A linuxos biztonság jövője

Az, hogy a Linux más rendszereknél biztonságosabb is lehet, nem jelenti azt, hogy a saját Linuxunk is az. Vajon a fejlesztők és a terjesztések készítői hogyan segíthetik a jövőben a rendszergazdákat?

Ki gondolta volna, hogy már öt éve írok rendszeresen biztonsági témájú cikkeket? És micsoda eseménydús öt év volt ez! Láthattuk, hogy a legnagyobb korábbi vetélytársai felkarolják a ***Linuxot***, amely fokozatosan utat talált az asztali gépek felé.
A ***Linux*** biztonsága terén szintén komoly fejlődés tanúi lehettünk. A ***Linux*** tűzfalszolgáltatásai olyan magas szintre fejlődtek, hogy mára számos beágyazott tűzfalkészülék alapjául szolgálnak, a biztonsággal távolabbi kapcsolatban álló eszközök megszámlálhatatlan sokaságát nem is említve. A ***Linux*** elképesztő mennyiségű biztonsági eszköz támogatására képes, és a biztonsági auditorok és tanácsadók kedvenc eszközévé vált. Külön büszkeségre ad okot, hogy a ***Linux*** adta több kivételesen biztonságos, ***szerep alapú hozzáférés-vezérlést (ultra-secure role-based access control, RBAC)*** alkalmazó operációs rendszer alapját, mint például az ***NSA SELinuxa***.
De vajon milyen lesz a linuxos biztonság jövője? A jelen és a múlt linuxos biztonsági kérdéseiről sokat témáztunk, de a jövőről nem sok szó esett, leszámítva a ***Richard Thieme***-mel készített interjút. Ebben a hónapban szeretnék egy kicsit elrévedezni, és kifejteni, hogy véleményem szerint merre tart a linuxos biztonság, illetve merre ***kellene*** tartania.

Mi a baj a jelenlegi állapottal?

Az utóbbi időben egyre többen ismerik fel, hogy egy átlagos linuxos rendszer nem sokkal biztonságosabb egy átlagos ***Microsoft Windows*** rendszernél. Mielőtt méltatlankodó levelek árasztanák el a postaládámat, hadd fejtsem ki, mire gondolok. Először is, személy szerint úgy vélem, hogy a ***Linuxot*** biztonságosabbá lehet tenni, mint a ***Windowst***, és ezt a véleményemet e helyütt már többször megfogalmaztam az elmúlt évek során. A felhasználók egyszerűen nagyobb hatalmat kapnak linuxos rendszerük működésének ellenőrzéséhez, mint egy azzal egyenrangú windowsos rendszer esetében kapnának.
A baj az, hogy a ***Linux*** felhasználók, hasonlóan a ***Windows*** használókhoz, energiájuk egyre nagyobb hányadát arra fordítják, hogy gépük képes legyen érdemi feladatait ellátni, és túlságosan nagy bizalommal viseltetnek rendszerük beépített, alapértelmezett biztonsági beállításaiba. Az is vitathatatlan, hogy amikor egy-egy programhiba napfényre kerül (márpedig az ilyen esetek elkerülhetetlenek), akkor hatása jóval kiterjedtebb, mint amekkora a megfelelő óvintézkedések megtételével lehetett volna.
Ha például ***BIND v9*** névszolgáltatást akarok futtatni, belekerül némi munkámba és kutakodásomba, mire működni kezd. A ***BIND chroot jailben*** való futtatása és a kiszolgáló fájlrendszerének a ***named*** folyamat számára elérhető részének korlátozása ennél is több munkát igényel. Éppen ezért a ***BIND*** használóinak jelentős része ***nem chroot jailben*** futtatja a ***BIND***-et. Ha ismertté válik a ***BIND*** valamilyen sebezhetősége, akkor a ***BIND***-ot használók többsége jóval több kellemetlenséggel kénytelen szembesülni, mintha megküzdött volna a ***chrootos*** bebörtönzéssel. Lehet, hogy végül semmivel sem járnak jobban, mintha egy microsoftos, a ***BIND***-hez képest akár szerényebb tudású névkiszolgálót futtattak volna.

