Paranoid Pingvin: a Linux biztonságának kockázat alapú megközelítése

A kockázat elkerülhetetlen. Legyünk pesszimisták a programhibákkal kapcsolatban, készítsünk terveket a hibák kezelésére, így egész rendszerünket biztonságban tudhatjuk.

Amióta négy évvel ezelőtt elkezdtem írni ezt a rovatot, a ***Linux*** program sérülékenységei és fenyegetései nem sokat változtak. Veremtúlcsordulások, hibás beállítások (fájl jogosultsági kérdéseket is beleértve) nem megfelelő bemenet érvényesítés jelentik továbbra is a ***Linux*** sérülékenységek oroszlánrészét. Ha ugyanolyan típusú sérülékenységek bukkannak fel újra és újra, nem értelmetlen ez az egész foltverseny? Nincs egy szélesebb látókörű ***Linux*** biztonsági elgondolás?

Ebben a hónapban a ***Linux*** biztonságot kockázat alapú nézőpont szerint vizsgáljuk, és bemutatjuk, hogyan használjuk a kockázat alapú megközelítést az ismert ***Linux*** sérülékenységeken túl olyan problémák csillapítására, amelyeket még senki nem fedezett fel vagy adott közre.

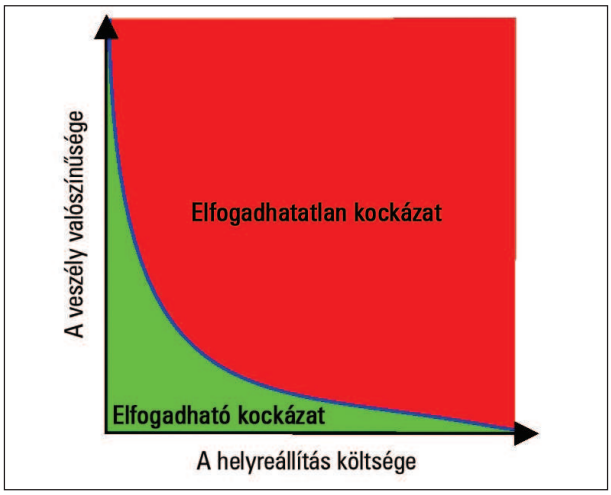
A biztonság kockázat alapú megközelítése

Biztosan vannak akik azt kérdezik magukban, mit értek egyáltalán kockázat alapú megközelítésen? Hát az információs biztonság nem elve a kockázatról szól? Tulajdonképpen igen, azonban ezt a kifejezést valójában a kockázat kezelés alapú megközelítés rövidítéseként használom.

Egy adott információs biztonsági kockázatot csak néhány módon lehet kezelni. Elkerülhetjük, azaz semmi olyat nem teszünk ami kitehetne bennünket a veszélynek. Megszüntethetjük, gyökerénél kezelve (amire sajnos a gyakorlatban ritkán van lehetőségünk). Enyhíthetjük, azaz, valami olyat csinálunk ami csökkenti a kockázat hatását valamilyen módon. Végül még egy lehetőségünk van: elfogadjuk.

Az egyik, szerencsére mostanra elavult elgondolás szerint a biztonság bináris egyenlet: a dolgok vagy biztonságosak vagy ostobák, és ha úgy találjuk, hogy egy tevékenység vagy eszköz nem biztonságos, akkor azt nem szabad csinálni vagy használni. Más szavakkal ez az iskola azt hirdeti, hogy a biztonság elsődleges irányelve az elkerülés.

Mint azt egyre többen felismerik manapság, tökéletes biztonság nem létezik. Nincsenek mágikus programkombinációk, program/rendszer összeállítások vagy hálózatszerkezet ami sebezhetetlenné tenne bennünket a betörésekkel szemben.



1. ábra Kockázati határok

Nincs ilyen összeállítás, legalábbis olyan, amivel dolgozni is lehet. A hálózati számítástechnikában valamilyen szintű kockázat elkerülhetetlen.

A kockázat kezelés alapú megközelítés felismerte, hogy a kockázat elkerülése, a kockázatcsökkentés és elfogadás közötti egyensúlyra kell törekedni, mégpedig a kockázatokat fontossági sorrendbe rendezve valószínűségük és romboló hatásuk alapján. Csökkentés vagy elkerülés szempontjából a legfontosabb kockázati tényezők azok lesznek, amelyek nagyobb valószínűséggel következnek be és helyreállításuk költséges. Az erősen valószínűtlen, vagy olcsón helyreállítható hatású kockázati tényezők viszont gyakran kerülnek az elfogadható kategóriába. Mondanom se kell, amikor a kockázati tényező bekövetkezésének költségéről vagy hatásáról beszélek, nem kizárólag a pénzügyi költségre gondolok. Az időbeli és termelékenységi veszteség valamint a jó hírnevünk éppúgy ide tartozik.

