HADMÉRNÖK

IV. Évfolyam 3. szám - 2009. szeptember

Fleiner Rita
fleiner.rita@nik.bmf.hu

KRITIKUS ADATBÁZISOKRA ÉPÜLŐ INFORMATIKAI RENDSZEREK ARCHITEKTÚRÁI ÉS BIZTONSÁGI SZEMPONTJAI

Absztrakt

A publikáció a kritikus adatbázisokat tartalmazó informatikai rendszerek felépítésével és ezek adatbázis-biztonságot érintő kérdéseivel foglalkozik. Ismertetésre kerül a többrétegű architektúra modellje, melyben négy réteget, nevezetesen a megjelenítési réteget, a távoli elérés kiszolgáló réteget, az alkalmazás réteget és az adatbázis réteget különböztetjük meg. A szerző megvizsgálja az adatbázis réteg magas fokú rendelkezésre állás megvalósításának módjait, a különböző mentési technikákat, az adatbázis rendszerek fürtözését és tükrözését. A publikáció elemzi a bemutatott architektúrák különböző pontjain jelentkező, adatbázisok biztonságával kapcsolatos sebezhetőségeket.

The publication studies the architecture of information systems containing critical databases and the related database security aspects. The multi-tier architecture model is described distinguishing four different tiers, namely the presentation tier, the remote access service tier, the application tier and the database tier. The author examines the methods of achieving high availability in the database tier, the different database backup techniques, the clustering of database servers and the mirroring of databases. The publication analyzes the database security related vulnerabilities contained in the studied architectures.

***Kulcsszavak**: kritikus adatbázis, többrétegű architektúra, magas fokú rendelkezésre állás, adatbázis-biztonság ~ critical database, multi-tier architecture, high availability, database security*

BEVEZETÉS

A fejlett XXI. századi társadalmak egyre nagyobb mértékben függenek a különböző infrastruktúráktól, melyek egymással is szoros kapcsolatban, kölcsönös függésben állnak. A társadalmi, gazdasági és hétköznapi élet működési folyamatai egyre inkább veszélyeztetettek a kritikus, más szóval létfontosságú infrastruktúrák működésének, szolgáltatásainak megszakadása esetén. A **kritikus infrastruktúrák** általános fogalma alatt olyan infrastruktúrákat (működtető személyzet, folyamatok, rendszerek, szolgáltatások, létesítmények, és eszközök összessége) értünk, amelyek megsemmisülése, szolgáltatásaik vagy elérhetőségük csökkenése egy adott felhasználói kör létére, lét- és működési feltételeire jelentős negatív hatással jár. A kritikus infrastruktúrákban gyakran találunk a működés szempontjából nélkülözhetetlen adatbázisokat, ezek biztonsága az adott kritikus infrastruktúra biztonságának alapvető összetevője. Ezeket a létfontosságú adatbázisokat **kritikus adatbázisoknak** nevezzük.

Az adatbázis kezelő rendszerek architektúrájában jelentős változás, fejlődés figyelhető meg [1]. A kezdetekre a legegyszerűbb felépítés az **egygépes** megvalósítás jellemző, ahol az adatbázis és az azt feldolgozó program ugyanazon a gépen található, az adatbázist egy adott időben csak egyetlen program használja.

A **file-szerver** architektúrában az adatbázis állományok már átkerülnek egy központi szerverre, ami csak az adatok tárolásáért felelős és egy időben több program is elérheti ezt a hálózaton keresztül. Ha a felhasználó adatműveletet akar végrehajtani, akkor az adatrekordoknak el kell jutniuk a felhasználóhoz a hálózaton. Ez nagy adatforgalommal jár, ami a hálózat túlterheléséhez vezethet.

A **kliens-szerver** architektúra esetén két egységet különböztetünk meg. Az adatok közvetlen kezeléséért az adatbázis-szerver a felelős, míg az ügyfél program feladata a felhasználóval való kapcsolattartás és az üzleti logika által megkívánt feladatok végrehajtása. A hálózaton a feldolgozandó adatoknak csak a szükséges része utazik a szervertől a kliensig. Az adatfeldolgozást a szerver végzi a kliens parancsai szerint, a parancsokat SQL nyelvben adjuk meg.

A **többrétegű** (angolul multi-tier) adatbázis architektúrában a kliens nem közvetlenül az adatbázis-szerverhez, hanem a középen elhelyezkedő alkalmazás szerverhez kapcsolódik. Az alkalmazás szerver végzi el az üzleti logika által megkívánt számításokat, feldolgozásokat és hajtja végre az adatbázis-szerverrel a kommunikációt. A kliens az alkalmazás szervertől kapja a szükséges adatokat, feladata csak a felhasználóval való kapcsolattartás (vékony kliens). A modern rendszerekre ez a felépítés jellemző, ezért a következőkben erre koncentrálunk.

Adatbázis-kezelő rendszer alatt több felhasználós, hálózatos környezetben működő, adatbázisokhoz való hozzáférést biztosító, felhasználói folyamatok zavartalan működését ellátó szoftverrendszert értjük. **Adatbázisnak** nevezzük a bizonyos struktúra szerint felépülő felhasználói és rendszeradatok összességét, melyet az adatbázis-kezelő rendszer kezel. Az adatbázis-kezelő rendszert **adatbázis szerverpéldánynak** (database instance) is szokták nevezni. **Adatbázis szervernek** nevezzük az egy vagy több adatbázis szerverpéldányt futtató számítógépet.

