HADMÉRNÖK

XI. Évfolyam 4. szám - 2016. december

Szádeczky Tamás
szadeczky.tamas@uni-nke.hu

KRIPTOGRÁFIAI PROTOKOLLOK MEGFELELŐSÉGE

Absztrakt

Az informatikában a biztonságos kommunikációt kriptográfiai algoritmusokkal és rájuk épülő protokollokkal tudjuk megvalósítani. Ahogy az elmúlt két évben kiderült, a biztonsági szoftverek – hasonlóan minden más PC alkalmazáshoz – tartalmaznak szoftverhibákat. A cikk a hibák kezelésének műszaki és üzleti oldalát elemzi, a szabványalkotó, a jogalkalmazó és az üzleti gyakorlat szempontjából, így különös tekintettel a kriptográfiai protokollok elavulásának a szabványalkotó általi kezelésére.

Secure IT-based communication is achievable with the application of cryptographic algorithms and protocols. As it became clear in the last two years, security software, similarly to any other PC software, contains bugs. The article analyses the technical and business sides of treating those bugs from the aspects of the standardizing body, the regulating authority and the business practice. The focus is the treatment of the aging of cryptographic protocols by standardization bodies.

Kulcsszavak: *kriptográfia, SSL, TLS, PCI DSS, Java, böngésző, szabvány-megfelelőség ~ cryptography, SSL, TLS, PCI DSS, Java, browser, standard compliance*

BEVEZETÉS

A 2000-es évek óta megnövekedett a nyomás a szoftverfejlesztőkön, hogy a termék minél előbb a piacra kerülhessen, így az üzleti igény kielégítésre kerüljön. Ez nyilván rövidebb fejlesztési időt és sajnos kevesebb tesztelést jelent. Ezt az üzleti hozzáállásbeli változást lekövette a fejlesztési módszertan is és a korábbi vízesés modellt felváltotta az agilis szoftverfejlesztés. [1] Ez azt jelenti, hogy szemben a korábbi specifikáció – fejlesztés – tesztelés folyamattal, maga a specifikáció sem készül el a munka megkezdése előtt, hanem az is folyamatosan változik az üzleti igényeknek megfelelően. Ehhez átalakult a tesztelés módszertana is, de sajnos az nem megfelelően hatékony a hibák kiszűrésére. Ebből kifolyólag a fejlesztett szoftverek túlnyomó többsége hibákat hordoz magában, amelyek a felhasználó számára jobb esetben csak a nagyszámú rendszeres frissítés megjelenésével válik láthatóvá. Az informatika felhasználójaként természetessé vált, hogy például havi rendszerességgel kapunk operációs rendszerünkhöz és felhasználói szoftvereinkhez kritikus biztonsági frissítéseket. Súlyosabb esetben ez oda is vezet, – mint az Adobe Flash Player esetében – hogy egyes szoftver beszállítók letiltják ezen túlzottan sérülékeny szoftverek alkalmazását. [2] Ezekhez a problémákhoz sajnálatosan hozzászoktunk, de alapjaiban rengeti meg az információbiztonsági szakmát, ha pont a sérülékenységekből fakadó kockázatok minimalizálásra létrehozott biztonsági szoftverekben találnak kritikus sérülékenységeket.

A TECHNIKAI PROBLÉMA

A 2014-es év több ilyen kritikus sérülékenységet hozott és azóta is több probléma került napvilágra. 2014 áprilisában került nyilvánosságra a HeartBleed bug, amely az OpenSSL kriptográfiai függvénykönyvtárában lévő hiba, ami harmadik fél részére lehetőséget biztosít a RAM-ban tárolt adatokhoz való hozzáféréshez, így akár a felhasznált kriptográfiai kulcsokhoz, jelszavakhoz. [3] Így a kommunikáció titkosítására használt program nyitva hagyta a támadók előtt az egyébként elvileg megfelelő titkosítási algoritmussal kommunikáló felek rendszereit. Ezzel, mint kiderült, egy lavina indult el és újabb és újabb sérülékenységekre derül fény. Ez a helyzet technikai oldalról egyértelműen kezelhető, az adott szoftverkomponenst újabb, javított verzióra kell frissíteni, vagy ahogy a Linux rendszereknél jellemző, a korábbi verziót javítják és a teljes alkalmazás frissítése nélkül az adott hiba orvoslásra kerül (backporting). Nem olyan egyszerű viszont a kérdés a biztonsági megfelelőség (compliance) illetve az üzleti követelmények szempontjából.