Az *1. ábra* mutatja a kockázat valószínűsége, költsége és elfogadhatósága közötti általános viszonyt. Az elfogadható és elfogadhatatlan kockázati területeket kijelölő görbe pontos alakja szervezetről szervezetre változhat.

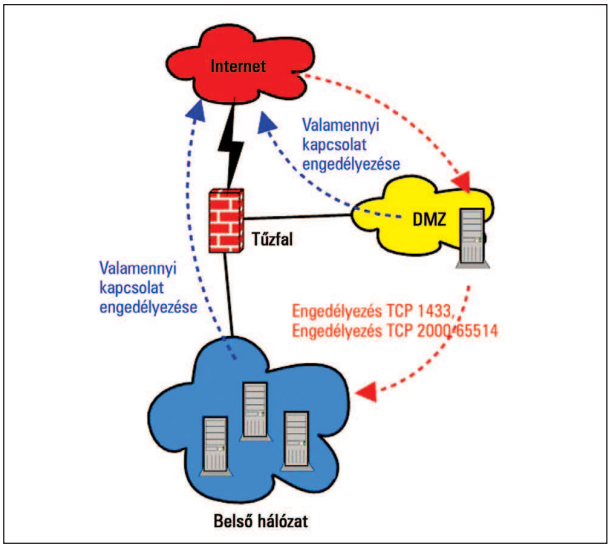
Egy pénzügyi cég például sokkal nagyobb vörös zónával rendelkezik mint egy egyetemi hálózat. A kockázat biztonság alapú megközelítése végül is annak felismerését jelenti, hogy nem minden kockázat azonos, ezért harcteret kell választatnunk. Azonban ahhoz, hogy ezt hatékonyan tehessük meg, becsületesen és találékonyan kell beazonosítanunk és felbecsülnünk az adott vállalkozás kockázati tényezőit. Egy létező hiba figyelmen kívül hagyása sokkal veszélyesebb mint tudomásul venni és elfogadni a hibát és elkészíteni a visszaállítási terveket arra az esetre, ha a legrosszabb bekövetkezne.
Ezzel el is érkeztünk a kockázat alapú megközelítés egy másik fontos tételéhez: a kockázat elfogadása még nem jelenti azt, hogy elégedettek lehetünk. Bármilyen kockázatot, amit nem tudunk elkerülni, vagy legalább enyhíteni, figyelembe kell vennünk az folyamatosság fenntartási és helyreállítási terveinkben. Csak nagyon kevés információs kockázat nem enyhíthető valamilyen mértékben; vannak veszélyforrások amelyek nem szüntethetőek meg, de a legtöbb felhígítható vagy lefékezhető.

Sérülékenységek és veszélyek

Rendben, tehát a ***Linux*** biztonsága legjobban a kockázat alapú szemléletmóddal kezelhető. Hogyan is nézne ez ki? Az első lépés ***Linux*** rendszerünk ismert és potenciális sérülékenységeinek számbavétele. A legtöbb ***Linux*** alkalmazás és rendszer-sérülékenység az alábbi kategóriák valamelyikébe esik:

- veremtúlcsordulás sérülékenység (nem megfelelő határ ellenőrzés).
- Elégtelen bemenet hitelesítés.
- Helytelen fájl jogosultságok.
- Nem megfelelő rendszer jogosultságok (felesleges root használat).
- Hanyag beállítások.
- Ideiglenes állományok nem biztonságos használata.
- Megjósolható vagy ismert alapértelmezett jelszavak alkalmazása.
- Adminisztrációs hátsó kapuk (tesztelési és hibakeresési felhasználónevek).

Valószínűleg a lista legelső sérülékenysége, a veremtúlcsordulás a legriasztóbb. A veremtúlcsordulások általában közvetlenül távoli root betöréshez vezetnek. A veremtúlcsordulás feltételeihez hasonlóan a fenti sérülékenységek egy része közvetlenül programozási hibák következménye. Ilyen például az ideiglenes állományok helytelen használata és adminisztrációs hátsó ajtók. Mások inkább felhasználótól függőek, ilyen a kiszámítható jelszó és a hanyag beállítások. Egyetlen sérülékenység sem jelent veszélyt, ha senki sem próbálja meg kihasználni azt. Más szavakkal, a fenyegetés két alkotóeleme a sérülékenység és a támadó.
A második lépés végiggondolni, milyen módszerekkel lehet kihasználni ezeket a sérülékenységeket. Bár a ***Linux*** biztonsági kockázatai nem sokat változtak az elmúlt évek során, az őket kihasználni szándékozó szereplők mások. Hatékonyabbá és butábbá váltak. Az a rémisztő igazság, hogy a törőkódok és szkriptek könnyű elérhetősége miatt a képzetlen támadók is egyre könnyebben tudnak kifinomult támadásokat levezényelni.