A publikáció alapvető célja a kritikus adatbázisokat tartalmazó informatikai rendszerek

architektúrájának vizsgálata és az ebben rejlő sebezhetőségi pontoknak a feltárása. Ennek érdekében a publikáció:

1 Ismerteti a többrétegű architektúrák modelljeit és ezek komponenseit.
2 Bemutatja a kritikus adatbázis rendszerek különböző architektúráit, különös tekintettel a kritikusság szempontjából egyik leglényegesebb biztonsági jellemző, a magas fokú rendelkezésre állás felépítésére.
3 Elemzi a bemutatott architektúrák különböző pontjain jelentkező, adatbázisok biztonságával kapcsolatos sebezhetőségeket.

A TÖBBRÉTEGŰ ARCHITEKTÚRA MODELLJEI

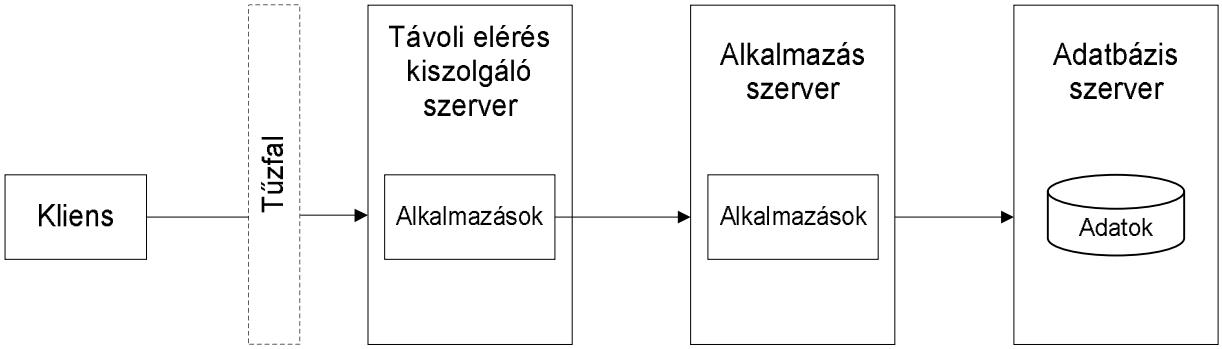
Ebben a fejezetben megvizsgáljuk a kritikus adatbázisokat tartalmazó informatikai rendszerek felépítését. Feltételezzük, hogy a rendszert sok felhasználó használja és az adatbázisok hálózati összeköttetés útján érhetők el. Napjainkban ezekre a rendszerekre gyakori a többrétegű architektúra szerinti felépítés, bár elterjedőben van egy új modell, a szolgáltatás orientált architektúra is. A réteg egy funkcionálisan elkülönített hardver és szoftver komponenst jelent, a legtisztább estben önálló számítógépre telepítve. A rendszerek egyik jelentős csoportját alkotják a webes alkalmazások, ahol a rétegeknek speciális elnevezéseik vannak. A következő rétegeket különböztetjük meg:

A **megjelenítési réteg** (felhasználói felület, kliens, user interface) felelős a felhasználói felületért és a felhasználóval való kapcsolattartásért, az architektúra legtetején helyezkedik el. A kliens felületnek felhasználóbarátnak, ugyanakkor elronthatatlannak kell lennie. Webes alkalmazások esetén ezt a réteget a böngésző jeleníti meg, mely HTTP protokollon keresztül kapcsolódik a web szerverhez.

A **távoli elérés kiszolgáló réteg** felelős a felhasználói felülettel való kapcsolattartásért. A kliens kéréseit továbbítja az alkalmazás réteg felé, illetve az onnan érkezett válaszokat küldi vissza a kliensnek. Leggyakrabban ez a réteg képezi a választóvonalat a szervezet megbízható belső hálózata és a megbízhatatlan külső hálózat (például az internet) között. Webes alkalmazások esetében ez a réteg a web szervert, a HTTP forgalom kezelésével kapcsolatos részt jelenti, melyet tűzfallal védenek.

Az **alkalmazás réteg** (alkalmazás logika, üzleti logika) felelős az alkalmazás által megfogalmazott feladatok végrehajtásáért, az egy szinttel lejjebb elhelyezkedő adatbázis rétegtől a szükséges adatok megszerzéséért, illetve ezen adatok módosításának, törlésének kezdeményezéséért. Ennek a rétegnek a feladatát egy vagy több alkalmazás szerver látja el. Ezek a biztonságos belső hálózatban találhatók, méghozzá a web szerver és az adatbázis szerverek között.

