SZAKDOLGOZAT

Búzás Beáta

Debrecen

2009.

Debreceni Egyetem

Informatikai Kar

Adatbázis biztonság

Témavezető:
Kollár Lajos
Egyetemi tanársegéd

Készítette:
Búzás Beáta
Informatika tanár

Debrecen

2009.

TARTALOMJEGYZÉK

Bevezetés

Szakdolgozatom célja, hogy áttekintést nyújtson a jelenlegi Oracle adatbázis-kezelő nyújtotta security opciókról.

A számítógépeken, adathordozókon tárolt információk védelme mára elengedhetetlenné vált. Gyakorlatilag nincs már olyan vállalat, intézmény, amelyik ne elektronikus formában tárolná adatait, üzleti titkait, vagy ügyfeleinek elérhetőségeit. Ezek az információk néha jelentéktelennek tűnhetnek de egy versenytárs számára akár kulcsfontosságúak is lehetnek. Gyakran lehet hallani, hogy milyen trükköket, cseleket vetnek be egy-egy információ megszerzésére, gondoljunk csak az adathalász email-ekre.

A rendszerek védelme fontos és igenis tenni kell érte. Nem szabad az emberek jóindulatára, becsületére bízni értékes információk sorsát, amik a vállalat stabilitását veszélyeztethetik. Annál is inkább hogy napjainkban ezt törvényi előírások is szabályozzák.

Az adatok tárolásának többféle módja használatos, de legjellemzőbb és nagymennyiségű adatok esetében általában egy adatbázist használunk.

Az információ az egyik legértékesebb dolog, így az adatbázis-kezelő rendszerek gyakori támadásoknak vannak kitéve, mivel nagy mennyiségű, kényes, titkos információt tárolnak. Valamint roppant elterjedtek, hiszen a dinamikus tartalmat szolgáltató webes alkalmazások alapjául is adatbázisok szolgálnak.

A támadásokat két fő csoportra oszthatjuk a szerint, hogy honnan érkeznek. Lehetnek külső illetve belső támadások. A belső támadások olyan személyektől érkeznek, akik valamilyen szinten hozzáférnek a tárolt adatokhoz. Míg a külső támadások hátterében olyan emberek állnak, akik egyébként nem férhetnének hozzá a tárolt adatokhoz.

Külső fenyegetést okozhat, ha közvetlenül kirakjuk az adatbázist az Internetre. Védhetjük az adatokat a támadások ellen, ha tűzfal mögé helyezzük, de ekkor még mindig nem lesznek biztonságban. Ekkor webalkalmazásokon keresztül nyújtunk adatokat. Ilyenkor

még mindig érkezhetnek támadások az alkalmazási rétegen keresztül. Illetve belülről.

Külső támadások közé tartozik például, a kártékony kódok jogosulatlan távoli futtatása. Az SQL utasítások könnyű megbabrálhatóságával sok SQL fejlesztő nincs tisztában, így ezért ezeket az utasításokat biztonságosnak feltételezik. És nem vizsgálják meg a beérkező utasítás kódot elég alaposan. Ezért az olyan alkalmazások esetében, ahol a felhasználótól származó adatokból és statikus paraméterekből állítanak össze SQL lekérdezéseket, közvetlen SQL utasítás befecskendezéssel a támadó a régi SQL utasításokat módosíthatja vagy újakat adhat hozzá. Így titkos információkhoz juthat, vagy akár rendszerszintű parancsokat is futtathat az adatbázis gazdagépén.

Az SQL befecskendezéshez hasonló az Oracle shell kódolás-dekódolás támadás is. A shell kód valójában egy gépi utasítás, amit a hekker küld a szervernek a puffer elárasztásával, vagy egyszerűen megszerzi a szerver irányítását. A hekker ehhez a támadáshoz talál egy helytelenül megírt kódrészletet, ami lehetővé teszi számára, hogy hamis utasításokat küldhessen és ezzel átvegye a szerver felett az irányítást. A támadások közel 68%-a PL/SQL feldolgozássorán az eljárásba beépített hosszú sztringek elküldését használja ki. Ezen funkción keresztül küldenek shell kódot az adatbázis szervernek. A puffer elárasztásával a támadó megkapja a stackre mutató pointert, ahova így már maga is képes utasításokat küldeni. Így már kiadhat olyan parancsot a hekker, hogy az ő kódját futtassa a gép. Ezen támadás ellen úgy próbálnak védekezni, hogy egyre újabb és változatosabb szűröket helyeznek a hekker és az adatbázis közzé, hogy kiszűrjék a lehetséges gépi kódokat. Ez főleg ott lehetséges ahol a program megengedi a hekkernek, hogy bármilyen érvényes karaktert használjon, dekódolja és végrehajtsa.

Az adatok titkosságát az ügyfél és a kiszolgáló között mozgó adatokat védi az SSL kapcsolati protokoll. De ha az adatbázisban tárolt adatokat nem védjük, titkosítással, akkor ha a támadó egyszer hozzáférést szerez az adatbázishoz, megkerülve a webszervert, akkor az ott tárolt adatok védtelenné válnak.

Egy másik támadási módszer lehet az, amikor is a támadok a belépési neveket és jelszavakat törik fel. Az adatbázis kezelők a jelszavakat hash-elt értékükkel tárolják. Ezen

függvényeket könnyű számolni, de nehéz visszafejteni. Azonban léteznek szótárak, ahol több millió hash-érték pár érhető el, ebből a gyakori módszerekkel előállított jelszavakhoz a szótár alapján könnyen visszafejthető a karaktersorozat.

Könnyű támadási lehetőséget nyújtottak korábbi Oracle verziókban (7,8,9), a bennük lévő alapértelmezett (default) jelszavak. Ez nyilván könnyű támadási lehetőséget adott, ha valaki nem változtatta meg ezeket a jelszavakat. Ezt a támadási rést az Oracle a 10*g* bevezetésével megszüntette, ettől a verziótól kezdve már nincsenek benne alapértelmezett jelszavak.

A belső támadás egy lehetséges módja, mikor egy felhasználó, aki jogosan fér hozzá adatokhoz visszaél ezen jogával. Mint például kilopja az adatot a cégtől, és másoknak is hozzáférhetővé teszi, esetleg pénzért eladja. Ezen támadás ellen nehéz védekezni, de amennyiben ezt egy átlagos felhasználó teszi, akkor nem tud sok adatot kijuttatni. Amennyiben ezt egy rendszergazda, vagy hasonló erős jogokkal rendelkező felhasználó teszi, akik nagy mennyiségű adathoz férnek hozzá, már nagyobb kárt tud okozni.

A belső támadások ellen védelmet nyújt, ha odafigyelünk pár dologra. Kevesebbet figyelünk arra, hogy mely emberek férjenek be a munkakörbe, többet arra, hogy az adatforrást megvédjük. Figyelni a DBA-k, sysadminok és egyéb legitim felhasználók támadásaira. Nem csak a sebezhetőségekre kell figyelni, hanem arra is, hogy láthassuk ki hogyan fér hozzá az alkalmazásunkhoz. Felülvizsgálatot tartani, és felelősségre vonhatóvá tenni a felhasználókat és az adminisztrátorokat is. Illetve fontos, hogy a rosszindulatú cselekedeteket azonosítani tudjuk.

Én a következőkben az Oracle adatbázisok védelmi technikáiról fogok írni egy adatbázis adminisztrátor szemszögéből nézve.

Alapvető biztonsági követelmények

Az utóbbi néhány évben fontossá vált, hogy ne csak a kormányzati, katonai adatok nagy biztonságáról gondoskodjanak, hanem ezt vállalati szinten is megköveteljék. Törvények, előírások, standardok születtek az adatvédelemre. Ezen előírások többsége megköveteli a hozzáférések korlátozását, auditálását, az adatmódosítások naplózását.

Csak említésként néhány példa ilyen előírásra (nem célom ezek bemutatása):

- Sarbanes – Oxley (SOX) (USA törvény de világszerte ismert)
- HIPAA
- FERPA
- ISO17799
- CERT/CC (jelenleg ez a legismertebb szabvány)

A törvényi szabályozások és szabványok előírnak néhány követelményt aminek meg kell felelnie minden adatbázisnak. Ez 3 fogalomba sűrítve az alábbit jelenti:

- Bizalmasság
- Integritás
- Rendelkezésre állás

Bizalmasság: Egyénenként csak az az információ legyen látható amire feltétlenül szükség van. Amit az illető személy láthat, használhat, esetleg módosíthat.

Szükség lehet az adatok kódolására az adatbázisban és a közvetítő csatornán ahhoz, hogy megvalósíthassuk:

- A kommunikáció bizalmasságát
- Az adatok és mentések biztonságos tárolását
- Az alkalmazások korlátozott hozzáférését
- A felhasználók autentikációját
- A hozzáférés korlátozását

Integritás: Egy biztonságos rendszer garantálja, hogy az általa szolgáltatott adat érvényes és hamisítatlan. Ez azt jelenti hogy az adat védve kell legyen a nem kívánt törléstől, módosítástól az adatbázisban és a felhasználóig való eljutás során is.

Mire kell vigyázni?

- a rendszerelemekhez és az adatbázis objektumokhoz, csak az férjen hozzá akinek erre engedélye van.
- Korlátozzuk az operációs rendszer szintű felhasználók jogait
- Az Oracle szolgáltatások adminisztrálása történjen egy külön erre a célra létrehozott felhasználóval(legyen ez egy oracle nevű user a továbbiakban)
- Az oracle operációs rendszer szintű felhasználó objektumaihoz más felhasználó ne férjen hozzá. (természetesen a root, vagy admin felhasználó kivétel)
- Használjunk megszorításokat (constraint), amivel kikényszeríthető az adatok érvényessége az adatbázisban. Szabályokat fogalmazhatunk meg melyek segítségével fenntarthatóak a más kialakult, vagy kialakítandó összefüggések az adatok között.
- Természetesen szükség van vírusvédelemre, hisz léteznek olyan vírusok amik az adatok módosítására korruptá válására.
- Végül, de nem utolsó sorban gondoskodni kell arról is hogy ne sérüljenek az adatok „utazás” közben sem.

Rendelkezésre állás: A biztonság és a rendelkezésre állás között felfedezhető egyfajta fordított arányosság. Ugyanis egy erős biztonsági komponens, szabály, intézkedés blokkolhatja azon felhasználók hozzáférését is akiket egyébként nem szeretnénk kizárni a rendszerből.

Tökéletesen védettnek nem mondható egy rendszer sem amihez hozzáférnek.

Ezért meg kell találni a köztes utat, és folyamatosan nyomon követni a biztonsági frissítéseket mind az adatbázis mind az operációs rendszer szintjén.

A biztonság alappillérei

- Autentikáció: Azonosítani kell ki van a túloldalon, ki próbál hozzáférni az adatokhoz. Ez gyakorlatilag egy felhasználó/jelszó páros. Különösen fontos ebben az esetben a jelszó erőssége. Alapértelmezett és egyszerű, vagy rövid jelszó használata nem ajánlott.

  Korábban nagy gondot jelentettek az Oracle által beállított alapértelmezett jelszavak. Amit kevésbé tapasztalt. DBA-k nem változtattak meg, így nyitott kaput hagytak az illetéktelen betolakodóknak. Mára ezt az Oracle már javította. Nem hagy alapértelmezett jelszavakat a rendszerben telepítés után.
  A jelszavak erősségéről kiegészítésként annyit, hogy nem ajánlott a szótár alapján kikereshető sem. Legyen benne szám, kis- és nagybetű és legalább 10 karakter hosszú legyen.

  Ezt még bővíthetjük, kiegészíthetjük biometrikus azonosításra alkalmas eszközzel, vagy token (Secure Key) kártyával, vagy használhatunk tanúsítványokat.
- Autorizáció: Ez az a folyamat amikor a már bejelentkezett, engedélyezett user jogokat kap az adatok eléréséhez, az általa elvégezhető adatbázis-műveletekhez.
- Elérés-szabályozás: Ezzel szabályozhatjuk az alkalmazás felhasználók jogait, hogy mely adatokat érhessék el. Elosztott rendszerek esetében kritikus szempont.
- Auditálás: más néven logolás, naplózás.
  Vannak olyan kivédhetetlen rizikók amik ellen csak a felhasználó becsülete véd. Pl. : 1.) egy banki tisztviselő számára nem lehet megtiltani, hogy utalni tudjon. De ez azt is jelenti hogy saját magának is tudna bármikor, bárki számlájáról utalni. Ez ellen használjuk a logolást, így ha megtörténik a visszaélés, megkereshető viszonylag egyszerűen az elkövető. 2.) Alvó számlák esetében nagy lehet a kísértés, hisz valószínűleg már nem használják. Ezért ezeket külön figyelik, és ha történik valamilyen változás, azt logolják.
  Az auditálással később részletesen fogok foglalkozni.

Biztonsági rizikófaktorok

A hozzáférés természetes velejárója a rizikó. Sokat olvashatunk nagy rendszerek ellen irányuló külső támadásokról amikor a bűnözők sikerrel jártak. De a szakértők szerint 80-90%-a a károkozóknak belső, autorizált alkalmazott.

Csoportosíthatjuk a biztonsági veszélyeket az alábbiak alapján:

Külső támadások: (internet, hálózatok kialakulása óta jelentős)

- hozzáférés, jogosultság nélküli felhasználók betörési kísérletei: Az eszközeik különfélék lehetnek arra, hogy magasabb jogosultságot kényszerítsenek ki. (gyenge jelszavak megtalálása, jelszó feltörő szoftverek, szoftver hibák, biztonsági rések megtalálása)

- Social Engineering: Kihasználva a jogosult felhasználók jóhiszeműségét, különféle módszerekkel megszerzik azok azonosítására szolgáló információkat. Példa erre a közelmúltban történt bankok elleni támadás amikor a bank nevében email-be kérték el az ügyfelek azonosítóit. Ehhez nem volt másra szükségük csak a bank honlapjára, ill. a netbank eléréséhez használt felület képére. Az álca megtévesztő és az apróbb jelek kivételével (mint pl. az URL) teljesen ugyanolyan mint az eredeti.

- TELNET, FTP használat de akár még a kuka is tartalmazhat fontos információt a betolakodónak.

Belső támadások:

Itt az elkövetők azok a felhasználóek akiknek van hozzáférése a rendszerhez. Megpróbálnak több jogra szert tenni, vagy egy külső támadásban lehetnek a hacker segítségére. Egy cég esetében számolni kell azzal is, hogy a konkurenciát segítheti belső dolgozó azzal, hogy kiadja az ügyfélkörét, esetleg belső használatú dokumentumok kerülhetnek ki.

A takarító személyzet is lehet potenciális veszélyforrás például egy nem megfelelően tárolt adatbázismentés ellopásában. (Javasolt a mentéseket egy távoli helyen, páncélszekrénybe

zárva tartani, hisz ez a legegyszerűbb módja annak hogy a teljes adatállományt megszerezzék,).

Szintén a belső támadások közzé sorolnám a szabotázst is. Amennyiben bekövetkezik egy katasztrófaterv nagy segítségünkre lehet a helyreállítás során.

A katasztrófatervnek tartalmaznia kell a visszaállításhoz szükséges minden információt olyan részletességgel, hogy ha azt egy idegen adatbázisgazda kezébe adják, az minden előzetes környezetismeret nélkül helyre tudja állítani a rendszert.

Biztonsági analízis

Egyetlen rendszer sem lehet 100%-os biztonságban. Való életben általában komplex szoftverrendszerekről van szó amit emberek hoznak létre, akik hibázhatnak. Meg kell vizsgálni, hogy az egyes veszélyforrások mennyire vannak jelen a rendszerünkben és amelyik nagy valószínűséggel támadás eszköze lehet arra megoldást kell találni. A biztonsági rések foldozását kezdjük mindig a legveszélyesebbel. Legyen cél az, hogy megfelelően komplex, nehéz legyen a bejutás, hogy ne érje meg feltörni.

Biztonsági szabályrendszer – Security Policy

A Security Policy kialakítása lényegében nem más mint meghatározni azokat a biztonsági szabályokat amiket elvárunk, majd jóváhagyatni a vezetőséggel, és biztosítani, hogy betartják őket. Ezzel kaptunk egy újabb eszközt az emberi hiba elleni védekezéshez.
A felhasználókat megfelelően tájékoztatni kell az érvényben lévő szabályokról, tudatni velük, hogy folyamatos monitorozás mellett figyelik ezek betartását.

Mire érdemes odafigyelni a szabályrendszer kialakításánál:

A legfontosabb biztonsági alapszabály a legkevesebb jogosultság elve.

- Csak a legszükségesebb szoftvereket, komponenseket telepítsük a gépre. Így csökkenthetők a karbantartási, frissítési munkák, a lehetséges biztonsági rések száma.
- Csak a használt szolgáltatásokat indítsuk el, így csökken a nyitott portok száma
- Csak azok a felhasználók kapjanak hozzáférést az adatbázishoz, és az operációs rendszerhez, akiknek feltétlenül fontos.
- Korlátozzuk a root és az adminisztrátor, SYS, SYSDBA felhasználói fiókok használatát.
- Adatbázis felhasználók jogainak korlátozása. Sok alkalmazás működik úgy (helytelenül), hogy a felhasználó az alkalmazás séma felhasználó névvel kapcsolódik az adatbázishoz és ennek nevében ad ki utasításokat. Az egyes felhasználók saját felhasználónévvel használják az alkalmazást (itt már egyfajta korlátozás és jogosultságadás megtörténhet) és az alkalmazáson keresztül is saját felhasználóneveikkel jussanak be az adatbázisba, ahol a jogaik már korlátozva vannak adatbázis szinten (pl kvóta, profil, stb), illetve auditálhatóak.

