© Kiskapu Kft. Minden jog fenntartva

Vírusok, férgek, hátsó ajtók, trójaiak (16. rész)

Az előző részben az operációs rendszerek védelmi módszereiről esett szó, azaz megnéztük, miként tárolja a rendszer, hogy ki milyen erőforrásokhoz férhet hozzá, és azokkal mit tehet. A biztonság kérdése azonban itt nem merül ki. Hiába jó egy védelmi módszer, a rendszerre számtalan más veszély is leselkedik.

Ez a rész egy kicsit rendhagyó lesz, mivel olyan dolgokról ejtünk szót, amelyek nem feltétlenül esnek bele az operációs rendszert alkotó „fogaskerekek” ismertetésébe. Erre mégis szükség van, mert a rendszer biztonságát olyan dolgok is fenyegethetik, amelyek ellen az operációs rendszer egymaga nem tud védekezni – a felhasználónak is tennie kell az ügy érdekében. Ez azonban visszafelé is igaz: hiába tesz meg mindent a felhasználó, ha az operációs rendszer nem biztonságos, nem ér semmit az egész. A biztonság pedig nem egy utólag is hozzáadható tulajdonság. Erre már akkor készülni kell, amikor lerakjuk a rendszer első „alapköveit”.

Vegyük például a vírusokat: az operációs rendszer számára a vírusok ugyanolyan programok, mint a szövegszerkesztő vagy a parancsértelmező. A víruskereső alkalmazás is felhasználói szinten fut. Mégis létezik olyan operációs rendszer, amelyik kevésbé van kitéve a vírusok támadásának. A Unix-rendszerekben például a felhasználók nem módosíthatják a programok kódját, így a vírus nem képes további alkalmazásokat megfertőzni. Az MS-DOS-felhasználók azonban hiába rettegnek a vírusoktól és tennének meg mindent a fertőzés elkerülése végett, ilyenfajta védelemről nem is álmodhatnak. Hiába használunk azonban Linuxot, ha rendszergazdaként indítgatunk mindenféle kétes származású alkalmazást – a fertőzés veszélye emiatt ugyanúgy fennáll.

Még mielőtt azonban elmélyednénk a férgek és a vírusok természetében, az előző részből maradt adósságunkat törlesztjük.

Felhasználók hitelesítése

Mi, emberek olyan okosak vagyunk, hogy embertársainkat könnyedén felismerjük. A gépek azonban buták, ma még nem képesek úgy felismerni a felhasználót, mint ahogy mi a szomszédunkat a liftben. Ezért az operációs rendszernek nincs más választása: meg kell kérdeznie, hogy éppen ki akarja őt használni. Egyes rendszerek feltétel nélkül elhiszik, amit a felhasználó mond, más rendszerek azért ellenőriznék valamilyen módon. Ezt az eljárást nevezzük hitelesítésnek.

A hitelesítés három dolgon alapulhat: vagy a felhasználó tudását ellenőrizzük (jelszó), vagy a tulajdonát (mágneskártya), vagy egy olyan dolgot, amivel a felhasználó azonos (például ujjlenyomat). Most csak a jelszavas hitelesítéssel foglalkozunk, azt azonban érdemes megjegyeznünk, hogy igazán biztonságos csak akkor lehet a dolog, ha e három hitelesítési eljárás közül legalább kettőt együttesen és egymástól függetlenül használunk. (A bankautomaták például ilyenek. A pénz felvételéhez a PIN kódra is szükség van. Az egymástól független tárolás jelen esetben azt jelenti, hogy a PIN kód nem a mágneskártyán van tárolva.)

Jelszavas hitelesítés

Ez mind közül a legelterjedtebb, mivel könnyedén megvalósítható, és semmilyen különleges eszközt nem igényel, például mágneskártyát vagy ujjlenyomat-leolvasót. Csupán a felhasználó azonosítóját és jelszavát kell bekérni. A Unix esetében ezt a `login` nevű program végzi, amely a kapott jelszót egyből titkosítja is, majd összehasonlítja a jelszófájlban tárolttal. Ha ez egyezik, akkor beengedi a felhasználót, ha nem, akkor a belépési kérelmet visszautasítja.