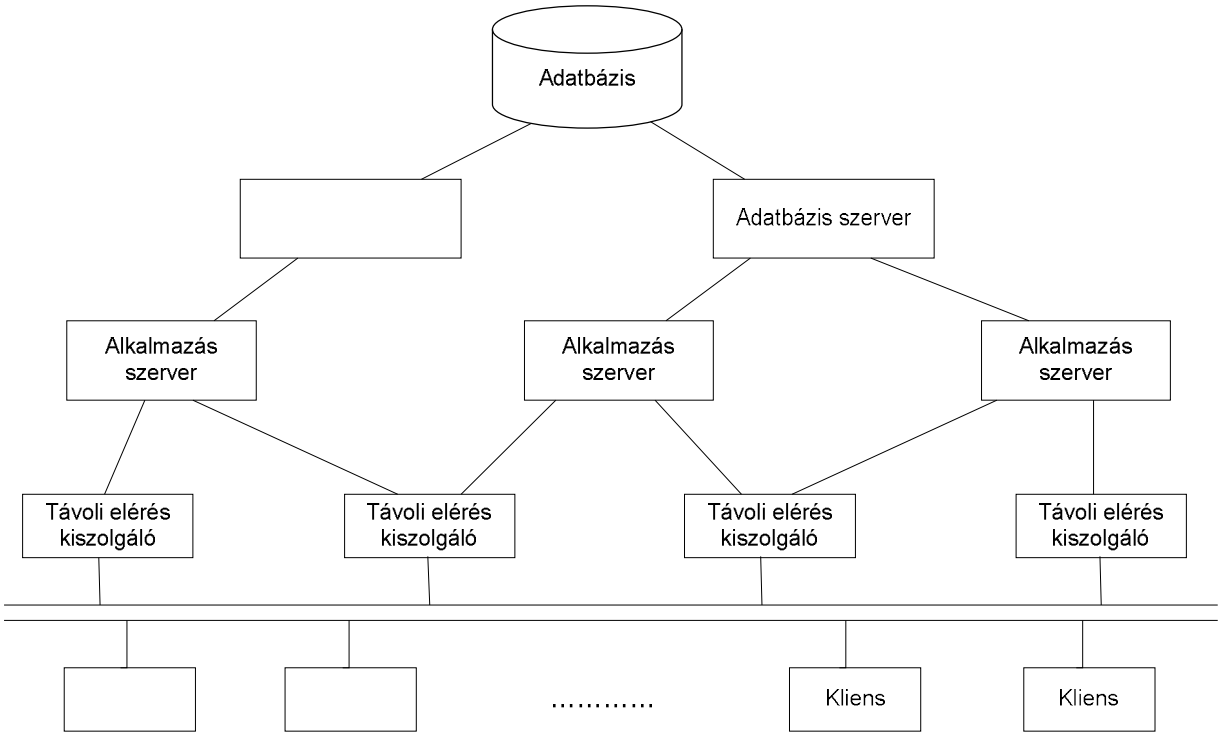
Az **adatbázis réteg** a többrétegű architektúra legalsó szintjén található, az adatok fizikai eléréséért, feldolgozásáért felelős. E réteg feladata például az adatbázis állományok nyitása, zárása, új adat felvitele, törlése, módosítása, indexek kezelése, zárolási konfliktushelyzetek feloldása. Ez a réteg az adatok alkalmazásoktól független tárolásáért felelős. Ebben a rétegben kaphatnak helyet az adatbázisok, adatbázis szerverek, fájl szerverek, különböző háttértárak.



1. ábra. A 4-rétegű architektúra [2]

Az ábrán látható felépítés egy javasolt architektúra tervet mutat be. A rétegek fizikai és logikai szétválasztása a sérülékeny pontok helyes kezelését és az érzékeny adatok védelmét segíti. Vannak azonban olyan helyzetek, amikor a fenti architektúra módosított változatát lehet, illetve célszerű használni. Előfordulhat, hogy a web szervert és az alkalmazás szervert fizikailag ugyanarra a gépre kell, illetve célszerű helyezni. Léteznek olyan informatikai rendszerek, melyek a külső, megbízhatatlan hálózattól teljesen szeparáltan működnek, tehát az összes réteg a megbízható tartomány része. Máskor egy proxy szerver segítségével történik a külső és a belső hálózat elválasztása. A proxy szerver fogadja a kliensek kéréseit és továbbítja ezeket az azonos gépen elhelyezkedő web/alkalmazás szerver felé.

A hatékonyság és a biztonságos működés érdekében a nagy rendszerek esetén egy-egy réteg feladatát több szerver látja el párhuzamosan. A következő ábra ezt a megvalósítást szemlélteti:



2. ábra. A 4-rétegű architektúra több-szerveres környezetben [3]

MAGAS RENDELKEZÉSRE ÁLLÁS ADATBÁZISOK SZEMPONTJÁBÓL

Az általunk vizsgált informatikai rendszerek kritikus adatbázisokat tartalmaznak, ebből kifolyólag az adatbázis réteg egyik lényeges követelménye lesz a rendelkezésre állás biztosítása. Informatikai rendszerek rendelkezésre állásán azt az időarányt értjük, amellyel egy definiált időintervallumon belül a rendszer a tervezéskor meghatározott funkcionalitási szintnek megfelelően a felhasználó által használható. A definíciót csak javítható (azaz meghibásodás esetén visszaállítható) rendszerek esetén értelmezzük. A rendelkezésre állás (availability, A) kiszámításának módja:

$$A = (MTBF)/(MTBF + MTTR),$$

ahol MTBF a meghibásodások közötti átlagos idő (Mean Time Between Failures), MTTR pedig a visszaállítás átlagos ideje (Mean time to repair). Magas fokú rendelkezésre állás esetén az arány (A) egyhez közeli érték (pl. 99%).

Adatbázis rendszerek tekintetében a magas fokú rendelkezésre állás biztosítása az adatok mentése és a rendszerek meghibásodásának kivédése köré csoportosul. A következőkben az adatbázis mentési módszereket tekintjük át.

Az adatbázis-kezelő rendszerek fejlett **mentési és helyreállítási** eszközökkel rendelkeznek, melyek az adatok védelmének szükséges eszközei. Többféle biztonsági mentési technika létezik. Időnként minden adatról érdemes biztonsági mentést készíteni, de mivel ez túlzottan idő- és erőforrás igényes, ezt a gyakorlatban csak bizonyos időközönként végzik el. Előnye, hogy a visszaállítás viszonylag gyors. Az adatok értékétől, változásuk sűrűségétől függően határozzák

meg a **teljes biztonsági mentések** gyakoriságát (például naponta, hetente, havonta).

