3 KRITIKUS RENDSZEREK

TÉMAKÖRÖK

Ez a fejezet a kritikus rendszerekkel foglalkozik, amelyek legfontosabb tulajdonsága az üzembiztonság. A fejezet célja, hogy:

- megértsük az üzembiztonság négy aspektusát: rendelkezésre állás, megbízhatóság, biztonságosság és védettség;
- megértsük, hogy az üzembiztonság biztosításához el kell kerülni a rendszer fejlesztése során előforduló hibákat, meg kell keresni és el kell távolítani a rendszer használata közben fellépő hibákat, és minimalizálni kell az operációs hibák által okozott károkat.

TARTALOM

Sok szoftver által vezérelt rendszer hibája kényelmetlenséget okoz, de nem eredményez komoly, hosszú távú károkat. Vannak azonban olyan rendszerek, amelyeknél a hibák jelentős gazdasági, fizikai károkat okozhatnak, vagy veszélyt jelenthetnek az emberre. Ezeket a rendszereket általában *kritikus rendszerek*nek nevezzük. Az üzembiztonság a kritikus rendszerek lényeges jellemzője, és annak minden összetevője (rendelkezésre állás, megbízhatóság, biztonságosság és védettség) fontos lehet. A magas szintű üzembiztonság általában a kritikus rendszerek legfontosabb követelménye. Ezeknek a rendszereknek három fő típusa van:

1. *Biztonságosságkritikus rendszerek.* Olyan rendszerek, amelyeknek a hibája sérülést, életvesztést vagy jelentős környezeti károkat eredményezhet. Ilyen rendszer például egy vegyipari üzem irányítórendszere.
2. *Küldetéskritikus rendszerek.* Olyan rendszerek, amelyeknek a hibája valamilyen célorientált tevékenység sikertelenségét eredményezi. Ilyen például egy űrhajó navigációs rendszere.
3. *Üzleti szempontból kritikus rendszerek.* Olyan rendszerek, amelyeknek a hibája a rendszert használó cégeknél eredményezhet valamilyen hibát. Ilyen például az ügyfélszámla kezelésére használt rendszer egy bankban.

A kritikus rendszerek legfontosabb eredendő tulajdonsága az üzembiztonság. Az *üzembiztonság* fogalmát Laprie vezette be (Laprie, 1995) a rendelkezésre állás, megbízhatóság, biztonságosság és védettség együttes kifejezésére. Mint ahogy arról a 3.2. alfejezetben szó lesz, ezek a tulajdonságok kibogozhatatlanul össze vannak fonódva, ezért is célszerű, hogy egyetlen szóval kifejezhetjük.

Számos oka van, amiért az üzembiztonságot tartjuk a kritikus rendszerek legfontosabb tulajdonságának:

1. *Azokat a rendszereket, amelyek nem megbízhatók, nem biztonságosak és nem védettek, általában nem használják.* Ha egy rendszerben nem bízik a felhasználója, akkor nem hajlandó azt használni. Sőt lehet, hogy elutasítja majd azokat a termékeket is, amelyek ugyanattól a cégtől származnak, amely a megbízhatatlan rendszert gyártotta, mert úgy véli, azok is megbízhatatlanok.
2. *A rendszerhiba költségei hatalmasak lehetnek.* Néhány alkalmazásnál, mint például egy reaktor irányítórendszere vagy egy repülőgép navigációs rendszere, a rendszerhiba költségei nagyságrendekkel nagyobbak, mint magának a rendszernek a költségei.
3. *A megbízhatatlan rendszerek az információ elvesztését eredményezhetik.* Az adatok összegyűjtése és kezelése nagyon drága; néha többet érnek, mint maga a számítógépes rendszer, amelynek segítségével feldolgozzák azokat. Rengeteg erőfeszítést és pénzt kell áldozni arra, hogy az adatokat többszörözzék és megvédjék a rongálódástól.

A kritikus rendszerek magas költségei miatt megbízható módszereket és technikákat kell alkalmaznunk azok fejlesztéséhez. Következésképpen, a kritikus rendszereket általában már jól bevált módszerekkel fejlesztik, nem pedig olyan új technikákkal, amelyeket még nem sokszor alkalmaztak. A kritikus rendszerek fejlesztői általában konzervatívak. Előnyben részesítik az olyan régebbi technikákat, amelyek jó és rossz tulajdonságait már ismerik, és nem szívesen használnak olyan új technikákat, amelyek lehet, hogy jobbak, de hosszú távú problémái még ismeretlenek.

Olyan szoftvertervezési technikákat is alkalmazhatnak azonban, amelyek nem kritikus rendszerek esetében nem gazdaságosak. Például sikeresen alkalmaztak már kritikus rendszerek biztonságosságához és biztonságához formális matema-

tikai módszereket is (10. fejezet) (Hall, 1996; Hall és Chapman, 2002). Ennek oka, hogy csökkenti a szükséges tesztelés mennyiségét. Kritikus rendszereknél a verifikáció és validáció költségei általában nagyon magasak, elérhetik a teljes rendszerfejlesztési költség 50 százalékát is.

Habár csak kevés számú vezérlőrendszer működik teljesen automatikusan, a legtöbb kritikus rendszer szociotechnikai rendszer, ahol emberek monitorozzák és felügyelik a számítógépes rendszer működését. A kritikus rendszerek meghibásodásainak költségei általában igen magasak, és szükségünk van olyan emberekre is, akik ki tudják az illető hibát javítani. Természetesen a rendszeroperátorok megoldhatják a problémát, de újabbat is okozhatnak, ha hibáznak.

Háromféle „rendszerösszetevő" van, amely hibára hajlamos:

1. Rendszerhardver, amely tervezési hibák miatt romolhat el, illetve az alkatrészek romolhatnak el gyári hiba miatt, vagy mert élettartamuk végére érnek.
2. Rendszerszoftver, amely a specifikációban, a tervezésben vagy az implementációban elkövetett hiba miatt romolhat el.
3. A rendszer használói, akik nem működtetik helyesen a rendszert. Ahogy a hardver és a szoftver egyre megbízhatóbb, úgy ez utóbbi jelenti leginkább a problémát.

Ezek a hibák általában összefüggnek. Egy meghibásodott hardverkomponens azt eredményezheti, hogy a rendszer üzemeltetői nem várt szituációval kerülnek szembe, túlterheltséghez és stresszhez vezethet, és ilyenkor az emberek gyakrabban hibáznak, a gyakoribb hibák még több stresszhez vezethetnek, ami még több hibát eredményez...

Nagyon fontos tehát, hogy a kritikus rendszerek tervezői a rendszer egészét tekintsék, mintsem annak egy aspektusát. Ha a hardver-, szoftver- és az üzemeltetési folyamatokat különállóan tervezték, nagy esély van arra, hogy hiba következzen be a rendszerrészek közötti interfészek esetében.

3.1. EGYSZERŰ BIZTONSÁGOSSÁGKRITIKUS RENDSZER

Számos különböző számítógép-alapú kritikus rendszer létezik a vezérlőrendszerektől kezdve az e-kereskedelmi rendszerekig. Mindegyik jó esettanulmány lenne egy szoftvertervezői könyvhöz, amelyen be lehet mutatni a szoftvertervezői technikákat. Mindazonáltal ezeknek a rendszereknek a megértése nehéz, mert értenünk kell magához az alkalmazási szakterülethez is, ahol a rendszer majd üzemelni fog.