Fontos figyelembe venni olyan eseményeket is mint pl. alkalmazott kilépése esetén milyen intézkedéseket tegyünk, hogy már ne érhessen el semmilyen rendszert, szervert, gépet.(a felhasználói zárolva/törölve legyenek, VPN bejelentkezést ne lehessen végrehajtani, tokeneket, egyéb azonosítókat vegyük vissza). Ez a lépés gyakran elfelejtődik, de komoly kockázatot jelenthet.

A személyi tényezők mellett fontos technikai oldalról is biztosítani a sérthetetlenséget. Gondolok itt a víruskeresők frissítésére, tűzfalak pontos konfigurálására és a szoftverfrissítésekre is. Az egyes eszközök gyártói folyamatosan javítják a hibákat, riasztást küldenek egy-egy felfedezett gyenge pontról. Gondoskodni kell a frissítések telepítéséről, tehet a policynak tartalmaznia kell az esetleges tervezett leállások folyamatát.

Mi is a security policy?

- Szabályok halmaza (dokumentumhalmaz) amik:

✓ egy-egy területre specializáltak

✓ A vezetés jóváhagyása után szükségszerűen betartandó szabályok

✓ Emberi hiba elleni védekezés egy eszköze

✓ Ajánlásokat tesz a legnagyobb biztonság eléréséhez és ellenőrző listát ad a megvalósításhoz.

Néhány példa ilyen szabályokra teljesség igénye nélkül:

- nem lehet más felhasználónevét használni
- kötelező havonta jelszót váltani (Oracle-ben megvalósítható a kikényszerítése)
- nem írható fel jelszó, sem monitorra, sem füzetre, semmilyen más által elérhető helyre.
- Már meglévő keretrendszerbe miként és milyen feltételekkel kerülhetnek be új alkalmazások
- Új/kilépő alkalmazottak felhasználóinak kezelése
- Stb.

Általában a nagyobb cégeknek van egy a biztonságért felelő csoportja (IT SECURITY). Ajánlott hogy egy ilyen csoportban legyen jogász, informatikus, DBA, rendszergazda, operációs rendszer gazda.
A policy alakítása folyamatos kell hogy legyen. Egy-egy ismétlődő tevékenységet, újra és újra felül kell vizsgálni, hogy megfelel-e a kor technikai követelményeinek.

Az IT Security (Információ Biztonsági)csoport feladata még:

- a felhasználók oktatása a biztonsági szabályokra, tudatni kell velük hogy a rendszerbeli tevékenységeik naplózásra kerülnek.
- Figyeli a megjelenő javításokat, koordinálja ezek telepítését.

Összefoglalva az alábbi ábra mutatja a főbb területeket a biztonság megvalósításához:

Biztonsági megoldások

Az alap feladat az adatbázis, az operációs rendszer és a hálózat megerősítése biztonsági elemekkel.

Ahhoz, hogy a teljes rendszer védve legyen nagyvonalakban az alábbiakat tartsuk szem előtt:

- *Legyen védve maximálisan a támadásoktól*

- Használjuk az adatbázisban rendelkezésre álló eszközöket a felhasználók auditálására, autorizációjára és autentikációjára.
- OS szintes is alkalmazzuk a „*legkevesebb jog*” elvét.
- A hálózatot biztosítsuk, tűzfallal a külső támadások miatt, ugyanakkor valósítsuk meg hogy a belső hálózatról is csak a szükséges részek legyenek elérhetők az egyes felhasználóknak.

- *Az adatok sértetlenségét biztosításuk*

- Felhasználó Audittal
- Fine-grained audittal
- Privilegizált accountok auditálásával
- Hálózat titkosítással
- Constraint, trigger használattal
- Monitorozzuk az eseményeket

- *Hozzáférés korlátozás megvalósítás*

- VPD (Virtual Private Database) Application context + FGCA (Fine-grained access control.

- Label Security
- Audit

A jelenleg az Oracle által nyújtott eszközök a következőek:

- **Oracle Advanced Security Option**: az adatbázisban tárolt és a hálózaton úton levő adatok titkosítására

- **Oracle Database Vault**: Jogosulatlanul hozzáférni szándékozó belső munkatársaktól véd

- **Oracle Label Security**: Sor szintű hozzáférés szabályozással védi az adatokat

- **Oracle Secure Backup**: titkosított szalagos mentés az adatbázisról és a fájlrendszerekről.

Az adatbázisok belső biztonsága

Ebben a fejezetben alapvető biztonsági elemekkel, (mint például a felhasználó profile, jelszó lejáratás, jelszókezelés, szerepkör (role), az adatszótár védelme) szeretnék foglalkozni.

Az adatbázis adminisztrátor feladata, hogy meghatározza a felhasználó kezelési szabályokat, figyelembe véve az egyes felhasználók sajátosságait. Természetesen más szabályokat érdemes hozni egy alkalmazás felhasználó (akinek a tulajdonában vannak az alkalmazás objektumai) és egy felhasználó (aki ezeket az objektumokat csak használja közvetlen jogon, vagy az adott alkalmazásban szereplő felhasználón keresztül) között.

Az auditálás miatt fontos, hogy megköveteljük, a felhasználók csak a saját felhasználónevüket használhatják az adatbázisban, tilos az alkalmazás felhasználónévvel belépni.

SZEREPKÖRÖK (ROLE)

A szerepkör (role) jogosultságok vagy privilégiumok egy csoportját gyűjtő elem.

A szerepköröket azzal a céllal hozzák létre, hogy a felhasználók számára ne egyenként kelljen a különböző jogosultságokat megadni, hanem szerepkörük kijelölésével az adott feladatkör ellátáshoz szükséges privilégiumokat egyszerűen biztosíthassák részükre. Ezen kívül a szerepkörökön keresztül egyszerűen és gyorsan lehet a különböző felhasználói csoportok számára szükséges hozzáférési jogosultságokat módosítani.

Vannak beépített role-ok, vagy létrehozhatunk a *create role* utasítással aminek később mi magunk adhatjuk meg a jogosultságokat, ami mutathat közvetlenül egy objektumra vagy egy másik rolera..

A SYS és SYSTEM fiókok védelméről szeretném pár szóban írni. Ez a két fiók minden joggal rendelkezik, méghozzá admin jogkörrel, tehát ők tudnak jogokat osztani kezdetben. A SYS felhasználó as sysdba klauzulával tud bejelentkezni, és minden más olyan felhasználó

akinek SYSDBA jogköre így tud belépni és ezáltal hozzáfér a SYS fiókhoz. A SYSDBA jogkör tulajdonosai tudják csak az adatbázis leállítani/elindítani.
A SYS és SYSDBA fiók alapértelmezettként létrejön az adatbázis létrehozásának alkalmával, de ezek olyan fiókok amit zárolni tudunk, de eldobni nem (szükség van rájuk)
A másik nagyon fontos típusú felhasználó akihez DBA role van rendelve. Ők általában az adatbázis adminisztrátorok. A DBA jogkörnek a SYSTEM felhasználói fiókhoz van hozzáférése, de nincs SYSDBA jogosultsága.
Fontos még megemlíteni a SYSOPER jogkört. Annyiban különbözik a SYSDBA jogkörtől, hogy csak a PUBLIC sémának grantolt objektumokat tudja olvasni. Tehát adatokhoz nem fér hozzá.
A SYSDBA és SYSOPER jogkört grantolni kell de ez mégsem egy role.

A SYS, SYSTEM, SYSMAN, DBSNMP, MGMTVIEW olyan alapértelmezett felhasználók amiknek a létrehozás utáni alapértelmezett jelszavát meg kell változtatni, és célszerű megfelelő PROFILE-t beállítani.

PROFILE / JELSZÓKEZELÉS

A profile segítségével limitálhatjuk a felhasználók erőforrás-hozzáférését. Kernel és jelszó korlátozást tudunk benne beállítani, majd egy felhasználóhoz egy profilet kell rendelnünk. Alapértelmezett profilet (DEFAULT) kap az a felhasználó amelyiknél nem rendelkezünk külön profilról létrehozáskor.
Profilet a *create profile* paranccsal tudunk létrehozni, valamint meg kell adni a különböző beállítások értékeit. Ezeket részletesen az oracle dokumentációból elérhetjük vagy az alábbi linken:
http://download.oracle.com/docs/cd/B14117_01/server.101/b10743/security.htm#sthref2731

Egy példa:

```
CREATE PROFILE proba_profile LIMIT
  SESSIONS_PER_USER 4
  CPU_PER_SESSION UNLIMITED
  CPU_PER_CALL UNLIMITED
  CONNECT_TIME UNLIMITED
```

```
IDLE_TIME UNLIMITED
LOGICAL_READS_PER_SESSION UNLIMITED
LOGICAL_READS_PER_CALL UNLIMITED
COMPOSITE_LIMIT UNLIMITED
PRIVATE_SGA UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME UNLIMITED
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX UNLIMITED
PASSWORD_LOCK_TIME UNLIMITED
PASSWORD_GRACE_TIME UNLIMITED

PASSWORD_VERIFY_FUNCTION NULL;
```

A profile egyik legnagyobb előnye hogy használatával kikényszeríthetjük a jelszavak bizonyos időnkénti cseréjét és a jelszóerősségre vonatkozó követelményeket. (`PASSWORD_LIFE_TIME, PASSWORD_VERIFY_FUNCTION`). Az illetéktelen használattól is védi az adatbázist ha definiáljuk a `FAILED_LOGIN_ATTEMPTS paramétert`.

Az adatszótár védelme

Alapértelmezett beállításként a 07_DICTIONARY_ACCESSIBILITY = FALSE init.ora paraméter gondoskodik arról hogy azok a felhasználók akiknek ANY rendszer privilégiumuk van ne férhessenek hozzá az adatszótárhoz. Természetesen ez nem vonatkozik a DBA és SYYDBA jogkörökre.

Ha valakinek szüksége van rá hogy lássa az adatszótárt, akkor a select any dictionary joggal ez megoldható.

Auditálás

Auditálásra azért van szükség, mert az autorizált userek is kompromittálhatják az adatbázisunkat. Az auditálás során minden tevékenységet naplózhatunk ami az adatbázisban történik.

Oracle 10*g*-ben 3 beépített lehetőség van:

- Adatbázis audit
- Érték alapú audit
- FGA (Fine-grained Auditing)

Röviden a 3 lehetőségről:

- Az adatbázis audit alapvetően a ki- és belépések monitorozására szolgál. Itt nem adhatunk meg feltételeket, minden ki és belépést naplózni fog.
- Az érték alapú (Value-based) audit során a módosításokat tudjuk naplózni. Update/insert/delete utasításokat kap el, és nemcsak hogy ez a figyelt esemény megtörtént hanem azt is hogy milyen értéket mire módosított. Trigger segítségével DML utasítást lehet vele figyelni.
- Az FGA segítségével SQL utasításokat tudunk figyelni. Az előzőnél nem volt mód magát az utasítást is látni és select utasítást egyáltalán nem tud figyelni. Az FGA erre jó. Később erre bővebben visszatérek

Adatbázis Audit

Az auditálási tevékenységet az AUDIT_TRAIL paraméterrel tudjuk engedélyezni. Ennek 3 értéke lehet:

- NONE: nincs audit

- DB : engedélyezi az auditot és az audit rekordok az adatbázisban keletkeznek, az AUD$ nevű táblába. Ennek a tartalmát az alábbi nézeteken keresztül is elérhetjük:
  DBA_AUDIT_EXISTS
  DBA_AUDIT_OBJECT
  DBA_AUDIT_SESSION
  DBA_AUDIT_STATEMENT
  DBA_AUDIT_TRAIL
  DBA_OBJ_AUDIT_OPTS
  DBA_PRIV_AUDIT_OPTS
  DBA_STMT_AUDIT_OPTS

- OS : engedélyezi az auditot és a rekordok az operációs rendszer alatt jönnek létre. Unix esetén egy szöveges állományban (alapértelmezésként az /rdbms/audit alatt), Windows esetén az Event alrendszerben.

Az AUD$ tábla DBA jogkörrel törölhető, ezért ha ezt is szeretnénk elkerülni akkor érdemes az OS értéket beállítani.

Hogyan adhatunk meg audit opciókat?

- SQL utasítás auditálása: pl. AUDIT TABLE; -minden olyan utasítást auditálni fog amiben a „table” benne van.
- System jogosultságok auditálása: pl. AUDIT SELECT ANY TABLE, CREATE ANY TRIGGER; AUDIT SELECT ANY TABLE BY USERNAME; Ha az adott felhasználó él valamelyik megadott privilégiumával feljegyzésre kerül
- Objektum audit: pl. AUDIT UPDATE, DELETE ON TABLENAME BY ACCESS; Minden egyes update és delet logolásra kerül ami az adott táblán történt.

Hatékony adatbázis auditálást érhetünk el az alábbi beállításokkal(tapasztalat alapján):

```
AUDIT ALTER SYSTEM;
AUDIT CREATE SESSION;
AUDIT CREATE TABLESPACE;
AUDIT ALTER TABLESPACE;
AUDIT DROP TABLESPACE;
AUDIT CREATE USER;
AUDIT ALTER USER;
AUDIT DROP USER;
AUDIT CREATE TABLE;
AUDIT CREATE ANY TABLE;
AUDIT ALTER ANY TABLE;
AUDIT BACKUP ANY TABLE;
AUDIT DROP ANY TABLE;
AUDIT DELETE ANY TABLE;
AUDIT ALTER ANY INDEX;
AUDIT DROP ANY INDEX;
AUDIT DROP PUBLIC SYNONYM;
AUDIT DROP ANY VIEW;
AUDIT DROP ANY ROLE;
AUDIT GRANT ANY ROLE;
AUDIT ALTER ANY ROLE;
AUDIT ALTER DATABASE;
AUDIT CREATE ANY PROCEDURE;
AUDIT ALTER ANY PROCEDURE;
AUDIT DROP ANY PROCEDURE;
AUDIT CREATE TRIGGER;
AUDIT CREATE ANY TRIGGER;
AUDIT ALTER ANY TRIGGER;
AUDIT DROP ANY TRIGGER;
AUDIT ALTER PROFILE;
AUDIT GRANT ANY PRIVILEGE;
AUDIT TABLE;
AUDIT PROCEDURE;
AUDIT ALTER TABLE;
```

```
AUDIT GRANT TABLE;
AUDIT GRANT SEQUENCE;
AUDIT GRANT PROCEDURE;
AUDIT GRANT DIRECTORY;
AUDIT GRANT TYPE;
AUDIT SYSTEM GRANT;
AUDIT ROLE;
AUDIT INDEX;
AUDIT TABLE;
```

Mindent nem érdemes auditálnunk, mert nagyobb adatbázis vagy sok felhasználó esetén hatalmas adatmennyiség gyűlhet össze, aminek a kezelése, tárolása, karbantartása külön feladatot jelent. Pl.:az audit rekordok a system táblatérben keletkeznek, ha ez betelt senki nem fog tudni belépni az adatbázisba, hisz nem tudja logolni a belépést.

Hol tudjuk megtekinteni az audit opciókat? Az alábbi nézetekben:

ALL_DEF_AUDIT_OPTS : milyen audit opciók vannak bekapcsolva

DBA_STMT_AUDIT_OPTS: utasításra irányuló audit opciók

DBA_PRIV_AUDIT_OPTS: privilégiumokra irányuló audit opciók

DBA_OBJ_AUDIT_OPTS: Séma objektumokra irányuló audit opciók

SYSDBA, SYSOPER userek auditja

AUDIT_SYS_OPERATIONS paraméter további sysdba és sysoper által végzett műveletekek auditálását engedélyezi. Ezek a rekordok nem adatbázisba, hanem fájlba kerülnek. A fájl helyét az AUDIT_FILE_DEST paraméterrel tudjuk megadni általában unix környezeten $ORACLE_HOME/rdbms/audit könyvtár alatt található. Windows esetén Event Log.

A SYS (és minden SYSDBA, SYSOPER felhasználót) belépése után minden amit begépel auditálva lesz. A megoldás nem tökéletes, hiszen az operációs rendszer gazdája, vagy az oracle user jelszavát ismerő adminisztrátor törölheti ezt a fájlt.

Érték alapú audit (Value-Based Auditing)

Ez az opció kiegészíti az előzőekben tárgyalt adatbázis audit-ot azzal, hogy azt az értéket amit megváltoztattunk eltárolja. Triggerek segítségével oldható meg, hogy amikor egy user változtatást kér, a trigger létrehoz egy audit rekordot amivel egy időben megtörténik a kért update/delete/insert Ez a megoldás lassítja az adatbázis működését, hiszen plusz terheket ró rá. A trigger kódjában van mód a hangolására.

Fine-Grained Audit / FGA

Az Oracle10g adatbázisok a 9i-hez képest már nemcsak a select utasításokat tudják auditálni, hanem minden DML utasítást mint az insert, delet, update. Ennek a funkciónak a támogatására egy új eljárást vezettek be az add_policy eljárást a DBMS_FGA package-ben.

Alprogram	**Leírás**
ADD_POLICY Procedura	Létrehoz egy új audit szabályt
DISABLE_POLICY Procedura	Letílt egy udit szabályt
DROP_POLICY Procedura	Eldob egy szabályt
ENABLE_POLICY Procedura	Engedélyez egy audit szabályt

Egy példa a DBMS_FGA package alkalmazására:

```
begin
                                dbms_fga.add_policy                                 (
                                                object_schema=>'CLAIM_SCHEMA',
                                                        object_name=>'CLAIMS',
                                                    policy_name=>'LARGE_CLAIM',
                                                                audit_condition=>
                    'CLAIM_AMOUNT>500              OR              PAID_AMOUNT>500',
                                                                    audit_column=>
                                        'SSN,                                   PROC_CODE’,
                statement_types         =>          ‘SELECT,INSERT,UPDATE,DELETE’
                                                                                );
end;
/
```

A fenti kód egy bejegyzést ír az **fga_log$** táblába amikor a figyelt táblához select, insert,delete, update utasítással akarnak hozzáférni az audit_column és audit_condition beállított feltételei szerint.

