowanych w tzw. role.
- W innych systemach (np. Centura, Informix) użytkownicy dzieleni są na klasy, którym są przyporządkowane różne zakresy uprawnień (DBA – wszelkie prawa w bazie danych, RESOURCE – prawo tworzenia tablic i dysponowania nimi, CONNECT – tylko możliwość korzystania z przyznanych uprawnień).

Nadawanie uprawnień obiektowych wiąże się z przyznaniem danemu użytkownikowi prawa wykonywania określonej operacji języka SQL (SELECT, INSERT, DELETE, INDEX, ALTER, UPDATE) na określonym obiekcie (tablicy lub perspektywie). Jest ono relizowanie podobnie we wszystkich wymienionych systemach baz danych (polecenie GRANT).

Specyficznym aspektem rozważanych zagadnień jest umożliwienie propagacji uprawnień, tzn. przekazanie użytkownikowi prawa do przekazania nadanych mu uprawnień innym użytkownikom (opcja WITH GRANT OPTION). Wykorzystanie tej opcji może w sposób mało kontrolowany rozszerzać grono użytkowników operujących na zasobach bazy danych.

W serwerach baz danych wykorzystujących prezentowany model bezpieczeństwa stosowane są zwykle dodatkowe mechanizmy kontroli i zabezpieczeń. Omówimy je kolejno.

4.3.1. Perspektywy

Do szczególnych obiektów, objętych nadawaniem uprawnień, należą perspektywy (VIEWS). Mechanizm perspektyw pozwala tworzyć wirtualne tablice udostępniające użytkownikowi fragmenty zasobów jednej lub wielu tablic rzeczywistych, w formie prostej lub przetworzonej. Przy wykorzystaniu tego mechanizmu mogą być przyznawane

użytkownikom prawa dostępu nie do tablic rzeczywistych, a tylko do perspektyw. Pozwala to na wprowadzenie różnorodnych ograniczeń, w tym np. udostępnienie tylko wybranych kolumn bądź też wybranych wierszy tablic.

4.3.2. Limity zasobów – profile użytkowników

Poza ograniczeniami dostępu do danych (wynikającymi z przyznanych uprawnień) na użytkowników mogą być nakładane ograniczenia dotyczące zasobów systemowych, kontrolowanych przez System Zarządzania Bazą Danych. Należą do nich np.:

- ilość czasu procesora przeznaczonego na obsługę zadań określonego użytkownika,
- liczba równoczesnych sesji otwartych przez użytkownika,
- liczba odczytów (logicznych) z dysku przez użytkownika,
- dopuszczalny czas bez wykonywania operacji na bazie danych.

Zestaw nałożonych ograniczeń tworzy tzw. profil użytkownika.

4.3.3. Monitorowanie bazy danych

Z uznaniową kontrolą dostępu powiązana jest dostępna w niektórych serwerach baz danych możliwość tzw. audytu, czyli obserwacji i rejestrowania informacji o działaniach użytkowników. Taka funkcja, którą określamy jako monitorowanie, może np. obejmować:

- monitorowanie operacji: śledzenie wskazanych operacji (instrukcji) języka SQL (wykonanych lub odrzuconych),
- monitorowanie uprawnień: śledzenie wykorzystania uprawnień systemowych przez wskazanych użytkowników,
- monitorowanie obiektów: śledzenie wykonania wskazanych operacji (instrukcji) SQL na wskazanych obiektach.

Monitorowanie może być wykorzystane do podniesienia bezpieczeństwa systemu, przede wszystkim poprzez kontrolę działań użytkowników, którzy próbują przekraczać przyznane im uprawnienia. Poza tym monitorowanie jest wykorzystywane do strojenia serwera bazy danych.

4.3.4. Szyfrowanie w bazach danych

Dodatkowym mechanizmem zabezpieczeń dla baz danych może być szyfrowanie. Szyfrowanie może dotyczyć różnych informacji związanych z bazą danych. Najczęściej stosowanym zabiegiem jest szyfrowanie haseł użytkowników. Hasła przechowywane na dysku w postaci jawnej mogą się bowiem stać dość łatwym łupem włamywaczy penetrujących przestrzeń dyskową. Z podobnych powodów szyfruje się w pierwszej kolejności niektóre obiekty bazy danych takie jak procedury pamiętane, funkcje itp.

Ewentualna nieuprawniona ingerencja w ich treść mogłaby bowiem doprowadzić do uszkodzenia zawartości bazy danych. Ostatnim krokiem w stosowaniu omawianych mechanizmów jest szyfrowanie danych. Może ono dotyczyć szyfrowania zawartości plików jako całości, bądź też szyfrowanie poszczególnych rekordów. Celem szyfrowania danych jest przede wszystkim zabezpieczenie przed odczytaniem danych przy nieuprawnionym dostępie do fizycznych struktur danych (np. z poziomu systemu operacyjnego lub nawet poza nim, np. kradzież dysku).

4.4. Bezpieczne kanały komunikacyjne

W architekturach wielowarstwowych komunikacja pomiędzy warstwami odbywa się za pośrednictwem sieci z wykorzystaniem protokołu IP. W zależności od rozwiązania niektóre z tych sieci mogą być sieciami zaufanymi (np. wyodrębniony Intranet) inne natomiast publicznie dostępne – np. sieć Internet. Ze względu na charakter komunikacji w sieciach opartych na protokole IP można założyć, że komunikacja w sieci Internet może być w całości podsłuchana, zarejestrowana, przechwycona lub sfałszowana. Dlatego też należy przedsięwziąć środki mające na celu odpowiednie zabezpieczenie komunikacji. W Internecie stosowane są 2 standardowe protokoły zabezpieczeń: SSL i IPSec.