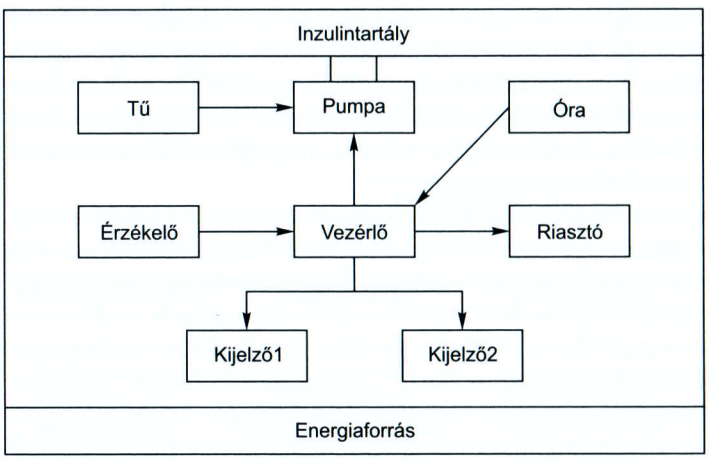
A jelen könyvben használt esettanulmány egy orvosi rendszer, amely a hasnyálmirigy működését szimulálja. Azért választottam ezt, mert általában mindenkinek van valamilyen ismerete az ilyen jellegű orvosi panaszokkal kapcsolatban, és ebben az esetben a biztonságosság és megbízhatóság kérdése is tisztábban értelmezhető. A rendszer arra használható, hogy segítsen a cukorbetegeken.

A cukorbetegség viszonylag gyakori betegség, amelynél az emberi szervezet képtelen megfelelő mennyiséget termelni az inzulin nevű hormonból. Az inzulin a vérben levő glükózt alakítja át. A cukorbetegség hagyományos kezelése során géntechnológiával létrehozott inzulint adnak be a betegnek injekcióval.

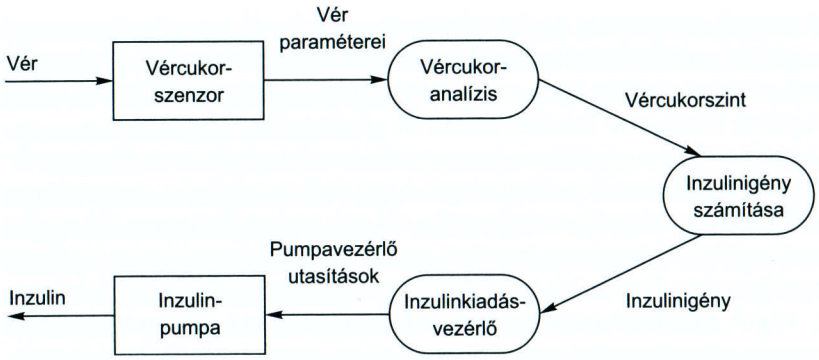
Ezzel a kezeléssel az a probléma, hogy a vér inzulinszintje nem függ a vércukor szintjétől, hanem annak a függvénye, hogy mikor adták be az inzulininjekciót. Ez túl alacsony vércukorszinthez (ha túl sok az inzulin) vagy túl magas vércukorszinthez (ha túl kevés az inzulin) vezethet. Az alacsony vércukorszint rövid távon súlyosabb állapot kialakulásához, az agy működésének ideiglenes leállásához, végül eszméletvesztéshez és halálhoz vezethet. Hosszú távon az állandóan magas vércukorszint szemkárosodáshoz, vesekárosodáshoz és szívproblémákhoz vezethet.

A miniatürizált érzékelők fejlesztésében elért legújabb eredményeknek köszönhetően már lehetőség van automatikus inzulinadagoló rendszerek létrehozására. Ezek figyelik a vércukor szintjét, és megfelelő adag inzulint adnak be, amikor szükséges. Ilyen inzulinadagoló rendszerek már léteznek a kórházi betegek kezelésére. A jövőben talán sok cukorbetegnek lesz ilyen rendszere, állandóan a testéhez erősítve.

Az inzulinadagoló rendszer működhetne a páciensbe beépített mikroszenzor segítségével, amely egy olyan vérparamétert mér, amely arányos a cukorszinttel. Ezt elküldi a pumpairányítóba, amely kiszámítja a cukorszintet, és eldönti, mennyi inzulin szükséges, majd jeleket küld egy miniatürizált pumpának, hogy adja be az inzulint egy tartósan odaerősített tűn keresztül. Az inzulinadagoló rendszereket valószínűleg szoftver fogja irányítani. A 3.1. ábrán láthatók az inzulinpumpa részei és elrendezésük. A 3.2. ábra egy adatfolyammodell, amely azt mutatja be, hogyan alakul át a vércukorszint (input) a pumpairányító parancsainak egy sorozatává.



3.1. ábra. Az inzulinpumpa struktúrája



3.2. ábra. Az inzulinpumpa adatfolyammodellje

Az üzembiztonság két legfontosabb összetevője az alábbi:

1. Fontos, hogy a rendszer elérhető legyen, hogy beadja az inzulint, amikor szükséges.
2. Fontos, hogy a rendszer megbízhatóan és biztonságosan működjön, és az aktuális vércukorszintnek megfelelő mennyiségű inzulint adagolja, nem veszélyeztetve a beteget.

Az ilyen rendszerek meghibásodása, az inzulin túladagolása veszélyeztetheti a felhasználó életét. Ezért kifejezetten fontos, hogy túladagolás nem fordulhat elő.

3.2. A RENDSZER ÜZEMBIZTONSÁGA

Bizonyára mindannyian ismerjük a számítógépes rendszerek hibáit. Valamilyen rejtélyes okból a számítógépes rendszerek néha tönkremennek, és nem sikerül végrehajtaniuk a felhasználó által kért szolgáltatást. Lehet, hogy az ilyen gépeken futó programok nem úgy működnek, ahogy kellene, és előfordulhat, hogy kárt tesznek azokban az adatokban, amelyeket a rendszer kezel. Megtanultunk együtt élni ezekkel a hibákkal, és többségünk teljesen megbízik azokban a személyi számítógépekben, amelyeket használunk.

Egy számítógépes rendszer üzembiztonsága a rendszernek az a tulajdonsága, amely a megbízhatóságnak felel meg. A megbízhatóság lényegében azt mutatja meg, hogy a felhasználó mennyire bízik abban, hogy a rendszer úgy fog működni, ahogy azt elvárják, és normális használat során nem romlik el. Ezt a tulajdonságot nem lehet numerikusan kifejezni, de használunk olyan viszonylagos kifejezéseket, mint például a „nem üzembiztos", „nagyon üzembiztos" és „ultraüzembiztos", hogy kifejezzük, mennyire bízunk a rendszerben.

A megbízhatóság és a hasznosság természetesen nem ugyanazt jelenti. A szövegszerkesztő, amit a jelen könyv megírására használtam, véleményem szerint nem nagyon üzembiztos rendszer, de nagyon hasznos. A rendszerbe vetett bizalmam hiánya abban nyilvánul meg, hogy gyakran mentem a munkámat, és

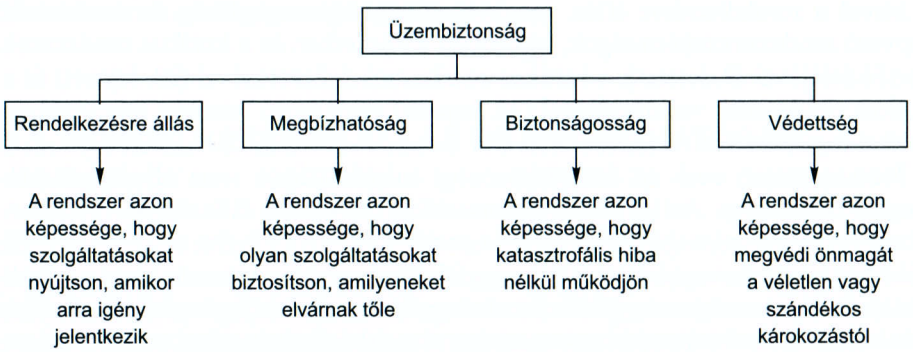
több másolatot is készítek. Így ellensúlyozom a rendszer üzembiztonságának hiányosságát, és csökkentem az esetleges rendszerhiba által okozott kárt.