Ez az új lehetőség kiválthatja szinte minden esetben a sorszintű triggerek használatát, jóval egyszerűbb és áttekinthetőbb sorszintű auditot tesz lehetővé. Szintén lehetőséget ad arra hogy ne csak egy új bejegyzést írjon, hanem email küldést is lehetővé tesz egy adott eseményről.

Hogy támogassa az új lehetőségeket, kicst megváltoztatták a **dba_audit_policies** viewt.

Most így néz ki Oracle 10g-ben:.

OBJECT_SCHEMA	NOT	NULL	VARCHAR2(30)
OBJECT_NAME	NOT	NULL	VARCHAR2(30)
POLICY_NAME	NOT	NULL	VARCHAR2(30)
POLICY_TEXT			VARCHAR2(4000)
POLICY_COLUMN			VARCHAR2(30)
PF_SCHEMA			VARCHAR2(30)
PF_PACKAGE			VARCHAR2(30)
PF_FUNCTION			VARCHAR2(30)
ENABLED			VARCHAR2(3)
SEL			VARCHAR2(3)
INS			VARCHAR2(3)
UPD			VARCHAR2(3)
DEL	VARCHAR2(3)		

Az utolsó négy oszlop új. Ez jelzi ennek a szabálynak a érvénybe lépési eseményét. YES vagy NO értéket kaphat. Például: ha egy szabályt Select-re hoztak létre akkor a SEL mező értéke Yes lesz.

Hasonlóképpen a **dba_fga_audit_trail** nézetben:

STATEMENT_TYPE	Az utasítás típusa – INSERT, UPDATE, DELETE,

	SELECT ami kiváltotta a szabály érvénybe lépését.
EXTENDED_TIMESTAMP	A bejegyzés időpecsétje
PROXY_SESSIONID	Ha a user egy alkalmazás és proxy useren keresztül kapcsolódott akkor ennek a munkamenetnek a SID-je(azonosítója) lesz ide bejegyezve.
GLOBAL_UID	Ha a user egy alkalmazás és proxy useren keresztül connectált akkor ennek a usernek a global ID-ja lesz ide bejegyezve
INSTANCE_NUMBER	A példány száma ha RAC-ot használunk. (RAC-Real Application Cluster)
OS_PROCESS	Az op. rendszerbeli folyamatazonosító
TRANSACTIONID	.A tranzakció azonosítója ha a utasítás része egy tranzakciónak
STATEMENTID	Az utasítás ID-ja.
ENTRYID	A bejegyzés ID-ja.

DBVault

A Database Vault (ODV) Oracle adatbázis biztonsági opciójaként telepíthető eszköz (nem önálló eszköz, része az adatbázis-kezelőnek), mely lehetőséget ad arra, hogy az adatbázisban tárolt adatok védve legyenek az adminisztrátori és egyéb erős jogosultsággal bíró felhasználók elől is. Eszköz arra, hogy szétválasszuk a feladatközöket.

A Database Vault az adatbázis kezelő magjának (kernel) része, így sokkal nagyobb biztonságot nyújt mint az egyéb PL/SQL-ként implementált, az adatbázisra épülő kódok.

Három kiemelt dologra használható:

- DBA kizárása az alkalmazásokból

- Az alkalmazásbeli felhasználó részleges kizárása az alkalmazásából, hogy ne tudjon pl. alter és drop utasítást futtatni csak ha a security admin jogot ad neki hozzá. Az alkalmazás userek kizárása más alkalmazásokból (mert nagyon sok programozó úgy írja meg az alkalmazást, hogy DBA jogokat ad az alkalmazás usernek, így az összes alkalmazás user mindent lát az adatbázisban és utólag már nehéz a korlátozásokat bevezetni)

- Időkorlátokat vezethetünk be arra vonatkozóan, hogy ki mikor férhet hozzá és milyen alkalmazáson keresztül az adatbázishoz.

Nagy előnye, hogy mindehhez nem kell megváltoztatni az alkalmazások kódját.

Database Vault fő komponensei

Az ODV fő összetevői:

Realms - Tartományok

Rules - Szabályok

Factors - Tényezők

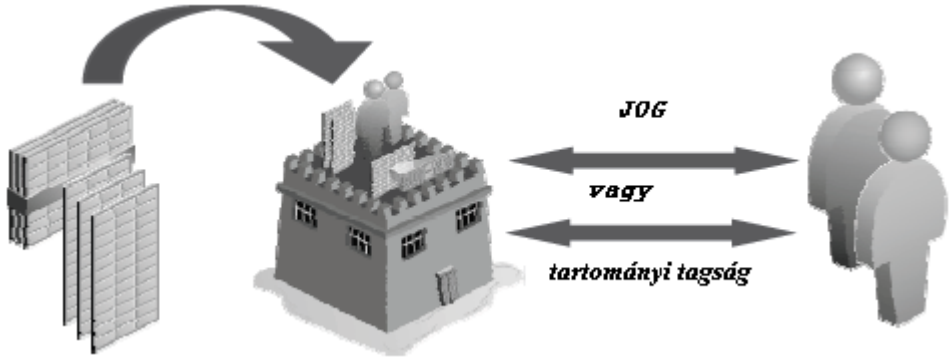
Identities – azonosítók

Rule Sets - szabályhalmazok

Command rules – utasítás szabályok

Secure application roles – biztonsági alkalmazás szerepkörök

Realms - Tartományok:



Ezekben foglaljuk/gyűjtjük össze a védendő objektumokat, mint táblák, szerepkörök, tárolt eljárások, csomagok. Tartalmazhat felhasználókat is. A tartományon belüli objektumok védettséget élveznek azon felszanálóktól akik pl. „ select any table „ jogukon keresztül érnék el. Csak akkor fogják tudni elérni ha a tartományba bekerülnek, vagy a tartományon belüli jogot megkapják az adott tábla elérésére. Amennyiben közvetlen jogon keresztül éri az adott objektumot úgy a tartomány nem nyújt védettséget. Ugyanúgy eléri mint mielőtt a tartomány létezett volna.

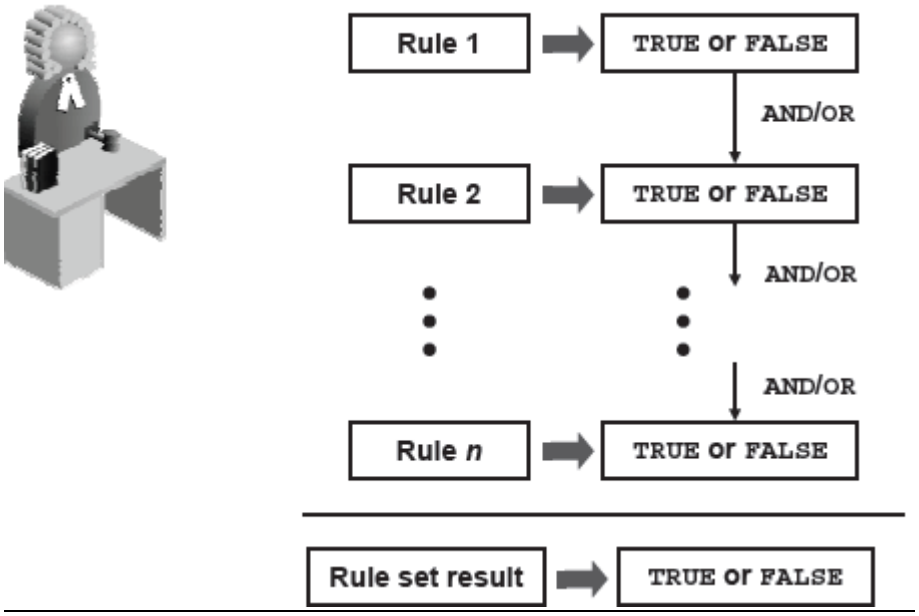
Factors - Tényezők:

A tényező egy nevesített változó, vagy attribútum ami az adott kapcsolathoz tartozik. Tartalmazhatja a kapcsolat azonosítására szolgáló IP címet, host nevet, stb. A faktorok hoz tartozó érték az azonosító (identity)

Identities – azonosítók:

Az azonosítók egy értéket jelentenek. Ez van egy faktorhoz rendelve. Egy faktor azonosítója a nap folyamán változhat a hely idő és alkalmazás függvényében amelyen keresztül a felhasználó kapcsolódik. Pl. TESZT1 felhasználó a munkahelyéről kapcsolódik a rendszerhez a DOMAIN faktor értéke INTANET. Ha otthonról akkor INTERNET. A szintje lehet TRUSTED és UNTRASTED. Ezzel megadható hogy otthonról ne lehessen csatlakozni, csak akkor ha az INTERNET faktor TRUSTED szintű.

Rule Sets - szabályhalmazok:



Egy vagy több szabály összefoglalva. Hozzárendelhető tartományokhoz, faktorokhoz, utasítás szabályokhoz. Ezek a szabályhalmazok kiértékelődnek. Igaz/hamis éréket kaphatnak a bennük található szabályok kiértékelésétől függően. 2 féle szabály alkalmazható a kiértékelés során (mind igaz, vagy bármely igaz).

Leírhatjuk velük az alábbi tevékenységeket:

- mikor legyen érvényes egy utasítás szabály
- engedélyezni egy biztonsági alkalmazás role-t
- meghatározni mikor rendeljük az egyes azonosítókat a faktorokhoz

Command rules – utasítás szabályok:

Meghatározzák azokat a szabályokat amik IGAZ értéke esetén egy adatbázis objectum módosítható.

„hajts végre ezt az adott utasítást x felhasználó y objektumán, ha a szabályok halmaza igaz”

Secure application roles – biztonsági alkalmazás szerepkörök:

Arra használható a felhasználóhoz rendelt szerepkör adta jogosultságok egy plusz feltétel megléte esetén legyenek csak valóbban használhatóak.

Telepítés

Az alábbiakban Windows operációs rendszeren fogom bemutatni a telepítés lépéseit, Az ORCL nevű adatbázisba.

Az alábbi 2 termékre lesz szükségünk:

- Database 10*g* Enterprise Edition

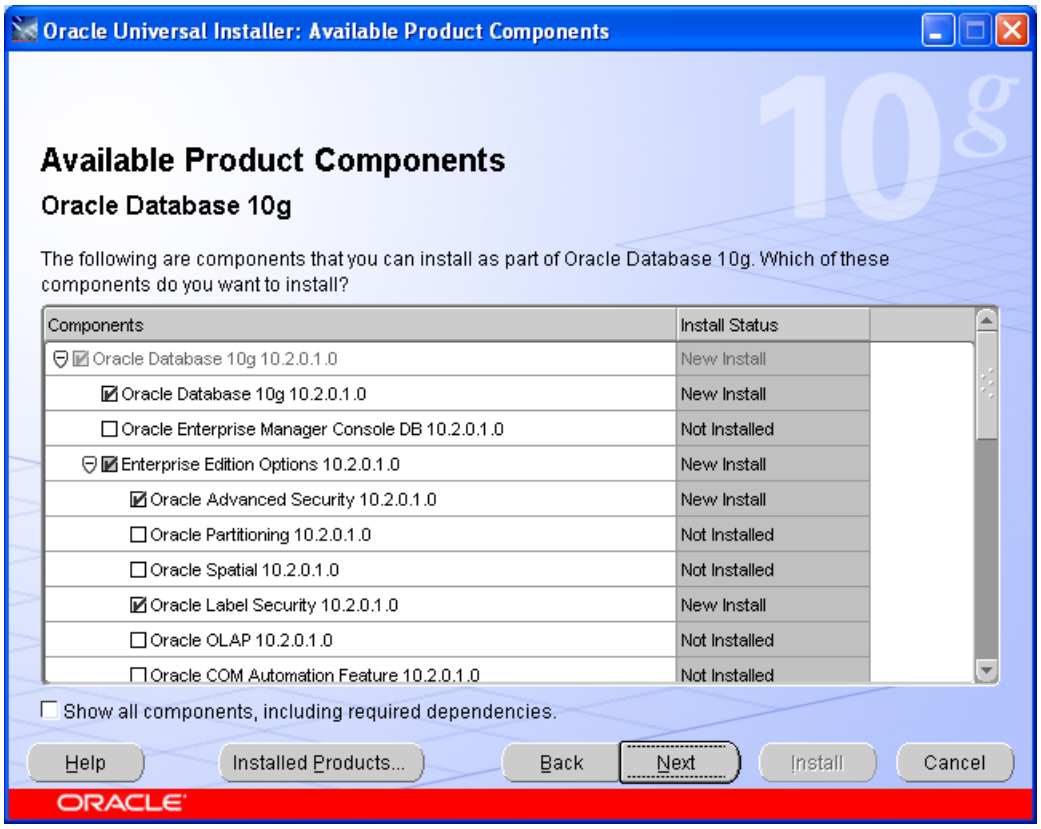
- Oracle Database Vault 10g Release 2 (10.2.0.3.0) for Windows (32-bit)

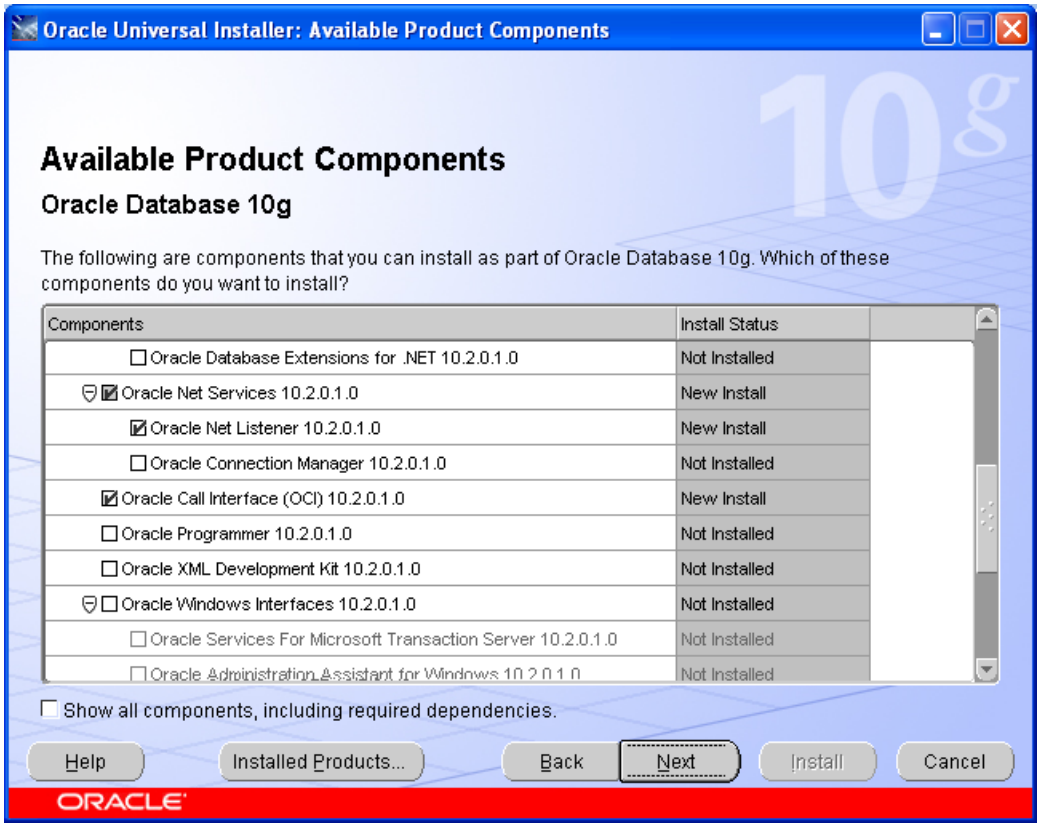
Mindkét termék letölthető az Oracle hivatalos honlapjáról az alábbi linken: *http://www.oracle.com/technology/software/index.html*

Az adatbázis részletes telepítési leírását mellőzöm, csak az ODV-ot fogom a továbbiakban kifejteni.
Az adatbázisnak amihez a Database Vault-ot szeretnénk telepíteni tartalmaznia kell a Label Security-t. Ha ez eddig nem volt része akkor mindenképp telepítenünk kell hozzá. Érzékeny a program arra ha nem megfelelő (azaz a Database Vault verziószámával azonos) verziószámú adatbázba próbálnánk telepíteni, ezért célszerű odafigyelni erre letöltés előtt.