2. ábra Egyszerű tűzfal összeállítás

Korábban például egy veremtúlcsordulás támadás levezényléséhez komoly programozási tudás kellett, ki kellett találni a memóriában hová kerül a túlírt adat, a támadónak létre kellett hozni vagy be kellett szereznie a célrendszerre (például ***i386*** vagy ***SPARC***) jellemző assemblyben írt törőkódot vagy héjkódot. A héjkód az a kód ami majd túlcsordul és végrehajtódik, létrehozva egy héjat a célrendszeren, ideális esetben root jogosultságokkal.
A régi időkben az eltolások meghatározása és működő héjkód megírásnak nehézségei a veremtúlcsordulás alapon támadók táborát erősen leszűkítette. Manapság azonban, ha ki szeretnénk használni egy jól ismert veremtúlcsordulás sérülékenységet, nem kell mást tennünk mint kiadni egy jól formázott ***Google*** keresést, és máris megszereztük a törőkódot a hozzá tartozó, különféle rendszerekhez készült héjkóddokkal.
A törőkódokat író és az interneten közreadó emberek elég nagy problémát jelentenek. De nem ők az egyedüli szereplői a nagy egyenletnek; aki örömét leli benne, hogy script kiddie-ket fegyverez fel, már nem sok választja el attól, hogy féregbe vagy vírusba csomagolva automatizálja törőkódját. A vírusok természetesen magukat nem tudják terjeszteni; mindig valami másba ágyazva találhatóak, például e-mail csatolmányban vagy végrehajtható állományokban. A férgek viszont magukat terjesztik, így lényegesen félelmetesebbek, hiszen tulajdonképpen szárnyas vírusok. Ha féregtámadás idején a naplóbejegyzéseket nézzük, nagyon nehéz megkülönböztetni őket egy ember vezérelte támadástól. A féreg tulajdonképpen egy támadó robot.
Így a támadó lehet ember és lehet program. A jó hír, hogy mivel a ugyanolyan típusú sérülékenységet támadnak, a védekezés is hasonló. A rossz hír, hogy a támadó szkriptek férgek és vírusok exponenciális mértékben csökkentik a sérülékenység felfedezése és közzététele, valamint a rendszerünk valószínűsíthető megtámadása közötti időt.

Első védelmi megoldás: Tűzfal rendszabályok

Kezdjük el ezeket a fenyegetéseket a megfelelő védelemhez rendelni. Itt kezd majd a kockázat alapú megközelítés igazán fontossá válni.

Amennyiben mindent vagy semmit alapon tekintünk a biztonsági kérdésekre, a védelem egyszerű. A programokat nem képességeik, támogatottságuk és biztonságosságuk összesítése alapján, hanem tisztán a biztonságosságuk szerint válogatjuk. Minthogy a fő programfeltételünk a biztonság egyetlen dologra kell figyelnünk, nevezetesen a foltozásra, és minden rendben lesz.

Elképzelhető, hogy úgy állítjuk be a tűzfalunkat, hogy semmilyen kívülről jövő kapcsolatban se bízzon, de fogadjon el minden belülről érkezőt, hiszen, természetesen minden külsős gyanús és minden belső megbízható. Ami azt illeti, a programfoltok és tűzfalszabályok olyannyira fontosak ebben az elképzelésben, hogy tulajdonképpen semmi más nem számít.

A program foltok és tűzfal szabályok tényleg nagyon fontosak. Azonban a programfoltoktól való függésünk foka és tűzfalhasználatunk mikéntje egy kicsit eltérő lehet, ha vesszük magunknak a fáradtságot és meggondoljuk milyen valós fenyegetéstől óvnak bennünket. Képzeljük el a *2. ábrán* felvázolt helyzetet. A tűzfal a ***DMZ*** hálózatot védi a külső világtól, az pedig a belső hálózatot védi a külső világtól és a ***DMZ***-től.