Az üzembiztonságnak négy jellegzetes összetevője van, amelyeket a 3.3. ábrán láthatunk:

1. *Rendelkezésre állás.* Egy rendszer rendelkezésre állása (elérhetősége) annak a valószínűsége, hogy bármely adott pillanatban működik és képes végrehajtani a parancsokat.
2. *Megbízhatóság.* Egy rendszer megbízhatósága annak a valószínűsége, hogy egy adott időtartamon belül a rendszer helyesen teljesíti a felhasználó által adott parancsokat.
3. *Biztonságosság.* Egy rendszer biztonságossága azt mutatja meg, hogy mennyire képes úgy működni, hogy ne okozzon kárt az emberekben vagy a környezetében.
4. *Védettség.* Egy rendszer védettsége azt mutatja meg, hogy a rendszer mennyire képes ellenállni véletlen vagy szándékos károkozásnak.

Ezek komplex tulajdonságok, amelyek számos más egyszerű tulajdonságból származhatnak. Például a *védettség* magában foglalja az *integritás*t (annak a biztosítását, hogy egy rendszerprogram vagy adat nem sérül), valamint a *bizalmasság*ot is (annak biztosítását, hogy a bizalmas adatokhoz csak a jogosultak férjenek hozzá). A *megbízhatóság* magában foglalja a *helyesség*et (annak biztosítása, hogy a rendszer szolgáltatásai jól specifikáltak), a *precizitás*t (a szolgáltatott információk megfelelően részletesek), illetve az *időszerűség*et (hogy a szolgáltatott információk a megfelelő időben érkeznek meg).

Az üzembiztonság tulajdonságai, a rendelkezésre állás, a megbízhatóság, a biztonság és a védelem összefonódnak. A védett rendszerek műveletei függnek a rendszer rendelkezésre állásától és megbízhatóságától. Egy rendszer megbízhatatlanná válik, ha az adatait felülírta egy behatoló. Egy szolgáltatás megtagadását eredményező támadás elérhetetlenné tehet egy rendszert. Ha egy védett rendszert megfertőz egy vírus, annak műveletei már nem tekinthetők biztonságosnak. A fogalmak összefonódásából jól látható, hogy az üzembiztonság egy összetett tulajdonság.



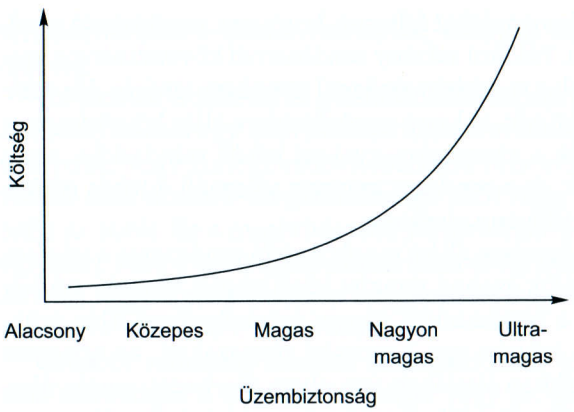
3.3. ábra. Az üzembiztonság dimenziói

Ehhez a négy fő dimenzióhoz hasonlóan a többi rendszertulajdonság is kapcsolható az üzembiztonsághoz:

1. *Javíthatóság.* A rendszerhibák elkerülhetetlenek, de az általuk keltett zavarok minimalizálhatók, ha a rendszer gyorsan javítható. Ebből szükséges a probléma diagnosztizálhatósága, a hibát okozó komponens elérhetősége, valamint a komponens kijavíthatósága. Egy szervezet által használt szoftver javíthatósága fokozható, ha a szervezet hozzáférést biztosít a szoftver forráskódjához, és rendelkezik a kód megváltoztatásához elengedhetetlen szakértelemmel. Sajnálatos módon ez egyre inkább lehetetlenné válik, ahogy egyre több harmadik fél által szállított, úgynevezett feketedoboz-komponenseket alkalmazunk a fejlesztések során (19. fejezet).
2. *Karbantarthatóság.* A rendszer használata során új követelmények jelenhetnek meg. Azok követése, a rendszerhez való illesztése nagyon fontos a rendszer használhatósága szempontjából. A karbantartható szoftver olyan szoftver, amely gazdaságosan adaptálható az új követelményekhez, és kicsi annak a valószínűsége, hogy a módosítások új hibákat okoznának a rendszerben.
3. *Túlélési képesség.* Az internetalapú rendszerek esetében ez egy nagyon fontos tulajdonság, és szorosan kapcsolódik a biztonságossághoz és a rendelkezésre álláshoz (Ellison *és mások,* 1999). A túlélési képesség annak a képessége, hogy a rendszer folytatni tudja a szolgáltatási működését egy esetleges támadás alatt is, akár a rendszer bizonyos részeinek kiesése mellett is. Ehhez az kell, hogy azonosítsuk a kulcsrendszerkomponenseket, és gondoskodjunk róla, hogy azok mindig biztosítsanak egy minimális szolgáltatást. Három fő stratégiával javítható a túlélési képesség: a támadással szembeni ellenállás, a támadás észlelése és a támadás okozta sérülések kijavítása (Ellison *és mások,* 1999, 2002).
4. *Hibatűrés.* Ez a tulajdonság tekinthető a rendszer használhatóságának (16. fejezet) részeként, és azt fejezi ki, mennyire terveztek meg egy rendszert úgy, hogy elkerülje a hibás felhasználói bemeneteket, és hogy mennyire tolerálja azokat. Amikor felhasználói hibák következnek be, a rendszernek (amennyire lehetséges) detektálnia kell azokat, és vagy automatikusan kijavítani őket, vagy felkérni a felhasználót az adatbevitel megismétlésére.

Mivel a rendelkezésre állás, megbízhatóság, biztonságosság és a védelem alapvető rendszertulajdonságok, így ebben a fejezetben és a kritikus rendszerek specifikációjával (9. fejezet), a kritikus rendszerek fejlesztésével (20. fejezet) és a kritikus rendszerek validációjával (24. fejezet) foglalkozó későbbi fejezetekben ezekkel foglalkozom.

Természetesen ezek az üzembiztonsági tulajdonságok nem alkalmazhatók minden rendszerre. Az inzulinpumpa esetében, melyet a 3.1. alfejezetben mutattam be, a legfontosabb tulajdonság a rendelkezésre állás (ha szükség van rá, működnie kell), a megbízhatóság (a megfelelő mennyiségű inzulint kell szolgáltatnia) és a biztonságosság (soha nem adagolhat nagyobb mennyiségű inzulint, mint kellene). A védelem ebben az esetben kevésbé fontos, mivel a pumpa nem hordoz bizalmas információkat, és nem biztosít hálózati szolgáltatást, így nincs kitéve rosszindulatú támadásoknak.



3.4. ábra. Költség/üzembiztonság görbe

A tervezőknek kompromisszumot kell kötniük a rendszer teljesítménye és az üzembiztonságosság között. Általában véve a nagyon üzembiztos rendszerek csak nagyon költségesen tudnak hatékonyan üzemelni. Az üzembiztos szoftver extra, gyakran redundáns kódokat tartalmaz a kivételes események lekezelésére és a hibák kijavítására. Ezek a kódok csökkentik a rendszer teljesítményét és növelik a szoftver méretét, és nem utolsósorban megnövelik a szoftver kifejlesztésének költségeit.

A járulékos tervezési, implementációs és validációs költségek miatt a rendszer üzembiztonságossága nagyban megnöveli a fejlesztési költségeket. Konkrétan a validációs költségek a legmagasabbak a kritikus rendszerek esetében. Ugyanis nemcsak azt kell validálni, hogy a rendszer megfelel-e a követelményeknek, hanem meg kell felelni bizonyos külső szabályozásoknak is (például a Szövetségi Légügyi Hatóság előírásainak) ahhoz, hogy a szoftver üzembiztonságos legyen.

