STUDIA INFORMATICA 2002
Volume 23 Number 2B (49)

Henryk KRAWCZYK, Michał WIELGUS
Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

MODELE BEZPIECZEŃSTWA APLIKACJI ROZPROSZONYCH

Streszczenie. Przedstawiono charakterystykę aplikacji rozproszonych, typowych zagrożeń oraz mechanizmów zabezpieczających. Wskazano zasady bezpieczeństwa wykorzystywane w systemach wielowarstwowych. Wprowadzono nowe modele bezpieczeństwa i pokazano kierunki ich integracji.

SECURITY MODELS FOR DISTRIBUTED APPLICATIONS

Summary. Distributed applications are discussed and typical threats and relevant security mechanisms are presented. Main security rules suitable for N-tier environments are discussed. New security models are proposed and directions of their integration are shown.

1. Wprowadzenie

Problem bezpieczeństwa systemów rozproszonych staje się coraz bardziej istotny i stanowi autentyczne wyzwanie dla projektantów i administratorów serwerów WWW (*World Wide Web*) czy serwerów aplikacji. Zapewnienie całkowitego bezpieczeństwa systemu jest w praktyce niemożliwe, dlatego rozważa się kryteria pozwalające na klasyfikację oprogramowania względem zadanych poziomów bezpieczeństwa. Wyróżniamy tzw. kryteria elementarne CC (*Common Criteria*), które umożliwiają zarówno określenie wymagań bezpieczeństwa, jak i dostarczają odbiorcom produktów mechanizmy ich weryfikacji i walidacji. Definiuje się siedem następujących poziomów oceny zabezpieczeń EAL (*Evaluation Assurance Levels*) [1]:

EAL1 zgodność implementacji ze specyfikacją (ang. *functionally tested*),

EAL2 spełnienie wymagań konstrukcyjnych (ang. *structurally tested*),

EAL3 sprostanie systematycznej analizie i kontroli (ang. *methodically tested and checked*),

EAL4 spełnienie wymagań właściwej konstrukcji, testowania i inspekcji (ang. *methodically designed, tested and reviewed*),

EAL5 zapewnienie półformalnej metody konstrukcji i testowania (ang. *semi formally designed and tested*),

EAL6 zapewnienie półformalnej weryfikacji etapu konstrukcji i testowania (ang. *semi formally verified design and tested*),

EAL7 osiągnięcie formalnej weryfikacji etapu konstrukcji i testowania (ang. *formally verified design and tested*).

Wyższy numer kryterium oznacza wyższy stopień bezpieczeństwa oprogramowania. Poziomy EAL1 – EAL3 wykorzystują podstawowe techniki inżynierii oprogramowania, poziomy EAL4 – EAL7 wymagają uwzględnienia specjalistycznych technik inżynierii bezpieczeństwa. Techniki te bazują na odpowiednich modelach formalnych.

W pracy scharakteryzowano aplikacje rozproszone pod względem bezpieczeństwa i zaprezentowano wybrane modele przydatne do jego analizy. Szczególną uwagę zwrócono na analizę zagrożeń i minimalizację skutków ich zaistnienia, a także na ocenę zaufania do wytwarzanej aplikacji.

2. Model środowiska

Typowe środowisko rozproszone prezentuje architekturę trójwarstwową [2]:

- warstwa serwera WWW / warstwa zewnętrzna,
- warstwa serwera aplikacji,
- warstwa serwera bazy danych.

W warstwie WWW powinny funkcjonować wyłącznie te usługi, które są świadczone klientom internetowym. Zazwyczaj oznacza to obecność serwisów DNS (*Domain Name System*), SMTP (*Simple Mail Transfer Protocol*), HTTP (*Hyper Text Transfer Protocol*). W żadnym wypadku nie powinny się tu znaleźć serwery świadczące usługi z przeznaczeniem dla zamkniętego środowiska wewnętrznego. Systemy tej warstwy są wystawione na największe zagrożenia ze względu na bezpośredni kontakt z Internetem.

