

KIALAKULÓ TECHNOLÓGIÁK

Míg a szoftverfejlesztés alapjai igen stabil lábakon állnak, addig a szoftverfejlesztési folyamatot támogató technológiák állandóan változnak. A kutatólaboratóriumok megközelítési módszerek széles sorát fejlesztették ki, melyek lassan a gyakorlatban is érvényre jutnak. Ebben a részben három, általam különösen fontosnak ítélt megközelítést ismertetek. Hiszem, hogy a jövőben ezek a megközelítések jelentős hatással lesznek a szoftverfejlesztésre.

A három megközelítés a fejlesztés eltérő szakaszaiban jelentkezik. A *biztonsági tervezés*, amiről a 30. fejezet szól, már megjelent a rendszerfejlesztés gyakorlatában. A *szolgáltatásorientált tervezés* még mindig gyerekcipőben jár, de a mögötte álló tekintélyes beruházások arra engednek következtetni, hogy a közeljövőben fontos szoftverfejlesztési technológiává válik. Az *aspektusorientált tervezés* kevésbé érett, de tartogat néhány olyan izgalmas elképzelést, amelyek gyökeresen átalakítanák a szoftverek szerkezetét.

A könyv hetedik része a következő fejezeteket tartalmazza:

1. A 30. fejezet a biztonsági tervezéssel foglalkozik. Úgy találtam, hogy ez a téma külön fejezetet érdemel, mert sajnos a rendszereket veszélyeztető támadások száma egyre nő. A fejezet a kritikus rendszerek tervezéséről szóló anyag kiegészítésének is tekinthető. Ebben a fejezetben alapvető biztonsági fogalmakat ismertetek, bemutatom a biztonsági követelmények kockázatelvű meghatározását, tervezési útmutatót nyújtok biztonságos rendszerek készítéséhez, és bevezetem a rendszerek túlélőképességét szem előtt tartó tervezés fogalmát.
2. A 31. fejezet témája a szolgáltatásorientált szoftverfejlesztés. Ez a 12. fejezetben tárgyalt szolgáltatásorientált architektúrákról szóló rész kibővítése. A szolgáltatást mint újrafelhasználható komponenst mutatom be, és megtárgyalom a szolgáltatások tervezésének és implementálásának folyamatát (szolgáltatásfejlesztés). Ezután elmagyarázom, hogyan lehet szoftvereket szolgáltatásokból mint alapvető építőelemekből kifejleszteni.
3. A 32. fejezet az aspektusorientált szoftverfejlesztéssel foglalkozik. A szoftverfejlesztés ezen új megközelítése a vonatkozások szétválasztására fekteti a hangsúlyt. Mivel a megközelítés még kísérleti fázisban van, a főbb elvek bemutatását és az aspektusorientált szemléletnek a szoftverfolyamat különböző szintjein történő alkalmazását helyeztem előtérbe. A fejezet során kitérek az aspektusorientált programok széles körű elterjedésének egyik fontos akadályára: az aspektusorientált programok szisztematikus tesztelésének nehézségeire.

30 BIZTONSÁGI TERVEZÉS

TÉMAKÖRÖK

Ez a fejezet olyan kérdéseket mutat be, amelyeket figyelembe kell vennünk a biztonságos szoftverek specifikációjánál és tervezésénél. A fejezet elolvasása után az alábbiakkal leszünk tisztában:

- megértjük a biztonsági kockázatkezelés jelentőségét, valamint hogy miként használhatjuk a kockázatelemzést a biztonsági követelmények meghatározásánál;
- tisztában leszünk azzal, hogy a biztonsági megfontolások hogyan befolyásolhatják a rendszer-architektúrák tervezését és javíthatják a biztonságos rendszerek tervezésének minőségét;
- megértjük a rendszer túlélőképességének fogalmát, valamint hogy miért fontos a túlélési elemzés az összetett szoftverrendszerek esetében.

TARTALOM

- 30.1. Biztonsági fogalmak
- 30.2. Biztonsági kockázatkezelés
- 30.3. Tervezés a biztonság szempontjából
- 30.4. A rendszer túlélőképessége

Az internet rohamos terjedése az 1990-es években új kihívással szembesítette a szoftvertervezőket: olyan rendszereket kellett tervezniük és implementálniuk, amelyek biztonságosak voltak. Az internethez csatlakozó rendszerek száma egyre nőtt, ezzel együtt pedig megjelentek a rendszereket fenyegető külső támadások. Ez megnehezítette a megbízható rendszerek készítését. A szoftvertervezőknek már nemcsak a fejlesztési folyamatba becsúszó esetleges hibákkal kellett számolniuk, hanem a technikailag szakképzett, rosszindulatú támadók által jelentett veszélyeket is figyelembe kellett venniük.

Ma már elengedhetetlen úgy megtervezni a rendszereket, hogy azok állják a külső támadásokat, és helyre tudjanak állni egy-egy ilyen ártó kísérlet után. Biztonsági óvintézkedések híján szinte biztos, hogy a hálózati rendszer veszélybe kerül. A támadók kárt tehetnek a rendszer hardverében, bizalmas adatokat sze-

rezhetnek meg, vagy akadályozhatják a rendszer által nyújtott szolgáltatásokat. Nyilvánvaló tehát, hogy a biztonsági tervezés a szoftverfejlesztési folyamat egyre fontosabb részévé válik.

A biztonsági tervezés olyan rendszerek fejlesztésével és karbantartásával foglalkozik, amelyek ellenállnak a számítógép-alapú rendszerek és a bennük tárolt adatok megrongálására irányuló támadásoknak. A biztonsági tervezés az átfogóbb számítógépes biztonság témakörébe tartozik. Mivel egyre több bűnöző próbálja illegális célokból kijátszani a hálózati rendszereket, a biztonság elsődleges szempont a vállalkozások és az egyének számára. A szoftvertervezőknek tisztában kell lenniük a rendszerek biztonságát veszélyeztető tényezőkkel, és tudniuk kell ezeket a veszélyeket semlegesíteni.

Ebben a fejezetben az a célom, hogy megismertessem a szoftvertervezőket a biztonsági tervezéssel, kiemelve az alkalmazás biztonságának specifikációját és tervezését. A bemutatott fogalmak és elvek elősegítik a biztonságos rendszerek fejlesztését. Mindazonáltal nem áll szándékomban a számítógépes biztonságot a maga teljességében leírni, éppen ezért nem szólok például a titkosításról, a hozzáférések szabályozásáról vagy a hitelesítési mechanizmusokról. Ezekkel a témákkal több általános írás is részletesen foglalkozik (Anderson, 2001; Pfleeger és Pfleeger, 2003; Bishop, 2005).

Ez a fejezet kiegészíti a más fejezetekben tárgyalt biztonsági témákat. Az itt szereplő anyagot az alábbi alfejezetekkel együtt olvassuk:

- 3.5. alfejezet, amely a biztonságot üzembiztonsági jellemzőként tárgyalja;
- 9.3. alfejezet, amely a biztonság specifikációját tárgyalja;
- 24.3. alfejezet, amely a biztonság kiértékelését tárgyalja.

A biztonsági kérdések megfontolásakor egyaránt figyelembe kell vennünk az alkalmazásszoftvert (a vezérlőrendszer, az információs rendszer stb.) és a rendszer alapját képező infrastruktúrát (30.1. ábra). Összetett alkalmazásoknál az infrastruktúra magában foglalja az operációs rendszert (például Linux vagy Windows), a rendszeren futó egyéb általános alkalmazásokat (például böngésző és e-mail kliensek), az adatbázis-kezelő rendszert, az elosztott számításokat és az adatbázis-elérést lehetővé tevő köztes szoftvert, valamint az alkalmazás által használt újrafelhasználható komponenseket tartalmazó könyvtárakat. A valóságban a legtöbb támadás a rendszer infrastruktúráját célozza meg, mivel az közismert és széles körben elterjedt komponenseket (például böngészők) tartalmaz.

Alkalmazás
Újrafelhasználható komponensek és könyvtárak
Köztes szoftver
Adatbázis-kezelő
Általános, megosztott alkalmazások (böngészők, e-mail stb.)
Operációs rendszer

30.1. ábra. A rendszer sebezhető rétegei

A gyakorlatban fontos különbséget kell tennünk az alkalmazás biztonsága és az infrastruktúra biztonsága között:

1. Az alkalmazásbiztonság szoftvertervezési probléma, azaz a szoftvertervezők feladata, hogy olyan rendszert *tervezzenek*, amely ellenáll a támadásoknak.
2. Az infrastruktúra biztonsága rendszerkezelői probléma, azaz a rendszerkezelők feladata, hogy az infrastruktúrát ellenállónak *konfigurálják*. A rendszerkezelőknek az infrastruktúra biztonsági adottságait legjobban kihasználva kell felállítaniuk az infrastruktúrát. Ezenfelül ki kell javítaniuk a szoftver használata során sérülékenynek bizonyuló pontokat.

A biztonságkezelés nem egyetlen feladatot jelent, hanem tevékenységek sorából áll, mint például a felhasználók és az engedélyek kezelése, a rendszerszoftver telepítése és karbantartása, továbbá a támadások figyelése, felderítése és elhárítása:

1. A felhasználók és az engedélyek kezelése magában foglalja a felhasználók törlését és a rendszer bővítését új felhasználókkal, megfelelő hitelesítő mechanizmusok használatát, valamint az engedélyek kiosztását a rendszeren belül, ami biztosítja, hogy a felhasználók csak a számukra szükséges erőforrásokhoz férhetnek hozzá.
2. A rendszerszoftver telepítése és karbantartása magában foglalja a rendszerszoftver és a köztes szoftver telepítését és ezek helyes konfigurálását a sérülékenység elkerülése érdekében. Szintén ide tartozik a szoftver rendszeres frissítése és a legújabb hibajavító csomagok telepítése, amelyek kijavítják az eddig feltárt biztonsági hibákat.
3. A támadások figyelése, felderítése és elhárítása olyan tevékenységeket foglal magában, amelyek figyelik a rendszer jogosulatlan használatát, felderítik és kivédik a támadásokat, továbbá biztonsági mentéseket készítenek, hogy egy külső támadás után vissza lehessen térni a rendszer normál működéséhez.

A biztonságkezelés létfontosságú a biztonságos rendszerek karbantartásánál. A szoftvertervezők számára azonban mégsem a kezelési folyamatok okoznak fejtörést, hanem az, hogy miként tervezzenek olyan alkalmazásrendszereket, amelyek támogatják a biztonságkezelést. Ezzel a kérdéssel a 30.3.3. alfejezetben foglalkozom.