4.4.1. SSL

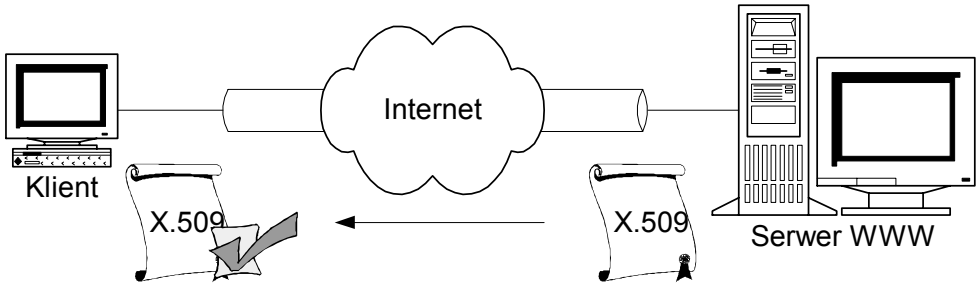
Protokół SSL (Secure Socket Layer), opracowany przez firmę Netscape (specyfikacja znajduje się w [10]), jest często wymieniany razem z nowszym protokołem TLS (Transport Layer Security) opracowanym przez IETF i spełniającym analogiczne funkcje. Protokoły te działają w warstwie aplikacji TCP/IP i realizują następujące 4 zadania:

1. **Identyfikacja serwera** – pozwala na potwierdzenie tożsamości serwera. Oprogramowanie klienta, wykorzystując mechanizmy kryptografii publicznej, może jednoznacznie stwierdzić, że serwer posiada certyfikat i klucz publiczny wydany przez zaufanego wystawcę certyfikatów (CA – Cerficate Authority).
2. **Identyfikacja klienta (opcjonalna)** – ten mechanizm umożliwia weryfikację certyfikatu klienta przez serwer, a co za tym idzie identyfikację klienta, na podobnej zasadzie na jakiej następuje identyfikacja serwera. Metoda ta może być stosowane zamiast np. wektora użytkownik - hasło. Ze względu na małą popularność certyfikatów X.509 wśród osób fizycznych, metoda ta nie jest powszechnie stosowana.
3. **Ochrona transmisji przed podsłuchem** – cała komunikacja klient – serwer jest szyfrowana już od samego początku, co uniemożliwia przechwycenie przekazywanych danych.

4. **Ochrona transmisji przed modyfikacją** – przesyłane zaszyfrowane pakiety dodatkowo chronione są przez funkcję skrótu, która umożliwia weryfikację ich poprawności i zabezpiecza przed nieuprawnioną ich modyfikacją.

Po nawiązaniu połączenia, w protokole SSL następuje negocjacja wersji protokołu oraz wybór algorytmów kryptograficznych (algorytmy szyfrowania oraz funkcja skrótu). Następnie następuje jedno- lub dwustronne uwierzytelnienie stron oraz wygenerowanie i bezpieczne przesłanie kluczy szyfrujących. Cała dalsza komunikacja jest szyfrowana za pomocą szyfru symetrycznego, co w małym stopniu wpływa na szybkość transmisji. W czasie transmisji możliwa jest renegocjacja parametrów oraz powtórne wygenerowanie kluczy sesyjnych.

Protokół SSL wprowadza dość duży narzut przy nawiązywaniu połączenia (negocjacje algorytmów, uwierzytelnianie i wymiana kluczy), natomiast sama transmisja danych nie jest kłopotliwa. Stwarza to pewne problemy (i obciąża serwer) przy transmisjach WWW, gdy strona składa się z wielu małych elementów i wykonywane jest wiele transmisji. W tej sytuacji stosowana jest opcja serwera *Keep-alive* a sam protokół SSL poprzez zapamiętywanie identyfikatorów sesji przyśpiesza renegocjację połączenia.



Rys. 7. Bezpieczna komunikacja za pomocą protokołu SSL.

Podkreślić należy, że protokół SSL umożliwia zabezpieczenie wyłącznie transmisji realizowanych za pomocą protokołu TCP/IP i jego implementacja wymaga modyfikacji aplikacji (ewentualnie możliwe jest tunelowanie). Do poprawnego działania protokołu SSL wymagany jest certyfikat X.509 serwera (oraz opcjonalnie klienta). Protokół ten najczęściej stosowany jest do bezpiecznego dostępu do stron WWW, rzadziej do szyfrowania komunikacji z bazami danych.