Komunikacja warstwy aplikacji zarówno z warstwą zewnętrzną, jak również warstwą danych sprawia, że zapewnienie bezpieczeństwa staje się dwukierunkowe. Dodatkowo w tej warstwie będą działały systemy, z którymi zachodzi komunikacja z warstwy WWW, ale nie świadczą one usług internetowych. Podstawowa zasada dotycząca zabezpieczania środowisk

rozproszonych mówi, że serwer świadczący określone usługi powinien zostać umiejscowiony w najniższej, jak to tylko możliwe, warstwie architektury.

Warstwa danych stanowi najbardziej newralgiczny obszar całego środowiska. Serwery w tej warstwie świadczą usługi zarówno warstwie aplikacji, jak również warstwie danych. Fakt braku dostępu do tej warstwy z zewnątrz nie zwalnia z obowiązku jej zabezpieczenia.

Podstawowa zasada podejścia obronnego (ang. *defense-in-depth*) stanowi, że żadna warstwa środowiska nie może stać się punktem zagrożenia systemu. Izolacja i odseparowanie usług w odpowiednich warstwach stanowią zabezpieczenie w tym sensie, że newralgiczne dane znajdują się w najniższej warstwie i żeby uzyskać dostęp do nich, trzeba pokonać zabezpieczenia wszystkich warstw wyższych. Nieuprawniony dostęp może się odbywać zarówno z zaufanego środowiska wewnętrznego, jak i z zewnątrz. Implementacja rozdzielnych sieci i warstw architektury zwiększa dodatkowo zdolności odpowiedzi na zagrożenia bezpieczeństwa w czasie rzeczywistym.

Fizyczna separacja lub warstwowy podział systemów świadczących podobne usługi i niosących podobne zagrożenia są kluczowym aspektem modelu bezpieczeństwa, a grupowanie komponentów stanowi podstawy do budowy rozproszonych środowisk wielowarstwowych. Proces zabezpieczania systemu rozproszonego wymaga podjęcia następujących kroków:

- wprowadzenia dedykowanej funkcjonalności,
- systematycznego podnoszenia wymagań bezpieczeństwa,
- instalacji filtrów ochronnych,
- minimalizacji świadczonych usług.

3. Modele bezpieczeństwa aplikacji

3.1. Bezpieczeństwo apletów

Aplety postrzegane są jako kod programu, który może być załadowany z serwera za pomocą sieci i natychmiast wykonywany na maszynie klienta. Ze względów bezpieczeństwa zbiór funkcji dostępnych dla apletów jest znacznie ograniczony. Najważniejsze zasady, które muszą zostać spełnione pod kontrolą zarządcy bezpieczeństwa (ang. *security manager*) w odniesieniu do apletów [3]:

- nie mogą ładować bibliotek lub definiować podprogramów,
- nie mogą w zwykły sposób czytać i zapisywać plików na komputerze, na którym są uruchomione,

- nie mogą tworzyć bezpośrednich połączeń sieciowych poza połączeniami z serwerem, z którego pochodzą,
- nie mogą uruchamiać żadnych programów na komputerze, na którym zostały uruchomione,
- nie mają dostępu do wszystkich właściwości systemu (ang. *system properties*).

Przeglądarka apletów pozwala uruchamiać z pełnymi prawami każdy aplet umieszczony w pliku JAR (*Java Archive*) podpisany przez zaufaną stronę (wytwórca oprogramowania, organizacja certyfikująca). Aplet taki nie podlega wymienionym zasadom modelu bezpieczeństwa.

Różnica między klasycznym modelem bezpieczeństwa opartym na zarządcy bezpieczeństwa a przedstawionym mechanizmem opartym na podpisach elektronicznych polega na kontroli dostępu do zasobów systemowych. W pierwszym przypadku kontrola odbywa się na poziomie elementarnych operacji dokonywanych przez aplet tzw. kontrola "drobnoziarnista"; w przypadku apletów podpisanych "binarny" model zaufania umożliwia kontrolę dostępu typu "wszystko albo nic". Najnowsze wersje Javy pozwalają również podpisanym apletom na selektywny dostęp do zasobów, a ich wybór pozostawiono do decyzji użytkownika.

