УДК 004.9

Ю. Рашкевич, Д. Пелешко, Н. Кустра, М. Пасєка
Національний університет "Львівська політехніка",
кафедра автоматизованих систем управління

СТРУКТУРА ОБ'ЄКТНИХ БРОКЕРІВ ОРГАНІЗАЦІЇ ДОСТУПУ

Запроновано ідеологію об'єктних брокерів доступу для систем з політикою груп, які б гарантували для будь-якого об'єкта одночасний і різноманітний доступ до різних сервісів в одному – термінальному чи віддаленому – сеансах сучасних інформаційних систем.

There is proposed technology of objective brokers for access to the system with policy group. Using of this brokers provides concurrent and different access to not the same services in the single, terminal or remote seances for the any objects.

Вступ

Усі об'єкти сучасних комп'ютерних систем (КС) потребують захисту. До таких об'єктів належить апаратне або програмне забезпечення (файли баз даних, семафори тощо). Одним із способів забезпечення захисту є процедура організації безпечного доступу до цих об'єктів.

Оскільки кількість об'єктів сучасних КС є великою, то з метою пришвидшення організації доступу використовується ідеологія політики захисту (від кого і які дані повинні захищатись) і механізму її реалізації. У деяких системах захист реалізується за допомогою ідеології монітора повідомлень [1]. Ідея цього монітора полягає в тому, що при спробі доступу до деякого ресурсу система спочатку "просить" монітор перевірити законність цього доступу. Монітор звертань переглядає таблиці політики і приймає рішення стосовно надання чи відмови в доступі.

1. Постановка задачі

Реалізація оточення, в якому працює монітор звертань, є різною. Серед найвживаніших розглядаються домени захисту, списки доступу, політики груп і ін. [1, 2]. Усіх їх об'єднує одна спільна риса – існування персоніфікованого запису, на основі якого приймається рішення про доступ. Зазвичай такий запис є унікальним для кожного користувача чи об'єкта, які запитують доступ, і має назву policy.

Основною метою статті є розроблення об'єктних брокерів доступу для реалізації ідеології multi policy в системах з політикою груп.

Для досягнення цієї мети завдання необхідно вирішити такі завдання:

- розробити систему персоніфікованого безпечного доступу одночасно до багатьох об'єктів;
- розробити систему об'єктних брокерів, які призначені для вирішення завдання надання чи відмови в доступі.

2. Структура користувацької системи прав

На рис. 1 наведено схему користувацької системи безпеки доступу до багатьох об'єктів (user multi policy, *UMP*), якою пропонується замінити існуючу (simple policy, *SP*). Уособленням *SP* в схемі *UMP* є *USP* (user simple policy). Відповідність *UMP* і *USP* верхнім індексом, наприклад, для наведеного на рис. 1 UMP^1, відповідає таблиця USP^1.

Фактично UMP^i є таблицею вказівників USP^i *ptr*[*j*] на об'єкти безпеки (Policy objects, *PO*), для "користувачів" із системним індексом *i*. Тут користувачами, окрім звичайних зареєстрованих в системі користувачів, можуть бути системні об'єкти, сервіси тощо.

Довжина таблиці USP^i є довільною, що дає можливість в одній системі мати багато об'єктів *SP* з різними правами доступу.

$PO^i j$ є фізично існуючими об’єктами безпеки. Саме вони реалізовують ідеологію доступу до інших об'єктів (надалі сервісів) в системі. При цьому $USP^i ptr[j]$ забезпечують лише доступ до цих об'єктів. Відповідність $USP^i ptr[j]$ і $PO^i j$ визначається індексом j.

Зазначимо, що кожен $PO^i j$ належить лише одному UMP^i. Більше того, самостійно $PO^i j$ також не може існувати. Відсутність реально існуючого $USP^i ptr[j]$ на нього є командою на знищення для менеджера системи безпеки.