4.4.2. IPSec

Protokół IPSec jest protokołem opracowanym przez IETF (specyfikacja dostępna w [5]). Działa on w warstwie sieciowej, pomiędzy protokołami TCP i IP i umożliwia bezpieczne tunelowanie pakietów w sieci Internet. Zadania realizowane przez protokół są następujące:

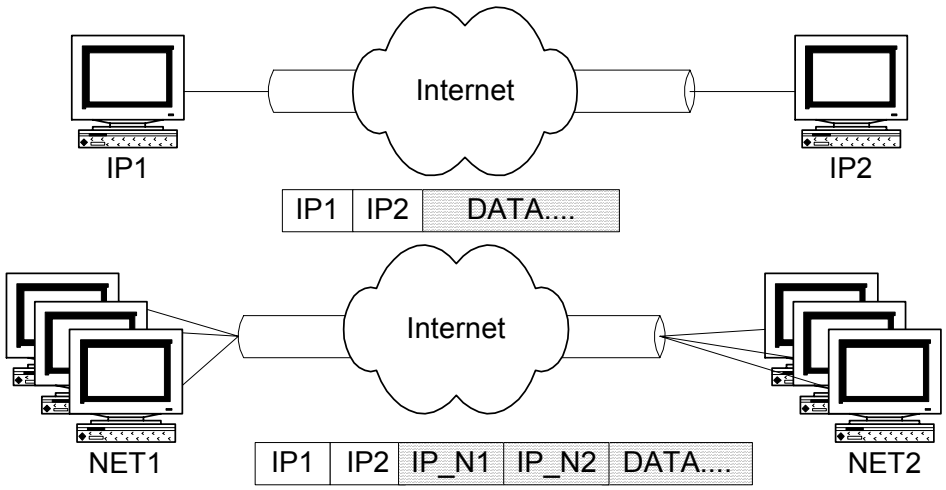
1. Weryfikacja nadawcy.
2. Ochrona danych przed podsłuchem i modyfikacją.

3. Zarządzanie kluczami kryptograficznymi.

W skład IPSec wchodzą 3 klasy protokołów: ESP – protokół kapsułkowania, AH – protokoł nagłówków uwierzytelniania, ISAKMP – protokół zarządzania kluczami. Podobnie jak w protokole SSL, w protokole IPSec również możliwa jest negocjacja funkcji skrótu i algorytmów kryptograficznych stosowanych do szyfrowania transmisji. Samo szyfrowanie jest dokonywane za pomocą szyfrów symetrycznych. Bezpieczeństwo może opierać się na certyfikatach X.509 lub dodatkowo współdzielonych kluczach.

Protokół IPSec działa na poziomie systemu operacyjnego i umożliwia stworzenie bezpiecznego kanału komunikacyjnego w 2 trybach (jak przedstawiono na rysunku 8):

1. **tryb transportowy** - w tym trybie szyfrowane są tylko dane z pakietu IP, gdyż komunikcja odbywa się pomiędzy dwoma komputerami.
2. **tryb tunelowy** – w tym trybie szyfrowane są wszystkie dane oraz nagłówki oryginalnych pakietów IP, co oznacza, że nadawca i odbiorca nie są znani. Tryb ten stosuje się do utworzenia wirtualnych sieci prywatnych pomiędzy dwoma sieciami.



Rys. 8. Tryby pracy protokołu IPSec.

Tunelowanie IPSec jest przezroczyste dla działających aplikacji i nie wymaga ich modyfikacji. W rozważanych systemach bazodanowych protokół ten stosuje się do komunikacji serwera aplikacji i serwer bazy danych, jeżeli nie znajdują się one w zaufanej i wydzielonej sieci.

5. Zagrożenia aplikacji internetowych i metody przeciwdziałania tym zgrożeniom

Aplikacje internetowe udostępniające bazy danych poprzez Internet narażone są na wiele zagrożeń. Zagrożenia te mogą być podzielone pod względem różnych kryteriów. Typowy podział ([8] i [9]) rozgranicza zagrożenia na wynikające z celowego działania nieuprawnionych użytkowników oraz na te, które nie są skutkiem celowego działania. Celowe działania osób nieuprawnionych mogą obejmować akty sabotażu, ataki czy kradzieże informacji. Do drugiej grupy można zaliczyć: awarie sprzętu, błędy i niedopatrzenia użytkowników, działania czynników zewnętrznych. Zagrożenia można także rozpatrywać zupełnie z innego punktu widzenia, nie rozróżniając źródła i charakteru zagrożeń, lecz poprzez wyznaczenie kto ma obowiązek im przeciwdziałać. W każdej fazie życia systemu występują zagrożenia, za które ktoś jest odpowiedzialny. I tak pewne rodzaje zagrożeń mają taki charakter, że muszą być przewidziane w procesie projektowania aplikacji internetowej przez programistę. Aplikacja musi być także poprawnie wdrożona, a całość systemu dobrze skonfigurowana przez administratora. Ostatecznie musi zostać przyjęta odpowiednia polityka bezpieczeństwa, tak aby poprawnie zaprojektowana i wdrożona aplikacja, pracująca w ramach dobrze skonfigurowanej architektury, nie była narażona na niebezpieczeństwa wynikające z błędów, nieuwagi, niedopatrzeń lub celowego działania ludzi.

5.1. Aspekt projektowania aplikacji

Aplikacje udostęniające dane przez Internet za pomocą protokołu HTTP posiadaja pewne specyficzne cechy, z powodu których ich projektowanie wymaga podjęcia przez projektanta wielu ważnych decyzji dotyczących problemów niewystęujących w innych architekturach.

Cechy te, to przed wszystkim:

- Brak stałego połączenia klienta z serwerem. Każde działanie użytkownika jest traktowane jako kolejne, niezależne połączenie. Wymaga to ustalenia sposobu podtrzymywania połączenia z użytkownikiem.
- Ogólnodostępność serwera. Z zasady serwer WWW odpowiada na zgłoszenia wszystkich komputerów w sieci Internet (lub wewnętrznej w przypadku rozwiązań intranetowych). Wymaga to dużej uwagi przy przydzielaniu uprawnień użytkownikom anonimowym.
- Możliwość realizacji kodu zarówno po stronie klienta jak po stronie serwera. Realizacja całości kodu po stronie serwera może poważnie ograniczyć funkcjonalność aplikacji. W związku z tym niektóre zadania przerzucane są na

komputer klienta. Poważnym problemem jest tu decyzja, które elementy aplikacji powinny być tak przerzucone.