3.2. Bezpieczeństwo agletów

Aglety są mobilnymi obiektami mogącymi przenosić się z maszyny na maszynę za pomocą sieci. Raz uruchomiony aglet samodzielnie decyduje, w jaki sposób i na jakiej maszynie będzie kontynuował swoje obliczenia. Pozwalają na sterowanie ruchem w sieci, ale potencjalnie mogą też spowodować efekt odwrotny do zamierzonego, polegający na zawłaszczeniu całej dostępnej przepustowości sieci, i w konsekwencji doprowadzić do sytuacji, w której nastąpi odmowa obsługi napływających zgłoszeń.

Oprogramowanie typu *Aglets Framework* implementuje rozbudowany wielowarstwowy model bezpieczeństwa. Do zadań pierwszej warstwy należy wieloetapowe sprawdzanie kodu agletu w zakresie poprawności i weryfikacji spójności kodu bajtowego Javy. Kontrolę nad kolejną warstwą sprawuje zarządca bezpieczeństwa. Wykonanie przez aglet jakiejkolwiek potencjalnie niebezpiecznej dla systemu operacji jest przez zarządcę bezpieczeństwa automatycznie blokowane. Ostatnią warstwą jest interfejs funkcji *Java Security*. Swoim zasięgiem obejmuje on wszelkiego rodzaju funkcje kryptograficzne, pozwalające na szyfrowanie, tworzenie podpisów elektronicznych, na autoryzacji kończąc.

Dostępne obecnie systemy mobilnych agentów wykorzystują następujące mechanizmy:

- identyfikacja i weryfikacja autentyczności użytkowników i domen,
- kontrola integralności komunikacji między serwerami danej domeny,

– "drobnoziarniste" mechanizmy autoryzacji oparte na modelu bezpieczeństwa JDK.

Poważnym zagrożeniem dla bezpieczeństwa systemu związanym z odmową świadczonych usług jest rozproszony atak odmowy obsługi DDoS (*Distributed Denial of Service Attack*). Nie istnieją skuteczne środki pozwalające na wyeliminowanie zagrożeń związanych z tego rodzaju atakiem. Prawie zawsze istnieje możliwość spowodowania odmowy obsługi zgłoszeń napływających do serwera.

Strategia ataku DDoS opiera się na czterech filarach [4]:

- wykorzystaniu niejawnych kanałów komunikacyjnych,
- maksymalnym zawłaszczeniu przepustowości,
- całkowitym wyeliminowaniu zdolności obronnych,
- zachowaniu anonimowości intruza.

Analiza bezpieczeństwa powinna dotyczyć zarówno jawnych, jak i niejawnych kanałów komunikacyjnych (ang. *covert channel analysis*). Pod pojęciem jawnych kanałów komunikacyjnych kryje się chroniony przepływ informacji, natomiast niejawne kanały komunikacyjne wykorzystują mechanizmy, które z założenia nie są obiektami danych. Niejawne kanały komunikacyjne można podzielić na dwa rodzaje, przy czym należy pamiętać, że rozróżnienie to nie jest jednoznaczne [5]:

- kanały magazynujące (ang. *storage channels*),
- kanały czasowe (ang. *timing channels*).

Ryzyko związane jest z tymi kanałami, które pozwalają na komunikację wbrew przyjętej dla systemu polityki bezpieczeństwa. Do podstawowych własności niejawnych kanałów komunikacyjnych należą: fakt ich istnienia, co oznacza występowanie kanałów zdolnych do transmisji danych, oraz przepustowość, czyli szybkość, z jaką informacja może zostać przesłana.