A fentiekhez hasonló problémakör a kriptográfiai algoritmusok és protokollok elavulása. Minden jelenleg alkalmazott negyedik generációs titkosítás tökéletlen (kivéve a szervezési okokból hatékonyan nem használható One Time Pad algoritmust), tehát feltörhető, viszont ez jó esetben időben lehetetlen feladat. Megfelelő algoritmusnak tekintjük például azt, ami a Föld összes számítógépének együttes számítási kapacitásával évezredek-évmilliók alatt törhető fel. Ezek a nagy számok kifejezetten hangzatosak, de valójában – a számítási kapacitás Moore-törvény szerinti növekedése és kódtörési feladatok többezer magos grafikai processzorokra való áthelyezése miatt – már az eredmény, hogyha évtizedekig megfelelő védelmet nyújt az adott algoritmus az adatainknak. Így folyamatosan avulnak el a régebben biztonságosnak tekintett algoritmusok, mint ahogyan az a DES és az RC4 szimmetrikus algoritmusokkal történt. A probléma megoldása mindössze annyi, hogy a korábbi algoritmusokat a kor színvonalának megfelelő jobb algoritmussal helyettesítjük, az adatokat újrakódoljuk. Külön kihívást jelent a speciális célú kriptográfiai algoritmusok – mint a katonai szenzorhálózatokban alkalmazott MINISEC algoritmus – jóságának megítélése, hiszen ezek alkalmazása csak nagyon szűk területen történik, így az azt tesztelő hackerek is kevesen vannak. [4] Mindemellett a védelmi rendszer kialakításakor figyelembe kell venni, hogy a

kriptográfiai eljárások kijátszhatóak a rejtjelezés előtt történő kompromittálással, például a billentyűzet közvetlen rádiófrekvenciás lehallgatásával. [5] Hasonló a probléma a lenyomatképző (hash) függvények esetében is, ahol a képzett lenyomat elveszíti az egyediségét. A lenyomatképző függvények alkalmazásának célja az, hogy egy tetszőleges hosszúságú bemenet egy állandó hosszúságú kimenetre (például 128-256 bitre) kerül leképzésre. Magától értetődő, hogy minden tetszőleges hosszúságú bemenetet nem lehet párszáz biten ábrázolni úgy, hogy minden hash érték egyedi legyen. Szükségszerűen lennie kell ütközésnek, tehát két különböző bemenet egyszer csak ugyanazt a kimenetet fogja produkálni. Egy lenyomatképző algoritmust addig tekintünk megfelelőnek, amíg nem találnak valamilyen ütközést. [6] Ez történt az MD4, MD5 és az SHA-1 algoritmusok esetében is. A megoldás itt is az algoritmus leváltása.

Technikai szempontból tehát a megoldás az alkalmazott kriptográfiai algoritmusok és az azt alkalmazó protokollok avulásának és sérülékenységeinek nyomon követése és szükség esetén azok leváltása egy korszerű megoldásra. Maga a leváltás a tárolt adatok esetében azok dekódolását, majd az új megoldás szerinti rejtjelzését jelenti. Adatátvitel esetében pedig mindössze az alkalmazott kriptográfiai algoritmust, illetve protokollt kell lecserélni.