A 3.4. ábra a költségek és az üzembiztonság járulékos növekedése közötti kapcsolatot mutatja meg. Ez az ábra teljesül az üzembiztonság mind a négy összetevőjére – a megbízhatóságra, rendelkezésre állásra, biztonságosságra és védettségre. Mivel ez a költség/üzembiztonság görbe exponenciális, nem lehet azt bizonyítani, hogy egy rendszer 100 százalékosan üzembiztos, mert az üzembiztonság biztosításának a költsége végtelen lenne.

3.3. RENDELKEZÉSRE ÁLLÁS ÉS MEGBÍZHATÓSÁG

A rendelkezésre állás (elérhetőség) és a megbízhatóság két, egymáshoz nagyon közeli fogalom, mindkettő valószínűségként adható meg. Egy rendszer rendelkezésre állása annak a valószínűsége, hogy a rendszer képes teljesíteni a felhasználó által kért szolgáltatásokat akkor, amikor kell. Egy rendszer megbízhatósága pedig annak a valószínűsége, hogy a rendszer úgy teljesíti a kért szolgáltatásokat, ahogy kérték tőle.

Jóllehet szoros a kapcsolat, de nem lehet feltenni, hogy egy megbízható rendszer mindig elérhető, és fordítva. Például néhány rendszernél követelmény a magas rendelkezésre állás, de kisebb a megbízhatósággal szembeni elvárás. Ha a felhasználó folyamatos szolgáltatást vár, akkor a rendelkezésre állás követelménye magas. Ugyanakkor elviselhetők a viszonylag gyakori hibák mindaddig, amíg ezek a hibák hamar kijavíthatók, és a rendszer gyorsan visszaáll a hibás működésből – a megbízhatósági követelmény alacsony.

Jó példa a magas fokú rendelkezésre állást megkövetelő rendszerre a telefonközpont kapcsolója. A felhasználók szabad vonalat jelző búgást várnak, amikor felemelik a telefonkagylót, így a rendszernek magas rendelkezésre állás szükséges. De ha egy rendszerhiba folytán egy kapcsolat megszakad, az könnyen helyreállítható. A vonalkapcsolókban általában van olyan szerkezet, amely újraindítja a rendszert és megpróbálja újra létrehozni a kapcsolatot. Ez elég gyors lehet, így a telefon használója talán észre sem veszi, hogy hiba történt. Ezért ilyen esetben inkább a rendelkezésre állás az alapvető üzembiztonsági követelmény, nem pedig a megbízhatóság.

További különbség e tulajdonságok között az, hogy a rendelkezésre állás nemcsak magától a rendszertől függ, hanem attól az időintervallumtól is, ami azon hibák helyreállításához szükséges, amelyek a rendszert elérhetetlenné teszik. Ezért ha az A rendszer egy évben egyszer romlik el, a B rendszer pedig havonta egyszer, akkor az A rendszer megbízhatóbb, mint a B. Ha azonban az A rendszer újraindítása három napig tart, a B rendszeré pedig 10 percig, akkor a B rendszer éves rendelkezésre állása nagyobb, mint az A rendszeré. A felhasználók valószínűleg a B rendszert választják szívesebben, nem az A-t.

A rendszer megbízhatósága és rendelkezésre állása pontosabban definiálható a következő módon:

1. *Megbízhatóság.* A hibamentes működés valószínűsége egy meghatározott idő alatt, egy adott környezetben és egy meghatározott céllal.
2. *Rendelkezésre állás.* Annak a valószínűsége, hogy a rendszer egy pillanatban működőképes és képes teljesíteni a kért szolgáltatásokat.

A megbízható rendszerek fejlesztésénél az egyik gyakorlati probléma az, hogy a megbízhatóságról és a rendelkezésre állásról alkotott ösztönös fogalmaink ezeknél a korlátozott definícióknál gyakran tágabbak. A *megbízhatóság* definíciója szerint azt a környezetet, amelyben a rendszert használják, és azt a célt, amellyel használják, is figyelembe kell venni, amikor a megbízhatóságát vizsgáljuk. Ezért a rendszer megbízhatóságának mértéke az egyik környezetben nem feltétlenül egyezik meg ugyanezen rendszer megbízhatóságának mértékével egy másik környezetben, ahol másra használják.

Mérhetjük például egy szövegszerkesztő megbízhatóságát irodai környezetben, ahol a felhasználókat nem érdekli a rendszer működése. Követik a használati utasításokat, és nem próbálnak kísérletezni a rendszerrel. Egyetemi környezetben a megbízhatóság kissé más lehet. Itt a diákok megpróbálják a rendszer korlátait felderíteni, és nem várt módokon használhatják a rendszert. Ezek olyan

rendszerhibákat eredményezhetnek, amelyek irodai környezetben nem fordulnak elő.

Az emberi érzékelés és a felhasználás módja is fontos. Tekintsük például azt az esetet, amikor egy autó ablaktörlőjének rendszere meghibásodik, aminek eredménye az, hogy zuhogó esőben időnként nem működik az ablaktörlő. A rendszer vezető által érzékelt megbízhatósága attól függ, hogy hol él a vezető, hol használja az autót. Ez a meghibásodás jobban érinti a Seattle-ben élő vezetőt (nedves éghajlat), mint a Las Vegasban élőt (száraz éghajlat). A seattle-i vezető észrevétele az lesz, hogy a rendszer nem megbízható, míg előfordulhat, hogy a Las Vegas-i soha nem veszi észre a problémát.

További nehézség ezekkel a definíciókkal kapcsolatban az, hogy nem veszik számításba a hiba komolyságát vagy az elérhetetlenség következményeit. Természetesen az embereket jobban érdeklik azok a hibák, amelyeknek súlyos következményei vannak, és a rendszer megbízhatóságának észlelését ezek a következmények befolyásolják. Például egy autó motorja, ami indítás után kihagy, de újraindítás után mindig rendesen működik, idegesítő. Ez azonban nem befolyásolja a normális működést, és sok vezető nem gondolná, hogy emiatt az autó megbízhatatlan. Ezzel szemben, ha motorvezetés közben, nagy sebességnél mondjuk havonta egyszer kihagy, akkor a legtöbb vezető azt gondolja, hogy megbízhatatlan.

A megbízhatóság szigorú definíciója a rendszer implementációját összefüggésbe hozza a specifikációjával. Vagyis a rendszer akkor működik megbízhatóan, ha a működése megfelel a specifikációjában definiáltaknak. Az érzékelt megbízhatatlanság gyakori oka azonban, hogy a rendszer specifikációja nem felel meg a felhasználó elvárásainak. Sajnos sok specifikáció nem teljes vagy hibás, és a szoftvertervező feladata értelmezni, hogyan kell a rendszernek viselkednie. Mivel ők nem a szakterület szakemberei, nem biztos, hogy produkálni tudják azt a működést, amit a felhasználó elvár.

A megbízhatóságot és rendelkezésre állást veszélyeztetik a rendszerhibák. Előfordulhat, hogy a rendszer nem teljesíti, nem a kért módon vagy nem biztonságosan teljesíti a szolgáltatást. E hibák egy része specifikációs hiba vagy a kapcsolódó rendszerek (például a telekommunikációs rendszer) nem megfelelő működéséből ered. Sok sikertelenség azonban a rendszer meghibásodásából származó hibás működés következménye. A megbízhatóság tárgyalásánál érdemes különbséget tenni a *sikertelenség*, a *működési hiba* és a *rendszerhiba* fogalma között. A 3.5. ábra tartalmazza e fogalmak definícióit.