Poniżej przedstawiono szersze opisy wymienionych właściwości aplikacji internetowych.

5.1.1. Brak stałego połączenia z serwerem

Protokół HTTP powstał w celu prostego udostępniania statycznych dokumentów HTML. W związku z tym komunikacja klienta z serwerem sprowadza się do wysłania żądania strony (jej adresu na serwerze) przez klienta i przesłania tej strony przez serwer. Na tym komunikacja się kończy. Jeśli użytkownik chce otrzymać następną stronę, ponownie nawiązuje kontakt z serwerem.

Rozwiązanie to jest bardzo niewygodne, szczególnie w przypadku konieczności autoryzacji przez użytkownika. Nie można przecież wymagać, aby z każdym żądaniem strony użytkownik wysyłał informacje uwierzytelniające w postaci wektora login, hasło. W związku z tym po pierwszym połączeniu użytkownika z serwerem i podaniu przez niego swoich danych identyfikacyjnych, dane te zostają zapamiętane na serwerze w postaci tak zwanej „sesji". Komputer klienta uzyskuje informację na temat numeru sesji, w której pracuje i od tej pory każde zgłoszenie klienta jest podpisywane jego numerem sesji. Po stronie klienta numer sesji przechowywany jest zwykle w polach ukrytych (specjalne pola w dokumencie HTML, które nie są widoczne dla użytkownika) lub w tzw. „cookies" – specjalnych strukturach danych przechowywanych przez przeglądarkę internetową. Rozwiązanie to jest na tyle powszechne, że wiele środowisk do tworzenia aplikacji internetowych ma wbudowane mechanizmy tworzenia i zarządzania sesjami.

Poważnym problemem takiego rozwiązania jest to, że użytkownik znający numer sesji i adres aplikacji na serwerze, może ominąć kolejne procedury zabezpieczające przez bezpośrednie wpisanie adresu żądanej strony. Jeśli na przykład wie, że naciśnięcie na stronie przycisku „Administracja kontami" powoduje wywołanie skryptu `administracja.asp` to, nawet jeśli dla niego przycisk ten jest nieaktywny, może spróbować połączyć się bezpośrednio ze skryptem wpisując:

`http://adres.serwera/administracja.asp`.

Tego typu wejście „tylnymi drzwiami" do aplikacji jest specyficzne dla aplikacji internetowych. Bronić się przed nim można na dwa sposoby:

- Przez zabezpieczenie aplikacji. W każdym skrypcie uruchamianym na serwerze sprawdzane jest, czy dany użytkownik ma prawo do uruchomienia zawartych w nim funkcji.
- Przez zabezpieczenie bazy danych. Każde odwołanie do bazy jest dokonywane w kontekście aktualnego użytkownika.

5.1.2. Ogólnodostępność serwera

Serwer WWW odpowiada na zgłoszenia wszystkich klientów wysyłających żądania na odpowiedni port. Możliwe jest oczywiście wprowadzenie ograniczeń dotyczących zakresu adresów IP, z których można dokonać połączenia, ale jest to rozwiązanie wysoce nieefektywne, ponieważ:

- ogranicza mobilność użytkowników,
- wymaga reakcji na wszelkie zmiany w konfiguracji sieci któregokolwiek z użytkowników.

Rozwiązanie to wymaga poza tym uważnej konfiguracji, tak aby zabezpieczyć się przed możliwością podszywania się (spoofingu).

W tej sytuacji projektant systemu powinien założyć, że dostęp do serwera może mieć każdy użytkownik sieci. Podstawowym problemem jest więc maksymalne ograniczenie uprawnień takiego nieautoryzowanego użytkownika. Użytkownik taki w szczególności powinien uzyskać jak najmniej informacji na temat wewnętrznej struktury serwera, co mogłoby mu pozwolić na przeprowadzenie ataku.

Częstym błędem jest pozostawianie w katalogu zawierającym pliki udostępniane przez Internet, dokumentów dotyczących konfiguracji serwera (na przykład pliku z hasłami użytkowników do bazy danych). Mimo, że nie ma możliwości dotarcia do tych plików z poziomu aplikacji, istnieją sposoby wyświetlenia listingu katalogu.

Zwykle - dla wygody projektowania - skrypty aplikacji uruchamiane są w kontekście użytkownika o dużych uprawnieniach systemowych. W związku z tym wszelkie błędy w skryptach mogą zostać wykorzystane przez potencjalnych włamywaczy do uruchamiania na serwerze kodu, który zagrozić może bezpieczeństwu danych.

Ponieważ użytkownik może próbować zgłaszać do serwera żądania o dowolnej treści (patrz przykład z poprzedniego punktu), podstawową zasadą projektowania aplikacji internetowych powinien być całkowity brak zaufania do nadchodzących zgłoszeń. Należy założyć, że z sieci może przyjść zgłoszenie o dowolnych parametrach przyjmujących dowolne wartości i być na to przygotowanym. Przykładowo, jeśli na jakiejś stronie pytamy użytkownika o jego wiek i w kodzie JavaScript po stronie klienta zabezpieczamy, żeby wprowadzona przez niego wartość była z zakresu 0-100, to aplikacja przyjmująca na serwerze wypełnioną stronę nie może zakładać, że ten warunek jest spełniony. Aplikacja musi być przygotowana na dowolną wartość pola wiek – także, na przykład, na ciąg tekstowy składający się z 8000 znaków (to bardzo popularna metoda ataków liczących na tzw. „przepełnienie bufora"). Należy także zwrócić uwagę na to, że dowolnym zmianom po stronie klienta podlegać mogą również informacje normalnie nie przeznaczone do edycji, jak pola ukryte w formularzach czy pliki cookies.