3.3. Etapy zabezpieczeń

Zabezpieczenie serwerów aplikacji przed potencjalnym zagrożeniem (atak intruza) ma trzy podstawowe aspekty: techniczny (wspierany przez kryptografię, kodowanie informacji, wykorzystanie biometrii dla identyfikacji użytkownika czy wprowadzenie serwerów proxy w celu eliminacji bezpośredniego kontaktu między klientem a serwerem), prawny (polegający na zbieraniu dowodów w celu ustalenia winnego skutków zaistniałych sytuacji) oraz finansowy (związany z wydatkami na mechanizmy zabezpieczające i kosztami niezbędnymi na eliminację zaistniałych problemów). Biorąc pod uwagę te trzy aspekty, wyróżnia się sześć etapów procesu zabezpieczania aplikacji:

1. Wykorzystanie wiedzy (dzięki metodom sztucznej inteligencji ciągła rozbudowa wiedzy o zagrożeniach i skutkach w celu szybkiego ich wykrywania i alarmowania osób odpowiedzialnych za bezpieczeństwo).
2. Ocena zagrożenia (dzięki metodom modelowania określenie, jak dany atak lub zagrożenie zamanifestuje się w systemie i w jaki sposób będzie oddziaływał na jego środowisko).
3. Unikanie zagrożeń (dzięki usprawnieniu metod projektowania sieci i serwerów, wprowadzeniu dodatkowego oprogramowania oraz szkoleniu personelu administracyjnego).
4. Wykrywanie ingerencji (poprzez monitorowanie pracy systemu – agenci SNMP (*Simple Network Management Protocol*), wykorzystanie analizatorów pakietów w celu odróżnienia normalnego i potencjalnie niebezpiecznego zachowania).
5. Podjęcie reakcji (poprzez szybką odpowiedź na wykrycie intruza, zewnętrzny audyt zdarzeń oraz modyfikację przepisów prawnych w celu zaostrzenia kar).
6. Odtwarzanie poprawnego stanu (dzięki zapewnieniu możliwości eliminacji skutków ataku, powrotu do stanu przed atakiem, odpowiednią archiwizację stanów i zdarzeń).

Te poziomy zabezpieczeń mogą być implementowane w różny sposób. W każdym przypadku powinny zostać wkomponowane w środowisko systemu, a ściślej w jego funkcje monitoringu i zarządania pracą.

4. Modele transakcji elektronicznych

Zmienne stosowane do pomiaru zaufania (koszt transakcji, historia transakcji, zabezpieczenie transakcji, wzory nabywanych usług, obciążenie zasobów systemowych, czas i miejsce transakcji) i działania podejmowane w celu jego wyznaczenia (weryfikacja zdolności kredytowych klienta lub wytwórcy, autoryzacja) stanowią podstawę do budowy modeli zaufania transakcji elektronicznych. Modele te oparte są na [6]:

- relacjach binarnych,
- systemach logiki rozmytej,
- automatach stanów transakcji,
- procesach transakcji.

4.1. Modele binarne

Dwie lub więcej zmiennych i parametrów może zostać użytych do wyznaczenia poziomu zaufania wybranej transakcji. Pomiędzy zmiennymi powinna istnieć oczywista zależność, aby dobrze zdefiniować model zaufania. Zależność najlepiej przedstawić za pomocą macierzy wpisując na przecięciu odpowiedniego wiersza i kolumny wymagane działania (patrz rys. 1) związane z wyznaczeniem zaufania (strefę zaufania oznaczono kolorem szarym). Wytwórcy skłaniają się do weryfikacji transakcji klientów ze słabą historią. Przez historię rozumie się wiedzę nagromadzoną na podstawie analizy dotychczasowych zachowań klienta (skala została dobrana arbitralnie). Dla klientów z dobrą historią prawdopodobieństwo weryfikacji rośnie wraz ze wzrostem wartości transakcji.