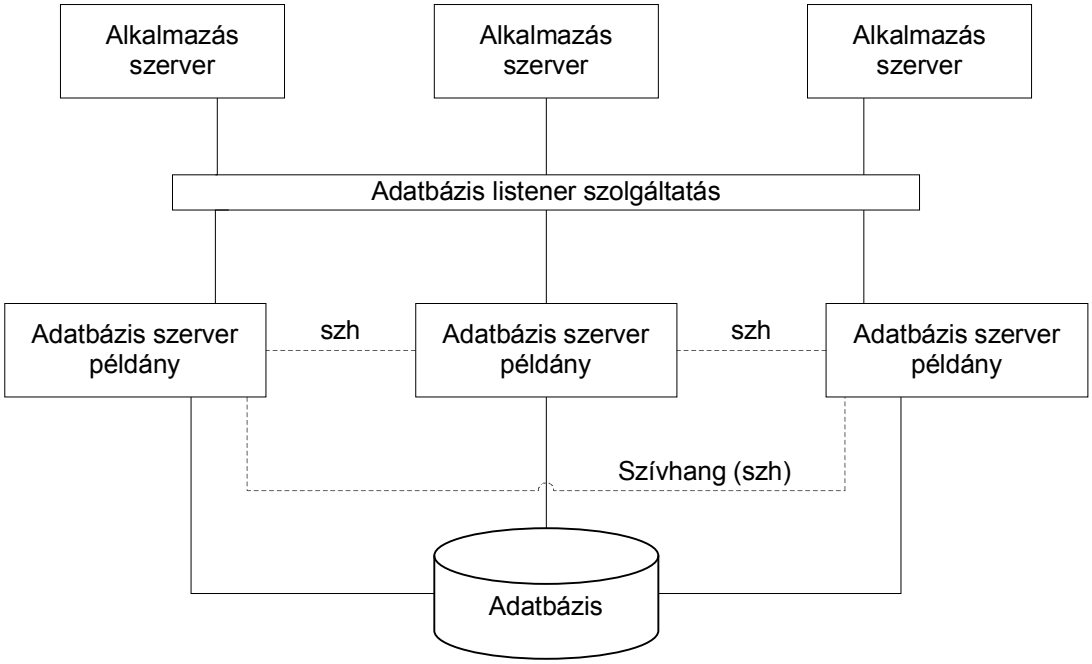
A köztes időszakban az adatok **részleges mentését** végzik el, szintén egy meghatározott gyakoriság szerint. A részleges mentés során csak a korábbi mentés óta megváltozott adatok mentésére kerül sor. Ekkor a visszaállításhoz természetesen több biztonsági mentésre is szükség van, egy teljesre, és az azóta történt változásokat tartalmazókra. A részleges mentésnek két alapvető fajtája van: az inkrementális és a differenciális mentés. Ezek segítségével többféle mentési stratégia kidolgozható.

Az **inkrementális mentés** során mindig az utolsó teljes mentés óta megváltozott adategységek kerülnek elmentésre. Ha egy adategység a legutóbbi teljes mentés óta valamikor megváltozott, akkor az minden inkrementális mentés alkalmával ismét mentésre kerül, egészen a következő teljes mentésig. Ez a mentési módszer gyorsabb a teljes mentésnél, és kevesebb helyet is kíván, azonban a differenciális mentésnél lassabb, és a tárigénye is nagyobb. Előnye, hogy visszaállításkor csak a legutóbbi teljes mentésre és a legutóbbi inkrementális mentésre van szükség, a korábbiakra nincs.

A **differenciális mentés** során az utolsó részleges vagy teljes mentés óta megváltozott adategységek kerülnek elmentésre. Ha két teljes mentés között több differenciális mentést végzünk, akkor pl. a második differenciális mentés csak az első differenciális mentés óta történt változásokat fogja rögzíteni. Ennek köszönhetően maga a mentés folyamata gyorsabbá válik, és esetenként kevesebb helyet foglal el. Hátránya azonban, hogy a visszaállításhoz a legutolsó teljes mentésre, és az azt követő összes differenciális mentésre szükség van [4].

A következőkben a magas fokú rendelkezésre állás megoldási módszereit tekintjük át.

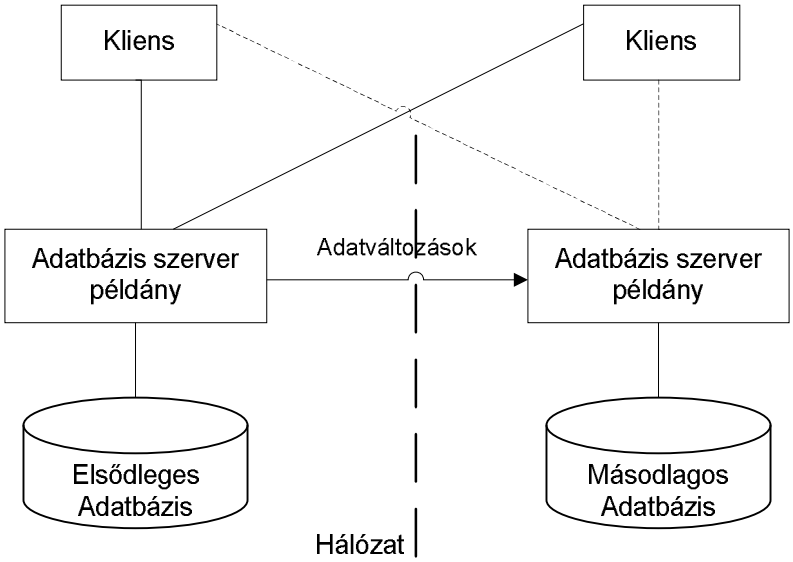
Adatbázis szerverek fürtözése (database cluster) a szerverpéldányok meghibásodása ellen nyújt védelmet, az adatbázisok adatait tároló periféria meghibásodása ellen nem. Fürtözés esetén az adatbázis szerverpéldányokat (csomópontok, node-ok) kapcsolunk össze privát hálózaton keresztül, ezek fizikailag nincsenek egymástól messze és ugyanazt az adatbázis állományt érik el, mely külön periférián, diszken helyezkedik el. Ha az egyik csomópont meghibásodik, egy másik veszi át a szerepét. Az adatbázis szerver fürtözésének alapja egy üzemelés figyelő („létfenntartó") szolgáltatás, a szívhang (heartbeat), mely a fürtben található csomópontok egészségi állapotát ellenőrzi. Ha a főkiszolgáló kiesik (meghibásodás vagy tervezett leállítás, pl. karbantartás miatt), akkor a kiszolgálást a másodkiszolgáló veszi át, az adatbázis réteg listener szolgáltatását értesítvén arról, hogy mostantól ő az elsődleges kiszolgáló. Tehát, ha az alkalmazás rétegből kapcsolatot kezdeményeznek az adatbázis réteg felé, akkor az adatbázis listener szolgáltatás már a működő adatbázis szerverpéldányhoz irányítja a kapcsolatot. A főkiszolgáló visszaállítása után, az új adatok visszakerülnek rá, a szolgáltatást újra átveszi, míg a másodkiszolgáló készenlétben vár [5]. A következő ábra az adatbázisok fürtözésének felépítését szemlélteti:



3. ábra. Adatbázis szerverek fürtözése [6]

Adatbázisok tükrözése (database replication) során a fő cél az adatállományok sérülésének kivédése, az adatvesztés elkerülése, például katasztrófák hatására bekövetkezett veszteségek esetén. A rendszer egy éles (elsődleges) és egy vagy több készenléti (másodlagos) adatbázis szerverből - szerverpéldányból és adatbázis állományból - áll, amik földrajzilag eltérő helyeken lehetnek, egymás közötti kommunikációjuk hálózati összeköttetés útján biztosított. A készenléti adatbázisokat kezdetben az elsődleges adatbázis biztonsági másolatából hozzák létre. A már létrehozott készenléti adatbázist a tükrözés automatikusan és folyamatosan szinkronban tartja az elsődleges adatbázissal, biztosítva, hogy tranzakció szinten az elsődleges adatbázis teljes mértékben konzisztens másolata maradjon. Ehhez az elsődleges adatbázis tranzakciós ismétlési adatait (az adatbázison elvégzett, még nem véglegesített műveleteket, Oracle rendszerben redo logokat) folyamatosan továbbítja a tartalék rendszernek, amely ezeket az ismétlési naplókat alkalmazza a készenléti adatbázis adataira.

Egyes tükrözési megvalósításokban van egy szemtanú szerver, mely figyeli, hogy az elsődleges adatbázis rendelkezésre áll-e. Amennyiben nem érhető el az elsődleges adatbázis, illetve adatbázis szerverpéldány a szemtanú automatikusan kezdeményezi a tüköradatbázis kinevezését elsődleges adatbázissá, az elsődleges adatbázist pedig átkapcsolja készenléti üzemmódba Az adatbázis-tükrözés szemtanú nélküli kiépítése esetén nincs automatikus átállás, azaz hiba esetén manuálisan kell ezt végrehajtani. Az adatbázis-tükrözéshez kliensoldali támogatás is tartozhat, az ügyfelek kapcsolati beállításai között megadhatjuk a tükörszerver nevét. Az elsődleges szerverrel való kapcsolat lezárását vagy elvesztését követően az újrakapcsolódás során az ügyfélalkalmazás az általunk megadott tükörszerverre kapcsolódik, amennyiben nem érhető el az elsődleges szerver [7] , [8] , [9] . A következő ábra az adatbázisok tükrözésének felépítését szemlélteti:



4. ábra. Adatbázis szerverek tükrözése [7]

A tükrözési technikák különböző adatvédelmi üzemmód alapján működhetnek, ugyanis egyes esetekben az adatvesztés elkerülése a fő cél, máskor viszont az adatbázis maximális teljesítménye a követelmény és a kisebb adatvesztések nem lényegesek.

Maximális védelem biztosítása esetén az adatváltozások azonnal továbbítódnak az elsődleges adatbázisból a készenléti adatbázisba, és az elsődleges adatbázisban a tranzakciók mindaddig nem véglegesítődnek (commit), amíg a változás adatai a készenléti adatbázisban rendelkezésre nem állnak. Ha hiba esetén a készenléti adatbázis leáll, az elsődleges adatbázisban is leáll a feldolgozás. Ez a működési mód biztosítja a legmagasabb szintű adatvédelmet.

Maximális rendelkezésre állás biztosítása esetén az adatváltozások azonnal továbbítódnak az elsődleges adatbázisból a készenléti adatbázisba, azonban ha a készenléti adatbázis elérhetetlenné válik (például mert megszakad a hálózati kapcsolat), a feldolgozás az elsődleges adatbázisban tovább folytatódik. A hiba elhárítását követően a készenléti adatbázis automatikusan újra szinkronizálódik az elsődleges adatbázissal.