5.1.3. Możliwość realizacji kodu po stronie serwera lub klienta

W klasycznym rozwiązaniu aplikacja internetowa komunikuje się z klientem przesyłając mu statyczne dokumenty HTML. Dokumenty te zawierają najczęściej obszar formatki, na której znajdują się kontrolki pozwalające na odesłanie przez użytkownika informacji serwerowi.

Takie rozwiązanie ogranicza jednak możliwości dynamicznej współpracy użytkownika z aplikacją. Użytkownik nie może w żaden sposób wpłynąć na wygląd dokumentu-formatki, gdy została ona już przesłana na jego komputer.

W związku z tym, dla zwiększenia możliwości interfejsu użytkownika, często stosuje się częściowe przerzucenie przetwarzania na jego przeglądarkę. Może to odbywać się przez uruchomienie na komputerze użytkownika kodu – na przykład kontrolki ActiveX lub appletu Java – lub za pomocą „wzbogacenia" o większe możliwości kodu HTML przesyłanej strony. Tę drugą opcję realizuje się za pomocą wbudowanego w najpopularniejsze przeglądarki interpretera języka JavaScript.

Projektując aplikację mieszaną (składającą się z kodu po obu stronach kanału komunikacyjnego) należy pamiętać o tym, że kod uruchamiany po stronie klienta narażony jest na niebezpieczeństwo przechwycenia. Jest to banalnie proste w przypadku JavaScript i bardziej skomplikowane w przypadku appletów czy ActiveX (wymaga dekompilacji kodu).

Zasadą projektowania aplikacji internetowej powinno więc być nie umieszczanie w kodzie wykonywanym po stronie klienta żadnych informacji związanych z bezpieczeństwem (nazw użytkowników, haseł czy adresów). Kod ten służyć powinien tylko i wyłącznie rozszerzeniu możliwości graficznego interfejsu użytkownika, natomiast wszelkie działania związane z dostępem do bazy danych powinny być wykonywane dopiero przez kod umieszczony na serwerze.

5.2. Aspekt wdrażania i konfiguracji

Jednym z głównych zadań administratora systemu jest konfiguracja całości systemu w takim stopniu aby zapewnić maksymalne bezpieczeństwo pracujących aplikacji. Zakres zagrożeń z jakimi musi borykać się administrator jest ogromny, a każde z nich w istotny sposób wpływa na bezpieczeństwo przechowywanych w systemie informacji. Ich podział może być następujący:

- ataki na systemy,
- wirusy komputerowe,
- awarie sprzętowe.

5.2.1. Ataki

Konfiguracja systemów w taki sposób, aby zapobiec dostępowi do informacji przez nieuprawnionych użytkowników jest jedną z najważniejszych czynności każdego administratora. Ponieważ internetowy system baz danych jest ogólnie dostępny, jest on szczególnie narażony na różnego rodzaju ataki. Można je podzielić na:

- ataki pasywne – polegają na szukaniu lub podsłuchiwaniu przesyłanych informacji,
- ataki aktywne – opierają się na modyfikowaniu lub tworzeniu fałszywych danych.

Wśród ataków pasywnych można wymienić inżynierię społeczną, która polega na wyłudzaniu od osób informacji o hasłach bądź innych elementach bezpieczeństwa. Częste są także ataki na hasła, polegające na próbie siłowego złamania hasła lub wykorzystania ataku słownikowego. Obroną przed obu powyższymi atakami jest wprowadzenie i przestrzeganie odpowiednich procedur bezpieczeństwa oraz zaimplementowanie mocnego uwierzytelniania, opartego o dodatkowe mechanizmy jak certyfikaty klienta, weryfikacja posiadania dodatkowego fizycznego potwierdzenia tożsamości (stosowanie kart lub innych urządzeń).

Sniffing określa atak polegający na podsłuchiwaniu informacji przesyłanych siecią. Pozwala on na dostęp atakującego do cennych informacji jak numery kart kredytowych, hasła dostępu itp. Jest to szczególnie ważne w przypadku sklepów internetowych oraz aplikacji bankowych. Metodą obrony przed podsłuchiwaniem jest szyfrowanie transmisji w ramach bezpiecznego kanału informacyjnego.

Scanning to penetrowanie atakowanego komputera w celu znalezienia jego słabych punktów a następnie próba ich wykorzystania. Istnieje wiele programów i usług sieciowych systemów operacyjnych, które są potencjalnymi furtkami dla atakujących przykładowo sam serwer internetowy. Z tego powodu zadaniem administratora jest regularna analiza stanu systemów w celu znajdowania słabych punktów i ich usuwanie.

Spoofing to jedna z technik ataków aktywnych, polegającą na podszywaniu się pod inny komputer. Wśród wielu odmian tej techniki wymienia się podszywanie IP oraz DNS.