A 2. ábrán pontozott vonallal jelölt tűzfal szabályok a következőképpen nézhetnek ki:

1. Az összes belső gép elérheti az Internetet bármelyik kapu/protokoll párossal.
2. Az ***DMZ*** gépnek engedélyezzük az Internet elérést bármely kapun/protokollon.
3. Az összes internet gép elérheti a ***DMZ***-t a 80-as TCP kapunk keresztül (HTTP).
4. A ***DMZ*** webkiszolgálója elérheti a belső gépeket a 1433 és 2000-65514 közötti TCP kapukon.

Elsőre egész jónak tűnik. A belső felhasználók mindenféle dolgokat csinálnak az Interneten, így ezt korlátozni zűrös lenne. A naplók miatt a ***DMZ DNS*** lekérdezéseket végez, szóval miért ne adnánk Internet elérést neki is? Van ezen felül egy háttéralkalmazás amit a ***DMZ***-be tett webkiszolgálónak el kell tudni érnie a belső hálózaton és ahol adatbázis lekérdezéseket adhatunk ki a 1433-as ***TCP*** kapun valamint egy véletlenszerűen választott magas kapun ami egy olyan véges tartományba esik, amit senkinek nem sikerült dokumentálnia. Ezért aztán a legegyszerűbb ha megnyitjuk az összes 1999-nél magasabb ***TCP*** kaput.

De nézzünk három kézenfekvő kockázati tényezőt:

1. Az internet alapú támadó feltöri a webkiszolgálót és más gépeket támad vele az interneten.
2. Egyik belső gépet féreg fertőzi meg valamilyen ***RPC*** sérülékenységen keresztül, és a fertőzött rendszer hatalmas mezőket kezd pásztázni az interneten más sérülékeny rendszereket kutatva.
3. Egy féreg megfertőzi a belső rendszert és hátsó kaput nyit a 6666-os ***TCP*** kapun. A támadó feltöri a webkiszolgálót, letapogatja a tűzfalat, felderíti a jól ismert féreg ajtót és belép a belső rendszerbe.

Az első kockázati kérdésben egyértelműen jogi problémáknak tesszük ki magunkat. Ha a webkiszolgálót feltörik és

a tűzfalunk nem akadályozza meg rajta keresztül a külső világ elérését, felelősnek érezhetjük magunkat, hiszen a mi rendszerünkről támadnak más rendszereket az interneten. A webkiszolgáló kimenő elérését a legszükségesebb szolgáltatásokra korlátozva csökkenthetjük a kockázatot. A gyakorlatban egy ***DMZ***-be helyezett webkiszolgálónak nagyon kevés adatfolyamra van szüksége a külső világ felé, ha szüksége van egyáltalán ilyesmire. Az a feladata, hogy az internetről érkező ***HTTP*** lekérdezésekre válaszoljon, nem pedig, hogy maga kezdeményezzen kapcsolatokat.

A második forgatókönyv szerint hasonló problémáknak tesszük ki magunkat, de itt a jogi következményeknél komolyabb gondot okoz a dolog hálózati teljesítmény oldala (a pásztázás forgalma eldugíthatja az Internetes kapcsolatunkat). Akárcsak az előbb, egy szigorúbb tűzfal rendszabály a kimenő forgalomra egyértelműen csökkenti a kockázatot.

A harmadik forgatókönyv egy kicsit szokatlanabb mint a többiek, végül is mennyi az esélye, hogy egy féreg megfertőzze a belső rendszert és egyúttal a ***DMZ***-ben található webkiszolgálónkat is feltörjék? Nos, tulajdonképpen a kettőnek nem kell egyszerre bekövetkeznie. Ha a féreg csendben marad miután elkészíti a hátsó ajtót a 6666-os ***TCP*** kapun, jó ideig nem fogjuk felfedezni. Tehát a webkiszolgáló feltörése nem kell, hogy aznap vagy akár abban a hónapban történjen, hiszen a féreg munkáját a fertőzött rendszerben nem hatástalanítottuk elég hamar. Akárcsak az előző két forgatókönyvben, egy szigorúbb rendszabály csökkentheti a kockázatot és minimalizálja annak az esélyét, hogy a féregfertőzést külsősök ki tudják használni.

Azon felül, hogy szigorúbb szabályokkal veszélyességük csökkenthető, e három veszélyforrásnak van még egy közös jellemzője. Csökkentésükhöz nem kell pontosan megjósolni őket. Elég ha csak arra gondolunk „mi van ha a tűzfal szabályaim ellenére valamilyen féreg vagy vírus bejön, és váratlan típusú kimenő forgalmat kísérelnek meg?" Nem tudom eléggé hangsúlyozni, nagyon fontos, hogy a tűzfal szabályok készítésekor ne csak a támadások megelőzésére koncentráljunk. Legalább ilyen fontos, hogy végiggondoljuk, mi történik ha a védelmünk csődöt mond. Az információs biztonság terén a pesszimizmus épít és nem rombol.