A jelszavas azonosításnak akad egy rendkívül nagy hátránya, mégpedig az, hogy könnyű becsapni. A rendszer nem tud különbséget tenni a jogosult felhasználó és azon személy között, aki kitalálta vagy valamilyen módon megtudta a másik jelszavát. Ezért az olyan rendszerekben, ahol a védelmet jelszó segítségével oldják meg, a felhasználókat is komoly felelősség terheli. Egyrészt azért, hogy ne olyan helyre írják fel jelszavukat, ahonnan más is elolvashatja (de az a legjobb, ha le sem írják), másrészt azért, hogy ne könnyen kitalálható jelszót válasszanak maguknak.

Erre a két dologra ugyan már végtelenszer felhívták a felhasználók figyelmét, a többség még mindig egyszerű jelszavakat (például neveket, szótári szavakat) használ. Ezért komoly veszélyt jelenthet az úgynevezett szólistás módszer, amikor a támadó egy listát készít a gyakori jelszavakról, majd ezeket egyenként titkosítja; ezután az egészet összeveti a felhasználók rendszerben tárolt titkosított jelszavaival.

Itt védelmet nyújthat az, ha kikötjük, hogy a felhasználók csak olyan jelszavakat használhatnak, amelyben egyszerre szerepel kis- és nagybetű, illetve szám. Egy másik megoldás lehet a jelszófájl sózása. Ez azt jelenti, hogy a felhasználók jelszavához titkosítás előtt hozzáadunk egy n bites véletlen számot. Ezt az n bites számot is titkosítjuk, majd a jelszófájlban tároljuk. Természetesen, ha a felhasználó jelszót változtat, ez a szám is változik.

A sózás abban nehezíti meg a támadó dolgát, hogy jelentősen megnöveli jelszó-listájának a méretét. Ha például a támadó úgy gondolja, hogy az „almafa” karaktersorozatot valaki jelszóként használhatja, akkor a listában fel kell még tüntetnie az „almafa1”, „almafa2” stb. jelszavakat is. Ez azt jelenti, hogy a lista mérete az eredetinek 2^n-szeresére fog változni. A Unix-rendszerek is használják a sózást, ők az n értékének a 12-t választották. Ez már tekintélyes növekedést jelenthet a jelszófájlban.

A sózásnak elsősorban nem az az érdeme, hogy növeli az amúgy is csak pár megabájt méretű, gyakori jelszavakat tartalmazó lista méretét. Inkább az, hogy a nyers erő (broute force) módszerét alkalmazóknak keseríti meg az életét. Minden jelszó feltörhető, csak idő kérdése. Ha viszont a felhasználó kizárólag kisbetűkből álló, öt karakter hosszúságú jelszót használ, akkor az összes lehetőség végigpróbálása már nem tűnik annyira

kivitelezhetetlennek. A sózás használatával azonban jelentősen növelhetjük ezt az időt, bár az igazi megoldás az lenne, ha a felhasználók kizárólag legalább nyolc, kis- és nagybetűket, illetve különleges jeleket egyaránt tartalmazó jelszavakat használnának.
Az újabb Unixok abban is segítik a felhasználói jelszavak védelmét, hogy nem a mindenki számára olvasható */etc/passwd* állományban tárolják a titkosított jelszavakat, hanem egy, a felhasználóktól elzárt helyen. Azok a programok, amelyek ehhez hozzáférhetnek (például `login`, `passwd`) csak késleltetéssel adják vissza az eredményt, ezzel is lassítva a támadó munkáját.