AZ ÜZLETI PROBLÉMA

Ami technikai szempontból egy „egyszerű” frissítéssel, vagy cserével megoldható lenne, az az üzleti valóságban sajnos nehezen kivitelezhető. A szabályozó, normaalkotó szervek jobb esetben előremutatóan megkövetelik a gyenge kriptográfia cseréjét, majd az érdekelt felek kérésére gyakran kénytelenek meghosszabbítani a határidőket, mint ahogy azt a Payment Card Industry Security Standards Council (PCI SSC) is tette. A bankkártyás fizetések biztonságát a Visa Inc., MasterCard, American Express, Discover Financial Services, és JCB International által alapított PCI SSC több, általa kiadott szabványban határozza meg. Ezek közül a legszélesebb körben ismert és alkalmazott Payment Card Industry Data Security Standard (PCI DSS) szabvány határozza meg a bankkártya elfogadók által alkalmazandó biztonsági szabályokat. Ennek alkalmazása minden bankkártyás fizetést elfogadó és az annak lebonyolításában résztvevő szervezet számára kötelező. A szabvány – szemben más információbiztonsági szabványokkal, mint az ISO 27001 – egy bizonyos részterületre koncentrálva különösen részletes technikai szabályokat határoz meg. Az egyik ilyen szabálycsoport a bankkártya adatok nyilvános hálózatokon való átküldésének szabályait rögzíti. A 4.1. pontban meghatározottak szerint csak megfelelő kriptográfiai algoritmussal és azt alkalmazó protokollal szabad bankkártya-adatokat nyilvános hálózatokon átküldeni. A 2013 novemberében kiadott 3.0 szabványban az SSL és TLS protokollok még jó példaként szerepeltek. A 2015 áprilisában megjelent 3.1-es verzióban éppen a fent leírt OpenSSL sérülékenységek miatt már az SSL és a korai TLS protokollokat nem tekintik biztonságosnak és az új alkalmazások esetében nem is használhatóak. Annak ellenére, hogy a PCI DSS szerint csak a szabványnak leírtak tekinthetők normatív leírásnak a PCI SSC a szabvány átírásának mellőzésével több dokumentumot is kibocsátott, amelyek az átállás időpontját és módját határozzák meg. [7] Mindemellett viszont sehol nem definiálják a korai TLS verzió fogalmát, kételyek közt hagyva az azt alkalmazó szakembert. Amerikai szervezet lévén a PCI SSC elsősorban a National Institute of Standards and Technology (NIST) szabványaira hivatkozik. Az NIST SP 800-52 Rev.1-ben részletesen foglalkoznak a TLS protokoll különböző verzióinak biztonságos paraméter beállításaival, így ebből kiderül, hogy a TLS 1.0 egyáltalán nem tekinthető biztonságosnak, a TLS 1.1 megfelelő paraméterezéssel biztonságosnak tekinthető és a TLS 1.2 jelenleg feltétel nélkül biztonságos. A TLS 1.1 esetében le kell tiltani a már nem biztonságos kriptográfiai algoritmusok alkalmazását, valamint ki kell kapcsolni a visszafele kompatibilitást engedélyező paramétereket. Annak ellenére, hogy az átállás végső

határidejére a PCI SSC a 2016. június 30-at irányozta elő, szembesülnie kellett azzal, hogy egyre növekvő ellenállásba ütközik az alkalmazói oldalról. [7] Az a probléma ugyanis, – amit a szerző maga is, egy webportál biztonsági konfigurálásakor is tapasztalt – hogy egyes, széles körben elterjedt alkalmazások a TLS 1.2 bevezetése után nyolc évvel még mindig nem támogatják azt. A probléma tehát nem az, hogy az üzemeltető ne tudna frissíteni egy függvénykönyvtárat, vagy telepíteni egy új alkalmazást, hanem jelentős beruházást igénylő új fejlesztések, adatmigrálások és szerkezeti átalakítások válhatnak szükségessé. A pénzügyi világ alapvetően törekszik a biztonságos megoldások alkalmazására, viszont előtérben mégiscsak a fizetések lebonyolítása, a tranzakciók lefuttatása áll. Nem engedhető meg a rendszerek kiesése és minden fejlesztés elég komoly tesztelésen kell, hogy túlessen. Megértve az üzleti oldal nehézségeit a PCI SSC az átállás határidejét 2018. június 30-ra tolta ki, de hogy megakadályozza az alkalmazói oldal esetleges passzív, elodázó hozzáállását, rendszeres kockázatértékelési és jelentési kötelezettséget írt elő a meghosszabbított határidő mellé.[1] Így tehát az átállást elodázó alkalmazók megnövekedett adminisztratív terhekkel néznek szembe és minden auditon magyarázkodásra szorulnak. Különleges helyzetben vannak a kártyával történő fizetést biztosító POS terminálok. Itt is széles körben alkalmazott az SSL és TLS protokollok teljes köre, viszont mivel itt – honlapok, webalkalmazások és háttéradatbázisok híján – a sérülékenységek kisebb valószínűséggel használhatók ki, ezért a szabályok is megengedőbbek. Tekintettel arra, hogy az OpenSSL eddig ismert sérülékenységei csak webalkalmazások esetén használható ki, a POS terminálok ennek hiányában eddig nem veszélyeztetettek.