Hijacking to metoda ataku polegająca na przechwytywaniu sesji w protokole TCP i ściśle wiąże się z poprzednią. Metodami zapobiegającymi atakom tego typu są szyfrowanie transmisji na poziomie aplikacji (SSL), wykorzystanie kryptograficznej odmiany protokołu IP (IPSec) lub stosowanie analizatorów mogących wykrywać anomalie związane z atakiem.

DoS (Denial Of Service) to bardzo popularny atak polegający na próbie ograniczenia dostępu do usług. To ataki bardzo niebezpieczne i trudno się przed nimi zabezpieczać. Istnieje bardzo wiele odmian tego typu ataków wykorzystujących niedociągnięcia protokołów komunikacyjnych.

Ataki typu ***Buffer Overflow*** wykorzystują efekt przepełnienia bufora danymi o zbyt dużym rozmiarze, co może doprowadzić do nadpisania pewnych danych np. adresu powrotu

z funkcji, który znajduje się na stosie, dzięki czemu można wywołać dowolny kod. Zadaniem administratora systemu jest aktualizacja oprogramowania i rezygnowanie z usług wrażliwych na tego typu zagrożenia.

5.2.2. Wirusy komputerowe

Wirusy komputerowe są to programy komputerowe, które zakłócają prawidłową pracę systemów i aplikacji. Programy te mogą niszczyć dane, blokować lub obciążać system poprzez wykonywanie dodatkowej pracy. Próbując sklasyfikować wirusy komputerowe można je podzielić na:

- Wirusy – programy, które potrafią się rozmnażać wewnątrz plików wykonywalnych lub sektorach dysków.
- Makrowirusy – programy tworzone za pomocą makropoleceń rozprzestrzeniające się wraz z dokumentami np. edytorami tekstu lub arkuszami kalkulacyjnymi.
- Bakterie – to grupa programów samopowielających się, które zużywają zasoby komputera, jak procesor czy pamięć dyskowa.
- Robaki – grupa programów podobnych do bakterii, jednak polem ich działania jest sieć komputerowa, gdzie zmniejszają przepustowość, obciążają czy blokują serwery.
- Bomby logiczne – to wyszczególniona grupa wirusów, które aktywują się w określonym momencie zdeterminowanym wystąpieniem konkretnych okoliczności.
- Konie trojańskie – to wirusy, które udają pożyteczne programy. Pozyskując zaufanie użytkowników zdobywają cenne dane lub wykonują pewne czynności.

Najskuteczniejszą metodą walki z wirusami komputerowymi jest niedopuszczanie ich do systemów, co można osiągnąć przez zdefiniowanie odpowiedniej polityki bezpieczeństwa dla pracowników systemu. Jednak niezbędnym elementem bezpiecznego systemu jest uaktualniane i zaawansowane programowanie antywirusowe, opierające się na poszukiwaniu sygnatur wirusów, analizie heurystycznej i sprawdzaniu integralności plików.

5.2.3. Awarie sprzętowe

Ciągłe ulepszanie sprzętu komputerowego obniża jego awaryjność, jednak wzrastający stopień jego skomplikowania nadal jest źródłem możliwych problemów. Awarie sprzętowe można podzielić na:

- Awarie komputerów – drobne i pojedyncze awarie podzespołów komputerowych takich jak płyty główne, procesory, pamięci operacyjne czy monitory. To najczęściej spotykane uszkodzenia.

- Awarie nośników – uszkodzenia dotyczące nośników danych, czyli dysków twardych, dysków optycznych, taśmowych są szczególnie niebezpieczne, gdyż mogą wiązać się z trwałą utratą danych.
- Awarie sieci komputerowych – awarie sieci komputerowych mogą obejmować fizyczne zniszczenie okablowania sieciowego, skutkują one brakiem dostępu do systemów.
- Zdarzenia w centrum danych – to zdarzenia obejmujące największe zagrożenia takie jak pożar, trzęsienia ziemi czy kradzież.

Nie ma możliwości stworzenia bezawaryjnie pracującego systemu komputerowego, dlatego administrator musi zaimplementować mechanizmy, które mają na celu minimalizację utraty danych i jak najszybsze podjęcie pracy. Oprócz programowych mechanizmów ochrony danych, o których wspomniano w p. 4.3, stosowane są macierze dysków RAID. Głównym celem ich stosowania jest zwiększenie bezpieczeństwa przy awariach pojedynczych dysków oraz powiększenie pojemności i poprawienie efektywności. Popularnym rozwiązaniem jest stosowanie zasilaczy awaryjnych, niwelujące zakłócenia sieci energetycznej i potrafiącej podtrzymać pracę systemu w przypadku zaniku zasilania. W budowie serwerów stosowane sa technologie „hot" pozwalające na wymianę uszkodzonych podzespołów bez potrzeby wyłączania sprzętu. W przypadku dużych i ważnych centr danych stosowane są tzw. „gorące lokalizacje", czyli całe zapasowe systemy, skonfigurowane i gotowe do użytku.

6. Procedury bezpieczeństwa w systemach informatycznych

Dotychczas w niniejszej pracy rozważane były mechanizmy i środki bezpieczeństwa stosowane w procesie projektowania i budowy systemów bazodanowych dostępnych przez Internet (wbudowanych w te systemy). Środki te mogą się jednak okazać niewiele warte, jeśli nie towarzyszy im ogólniejsza struktura zabezpieczeń systemów komputerowych.