Vírusok

A biológiaórán megismert vírusok olyan életformák, amelyek rendkívül egyszerű szerkezetűek: csak egyetlen DNS-ből (vagy RNS-ből) és az azt védő fehérjeburokból állnak. A vírusok ezért nem is tekinthetők valódi élőlényeknek, maguktól szaporodni sem képesek. Így hát önmaguk reprodukálása céljából hihetetlenül gonosz tetteket hajtanak végre: egészséges sejteket támadnak meg, és beépítik a saját DNS-üket a sejt élettani folyamatait szervező központjába. Tulajdonképpen átveszik az irányítást, azaz a sejtet arra kényszerítik, hogy mindennapi tevékenysége helyett inkább további vírusok előállításával foglalatoskodjon.
Az informatikaórán megismert vírusok olyan programrészek, amelyeknek a legfontosabb feladatuk az, hogy saját magukat ismételjék. Például más programokat úgy változtassanak meg, hogy azok tartalmazzák magának a vírusnak a kódját is (vagy annak egy másik változatát), ezáltal már a fertőzött program is a vírus újbóli megismétlésével fog foglalatoskodni. A számítógépes vírus működési elve tehát a biológiai víruséra rímel. Ahogy a biológiai vírus sem önálló élőlény, és a szaporodásához szüksége van egy gazdasejtre, úgy a számítógépes vírus sem önálló program, inkább csak gépi kódú utasítások halmaza, amelyeknek a végrehajtásához egy gazdaprogramra van szükség.
A vírussal megfertőzött sejt – miután több ezer új vírust gyártott le – rövid időn belül elpusztul. Vannak azonban olyan vírusok (az úgynevezett csendesen fertőző vírusok), amelyek nem okoznak jelentős károsodást a gazdasejtben, csupán a vírus további szaporodását teszik lehetővé. A számítógépes vírusok az utóbbi elvet követik, mivel lételemük a rejtőzködés. Ha ezt nem tennék, a felhasználó rövid úton felfedezné őket. Ezért egy életképes vírus a megfertőzött programot mindig csak annyira változtatja meg, hogy az a további programok megfertőzése mellett eredeti feladatát is kifogástalanul végezhesse. Ez a gyakorlatban a legtöbbször úgy működik, hogy a vírus kódja a megtámadott futtatható állomány végére kerül, és úgy változtatja meg a fejlécet, hogy először a vírus kódja hajtódjon végre. Miután a vírus elvégezte a dolgát (további állományokat fertőzött meg), az eredeti program is végrehajtásra kerül. Ezek az úgynevezett linkvírusok – ez a legősibb és legelterjedtebb vírusfajta, ám bizonyos alkalmazások létrejöttével megjelent egy, talán még tíz évvel ezelőtt lehetetlennek is mondott új linkvírus alfaj: a makróvírus. Legfontosabb tulajdonsága, hogy adatfájlokkal érkezik. Egy W betűvel kezdődő nevű operációs rendszernek van például egy szintén W-vel induló nevű szövegszerkesztője, amely lehetővé teszi, hogy a dokumentumokban a felhasználó úgynevezett makrókat