Ebben az esetben is szükséges viszont a rendszeres kockázatértékelés elvégzése, amelynek keretében a gyártó vagy üzemeltető felméri, hogy valóban nem áll-e fent a POS terminál veszélyeztetettsége. Ebben az esetben a szabványalkotó nem is írt elő átállási határidőt. Fontos viszont, hogy mindez csak a jelenlegi kockázati környezetre és az eddig ismert sérülékenységekre igaz, így mindig követni kell az aktuális iparági riasztásokat, sérülékenységi listákat.

GYAKORLAT ÉS KITEKINTÉS

A szerző egy multinacionális cég megbízásából több nyugat európai szolgáltató és kereskedő PCI DSS auditját is elvégezte. A 2016-ban végzett auditok jelentős hányadánál már az átállás a TLS 1.2-re megtörtént, a PCI SSC által eredetileg kitűzött határidőre. Jellemzően problémás viszont a szerveroldali Java alkalmazásokat futtató webszolgáltatások átalakítása, ugyanis a Java csak a Java Development Kit (JDK) 7 verziótól kezdődően támogatja a TLS 1.2 alkalmazását és a JDK 8 használja azt alapértelmezetten. [8]

1. táblázat: Java Development Kit SSL/TLS titkosítási protokoll-támogatása [8]

Verzió	**JDK 6**	**JDK 7**	**JDK 8**
Megjelenés	2006	2011	2014
Alapértelmezett protokoll	TLS 1.0	TLS 1.0	TLS 1.2
Támogatott protokollok	TLS 1.1 (csak JDK 6 update 111 és újabb) TLS 1.0 SSL 3.0	TLS 1.2 TLS 1.1 TLS 1.0 SSL 3.0	TLS 1.2 TLS 1.1 TLS 1.0 SSL 3.0

[1] lásd PCI DSS 3.2 Appendix A2

A legfontosabb PC böngészők piacán a helyzet hasonló, itt a 2006 áprilisában megjelent TLS 1.1-et és a 2008 augusztusában megjelent TLS 1.2-t a böngészők 7 évvel később kezdték teljes körűen támogatni. Ami nehezen érthető, hogy mind a Firefox-ban, mind az Internet Explorer-ben ezek alapértelmezetten ki voltak kapcsolva, így a felhasználók csak a részletes technikai beállítások megváltoztatásakor tudhatták volna használni azt. Jelentős különbség a JDE és a böngészők között, hogy amíg a böngészők gyakorlatilag automatikusan frissülnek, a fejlesztési keretrendszerek közötti váltásra migrációs folyamatot kell végrehajtani. Tehát amíg elavult Chrome és Firefox böngésző gyakorlatilag elenyésző számban, elavult Internet Explorer a régi Windows-verziókon van, addig Java 6-ot futtató szerverek még nagy számban vannak, attól függetlenül is, hogy a támogatása már a Java 7-esnek is véget ért.

2. táblázat: Asztali böngészők SSL/TLS titkosítási protokoll-támogatása [11] [12] [13]

Állapot	**Chrome**		**Firefox**		**Internet Explorer/Edge**	
	ver	év	ver	év	ver	év
TLS 1.0 támogatás	1	2008.12	1.0	2004.11	4 (letiltva) 7	1997.09 2006.10
TLS 1.1 támogatás	22	2012.09	23 (letiltva) 27	2013.08. 2014.02	8 (letiltva) 11	2009.03 2013.10
TLS 1.2 támogatás	30	2013.09	24 (letiltva) 27	2013.09 2014.02	8 (letiltva) 11	2009.03 2013.10
SSL 3.0 letiltva	40	2015.01	34	2014.12	IE 11 (security update 3038314)	2015.04
Aktuális verzió	52	2016.07	48	2016.08	IE 11 Edge 14	2013.10 2016.08

Emellett meglepően nagy számban tapasztalható a POS terminálok elavult verzióval való csatlakozása. Így akár egy több tízezres POS terminál hálózat is SSL 3.0-val kapcsolódik az elfogadó bankhoz. Itt a legnagyobb problémát a kiterjedt helyszíni szoftverfrissítés szükségessége vagy akár a teljes hardver csere jelenti. Ez egy fizetési szolgáltató (payment service provider) esetében akár milliárdos költséget is jelenthet a teljes infrastruktúra tekintetében. Itt tehát nagyon lassú változásra kell számítanunk. Maga a szabványalkotó sem merte feltenni a kérdést, hogy mi történik akkor, ha a POS terminálokat is érintő SSL/TLS sérülékenység kerül napvilágra.