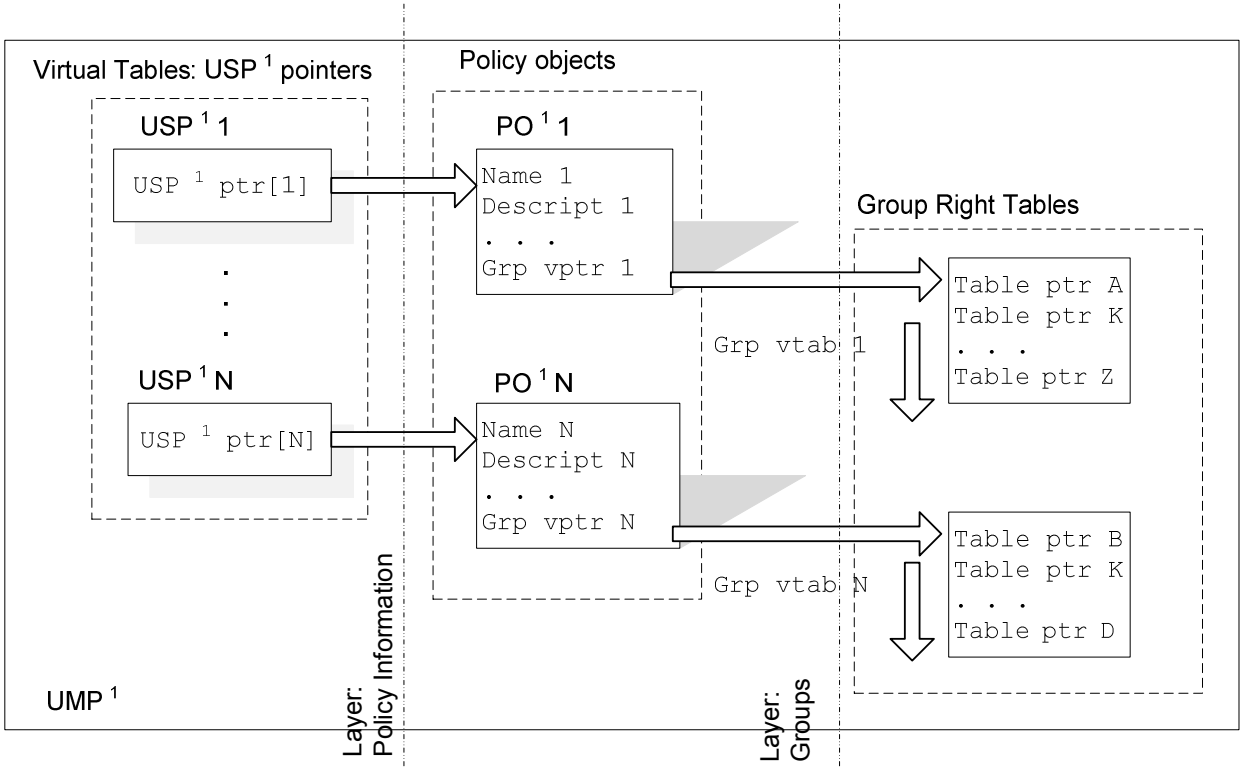

Рис. 1. Загальна структура UMP

Коментарі до рис.1:

- змінна і в позначенні $USP^i ptr[j]$ є системним індексом *UMP*.
- *USP* не можуть існувати окремо в системі, а лише у складі *UMP*.
- Доступ до окремих *USP* є довільним.
- Кількість *USP* в одному *UMP* є довільною (мається на увазі кількість вказівників $USP^i ptr[j]$).
- Зміна кількості *USP* або модифікація окремих *USP* в *UMP* можлива у двох випадках:

1. При формуванні нового policy і додаванні його до існуючого *UMP*.

2. При синхронізації екземплярів *UMP*, так званих мультиагентів аутентифікації (autentification multi agents, *AMA*), які існують при комунікаційних сеансах в інформаційних системах (information systems, *IS*).

- Структура $PO^i j$ є подібною до типового користувацького policy і має фіксований набір полів.
- Останнім полем об’єкта $PO^i j$ є вказівник $Grp\ ptr^i j$ на таблицю груп (Group Right Table), склад якої визначає права (permission) в системі для користувача з індексом *i*. Кожен елемент *Table ptr* цієї таблиці є вказівником на відповідну таблицю прав в системі. Зміщення в цій таблиці визначається вказівником $Grp\ vtab^i j$.

За цим рисунком належність до цієї групи прав визначається існуванням правила:

$$\forall UMP^i\ \exists USP^i:\ USP^i ptr[j] \rightarrow PO^i j \rightarrow \left[Grp\ ptr^i j:\ Grp\ vtab^i j\right].$$

При аутентифікації на основі *UMP* операційна система створює *AMA*. На цьому етапі *AMA* повинна забезпечувати потреби системних і користувацьких сервісів. Це здійснюється поступово з появою відповідних брокерів, які обслуговуватимуть запити авторизації. У випадку завантаження мережевих сервісів аутентифікований *AMA* повинен вирішити завдання синхронізації інформації з *USP* по активних станціях *IS*. З метою пришвидшення процесу завантаження ця синхронізація може бути відкладеною.

Зауважимо, що комунікація *AMA* можлива тільки з брокером авторизації.

3. Брокери авторизації

Брокер авторизації (autorization brokers, *AB*) є програмним об'єктом, який створюється системою автоматично у відповідь на запити аутентифікації від програмних сервісів. Зауважимо, що визначення категоризації *AB* дає можливість під'єднувати сервіс до існуючого брокера і не створювати у цій *IS* новий.

Основним призначенням *AB* є вирішення завдання авторизації. Це реалізується через комунікацію брокера з сервісом (чи сервісами), з одного боку, і комунікацію з *UMP* з іншого. Загальну структуру *AB* наведено на рис. 2.

Брокери авторизації повинні існувати двох типів:

1. Прості брокери (Simple autorisation broker, *SAB*)
2. Прозорі брокери (Transparency autorisation broker, *TAB*).

TAB інкапсулює *SBA* і додає до нього лише так званий Transparency layer (рис.2), який необхідний для вирішення завдань синхронізації.

Один *AB* може обслуговувати декілька сервісів. При цьому мережеві сервіси обслуговуються винятково *TAB*. Це зумовлено тим, що *TBA* може негайно чи згодом запустити процес синхронізації активного *UMP*-об'єкта стосовно усіх машин мережі. При цьому для кожного *UMP*, задіяного в процесі синхронізації, створюється власний об'єкт, який заноситься в пул *UMP*-об'єктів (рис. 3). Це є свідченням того, що об'єкти синхронізовані. Цей об'єкт створюється лише на машині, на якій активний об'єкт володіє іншим *UMP*.

Пул *UMP*-об'єктів дає змогу пришвидшити будь-які операції аутентифікації доступу сервісів до мережі.

Як показано на рис. 2, *AB* можна розбити на дві умовні частини:

1. Структурна частина (на рис. 2 позначена маркером *Spart*);
2. Функціональна частина (на рис. 2 позначена маркером *Fpart*);

Поля, які входять до *Spart*, виконують основне завдання *AB* – вирішення завдання авторизації. Поля функціональної частини містять дескриптори сторонніх об'єктів чи сервісів і використовуються лише для вирішення завдання синхронізації даних. Проте можуть використовуватись як канали обміну даними. Останній підхід є актуальним в тому випадку, коли об'єкти від сервісів відділяються прошарком брокерів. Такий підхід має свої переваги і недоліки, проте не є предметом дослідження у цій статті.

Розглянемо детальніше структуру *AB* по рівнях, які наведені на рис. 2.

Поле *Active USP Table pointer* є вказівником на віртуальну таблицю, яка містить авторизовані для кожного сервісу *USP*, які зазвичай належать різним *MUP*. В ідеальному випадку усі *USP* з таблиці *USP Table* відповідають записам пулу. Проте це необов'язково – оскільки пул керується окремо, то і вміст може відрізнятися.

Фактично в таблиці *Active USP Table* знаходиться лише один *USP* деякого *MUP*. Цей *USP* володіє найбільшими правами відносно цього сервісу.

Рівень *Transparency layer* існує лише в *TAB*. Поле цього рівня використовується лише для доступу до віддалених чи мережевих сервісів. Це зумовлено тим, що на віддаленій машині для авторизації може бути задіяний інший *USP*, відмінний від локального.

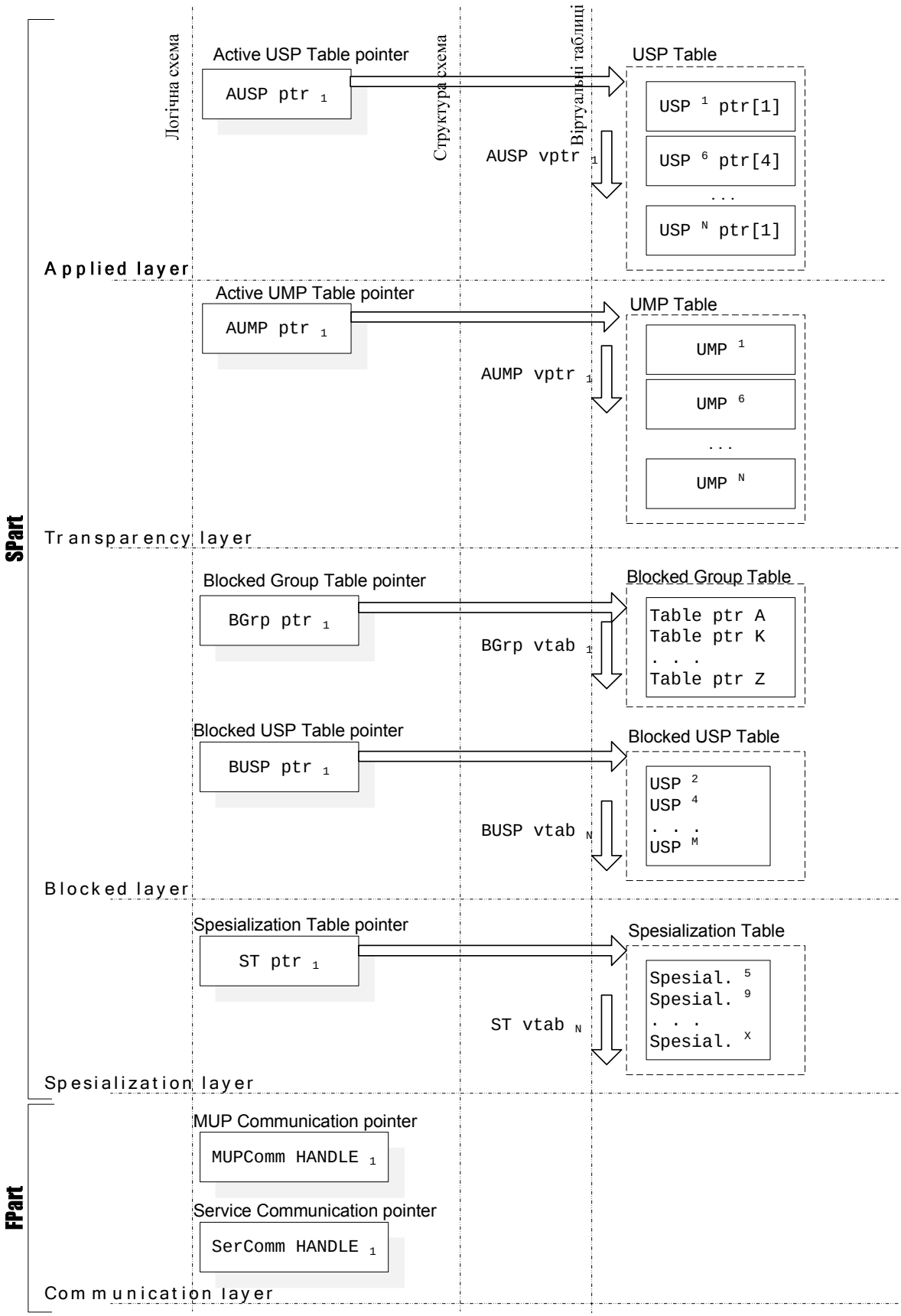

Рис. 2. Структурно-логічна схема брокера авторизації

Іншим призначенням цього рівня є *USP* з *MUP* різних машин. При цьому алгоритм синхронізації може бути різним. Наприклад, для *USP* з однаковими іменами інформація про права доступу поновлюється лише у бік збільшення прав.

TAB є мережевим брокером завдяки цільовому використанню *Transparency layer* стосовно мережевих інтерфейсів. Проте може використовуватися і стосовно локальних.

Рівень *Blocked layer* є набором з двох віртуальних таблиць, які забезпечують вирішення завдання авторизації за такими правилами:

- доступ для *MUP* вважається наданим, якщо не існує жодного *USP*, які входять до даного *MUP*, які по полю *Grp ptr* містили б заблоковані групи.
- жоден *USP* за полем *Name* не входить до таблиці заблокованих *USP*.

Цей підхід дає змогу швидко відмовити в авторизації, але не забезпечує доступу. Проте при організації доступу це є першою дією. Надалі, якщо *MUP* не входить до складу заблокованих, то переглядаються усі його *USP* для знаходження груп з правами стосовно цього сервісу. У випадку, коли такий *USP* (переглянуто, тобто знайдено групу), доступ до сервісу надається. Якщо таких *USP* є декілька, то вибирається *USP* з найбільшими правами доступу. При цьому *USP*, яке забезпечило авторизацію, заноситься в *Active USP Table*, а *MUP* – в *Active MUP Table* брокера, який вирішує завдання авторизації для цього сервісу.

Зазначимо, що один *TAB* чи *SAB* може обслуговувати декілька сервісів (рис. 3). При завантаженні одного брокера менеджер брокерів може динамічно керувати навантаженням авторизації, перекладаючи завдання авторизації їх з одних брокерів на інші (існуючі чи новостворені).

4. Схема організації доступу в IS

Схему функціонування брокерів авторизації в інформаційній системі наведено на рис. 3.

Існування прикладних об’єктів (на рис.3. позначені як safety objects, *SO*) обумовлює утворення так званого пулу *MUP*-об’єктів (*MUP Pool*). Мінімальна кількість *MUP*-об’єктів у пулі визначається кількістю об’єктів, які вимагають авторизації. Окрім того, в ньому можуть існувати *MUP* різноманітних мережевих чи віддалених сервісів. Тобто розмір пулу не менший за кількість прикладних об’єктів.

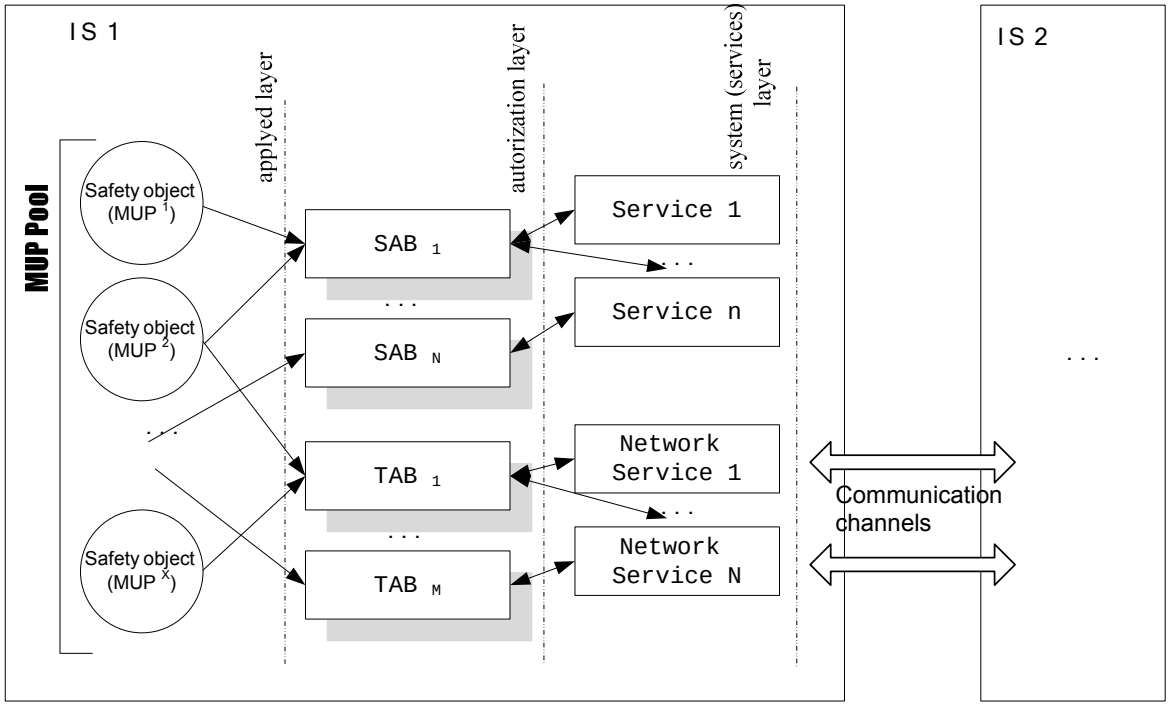

Рис. 3. Схема організації доступу всередині інформаційної системи та в мережевих сеансах

Доступ *SO* до сервісів здійснюється через прошарок *AB*. Це дає можливість строго відокремити питання надання доступу від прикладних завдань, для яких потрібна комунікація із сервісами. Більше того, запровадження на Communication Layer всередині брокера будь-яких тунельних чи канальних технологій дає змогу зробити брокерів прозорими. В цьому випадку брокери можуть ще й відігравати роль безпечних і контрольних інтерфейсів обміну даними між *SO* з одного боку та сервісами з іншого.

Як показано на рисунку, доступ до внутрішніх сервісів *IS* забезпечується *SAB*. У випадку доступу до мережевих сервісів, тобто там, де інформація про *MUP* потрібна надалі, треба використовувати *TAB*.

З іншого боку, якщо запитуваний сервіс потребує результатів роботи іншого, то для отримання доступу до останнього, можливо, також буде потрібний *MUP*. У цьому випадку також необхідно використовувати *TAB*.

Висновки

Основними перевагами *UMP* є:

- Можливість існування різних policy для однієї аутентифікації;
- Автоматичне оновлення *UMP* (реалізовується в результаті синхронізації між різними *AMA*);
- Можливість для одного *UMA* запускати сервіс (чи сервіси) під різними рахунками (accounts, які визначають права доступу).

Основними перевагами *AB* є:

1. Відділення логістики авторизації від прикладного рівня;
2. Уніфікація логістики авторизації для різних локальних сервісів і для віддалених сервісів;
3. Можливість синхронізації policy для одного користувача, які створені на різних машинах;
4. Можливість організовувати доступ до різних сервісів з різними правами доступу.

1. *Таненбаум Е. Современные операционные системы. – СПб., М.: Питер, 2002. – 1037 с.*
2. Русинович М., Соломон Д. Внутренее устройство Microsoft Windows: Windows Server 2003, Windows XP, Windows 2000. – СПб., М.: Питер, 2005. – 992 с.