helyezhessen el. A makró tulajdonképpen egy egyszerű programnyelv, amelyet a szövegszerkesztő értelmez, majd végrehajt. A makró eredetileg minden bizonnyal a „felhasználói beavatkozást igénylő (interactive) dokumentumok" létrehozását szolgálta, de elég fejlett volt ahhoz, hogy például fájlkezelő műveletekre használhassunk. Ennek köszönhetően megfelelő teret kínált vírusok számára, amelyek magát a szövegszerkesztőt használták futási közegként. A makróvírusok megjelenése két dolog miatt jelentett nagy csapást a személyi számítógépek biztonságára nézve: egyrészt azért, mert makróvírus írásához nem volt szükség magas szintű programozási ismeretekre; másrészt azért, mert a dokumentumcserék akkor még teljesen gyanútlanul mentek végbe, se a felhasználók, se a vírusirtók nem számítottak a dokumentumokból előbújó szörnyetegekre. Az internet elterjedése, és a különböző internetes alkalmazások egy rendszerbe történő „összevonása" a makróvírusok szétáramlását még jobban megkönnyítette…
Akadnak olyan vírusok is, amelyek másképp támadnak a rendszerre (bár többségük már csak az emlékezetünkben él). Ilyenek voltak például a társvírusok, amelyek nem nyúlkáltak a gazdaprogram kódjába, hanem az azt tartalmazó futtatható állományt nevezték át és tették rejtetté, majd a helyére saját magukat másolták. A gazdaprogram elindításakor valójában a vírus indult el, amely természetesen elindította az eredeti programot is. Ezek a vírusok azonban hamar lebuktak, ha valaki a gazdaprogram eredeti állományában keresett adatot vagy programkódot (azaz az úgynevezett overlay technikát használták). A társvírusok egyik rendszerre jellemző alfaját az „EXECOM" vírusok képezték, amelyek az MS-DOS-nak azt a tulajdonságát használták ki, hogyha parancssorból történő indításkor a felhasználó nem ad meg kiterjesztést, akkor a rendszer először a ***.COM*** kiterjesztéssel keres állományokat, s ha nem talál, akkor ***.EXE***-vel folytatja, végüla ***.BAT***-tal zárja a próbálgatást. Ha például létezett egy ***valami.exe*** nevű állomány, a vírus mellérakott egy ***valami.com*** nevűt, amely magát a vírust tartalmazta. Ezután már csak abban kellett reménykedni, hogy a felhasználó a programot a `valami` parancs kiadásával indítja el. Az EXECOM vírusok azonban egy Norton Commander nevű alkalmazás elterjedése következtében teljesen kihaltak.
Az eddig említett vírusok nagy hátránya, hogy addig teljesen hatástalanok, amíg egy fertőzött állomány végrehajtásra nem kerül. Ezért vannak olyan vírusok, amelyek már a gép indulásakor, még az operációs rendszer betöltődése előtt szeretnek aktiválódni. Ezek az úgynevezett indítóvírusok (boot virus), amelyek az indítórészben laknak,