Röviden szólva, azt akarom kifejezni, hogy a *Linux* biztonsági szolgáltatásainak jelentős részét egyszerűen nem veszik igénybe a felhasználók. A végeredmény – legalábbis profi behatoláspróbákat végző barátaim szerint – az, hogy egy átlagos *Red Hat Enterprise* rendszerbe nem sokkal nehezebb betörni, mint egy átlagos *Windows 2003 Server* rendszerbe.

Szerencsétlen végkifejlet, és talán egy kicsit meglepő is. Kódbázisának teljes nyíltsága ellenére a *Linux* továbbra is ugyanolyan programhibákat tartalmaz, mint a *Windows*, nagyságrendileg hasonló mennyiségben, és ezek felbukkanási gyakorisága is közel áll egymáshoz. Jobban belegondolva, miért is lenne ez másként?

A *Windowshoz* hasonlóan a *Linux* is egy elképesztően összetett, több száz ember munkája nyomán létrejött kódhalmazt jelent. Minél több a kód, annál több a hiba is, nemde?

Nemrég készített velem egy interjút a *SearchSecurity.com*, a téma egy a *Microsoft* támogatásával a *Security Innovation, Inc* által készített tanulmány volt. A tanulmány végső következtetése az volt, hogy a *Windows* biztonságosabb a *Linuxnál*. Véleményüket főként a biztonsági hibák megjelenésének gyakoriságára és a foltok megjelenéséig átlagosan eltelt idő hosszára alapozták.

Azt gondolom, korrekt kritikát adtam a tanulmányról, amely a biztonságnak kizárólag ezeket a könnyen számszerűsíthető értékeit vette figyelembe, a *Linux* más a biztonság terén jelentkező előnyeiről, mint a testreszabhatóság és a bővebb szoftverválaszték, egyszerűen elfeledkezett. Fogalmazhatnék úgy is, a tanulmány inkább az alapértelmezett telepítésekre volt érvényes, és nem vette figyelembe, hogy az egyes operációs rendszerek mennyi lehetőséget adnak használóiknak a biztonság fokozására.

Minél többet gondolkodom ezen, annál tisztábban fogalmazódik meg bennem az a gondolat, hogy az adott rendszerben rejlő biztonsági lehetőségek mit sem érnek, amíg a rendszert futtató gépek nem használják ki őket. Itt nem kizárólag a végfelhasználókra gondolok, nem a rendszergazdákat szeretném bírálni. Később még kifejtem részletesebben is, de úgy vélem, a *Linux* fejlesztőinek és terjesztőinek továbbra mindent meg kell tenniük annak érdekében, hogy a biztonsági szolgáltatások mindent átszőjenek, átlátszók, könnyen beállíthatók és egyszerűen használhatók legyenek.

Egyébként, ha már összehasonlítom a *Linuxot* és a *Windowst*, meg kell említenem, hogy a *Windows* is túlságosan sok a használói által parlagon hagyott biztonsági szolgáltatással rendelkezik.

Rendben, alapállapotában a *Linux* és a *Windows* egyaránt rosszabb biztonságot ad, mint amire képes volna, és mindkettő megnyerhetetlen versenyfutásba kényszerít bennünket a programhibák és a biztonsági foltok között. Mivel kell még megküzdenünk?

Nos, mindkét operációs rendszer a meglehetősen primitív önkényes (vagy önkéntes) hozzáférés-vezérlési modellt használja, ahol a biztonsági beállítások egész osztályainak használata, illetve ezek viselkedése kiegészítő, elhagyható jellegű. Ebben a modellben van egy szuperfelhasználói fiók – *Linux* alatt a root, *Windows* alatt a rendszergazda –, mely istenségszerű hatalommal rendelkezik a teljes rendszer felett, ide értve más felhasználók fájljait is. Mindkét operációs rendszerben csoporttagságokkal lehet különféle hozzáférési szinteket létrehozni, más szóval a szuperfelhasználói jogokat átadni.

A gyakorlatban a legtöbb rendszerre szuperfelhasználóként kell bejelentkeznünk, illetve ideiglenesen az ő szerepét kell felvennünk, ha valamilyen fontosabb műveletet akarunk elvégezni.