30.1. BIZTONSÁGI FOGALMAK

A 3. fejezetben már ismertettem néhány alapvető biztonsági fogalmat és terminológiát. Ezek láthatók a 30.2. ábrán, ami a 3.9. ábra kissé módosított változata. A fogalmak jobb megértése érdekében tekintsük a következő helyzetet.

Egy idegklinika információs rendszerében nyilvántartják a klinikán kezelt betegek személyes adatait. Mivel a mentális betegségek igen kényes témának számítanak, elengedhetetlen, hogy a páciensek adatait bizalmasan kezeljék. A rendszer több kórházból és klinikáról is elérhető, ezért a felhasználók webes felületen keresztül jelentkezhetnek be egy megfelelő azonosító és jelszó beírásával. A jelszóval kapcsolatban a rendszer egyetlen megkövetése annyi, hogy az legalább nyolc karakter hosszú legyen.

Egy bűnözőnek tudomására jut, hogy egy közismert sportolót mentális problémákkal kezelnek az említett klinikán. A bűnöző illegális úton hozzá szeretne jutni a rendszerben tárolt adatokhoz, hogy megszarolhassa a sportolót. Az aggódó rokon szerepét eljátszva szóba elegyedik az ápolókkal, és kifigyeli, hogyan lehet a rendszerbe belépni. Eközben a dolgozók kitűzőiről leolvassa a neveket, majd ezeket azonosítóként használva és jelszavakat tippelve megpróbál belépni a rendszerbe.

A 30.3. ábra megmutatja, hogyan kapcsolódnak a 30.2. ábrán szereplő fogalmak az előző példában felvázolt klinikai információs rendszerhez.

A biztonsági fenyegetések három fő csoportba sorolhatók:

1. A rendszer és az adatok *bizalmasságát* érintő fenyegetések. Ezek olyan személyeknek vagy programoknak adhatnak ki információt, akik (amik) ezen információk megszerzésére nem lennének jogosultak.
2. A rendszer és az adatok *integritását* érintő fenyegetések. Ezek a fenyegetések károsíthatják, vagy megghamisíthatják a szoftvert vagy az adatokat.
3. A rendszer és az adatok *rendelkezésre állását* (elérhetőségét) érintő fenyegetések. Ezek a fenyegetések korlátozhatják a rendszerhez való hozzáférést.

Ezek a fenyegetések természetesen nem függetlenek egymástól. Ha egy támadás következtében a rendszer rendelkezésre állása sérül, nem tudjuk frissíteni az

Fogalom	Leírás
Vagyon	Értékes és védelmet igénylő rendszererőforrás.
Függőség	Egy sikeres támadás következtében keletkező kár vagy veszteség. Ez lehet adatsérülés vagy adatvesztés, esetleg idő- és energiaveszteség, amennyiben a rendszer helyreállítást igényel.
Sérülékenység	Egy számítógép-alapú rendszer gyenge pontja, amelyet kihasználva kár vagy veszteség okozható.
Támadás	A rendszer sérülékenységének kihasználása. Általában a rendszeren kívülről érkezik és ártó szándékú.
Fenyegetések	Olyan körülmények, amelyek veszteséget vagy kárt okozhatnak. Ilyen például, ha a rendszer egyik sérülékeny pontja támadásnak van kitéve.
Ellenőrzés	A rendszer sérülékenységét csökkentő védelmi mérték. Egy gyenge hozzáférés-vezérlő rendszer sérülékenységét csökkentő ellenőrzés például a biztonsági kódolás.

30.2. ábra. Biztonsági fogalmak

Fogalom	Leírás
Vagyon	A klinikán (valaha) kezelt betegek adatai.
Függőség	Anyagi veszteség a jövőbeli páciensek részéről, akik más egészségügyi intézményt választanak, mert nem merik a klinikára bízni adataikat. Anyagi veszteség a sportoló által indított per miatt. A hírnév elvesztése.
Sérülékenység	Gyenge jelszókezelő rendszer, amely megengedi, hogy a felhasználók könnyen kitalálható jelszavakat adjanak meg. Valódi nevekkal megegyező felhasználói azonosítók.
Támadás	Valaki jogosult felhasználónak adja ki magát.
Fenyegetések	Jogosulatlan felhasználó hozzáférhet a rendszerhez egy jogosult felhasználó bejelentkezési nevét és jelszavát kitalálva.
Ellenőrzés	Olyan jelszóellenőrző rendszer, amely tiltja a tulajdonnevek és a szótárban általában megtalálható szavak jelszóként történő megadását.

30.3. ábra. Példák biztonsági fogalmakra

időközben megváltozott adatokat, ami a rendszer integritásának sérülését jelenti. Ha egy sikeres támadás károsítja a rendszer integritását, előfordulhat, hogy a rendszert le kell állítani a probléma kijavítása miatt. Eközben a rendelkezésre állás, az elérhetőség csökken.

A rendszer biztonságosságát növelő ellenőrzéseket az alábbi három osztályba sorolhatjuk:

1. A támadás sikertelenségét biztosító ellenőrzések. Ennél a stratégiánál eleve úgy tervezzük a rendszert, hogy elkerüljük a biztonsági problémákat. Az igen érzékeny katonai rendszereket például nem kötik nyilvános hálózatokhoz, így a külső hozzáférés kizárt. A kódolás is az elkerülés elvén alapuló ellenőrzőeszköz, ugyanis a támadó nem tudja elolvasni az esetleg megszerzett adatot. A kód feltörése pedig a gyakorlatban nagyon drága és időigényes.
2. A támadások felderítését és visszaverését segítő ellenőrzések. Ezeknél az ellenőrzéseknél olyan funkcionalitást építünk a rendszerbe, amely figyeli a rendszer működését, és jelzi, ha szokatlan tevékenységet észlel. Ilyenkor védelmi lépéseket tehet, például leállíthatja a rendszer egyes részeit vagy bizonyos felhasználókra korlátozhatja a hozzáférést stb.
3. A rendszer helyreállítását támogató ellenőrzések. Ezek az ellenőrzések az automatikus mentési stratégiáktól egészen a biztosításokig terjedhetnek, amelyek fedezik a sikeres támadást követő helyreállítás költségeit.

A biztonsági tervezés az általánosabb üzembiztonsági tervezési folyamat része, ezért sok hasonlóságot mutat a biztonságossági és a megbízhatósági tervezéssel. Az alapvető stratégiák a problémák elkerülésére, felderítésére és a probléma utáni helyreállításra épülnek. A jó rendszerek nemcsak egyetlen stratégiára támaszkodnak, hanem azoknak jól megfontolt ötvözetét használják.

30.2. BIZTONSÁGI KOCKÁZATKEZELÉS

A biztonsági kockázatok kiértékelése és kezelése lényegbevágó a hatékony biztonsági tervezésnél. A kockázatkezelés szerepe a rendszer vagyona ellen intézett támadásokból származó veszteségek kiértékelése, valamint a veszteségek és a veszteségeket redukáló biztonsági eljárások költségének egyensúlyban tartása. A hitelkártya-társaságok állandóan ezt teszik. Új technológiát nem nehéz bevezetni a hitelkártyacsалások csökkentésére, a technológia költsége azonban meghaladná a hitelkártya-használók kárának megtérítését. A költségek csökkenésével és a támadások növekedésével a mérleg billenhet. Manapság például a hitelkártya-társaságok mágneses csík helyett egy beépített chipbe kódolják az információt, ezzel nagymértékben megnehezítve a kártyák másolását.

A kockázatkezelés inkább üzleti, semmint technológiai kérdés, így nem a szoftvertervezők feladata, hogy eldöntsék, milyen ellenőrzéseket tartalmazzon egy rendszer. A felső vezetés dönti el, hogy a biztonság kialakítása vagy a biztonság hiányából származó károk megtérítése a kifizetődőbb. Mindazonáltal a szoftvertervezők feladata, hogy naprakész technikai tanácsokat és véleményt nyújtsanak biztonsági kérdésekben, így aktív szereplői a kockázatkezelési folyamatnak.

A kockázatkértékelési és -kezelési folyamat kritikus összetevője a szervezeti biztonságpolitika, ami az összes rendszerre érvényes, és meghatározza, mi megengedett, és mi nem. Például a katonai biztonságpolitika egyik elve a következőképpen szólhat: „Az olvasók csak azokat a dokumentumokat tekinthetik meg, amelyek az olvasó kezelői szintjével megegyező vagy annál lejjebb lévő osztályba tartoznak.” Ez azt jelenti, hogy ha az olvasó „titkos” kezelői szinttel rendelkezik, akkor hozzáférhet a „titkos”, „bizalmas” vagy „nyílt” dokumentumokhoz, a „szigorúan titkos” osztályba tartozó iratokhoz viszont már nem.

A biztonságpolitika megszabja a biztonsági rendszertől elvárt feltételeket, ezáltal segít a lehetséges fenyegetések azonosításában. Kockázatnak számít bármi, ami veszélyeztetheti az üzleti biztonságot. Elvben, amint azt Bishop (2005) kifejti, a biztonsági irányelvek formálisan is leírhatók és automatikusan ellenőrizhetők. Ezzel szemben a gyakorlatban ezek legtöbbször informális dokumentumokban jelennek meg, amelyek rögzítik, hogy mit szabad, és mit nem.

A kockázatkértékelés még a rendszer beszerzéséről született döntés előtt elkezdődik, és végig kell kísérnie a szoftverfejlesztési folyamatot. Fontos tényező a rendszerről rendelkezésre álló információ, ezért a kockázatkezelés szakaszokra osztható:

1. *Előzetes kockázatkértékelés.* Ezen a szinten még nem történt döntéshozatal a részletes rendszerkövetelményekkel, a rendszer tervezésével vagy az implementációs technológiával kapcsolatban. A kiértékelési folyamat célja ebben a szakaszban először is annak kiértékelése, hogy a rendszer által nyújtott előnyök megéri-e a kapcsolódó kockázatokat, majd a megvalósítandó rendszerrel szembeni biztonsági követelmények meghatározása. Nincs információnk a rendszer potenciálisan sérülékeny pontjairól vagy az újrafelhasznált rendszerkomponensekben és a köztes szoftverben található ellenőrzésekről.

2. *Kockázatkéértékelés az életciklusban.* Ez a kockázatkéértékelés a rendszerfejlesztési életciklus során történik, és figyelembe veszi a rendszer tervezésével és implementációjával kapcsolatos döntéseket. A kéértékelés tájékoztatja a fejlesztési folyamatot a biztonsági követelmények tervezéséről. Az ismert és a potenciálisan sérülékeny pontok azonosítása során szerzett információt felhasználják a rendszer funkcionalitásáról, implementációjáról, teszteléséről és alkalmazásáról hozott döntéseknél.

Míg a folyamatok részletei szervezetről szervezetre változnak, bizonyos alaptevékenységek minden esetben jelen vannak. Ezekről szól a következő két alfejezet.