Historia	Bardzo mała	Mała	Srednia	Wysoka	Bardzo wysoka
Znakomita					W
Dobra				W	W
Normalna			W	W	W
Zła	W	W	W	W	W
Brak	W	W	W	W	W

Wartość pojedynczej transakcji

Oznaczenia:
W oznacza weryfikację transakcji

Rys. 1. Macierz zaufania ze strefą zaufania (kolor szary)
Fig. 1. Trust matrix showing trust zone (grey color)

4.2. Modele rozmyte

Historia	Bardzo mała	Mała	Srednia	Wysoka	Bardzo wysoka
Znakomita	W/30	W/40	W/50	W/70	W/50
Dobra	W/10	W/20	W/30	W/10	W/13
Normalna	W/5	W/10	W/10	W/7	W/2
Zła	W	2W	W	3W	2W
Brak	2W	4W	3W	5W	20W

Ilość przeprowadzanych transakcji

Oznaczenia:
W oznacza weryfikację transakcji

Rys. 2. Macierz zaufania z ważoną powierzchnią zaufania (kolor szary)
Fig. 2. Trust matrix showing weighted trust surface (grey color)

Różnią się od modeli binarnych tym, że wprowadzają zmienne wagowe w celu rozróżnienia stopnia pomiarów. Nie jest uzasadnione przeprowadzanie weryfikacji transakcji dla klienta z dobrą historią transakcji w tym samym stopniu, co dla klienta ze słabą historią nawet w wypadku transakcji o dużej wartości. Takie postępowanie formuje trójwymiarową powierzchnię ważonego zaufania (patrz rys. 2). Ustalenie progów zaufania i przypisanie wartościom macierzy pewnej gradacji (patrz rys. 3) umożliwia zastosowanie logiki rozmytej do analizy bezpieczeństwa.

Historia	Bardzo mała	Mała	Średnia	Wysoka	Bardzo wysoka
Znakomita	MW	NW	NW	BNW	BNW
Dobra	MW	NW	NW	NW	SW
Normalna	SW	MW	NW	SW	DW
Zła	SW	SW	SW	DW	WW
Brak	SW	DW	DW	WW	BWW

Ilość przeprowadzanych transakcji

Oznaczenia:
BWW oznacza bardzo wysoką częstotliwość weryfikacji
WW oznacza wysoką częstotliwość weryfikacji
DW oznacza dużą częstotliwość weryfikacji
SW oznacza średnią częstotliwość weryfikacji
MW oznacza małą częstotliwość weryfikacji
NW oznacza niską częstotliwość weryfikacji
BNW oznacza bardzo niską częstotliwość weryfikacji

Rys. 3. Macierz zaufania z ważoną powierzchnią zaufania dla systemu logiki rozmytej
Fig. 3. Trust matrix showing weighted trust surface for fuzzy logic

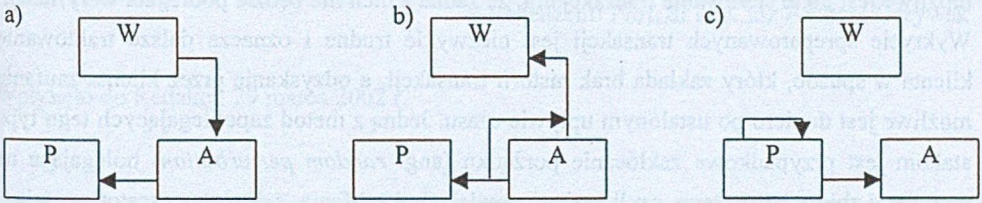
4.3. Modele stanów

Dozwolone stany transakcji to: zakończona pomyślnie, zakończona błędem, w trakcie wykonywania i w stanie ataku. Transakcja zostanie zakończona pomyślnie, jeżeli strona nadzorująca przebieg transakcji otrzyma pozytywne potwierdzenie od wszystkich stron biorących w niej udział (protokół uzgadniania). Nie oznacza to jednak, że stan transakcji nie może ulec zmianie. Strona nadzorująca przebieg transakcji może uznać, że naruszone zostało zaufanie i zmienić stan transakcji z zakończonej pomyślnie na stan ataku. Transakcja jest w trakcie wykonywania, jeżeli strona nadzorująca transakcję nie otrzymała pozytywnego potwierdzenia od przynajmniej jednej ze stron biorących w niej udział. Transakcja zostanie zakończona błędem, jeżeli strona nadzorująca nie otrzyma potwierdzeń od stron w niej