Ha tehát egy linuxos vagy windowsos rendszer felett korlátlan hatalmat akarunk szerezni, akkor csak egy a szuperfelhasználó jogaival futó folyamat biztonságát kell megtörnünk. Hogyan? A fontosabb démonokat jogosultságok nélküli felhasználók nevében futtatjuk, így hiába találnak hibákat ezekben a démonokban, a rendszer biztonsága nem sérülhet. Vagy mégis? Nem, közvetlenül valóban nem, de az egyéb szoftverekben található hibák miatt egy alacsonyabb jogosultságokkal futó program is szert tehet szuperfelhasználói jogokra. Tegyük fel például, hogy van egy *Apache* alapú webkiszolgálónk, és egy napon egy támadónak sikerül kihasználnia egy még befoltozatlan, az *Apache*-ban található puffertúlcsordulásos hibát, amivel sikerül héjhozzáférést szereznie a kiszolgálón. Ezen a ponton a támadó a webfelhasználó jogaival élhet, ugyanis az Apache ennek a felhasználónak a jogosultságaival fut. Tegyük fel, hogy ugyanezen a rendszeren a rendszermagnak is van egy befoltozatlan hibája, ami lehetővé teszi a helyi jogosultságok megemelését.

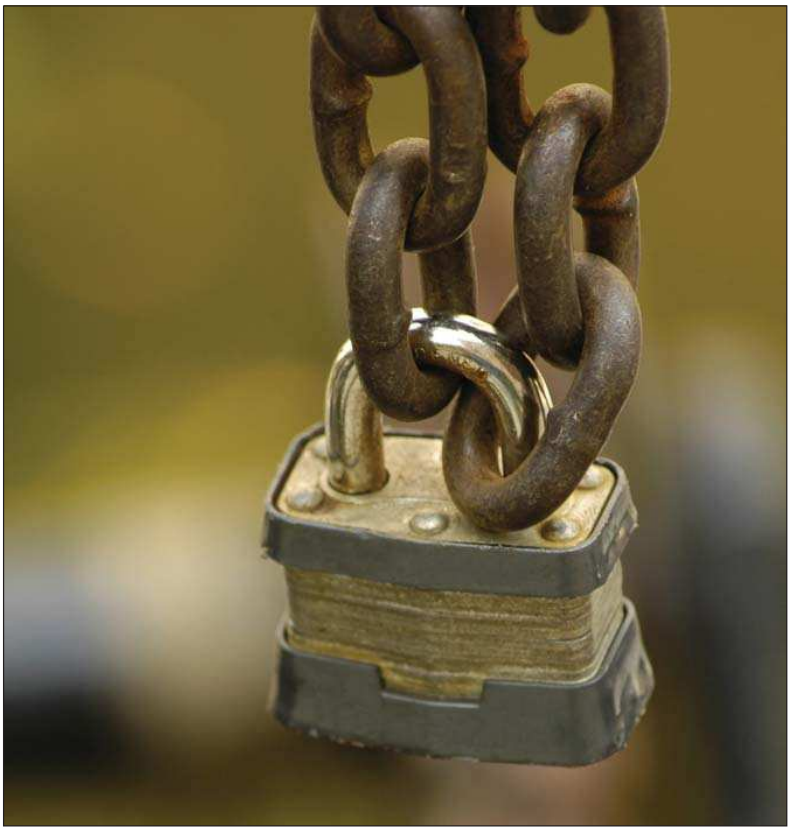
Lehet, hogy a rendszergazda tud is a hibáról, de nem foglalkozott a javításával, hiszen csak helyből használható ki, és rajta kívül senki nem rendelkezik héjhozzáféréssel a rendszerhez – és persze ki akarná újraindítani a gépet csak a rendszermag megfoltozása miatt? Csakhogy mostanra a távoli támadó helyi héjhozzáférést szerzett, sikeresen kihasználja a rendszermag sebezhetőségét, és máris rootként garázdálkodhat a gépen! Nem is gondolnánk, milyen gyakori ez a forgatókönyv, és kiválóan szemlélteti, hogy a hibák önmagukban is sok fejtörést okoznak, de a root mindenhatóságára épülő biztonsági modellel kombinálva a bajok megsokszorozódnak.

Nagyjából ez jellemzi a linuxos biztonság jelenlegi állapotát.