W przypadku aplikacji bazodanowych z dostępem przez WWW mamy do czynienia z przetwarzaniem danych i udostępnianiem ich użytkownikom. System taki podatny jest na różnego rodzaju zagrożenia, które mogą zakłócić jego działanie lub umożliwić dostęp do danych nieuprawnionym użytkownikom. W różnych instytucjach lub przedsiębiorstwach istnieją różne zagrożenia, a więc i różne sposoby przeciwdziałania im, dlatego też istotne jest istnienie polityki bezpieczeństwa informacji, czyli dokumentu określającego:

- jakie informacje są przetwarzane, jakie informacje są chronione oraz jakie są zagrożenia utraty i wykradzenia danych,
- jakie nakłady na zabezpieczenie danych są niezbędne oraz jaki jest koszt utraty informacji,

- kto i do jakich informacji powinien mieć dostęp,
- kto jest odpowiedzialny za zarządzanie informacją i jej gromadzeniem,
- w jaki sposób przechowywane są kopie zapasowe danych,
- jakie procedury są przewidziane na wypadek utraty danych.

Dokument taki powinien mieć charakter formalny i stanowić podstawę do opracowania procedur bezpieczeństwa. Poprawne opracowanie i wdrożenie takich dokumentów umożliwia identyfikację zagrożeń i opracowanie sposobów przeciwdziałania nim.

Rozpatrując problemy bezpieczeństwa systemów informatycznych nie można zapomnieć o czynniku ludzkim. Niedbalstwo i niewiedza ludzi, którzy pracują z systemem, może być bardzo dużym zagrożeniem bezpieczeństwa firmy. Z tego powodu musi istnieć strategia bezpieczeństwa, która dostarcza reguł ustalających sposób konfiguracji systemów oraz prawidłowe zachowania się poszczególnych pracowników zarówno w sytuacjach normalnych jak i niezwykłych okolicznościach. Ważne jest, aby w procesie utrzymywania bezpieczeństwa uczestniczyli wszyscy pracownicy, zarówno pracownicy odpowiedzialni za bezpieczeństwo, administratorzy systemów, jak i zwykli pracownicy.

Bezpieczeństwo całego systemu zależy w dużej mierze od poprawności pracy zwykłych pracowników. Poszczególne strategie jakie mogą być zdefiniowane dla pracowników mogą być następujące:

- strategia bezpieczeństwa,
- strategia korzystania z komputerów,
- strategia korzystania z Internetu,
- strategia korzystania z poczty.

Strategia bezpieczeństwa – powinna obejmować wytyczne związane z bezpieczeństwem firmy. Pracownicy powinni być poinformowani o zasadach doboru mocnych haseł, o zakazie ich udostępniania czy konieczności wylogowania się lub zablokowania komputera przy opuszczaniu miejsca pracy. Pracownikom powinny być także przekazane wytyczne związane ze strategią informacji, jak określenie cennych informacji, metody ich oznaczania, przechowywania, przesyłania i niszczenia.

Strategia korzystania z komputerów – pozwala określić jakie osoby i na jakich zasadach mogą korzystać z komputerów, które są własnością firmy. Strategia może określać, że komputery mogą być tylko wykorzystywane do celów służbowych, że nie wolno instalować nieautoryzowanego oprogramowania, zmieniać konfiguracji programowej i sprzętowej. Ważna może być także informacja, że brak jest prywatności pracowników, czyli ich prywatne dane mogą być przeglądane a nawet usuwane w związku z zapewnieniem bezpieczeństwa.

Strategia korzystania z Internetu – wyodrębniona część strategii korzystania z komputerów może dotyczyć Internetu. Powinna jasno definiować zakres korzystania z zasobów sieciowych i zabraniać odwiedzania stron niezwiązanych z pracą, ściągania nielegalnego oprogramowania, plików muzycznych lub filmowych itp. Nierozsądne korzystanie z Internetu może doprowadzić do zainfekowania komputerów.

Strategia korzystania z poczty – sposób korzystania przez pracowników z poczty elektronicznej także może być objęty oddzielną strategią w związku możliwością nipożądanego wysłania z firmy ważnych informacji. Ważne jest poinformowanie pracowników, iż zarówno poczta wychodząca jak i przychodząca może być przeglądana w celach bezpieczeństwa.

Sformalizowana strategia postępowania powinna dotyczyć także administratorów systemów. Przykładowe strategie są następujące:

- strategia administracji użytkownikami,
- strategia administracji systemu,
- strategia zmiany konfiguracji,
- strategia reakcji na incydent,
- strategia odzyskiwanie systemu po awarii.

Strategia administracji użytkownikami – powinna obejmować zagadnienia związane z przyjmowaniem nowego pracownika w firmie, przeniesienia pracownika wewnątrz firmy lub zwolnienia pracownika. Poszczególne procedury powinny jasno określać jakie kroki musi podjąć administrator w celu zapewnienia bezpieczeństwa.

Strategia administracji systemu – może określać jakie czynności administracyjne i jak często powinny być wykonywane. Mogą one obejmować unowocześnianie oprogramowania (patch'e), przegląd plików logów, szukanie słabych punktów konfiguracji systemu.

Strategia zmiany konfiguracji – ma definiować poszczególne podejmowane kroki przy próbie modyfikowania ustawień systemu bądź aplikacji. Przede wszystkim przed modyfikacją proponowane zmiany muszą być przeanalizowane, udokumentowane oraz przetestowane na testowym systemie, tak aby zapewnić maksymalne bezpieczeństwo. Ewentualne cofnięcie zmian powinno doprowadzić do poprzedniego spójnego, poprawnego stanu systemu.

Strategia reakcji na incydent – określa cele firmy w razie określonego incydentu. Strategia powinna określać, kto decyduje o podejmowanych krokach i jakie są priorytety działania np. ochrona danych, przywrócenie normalnej pracy, identyfikacja sprawcy.

Strategia odzyskiwanie systemu po awarii – powinna obejmować zagadnienia związane z odbudową infrastruktury komputerowej po awariach różnego typu. Może ustalać

kroki podjęte w celu uruchomienia zapasowego systemu, jeśli istnieje, lub utworzenia nowego.

Strategia pozwala wyznaczyć cele działań podejmowanych dla zapewnienia bezpieczeństwa i pozwala do nich dążyć. Jednak aby to zrealizować, wszyscy pracownicy muszą postępować zgodnie ze strategiami, które ich dotyczą. Nie jest to problem łatwy, gdyż tego typu strategie są zazwyczaj bagatelizowane lub specjalnie ignorowane w związku z utrudnieniami jakie wprowadzają. Dlatego ważne jest, aby każdy z pracowników odbył solidne szkolenie z zasad bezpieczeństwa, a całość akcji była promowana przez dyrekcję.

7. Ogólne bezpieczeństwo systemów informatycznych instytucji

Systemy bazodanowe z dostępem przez Internet nierzadko wykorzystywane są w dużych instytucjach, gdzie wartość przetwarzanych danych jest duża i stanowi podstawę działania przedsiębiorstwa. W takich przypadkach bezpieczeństwo systemów należy rozpatrywać w szerszym kontekście:

- bezpieczeństwo administracyjno-organizacyjne,
- bezpieczeństwo techniczno-programowe,
- bezpieczeństwo fizyczne,
- bezpieczeństwo prawne.

Bezpieczeństwo administracyjno-organizacyjne – umożliwia wykorzystywanie posiadanych mechanizmów obronnych oraz zarządzanie bezpieczeństwem systemów informatycznych. Podstawę bezpieczeństwa administracyjno-ogranizacyjnego stanowi strategia i polityka bezpieczeństwa informacji oraz opracowane na ich podstawie procedury.

Bezpieczeństwo techniczno-programowe - jest najszerszą grupą środków bezpieczeństwa. Dotyczy on projektu samych aplikacji i systemów przetwarzania danych, a w tym: identyfikacji i autoryzacji użytkowników, kontrolę dostępu, kryptograficzną ochronę danych, mechanizmów audytu i rozliczalności. Istotny jest również sposób przeprowadzania testowania i przeprowadzania instalacji nowych wersji aplikacji w czasie działania systemu, dokonywania i dokumentowania zmian. Na bezpieczeństwo ma istotny wpływ sposób i częstotliwość wykonywania zapasowych oraz stosowana ochrona przeciwko wirusom komputerowym. W obszarze tym znajdują się również stosowane systemy wykrywania nadużyć (IDS).

Bezpieczeństwo fizyczne – stanowi podstawę dla bezpieczeństwa techniczno-programowego. Bezpieczeństwo to dotyczy miejsc przetwarzania danych i centrów

zapasowych, a w tym: kontroli dostępu, ochrony przeciwwłamaniowej, ochrony przeciwpożarowej oraz fizycznego bezpieczeństwa miejsc przechowywania danych. Dla systemów komputerowych istotne jest zapewnienie dostępności danych – podtrzymywanie zasilania, bezpieczeństwo ciągów kablowych i łącz komunikacyjnych.

Bezpieczeństwo prawne – jest częścią zabezpieczeń formalnych i ma na celu zabezpieczenie interesów instytucji. Dotyczy umów serwisowych i zapewnienia ciągłości działania systemów informatycznych oraz ochrony danych przetwarzanych w instytucji – klauzule poufności, upoważnienia i pełnomocnictwa.

Podsumowując, należy podkreślić, że środki i zakres analizy i wdrożenia polityki bezpieczeństwa powinny być adekwatne do wielkości i istotności systemu informatycznego dla przedsiębiorstwa. Należy też zwrócić uwagę na to, aby funkcjonalność i efektywność systemu nie obniżyły się zbyt drastycznie, gdyż bezpieczeństwo uzyskiwane jest zwykle ich kosztem, a ograniczenie to może powodować obniżenie dynamiki funkcjonowania organizacji.

LITERATURA

1. Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication, http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnnetsec/html/secnetlpMSDN.asp.
2. Castano S., Fugini M. G., Martella G., Samarati P.: Database Security. Addison-Wesley Publishing Company, 1995.
3. Date C. J.: An Introduction to Database Systems (7th ed.). Addison-Wesley, 2000.
4. Elmasri R., Navathe S.: Fundamentals of Database Systems. Addison-Wesley, 2000.
5. IP Security Protocol (ipsec) Charter, http://www.ietf.org/html.charters/ipsec-charter.html.
6. Java Servlet Technology - Documentation, http://java.sun.com/products/servlet/docs.html.
7. Java 2 Platform, Enterprise Edition – Documentation, http://java.sun.com/j2ee/docs.html.
8. Maiwald E.: Bezpieczeństwo w Sieci. Wydawnictwo edition, 2000.
9. Stokłosa J., Bilski T., Pankowski T.: Bezpieczeństwo danych w systemach informatycznych. Wydawnictwo Naukowe PWN, 2001.
10. SSL 3.0 Specification, http://wp.netscape.com/eng/ssl3/.

Recenzent: Prof. zw. dr inż. Stefan Węgrzyn