Az emberi hibák nem mindig vezetnek rendszerhibákhoz. Lehet, hogy hatásukra olyan rendszerrészek hibásodnak meg, melyek sosem kerülnek használatra. A sikertelenségek sem mindig eredményeznek rendszerhibákat, mert lehet, hogy egy sikertelen állapot csak átmeneti, és kijavításra kerül, mielőtt a rendszer hatására tévesen kezdene el működni. A működési hibák sem mindig vezetnek rendszerhibákhoz, mert a hibás viselkedés is lehet átmeneti, és így lehet, hogy nem lesz megfigyelhető hatással a rendszerre, vagy a rendszer eleve tartalmaz valamilyen védelmet, és így a hibás működés kijavításra kerül, mielőtt a rendszer által nyújtott szolgáltatások sérülnének.

Fogalom	Leírás
Sikertelenség	Olyan esemény, amely akkor következik be, ha a rendszer nem nyújtja a felhasználói által elvárt szolgáltatást.
Rendszerhiba	Hibás a rendszer viselkedése, amikor a rendszer működése nem felel meg a specifikációjának.
Működési hiba	A szoftverrendszer olyan jellemzője, amely rendszerhibához vezet. Például egy változó hibás inicializálása azt eredményezi, hogy hivatkozásnál az értéke rossz lesz.
Emberi hiba vagy tévesztés	Olyan emberi tevékenység, amely a rendszer hibás működését eredményezi.

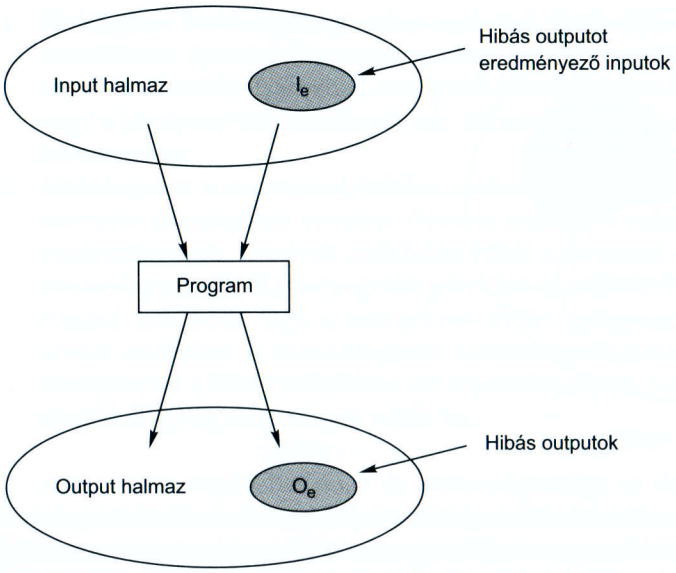
3.5. ábra. A megbízhatóság fogalmi rendszere

A 3.5. ábrán láthatjuk a különböző fogalmak közötti különbséget, ami hasznos, mert segít felismerni azt a három alternatív szemléletet, amelyeket egy rendszer megbízhatóságának a növelésénél alkalmazhatunk:

1. *Hiba elkerülése.* Olyan fejlesztési technikákat alkalmazunk, amelyek minimalizálják a hibák valószínűségét, és/vagy összegyűjtik a hibákat, mielőtt azok meghibásodáshoz vezetnek. Ilyen technika például a hibákra hajlamos programnyelvstruktúrák elkerülése, mint például a mutatók és a program rendellenességeinek keresésére használt statikus analízis.
2. *Hibakeresés és -eltávolítás.* Olyan verifikációs és validációs technikák alkalmazása, amelyek növelik annak esélyét, hogy a hibákat felismerjük és megszüntessük, még mielőtt a rendszer használatára sor kerül. Ilyen hibakeresési technikák például a szisztematikus rendszertesztelés és a nyomkövetés.
3. *Hibatűrés.* Olyan technikák alkalmazása, amelyek biztosítják, hogy a működési hibák nem okoznak rendszerhibákat, vagy a rendszerhibák nem okoznak sikertelenségeket. Ilyen módszer például az önellenőrző eszközök beépítése a rendszerbe és redundáns rendszermodulok alkalmazása.

A hibatűrő rendszerek fejlesztését a 20. fejezetben tárgyalom, ahol röviden szólok majd néhány hibaelkerülő módszerről is. A hibaelkerülés folyamatalapú szemléletéről a 27. fejezetben lesz szó. A hibakeresést a 22. és a 23. fejezetben tárgyalom.

A szoftver hibás működése sikertelenséget eredményez, amikor a hibás kódot végrehajtják olyan inputtal, amely a felszínre hozza a szoftver hibáját. A legtöbb input esetében a kód megfelelően működik. A 3.6. ábra, amely Littlewoodtól származik (Littlewood, 1990), a szoftverrendszert egy input halmaznak egy output halmazra való leképezéseként ábrázolja. Egy programnak több inputja lehet (az egyszerűség kedvéért az összetett inputokat és inputok sorozatát egyszerű inputnak tekintjük). A program ezekre az inputokra egy vagy több outputtal válaszol. Például egy webböngésző inputja lehet egy URL, amire az output a kért lap megjelenítése.

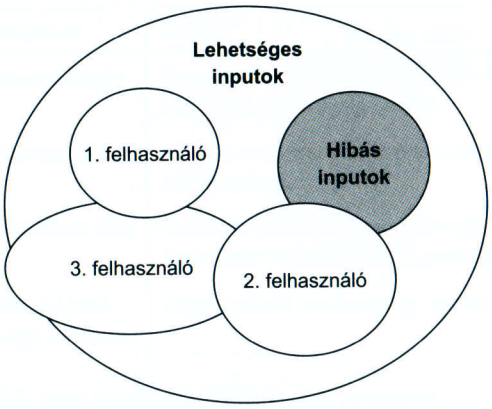


3.6. ábra. Egy rendszer mint input/output leképezés

Ezen inputok közül néhány (amelyet a 3.6. ábrán a szürke ellipszis jelöl) a rendszer sikertelenségét eredményezi, ahol a program hibás outputokat generál. A rendszer megbízhatósága arányos annak a valószínűségével, hogy a program egy adott alkalmazásánál a rendszerinput annak az input halmaznak lesz az eleme, amely hibás outputot eredményez. Általában számos olyan eleme van a rendszernek, amit nagyobb valószínűséggel használnak. Az itt bekövetkezett hibák tehát gyakrabban fognak előfordulni.

Egy rendszer minden egyes felhasználója más-más módon használja a rendszert. Azok a működési hibák, amelyek a rendszer megbízhatóságát befolyásolják az egyik felhasználó esetében, talán soha nem nyilvánulnak meg egy másik alkalmazás során (3.7. ábra). A 3.7. ábrán a 3.6. ábrán besatírozott ellipszisnek megfelelő hibás inputok halmaza van besatírozva. A 2. felhasználó inputjainak halmaza metszi a hibás inputok halmazát. Ezért a 2. felhasználó találkozik néhány működési hibával. Az 1. és a 3. felhasználó azonban soha nem használ inputot a hibás halmazból. Számukra a szoftver mindig megbízható lesz.

Egy program megbízhatósága leginkább a rendszer normális használata során hibás outputot eredményező inputok számától függ. Nem jelent túl nagy különbséget az érzékelt megbízhatóság szempontjából az, ha eltávolítjuk a hibás működést okozó szoftverelemeket a rendszer ritkán használt részeiből. Millsék (Mills *és mások*, 1987) azt tapasztalták a szoftverük esetében, hogy a termék hibáinak 60 százalékát eltávolítva csak 3 százalékkal nőtt a megbízhatóság. Ezt megerősítette egy másik tanulmány, amelyben az IBM szoftvertermékeinél előforduló hibákat vizsgálták. Adams (Adams, 1984) szerint a termékekben levő hibák nagy része csak több száz vagy ezer hónap használat után okoz sikertelenséget.