A *Linux* biztonságossá tételéhez komoly erőfeszítéseket kell tennünk: ki kell használnunk az alapesetben legtöbbször nem engedélyezett, időnként meglehetősen bonyolult biztonsági szolgáltatásokban rejlő lehetőségeket, és minden rendelkezésünkre álló biztonsági foltot telepítenünk kell – mindezt a Linux egyszerű biztonsági modellje által biztosított keretek között. De ne aggódjunk, jó társaságba keveredtünk: napjaink operációs rendszerei ugyanezekkel a korlátokkal és kihívásokkal küzdenek.

Kötelező hozzáférés-vezérlés

Már utaltam rá, hogy a *Linux*, az általános *UNIX* és a *Windows* rendszerekben látható hozzáférés-vezérlés és a fájleléréseket szabályozó megoldás önkényes, ami biztonsági modellként gyengének számít. Mi a helyzet az *SELinuxszal*? Az *RBAC*-k és a *típus-*



szabályok (type enforcement, TE) vajon jó példát adnak a kötelező hozzáférés-vezérlésre? Igen. Tartok azonban attól, hogy a linuxos biztonságnak nem ez a jövője, pontosan azért, amiért az *SELinux* sem tudott számottevő szerepet szerezni e téren.

Az *RBAC*-k gondosan körülírt szerepek alapján korlátozzák a felhasználók viselkedését és a rendszer erőforrásaihoz való hozzáférését. Ezek a szerepek hasonlóak a hagyományos *UNIX* csoportokhoz, bár jóval messzebbre hatnak. A típusszabályok hasonlóan, előre megadott műveletartományok alapján korlátozzák a folyamatok által végrehajtható műveletek körét. Az *RBAC*-k és a *TE*-k használatával végül elválasztott silók (én legalábbis így hívom őket) jönnek létre, ezekben tevékenykednek a felhasználók és a folyamatok, és a silók között csak korlátozott párbeszédet engedünk meg.

Ez egy elegáns és hatékony biztonsági modell, csakhogy az emberek túlnyomó része számára az *RBAC*-k, a *TE*-k és az egyéb hozzáférés-vezérlési módszerek túlságosan bonyolultak, és túlságosan sok felügyeleti többletmunkát okoznak. Sokak számára ezek a tényezők az *SELinuxot* és a hozzá hasonló operációs rendszereket elismert, de elkerült megoldásokká teszik – olyan operációs rendszerek ezek, melyek bizonyos csoportok igényeinek megfelelnek, széles körben azonban nem tudnak elterjedni. Bár magam is csodálom az *SELinux* biztonsági architektúráját, sőt, rajongok az *RBAC*-k használatának ötletéért, nem hiszek abban, hogy az általuk megvalósított kötelező hozzáférés-vezérlésnek esélye volna a linuxos biztonság megújítására.

Hiperfelhasználók és virtuális gépek

Ha az *RBAC*-k és a *TE*-k használata az operációs rendszer szintjén túlságosan kényelmetlennek bizonyul a betörések hatókörének korlátozására, a hiperfelhasználók (hipervizorok) és a *virtuális gépek (virtual machine, VM)* egy magasabb szinten talán megoldást kínálnak. A virtuális gépeket már két területről is ismerhetjük: egyrészt a virtuális futtatási környezetek révén, ilyet használnak például a Java programok, másrészt a virtuális gépek által, ilyen a *VMware*, a *plex86* és a *VirtualPC*; ezek virtuális hardverkörnyezetben teljes operációs rendszerek futtatására képesek.

A *Java* virtuális gépet figyelemre méltó biztonság szolgáltatásokkal ruházták fel, ezek közül a legfontosabb talán a *Java* homokozó. Általában véve a Java biztonsága abból fakad, hogy a Java kisalkalmazások a nyers, valós rendszererőforrásoktól elkülönítve futnak, minden művelet a Java virtuális gép közvetítői segítségével történik. Ez is kiváló biztonsági modell, és a programozók és a végfelhasználók számára egyaránt könnyen használható. A *Java* – több okból – napjainkra széles körben elterjedt.