Tehát az alábbi adatbázis komponensek lesznek fontosak:

- Database 10g
- Enterprise Edition
- Label Security
- Net Services





A telepítésről készül egy logfile amiben később nyomonkövethetjük a történteket:

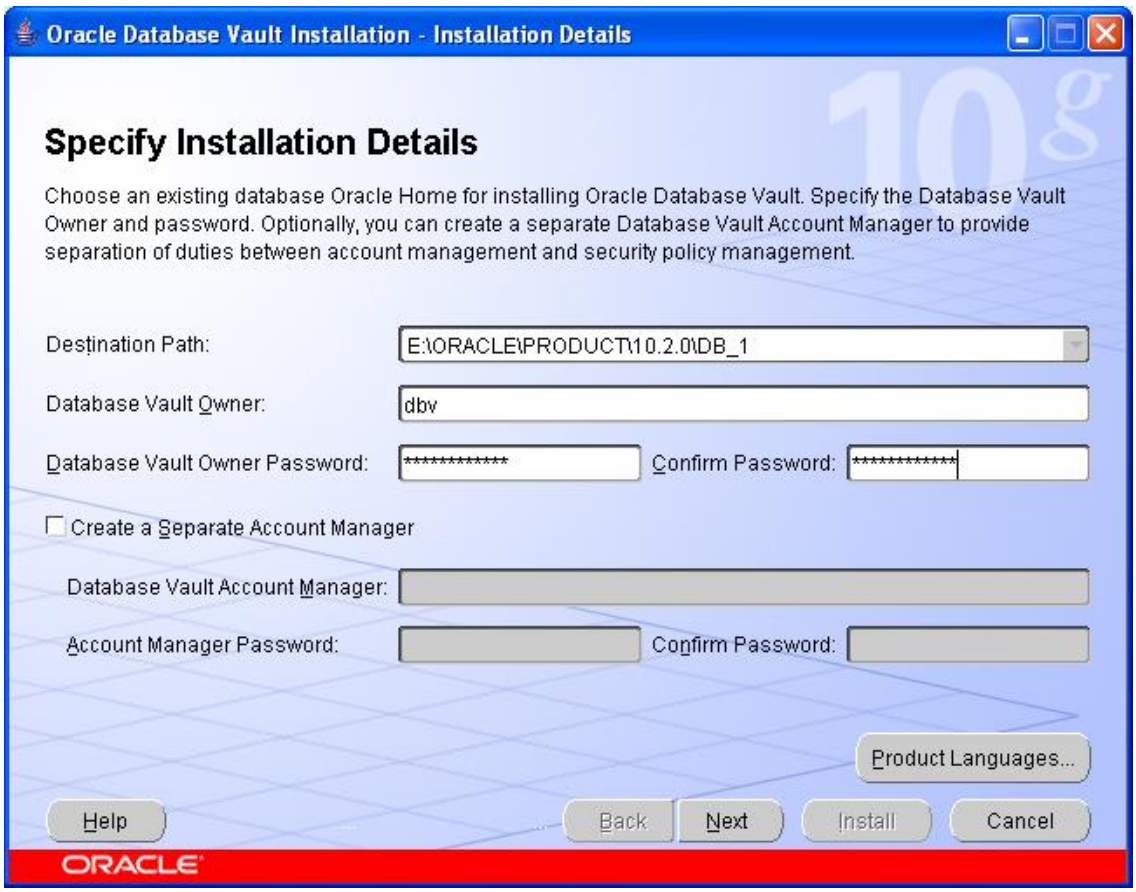
C:\Program Files\Oracle\Inventory\logs\installActions2009-05-06_07-21-49PM.log

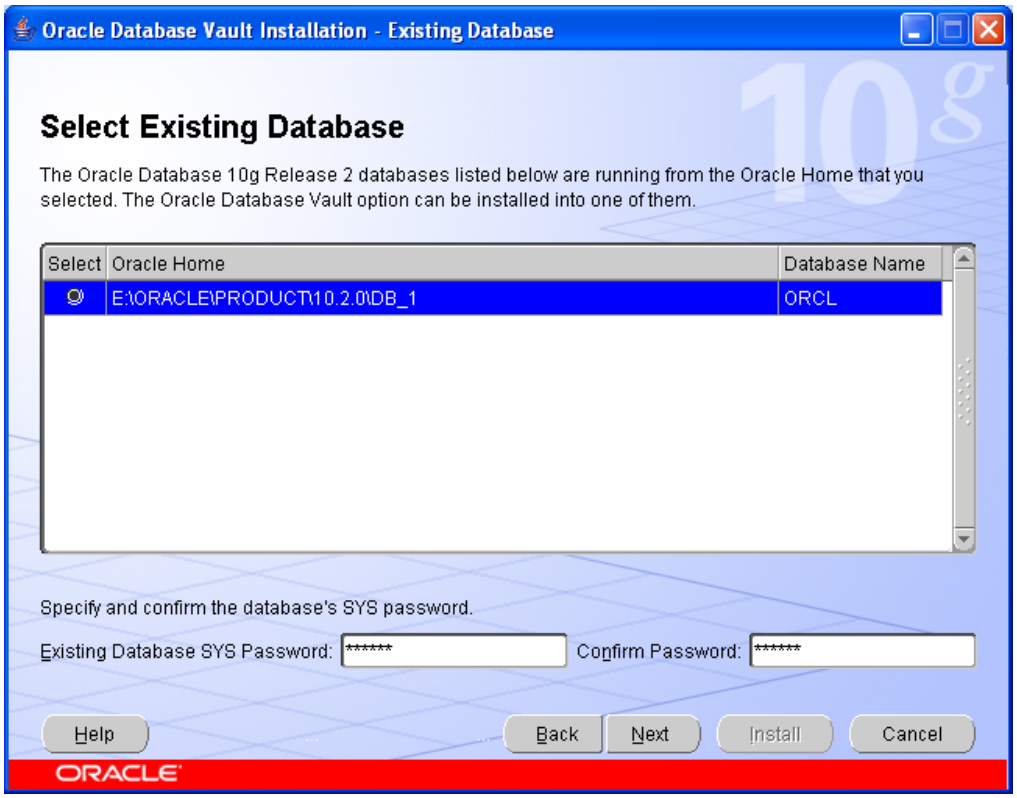
Mielőtt a Database Vault telepítését indítanánk el le kell állítani az adatbázist, a listenert és az Enterprise Manager Controlt.

A telepítés a setup.exe programmal indul.

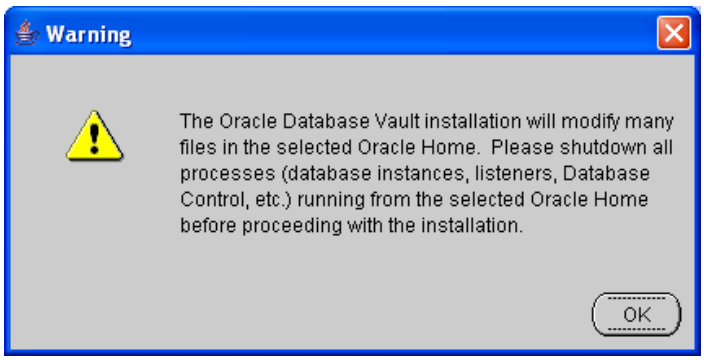
Meg kell adni Database Vault Owner azonosítóját és jelszavát, majd meg kell adni melyik létező adatbázisba szeretnénk telepíteni. Ezekután megkérdezi a SYS felhasználó jelszavát.

A telepítés során előfordult, hogy nem találta a jelszavakat „elég" komlpexnek a rendszer. Csak olyan jelszót fogad el ami legalább 10 karakter, tartalmaz számot, betűt és valamilyen írásjelet pl. _-t.





Módosítani fog az Oracle Home-ban néhány fájlt, ezért rákérdez, hogy megteheti-e. Természetesen OK-t kell mondanunk rá hogy továbblépjen.



Ebben a telepítési pontban megtörténik néhány (pontosan 7) új szerepkör kialakítása az adatbázisban.

DV_ kezdetű nevük megkülönbözteti őket a többi alapértelmezett adatbázis szerepkörtől.

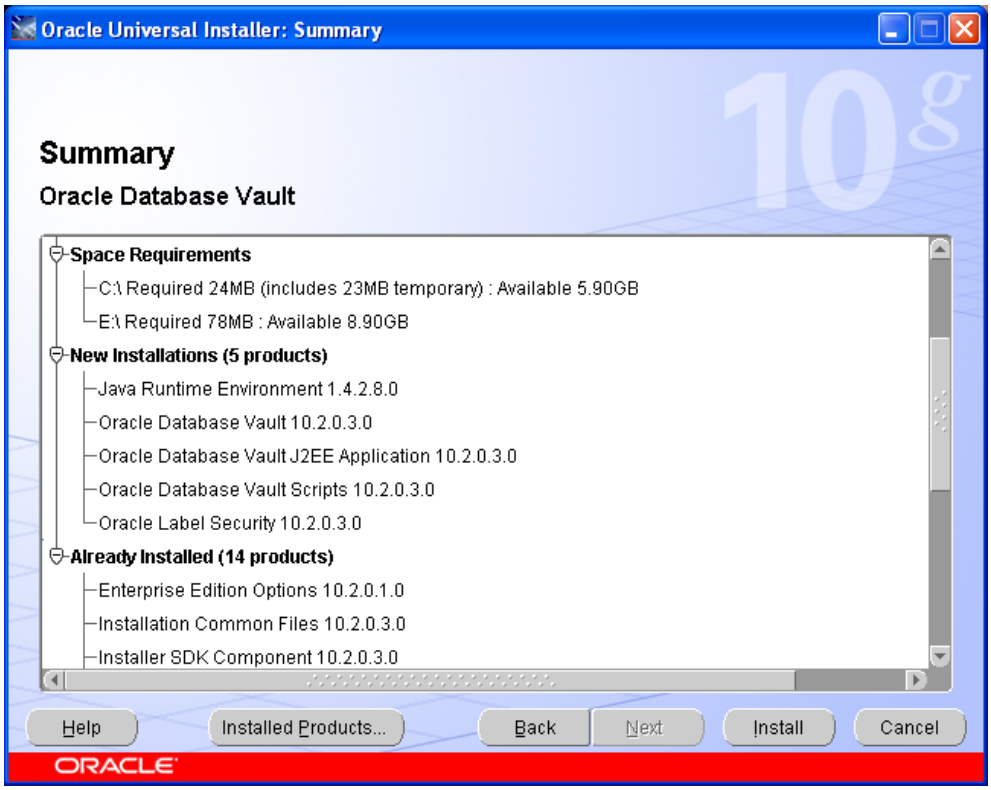
- DV_PUBLIC
- DV_OWNER
- DV_ADMIN

- DV_SECANALYST
- DV_ACCTMGR
- DV_REALM_RESOURCE
- DV_REALM_OWNER

Ugyanebben a lépésben léterjön 2 új felhasználó az adatbázisban:

- DVSYS (ODV nézetek és a ODV eljárások tulajdonosa)
- DVF (Faktor definíciók tulajdonosa)

Az Összefoglalóban jól látszik az az 5 új termék amit az adatbázismotor mellé feltelepít:



Az utolsó képernyőről visszajelzés a telepítés sikerességéről.

Az adatbázist és a listener-t a telepítő visszaindítja, nem kell külön elindítani.

A Database Vault Adminisztrátori felület (DVA)

A Database Vault konfigurálását legkönnyebben a Database Vault Adminisztrator-on keresztül tudjuk megtenni. Ez nem része az Enterprise Manaeger-nek, de ugyanazon a port-on érhető el. Belépéshez a telepítésnél megadott Database Vault Owner azonosítójára és jelszavát kell megadni.

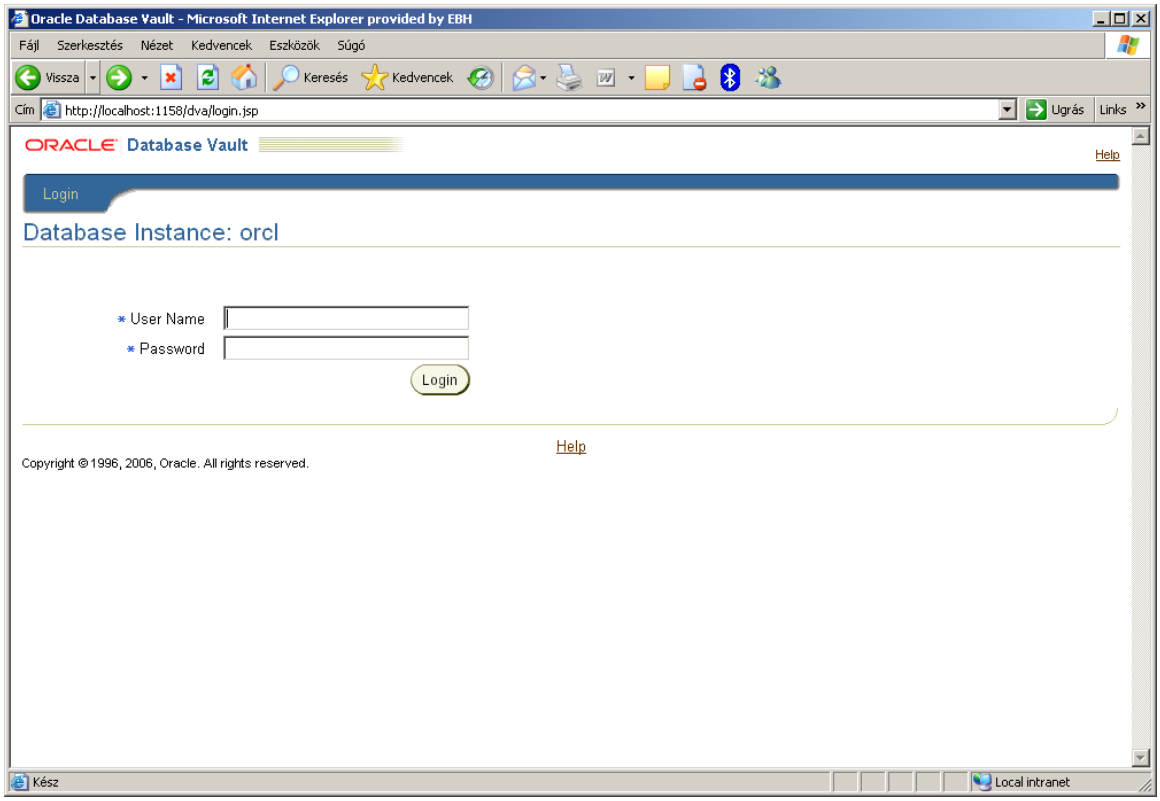
Az elérhetősége:

http://gépnév:port/dva

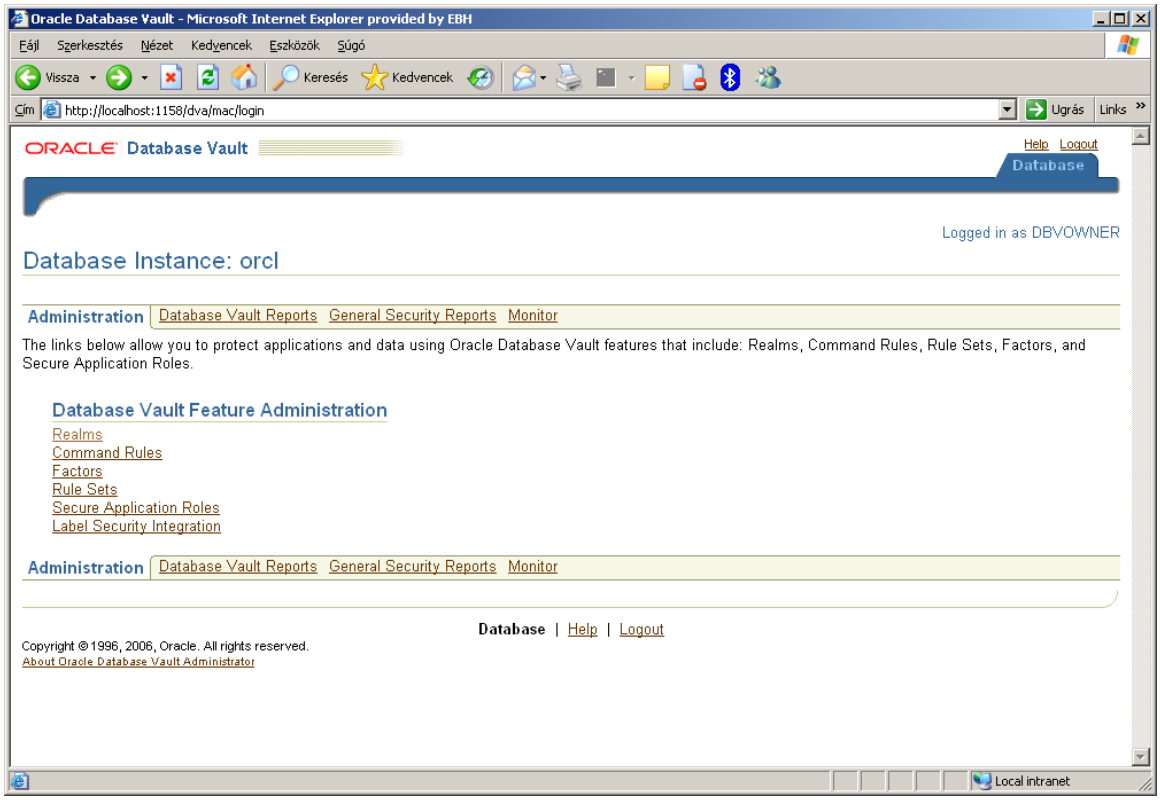
Alapesetben a link megegyezik az Enerprise Manager Control linkjével, csak /EM helyett /DVA szerepel.