3.7. ábra. Szoftverhasználati minták

A szociotechnikai rendszerek felhasználói adaptálhatnak szoftvereket, amelyeknek ismerik a hibáit, és megoszthatják egymás között azokat az információkat, amelyekkel megmenekülhetnek a hibás működéstől. Így elkerülhetik az olyan inputok használatát, amelyek sikertelen működéshez vezetnek, és szoftverhibák szinte sohasem fordulnak elő. Sőt a tapasztalt felhasználók gyakran „körbedolgozzák" a hibákat. Nem használják a problémás funkciókat. Például én szándékosan nem használom a szövegszerkesztő automatikus számozását a felsorolásokban, míg ezt a könyvet írom. A felhasználók szemében a rendszerek ilyen hibáinak kijavítása nincs gyakorlati hatással a rendszer megbízhatóságára nézve.

3.4. BIZTONSÁGOSSÁG

Ha egy kritikus rendszer alapvető sajátossága a biztonságosság, akkor az a rendszer biztonságosságkritikus rendszer. Egy ilyen rendszer soha nem veszélyeztetheti az embert vagy a rendszer környezetét. Biztonságosságkritikus rendszerek például a repülőgépek vezérlő- és megfigyelőrendszerei, vegyi és gyógyszergyárakban alkalmazott folyamatirányító rendszerek és a gépkocsik vezérlőrendszerei.

Biztonságosságkritikus rendszerek hardverirányítását egyszerűbb kivitelezni és elemezni, mint a szoftverirányítást. Manapság azonban olyan bonyolult rendszereket építünk, amelyeket nem lehet csak hardveresen vezérelni. Valamennyi szoftverirányítás szükséges, mert rengeteg érzékelőt és működtetőt kell kezelni bonyolult vezérlési szabályok alapján. Ilyen fokú bonyolultságot fejlett katonai repülőgépeknél láthatunk, amelyek aerodinamikailag instabilak. Ahhoz, hogy ne zuhanjanak le, a szoftvervezérlő folyamatos működése szükséges.

A biztonságosságkritikus szoftverek két osztályba sorolhatók:

1. *Elsődlegesen biztonságosságkritikus szoftverek.* Ezek olyan szoftverek, amelyeket vezérlőként építenek be a rendszerbe. Az ilyen szoftverek hibás működése olyan hardverhibát okozhat, amelynek következtében emberek sérülnek meg, vagy a környezetben keletkezik kár. Mi most ilyen típusú szoftverekre fogunk koncentrálni.
2. *Másodlagosan biztonságosságkritikus szoftverek.* Ezek olyan szoftverek, amelyek közvetve okozhatnak sérülést. Ilyenek például a számítógépes műszaki tervezőrendszerek, amelyek működési hibái a tervezett dolog tervezési hibáját eredményezhetik. Ez fenyegetést jelenthet az emberre, ha a tervezett rendszer rosszul működik. Egy, a betegeknek előírt gyógyszerek adatait tartalmazó orvosi adatbázis is másodlagosan biztonságosságkritikus szoftver. Ennek a rendszernek a hibás működése azt eredményezheti, hogy a betegeknek nem a megfelelő gyógyszeradagot adják be.

A rendszer megbízhatósága és biztonságossága az üzembiztonság egymáshoz kapcsolódó, de különálló sajátosságai. Természetesen egy biztonságosságkritikus rendszernek megbízhatónak kell lennie annyiban, hogy meg kell felelnie a specifikációjának, és hiba nélkül kell működnie. Rendelkezhet hibatűrő vonásokkal úgy, hogy állandó szolgáltatást nyújt működési hiba esetén is. A hibatűrő rendszerek azonban nem feltétlenül biztonságosak. A szoftver mégis működhet rosszul, és eredményezhet olyan működést, amely balesetet okoz.

Eltekintve attól a ténytől, hogy soha nem lehetünk biztosak abban, hogy egy szoftverrendszer hibamentes vagy hibatűrő, néhány egyéb oka is van annak, hogy néhány szoftverrendszer megbízható, de nem biztonságos:

1. Előfordulhat, hogy a specifikáció nem teljes, vagyis nem írja le a rendszer megfelelő működését néhány kritikus helyzetben. A rendszer hibás működése jórészt specifikáció miatt lép fel, és nem tervezési hiba miatt (Nakajo és Kume, 1991; Lutz, 1993). A beágyazott rendszerekben előforduló működési hibák vizsgálatánál egy tanulmányában Lutz az alábbi következtetésre jutott:
   ...a követelményekkel kapcsolatos nehézségek a legfőbb előidézői a biztonságossággal kapcsolatos szoftverhibáknak, amelyek az integrációig és a rendszer teszteléséig megmaradtak.
2. A hardver rossz működése is kiválthatja, hogy egy rendszer kiszámíthatatlan módon működjön, és váratlan környezetet biztosíthat a szoftvernek. Amikor az alkotórészek az élettartamuk végéhez közelednek, akkor előfordulhat, hogy szabálytalanul működnek, és olyan jeleket generálnak, amelyek kívül esnek a szoftver által kezelhető jelek terjedelmén.
3. A rendszer operátora generálhat olyan inputokat, amelyek önmagukban nem helytelenek, de egy bizonyos szituációban a rendszer helytelen működéséhez vezethetnek. Anekdotikus példa erre az az eset, amikor a szerelő utasította a repülőgép vezérlőszoftverét, hogy húzza be a futóművet. A szoftver végrehajtotta az utasítást annak ellenére, hogy a repülőgép a földön állt!

A biztonságosságkritikus rendszerek tárgyalásánál speciális kifejezéseket használunk, és fontos, hogy pontosan értsük a használt kifejezéseket. A 3.8. ábrán néhány olyan definíciót mutatok be, amelyek az eredetileg Leveson (Leveson, 1985) által definiált fogalmakból származnak.

Fogalom	Definíció
Baleset (vagy szerencsétlenség)	Olyan váratlan esemény vagy események sorozata, amely ember(ek) halálát vagy megsebesülését okozza, illetve kárt tesz a tulajdonban vagy a környezetben. Balesetre példa egy számítógéppel vezérelt gép, amely operátorát megsebesíti.
Veszély	Olyan körülmény, amely potenciálisan balesetet okozhat vagy hozzájárulhat baleset bekövetkeztéhez. Veszélyre példa annak az érzékelőnek a hibája, amelynek a felbukkanó akadályokat kell érzékelnie.
Kár	A szerencsétlenség okozta kár mértéke. A károk a kisebb sebesülésektől vagy tulajdonban esett károktól a több ember halálát okozó balesetekig terjedhetnek.
Veszély komolysága	Egy bizonyos veszélyből származható lehető legnagyobb kárra vonatkozó becslés. A veszély komolysága a katasztrofálistól, mely következtében akár emberek is meghalhatnak, a jelentéktelenig terjedhet, amely csak kisebb kárt okoz.
Veszély valószínűsége	A veszélyt okozó események bekövetkezésének valószínűsége. A valószínűségi értékek tetszőlegesen változhatnak a *valószínűtől* (a veszély bekövetkeztének esélye, mondjuk 1/100) a *valószínűtlenig* (nem képzelhető el olyan helyzet, amelyben a veszély bekövetkezhetne).
Kockázat	Ez annak a valószínűségének a mértéke, hogy a rendszer szerencsétlenséget okoz. A kockázat felmérésekor tekintetbe veszik a veszély valószínűségét, a veszély komolyságát és annak valószínűségét, hogy a veszély balesetet eredményez.

3.8. ábra. A biztonságosság terminológiája

A biztonságosság kulcsa annak elérése, hogy ne történjenek balesetek, vagy ha mégis előfordulnak, akkor azoknak minimális következményei legyenek. Ezt három módon érhetjük el:

1. *A veszély elkerülése.* A rendszert úgy tervezik, hogy kiküszöbölje a veszélyeket. Például egy vágórendszer, ami két gomb egyidejű megnyomásával működtethető, elkerüli annak a kockázatát, hogy a működtető keze a penge útjába kerüljön.
2. *A veszély felderítése és eltávolítása.* A rendszert úgy tervezik, hogy a veszélyeket megkeresse és megszüntesse, mielőtt még bajt okoznának. Például egy vegyi üzemben levő rendszer érzékelheti a túl nagy nyomást és kinyithat egy nyomáscsökkentő szelepet, mielőtt még robbanás következne be.

3. *A károk korlátozása.* A rendszer tartalmazhat olyan védelmi jellemzőket, amelyek minimalizálják azokat a károkat, amelyek egy baleset során bekövetkezhetnek. Például egy repülőgép motorjában lehetnek automatikus tűzoltó készülékek. Ha tűz üt ki, akkor ezek gyakran eloltják, még mielőtt megrémisztené az utasokat vagy a legénységet.

Balesetek általában akkor történnek, amikor egyidejűleg több dolog romlik el. Komoly baleseteket elemezve Perrow (Perrow, 1984) azt tapasztalta, hogy szinte mindegyik hibás működések kombinációjának volt a következménye, nem pedig egy egyszerű hibáé. A váratlan kombináció olyan kölcsönhatásokhoz vezetett, amelyeknek az eredménye rendszerhiba lett. Perrow szerint lehetetlen előre kiszámítani a rendszer működési hibáinak valamennyi kombinációját, és a balesetek elkerülhetetlen velejárói a bonyolult rendszerek használatának. A szoftver használata általában növeli a rendszer bonyolultságát, így a szoftvervezérlés növelheti a balesetek bekövetkezésének a valószínűségét.

A szoftverek vezérlése és monitorozása megnövelheti a rendszer biztonságosságának fokát. A szoftvervezérelt rendszerek a feltételek sokkalta szélesebb skáláját monitorozhatják, mint a hagyományos elektromechanikai rendszerek. És ezenfelül relatív könnyen adaptálhatók. Ezek magukban foglalják az olyan hardver használatát is, mely jellegénél fogva nagy megbízhatósággal rendelkezik, de emellett fizikailag kicsi és könnyű. A szoftvervezérelt rendszerek kifinomult biztonsági zárakat biztosíthatnak. Támogathatják a vezérlési stratégiákat, és csökkenthetik az emberek által veszélyes környezetben eltöltendő időt. És habár a szoftvervezérlés megnöveli egy rendszer elromolhatóságának lehetőségeit, nagyobb monitorozási lehetőségével és védelmével mégiscsak megnöveli a rendszer biztonságosságának szintjét.

Minden esetben fontos figyelmet szentelni a rendszer biztonságosságára. Lehetetlen egy rendszert 100 százalékban biztonságosra elkészíteni, és a társadalomnak kell eldöntenie, hogy egy-egy nem biztonságos esemény elkerülésére befektetett erők és az alkalmazott új technológiák megérik-e az erőfeszítéseket vagy sem. Szintén szociális és politikai döntés az is, hogy milyen mértékben alkalmazhatók a korlátozott nemzeti erőforrások a népességre leselkedő kockázatok elkerülése érdekében.

3.5. VÉDETTSÉG

Egy rendszer védettsége azt mutatja meg, hogy mennyire képes a rendszer megvédeni magát véletlen vagy szándékos külső támadásoktól. A védettség fontossága egyre nő, ahogy egyre több rendszer kapcsolódik az internethez. Az internetkapcsolatok a rendszer további funkcióit teszik lehetővé (például az ügyfelek közvetlenül hozzáférhetnek a bankszámlájukhoz), de ez azt is jelenti, hogy a rendszert rossz szándékú emberek megtámadhatják. Az internet azt is lehetővé teszi, hogy a rendszer bizonyos sérülékenységei elterjedjenek, és ezáltal a rend-

szert több ember támadhatja meg. Persze a kapcsolatok növekvő száma a támadási felületetek csökkentését szolgáló hibajavítások gyors elterjedését is szolgálhatja.

A támadások lehetnek például a vírusok, a rendszer szolgáltatásainak jogtalan használata, a rendszer vagy annak adatainak engedély nélküli megváltoztatása stb. A védettség minden kritikus rendszernél fontos. Megfelelő szintű védettség hiányában a rendszer rendelkezésre állása, megbízhatósága és biztonságossága veszélybe kerülhet, ha külső hatások kárt okoznak a rendszerben.

Ennek az az oka, hogy minden módszer, amely a rendelkezésre állást, a megbízhatóságot és biztonságosságot szolgálja, azon a tényen alapul, hogy az operációs rendszer pontosan az a rendszer, amelyet installáltak. Ha ez az installált rendszer valamilyen módon megsérül (például a szoftver vírusos lesz), akkor a megbízhatóság és biztonságosság eredeti argumentumai már nem lehetnek érvényesek. A rendszerszoftver sérülhet, és kiszámíthatatlanul működhet.

Ezzel szemben a rendszer kifejlesztéséből adódó hibák biztonsági réseket okozhatnak. Ha egy rendszer nem reagál a váratlan inputokra, vagy nem ellenőrzi a tömbhatárokat, akkor a támadók kihasználhatják ezeket a gyengeségeket, és hozzáférhetnek a rendszerhez. A nagyobb biztonsági incidensek, például az internetféreg (Spafford, 1989) vagy 10 évvel később a Code Red féreg (Berghel, 2001) kihasználták, hogy a C nyelven írt programokban nincs tömbhatár-ellenőrzés, így a memória azon része, amely a rendszerhez való jogos hozzáférést ellenőrző kódot tartalmazza, felülírható.

Természetesen vannak olyan kritikus rendszerek, amelyeknél a rendszer üzembiztonságának legfontosabb dimenziója a védettség. A katonai rendszereket, az elektronikus kereskedelemben használt rendszereket és a bizalmas információ kezelésére használt rendszereket úgy kell tervezni, hogy a védettségi szintjük elég magas legyen. Ha egy repülőtéri helyfoglaló rendszer nem elérhető, kényelmetlenséget és a jegy kiadásának késedelmét okozza. Ha azonban a rendszer nem elég védett és hamis helyfoglalásokat is elfogad, akkor az a légitársaság jókora összeget veszíthet.

A külső támadások háromféle kárt okozhatnak:

1. *A szolgáltatás megtagadása.* A rendszer olyan állapotba kerülhet, amelyben a normális szolgáltatások elérhetetlenné válnak. Ez nyilvánvalóan befolyásolja a rendszer rendelkezésre állását.
2. *A programok vagy adatok megrongálása.* A rendszer szoftveres elemeit engedély nélkül megrongálhatják. Ez befolyásolhatja a rendszer működését és így annak megbízhatóságát és biztonságosságát. Ha a kár komoly, akkor a rendszer megbízhatósága is sérülhet.
3. *Bizalmas információ kiadása.* A rendszer által kezelt információ lehet titkos, és a külső támadás ezeket kiszolgáltathatja jogosulatlan személyeknek. Az adatok típusától függően ez befolyásolhatja a rendszer biztonságosságát, és lehetővé tehet későbbi támadásokat, amelyek befolyásolhatják a rendszer rendelkezésre állását és megbízhatóságát.

Fogalom	Definíció
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 azon gyenge pontja, melyet kihasználva kár vagy veszteség okozható.
Támadás	A rendszer sérülékenységének kihasználása. Általában a rendszeren kívülről érkezik és ártó szándékú.
Fenyegetések	Olyan körülmények, melyek veszteséget vagy kárt okozhatnak. Ilyen például, ha a rendszer egy sérülékeny pontja támadásnak van kitéve.
Ellenőrzés	A rendszer sérülékenységét csökkentő védelmi mérték. Egy gyenge hozzáférésvezérlő-rendszer sérülékenységét csökkentő ellenőrzés például a biztonsági kódolás.

3.9. ábra. A védettség terminológiája

Mint ahogy az üzembiztonság többi összetevőjének, a védettségnek is megvan a maga terminológiája. Néhány fontosabb kifejezést, amelyeket Pfleeger (Pfleeger, 1997) használ, a 3.9. ábrán foglaltam össze.

Látható az analógia a biztonságosság terminológiájával, így a károkozás analóg a balesettel, a sérülékenység megfelel a kockázatnak. Ezért egy rendszer védettségét hasonló módszerekkel lehet biztosítani:

1. *A sérülékenység elkerülése.* A rendszert úgy tervezik, hogy ne legyen sérülékeny. Például ha a rendszer nem kapcsolódik külső hálózathoz, akkor nincs esély külső támadásra.
2. *Támadás felderítése és semlegesítése.* A rendszert úgy tervezik, hogy a sérülékenységet felismerje és megszüntesse, mielőtt még kár keletkezne. Ilyenek például a víruskeresők, amelyek megkeresik és módosítják a vírusos állományokat, hogy eltávolítsák a vírust.
3. *A lehetséges károk csökkentése.* Sikeres támadás következményeit minimalizálják. Ilyenek például a gyakori rendszermásolatok és a konfigurációkezelési módok, amelyek lehetővé teszik egy megrongálódott szoftver helyreállítását.

A számítógépes rendszerek sebezhetősége persze legtöbbször az emberi tényezőn alapszik, és nem technikai okai vannak. Az emberek könnyen feltörhető jelszavakat használnak, vagy olyan helyre írják azokat, ahol más is megtalálhatja őket. A rendszer-adminisztrátorok is hibázhatnak, amikor beállítják egy rendszer rendelkezésre állását. A védettség növeléséhez a rendszer technikai karakterisztikáján túl át kell azt is gondolnunk, hogyan fogják használni a rendszert. Ezzel bővebben a 30., a biztonsági tervezést tárgyaló fejezetben foglalkozom, amely egy új részben van, ami a kialakuló technológiákról szól.

Kulcsfogalmak

- A kritikus rendszer olyan rendszer, amelynél a hibák jelentős gazdasági vagy fizikai károkat okozhatnak, vagy veszélyt jelenthetnek az emberre. A kritikus rendszerek három fontos csoportja: biztonságosságkritikus rendszerek, küldetéskritikus rendszerek és üzletkritikus rendszerek.
- Egy számítógépes rendszer üzembiztonsága a rendszer azon tulajdonsága, amely megmutatja, hogy a felhasználó milyen mértékben bízik a rendszerben. Az üzembiztonság legfontosabb dimenziói a rendelkezésre állás, a megbízhatóság, a biztonságosság és a védettség.
- Egy rendszer rendelkezésre állása a rendszer azon képessége, hogy nyújtson szolgáltatásokat, amikor arra igény érkezik. A megbízhatóság pedig a rendszer azon képessége, hogy olyan szolgáltatást biztosítson, amilyet elvárnak tőle.
- A megbízhatóság és a rendelkezésre állás (elérhetőség) az üzembiztonság legfontosabb dimenziói. Ha egy rendszer megbízhatatlan, akkor nehéz biztosítani a rendszer biztonságosságát és védettségét, mivel a rendszerhibák ezeket befolyásolhatják.
- A megbízhatóság összefügg a rendszer hibás működésének a valószínűségével. Egy programnak lehetnek olyan meghibásodásai, amelyek mellett a felhasználó még tekintheti a rendszert megbízhatónak. Lehet, hogy a felhasználó soha nem használja a rendszer azon részeit, amelyeknél a meghibásodás történt.
- Egy rendszer biztonságossága a rendszer azon képessége, hogy katasztrofális hiba nélkül működjön. A biztonságosság létfontosságú jellemzője az úgynevezett biztonságosságkritikus rendszereknek.
- A védettség minden kritikus rendszernél fontos. Megfelelő szintű védettség nélkül a rendszer rendelkezésre állása, megbízhatósága és biztonságossága is veszélybe kerül, ha egy külső támadás kárt okoz a rendszerben.
- Hogy növeljük az üzembiztonságot, a szociotechnikai megközelítést már a tervezésbe is be kell venni, és figyelembe kell venni az emberi tényezőt is a hardver- és a szoftvertényezők mellett.

További irodalom

„The evolution of information assurance". Ez egy nagyszerű cikk a kritikus céges információk védelmének szükségességéről. (Cummings, R. *IEEE Computer,* 35 (12), December 2002.)

Practical Design of Safety-critical Computer Systems. Ez egy általános áttekintése a biztonságkritikus rendszerek tervezésének, melyeket nem csak a szoftverek tekintetében kell figyelembe venni. (Dunn, W. R., 2002, Reliability Press.)

Secrets and Lies: Digital Security in a Networked World. Egy kiváló, jól olvasható könyv a számítógépes biztonság szociotechnikai megközelítéséről. (Schneier, B., 2000, John Wiley & Sons.)

„Survivability: Protecting your critical systems". Bevezetés a túlélés témájáról és annak fontosságáról. (Ellison, R. *és mások, IEEE Internet Computing,* Nov./Dec. 1999.)

Computer-related Risks. Az Internet Risks Forumról összeválogatott gyűjtemény automatizált rendszerekben bekövetkezett eseteket tartalmaz. Bemutatja, mi minden romolhat el valójában egy biztonságosnak mondott rendszerben. (Neumann, P. G., 1995, Addison-Wesley.)

Feladatok

3.1. Mi a kritikus rendszerek három fő típusa? Mi a különbség köztük?

3.2. Miért fontos az üzembiztonság a kritikus rendszereknél? Soroljon fel hat indokot.

3.3. Melyek egy rendszer üzembiztonságának legfontosabb dimenziói?

3.4. Miért nő az üzembiztonság megteremtésének költsége exponenciálisan?

3.5. Adja meg, mely üzembiztonsági attribútumok a legkritikusabbak az alábbi rendszerekben. A válaszát magyarázza is meg.
- internetszolgáltató, több ezer ügyféllel;
- számítógép-vezérlésű szike az orvosi mikroműtétek esetében;
- egy útvonalvezérlő rendszer a műholdas járműkövetés esetében;
- egy internetalapú személyi pénzügyi menedzsment rendszer.

3.6. Nevezzen meg hat olyan terméket, amely most vagy a jövőben majd tartalmazhat biztonságosságkritikus szoftverrendszereket.

3.7. A megbízhatóság és a védelem összekapcsolódnak, de mégis különböznek. Magyarázza el a legfontosabb különbségeket, és mutassa meg, miért nem lehet megbízható egy nem biztonságos rendszer, és fordítva.

3.8. Nevezzen meg egy kockázati tényezőt egy olyan orvosi rendszernél, amelyet arra terveztek, hogy a daganatok kezelésére szolgáló sugárzást adagolja. Mondjon egy olyan szoftvertulajdonságot, amely biztosítja, hogy a megnevezett kockázati tényező nem eredményez balesetet.

3.9. Miért van szoros összefüggés egy rendszer rendelkezésre állása és biztonságossága között?

3.10. Írja le, hogy biztonságosság tekintetében mi a különbség a támadás és a fenyegetés között.

3.11. Tisztességes az a tervező, aki ismert hibákkal rendelkező szoftverrendszert ad el a vásárlónak? Számít az, ha a vásárlót előre tájékoztatják e hibák létezéséről? Ésszerű lenne, ha ilyen esetekben a rendszer megbízhatóságát is jellemeznék?

3.12. Ön számítógépes biztonsági szakértő. Arra kéri egy jogvédő szervezet, hogy jogosulatlanul törjön be egy amerikai cég számítógépes rendszerébe, hogy el tudják dönteni, a cég hozzájárul-e direkt módon a politikai foglyok kínzásához. Vitassa meg ennek etikai vonatkozásait, és adjon választ, ön hogyan reagálna egy ilyen felkérésre.