A virtuális számítógépek eggyel tovább viszik ezt az elgondolást, és nemcsak programok, de teljes operációs rendszerek között is képesek közvetíteni. A biztonsági alrendszer esetükben még kevésbé kiforrott, mint a Java virtuális gép esetében. A biztonság kezelését túlnyomó részt a virtuális környezetben futó vendég operációs rendszerre bízzák. Egy *VMware*-ben futó *SUSE Linux* tehát se többé, se kevésbé nem biztonságos, mint egy a saját hardverén futó.

A hiperfelhasználós technológia az azonos hardveren futó virtuális gépek elkülönítésének és a párbeszédek korlátozásának gondjára kínál megoldást, illetve általa megelőzhető, hogy az egyik virtuális gép biztonságának sérülése a többi gépet is érintse. Az *IBM sHype* névvel már készített egy hiperfelhasználós biztonsági rendszert. Létezik egy nyílt forrású hiperfelhasználókra és virtuális gépekre alapozó fejlesztés is, ez *Xen* néven fut.

Bár a hiperfelhasználók alkalmazásának fő oka az, hogy az azonos fizikai gépen futó virtuális gépek – például a megosztott hardveres erőforrások kisajátításával – ne zavarhassák meg egymás működését, az önmagában sem rossz gondolat, hogy kellene valamilyen magasabb szintű, a rendszerek felügyeletére képes intelligenciát biztosítani. Ez a hagyományos *behatolásészlelő rendszerek (intrusion detection system, IDS)* működését legalábbis javítaná, de lehetséges, hogy át is venné azok szerepét a rendszer biztonságának megsértésére irányuló kísérletek felismerése és kibontakozásának megakadályozása terén.

A kötelező hozzáférés-vezérlés és

a hiperfelhasználókra és virtuális gépekre alapuló megoldások nem zárják ki egymás alkalmazását. Ugyanakkor az én meglátásom – nem kis részben barátom és biztonsági elemző munkatársam, ***Tony Stieber*** hatására – az, hogy a hiperfelhasználós megoldásnak jóval nagyobb lehetősége lesz a linuxos biztonság jövőjének alakítására, mint a kötelező hozzáférés-vezérlésnek. Természetesen együttes használatuknak sincs akadálya. Képzeljünk el egy nagyméretű, nagy teljesítményű rendszert, mely nagyszámú virtuális gépet futtat, és ezeket egy hiperfelhasználó felügyeli. Az egyik virtuális gépen egy általános célú operációs rendszer – például ***Linux*** – mint webkiszolgáló üzemel. A másik virtuális gép érzékeny adatokat kezelő adatbázisként szolgál, erre valamilyen kötelező hozzáférés-vezérlést alkalmazó operációs rendszer kerülhet, például ***SELinux***. Mindkét virtuális gép részesül a hiperfelhasználós modell által biztosított előnyökből, miközben az ***SELinux*** a maga játékterén további védelmet is nyújt.