Szemben a PCI DSS-ben meghatározott, viszonylag részletes követelményekkel és határidőkkel más normákban nem találunk ilyen előírást. A méltán népszerű és széles körben alkalmazott ISO27001 mindössze azt írja elő, hogy az alkalmazott kriptográfiai kontrollokról szabályzatot kell alkotni, szabályozni kell a kulcsok használatát, védelmét és az életciklusát, de konkrét algoritmust nem említ meg, így az esetleges átállást is az alkalmazóra bízza. Áttételesen tehát, de a kockázatértékelés és elemzés keretében írja elő az elavult algoritmusok kockázatának kezelését.

A hatósági gyakorlatban szintén kevéssé jellemző a fenti problémakör kezelése, de üdítő kivételként meg kell említeni a Nemzeti Média és Hírközlési Hatóság elektronikus aláírásokra vonatkozó határozatait, amelynek keretében például az EF/26838-x/2011 sz. határozatokkal kötelezővé tette a hitelesítés-szolgáltatók átállását az SHA-256 lenyomatképző algoritmusra. [9] Ezzel lekövetve az European Telecommunications Standards Institute (ETSI), a távközlés területén működő regionális szabványügyi testület vonatkozó ajánlását és meghonosítva a jó nemzetközi gyakorlatot hazánkban is. [10]

ÖSSZEGZÉS

Összefoglalásként elmondható, hogy a kriptográfiai algoritmusok és protokollok sérüléseiből illetve elavulásából adódó átállási feladatokat – amelyek egyébként minden informatikai üzemeltetőre és alkalmazóra feladatot rónak – a szabványalkotók és hatóságok mindössze kis hányada kezeli. Ennek oka egyrészről a szabályozásban általában elvárt technológiai függetlenség, másrészről a technikai változások üzleti szempontból történő lekövetésének gyakorlati nehézségei, ahogy azt a PCI DSS SSL és korai TLS verzióváltással kapcsolatos felhasználói ellenállása és az átállási idő módosítása is mutatja. Az auditálási tapasztalatok azt mutatják, hogy az új átállási idő tartható és a felhasználók többségénél a problémát már megoldották, de vannak még kivételek.

Felhasznált irodalom

[1] Indira Nurdiani, Jürgen Börstlera, Samuel A. Frickera, The impacts of agile and lean practices on project constraints: A tertiary study, Journal of Systems and Software, Volume 119, September 2016, Pages 162–183

[2] Matt Burgess: Google Chrome will start blocking Flash by default, Wired, 16 May 2016 [2016.08.01.] http://www.wired.co.uk/article/google-chrome-adobe-flash-html5

[3] CVE-2014-0160, CVE-2014-0346

[4] Nagy Dániel: Kriptográfiai kihívások a vezeték nélküli szenzorhálózatokban, Hadmérnök, XI. évf. 2016/1. sz.

[5] Szabó Tibor: Kormányzati informatikai hálózati infrastruktúrák védelmi rendszereinek új kihívásai a vezeték nélküli kommunikáció tükrében, Bolyai Szemle, 2015/3. sz.

[6] Dario Forte, The death of MD5, Network Security, Volume 2009, Issue 2, February 2009, Pages 18-20

[7] PCI SSC: Bulletin on Migrating from SSL and Early TLS https://www.pcisecuritystandards.org/pdfs/Migrating_from_SSL_and_Early_TLS_-v12.pdf [2016.08.05.]

[8] Erik Costlow: Diagnosing TLS, SSL, and HTTPS https://blogs.oracle.com/java-platform-group/entry/diagnosing_tls_ssl_and_https

[9] NMHH: Algoritmusokkal kapcsolatos határozatok http://nmhh.hu/tart/index/1442/Algoritmusokkal_kapcsolatos_hatarozatok

[10] ETSI TS 102 176-1 v 2.1.1 (2011-07) „Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures; Part 1: Hash functions and asymmetric algorithms”

[11] Mozilla: Firefox Releases https://www.mozilla.org/en-US/firefox/releases/ [2016.08.21.]

[12] Chrome Releases http://googlechromereleases.blogspot.hu/ [2016.08.21.]

[13] Transport Layer Security https://en.wikipedia.org/wiki/Transport_Layer_Security [2016.08.21.]