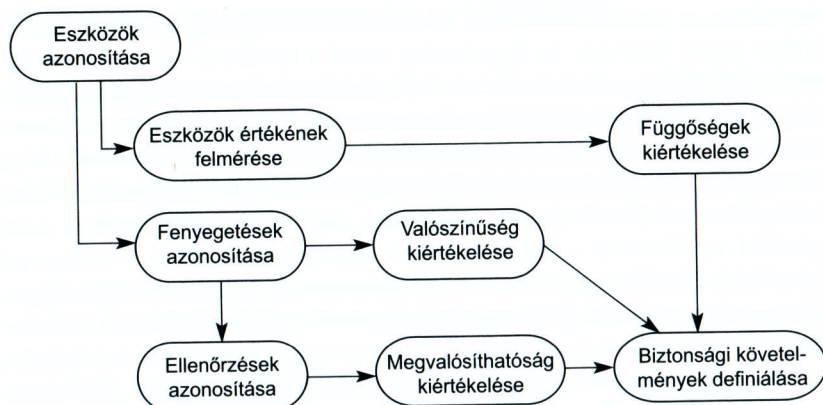
30.2.1. Előzetes kockázatkéértékelés

Az előzetes kockázatkéértékelés feladata a teljes rendszerre, nem csak a szoftverre vonatkozó biztonsági követelmények meghatározása. A követelmények befollyásolják a platform és a köztes szoftver kiválasztását, és alapként szolgálnak a szoftver részletesebb funkcionális követelményeinek meghatározásához.

Az előzetes kockázatkéértékelés legfontosabb szakaszai (30.4. ábra) a következők:

1. Eszközök azonosítása, melynek során azonosítjuk a védelmet igénylő rendszereszközöket. Maga a rendszer vagy bizonyos rendszerfunkciók is lehetnek eszközök, akárcsak a rendszerhez kötődő adatok.
2. Eszközök értékenek felmérése, melynek során megbecsüljük az azonosított eszközök értéket.
3. Függőségek kéértékelése, melynek során felmérjük az egyes eszközökhöz kapcsolódó potenciális veszteségeket.
4. Fenygetések azonosítása, melynek során azonosítjuk a rendszer eszközeit érintő fenygetéseket.
5. Valószínűség kéértékelése, melynek során megbecsüljük az egyes fenygetések valószínűségét.
6. Ellenőrzések azonosítása, melynek során javaslatokat teszünk az egyes eszközök védelmét szolgáló ellenőrzések bevezetésére.
7. Megvalósíthatóság kéértékelése, melynek során kéértékeljük a javasolt ellenőrzések megvalósíthatóságát és költségét.
8. Biztonsági követelmények definiálása, melynek során a függőségek, a fenygetések és az ellenőrzések kéértékeléséből egy sor rendszerbiztonsági követelményt származtatunk. A követelmények vonatkozhatnak a rendszer infrastruktúrájára vagy az alkalmazásrendszerre.

A folyamat illusztrálásához térjünk vissza a fejezet korábbi részében bemutatott klinika információs rendszeréhez. A könyv keretei nem engedik, hogy leírjam a teljes kockázatkéértékelést, ehelyett csak néhány példát mutatok. A 30.5.



30.4. ábra. Előzetes kockázatiértékelés

Vagyon	Érték	Függőség
Az információs rendszer	Magas. Biztosítania kell minden egyes klinikai konzultációt. Potenciálisan biztonságsgáskritikus.	Magas. Anyagi veszteség, mivel lehet, hogy egyes konzultációkat le kell mondani. A rendszer helyreállításának költségei. A beteg sérülése, ha nem kapja meg időben a kezelést.
A betegeket nyilvántartó adatbázis	Magas. Biztosítania kell minden egyes klinikai konzultációt. Potenciálisan biztonságsgáskritikus.	Magas. Anyagi veszteség, mivel lehet, hogy egyes konzultációkat le kell mondani. A rendszer helyreállításának költségei. A beteg sérülése, ha nem kapja meg időben a kezelést.
Egy páciens adatai	Általában alacsony, bár egyes kiemelt betegek esetében magas lehet.	Alacsony közvetlen veszteség, de a hírnév veszélybe kerülhet.

30.5. ábra. Vagyon elemzése egy előzetes kockázatiértékelési jelentésben

Fenyegetés	Valószínűség	Ellenőrzés	Megvalósíthatóság
Jogosulatlan személy belép a rendszerbe rendszerkezelőként, és elérhetetlenné teszi a rendszert	Alacsony	Csak meghatározott, fizikailag biztonságos helyekről lehessen a rendszert kezelni.	Alacsony költség, de ügyelni kell a kulcsok kiosztásánál, és biztosítani kell, hogy vészhelyzet esetén legyen tartalékkulcs.
Jogosulatlan személy egyszerű felhasználóként lép be a rendszerbe, és bizalmas információkhoz jut	Magas	A felhasználók biometrikus mechanizmusokkal történő azonosítása. Naplózni a páciensek adatainak változását a rendszer használatának nyomkövetéséhez.	Technikailag kivitelezhető, de igen magas költséggel. Várható a felhasználók ellenkezése. Egyszerű és könnyen implementálható, a helyreállítást is támogatja.

30.6. ábra. Fenyegetések és ellenőrzések elemzése egy előzetes kockázatiértékelési jelentésben

és a 30.6. ábrákon látható jelentésrészletek az előzetes kockázatkiértékelési folyamat eredményeképpen jöhetnek létre. Az így kapott előzetes kockázatelemzési jelentés a biztonsági követelmények meghatározásának egy bemenete.

A klinika információs rendszerének kockázatelemzéséből például az alábbi biztonsági követelmények származtathatók:

1. A betegek adatait minden klinikai munkamenet elején le kell tölteni az adatbázisból a dolgozók által használt rendszerkliens egy biztonságos területére.
2. A betegek adatait nem lehet a rendszerklienseken tárolni a munkamenet befejezése után.
3. Egy, az adatbázisszervertől független gépen naplózni kell az adatbázison végrehajtott minden egyes módosítást.

Az első két követelmény összefüggésben áll – a betegek adatai letöltésre kerülnek minden munkamenet elején, így a konzultációk akkor is folytathatók, ha támadás éri az adatbázisszervert. Másrészt viszont ezeket az adatokat törölni kell, hogy a klienst később használók ne tudjanak hozzáférni az információkhoz. A harmadik követelmény helyreállítási és felülvizsgálati szempontokat vesz figyelembe, azaz a változtatásokat vissza lehet állítani a napló segítségével, sőt az elkövetőre is fény derül.

30.2.2. Kockázatkiértékelés az életciklus során

Az előzetes és az életciklus során történő kockázatkiértékelés között az a lényeges különbség, hogy utóbbinál már ismert a rendszer architektúrája és az adatok szervezési módja. Megszülettek a közbeszerzési döntések, kiválasztották a rendszerplatformot és a köztes szoftvert. Előfordulhat, hogy egy adott fejlesztési stratégia (például „általános alkalmazás konfigurálása”) mellett döntöttek.

Mindez azt jelenti, hogy sokkal részletesebb információval rendelkezünk a védelmet igénylő részekről, és a rendszer sebezhetősége is ismertek lehetnek. Néhány sebezhető pont a kiválasztott módszerek velejárója (például minden jelszóalapú rendszer sérülékeny abban az értelemben, hogy egy jogos felhasználó elárulhatja a jelszavát egy jogosulatlan felhasználónak), de egyéb sérülékenységeket is jogunkban áll feltételezni.

A biztonsági kockázatkiértékelésnek a követelmények tervezésétől az alkalmazás telepítéséig minden egyes életciklusszakaszban jelen kell lennie. Ez a folyamat annyival több, mint az előzetes kockázatkiértékelés, hogy foglalkoznunk kell a rendszer sérülékeny pontjainak azonosításával és kiértékelésével. A sérülékenységek kiértékelése során azonosítjuk az adott sérülékeny ponthoz köthető vagyont és lehetséges rendszertámadásokat. A kockázatkiértékelés eredményeképpen egy sor tervezési döntés születik, amelyek kihatással vannak a rendszer tervezésére vagy implementációjára, esetleg korlátozzák annak alkalmazását.

Illusztrációként tegyük fel, hogy az egészségügyi szolgáltató úgy döntött, hogy egy készen kapható információs rendszert fog használni a klinikák beteg-

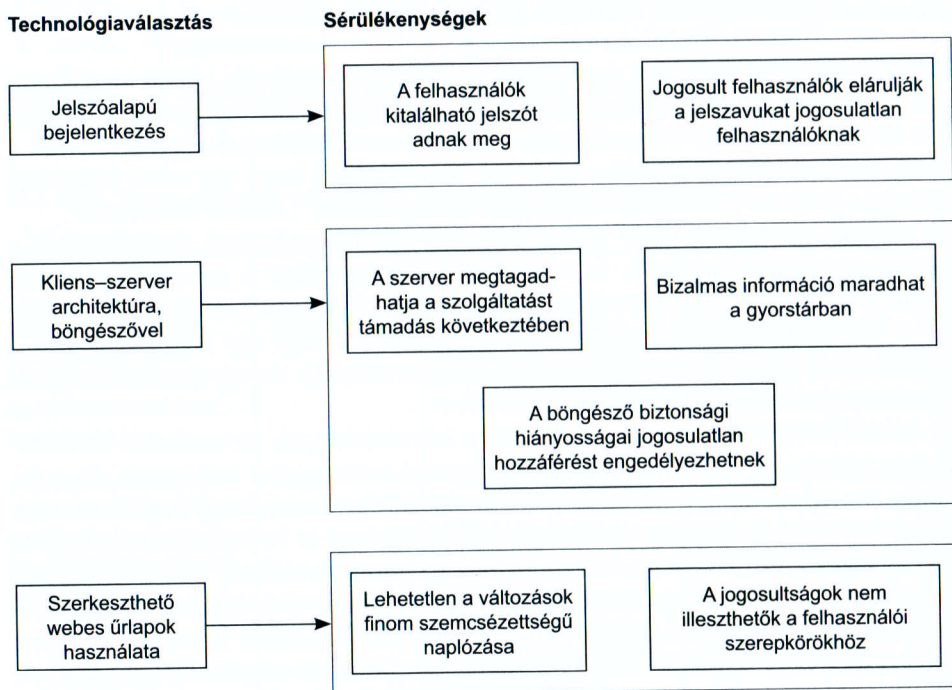
nyilvántartásához. A rendszert külön kell konfigurálni minden egyes klinika esetén. A döntést az motiválta, hogy látszólag ez a megoldás nyújtja a legolcsóbban és leggyorsabban a legszélesebb körű funkcionalitást.