Rendellenesség alapú behatolásészlelés és vírusvédelem

A kötelező hozzáférés-vezérlés és a hiperfelhasználós megoldás mellett egy további, már létező, de a jövőben valószínűleg a jelenleginél nagyobb szerepre érdemes technológia a rendellenesség alapú behatolásészlelés.
A rendellenesség alapú ***IDS***-ek ötlete egyszerű: meg kell adni, hogy mi jellemzi a rendszer és a hálózat normál működését, és váratlan vagy rendellenes viselkedés észlelésekor riasztást kell küldeni.
Ha maga az ötlet egyszerű, a megvalósítás pedig már megvan, akkor miért nem használjuk szélesebb körben is? Azért, mert nem olyan egyszerű és kiforrott, mint az aláírás-egyeztetés. Az aláírás alapú ***IDS***-eket már ismerjük: rendelkeznek a támadások aláírásainak adatbázisával, és összevetik ennek tartalmával a hálózati csomagokat, illetve a csomagok sorozatait. Ha egy csomag megegyezik egy az adatbázisban szereplővel, akkor támadás részeként kezeljük, és riasztást bocsátunk ki.
Ennek a megoldásnak az előnye, hogy könnyen használható, és kevés hamis riasztást ad ki. Az aláírás alapú rendszerek akkor mondanak csődöt, ha újfajta támadás éri a rendszert, esetleg a támadás túlságosan bonyolult ahhoz, hogy megfelelő aláírás tartozhasson hozzá az adatbázisban, és ezért nem sikerül felismerni.
A rendellenesség alapú megoldásnál ezzel szemben minden olyan új támadás felismerhető, amely kellően megkülönböztethető a normál működéstől. Ennek ára az, hogy az ***IDS*** rendszergazdájának be kell tanítania a rendszert (ennek során mutatja meg neki, hogy mi számít normál üzemnek), és a betanítást rendszeres időközönként meg is kell ismételni. Amíg a viszonyítási alapot nem sikerül finomhangolni, addig viszonylag gyakori hamis riasztásokra kell készülni.
Még 1999-ben volt szerencsém részt venni ***Marcus Ranum*** egy előadásán, ahol a rendellenességek felismerését a behatolásészlelő rendszerek jövőjeként jellemezte. Mondanom sem kell, ez a jövő még nem érkezett el, bár már vannak ilyen termékeket kínáló gyártók, például a ***Lancope*** és az ***Arbor Networks***. Remélem, előbb-utóbb valaki kitalálja, hogyan lehet ezt a feladatot valamilyen egyszerűbb módon elvégezni, a jelenlegieknél egyszerűbb és könnyebben használható rendszereket alkotva. Úgy látom, ezzel olyasfajta hálózati hiperfelhasználó jönne létre, aki hasonló intelligenciával ruházná fel a hálózatokat – álljanak azok akár virtuális, akár valós gépekből –, mint amilyet a virtuális számítógépeknek is biztosít.
Megjegyzem, a víruskeresők legalább annyira ki tudnák használni a rendellenességek felismerését, mint a behatolásészlelő rendszerek. Mi sem igazolhatja ezt jobban, mint az a tény, hogy azok a szervezetek, amelyek korszerű, de kizárólag vírusaláírásokra támaszkodó víruskereső programokat használnak, rendszeresen áldozatául esnek a nagyobb vírus-, trójai- és féregkitöréseknek. Egyértelmű, hogy a jelenlegi, aláírásokat alkalmazó víruskereső eszközök hatékonysága elégtelen.

Összefoglalás és búcsú

Nagyjából ezek a ***Linux*** biztonságával kapcsolatos gondolataim. Amíg nem sikerül előbbre lépni, használjuk tovább a meglévő, a jelen rovatban is sokat tárgyalt eszközöket: a tűzfalakat, a víruskeresőket, az önműködő foltozó és programfrissítő eszközöket, a virtuális magánhálózatokat és az egyedi biztonsági alkalmazásokat, mint a chroot jailek és a naplók.
Ezzel én búcsút intek, nemcsak erre a hónapra, hanem bizonytalan időre. Legalább egy kis ideig más dolgokra kell fordítanom a figyelmemet, és a rovatban új hangokat is hagynom kell megszólalni. Biztonsági szerkesztőként továbbra is ott leszek a háttérben, és ebben a szerepben a jövőben is segíteni fogom a ***Linux Journalt*** abban, hogy a biztonság témakörében kimagasló színvonalú tartalmat biztosíthasson. Remélem, hogy alkalmanként további írásokat is tudok majd készíteni, a rovat kizárólagos szerzőjeként azonban ez az utolsó cikkem. Köszönöm mindenkinek az elmúlt öt év támogatását, bátorítását és segítségét: írásaimban soha nem véthettem úgy hibát, hogy valaki ki ne javítson – mindig baráti hangnemben.
Nagyszerű öt év volt, hálás vagyok a lap fantasztikus gárdájának és olvasóinak, hogy részem lehetett benne!

Linux Journal 2005. 136. szám

Mick Bauer
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban. Az O'Reilly & Associates nemrég, 2005 januárjában jelentette meg Linux Server Security című könyvének második kiadását. Mick „indusztriális polka" zeneszerző, ám van annyira jó ízlésű, hogy ne nagyon játssza el saját szerzeményeit.

KAPCSOLÓDÓ CÍMEK

sHype ➲
www.research.ibm.com/secure_systems_department/projects/hypervisor

Xen ➲
www.cl.cam.ac.uk/Research/SRG/netos/xen