© Kiskapu Kft. Minden jog fenntartva

© Kiskapu Kft. Minden jog fenntartva

így a gépszintű program (firmware software) (PC esetében a BIOS) már a gép indítása után egyből nekik adja a vezérlést. Az indítóvírusok általában egyben vírusok is, tehát nemcsak a hajlékonylemez-meghajtóba tett lemezek indító szektorát, hanem a gépen található alkalmazásokat is megfertőzik. Az indítóvírusok azonban nem jelenthetnek veszélyt az igazi védett módú operációs rendszerekre (például a Linuxra vagy a Windows XP-re).

Rejtőzködés

A vírusok tehát olyan programkódok, amelyek saját magukat sokszorosítják. A vírusírók pedig olyasfélék, akik olyan programkódokat írnak, amik saját magukat sokszorosítják. Ez már önmagában is kihívás, de a vírusírók a szaporodáson kívül mindig szeretnek valamiféle „magasabbrendű” célt kitűzni gyermekeik számára. Ez lehet például adatszerzés, rombolás, vagy éppen valami ártatlan kis semmiség. Mindenesetre minél szaporább egy vírus, annál nagyobb esélye van arra, hogy a kitűzött célt teljesítse.

A vírus tevékenysége azonban mindig valamilyen változással jár. Például, ha megfertőz egy futtatható állományt, akkor annak a mérete megváltozik. Mivel ezek a változások felismerhetők, fennáll a veszély, hogy a felhasználó gyanút fog, ami egyben (legalábbis azon a rendszeren) a vírus pályafutásának a befejezését jelentheti. A korszerűbb vírusok nemcsak arra ügyelnek, hogy a fertőzött programok futtatása közben a felhasználó semmi rendelleneset ne észleljen, hanem arra is, hogy a szaporodással járó változásokat minél jobban eltitkolják. Hogy erre milyen módszereket használhatnak, az nagymértékben függ magától az operációs rendszertől is. Például az MS-DOS, ami tulajdonképpen nem más, mint egy eljáráscsomag, semmiféle korlátozást nem szabott a programok számára. Mi több, a teljes vezérlést az alkalmazás kezébe adja, így a vírusok nyugodtan módosíthatták az indítórészt, vagy magukra irányíthatták a különböző rendszerhívásokat. Ezáltal olyan dolgokra is képesek lehetnek, hogy például a fertőzött állomány méretének a lekérdezésekor hamis eredmények szülessenek (így leplezve a méretváltozást).

Ilyesmi nem lehetséges egy olyan valódi védett módú operációs rendszer esetében, amelyben minden folyamat szigorúan csak a saját memóriaterületével rendelkezik. Ez természetesen nem jelenti azt, hogy például Linuxra nem lehet vírust írni. Vírust minden operációs rendszerre lehet írni, csak nem mindig érdemes. Ha például a vírust alkotója azzal a képességgel ruházza fel, hogy egy ismert biztonsági rést használjon ki, akkor a vírus sok csúnya dologra képes lehet, megfertőzheti magát az operációs rendszert is.

Mindenesetre a többfelhasználós rendszerek nagyobb ellenállást tanúsíthatnak a vírustámadásokkal szemben, hiszen az egyszerű felhasználók a programok többségét csak futtathatják, de nem módosíthatják. Az elindított vírus így csak a felhasználó tulajdonában lévő állományokra jelenthet veszélyt, a rendszer egészére azonban nem. Kivéve, ha nem a rendszergazda indít fertőzött állományt. A makróvírusok azonban a többfelhasználós rendszerekben is életképesek, mivel a dokumentumok a felhasználók tulajdonában vannak.

Vírusok nyomában

Sem most, és valószínűleg a jövőben sem fog létezni olyan algoritmus, amelyik egyértelműen és tévedhetetlenül képes kimutatni bármilyen vírus jelenlétét a rendszerben. Bár többféle módszer létezik, amelynek alapján megsejthetjük, hogy vírustámadás áldozatai lettünk, illetve felismerhetünk már felfedezett vírusokat, de ez is csak abban az esetben lehetséges, ha az adott vírus ügyetlenül vagy egyáltalán nem védekezik az alkalmazott módszer ellen.

A legismertebb ilyen módszer a szignatúrakeresés. Ez arra a feltevésre épül, hogy a vírusok kódjának valamely része biztosan egyedi, azaz nem fordulhat elő olyan állományban, amelyik az adott vírussal nincs megfertőzve. Ez máig a leghatékonyabb vírusészlelő módszer, és a vírust még a működésbe lépése előtt felfedezhetjük. Hatástalan azonban az ismeretlen vírusokkal szemben, továbbá a korszerű polimorf vírusokkal sem tud mit kezdeni, mivel ők képesek a saját kódjuk megváltoztatására. Egy másik nem elhanyagolható dolog, hogy a szignatúra-adatbázist folyamatosan frissíteni kell, és az adatbázis növekedésével a keresés ideje is növekszik.

Mivel a polimorf vírusok könnyedén kijátszhatják a szignatúrakeresőket, a heurisztikus keresés kifejlesztése vált szükségessé. A módszer lényege annyi, hogy a víruskereső elkezdi értelmezni a program kódját, gyanús tevékenységeket keresve benne. (Ilyen gyanús dolgok lehetnek például a különböző titkosító és polimorfikus eljárások). Előnye, hogy az ismeretlen vírusokat is képes felismerni, viszont nagyon nagy a hibaszázalék: nagyszámú a téves riasztás, és nem minden esetben akad a vírus nyomára.

Mivel a vírusok kivétel nélkül változásokat idéznek elő, a változások megfigyelésével felismerhetjük őket. Ehhez az kell, hogy még a vírus nélküli rendszerről mindent fel kell jegyeznünk (például a futtatható állományok tartalmát, illetve annak „lenyomatát” a CRC algoritmus használatával). Ezután már csak rendszeres időközönként össze kell hasonlítani a rendszer jelenlegi állapotát az eredetivel – ha vírus van a rendszerben, akkor a víruskereső észreveszi, de csak azután, miután a vírus működésbe lépett. További gond, hogy változást nem csak vírus okozhat, ezért téves riasztás is előfordulhat.

Láthattuk tehát, hogy mindegyik módszernek megvan az előnye és hátránya, így igazán hatékony megoldást csak együttes használatukkal érhetünk el. A jövő azonban bíztató, mivel kutatások folynak olyan „intelligens” víruskereső programok kifejlesztésére, amelyek képesek csapdába ejteni a gépet megtámadó vírust. Ezt úgy érheti el például, hogy egy pontosan ismert csali mindenfelé programokat helyez el a rendszerben, majd azokba véletlen írásokat kezdeményez, ezáltal „bírja rá” a vírust a csaliprogram megfertőzésére. Mivel ismeri az eredeti csalit, a vírus kódja könnyedén kinyerhető, elemezhető – esetleg önműködően azt észlelő és eltávolító

eljárást is készíthet hozzá. Ez a módszer sem lehet százszázalékosan megbízható, amíg nem lehetünk biztosak abban, hogy a memóriába ugyanazokat az adatokat kapjuk vissza, mint ami a lemezen szerepel (és nem azt, amit egy kósza vírus módosított). Ugyanez a nehézség a többi módszer esetén is fennáll, így az igazi megoldás talán az lenne, ha a víruskereső is az operációs rendszer belsejében foglalna helyet. Ám a siker ekkor sem lenne garantált.

A féreg

Féregvírusnak is szokás nevezni, mivel a férgeket is szaporodásra tervezték. A féreg (worm) azonban nem vírus. Ha megint egy, a biológiaórán megismert szereplőhöz szeretnénk hasonlítgatni, akkor a féreg a baktérium megfelelője lehet. A baktérium egy önálló élő szervezet, amely képes saját magától szaporodni, méghozzá gyorsan – elegendő tápanyag jelenlétében akár exponenciális ütemben is. A féreg tehát olyan önálló program, amelyik saját magát sokszorosítja és indítja el, egészen addig, amíg a rendszer össze nem omlik.
A legtöbb féreg azonban mindig nagyban gondolkozik. Míg a vírusok minél több állomány megfertőzésére törekszenek, addig a férgek az internet segítségével minél több gépre el szeretnének jutni. Mindig „egyedül" utaznak, és amint megérkeznek egy helyre, egyből azon gondolkoznak, hogyan juthatnának el más gépekre.
Az első igazán nagy pusztítást végző féreg 1988-ban szabadult el, és a Berkeley Unixok biztonsági réseit használta ki. Ez a féreg két részből állt: a féreg magjából és egy áthúzó programból. Az utóbbit a megtámadni kívánt gépen kellett lefordítani és lefuttatni. Feladata csak annyi volt, hogy kapcsolódjon ahhoz a géphez, ahonnan érkezett, majd töltse át a féregmagot, majd fordítson és futtasson. A féreg magja megnézte a */etc/hosts* nevű állományt, amelyből kiderült, hogy az adott gép mely további gépekkel áll kapcsolatban. Ezekre áttöltötte az áthúzó programot, és kezdődött minden az elejéről.
Az igazán érdekes azonban az, hogy a féreg miként volt képes rávenni a távoli gépeket az áthúzó program lefuttatására. Három módszert is kipróbált rá. Először az `rsh` (remote shell) parancs segítségével próbálta meg elérni a távoli gép parancssorát. Ha sikerült, akkor az áthúzóprogramot áttöltve és lefordítva megvalósult a fertőzés.
Amennyiben ez nem jött volna be, a `finger` démon egy ismert távoli átmeneti tár túlcsordulási hibáját aknázta ki. Ennek a módszernek az a lényege, hogy értékként egy jó nagy, különlegesen összeállított adatblokkot adunk át. Ez az adatblokk bőven nagyobb volt, mint az átmeneti tár, így az túlcsordult, és a verem felülírásra került. Ha képesek vagyunk távoli program vermébe írni, akkor megváltoztathatjuk annak a visszatérési címét. Így a `finger` feladata befejezte után nem az eredeti helyre tért vissza, hanem elindított egy parancssort.
A féreg ehhez kapcsolódva könnyedén telepíthette magát.
A harmadik próbálkozás a Sendmail nevű levelezőrendszerrel történhet: a féreg egy másik gépre (levélként) átküldhette, majd futtathatta magát.

A hálózatok többségében a felhasználóknak nem csak egy géphez van hozzáférésük, így a féreg arra is vette a fáradságot, hogy nekiálljon feltörni a felhasználók jelszavát, és az ilyen módon megszerzett gyenge jelszavak segítségével megpróbáljon a többi gépbe is beférkőzni.
A vírusokhoz hasonlóan a férgeknek is rejtőzködniük kell, hogy minél tovább szaporíthassák önmagukat. Ezért nem jó, ha a féreg túl korán lelassítja a megfertőzött gépeket. A most tárgyalt féreg például a fertőzés előtt megnézte, hogy hány példányban fut már a kérdéses gépen. Ha már több mint hét példányban futott, akkor inkább más célpont után nézett.
Ez a féreg annyira hatékony volt, hogy rövid időn belül több ezer gépet volt képes megfertőzni és megbénítani (ami akkoriban az internet jelentős részét képezte). A féreg azért is mérföldkőnek számított a számítógépes biztonságtechnika terén, mert képes volt olyan biztonsági hibákat kihasználni, amelyek segítségével felhasználói beavatkozás nélkül tudott terjedni.
Azóta több féreg is átsöpört a világon, volt olyan is, amelyik nem volt ennyire összetett, több közülük csupán egy levelezőügyfél hibáját próbálta kihasználni. Általában azért a „felhasználó" is hibás volt, mert megnyitotta a férget tartalmazó levelet.

Hátsó ajtók és trójai falovak

A hátsó ajtó egy olyan kód, amelyik rést nyit a rendszeren, és illetéktelenek belépését teszi lehetővé, illetve adatokat juttat ki a rendszerből. A trójai faló egy olyan alkalmazás, ami valami mást is csinál, mint amit hirdet magáról, és azt a valamit nagyon jól titkolja.
Fontos, hogy sem a hátsó ajtó, sem a trójai faló nem vírus és nem is féreg, mivel egyik sem szaporodik. Az más kérdés, hogy sok vírus egyben hátsó ajtó is, tehát alkotója számára belépést biztosít a fertőzött gépekre. A trójai programok is tartalmazhatnak hátsó ajtót (ebben az esetben a hátsóajtó-készítés a titkos feladat), de lehetnek például vírusgazdák is. A vírusgazda nem egy vírussal fertőzött program, csak tartalmazza a vírust, és végrehajtáskor azt útjára engedi a rendszerben.
A következő részben szó lesz még a rejtett csatornákról és a felhasználói réteg alkalmazásairól. Elsősorban a démonokról írok majd, amelyek fontos építőkövei a Unix-rendszereknek (még ha nem is tekinthetők közvetlenül az operációs rendszer részének). Nem kanyarodunk el azonban véglegesen a biztonság kérdésétől, mivel megtudhatjuk azt is, hogy egy könnyen „korrumpálható" démon milyen mértékben áshatja alá a rendszer biztonságát (a válasz: nagymértékben).

Garzó András (garzoand@interware.hu)
Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

© Kiskapu Kft. Minden jog fenntartva