Mikor egy meglévő rendszerre építjük az alkalmazásunkat, el kell fogadnunk a rendszer fejlesztői által hozott tervezési döntéseket. Tegyük fel, hogy a fejlesztők többek közt az alábbi döntéseket hozták:

1. A rendszer felhasználóinak azonosítása bejelentkezési név/jelszó kombinációval történik. A rendszer semmilyen egyéb felhasználóazonosító mechanizmust nem támogat.
2. A rendszer kliens-szerver architektúrájú: a kliensek szabványos böngészőn keresztül férhetnek hozzá az adatokhoz.
3. Az információ szerkeszthető webes űrlap formájában jelenik meg. Ez helyben szerkeszthető, a módosított információt el kell küldeni a szervernek.

Alapjában véve egyik sem mondható rossz döntésnek, egy életciklus során történő kockázatiértékelés azonban rámutatna a lehetséges sérülékeny pontokra. Ezekre láthatunk példákat a 30.7. ábrán.

Miután azonosítottuk a sérülékeny pontokat, döntéseket kell hoznunk a kockázatok csökkentésével kapcsolatban. Ez gyakran további rendszerbiztonsági követelmények meghatározását vagy a rendszer használatának leírását jelenti.



30.7. ábra. A technológiaválasztással járó sérülékenységek

Hely hiányában nem tudok kitérni a sérülékenységek kijavítását célzó összes lehetséges követelményre, de íme, néhány lehetséges példa:

1. Jelszóellenőrző program használata, amelyet mindennap futtatva kiszűrhetők és jelenthetők a rendszer szótárában is szereplő, ezáltal könnyen kitalálható jelszavak.
2. A rendszert csak azokról a kliensgépekről lehessen elérni, amelyeket a rendszer-adminisztrátorok jóváhagytak és regisztráltak.
3. A kliensgépeken csak egyetlen, a rendszer-adminisztrátorok által jóváhagyott böngésző telepítése megengedett.

Mivel kész rendszerről van szó, nem építhetünk be jelszóellenőrzőt, hogy biztosítsuk, a felhasználók nem kitalálható jelszavakat használjanak. Mindazonáltal az első követelmény jól mutatja, hogy a sérülékeny jelszavakat viszonylag gyorsan könnyű azonosítani, és lépéseket lehet tenni annak érdekében, hogy a felhasználók módosítsák a jelszavukat. A második és a harmadik követelmény azt jelenti, hogy a felhasználók mindig ugyanazon a böngészőn keresztül érik el a rendszert. A rendszer telepítésekor eldönthetjük, melyik böngészőt tartjuk a legbiztonságosabbnak. Emellett a böngésző biztonsági frissítése is könnyebb, mert nem kell különböző rendszerekkel foglalkoznunk.

30.3. TERVEZÉS A BIZTONSÁG SZEMPONTJÁBÓL

Amint azt a 3. fejezetben is kifejtettem, a biztonság a rendszer eredendő tulajdonsága, és mint ilyen, nem lehet egyszerűen a rendszerhez hozzáadni, hanem a rendszer részeként kell megtervezni, még az implementáció előtt. A biztonság természetesen implementációs kérdés is, mivel sérülékeny pontok a szoftver programozása közben is keletkezhetnek. Jegyezzük meg azonban, hogy egy jó terv biztonságosságát el lehet rontani hibás implementációval, de egy gyatra terv sohasem válhat biztonságossá az implementációkor.

Ebben az alfejezetben főként a rendszer tervezésekor felmerülő biztonsági kérdésekkel szeretnék foglalkozni, mivel ez a téma nem kap kellő figyelmet a számítógépes biztonságról szóló könyvekben. A biztonsági kérdések nyilván a szoftverfolyamat többi szakaszában is fontosak:

1. A 9. fejezetben részleteztem a különböző típusú biztonsági követelményeket, jelen fejezet 30.2. alfejezetében pedig elmagyaráztam, hogyan származtathatunk biztonsági követelményeket a kockázatelemzésből. Javaslom Firesmith tanulmányát (Firesmith, 2003), ami kitűnő bevezetés a biztonsági követelmények tervezésébe.
2. Az implementációs kérdések és hibák nagymértékben befolyásolják a biztonságot, de ezek gyakran az alkalmazott technológiától is függenek. Javaslom Viega és McGraw könyvét (Viega és McGraw, 2002), ami jó bevezetést nyújt a biztonságos programozásba.

3. A 24. fejezetben röviden tárgyaltam a biztonsági validációt. A statikus elemzésről mint verifikációs technikáról a 22. fejezetben volt szó. Ez a technika különösen hasznos az implementáció sérülékenységeinek felderítéséhez (Chess és McGraw, 2004; Livshits és Lam, 2005; Evans és Larochelle, 2002; Larus és mások, 2004). A biztonsági tesztelés témakörében Whittaker és Thompson könyve lehet segítségünkre (Whittaker és Thompson, 2004).

Ebben az alfejezetben a biztonságos rendszertervezéshez kötődő néhány általános, alkalmazástól független kérdésre helyezem a hangsúlyt:

1. Architektúrális tervezés – milyen hatással vannak az architektúrális tervezésről hozott döntések a rendszer biztonságosságára?
2. Jó gyakorlat – mi az elfogadott jó gyakorlat a biztonságos rendszerek tervezésénél?
3. Tervezés a telepítéshez – milyen támogatást tervezzünk a rendszerhez annak érdekében, hogy elkerüljük a rendszer telepítésekor felmerülő sérülékenységeket?

Természetesen nem csak ezek a tervezési kérdések fontosak a biztonság szempontjából. Minden alkalmazás eltérő, és ennek megfelelően a biztonságos tervezésnek figyelembe kell vennie az alkalmazás célját, kritikusságát és működési környezetét. Hely hiányában azonban nem tudok kitérni ezekre az alkalmazás-specifikus tervezési kérdésekre.

30.3.1. Architektúrális tervezés

Amint azt már a 11. fejezetben elmondtam, a rendszer architektúrájának megválasztása jelentős kihatással lehet a rendszer eredendő tulajdonságaira. Nem megfelelő architektúra esetén előfordulhat, hogy gyakorlatilag lehetetlen a rendszerben tárolt információk bizalmasságát és integritását garantálni, vagy a rendszer rendelkezésre állását a kívánt szinten biztosítani.

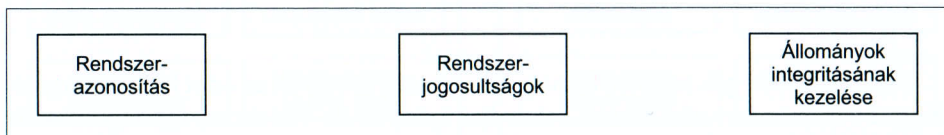
A biztonságosságot figyelembe vevő rendszerarchitektúra tervezésénél két alapvető kérdést kell megfontolnunk:

1. *Védelem.* Hogyan szervezzük meg a rendszert, hogy a kritikus vagyonok védve legyenek a külső támadásokkal szemben?
2. *Elosztás.* Hogyan osszuk el a rendszer vagyonát, hogy a sikeres támadás következtében keletkezett kár minimális legyen?

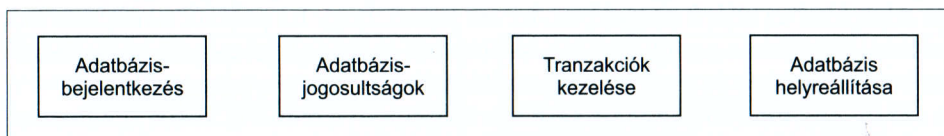
A két szempont részben ellentmondó. Ha az összes vagyont egy helyre helyezzük, akkor védelmi rétegekkel vehetjük körbe. Ha viszont a védelem sikertelen, az összes vagyon odavész. Ellenben ha szétosztjuk a vagyonokat, a védelmük költségesebbé válik, és annak is megnő az esélye, hogy a védelmet kijátsszák. Ekkor azonban nem ér minket teljes veszteség.

A rendszer védelmének biztosításához általában rétegelt architektúrát alkalmazunk, melynek legalsó szintjén a védelmet igénylő kritikus vagyont találhatók, fölötte pedig számos védelmi réteg helyezkedik el. A 30.8. ábrán egy betegnyilvántartó rendszer rétegelt védelmi architektúráját láthatjuk, ahol a kritikus vagyont a betegek rekordjai képezik.

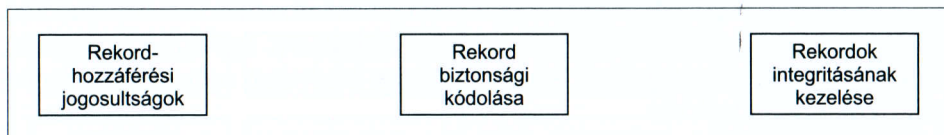
Védelem a platform szintjén



Védelem az alkalmazás szintjén



Védelem a rekordok szintjén



30.8. ábra. Rétegelt védelmi architektúra

Ahhoz, hogy a támadó hozzá tudjon férni a betegek rekordjaihoz és módosítsa azokat, három rendszerrétegen kell áthatolnia:

1. *Platformsztintű védelem.* A legfelső szinten elhelyezkedő ellenőrzések ahhoz a platformhoz férnek hozzá, amelyiken a betegek rekordjait kezelő rendszer fut. Idetartozik a felhasználó bejelentkezése egy konkrét számítógépre. A platform általában tartalmaz a rendszerben tárolt állományok integritásának karbantartását támogató eszközt.
2. *Alkalmazássztintű védelem.* A következő védelmi szint magába az alkalmazásba épül be. Idetartozik a felhasználó hozzáférése az alkalmazáshoz, beleértve a felhasználó azonosítását és az egyes tevékenységekhez (például adatok megtekintése vagy módosítása) fűződő jogosultságainak meghatározását. Az alkalmazás szintén tartalmazhat integritáskezelő eszköztámogatást.

3. *Rekordszintű védelem.* Ez a réteg akkor kerül előtérbe, ha egy bizonyos rekordhoz szeretnénk hozzáférni. A rendszer ellenőrzi, hogy a felhasználó rendelkezik-e a rekordon végrehajtani kívánt művelethez szükséges jogosultsággal. A védelemnek ezen a szintjén használhatunk biztonsági kódolást is, amely megakadályozza, hogy a rekordokat egyszerű állományböngészővel meg lehessen tekinteni. Az ellenőrző összegeket használó integritás-ellenőrzők például felfedezik, ha változás történt a normál rekordfrissítési folyamaton kívül.