Maximális teljesítmény biztosítása esetén az elsődleges adatbázis feldolgozza a tranzakciókat, de az adatváltozások aszinkron módon (késleltetve) továbbítódnak a készenléti adatbázishoz. Az elsődleges adatbázis véglegesítési mechanizmusa nem vár addig az írási műveletek elvégzésével, amíg a készenléti adatbázis visszaigazolja a változási adatok sikeres fogadását. Ha egy készenléti rendszer elérhetetlenné válik, a feldolgozás az elsődleges adatbázisban folytatódik, és az esemény gyakorlatilag nem befolyásolja az elsődleges adatbázis teljesítményét. Ez az üzemmód kevésbé szigorú adatvédelmet biztosít az elsődleges adatbázis számára, de nagyobb a teljesítménye, mint a maximális rendelkezésre állási üzemmódé.

SEBEZHETŐSÉGI PONTOK A MODELLEKBEN

A publikáció következő részében a fenti architektúrákban lévő, az adatbázis réteget érintő sebezhetőségeket vizsgáljuk meg. Adatbázis réteg fenyegetettségei alatt azokat a sérülékenységeket értjük, melyek közvetlenül az adatbázis réteg szolgáltatásait és adatbázis

állományait veszélyeztetik (Tehát például, ha egy webes támadásban a web szervert ellehetetlenítik és ezáltal a felhasználó az adatbázist nem tudja elérni, akkor ezt nem tekintjük adatbázis fenyegetésnek, mert a támadás után is kész az adatbázis szerver a kérések fogadására és kiszolgálására.)

A lehetséges támadási pontokat három csoportba sorolva gyűjtjük össze, nevezetesen a hálózat, a hosztok (szerverek, felhasználói számítógépek) és az alkalmazások felől [10], [2].

Hálózati infrastruktúra sebezhetőségei

Ebben a kategóriában az adatbázis rétegen belüli hálózat támadásait vizsgáljuk, illetve a teljes architektúrában lévő hálózati támadások közül azokat, melyek az adatbázis réteg szolgáltatásait zavarják meg. Az adatbázis rétegen belüli hálózat az adatbázis szerverek, illetve adatbázis állományokat tartalmazó háttértárak közötti kommunikációt jelenti.

A hálózati adatforgalom lehallgatásának veszélye megjelenik a magas rendelkezésre állás megvalósításánál (tükrözés, fürtözés), amikor is adatbázis szerverek kapcsolódnak hálózaton keresztül. Ha a közöttük lévő adatforgalom titkosítás nélkül (vagy gyenge titkosítással) működik, akkor a küldött adatok lehallgathatóak, ezzel pedig adatbázisok tartalma kerülhet illetéktelenül a támadók birtokába.

Hálózati szinten jelentkező támadások egyik fajtája a forráscím meghamisítása (spoofing). Ezzel a technikával beékelődéses támadást (man in the middle) lehet kiváltani, ami az adatbázis rétegen belül két, egymással kommunikáló adatbázis szerver esetén használható. A támadás következménye lehet az adatbázis tartalmának a megsértése, vagy az adatbázis szerverek szolgáltatásaiban való károkozás.

Hálózaton keresztül szolgáltatás megtagadása (DoS) típusú támadást lehet az adatbázis szerver ellen indítani, amivel a megtámadott szerver elérhetetlenné tehető. Konkrét példa a SYN csomagok elárasztásával (SYN flood attack) végrehajtott támadás [11].

Hosztok sebezhetőségei

A hosztok (a hálózatba kötött szerver és felhasználói számítógépek) sebezhetőségei alatt a rajtuk futó operációs rendszer és egyéb rendszerszoftverek (például IIS, .NET Framework, SQL szerver) sebezhetőségét értjük. Adatbázis réteg biztonságát vizsgálva feladatunk az adatbázis szervereket futtató platformok sebezhetőségeinek feltárása, a rajtuk lévő operációs rendszer és adatbázis szerver rendszerprogramok biztonságát vizsgálva.

Az adatbázis szerverekre vagy anonim kapcsolattal vagy autentikáció után lehet kapcsolódni. Adatbázis szerverre vagy az eggyel feljebb lévő rétegből vagy az adatbázis rétegen belülről (tükrözés, fürtözés esetén) érkezik a kapcsolat kezdeményezése. A támadó a hitelesítési mechanizmus és információk lehallgatásával, megszerzésével illetéktelenül tud az adatbázis szerverre bejutni. Rendszer szoftverek telepítésekor gyakran, automatikus módon, ismert nevű felhasználók jönnek létre, mely a támadás számára egy jó kiindulási pont. Ugyanis az ismert felhasználónévhez a támadónak csak a jelszót kell kitalálnia, ami gyakorta nem erős, azaz könnyen megfejthető.

Gyenge konfigurációs paraméterek használata, szoftver hibák (például puffer túlcsordulás, SQL injekció), felhasználói hibák, elavult verziójú programok használata és biztonsági frissítések feltöltésének hiánya kártékony kód lefuttatását, vírusok, trójaiak és férgek rendszerbe való bejutását eredményezheti, illetve szolgáltatás megtagadása típusú támadásra ad lehetőséget. Adatbázis kezelő rendszerek esetén a telepítéskor automatikusan létrejövő táblák, tárolt eljárások, alapértelmezett felhasználónevek és jelszavak sérülékenységi pontot jelentenek. (Célszerű ezeket törölni, és szükség esetén más névvel magunknak létrehozni.)

Rendszerszintű információk (például felhasználói adatok, operációs rendszer és adatbázis szerverprogram típusok és verziók, adatbázis szerver nevek, adatbázis séma részletei) felderítésével jelentős támadások készíthetők elő. A felderítésre használt eszközök közé tartoznak a port, illetve hálózaton elérhető hosztok szkennerei (ping sweep).

Pár évvel ezelőtt az adatbázis szerverek elleni új támadási vektor jelent meg, méghozzá az adatbázisok kommunikációs protokolljában rejlő sebezhetőségekre építve [12] . Az adatbázisok kommunikációs protokolljai szoftver gyártó függőek, magas szintű (TCP/IP réteg feletti) hálózati protokollok. Az SQL adatbázis lekérdező nyelv bizonyos kliens-szerver kommunikációhoz szükséges folyamatot egyáltalán nem definiál, például létfontosságú, de nem szabványosított a kliens kapcsolat (client session) létrehozása, a parancsok klienstől szerverhez való átadása, az adatok és a lekérdezés státuszának klienshez való eljuttatása. Ezeket a folyamatokat a gyártók által kifejlesztett, adatbázis kezelő rendszer hálózati kommunikációs szoftvere valósítja meg (például Oracle esetében az Oracle Net), melynek kódjai általában nem nyilvánosak, de számos sebezhetőséget hordoznak magukban. Bejelentett kommunikációs protokoll sebezhetőségek alapultak az üzenet struktúrájának elrontására [13] , mező méret megváltoztatására [14] , mező tartalmának manipulálására [15] , illetve üzenet sorszámának meghamisítására.

Jogosultságok helytelen beállításával, audit és mentési adatok nem megfelelően védett tárolásával, érzékeny (minősített vagy személyes) adatok titkosítás nélküli tárolásával bizalmas információk kerülhetnek illetéktelen kezekbe, ami szintén sérülékenységi pont a rendszerben.

Alkalmazások sebezhetőségei

Az alkalmazásokban rejlő sebezhetőségek, programozási hiányosságok az adatbázis réteg támadásainak egy fontos csoportját jelentik. A következő sérülékenységi kategóriákat különböztetjük meg.

A felhasználói inputok ellenőrzésének hiánya puffer túlcsordulásos, SQL injekciós, illetve XSS (cross-site scripting) támadásra adhat lehetőséget. A puffer túlcsordulásnál a szoftver egy fix hosszúságú tömböt (puffert) foglal le a memóriában, majd a tömb írásakor nem ellenőrzi annak határait. A támadó a lefoglalt tömböt túlírva (túl hosszú bemenet segítségével) felülírhat a program működése szempontjából lényeges memóriarészeket, így kártékony kódokat futtathat le.

SQL injekciós támadásnál az alkalmazás által előállítandó, tervezett tartalmú, dinamikusan szerkesztett SQL utasításba illesztenek káros tevékenységeket megvalósító kódot. Az alkalmazás a felhasználótól bekért paraméterek segítségével állítja elő az SQL szerverhez eljuttatandó lekérdezést. A támadó a paraméter értékének olyan kártékony karaktersorozatot ad meg, ami megváltoztatja az eredeti lekérdezés szintaktikáját, ezáltal az egészen más feladatot valósít meg, mint az eredeti elképzelés. [16]

XSS támadás jelenti napjainkban a webes alkalmazások leggyakoribb megsértését. A támadó a felhasználó böngészőjében futtathat le tetszőleges kódot (például javaszkriptet), miközben a felhasználó egy megbízható webhelyhez kapcsolódik. Igazából a felhasználót sebzi meg a támadás, a web alkalmazás, mint közvetítő közeg segítségével. A támadó munkafolyamat (session) azonosítókat tud a felhasználó gépéről megszerezni, ezáltal a felhasználó nevében be tud lépni az alkalmazáson keresztül az adatbázis rendszerbe. A session azonosítók megszerzésére építő támadást munkamenet-eltérítésnek (session hijacking) nevezik. Az XSS támadás különösen veszélyes az adatbázis rétegre nézve, ha a támadáshoz használt web alkalmazás a felhasználó által megadott adatokat az adatbázis rétegen belül tárolja, ekkor a támadó az adatbázisba tud illetéktelenül beleírni. [17]

Alkalmazások a felhasználók számára megjelenített hibaüzeneteikben adatbázisra, illetve az adatbázis szerverre jellemző információkat fedhetnek fel, ami támadások felépítéséhez ad segítséget, ezáltal sérülékenységi pontot jelent [16] . Célszerű a kliens felületen megjelenő adatbázis szerver hibaüzeneteinek letiltása és helyettesítése egy felhasználóbarát, általános üzenettel. Az adatbázis üzenetek információt tartalmaznak az adatbázisról, amiket a támadó ügyesen ki tud használni a sikeres támadáshoz szükséges bemeneti paraméterek elkészítéséhez. Az üzenetek letiltását egy egyszerű konfigurációs beállítással lehet elérni.

Az alkalmazás réteg egy konkrét alkalmazása az adatbázis szervert egyetlen adatbázis felhasználó nevében éri el. Ha az alkalmazás tulajdonosként vagy superuserként csatlakozik az adatbázishoz, sérülékenységet jelent, mivel bármilyen utasítást és lekérdezést lefuttathat, pl. a szerkezeti módosítást (táblák megszüntetése) vagy táblák komplett törlése. Mindig a lehető legkevesebb jogosultsággal rendelkező, az alkalmazás számára önálló és testreszabott felhasználókat kell használni. Ekkor, ha a behatoló meg is szerez valamilyen jogosultságot (hitelesítési információt), akkor is csak akkora változást tud okozni, mint az alkalmazás maga.

Az alkalmazás szintjén az adatbázis lekérdezések, hozzáférések naplózásának hiánya szintén sérülékeny pont lehet a rendszerben, hisz az adatbázis szerver naplóiban egyetlen, az alkalmazáshoz kötött felhasználó jelenik csak meg. Nyilvánvalóan a naplózás nem tud megakadályozni egyetlen ártalmas próbálkozást sem, de segítséget nyújthat annak felderítésében, hogy melyik alkalmazás és ki által lett kijátszva.

ÖSSZEFOGLALÁS

A cikk a kritikus adatbázisokat tartalmazó informatikai rendszerek felépítését és az adatbázis-biztonságot érintő kérdéseket tekintette át. A kapcsolódó fogalmak tisztázása után ismertetésre került a többrétegű architektúra modellje, melyben négy réteget, nevezetesen a megjelenítési réteget, a távoli elérés kiszolgáló réteget, az alkalmazás réteget és az adatbázis réteget különböztettük meg. Megvizsgáltuk az adatbázis réteg magas fokú rendelkezésre állásának megvalósításának módjait, a különböző mentési technikákat, az adatbázis rendszerek fürtözését és tükrözését.

A publikációban elemzésre kerültek a bemutatott architektúrák különböző pontjain jelentkező, az adatbázisok biztonságával kapcsolatos sebezhetőségek. Először meghatároztuk ezeket a teljes rendszer sebezhetőségein belül, majd a sérülékenységeket három kategóriába osztva elemeztük. A hálózati infrastruktúra támadásai közt az adatbázis rétegen belüli hálózat támadásait vizsgáltuk,

illetve a teljes architektúrában lévő hálózati támadások közül azokat, melyek az adatbázis réteg szolgáltatásait zavarják meg. A hosztok sebezhetőségeinek vizsgálatánál az adatbázis szervereket futtató számítógépek operációs rendszereinek és adatbázis szerver rendszerprogramjainak sérülékenységeit tanulmányoztuk. Végül az alkalmazásoknak az adatbázis-biztonságra kiható fenyegetéseit elemeztük.

FELHASZNÁLT IRODALOM

[1] Zakor Szilárd: Adatbázis-alapú alkalmazás-fejlesztés Borland Delphiben. Készletnyilvántartó program. Szakdolgozat, 2008 Debrecen http://ganymedes.lib.unideb.hu:8080/dea/bitstream/2437/4156/1/Szakdolgozat.pdf (2009.06.01.)

[2] http://msdn.microsoft.com/en-us/library/aa302420(printer).aspx (2009.06.01.)

[3] J. D. Ullman, J. Widom: Adatbázisrendszerek – Alapvetés, Második, átdolgozott kiadás. Panem kiadó, Budapest, 2009.

[4] http://www.biztostu.hu/mod/resource/view.php?id=530 (2009.06.01.)

[5] Göndör Gábor, Verseczki Roland: Több számítógép összekapcsolásának főbb módjai. A Clusterek http://architekturak.elte.hu/html/anyagok/06072/tobb_szgep_gondor_verseczki.pdf (2009.06.01.)

[6] Oracle® Real Application Clusters Administration and Deployment Guide 11*g* Release 1 (11.1) http://download.oracle.com/docs/cd/B28359_01/rac.111/b28254/admcon.htm (2009.09.09.)

[7] http://www.oracle.com/global/hu/database/collaterals/DATABASE10GR2_DATAGUARD_HUN.pdf (2009.06.01.)

[8] Kovács Zoltán: Új funkciók, kevesebb hiba http://download.microsoft.com/download/8/f/7/8f75bdd7-a0f9-4f53-a0ad-9d9aae86e21e/44-46.pdf (2009.06.01.)

[9] http://download.microsoft.com/download/d/9/4/d948f981-926e-40fa-a026-5bfcf076d9b9/ReplicationAndDBM.docx (2009.06.01.)

[10] http://msdn.microsoft.com/en-us/library/aa302418(printer).aspx (2009.06.01.)

[11] Gyányi Sándor: DDoS támadások veszélyei és az ellenük való védekezés, Robothadviselés 7. Tudományos Konferencia, Budapest, 2007.11.27 http://www.zmne.hu/hadmernok/kulonszamok/robothadviseles7/gyanyi_rw7.html (2009.06.01.)

[12] Amichai Shulman: Danger From Below: The Untold Tale of Database Communication Protocol Vulnerabilities http://www.imperva.com/resources/adc/db_comm_protocol.html

(2009.06.01.)

[13] http://www.securityfocus.com/archive/1/445298 (2009.06.01.)

[14] http://www.securityfocus.com/bid/2941 (2009.06.01.)

[15] http://www.kb.cert.org/vuls/id/871756 (2009.06.01.)

[16] Fleiner Rita: SQL injekcióra épülő támadások és védekezési lehetőségek, Hadmérnök, 2008 (III.)/4. (117-128.o.) http://hadmernok.hu/archivum/2008/4/2008_4_fleiner.html (2009.06.01.)

[17] Simon Whatley: What is a SQL Injection Attack http://www.simonwhatley.co.uk/what-is-a-sql-injection-attack (2009.06.01.)