uczestniczących przed upływem z góry ustalonego czasu (może wówczas nastąpić ponowne wznowienie transakcji) lub potwierdzenia będą błędne (wówczas nastąpi zmiana stanu transakcji z zakończonej błędem na stan ataku). Przejście transakcji w stan ataku powoduje przełączenie bieżących operacji w tryb prewencji, co zabezpiecza przed wyrządzeniem dalszych zniszczeń. Tryb prewencji powoduje czasowe zawieszenie działalności stron biorących udział w transakcji. Jeżeli będzie to tylko możliwe, nastąpi próba likwidacji wyrządzonych szkód i ponowne wznowienie zawieszonych działań.

4.4. Modele procesów

Proces transakcji w najprostszym przypadku przebiega według schematu potwierdzenie tożsamości klienta, autoryzacja, dokonanie płatności i dostawa zamówionych przez klienta usług (ang. AAP – *the authenticate-first; then authorize, pay and deliver model*) – patrz rys. 4a. Możliwe są również inne schematy w zależności od wymaganego stopnia zaufania do transakcji. Pierwszy z nich polega na dogłębnej weryfikacji tożsamości klienta w wypadku naruszenia bezpieczeństwa (ang. ATV – *the authenticate-if-trust-violated model*) – patrz rys. 4b, drugi wymaga od klienta uiszczenia przedpłaty, która poprzedza dostawę zamówionych usług (ang. PF – *the pay-first model*) – patrz rys. 4c. Ten sposób transakcji jest przydatny w sytuacji, kiedy klient nie posiada historii transakcji lub pragnie zachować anonimowość.



Oznaczenia:
W oznacza poświadczenie transakcji
A oznacza autoryzację transakcji
P oznacza płatność za transakcję

Rys. 4. Modele zaufania AAP (a), ATV (b) i PF (c)
Fig. 4. AAP (a), ATV (b) and PF (c) trust models

4.5. Wykorzystanie modeli

Zastosowanie modeli zaufania do analizy ataków związanych z bezpieczeństwem transakcji elektronicznych pozwala na lepsze poznanie metod wykrywania i zapobiegania atakom oraz opracowanie metod przywracania stanu transakcji sprzed ataku. O ile metody

kryptograficzne chronią informacje przesyłane w Internecie, to są jednak bezużyteczne w przypadku kradzieży haseł lub numerów kart kredytowych. Nawet model PF wydaje się niewystarczający dla tego typu ataków, a wszystko, co pozostaje do zrobienia, to próba minimalizacji strat takiej transakcji. Modele zaufania mogą pomóc w zrozumieniu przebiegu możliwego ataku i minimalizacji strat, ale żaden z omawianych modeli nie jest w stanie zapobiec ani wykryć takiego ataku. Zamiast tego zalecane jest wykorzystanie metod:

- detekcji ataku na podstawie analizy profili zachowań klienta, wzorów nabywanych przez klienta usług,
- prewencji na podstawie opóźnienia przekazania klucza odszyfrowującego nabyte przez klienta usługi,
- likwidacji szkód wyrządzonych przez atak na podstawie zabezpieczenia transakcji przez firmę ubezpieczeniową.

Podstawową słabością modeli opartych na systemach logiki rozmytej jest ich podatność na ataki powodujące ujawnienie konturu (ang. *contour discovery attacks*), czyli ustalonego zasięgu powierzchni zaufania, wewnątrz której weryfikacja transakcji jest znikoma ze względu na wysoki poziom zaufania funkcjonującego w relacji klient – wytwórca. Informacje potrzebne do tego rodzaju ataku (cena zamówionej usługi, nazwisko klienta, stan transakcji) są zazwyczaj przesyłane poza protokołem bezpieczeństwa. Jeżeli dodatkowo skradziony został numer karty kredytowej, to przy wykorzystaniu technik płatności z góry lub nadpłaty możliwe jest takie sterowanie transakcjami, że żadna z nich nie będzie podlegała weryfikacji. Wykrycie spreparowanych transakcji jest niezwykle trudne i oznacza dalsze traktowanie klienta w sposób, który zakłada brak historii transakcji, a odzyskanie przez klienta zaufania możliwe jest dopiero po ustalonym upływie czasu. Jedną z metod zapobiegających tego typu atakom jest przypadkowe zakłócanie porządku (ang. *random perturbation*) polegające na modulacji zbioru rozmytego, czyli zasięgu powierzchni zaufania, z dowolną częstotliwością.

5. Podsumowanie

Bezpieczeństwo systemu rozproszonego można uzyskać poprzez wprowadzenie różnego typu mechanizmów ochrony począwszy od szyfrowania, a na sterowaniu przepływem danych skończywszy. Systematyczność takiego podejścia ma sens jedynie w wypadku integracji przedstawionych rozwiązań. W pracy ograniczono się do opisu poziomu modeli bezpieczeństwa, analizując podstawowe elementy krytyczne środowiska rozproszonego. Ma to zasadniczy wpływ zarówno na metodologie wytwarzania oprogramowania, jak i na

politykę bezpieczeństwa systemów. Jednakże należy stwierdzić, że jak na razie, istnieje ogromna luka pomiędzy teorią a praktyką bezpieczeństwa aplikacji rozproszonych.

LITERATURA

1. Roback E. A.: Guidelines to Federal Organizations on Security Assurance and Acquisition/Use of Tested/Evaluated Products. National Institute of Standards and Technology, 2000.
2. Noordergraaf A.: Building Secure N-tier Environments, Sun Blue Prints OnLine, 2000.
3. Ślesicki M., Krawczyk H.: Metody zabezpieczania aplikacji sieciowych Java, Zeszyty Naukowe Politechniki Śląskiej, Informatyka z.36, Gliwice 1999.
4. Geng X., Whinston A. B.: Defeating Distributed Denial of Service Attacks, IEEE IT Pro, No 4, 2000.
5. A Guide to Understanding Covert Channels Analysis of Trusted Systems, National Computer Security Center, 1993.
6. Manchala D. W.: E-Commerce Trust Metrics and Models, IEEE Internet Computing, No 3, 2000.

Recenzent: Prof. dr hab. inż Andrzej Grzywak

Wpłynęło do Redakcji 29 marca 2002 r.

Abstract

In the paper we consider security models for distributed applications. Common Criteria stated earlier are to evaluate products and classify them to their applicable Evaluation Assurance Levels. Safety rules for creating secure Java applets and aglets are to be used at common and not only for sophisticated products. Otherwise we may expect serious software bugs that may lead to behavior such as DDoS among others security risks.

When modeling trust it is wise to estimate it with several variables like cost, time, place, etc. and introduce trust actions. Upon that pillars models based on binary relations (see Fig. 1), fuzzy logic (see Fig. 2 and Fig. 3), transaction process (see Fig. 4) and transaction automaton are build using trust matrices. Prevention, detection and recovery techniques are best study cases for mentioned models.

N-tier architecture is designed to use features of a multi tiered environment to increase security. Although the proposed architecture addresses RAS (Reliability, Availability and Serviceability) issues, the primary focus is to build a secure environment. It is critical do design RAS features from the ground up to prevent compromising the security of the architecture. By implementing presented recommendations N-tier environments may improve their architectures and provide improved security.

Integration of several different security models is needed to create both, the application development strategies and security services in distributed information processing systems.