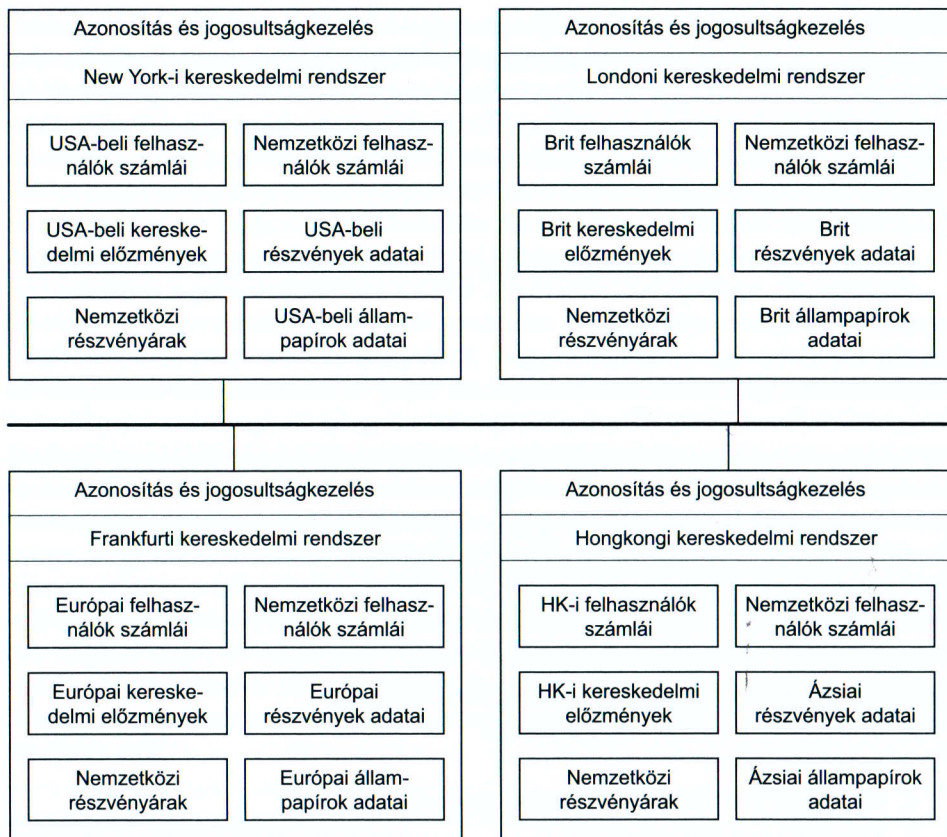
A védelmi rétegek száma bármely alkalmazás esetében az adat kritikusságától függ. Nem minden alkalmazás rekordjai igényelnek védelmet, így legtöbbször kevésbé kifinomult hozzáférés-ellenőrzőket használnak. A nagyobb fokú biztonság elérése érdekében egyazon felhasználó nem igazolhatja magát ugyanazokkal az adatokkal az eltérő rétegekben. Ez azt jelenti, hogy ha például a rendszerünk jelszóalapú, akkor az alkalmazás indításánál megadott jelszónak különböznie kell a rendszerhez és a rekordok hozzáféréséhez szükséges jelszavaktól.

Ha a rendszerben az adatok védelme kritikus követelmény, akkor kliens-szerver architektúrát célszerű alkalmazni. Ilyenkor a védelmet a szerveren kell kiépíteni. Ha azonban a védelmet kijátsszák, akkor a támadás nagy veszteséget okozhat, és a helyreállítás sokba kerülhet (például újra ki kell osztani a felhasználóneveket és jelszavakat). A rendszer elérhetősége sérülékeny, ugyanis egy támadás túlterhelheti a szerveret, és nem képes fogadni a kéréseket, azaz elérhetetlenné válik a rendszeradatbázis.

Ha a szolgáltatások megtagadását célzó támadásokat súlyos kockázatnak tartjuk, használhatunk elosztott objektumarchitektúrát. Ebben az esetben, mint a 30.9. ábrán látható, a rendszer értékeit elosztjuk több különböző platform között, mindegyiknél különböző védelmi mechanizmust alkalmazva. Ha támadás éri az egyik csomópontot, lehet, hogy egyes értékek elérhetetlenné válnak, viszont a rendszer továbbra is képes néhány szolgáltatást biztosítani. Az adatokról másolatokat tárolhatunk a csomópontokon, így a támadás utáni helyreállítás egyszerűsödik.

A 30.9. ábrán egy olyan bank rendszerének az architektúrája látható, amely részvényekkel és állampapírokkal kereskedik a New York-i, londoni, frankfurti és hongkongi piacokon. A rendszer elosztott abban az értelemben, hogy az egyes piacokra vonatkozó adatokat külön kezelik. A rendszer kritikus tevékenységét, azaz a részvényekkel való kereskedést biztosító vagyonok (számlák és árak) minden egyes csomóponton megtalálhatók. Így ha valamelyik csomópontot támadás éri, a kereskedés nem áll le.

Biztonságos rendszerek tervezésénél problémát jelent, ha a biztonságot leginkább támogató architektúra nem felel meg a többi alkalmazáskövetelménynek. Tegyük fel például, hogy egy alkalmazásnál abszolút követelmény az adatok bizalmas kezelése, valamint az adatokhoz való nagyon gyors hozzáférés. Ugyanazon architektúrával nehéz mind a két igénynek megfelelni. A magas szintű biztonság több védelmi réteget követel, ami egyértelműen rontja a teljesítményt, ezáltal csökken az adatok elérésének sebessége. Ha alternatív megoldást választunk, a védelem implementálása és a bizalmasság garantálása bonyolultabb és



30.9. ábra. Részvénykereskedelmi rendszer elosztott értékei

költségesebb lehet. Ilyen helyzetben mindenképpen beszéljük meg a problémát az ügyféllel, és közösen válasszunk megoldást.

30.3.2. Tervezési irányelvek

Nincsenek kötött, jól rögzített szabályok arra vonatkozóan, hogyan lehet elérni egy adott rendszer biztonságát. Az egyes rendszerek különböző technikai eljárásokat igényelnek a rendszer tulajdonosának megfelelő biztonsági szint kialakításához. A különböző felhasználói csoportok hozzáállása és elvárásai nagymértékben meghatározzák, mi elfogadható, és mi nem. Egy bankban például nagyobb biztonsági szintet várunk el, mint mondjuk egy egyetemnél, így a biztonsági eljárások is alaposabbak lesznek.

Ennek ellenére mégis léteznek olyan általános irányelvek, amelyeket széles körben alkalmazhatunk, és amelyek tartalmazzák a biztonságos rendszerek készítéséhez szükséges jó tervezői gyakorlatot. Véleményem szerint a biztonsággal kapcsolatos tervezési irányelveknek két fő felhasználási területe van:

1. A szoftvertervező csapatban a biztonsági tudatosság kialakításának egy módja. A szoftvertervezők gyakran csak rövid távú célokban gondolkoznak, azaz arra törekednek, hogy a szoftver mihamarabb működőképes legyen, és le lehessen szállítani a megrendelőnek. Eközben hajlamosak megfedkezni a biztonságosságról. Az említett irányelvek ismerete biztosíthatja, hogy a biztonsági kérdéseket is figyelembe veszik a tervezési döntések meghozatalakor.
2. A rendszer validációs szakaszában ellenőrző lista alapjául szolgálhat. Az itt tárgyalt magas szintű irányelvekből konkrét kérdéseket lehet származtatni, amelyek segítségével felmérhető, miként valósították meg a rendszer biztonságosságát.

Az általam felsorolt 10 irányelv (30.10. ábra) több forrásból (Schneier, 2000; Viega és McGraw, 2002; Wheeler, 2003) lett összegyűjtve. Azokat az irányelveket tartottam szem előtt, amelyek főként a szoftverspecifikációnál és a tervezésnél játszanak fontos szerepet. Az olyan általánosabb elvek, mint a „Biztosítsd a rendszer leggyengébb láncszemét”, vagy „Az egyszerű a nagyszerű” stb. szintén fontosak, de kevesebb közülük van a tervezési döntésekhez.

Irányelvek

1. Explicit biztonságpolitikára építsük a biztonsági döntéseket
2. Kerüljük a kritikus meghibásodási pontokat
3. Biztonságos bukás
4. A biztonság és a használhatóság legyen egyensúlyban
5. Gondoljunk a szociális tervezésre
6. Használjunk redundanciát és diverzitást a kockázat csökkentésére
7. Validáljuk a bemenetet
8. Tagoljuk a vagonokat
9. Gondoljunk tervezéskor a rendszer bevezetésére
10. Gondoljunk tervezéskor a rendszer helyreállítására

30.10. ábra. Tervezési irányelvek biztonságos rendszerek fejlesztéséhez

1. irányelv: Explicit biztonságpolitikára építsük a biztonsági döntéseket

A biztonságpolitika a szervezet felső szintjének állásfoglalása, amely meghatározza a szervezet alapvető biztonsági feltételeit. A biztonságpolitika a feltételeket szabja meg („mit”), nem pedig azok megvalósítását („hogyan”), így nem határozhatja meg a biztonság kialakítására szolgáló mechanizmusokat. Elvben a rendszerkövetelményeknek tükrözniük kell a biztonságpolitika összes szempontját, a gyakorlatban azonban, különösen a gyors alkalmazásfejlesztési módszereknél, ez nem így van. Ajánlatos tehát, hogy a tervezők megismerkedjenek a biztonságpolitikával, mert az keretként szolgálhat a tervezési döntések meghozatalához és kiértékeléséhez.

A példa kedvéért tegyük fel, hogy a fejezetben korábban bemutatott klinika információs rendszerének hozzáférés-ellenőrző rendszerét kell megterveznünk. A klinika biztonságpolitikája megszabhatja, hogy csak meghatalmazott dolgozók módosíthatják a betegek elektronikusan rögzített adatait. Ennek megfelelően a rendszerünknek tartalmaznia kell olyan mechanizmust, amely ellenőrzi, hogy a módosítást kezdeményező személy rendelkezik-e a szükséges meghatalmazással.

2. irányelv: Kerüljük a kritikus meghibásodási pontokat

Kritikus rendszereknél a jó tervezői gyakorlat megpróbálja elkerülni a kritikus meghibásodási pontokat. Ez azt jelenti, hogy a rendszer egy adott pontjának sérülése nem vezet a rendszer teljes összeomlásához. Biztonsági terminológiát használva, ne csak egyetlen mechanizmusra támaszkodjunk, hanem alkalmazunk több különböző technikát. Ezt gyakran hívják „mélységi védelemnek”.

Ha például jelszó segítségével azonosítjuk a rendszer felhasználóit, kiegészítésképp beépíthetünk egy kérdés-válasz alapú mechanizmust, melynek az a lényege, hogy a felhasználók előre rögzítenek néhány kérdést és a hozzájuk tartozó választ. Miután a felhasználók beírták a helyes jelszót, a belépéshez válaszolniuk kell a feltett kérdésekre. A rendszerben tárolt adatok integritásának ellenőrzéséhez naplózhatjuk az adatokon végzett összes módosítást, hogy meghibásodás esetén vissza lehessen állítani az adatok eredeti állapotát a naplónak megfelelően. Esetleg másolatot készíthetünk a módosítandó adatokról a változtatás végrehajtása előtt.

3. irányelv: Biztonságos bukás

A rendszer valamilyen szintű meghibásodása előbb-utóbb elkerülhetetlen, így akárcsak a biztonságosságkritikus rendszereknél, itt is biztosítanunk kell, hogy a rendszer „biztonságosan bukjon”. A rendszer meghibásodásakor nem használhatunk olyan mechanizmusokat, amelyek kevésbé biztonságosak, mint maga a rendszer. Az sem megengedett, hogy a támadó hozzáférhessen azokhoz az adatokhoz, amelyekhez normál körülmények között nem tudna.

A betegeket nyilvántartó információs rendszer esetében például azt javasoltam, hogy az adatok kerüljenek letöltésre egy rendszerkliensre minden egyes klinikai munkamenet kezdetén. Ez felgyorsítja az adatok elérését, és akkor is biztosítja a hozzáférést, amikor a szerver nem elérhető. Normál esetben a szerver a munkamenet végén törli ezeket az adatokat. Ha azonban a szerver meghibásodik, lehetővé kell tenni, hogy az adatok a kliensen maradjanak. Ilyenkor biztonsági megoldás lehet a kliensen tárolt adatok bekódolása. Ennek híján jogosultak személyek is el tudnák olvasni az adatokat.

4. irányelv: A biztonság és a használhatóság legyen egyensúlyban

A biztonság és a használhatóság gyakran ellentmondó szempontok. Ahhoz, hogy egy rendszer biztonságos legyen, be kell vezetnünk néhány ellenőrzőt, amelyek

biztosítják, hogy a felhasználók jogosultak a rendszer használatára, és a biztonságpolitikának megfelelő tevékenységet folytatnak. Mindez teher a felhasználók számára: bejelentkezési neveket és jelszavakat kell megjegyezniük, csak bizonyos számítógépekről férhetnek a rendszerhez stb. A felhasználóknak tehát jóval több időre van szükségük ahhoz, hogy elkezdhessék a munkájukat a rendszerben, és hatékonyan tudják azt használni. Minél több biztonsági intézkedést adunk a rendszerhez, annál jobban csökken a használhatóság.

Egy bizonyos pontnál abba kell hagyni az újabb és újabb biztonsági mechanizmusok bevezetését, mert olyan mértékben csökkenhet a rendszer használhatósága, hogy az már a biztonságot is veszélyezteti. Például ha azt várjuk el a felhasználóinktól, hogy több jelszót tartsanak fejben, amelyek megjegyezhetetlen karaktorsorozatokból állnak, és ráadásul rendszeres időközönként változtassanak rajtuk, egyszerűen le fogják írni a jelszavakat, hogy ne felejtsek el. Ekkor lehetővé válik, hogy a támadó (különösen ha belső emberről van szó) megtalálja a jelszavakat és hozzáférjen a rendszerhez.

5. irányelv: Gondoljunk a szociális tervezésre

Szociális tervezésről akkor beszélünk, amikor rászedik a jogosult felhasználót, hogy bizalmas információt szolgáltatson ki. Ezek a megközelítések kihasználják a felhasználók jóhiszeműségét és a felsőbb hatalomba vetett bizalmukat. Előfordulhat, hogy valaki fontos ügyvezetőnek adja ki magát, és arra kéri az újonc dolgozót, hogy segítsen neki, mert valamiért nem engedi be a rendszer. Az illető biztonsággal kapcsolatos kérdéseket tehet fel, és valószínű, hogy az újonc nem szívesen utasítja vissza a segítségnyújtást. A biztonsági részletek kiszolgáltatásával azonban lehetővé válik, hogy jogosulatlan személyek hozzáférjenek a rendszerhez.

Tervezési szempontból nagyon nehéz kiküszöbölni a szociális tervezést. Ha mindenképp fontos a rendszer magas szintű biztonsága, nem hagyatkozhatunk pusztán olyan felhasználóhitelesítő mechanizmusokra, amelyek bejelentkezési nevekre és jelszavakra épülnek. Szigorúbb hitelesítést kell alkalmazni, például digitális tanúsítványokat. A felhasználók kilétét és tartózkodási helyét naplózó mechanizmusok, valamint a naplóelemző programok segítségünkre lehetnek a biztonság megsértésének felderítésében.

6. irányelv: Használjunk redundanciát és diverzitást a kockázat csökkentésére

A redundancia azt jelenti, hogy a szoftverből vagy az adatokból több verzió is létezik a rendszerben. A diverzitás (szoftvereknél) annyit tesz, hogy a különböző verziók nem épülhetnek ugyanarra a platformra, vagy nem használhatják ugyanazt a technológiát. Így egy bizonyos platform vagy technológia sebezhetősége nem érinti az összes verziót, és nem vezet teljes meghibásodáshoz.

A redundanciára már láthattunk példákat – a betegek rekordjainak tárolása a szerveren és a kliensen egyszerre a klinika információs rendszerénél, valamint az elosztott részvénykereskedelmi rendszer a 30.9. ábrán. A betegnyilvántartó

rendszernél használhatunk eltérő operációs rendszert a szerveren és a kliensen (például Linuxot a szerveren, Windowst a kliensen). Ezáltal biztosíthatjuk, hogy az operációs rendszer sérülékenységére építő támadás nem érintheti egyszerre a klienst és a szervert.

7. irányelv: Validáljuk a bemenetet

Gyakori támadástípusnak számít, amikor a támadók váratlan bemenetet adnak a rendszernek, aminek következtében az kiszámíthatatlan viselkedést produkálhat. Ez vezethet rendszerösszeomláshoz, amikor a szolgáltatások elérhetetlenné válnak, de az is elképzelhető, hogy a bemenet ártalmas kódot tartalmaz, amit a rendszer lefordít. A hosszú bemeneti sztringek puffertúlsordulást okozhatnak. Ezt a sérülékenységet először az internetféreg használta ki (Spafford, 1989), de azóta is kedvelt támadási módszer (Berghel, 2001). Az ún. „SQL-méreg”, melynek során a rosszindulatú felhasználó a szerver által lefordításra kerülő SQL-utasítást ad meg bemenetként, szintén elég gyakori támadástípus.

Ezen problémák többségét elkerülhetjük, ha validáljuk a rendszer összes lehetséges bemenetét. Nagyon fontos, hogy alapvetően egyetlen bemenetet se hagyjunk valamilyenfajta ellenőrzés nélkül. A követelmények részeként határozzuk meg a bemenetekre vonatkozó ellenőrzéseket, a bemenet jellegének figyelembevételével. Például ha meg kell adni a vezetéknévét, ellenőrizhetjük, hogy tartalmaz-e szóközöket, az egyetlen megengedett írásjel pedig a kötőjel. Ellenőrizhetjük a bemeneti sztringek hosszát is, és elvethetjük az egyértelműen túl hosszú bemeneteket. Például senkinek sincs 70 karakternél hosszabb vezetéknéve vagy 100 karakternél hosszabb címe. Amint azt a 16. fejezetben kifejtettem, elkerülhetünk néhány bemenetvalidációs problémát, ha a bemenetet menü segítségével lehet megadni.

8. irányelv: Tagoljuk a vagyonekat

A tagolás azt jelenti, hogy a rendszerhez való hozzáférés nem „mindent vagy semmit” alapon történik, hanem úgy szervezzük meg a rendszerben az információt, hogy a felhasználók csak azokhoz az információkhoz tudnak hozzáférni, amire szükségük van, nem pedig a rendszerben tárolt összes adathoz. Ennek következtében korlátok közé lehet szorítani a támadások hatását. Egyes információk sérülhetnek vagy elveszhetnek, de nem valószínű, hogy ez a rendszerben tárolt összes információt érinteni fogja.

Például tervezhetjük a klinika információs rendszerét oly módon, hogy a dolgozók csak a konzultációkra feliratkozott betegek adatait láthassák, ne pedig az összes beteg rekordját. Ezenfelül beépíthetünk olyan mechanizmusokat is, amelyek váratlan helyzetekben biztosítják a hozzáférést (például egy súlyos betegnek sürgős kezelésre van szüksége, előzetes egyeztetés nélkül). Utóbbi esetben olyan alternatív biztonsági mechanizmusra van szükség, amely figyelmen kívül hagyja a rendszer tagolását.

9. irányelv: Gondoljunk tervezéskor a rendszer bevezetésére

Sok biztonsági problémának az a forrása, hogy a rendszert nem megfelelően konfigurálják a telepítéskor. Éppen ezért mindig úgy tervezzük meg a rendszerünket, hogy az tartalmazzon a bevezetést egyszerűsítő és a lehetséges konfigurációs hibákat felismerő eszközöket. Fontossága miatt ezt az irányelvet a 30.3.3. alfejezetben külön is részletezem.

10. irányelv: Gondoljunk tervezéskor a rendszer helyreállítására

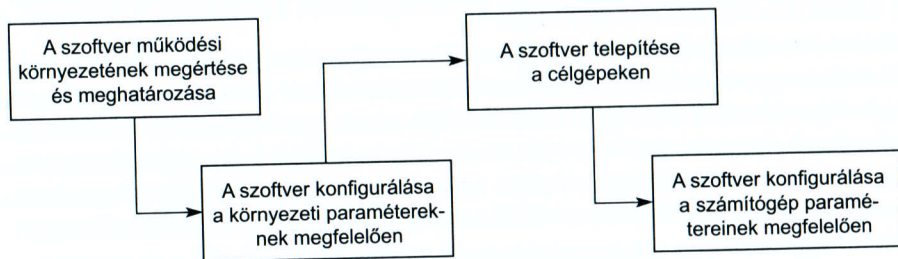
Függetlenül attól, hogy mennyi energiát fektetünk a rendszerbiztonság karbantartására, tervezéskor tartsuk szem előtt, hogy biztonsági hiba bármikor bekövetkezhet. Gondoljuk tehát végig, miként küzdenénk meg egy esetleges meghibásodással, és hogyan állítanánk vissza a rendszer biztonságos működési állapotát.

Tegyük fel például, hogy egy illetéktelen külső személy hozzáfért a betegek rekordjaihoz, de nem tudjuk, honnan szerezhette meg a szükséges bejelentkezési nevet és jelszót. Ilyen helyzetben változtassuk meg az összes jogosult felhasználó jelszavát, és gondoskodjunk arról, hogy ezt az illetéktelen személy ne tudja meg. Úgy kell tehát megterveznünk a rendszerünket, hogy addig ne engedélyezze a belépést, amíg a felhasználók nem módosították a jelszavukat, annak ellenére, hogy a jelszó nem a legmegbízhatóbb módszer. A jelszó megváltoztatásakor használhatjuk a már korábban említett kérdés-válasz mechanizmust is, amikor a felhasználóknak előre rögzített kérdésekre kell válaszolniuk.

30.3.3. Tervezés a rendszer bevezetésének figyelembevételével

Egy rendszer bevezetése alapvetően a következő szakaszokból áll: a szoftver konfigurálása a működési környezetnek megfelelően, a szoftver telepítése a környezetben található számítógépekre, majd a telepített szoftver konfigurálása az egyes gépeken (30.11. ábra). A konfigurálás lehet nagyon egyszerű, például a felhasználó választhat néhány beépített paraméter közül, vagy egészen bonyolult, mint a szoftver futását befolyásoló üzleti modellek és szabályok meghatározása.

A szoftverfolyamat ezen szakaszában gyakran becsúsznak a rendszer sebezhetőségét előidéző hibák. A legtöbb szoftvernél például konfigurálni kell a meg-



30.11. ábra. A szoftver bevezetése

engedett felhasználók listáját. A lista kezdetben csak egy általános adminisztrátori azonosítót (például „admin”) és alapértelmezett jelszót (például „password”) tartalmaz, ami könnyíti az adminisztrátorok munkáját. Első lépésben meg kell változtatni ezt az azonosítót és jelszót, erről azonban könnyű megfeledkezni. Ha egy támadó ismeri az alapértelmezett bejelentkezési paramétereket, privilegizált rendszerhozzáféréshez juthat.

A konfigurációt és a telepítést gyakran adminisztrátori tevékenységnek tartják, ami kívül esik a szoftverfejlesztési folyamat hatáskörén. Való igaz, hogy jó karbantartói gyakorlattal számos konfigurációs és telepítéssel kapcsolatos probléma elkerülhető. A szoftvertervezőknek viszont az a feladata, hogy a szoftver működésének figyelembevételével tervezzék meg a rendszert. Mindig gondoskodjunk olyan beépített eszközökről, amelyek csökkentik a rendszer-adminisztrátorok (vagy felhasználók) hibázási lehetőségeit a szoftver konfigurálásakor.

A szoftver bevezetésének támogatásához az alábbi négy módszert javaslom:

1. *Támogassuk a konfigurációk megtekintését és elemzését.* A rendszer mindig tartalmazzon olyan eszközöket, amelyekkel az adminisztrátorok vagy a jogosult felhasználók megnézhetik a rendszer aktuális konfigurációját. Meglepő módon ez a szolgáltatás a legtöbb szoftverrendszerből hiányzik, így a felhasználók csak nehézségek árán tudják megtalálni a konfigurációs beállításokat. Például ha a Microsoft Word 2005-ben elérhető verzióját használjuk, valószínűleg észrevettük, hogy lehetetlen az összes beállítást egy oldalon megjeleníteni vagy kinyomtatni. Ha az adminisztrátor teljes képet kapna a konfigurációról, könnyebben ki tudná szűrni a hibákat és a hiányosságokat. Ideális esetben a megjelenítésnek ki kell emelnie a potenciálisan nem biztonságos konfigurációs beállításokat (például nem adtunk meg jelszót).
2. *Minimalizáljuk az alapértelmezett privilégiumokat.* A szoftvert úgy kell megterveznünk, hogy a rendszer alapértelmezés szerinti konfigurációja lehetőleg minimális számban tartalmazzon privilégiumokat. Ilyenkor ugyanis a támadó csak korlátozott mértékben tehet kárt a rendszerben. Például az alapértelmezett adminisztrátori belépéssel csak olyan programhoz lehessen hozzáférni, amivel az adminisztrátor a bejelentkezéshez szükséges adatokat tudja átállítani, a rendszer további eszközeit viszont nem érheti el. Az új bejelentkezési név és jelszó beállítása után az alapértelmezetteket automatikusan törölni kell.
3. *Csoportosítsuk a konfigurációs beállításokat.* A rendszer konfigurációjának tervezésekor biztosítsuk, hogy az ugyanazon feladathoz tartozó beállítások egy helyen találhatók. Visszatérve a Microsoft Word példájához, az általam használt verzióban a dokumentumhoz kapcsolódó egyes biztonsági beállítások (például jelszó) a Beállítások almenüpont Biztonság menüjében találhatók, míg a többi hasonló beállítás az Eszközök / Dokumentumvédelem menüpontokon keresztül érhető el. Ha a konfigurációra vonatkozó információkat nem csoportosítjuk egy helyen, könnyű valamelyikről megfeledkezni, sőt az is előfordulhat, hogy bizonyos beállításokról nem is szerzünk tudomást.

4. *Könnyen ki lehessen javítani a biztonsági hibákat.* A rendszer tartalmazzon egyértelmű mechanizmusokat, amelyek lehetővé teszik a rendszer frissítését, amennyiben biztonsági hibát fedeztek fel. Ilyen mechanizmus lehet például a biztonsági frissítések folyamatos figyelése és azok mihamarabbi letöltése. Vigyünk figyelembe, hogy esetenként akár több száz számítógép frissítéséről is gondoskodni kell.

30.4. A RENDSZER TÚLÉLŐKÉPESSÉGE

Az eddigiekben a biztonsági tervezést a fejlesztés alatt álló alkalmazások szempontjából közelítettem meg. A rendszer megrendelője és fejlesztője a rendszer összes megtámadható része fölött rendelkezik. A valóságban, amint az a 30.1. ábrán is látszik, a modern elosztott rendszerek legtöbbször COTS és újrafelhasználható komponensekből állnak, amiket külön fejlesztettek.

Ez azt jelenti, hogy függetlenül attól, mennyi figyelmet fordítottunk biztonsági kérdésekre, nem biztosítható, hogy a rendszer ellenálló lesz a külső támadásokkal szemben. Emiatt jobb, ha tudomásul vesszük, hogy komplex hálózati rendszereknél előfordulhatnak behatolások, és a rendszer integritását sem garantálhatjuk. Ajánlott tehát a rendszert rugalmasra tervezni, hogy egy esetleges támadás esetén is tudjon alapvető szolgáltatásokat nyújtani.

A túlélőképesség (Westmark, 2004) a rendszer eredendő tulajdonsága, nem pedig az egyes komponensek jellemzője, melyek maguk nem feltétlenül túlélőképességek. Túlélőképességen a rendszer azon képességét értjük, amely lehetővé teszi, hogy támadás esetén vagy egy támadásból származó sérülés után a rendszer továbbra is biztosítani tudja a lényeges üzlet- vagy küldetéskritikus szolgáltatásokat.

Napjainkban a gazdasági és társadalmi életünk számítógép-alapú kritikus infrastruktúrákra épül. Ez a tény nagymértékben inspirálta a rendszer túlélőképességének kutatását. Kritikusnak tekinthetők a közszolgáltatásokat (energia, víz, gáz stb.) biztosító, továbbá az információ továbbítását és kezelését végző infrastruktúrák (telefon, internet, posta stb.). A túlélőképesség azonban nem csak a kritikus infrastruktúráknál jelenik meg. Minden szervezetnek, amely kritikus hálózati rendszert használ, végig kell gondolnia, hogyan érintené üzleti tevékenységét, ha a rendszer nem élne túl egy támadást vagy egy katasztrofális meghibásodást. Az üzleti-kritikus rendszereknél ezért a biztonsági tervezésnek tartalmaznia kell a túlélőképesség elemzését és tervezését.

A túlélőképesség lényege a kritikus szolgáltatások rendelkezésre állásának biztosítása. Ehhez az alábbiakat kell ismernünk:

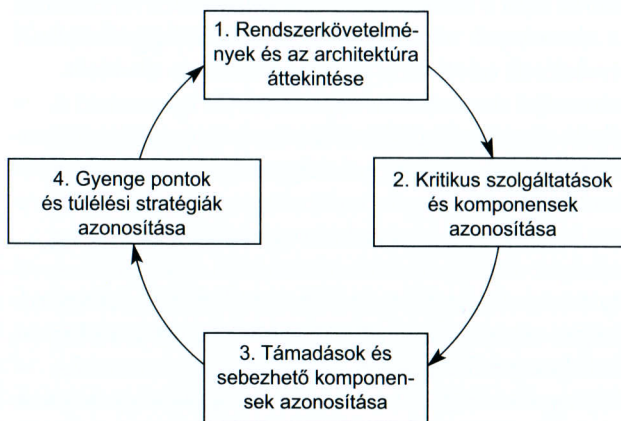
- melyek az üzleti szempontból legkritikusabb rendszerszolgáltatások;
- hogyan sérülhetnek ezek a szolgáltatások;
- mi a fenntartandó minimális szolgáltatási színvonal;
- hogyan lehet ezeket a szolgáltatásokat megvédeni;
- hogyan lehet gyors helyreállítást végezni, ha a szolgáltatások elérhetetlenné válnak.

Egy mentőszolgálat esetében, amely sürgősségi hívások alapján mentőket küld a helyszínre, kritikus szolgáltatás lesz a bejövő hívások fogadása és az autók kiküldése. Az olyan egyéb tevékenységek, mint a hívások naplózása vagy a mentőautók helyének meghatározása kevésbé kritikusak, mert nem igényelnek valós idejű feldolgozást, vagy mert helyettesíthetők más módszerekkel. Könnyen kideríthetjük például egy mentőautó tartózkodási helyét, ha felhívjuk a mentősöket, és egyszerűen megkérdezzük, hol vannak.

Ellison és kollégái (Ellison és mások, 1999; 1999a; 2002) kidolgoztak egy elemzési módszert, amit túlélőképes rendszerelemzésnek (Survivable Systems Analysis) neveztek el. A módszer segít a rendszer sebezhető pontjainak felderítésében, valamint támogatja a túlélőképességet szem előtt tartó rendszer-architektúrákat és eszközöket. Véleményük szerint a túlélőképesség három, egymást kölcsönösen kiegészítő stratégiával érhető el:

1. *Ellenállás.* A problémák elkerülése olyan rendszerképeségek beépítésével, amelyek elriasztják a támadásokat. Például a rendszer használhat digitális tanúsítványokat, ami megnehezíti a jogosulatlanul belépni próbálók dolgát.
2. *Felismerés.* A problémák felderítése olyan rendszerképeségek beépítésével, amelyek felfedezik a támadásokat és a meghibásodásokat, valamint megállapítják a keletkezett kár nagyságát. Például a kritikus adatok védelmére használhatunk ellenőrző összegeket.
3. *Helyreállítás.* A problémák elviselése olyan rendszerképeségek beépítésével, amelyek egy esetleges támadás esetén biztosítják az alapvető szolgáltatásokat, és segítik a rendszer teljes körű funkcionalitásának helyreállítását. Az egyes funkciók eltérő technológiákkal történő megvalósítását tekinthetjük effajta hibatűrő mechanizmusnak, mert segítségével helyettesíthető a megtámadott szolgáltatás.

A túlélőképes rendszerek tervezése egy négyszakaszos folyamat (30.12. ábra), amely elemzi az aktuális vagy a javasolt rendszerkövetelményeket és a rendszer-



30.12. ábra. A túlélési elemzés szakaszai

architektúrát, azonosítja a kritikus szolgáltatásokat, támadási forgatókönyveket és a rendszer gyenge pontjait, valamint változtatásokat javasol a rendszer túlélőképességének növelésére. Az egyes szakaszokhoz kötődő főbb tevékenységek:

1. *A rendszer megértése.* Egy létező vagy javasolt rendszernél tekintsük át a rendszer céljait (küldetési célnak is szokták hívni), a rendszerkövetelményeket és a rendszer-architektúrát.
2. *Kritikus szolgáltatások azonosítása.* Azonosítjuk a legfontosabb szolgáltatásokat és a működésükhöz elengedhetetlen komponenseket.
3. *Támadás szimulálása.* Elkészítjük a támadási forgatókönyvet vagy használati esetet, valamint azonosítjuk az egyes támadások által érintett komponenseket.
4. *Túlélőképességi elemzés.* Azonosítjuk a létfontosságú, de egyben megtámadható komponenseket, és meghatározzuk az ellenállásra, felismerésre és helyreállításra építő stratégiákat.

Ellison és kollégái egy kiváló esettanulmányon keresztül mutatják be módszerüket (Ellison és mások, 1999). A tanulmányukban egy mentális betegek kezelését támogató rendszert elemeznek. Ehelyett én a 30.9. ábrán látható kereskedelmi rendszert használok illusztrációként.

A 30.9. ábrán láthatjuk, hogy néhány túlélési módszerről már gondoskodtak. A felhasználói számlák adatait és a részvényárakat több szerveren tárolják, így ha a helyi szerver elérhetetlen lenne, a kereskedés nem áll le. Tegyük fel, hogy a rendszer által nyújtott kulcsfontosságú szolgáltatás a részvények rendelése. Annak érdekében, hogy a felhasználók bízzanak a rendszerben, garantálnunk kell az integritást. A megrendeléseknek pontosnak kell lenniük, és tükrözniük kell a felhasználó tényleges vétellel és eladással kapcsolatos tevékenységét.

A megrendelési szolgáltatás működtetéséhez három komponensre van szükség:

1. *Felhasználó hitelesítése.* Lehetővé teszi a felhasználó bejelentkezését a rendszerbe.
2. *Árjegyzés.* Lehetővé teszi a részvények vételi és eladási árának jegyzését.
3. *Rendelés.* Lehetővé teszi rendelések adott áron történő vételét és eladását.

Ezek a komponensek nyilván olyan nélkülözhetetlen adatvagyonokat használnak, mint például a felhasználók adatait vagy a megrendelési tranzakciókat nyilvántartó adatbázisok.

Többféle támadás érheti ezt a rendszert, nézzünk most közülük kettőt:

1. Egy rosszindulatú személy ártani akar egy jogos felhasználónak. A felhasználó bejelentkezéséhez szükséges adatokat megszerezve belép a rendszerbe, ad, vesz, rendel, és sok fejfájást okoz a felhasználónak.
2. Jogosulatlan illető engedélyt szerez SQL-utasítások közvetlen végrehajtásához, és megmásítja a tranzakciós adatbázist. Az eladások és a vételek összeegyeztethetetlené válnak.

Támadás	Ellenállás	Felismerés	Helyreállítás
Jogosulatlan személy rosszindulatú rendelőket tesz	A rendelésekhez a bejelentkezéstől eltérő jelszóra van szükség.	A megrendelés másolatának elküldése e-mailben a felhasználó részére, telefonszámmal együtt (rosszindulatú rendelések kiszűrése). A felhasználó rendeléseinek nyilvántartása, szokatlan tevékenység figyelése.	Mechanizmusok biztosítása a vásárlások automatikus visszavonásához és a felhasználói számlák visszaállításához. A sértett felhasználók kárainak megtérítése. További veszteségek elleni biztosítás.
A tranzakciós adatbázis megmásítása	A privilegizált felhasználók szigorúbb azonosítása, például digitális tanúsítvánnyal.	A tranzakciókról egy csak olvasható másolat tárolása egy nemzetközi szerveren. A tranzakciók rendszeres összehasonlítása. Kriptografikus ellenőrző összegek használata a tranzakciós rekordokban.	Az adatbázis helyreállítása biztonsági másolatokból. A kereskedések visszajátzásának lehetősége egy megadott időpillanattól kezdve a tranzakciók visszaállításához.

30.13. ábra. Egy részvénykereskedelmi rendszer túlélési elemzése

A 30.13. ábrán ellenállási, felismerési és helyreállítási stratégiákat láthatunk az iménti problémák megoldására.

A rendszer túlélőképességének biztosítása természetesen pénzbe kerül. A vállalatok nem szívesen költenek a túlélőképességre, ha korábban nem érte őket komolyabb támadás. Azonban éppen úgy, ahogy a jó zárat és a riasztót a lehetséges betörés előtt, és nem után vesszük meg, ugyanúgy érdemesebb a túlélőképességet egy esetleges támadás előtt, nem pedig utána kialakítani. A legtöbb szoftverfejlesztési folyamatnak egyelőre nem része a túlélési elemzés, azonban egyre több rendszer válik üzletkritikussá, ami előbb-utóbb szükségessé teszi ezt a típusú elemzést.

Kulcsfogalmak

- A biztonsági tervezés olyan rendszerek fejlesztésével és karbantartásával foglalkozik, amelyek ellenállnak a számítógép-alapú rendszerek és a bennük tárolt adatok megrongálására irányuló támadásoknak.
- Biztonsági fenyegetés lehet a rendszer vagy a benne tárolt adatok bizalmasságát, integritását vagy elérhetőségét fenyegető veszély.
- A biztonsági kockázatkezelés magában foglalja a lehetséges támadásokból származó károk felbecslését, valamint a károk elkerülését elősegítő biztonsági követelmények meghatározását.
- A biztonságot figyelembe vevő tervezés magában foglalja a biztonságos rendszer-architektúra tervezését, a jó tervezői gyakorlatot és olyan funkcionalitások beépítését, amelyek minimalizálják a rendszer bevezetésekor keletkező sebezhetőségeket.

- A biztonságos rendszer-architektúrák tervezésénél kulcsfontosságú a kritikus vagyontok védelmét szolgáló rendszerstruktúra kiépítése és a vagyontok oly módon történő szétosztása, hogy a támadásból származó veszteség minimális legyen.
- Az általános biztonsági irányelvek felhívják a tervezők figyelmét azokra a biztonsági kérdésekre, amelyekre esetleg nem gondolnának, de ellenőrző lista alapjául is szolgálhatnak.
- A biztonságos rendszerbevezetés érdekében biztosítsunk eszközöket a rendszer-konfigurációk megtekintéséhez és elemzéséhez, csoportosítsuk a konfigurációs beállításokat, minimalizáljuk a felhasználók alapértelmezett privilégiumait, és nyújtsunk segítséget a biztonsági hibák kijavításához.
- A túlélőképesség a rendszer azon képessége, amely lehetővé teszi, hogy támadás esetén vagy egy támadásból származó sérülés után a rendszer továbbra is biztosítani tudja a lényeges üzleti vagy küldetés-kritikus szolgáltatásokat.

További irodalom

Security Engineering: A Guide to Building Dependable Distributed Systems. Ez a könyv kimerítően és átfogóan tárgyalja a biztonságos rendszerekkel kapcsolatos problémákat. A hangsúly inkább a rendszerek, semmint a szoftverek tervezésén van, fontos szerepet kap a hardver és a hálózat leírása. Kitűnő példákat tartalmaz valódi rendszer-meghibásodások alapján. (Anderson, R., 2001, John Wiley & Sons.)

Building Secure Software: How to Avoid Security Problems the Right Way. Gyakorlatias könyv, mely programozási szempontból foglalkozik a biztonsággal. (Viega, J. és McGraw, G., 2002, Addison-Wesley.)

„Survivable network system analysis: a case study”. Ez a kiváló tanulmány vezeti be a rendszer túlélőképességének fogalmát, és egy mentális egészségügyi adatokat kezelő rendszeren keresztül mutatja be a túlélőképesség-módszer alkalmazását. (Ellison, R. J., Linger, R. C., Longstaff, T. és Mead, N. R., *IEEE Software*, 16 (4), July/August 1999.)

Feladatok

- 30.1. Magyarázza meg, mik a főbb különbségek az alkalmazás és az infrastruktúra biztonsága között.
- 30.2. A LIBSYS-rendszer dokumentumok elosztását szolgálja, és már szerepelt az eddigi fejezetekben. Hozzon fel egy-egy példát ennél a rendszernél vagyona, függősége, sérülékenysége, támadásra, fenyegetésre és ellenőrzésre.
- 30.3. Magyarázza meg, miért van szükség az előzetes kockázatkértékelésre és az életciklus során történő kockázatkértékelésre egyaránt egy rendszer fejlesztése során.

- 30.4. A 2. kérdésre adott válaszai alapján értékelje ki a LIBSYS-rendszert érintő kockázatokat. Javasoljon két rendszerkövetelményt a kockázatok csökkentésére.
- 30.5. Magyarázza el egy szoftverfejlesztésen kívüli analógiával, miért szükséges a vagyonok rétegelt védelme.
- 30.6. Magyarázza meg, miért fontos az eltérő technológiák használata olyan elosztott rendszereknél, melyeknél kritikus a rendszer rendelkezésre állása.
- 30.7. Mi a szociális tervezés? Miért nehéz védekezni ellene nagy szervezeteknél?
- 30.8. Elemezze egy ön által használt, készen kapott rendszer (például Microsoft Word) konfigurációs lehetőségeit, és fejtse ki a problémákat.
- 30.9. Magyarázza meg, hogyan használhatók az ellenállási, felismerési és helyreállítási stratégiák a rendszer túlélőképességének biztosításánál.
- 30.10. Soroljon fel további két elképzelhető támadást a 30.4. alfejezetben tárgyalt, és a 30.9. ábrán látható részvénykereskedelmi rendszer esetében, és javasoljon stratégiákat ezen támadások kivédésére.