Ebben az esetben: *http://localhost:1158/dva*

A bejelentkezési képernyő:


Oracle Database Vault - Microsoft Internet Explorer provided by EBH
Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó
Vissza
Keresés
Kedvencek
Cím
http://localhost:1158/dva/login.jsp
Ugrás
Links
ORACLE Database Vault
Help
Login
Database Instance: orcl
* User Name
* Password
Login
Help
Copyright © 1996, 2006, Oracle. All rights reserved.
Kész
Local intranet

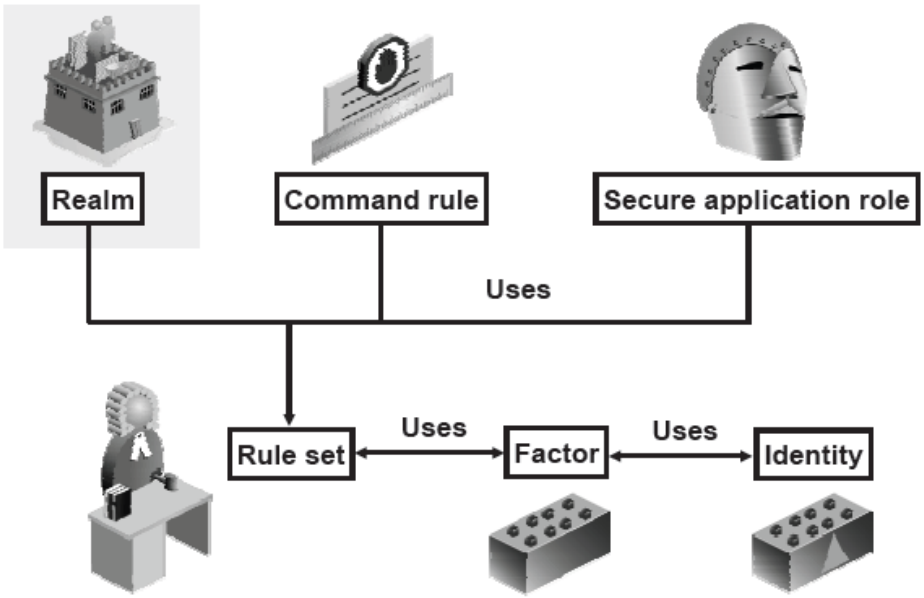
És a főmenü:



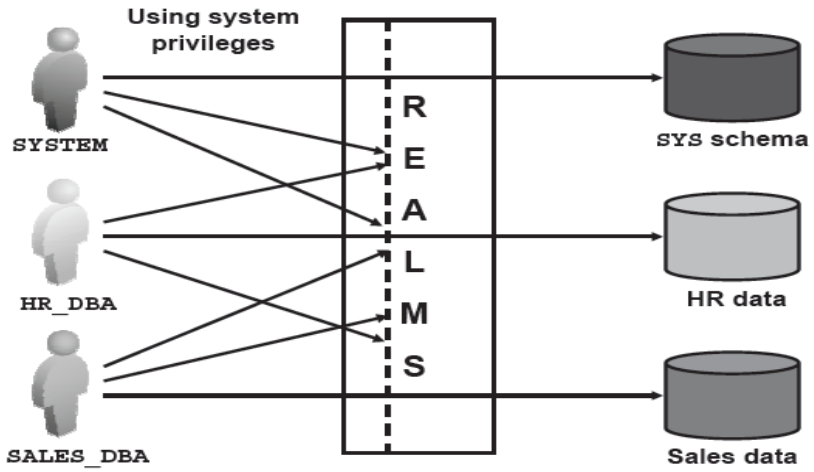
A továbbiakban a konfigurációról lesz szó, melyet ezen a felületen keresztül mutatok be.

Realm – Tartomány konfiguráció

Az alábbi ábrán a komponensek közötti összefüggések láthatók.



A tartomány használja a szabály halmazt, amely segít meghatározni, hogy milyen legyen a tartomány viselkedése egy adott konfigurációban.
A tartomány védelmet nyújt a benne foglalt objektumoknak, hogy a rendszerint DBA joggal rendelkező felhasználók ne érhessék el ezeket, kivéve ha a tartomány tagjaivá válnak.
Mindenki gyakorlatilag azt látja amit látnia kell.



Amennyiben a system felhasználó nem tagja sem a HR, sem a SALES objektumait védő tartománynak és nincs közvetlen hozzáférése hozzájuk akkor nem fogja látni a HR vagy SALES egyik objektumát sem.
Közvetlen hozzáférésen a SELECT on HR.xy to system utasítással kiadott jogokat értem. A SELECT ANY TABLE ez esetben nem jelent közvetlen jogosultságot, és a tartomány megvédi objektumait.
A HR séma DBA jogkörrel rendelkező felhasználója pedig más jogosultság hiányában csak a saját HR tartományának az objektumait fogja elérni.
Abban az esetben ha a HR felhasználó szeretné elérni saját objektumait szüksége lesz rá hogy a tartomány tagja legyen. Ellenkező esetben ő maga sem fogja ezeket látni.

Létrehozás:

Mielőtt létrehoznánk gondoljuk végig milyen objektumokat szeretnék egy tartományba zárni és melyek azok a felhasználók akiknek el kell tudni érni őket.
Néhány irányelv :

- sémákat, szerepköröket, amik egy alkalmazáshoz tartoznak célszerű egy tartományba zárni.
- A szerepköröket úgy kell meghatározni, hogy a lehető legkevesebb jogosultságot tartalmazza ami az alkalmazás eléréséhez kell.
- Egy adatbázis objektumot több tartományba is felvehetünk. Így lehetőség van arra, hogy korlátozzuk egy-egy felhasználó jogait egy sémán belül. Pl. hozzunk létre egy új tartományt amiben csak az az objektum és minimálisan szükséges jog szerepel amit el kell tudnia érni.
- El kell döntenünk hogy milyen auditálást fogunk beállítani, ugyanis az auditálási folyamat plusz terhelést eredményez a rendszernek és tárterület igényes. A DML és DDL utasítások nem fognak mérhető lassulást okozni. Audit always opciót érdemes nagyon megfontoltan kezelni.

Létrehozáskor az alábbiakat kell majd megadnunk:

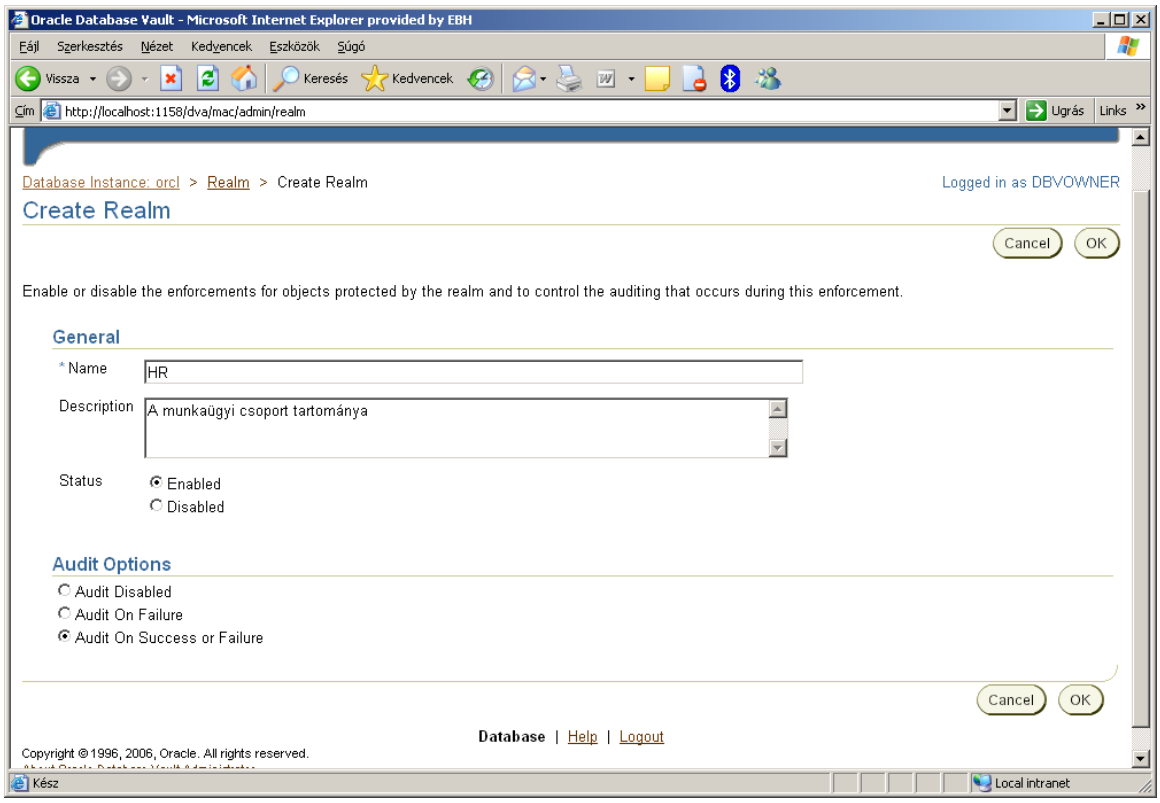
- a tartomány neve és leírása

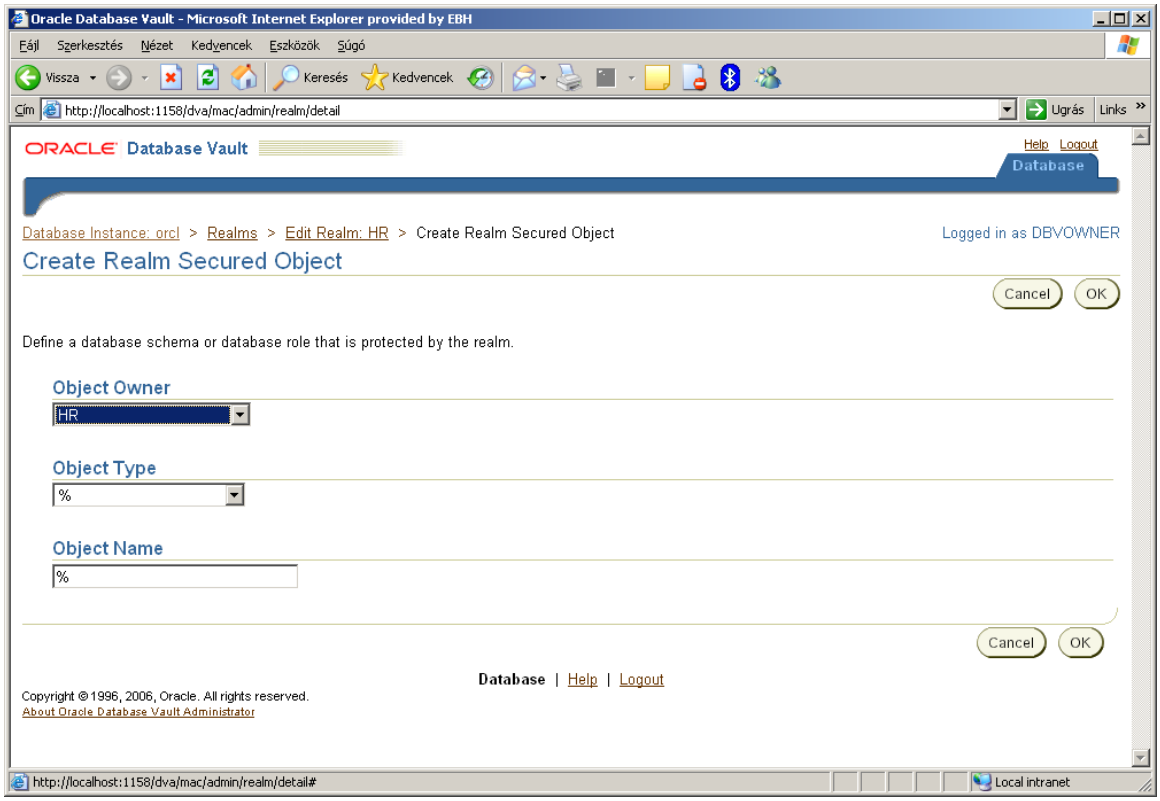
- a státusza : enabled(engedélyezett)/disabled(tíltott)
- az audit opciók
- a védendő objektumok listája, tulajdonossal, típussal, névvel megjelölve
- az engedélyezett felhasználók/szerepkörök listája

Objektumot csak azután lehet tartományhoz rendelni miután létrehoztuk a tartományt.

Az alábbi példában a HR tartományt hozom létre melyhez a HR séma összes objektumát hozzáadom:

- a DVA főmenüjében kattintsunk a REALM linkre, ezzel eljutottunk a tartományok adminisztrációs fülére.
- Create gombra kattintva hozhatunk létre új tartományt





- A realm secured objects opciónál válasszuk ki a HR sémát és a típus, object name mezőkbe a % jel (azaz minden típus és minden név) szerepeljen.
- A Realm Authorizations opciónál válasszuk ki a felhasználókat akiknek hozzáférést biztosítunk a HR séma objektumaihoz. Jelen esetben ez csak a HR felhasználó lesz.
  A hozzáférés típusa lehet
  - Participant/résztvevő: jogosult lesz a tartomány által védett adtokhoz hozzáférni
  - Owner/tulajdonos: az előzőeken felül joga van jogot adni másoknak a tartományban.
- Authorization and rule sets: a szabályokat itt határozzuk meg, hogy a tartománybeli felhasználók mikor vagy hogyan férhetnek hozzá az adatokhoz.

Egy tartományon belül Role/szerepkört is védhetünk. Tegyük fel a HR felhasználó létrehoz egy hr_role nevű szerepkört. Ezt a DVO (Database Vault Owner) a HR tartományba felveszi és tulajdonosaként a HR-t jelöli meg.

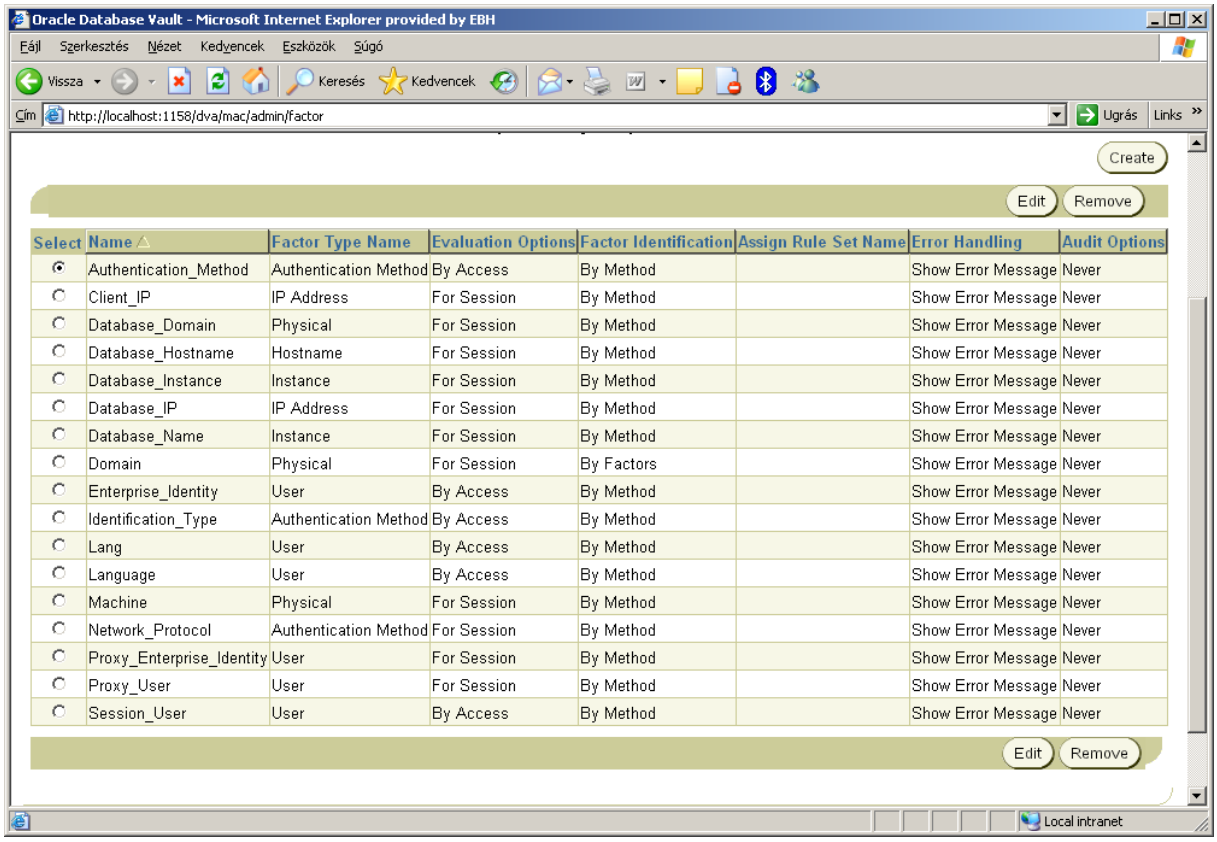
Ugyanebben a tartományban KISS nevű felhasználó csak részvevőként szerepel. Amikor ő megpróbálja ezt a szerepkört más felhasználónak odaadni hibaüzenetet fog kapni. Amennyiben KISS-t a DVO szintén megjelöli owner-ként akkor már ő is továbbadhatja a szerepkört másoknak.
Ezzel biztosítható, hogy system és DBA jogosultsági körrel rendelkező felhasználók elől role/szerepkör is elrejthető.

Alapértelmezettként 4 tartomány van a Database Vaultba beépítve:

- Database Vault Account Management: A Vault adminisztratív felhasználója rendelkezik hozzáféréssel a tartományhoz. Ez biztosítja hogy csak a DVSYS és a telepítés során megadott DV adminisztrátor felhasználója adhat ki CONNECT és DV_ACCTMGR szerepkört felhasználóknak.
- Oracle Data Dictionary: Az Oracle katalógus sémái (SYS és SYSTEM) van védve ebben a tartományban.
- Database Vault: Magát a Database Vault sémát védi
- Oracle Enterprise Manager: DBSNP és SYSMAN sémák védelmét biztosítja.

FAKTOROK meghatározása

Használhatunk előre meghatározott faktorokat, vagy definiálhatunk újakat.
Az előre megadott faktorok:

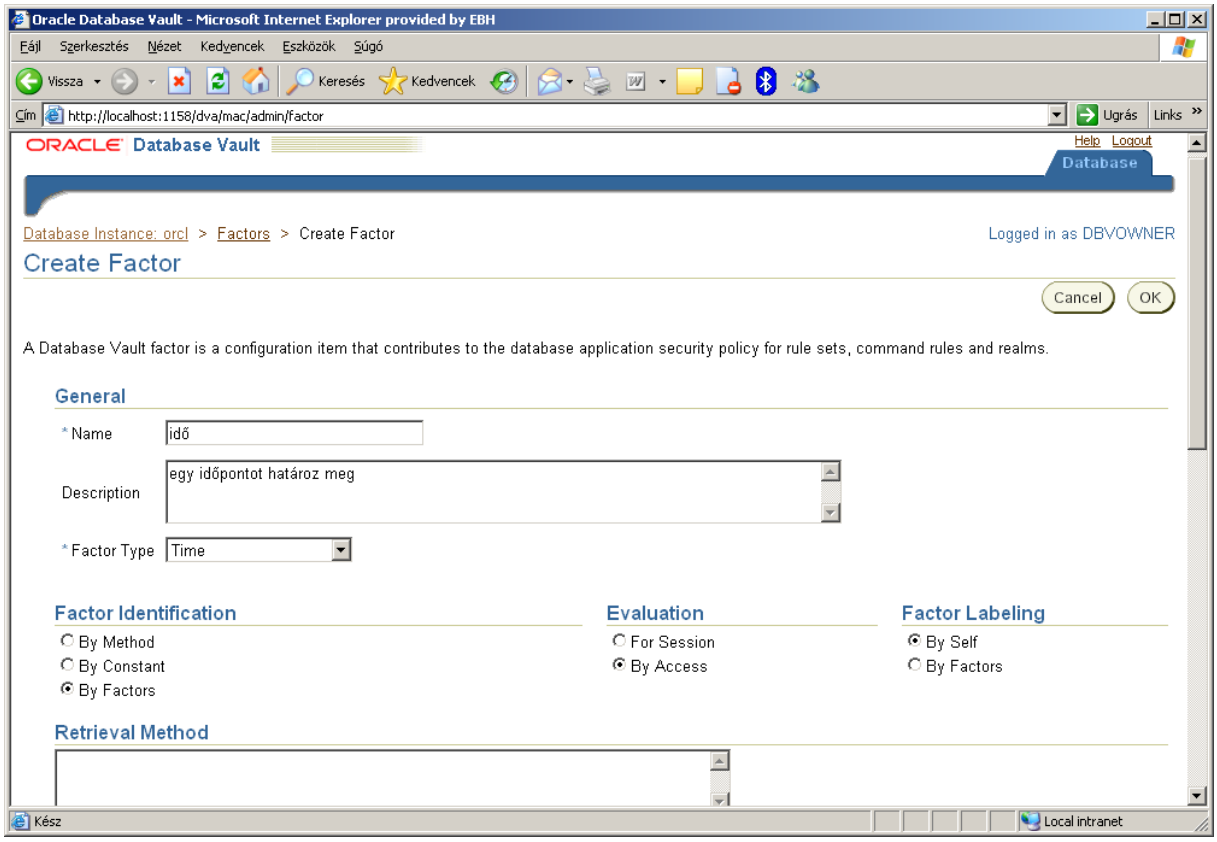


Select	Name	Factor Type Name	Evaluation Options	Factor Identification	Assign Rule Set Name	Error Handling	Audit Options
◉	Authentication_Method	Authentication Method	By Access	By Method		Show Error Message	Never
○	Client_IP	IP Address	For Session	By Method		Show Error Message	Never
○	Database_Domain	Physical	For Session	By Method		Show Error Message	Never
○	Database_Hostname	Hostname	For Session	By Method		Show Error Message	Never
○	Database_Instance	Instance	For Session	By Method		Show Error Message	Never
○	Database_IP	IP Address	For Session	By Method		Show Error Message	Never
○	Database_Name	Instance	For Session	By Method		Show Error Message	Never
○	Domain	Physical	For Session	By Factors		Show Error Message	Never
○	Enterprise_Identity	User	By Access	By Method		Show Error Message	Never
○	Identification_Type	Authentication Method	By Access	By Method		Show Error Message	Never
○	Lang	User	By Access	By Method		Show Error Message	Never
○	Language	User	By Access	By Method		Show Error Message	Never
○	Machine	Physical	For Session	By Method		Show Error Message	Never
○	Network_Protocol	Authentication Method	For Session	By Method		Show Error Message	Never
○	Proxy_Enterprise_Identity	User	For Session	By Method		Show Error Message	Never
○	Proxy_User	User	For Session	By Method		Show Error Message	Never
○	Session_User	User	By Access	By Method		Show Error Message	Never

Ezek minden kapcsolathoz automatikusan hozzárendelt értékek. Ezek felhasználhatóak újabb Faktor létrehozására ill. módosíthatók.

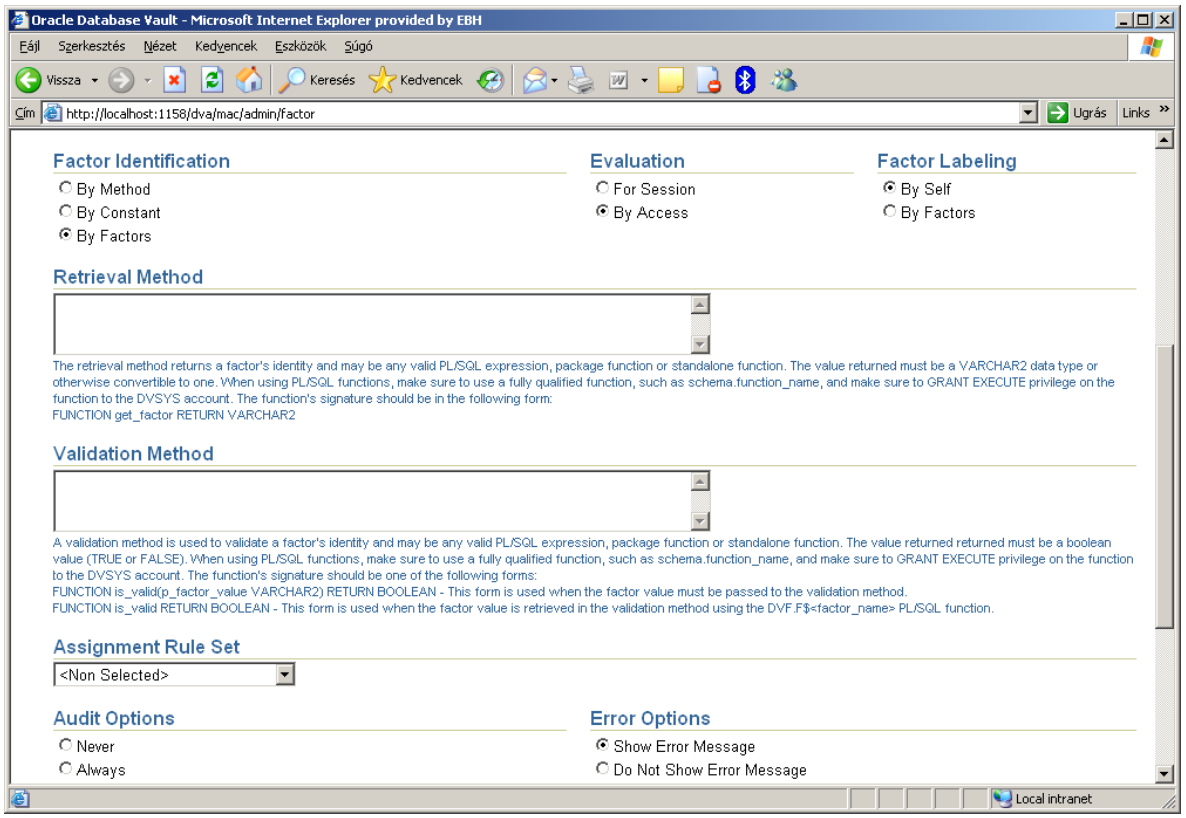
Létrehozás:

A create gombra kattintással az alábbi párbeszédpanelhez jutunk:



Meg kell adnunk:

- a faktor nevét, leírását
- típusát ami jelen eseben idő/time lesz mert olyan faktort szeretnénk létrehozni ami időpontokat fog megadni. lehetne még:



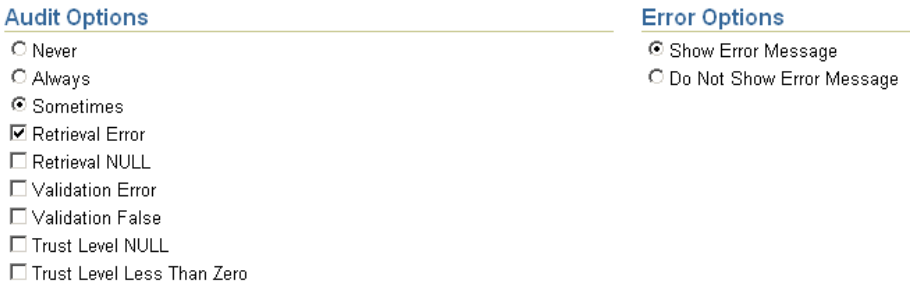
- A faktor azonosítás/identification az a folyamat amikor a faktorhoz hozzárendelünk egy értéket. Az értékek háromféleképpen rendelődhetnek a faktorhoz:

  Method- azaz függvény útján (egy PL/SQL utasítás amit a Retrieval Method mezőben megadhatunk, kikötés hogy Varchar2 típussal kell visszatérjen) Constans – egy VArchar2 konstans értéket kell megadnunk a Retrieval Method mezőben.

  Factors – előre meghatározott faktorok kiértékelése után ez a faktor az eredményt kapja meg.

- Faktor kiértékelés: Session vagy By Access, az a kapcsolat vagy a hozzáférés útján. A kapcsolat azt jelenti hogy a faktor kapcsolatonként egyszer fog kiértékelődni. Hozzáférés esetén, a kapcsolat kiépítése után és minden egyes esetben kiértékelődik amikor a faktorra hivatkozunk. Ez utóbbi magasabb terhelést fog előidézni.

- A faktorokat szintén auditálhatjuk:

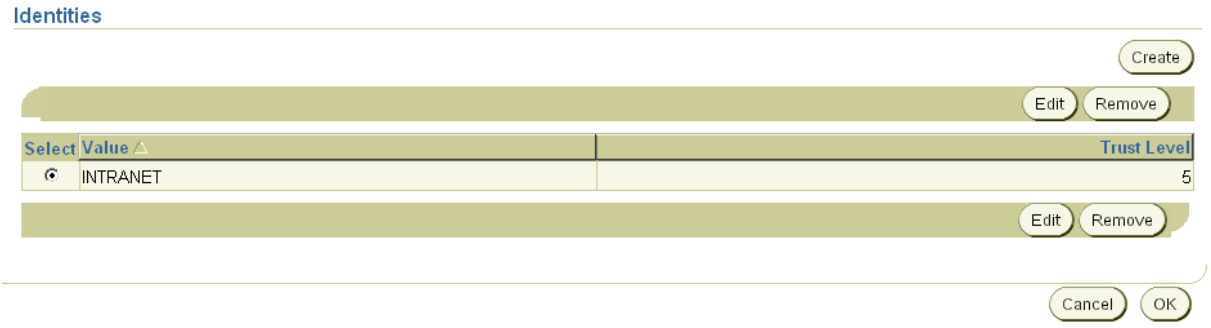


Jelen esetben csak akkor szeretnék audit bejegyzést ha a kiértékelés hibával tért vissza, és kérem hogy mutasson hibaüzenetet.

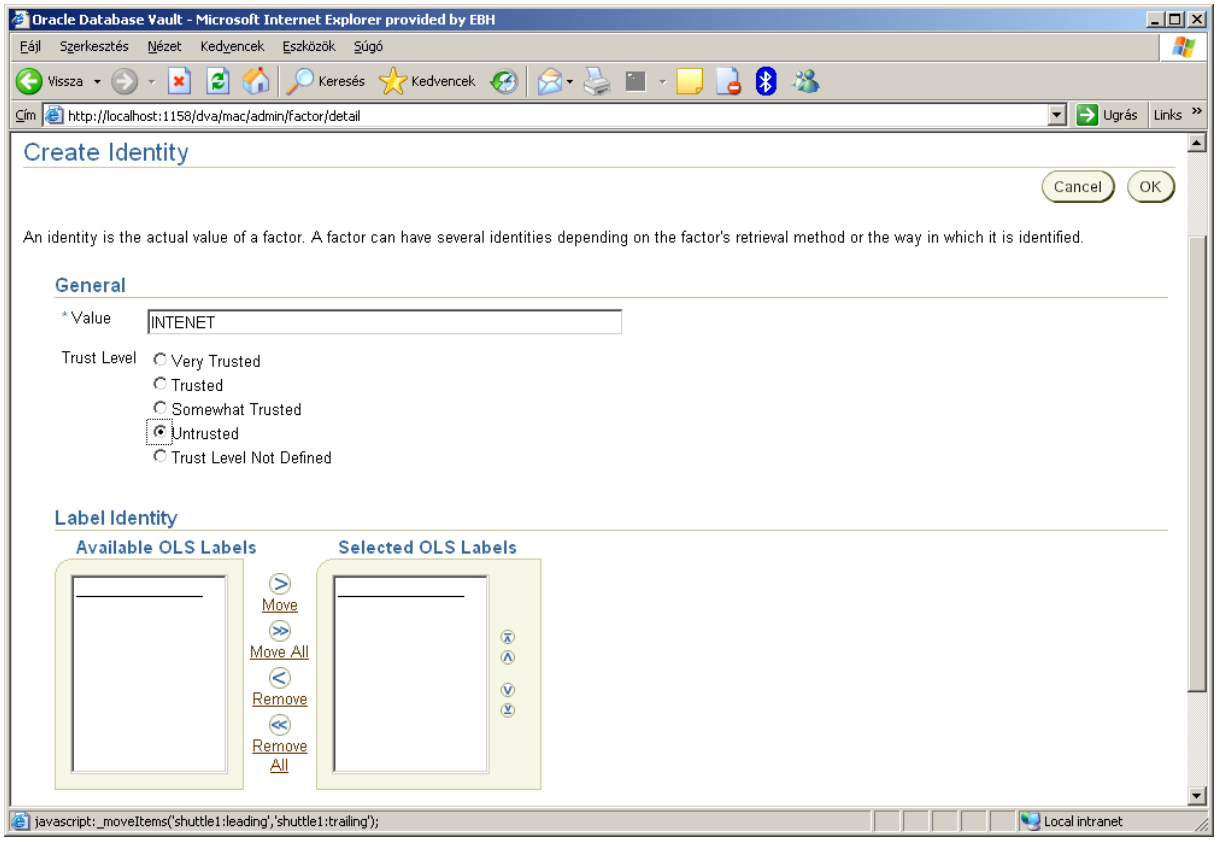
Faktort természetesen törölni is lehet, ezt a Create alatti Remove gombbal tehetjük meg.

Identities – Azonosítók:

Ez gyakorlatilag az érték amit a faktorokhoz rendelhetünk. Pl amikor egy felhasználó az irodából, belső hálózatrólcsatlahozik a DOMAIN faktor értéke INTRANET lesz TRUSTED zónában. Az előzőekben írtam arról milyen módon kaphat a faktor értéket (konstans, metódus, faktor) , azaz az azonosítókat hogyan határozhatom meg. A faktor létrehozásnál van mód az azonosító deklarációjára:

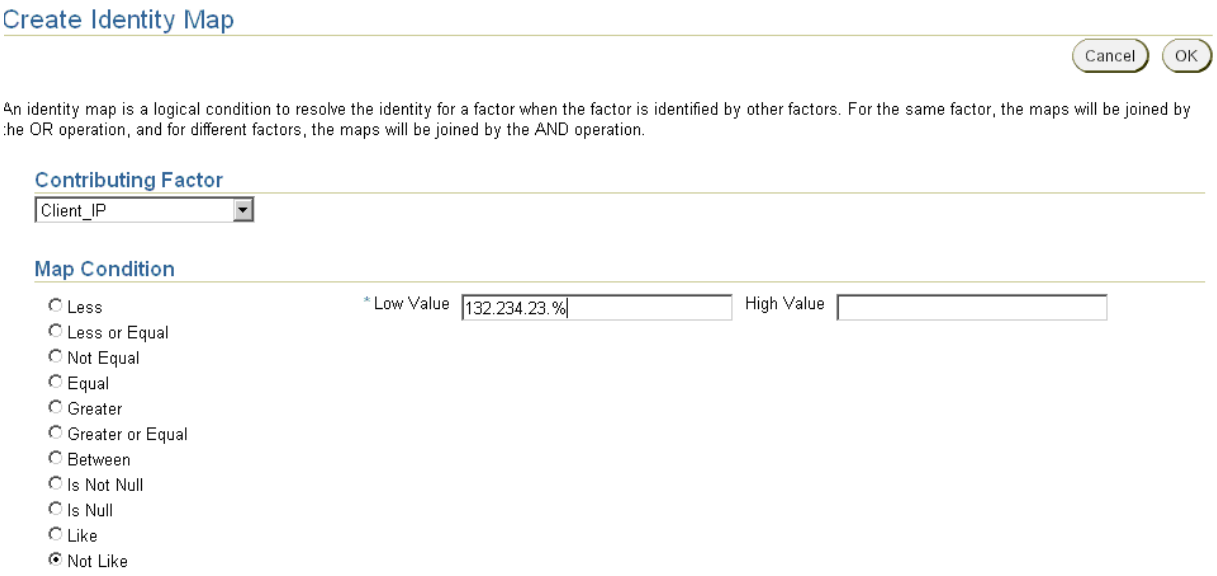


A create gombra kattintva az alábbi párbeszédpanelhez jutunk, ahol megadjuk a nevét, és jelen esetben DOMAIN faktornál a biztonsági szintjét. INTERNET esetén UNTRUSTED-et választok. Majd Ok gombra kattintással létrejön az új identity.



Trust Level, azaz mennyire bízunk menne ebben a nézetben egy szám. Ami annál nagyobb minél megbízhatóbb a kapcsolat módja.

Ugyanitt megadhatunk gyermek faktorokat. Azaz pl. megmondhatjuk hogy INTERNET az érték akkor a ha gyermek faktor, azaz a Client_IP nem a helyi hálózaton található.



Riportok

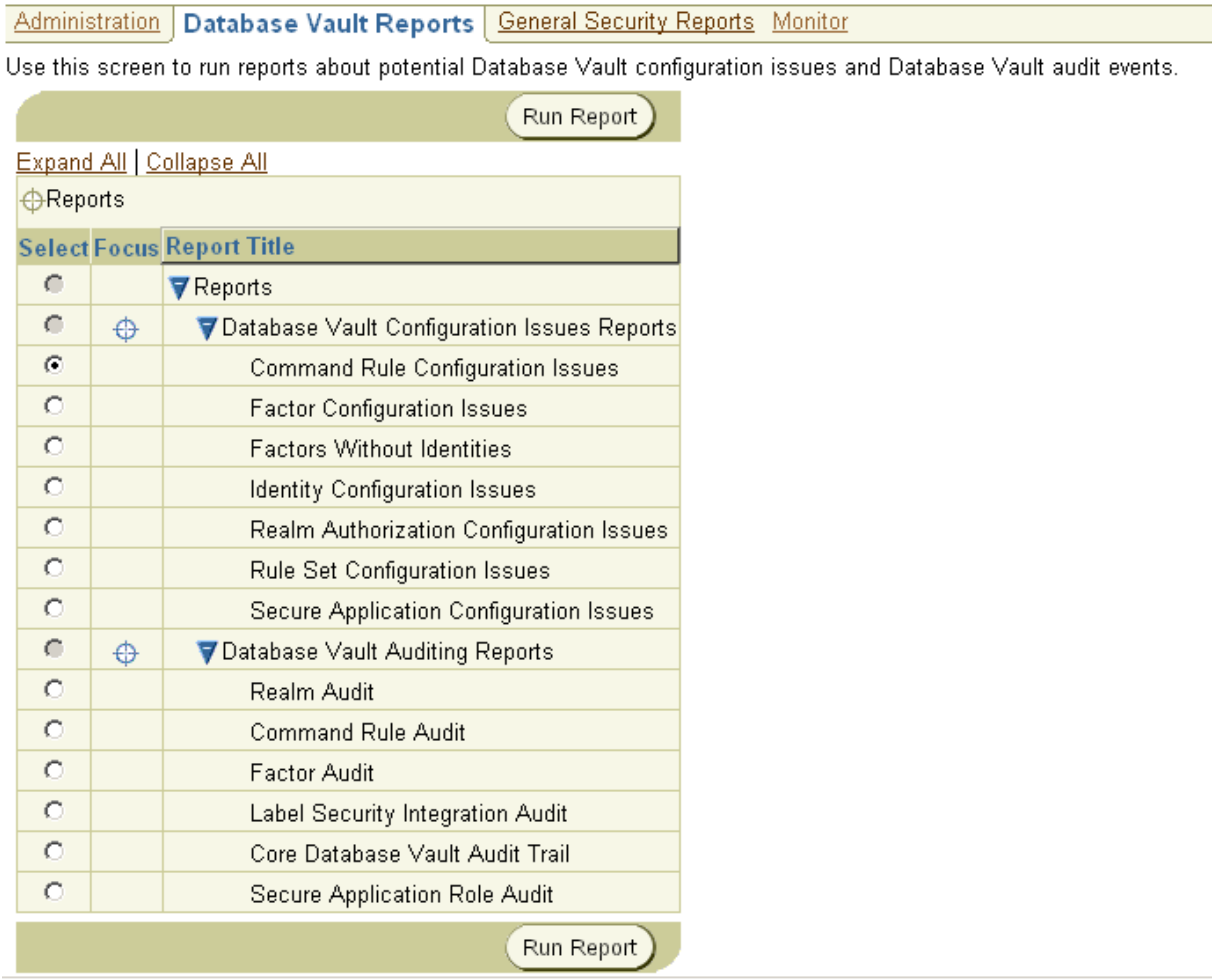
Mint a legtöbb rendszer ami biztonsági megoldásokat nyújt, a Database Vault is rendelkezik olyan riportokkal amik segítségével ellenőrizni tudjuk rendszerünket.
Két fő csoportra oszthatnánk ezeket:

- Database Vault riportok
- Általános biztonsági riportok

A Database Vault riportok segítségéve a Database Vault beállításokat tudjuk ellenőrizni, auditálni azokat a visszaéléseket amiket a Database Vault detektált.
Az általános riportok egyéb biztonsági opciókról adnak információt és ajánlást. Ilyen pl amely az alapértelmezett fiókok jelszógyengeségére, vagy a system privilégiumokra hívja fel a figyelmet.

Ahhoz hogy a riportokat lássuk szükségünk lesz a DV_ADMIN vagy DV_SECANALYST szerepkörre. Legyegyszerűbb ha létrehozunk egy DBSEC nevű usert a DV adminisztrátor felhasználóval, akinek kiosztjuk a DV_SECANALYST szerepkört.
A riportokat a Database Vault Administrator felületről tudjuk elérni.

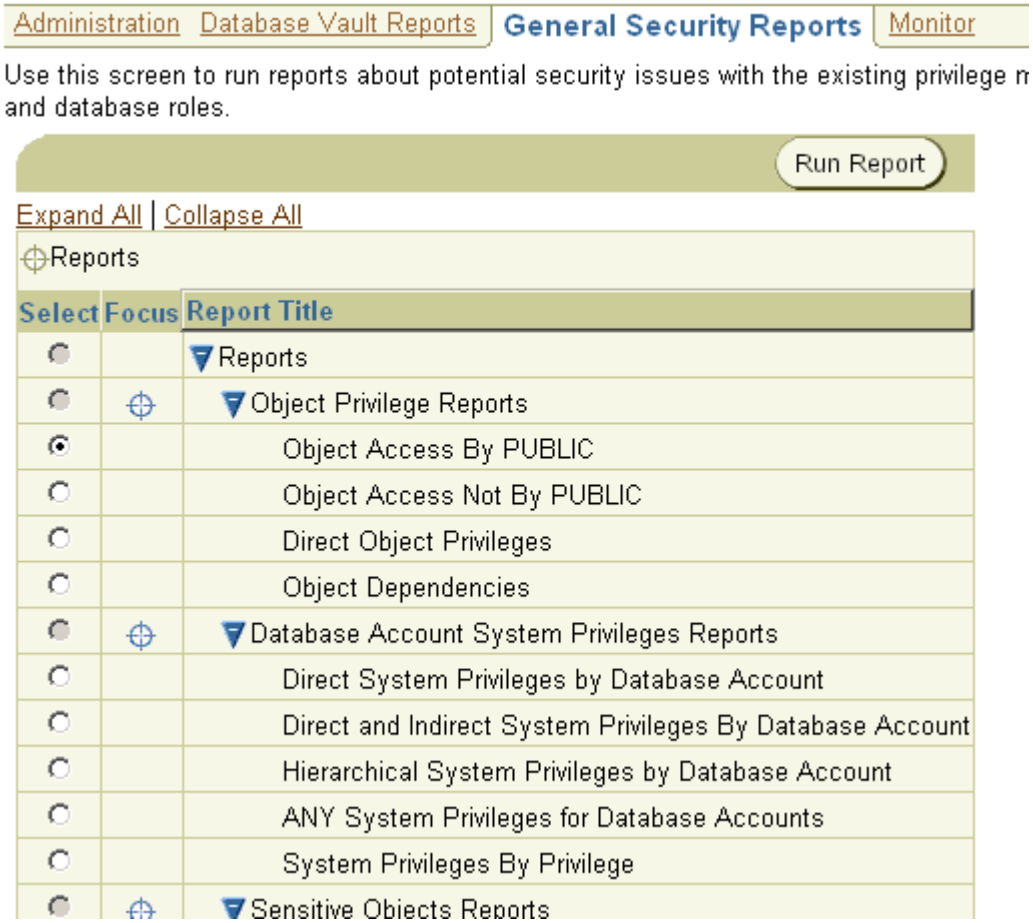
Database Vault Riportok



Ezek a riportok is két főcsoportba sorolhatók. Az első a konfigurációs a második az auditációs riportokat tartalmazzák.

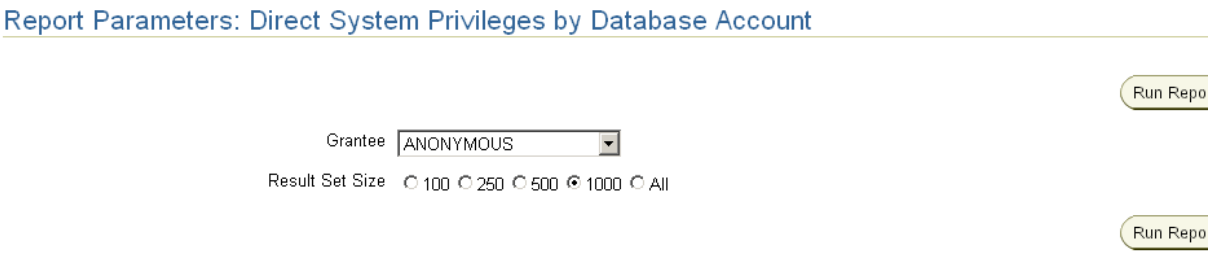
A riportok neve szerencsére beszédes, így ezekről bővebben nem írnék.

Általános riportok

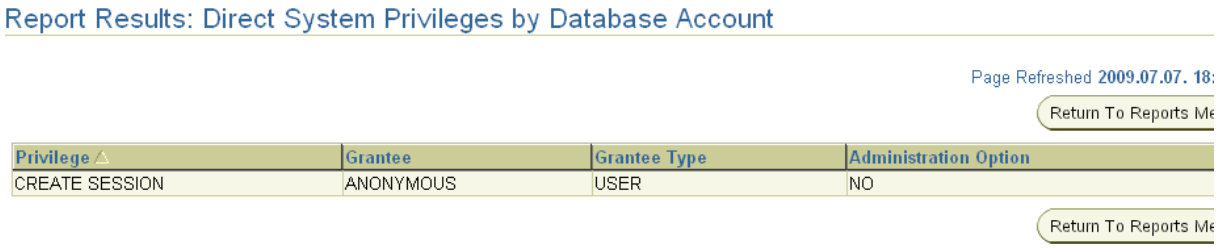


Ezek a riportok szintén kategóriákba vannak gyűjtve.

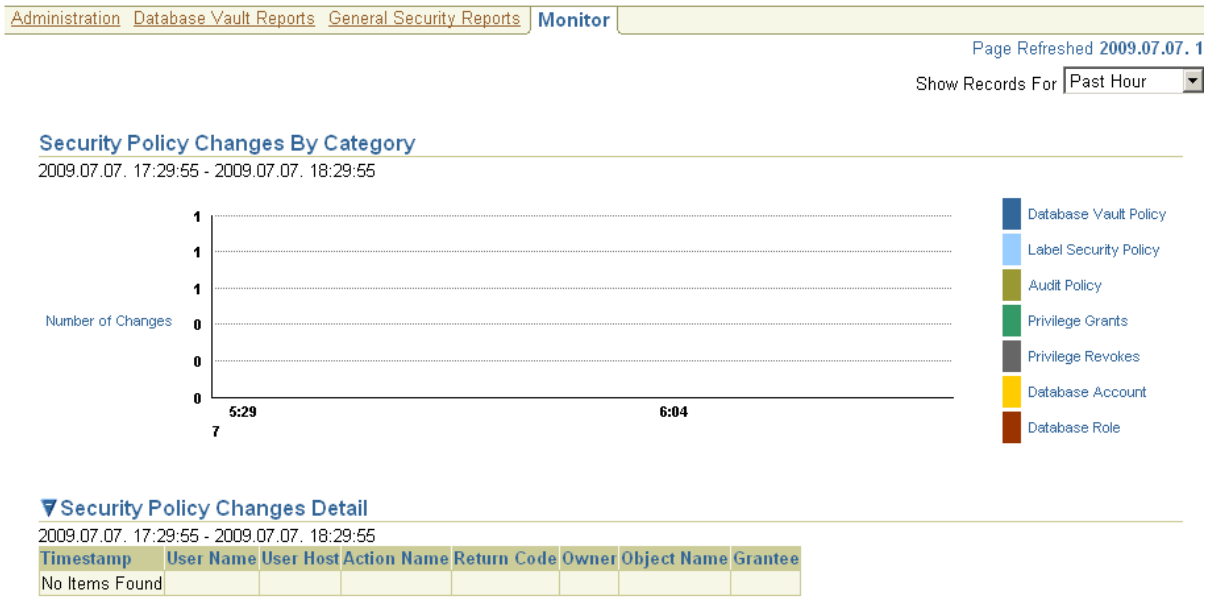
Én most egy példán keresztül arról készítek riportot, hogy az Anonymus felhasználónak milyen rendszerjogosultságokra van közvetlenül joga. (ha van ilyen)



Mikor futtatom a riportot egy párbeszédpanelben megkérdezi melyik felhasználóra vagyok kíváncsi és hány bejegyzést szeretnék látni.



A Database Vault Adminisztrator felület utolsó fülén egy általános riportot kérhetünk a rendszerben történt változásokról. Kategóriánként vizsgálhatjuk az esetleges módosításokat. Érdemes itt kezdenie a DBSEC felhasználónak amikor a rendszer monitorozását végzi.



Oracle Label Security

Az Oracle Label Security az adatvédelmi előírások kihívásaira nyújt megoldást. A hatósági és belső előírások teljesítését olyan erőshozzáférés-szabályozással segíti elő az Oracle adatbázis szintjén, amely az adatok biztonsági besorolását és a felhasználói jogosultságokat veszi alapul. Az adat besorolási címkék, amelyek az adatok érzékenységének fokát jelzik, az alkalmazás által használt adattábla egyes soraihoz rendelhetők, így sor szintű biztonsági szabályozást tesznek lehetővé. Emellett az alkalmazások moduljaihoz is külön besorolási címkéket lehet rendelni a hozzáférés részletes szabályozására. A felhasználói jogosultságoknak és adat besorolási címkéknek a szervezeten belüli következetes alkalmazása révén a biztonsági előírások többé nem csupán papírra írt szavak lesznek, hanem szorosan beépülnek az adatstruktúrákba.

Adatbesorolások

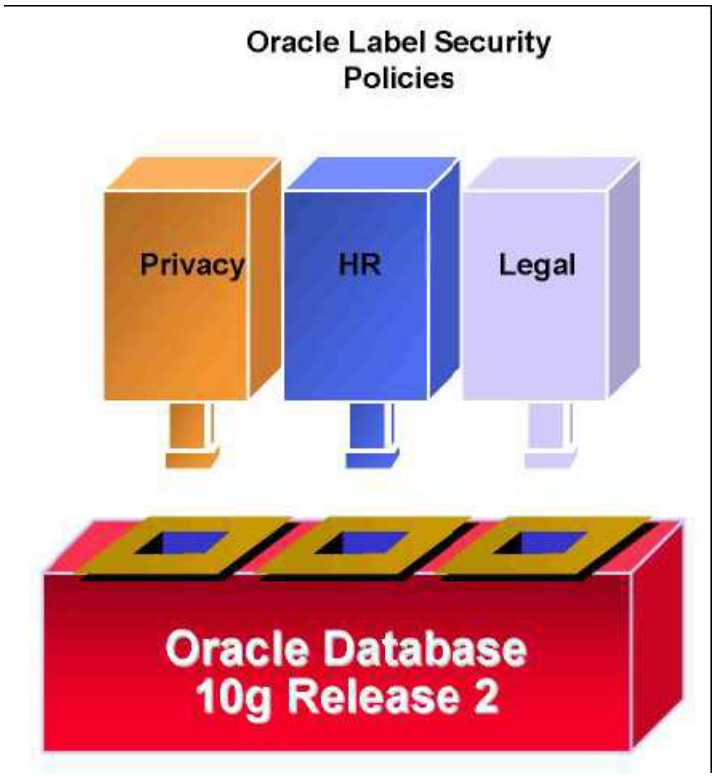
Az Oracle Label Security adat besorolási címkéi sor szintű hozzáférés-szabályozást biztosítanak. A sor szintű korlátozások kiegészítik a szerepkörökön alapuló jogosultságkezelését, így az alkalmazásfejlesztők még következetesebben alkalmazhatják azt az elvet, hogy mindenkinek csak az adott feladat elvégzéséhez minimálisan szükséges jogosultságokat kell biztosítani. A jelenlegi egyre szigorodó szabályozói környezetben az objektum szintű jogosultságkezelés nem mindig elégséges a hozzáférés megfelelő szabályozásához.

Az adat besorolási címkék igen széles körben alkalmazhatók, és a sor szintű biztonságon túl számos különböző jellegű hozzáférés-szabályozási elvi döntés gyakorlatba átültetésére alkalmasak. Például egy Oracle alkalmazás biztonsági szerepköréhez rendelt

szabálykészlet (policy) is tartalmazhat ilyen adat besorolási címkéket, illetve a címkék egy portálon az egyes weboldalak érzékeny jellegének jelzésére is alkalmazhatók.

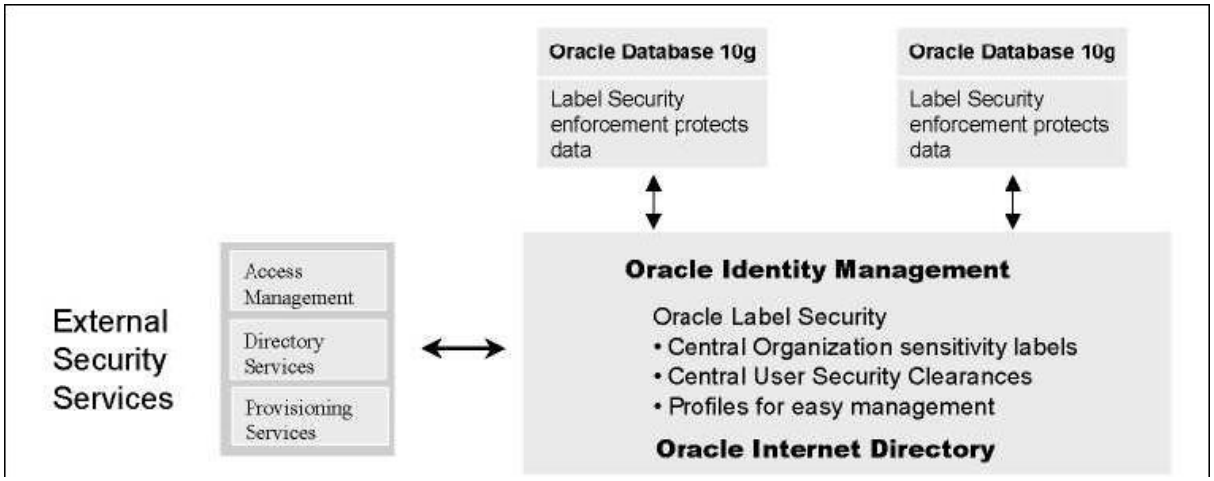
Szabálykészlet alapú architektúra

Az Oracle Label Security szabálykészlet (policy) alapú architektúrájában egy adatbázis több szabálykészletet is tartalmazhat. Ezáltal több – egyedi adat besorolási címkéket alkalmazó szabálykészletet lehet definiálni ugyanabban az adatbázisban az adatokat elérő különböző alkalmazásokhoz.


Oracle Label Security
Policies
Privacy
HR
Legal
Oracle Database
10g Release 2

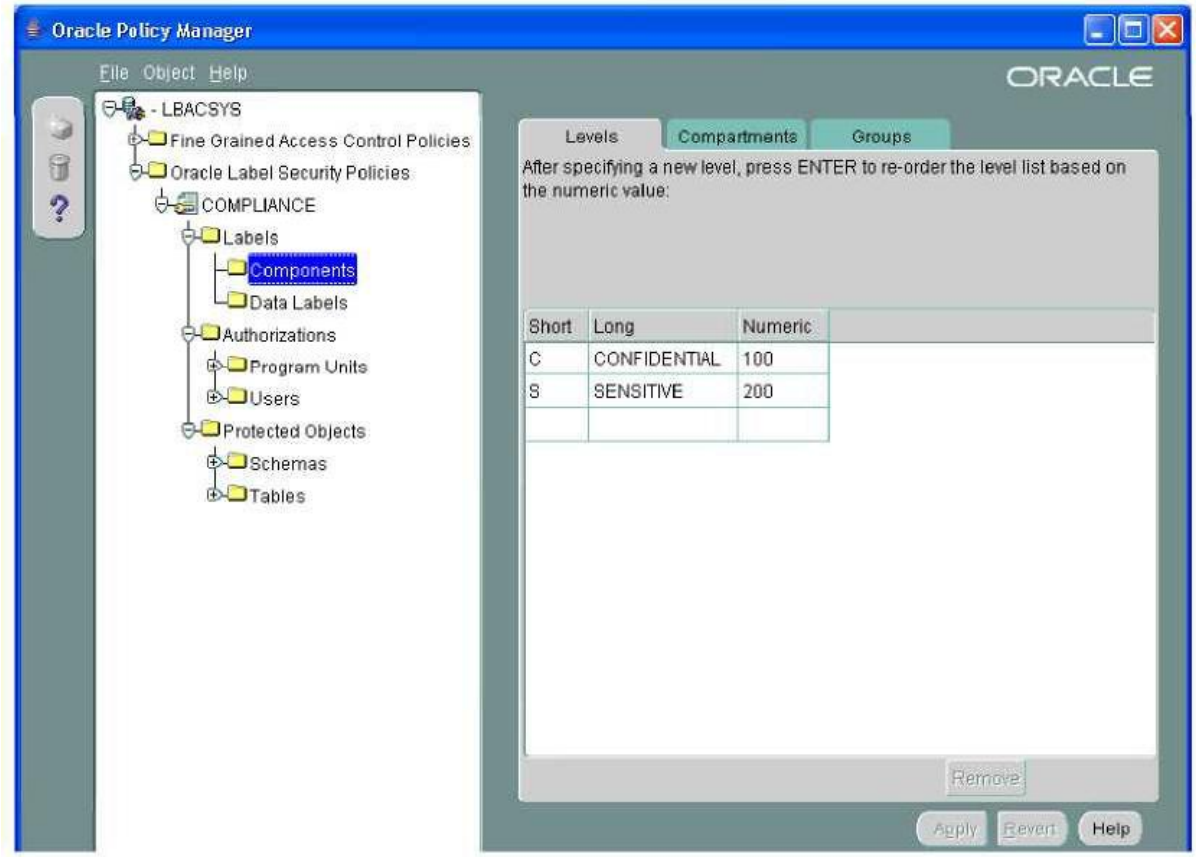
Együttműködés az Oracle Identity Management személyazonosság-kezelő funkcióival

Az Oracle Database 10*g*-ben újdonság, hogy az Oracle Label Security együttműködhet az Oracle Identity Management személyazonosság-kezelő funkcióival, így egyetlen, a szervezet egészére kiterjedő adattárban lehet kezelni a felhasználói biztonsági jogosítványokat. A felhasználói engedélyeket közvetlenül az Oracle Identity Management adattárában lehet kiosztani, majd onnan szinkronizálni más identitáskezelő megoldások adattáraival. Ezáltal a biztonsági és adatvédelmi szabályokat a szervezet egészében központilag lehet kezelni és megosztani, ami leegyszerűsíti a biztonság adminisztrációját. Az Oracle Identity Management az Oracle Application Server 10*g* egyik komponense.



Az Oracle Label Security adminisztrációs felülete

Az Oracle Label Security átfogó API-ja a meglévő alkalmazásokból is elérhetővé teszi a biztonsági funkciókat. Emellett egy Oracle Policy Manager nevű grafikus adminisztrációs eszköz segíti a szabálykészletek, adatbesorolási címkék és felhasználói biztonsági jogosítványok kezelését.



Megbízható tárolt programegységek

Az Oracle Label Security speciális hozzáférési jogosultságai tárolt eljárásokhoz is hozzárendelhetők, így a „megbízhatónak” ítélt tárolt eljárás az adatokhoz rendelt adatérzékenységi címkétől függetlenül az összes adatot láthatja. Ezáltal az érzékeny adatokhoz való hozzáférés még pontosabban behatárolható.

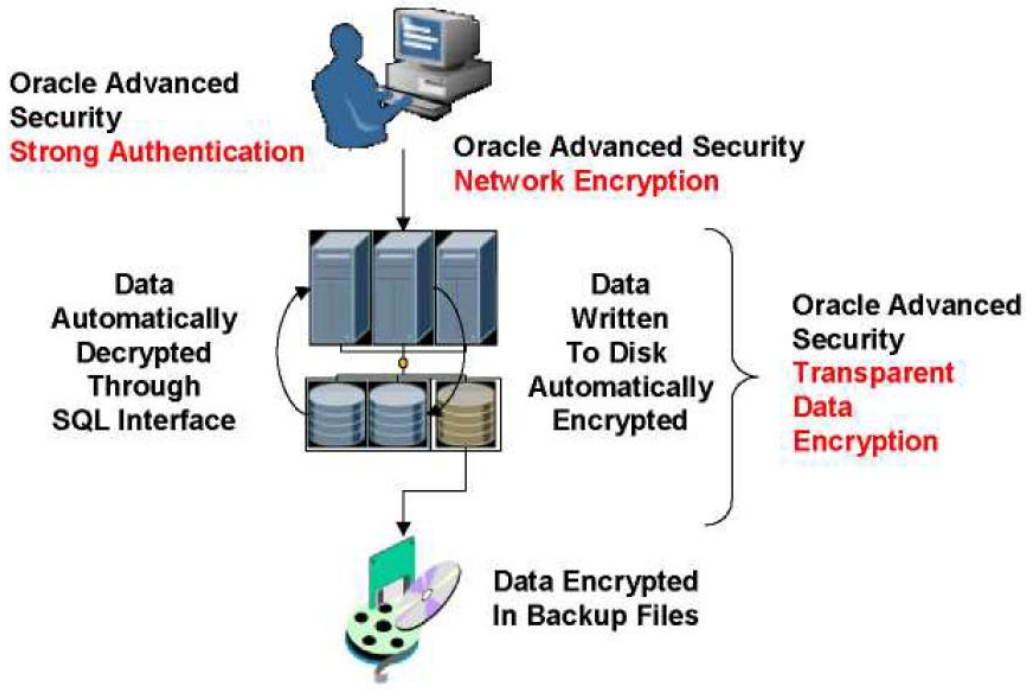
Titkosítás

A titkosítás célja a táblákban tárolt információ illetéktelen személyek előli elrejtése, értelmezhetetlenné tétele.
Az Oracle tartalmaz egy *Transparent Data Encryption* (TDE) nevű módszert. Ez a felhasználó számára átlátszó (transzparens) módon, titkosítja és fejti vissza az adatokat. A működéséhez az oszlop deklarációnál kell megmondani, hogy az Oracle titkosítson-e vagy sem.

Pl.: `CREATE TABLE Titok (titkos_adat VARCHAR2(10) ENCRYPT);`

Az Oracle INSERT, UPDATE esetén titkosít, SELECT esetén pedig azonnal visszafejt.

A TDE működéséhez telepíteni kell az *Oracle Wallet* komponenst, amely a tanúsítványokat, jelszavakat és kulcsokat tárolja. Néhány rendszerparamétert szintén módosítani kell.



Ez a módszer a rosszindulatú, külső támadó elől rejti el az adatokat, a belső támadások ellen nem véd. A fájlokba, fizikai szinten titkosítva kerül be az adat. Az illetéktelen semmire se megy a titkosított adatokkal és visszafejteni se tudja.. Mivel nem csak az INSERT, UPDATE esetén titkosít, hanem a SELECT esetén is visszafejt, így a felhasználó mindig az eredeti, adatot látja.

További nehézség, hogy az adat formátuma miután titkosítva lett nem a megszokot típusú INTEGER, VARCHAR, DATE, hanem úgynevezett RAW bájt sorozat. A titkosított adatot NEM lehet az eredeti oszlopban tárolni, mert nem ugyanolyan típusú. Akkor se lehetne, ha nem RAW, hanem VARCHAR lenne, mivel pl. egy DATE oszlopba nem lehet tartalmilag és típusát tekintve nem odaillő adatot beszúrni.

Példa:

- dátum: 2009.04.17 (DATE),
  - ugyanez titkosítva: AG7657SDFH6433NBV92PP3FGG

Az SQL-ben megírt alkalmazás módosítás nélkül zavartalanul működik tovább a titkosított adatokkal. Az alkalmazásokban nem módosul az a programkód, amely az adatoknak az adatbázistáblába való beillesztését végzi, az adatokat az Oracle adatbázis-kezelő titkosítja automatikusan a lemezre kiírás előtt. A meglévő biztonsági mentési programok ugyanúgy használhatók, de most már a szalagra mentett adatok is titkosítva lesznek. A későbbi Select műveletek szintén transzparensen visszafejtik az adatokat az alkalmazás működésének megzavarása nélkül.

A transzparens adattitkosítás által támogatott adattípusok

A transzparens adattitkosítás első változata az alábbi adattípusokat támogatja. A később megjelenő változatok további adattípusokat is támogatnak majd.

Adattípus	**A transzparens adattitkosítás támogatja**
varchar2	Igen
nvarchar2	Igen
number	Igen
date	Igen
binary float	Igen
binary double	Igen
timestamp	Igen
raw	Igen
char	Igen
nchar	Igen

A titkosító kulcs beállítása

Az Oracle transzparens adattitkosítása biztosítja a titkosítás működéséhez szükséges kulcskezelő infrastruktúrát. A titkosítás úgy működik, hogy a titkosító rutin megkapja a szöveges formátumú adatot, és emellé egy titkos kulcsot is kap. A kulccsal titkosítja a tiszta szöveges adatot, és visszaadja a titkosított adatot. Az adatbázis főkulcsának szerepe az, hogy ezzel titkosítják az Oracle adatbázis-kezelő által az egyes táblákhoz automatikusan létrehozott saját titkosító kulcsokat.

A főkulcs nélkül nem lehet visszafejteni semmilyen adatot. Az adatbázis elindításakor az adatbázis-adminisztrátor egy megfelelő jelszóval megnyit egy Oracle Wallet nevű objektumot. Ez a „pénztárca” tartalmazza a főkulcsot. Mindenképpen meg kell nyitni titkosított adat elérése előtt.

Mielőtt az adatbázisban első ízben alkalmaznák a titkosítást, inicializálni kell az adatbázis főkulcsát.

SQL> alter system set encryption key identified by “1wq!r23t”;

Az érzékeny adatok védelme a hálózaton

Az Oracle Advanced Security az Oracle adatbázisok felé irányuló és a kifelé menő forgalmat egyaránt védi. A felhasználók a hálózaton áthaladó adatok védelméhez választhatnak az Oracle Advanced Security natív titkosító és adatintegritási algoritmusai, illetve az SSL közül. Néhány tipikus helyzet, amely hálózati szintű titkosítást igényel:

- Az adatbázisszerver tűzfal mögött működik, és a felhasználók kliens–szerver alkalmazásokkal érik el a szervert
- A DMZ-ben működő alkalmazás-szerver és a második tűzfal mögötti adatbázis-kezelő közti kommunikációt titkosítani kell
- Az adatbázisok közti adatforgalmat titkosítani kell

Az Oracle Advanced Security natív titkosító és adatintegritási algoritmusai nem igénylik PKI kulcskezelő infrastruktúra létesítését. Az adatbázis-kezelő későbbiekben megjelenő verzióiban sorra megtalálhatók lesznek az elfogadott újabb titkosító algoritmusok. A legújabb algoritmus az Advanced Encryption Standard (AES), amely a DES-nél jobb biztonságot és teljesítményt nyújt. A jelenleg támogatott titkosító és adatintegritási algoritmusok:

- AES (128, 192 és 256 Key)
- RC4 (40, 56, 128, 256 Key)
- 3DES (2 Key és 3 Key)
- MD5
- SHA1

Az SSL alapú titkosítást a PKI infrastruktúrát alkalmazó szervezetek használhatják. A TLS 1.0 támogatása az Oracle Database 10*g* Release 1 változatában jelent meg. Az Oracle Advanced Security az Oracle Database 10*g* Release 1 változatában megjelenő TLS 1.0 protokollhoz biztosított először AES kódolási készletet („cipher suite”).

Alkalmazások módosítása

A hálózati titkosításhoz és adatintegritás-védelemhez – az Advanced Security opció kliens és szerver gépeken történő telepítését követően – elegendő a szerver és/vagy a kliens hálózati

beállításait módosítani. Ezért a legtöbb szervezet könnyen be tudja vezetni, mivel az alkalmazásokat nem kell módosítani.

Összefoglalás

A bemutatott lehetőségeket segíthetnek az adatbázisban tárolt adataink ne jussanak illetéktelen kezekbe, ám ne feledjük hogy 100%-os védelmet nem nyújtanak. Egy cég informatikai rendszereinek működésében nagyon fontos a rendelkezésre állás és a biztonság egyensúlya. Bármely előbb tárgyalt lehetőséggel is élünk a rendelkezésre állás, a gyors elérhetőség, gyors válaszadás fog hátrányt szenvedni. A DBA-k és feladata megtalálni a középutat.

Ez általában úgy történik, hogy mielőtt egy rendszer éles környezetbe kerülne legalább egy, de ideális esetben 2-3 tesztkörnyezeten tesztelődik. Itt elsősorban nem a funkcionális tesztre hanem az ez után következő performancia tesztekre gondolok. Ezek a jövőbeli éles környezettel egyező biztonsági beállítások, erőforrás adottságokkal rendelkeznek.

Az Oracle rendelkezik olyan monitorozó eszközzel (OEM) amivel jól mérhető a futásteljesítmény.
Így még azelőtt korrigálhatjuk a beállításokat mielőtt éles működés közben, a felhasználók panaszáradata jelezné, hogy használhatatlan a rendszer.(De legalább biztonságos ☺)

Az adatbázis biztonsága egyelőre nagymértékben függ az emberi tényezőtől, a becsület és lelkiismerettől. Bár létezik rá módszer mint ahogy az előzőekben láttuk, de gyakorlatban tökéletes biztonság nem létezik.

IRODALOMJEGYZÉK

Az informatikai biztonság kézikönyve (Lajos Muha, Ákos Bodlaki)

ORACLE Database 10g -Teljes referencia (Loney, Kevin)

www.oracle.com/.../DATABASE10GR2_ADVANCEDSECURITY_HUN.pdf

www.databasejournal.com/features/oracle/article.php/3065431/Oracle-Label-Security-Part-1-Overview.htm

Oracle Audit Vault – Administrator's Guide 10g Release 2 (10.2.2) August 2007

http://www.oracle.com/technology/deploy/security/database-security/database-vault/index.html

www.oracle.com/global/hu/database/collaterals/DATABASE10GR2_OLS_HUN.pdf -

http://books.google.hu/books?id=6HSlIOJyOrcC&pg=PP1594&lpg=PP1594&dq=Oracle+Listener++biztons%C3%A1g&source=bl&ots=-qEAyws-oL&sig=fyDxpgENfYqTtlcpmhup3_FAomU&hl=hu&sa=X&oi=book_result&resnum=2&ct=result#PPP1593,M1

http://download.oracle.com/docs/cd/B14117_01/server.101/b10739.pdf