Remélem mostanra világossá vált, hogy véleményem szerint nem a tűzfal szabályok adnak választ az összes Linux biztonsági kérdésünkre. Összefoglalva, hatékony tűzfal szabályokat akkor tudunk készíteni, ha nem csak az ismert fenyegetéseket, hanem a lehetséges fenyegetéseket is figyelembe vesszük.

Második védelmi megoldás: az alkalmazások biztonsága

Nos, ha a tűzfal nem csodaszer, akkor mit tehetünk? A rovatban korábban a hanyag beállításokat az egyik legnagyobb sérülékenységi forrásnak neveztem; ezt megfordítva, a figyelmes beállítás az egyik legjobb védelem.

Tegyük fel, a ***DMZ***-ben van egy ***SMTP*** átjáró ami az Internet és a belső hálózat közötti levelezést bonyolítja. Tegyük fel továbbá, hogy a szervezetünk műszaki személyzetének nagy tapasztalata van a ***Sendmail*** terén, viszont se idejük, se hajlandóságuk megtanulni a ***Postfix*** használatát, amit egyébként mint megrögzött biztonságmániás, sokkal biztonságosabbnak tartok. A vezetés úgy dönt az átjáró ***Sendmailt*** fog futtatni. Minden elveszett? Nem szükségszerűen. Először is, mint azt a rovatban korábban is említettem, a ***Sendmail*** biztonsági mutatói tulajdonképpen egészen jók lettek az utóbbi években. Azonban, ha mindez meg is változik egyetlen éjszaka alatt, és a rossz fiúk három újabb veremtúlcsordulás hibát találnak a ***Sendmailben*** amit nem adnak közre azonnal, a ***Sendmail*** fejlesztők egészséges pesszimizmusának hála, ***Sendmail*** átjárónk még nem bukott el feltétlenül.

A ***Sendmail***-nek van néhány fontos biztonsági képessége, amiből kettő különösen hasznos lehet veremtúlcsordulások esetén. A ***Sendmail*** futhat ***chroot*** ketrecben, ami korlátozza a látható fájlrendszer területet, valamint előjogok nélküli felhasználó és csoportjogosultságokkal is indítható, így minimalizálhatjuk annak az esélyét, hogy a ***Sendmail*** sérülékenység közvetlenül root eléréshez vezessen. Minthogy a ***Sendmail*** a kivételezett 25-ös ***TCP*** kapura hallgat, legalább az idő egy részében rootként kell futnia, ezért a ***Sendmail*** gyakorlatilag időnként lefokozza saját magát különleges jogosultság nélküli felhasználó/csoport szintre. Ez tehát az enyhítés nem teljes, csak részleges.

Manapság a legtöbb jól tervezett hálózati alkalmazás chroot-olható és futtatható előjogok nélküli felhasználó/csoport jogosultságokkal. Ahogy a jó tűzfal szabály egyszerre célozza a megelőzést és a behatárolást, a helyes alkalmazás beállításkor is számításba kell venni, hogy az alkalmazással visszaélhetnek vagy eltéríthetik. Ezért aztán az alkalmazás biztosíthatóságát nem csak az méri hány ***CERT*** tanácsadóban szerepel. Az alkalmazásnak rendelkeznie kell beépített mérséklési lehetőségekkel is.

Összefoglalás

A biztonság kockázat alapú megközelítése két fontos előnyt jelent számunkra. Először is, ahelyett, hogy egyfolytában nemet mondunk mindenre, az ilyen megközelítést gyakorló biztonsági szakemberek inkább az „igen, ha" kifejezést használhatják (azaz „igen, használhatjuk az eszközt, ha lecsökkentjük X veszélyét, visszatartjuk Y veszélyt" és így tovább). Másodsorban, azáltal, hogy nem csak az ismert veszélyforrásokra koncentrálunk, hanem általánosabb kockázatokat is figyelembe veszünk, a kockázat alapú megközelítés mélyebb védelmet tesz lehetővé, ahol a réteges vezérlés csökkenti az esélyét, hogy egyetlen fenyegetésnek globális következményei legyenek (tűzfal szabályok a chroot-olt alkalmazásokkal valamint behatolásjelző rendszerek használata és egyéb módszerek).

Remélem sikerült hasznos leírást készíteni, ami talán kicsit jobb rálátást ad a rovatunkban felbukkanó más, „inkább drótelvű" biztonsági eszközökre és módszerekre is.

Csak biztonságosan!

Linux Journal 2005. január, 129. szám

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél.