Felhő alapú architektúrák, és számítógépes hálózatok biztonsága

Vörös Péter[1], Kiss Attila[2]

[1] vopraai@inf.elte.hu
ELTE IK Információs rendszerek tanszék,
BalaBit IT Security
[2] kiss@inf.elte.hu
ELTE IK Információs rendszerek tanszék

Absztrakt. Egyre nagyobb teret hódit magának a „Cloud Computing”, ennek oka, hogy a rendszer gyors, és robusztus lesz, architektúrális felépítése pedig teljesen rejtve marad az átlagos felhasználók elől. A cikkben összefoglaljuk, hogy hogyan néznek ki ezek az architektúrák, ismertetjük a különböző cloudok fajtáit az általuk megvalósított szolgáltatás függvényében. Szó lesz a felhasználók túlzott bizalmáról a publikus felhők irányába, arról hogy milyen általános támadási módszereket ismerünk cloudok ellen, illetve hogy hogyan tudjuk megvédeni magunkat a rosszindulatú felhasználóktól.

1. Bevezetés

2010 óta a „Cloud Computing” egyre jelentősebb teret hódít magának, ennek oka, hogy a rendszer gyors, és robusztus lesz, architekturális felépítése pedig teljesen rejtve marad az átlagos felhasználók elől. A megvalósított szolgáltatások függvényében több egymásra épülő szolgáltatás réteget különböztethetünk meg. Ilyenek a Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS).

A biztonságot minden rendszer tervezésénél szem előtt kell tartani, különösen igaz ez, ha felhőről beszélünk, hiszen itt jellemzően nagyságrendekkel több adatot kezelünk, melyek egy jelentős része gyakran igen szenzitív. Erre egy egyszerű megoldást jelenthet az adatok titkosítása, ebben az esetben viszont kiemelt figyelemmel kell kezelni a kulcsokat, azokat felhőben tárolni komoly veszélyforrás.

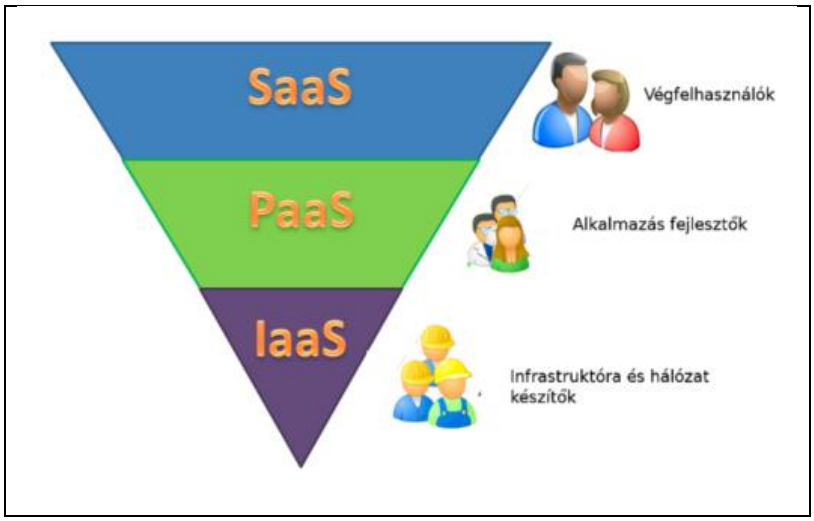
Több különböző támadási módszer ismert felhő alapú rendszerek ellen. A cikkben először részletesen ismertetjük a különböző cloud típusokat, majd szó lesz arról hogy mi az az információbiztonság, és hogyan gondoskodhatunk az adataink védelméről. Ezek után részletesen bemutatjuk a különböző szolgáltatáskiesést okozó (Denial of Service) más néven DoS támadásokat, és az ahhoz tartozó védelmi módszereket.

2. Cloud architektúrák

A szolgáltatásokban, amiket a különböző cloud rendszerek megvalósítanak, a közös, hogy ezek biztosítását elosztva több hardveren, nem egy dedikált eszközön, a felhasználóktól elrejtve végzik. A biztosított szolgáltatások publikus felhő esetén, az Interneten, privát felhő esetén az intraneten, helyi hálózaton érhetőek el a felhasználók számára.

A felhő rendszereket az általuk megvalósított szolgáltatások függvényében három nagy csoportra oszthatjuk:

- Szoftver szolgáltatás (Software as a Service)
- Platform szolgáltatás (Platform as a Service)
- Infrastruktúra szolgáltatás (Infrastructure as a Service)



1. ábra: A Software as a Service célközönsége a végfelhasználók, az alkalmazásfejlesztők eggyel alacsonyabb rétegben PaaS szinten kapcsolódnak be. IaaS szinten pedig csak nagyon szűk rétegek az infrastruktúra vagy hálózat készítők mozognak

2.1. Szoftver szolgáltatás

Már az 1960-as években is használtak úgynevezett centralizált számítóközpontokat az IBM-nél, amik lényege, hogy dedikált képekre csatlakozva, terminálból bizonyos szolgáltatások elérhetőek voltak. Az Internet terjedésével egyre elterjedtebbek lettek az Application Service Providerek ASP-k, ezek valamilyen ismert protokollon (jellemzően HTTP-n) tettek elérhetővé különböző főként üzleti szolgáltatásokat.

A SaaS leginkább úgy jellemezhető, mint az ASP-k kiterjesztése. Míg az ASP-k többnyire egy harmadik cég programját teszik elérhetővé, addig a Software as a Service-t biztosító cég jellemzően egy saját szoftvert fejleszt, és tart karban. A másik nagy különbség, hogy az ASP-knél gyakori a hagyományos kliens-szerver architektúra, ennek megfelelően többnyire telepíteni kell valamilyen kliensszoftvert a szolgáltatás igénybevételéhez. Ezzel szemben a SaaS esetében manapság már a böngésző elegendő a cloudban használható szoftver eléréséhez.

Néhány széleskörűen elterjedt és jól ismert SaaS megoldás:

- Google Apps
- SAP
- Microsoft Office 365

2.2. Platform szolgáltatás

A platform szolgáltatás a szoftver szolgáltatás alatti rétegként fogható fel. Azzal ellentétben nem csak egy használatra kész szoftvert, hanem egy olyan platformot biztosít, amire a felhasználó maga készítheti el, telepítheti fel az alkalmazásokat. Az alkalmazás üzemeltetéséhez szükséges környezetet biztosítja, terheléselosztással és feladatátvétellel, kezelő felülettel, ezek rendszeres biztonsági frissítésével.

Platform szolgáltatással találkozhatunk például az alábbi helyeken:

- Google App Engine
- Amazon Web Services
- Windows Azure

2.3. Infrastruktúra szolgáltatás

A legalacsonyabb szintű cloud szolgáltatás modell, fizikai vagy virtuális erőforrásokat (processzort memóriát, tárhelyet, stb...), szervereket, load-balancereket, hálózatot szolgáltat. Ahhoz hogy az alkalmazásunkat futtassuk ilyen környezetben először az operációs rendszert kell feltelepíteni, és beállítani, tehát gyakorlatilag egy platformot kell összerakni.

Infrastruktúra szolgáltatást megvalósító ismertebb programok:

- Amazon EC2
- Google Compute Engine

3. Információbiztonság

Az információbiztonság az adatok bizalmasságát, sértetlenségét, és rendelkezésre állását jelenti.

A bizalmasság, biztosítja, hogy minden információ csak a felhatalmazott felhasználók számára legyen elérhető. A sértetlenség, az információk és a feldolgozási módszerek teljességének és pontosságának megőrzését jelenti. A rendelkezésre állás pedig, annak biztosítása, hogy a felhasználók a szükséges adatokhoz mindig hozzá tudjanak férni.

Az információbiztonság minden számítógépes rendszer kialakításakor fontos szerepet játszik, különösen igaz ez a cloudok használata során. Felhőkben jellemzően nagyságrendekkel több adatot tárolunk, mint hagyományos különálló rendszerekben, így ezen rendszerek védelme is kritikus.

3.1. Biztonság publikus szolgáltatóknál

Ha valamilyen külső szolgáltatást veszünk igénybe, akkor a fizikai rendszer védelme nem a mi feladatunk, éppen ezért soha nem lehetünk benne teljesen biztosak, hogy az adataink nem kerültek, illetve kerülhetnek illetéktelen kezekbe. Éppen ezért külső szolgáltatóknál titkosítatlanul csak olyan adatokat szabad tárolni, amik között semmilyen szenzitív információ nincs.

Több megoldás született már kevésbé biztonságkritikus adatok biztonságos tárolására is, létezik például olyan proxyként használható tűzfalmegoldás, amivel a Google Calendar bejegyzéseket tudjuk titkosítani.

Bármilyen titkosításról is beszélünk, a visszafejtéshez illetve elkódoláshoz szükséges jelszavakat vagy kulcsokat különös figyelemmel kell illetni. Az egyik leggyakrabban elkövetett titkosítással kapcsolatos hiba nyilvános cloudok esetében, hogy a titkosítás kulcsát együtt tároljuk a titkosított adattal, vagy más nyilvános felhőben elérhetővé tesszük.

4. Támadás és Védekezés

Minden informatikai rendszer esetén a támadóknak a motivációja vagy adatszerzés, vagy a szolgáltatás megbénítása, éppen ezért bármilyen rendszer tervezésénél oda kell figyelnünk a megfelelő biztonsági körülmények megteremtésére. Ahhoz hogy ebbe részletesebben is belemenjünk először nézzük meg, hogy milyen támadási módszereket ismerünk.

A támadásokat két nagy csoportra oszthatjuk jellegüket, illetve céljukat tekintve. Adatokkal való visszaéléssel, vagy szolgáltatáskieséssel (Denial of Service DoS) járó támadások.

4.1. Adatokkal való visszaéléses támadások

Az ilyen jellegű támadások célja érzékeny vagy bármilyen szempontból védett adathoz való illetéktelen hozzáférést jelent. Ez vagy az Interneten távolról, vagy belső hálózatból lokálisan valósulhat meg.

4.1.1. Internetes behatolás

Távolról illetéktelen hozzáférés esetén általában valamilyen backdoort, vagy exploitot kihasználó támadásra gondolunk. A backdoor a normál autentikációt megkerülő, távoli hozzáférést biztosító program, ami saját működését igyekszik álcázni. Az exploitnak nem szándékosan előre telepített programot, hanem egy ártatlannak tűnő programban lévő olyan szoftverhibákat más néven bugokat nevezzük, amelyeket kihasználva nem várt működést idézhetünk elő.

Exploitokból az elmúlt fél évben két igen nagy kockázatúról is beszélhetünk. Az első az OpenSSL hibája, amit HeartBleed-nek kereszteltek el. Ezt kihasználva, a támadó titkosítatlan memóriatartalmakat olvashatott ki a szerverről, mindezt teljesen észrevétlenül. A másik pedig a ShellShock ami a Bash a linux termináljának bugja volt.

Az ilyen típusú támadások ellen a leghatékonyabb védekezés a rendszeres szoftverfrissítés, és egy megbízható (szintén naprakész) tűzfal használata.

4.1.2. Belső támadás

Belső támadástól akkor beszélhetünk, ha a támadó nem az Interneten keresztül támadja a célszervert, hanem egy lokális hálózatban lévő számítógéphez fér hozzá. Többnyire egy munkatárs felhasználónevének és jelszavának megszerzésével kivitelezhető módszer. Ilyenkor távolról, az illetékes számítógépéről, az ő nevében tud a támadó parancsokat futtatni, és ha a birtokba vett gép felhasználója elég magas szintű jogokkal rendelkezik, akkor ettől kezdve már könnyen hozzá tud férni a célszerverekhez.

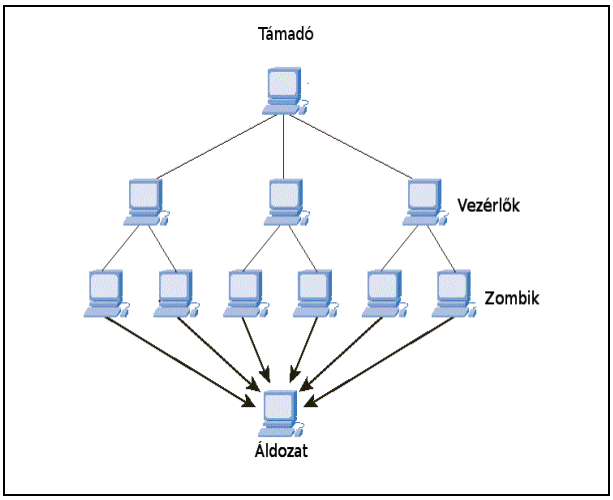
Az utóbbi években egyre komolyabb úgynevezett illetéktelen hálózati behatolást jelző rendszerek (intrusion detection system, IDS) [3][5] léteznek. Az alapműködési elvük, hogy monitorozzák az egyes felhasználók, vagy felhasználócsoportok tevékenységét, és ez alapján felhasználói profilokat építenek fel. Minden tevékenységhez a rendszer kiszámol egy pontszámot, az alapján hogy mennyire ítéli szokatlannak vagy gyanúsnak a felhasználótól ezt a tevékenységet az

adott időben. Ha ez a szám egy bizonyos értéknél magasabb, akkor a műveletről riasztást küld az illetékes embereknek, illetve akár a tranzakció megszakítását vagy megtagadását is megteszi.

4.2. Zombi szerzés

Az egyszerű egy az egy elleni támadások helyett a komolyabb támadóerő, és a nehezebb visszakövethetőség érdekében a támadók gyakran botneteket alakítanak ki. Ezek jellemzője, hogy számtalan gépet foglalnak magukba a világ minden tájáról. Botnettel egyszerűen lehet egyidejűleg sok gépről elosztott támadásokat indítani.

Egy speciális programmal feltérképezhető az Internet, ahol védtelen gépeket kereshet a támadó, majd ezeket fertőzi meg egy rosszindulatú programmal, ezzel alakítja ezeket a gépeket Zombivá. Zombi gépeknek azokat az Internetre csatlakoztatott eszközöket értjük, amiket a hacker, vagy egy trójai vírus irányítása alatt tart, és igény esetén ezekről az eszközökről tud támadást indítani.



2. ábra: a DDoS támadások tipikus felépítése (Forrás:[8])

A kiépített botnetek mérete igényektől függően a több tízezres, vagy akár több milliós is lehet. Ilyen hálózatokkal a támadó leggyakoribb célja az email spammelés, vagy elosztott DoS támadás kivitelezése.

4.2.1. Waterhole

A waterhole támadás a Zombi gépek szerzésének egy módja. A támadó nem közvetlenül gyűjt áldozatokat, hanem egy gyakran használt szolgáltatást térképez fel sebezhetőségi szempontból, és ezeket kihasználva juttat be a rendszerbe olyan kódot, ami megfertőzi a látogatókat. A szolgáltatást igénybe vevő klienseknél ez a kód nem várt működést eredményez, aminek következtében programok települhetnek a gépére, ezáltal a támadó irányítása alá vonható Zombi géppé válik.

4.2.2. Közvetlen hozzáférés

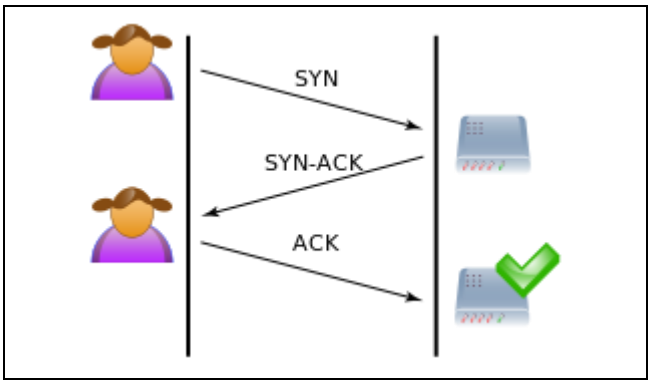
Direct access, vagyis közvetlen hozzáféréses támadásról akkor beszélünk, ha a támadó rendelkezik hozzáféréssel bizonyos gépekhez, és ezekbe bejelentkezve, a megfelelő programokat feltelepítve szervezi be azokat a botnetbe.

4.3. DoS / DDoS támadások

Az elosztott szolgáltatáskieséssel járó támadásokat az 1990-es évek óta tarthatjuk számon. Minden esetben a célgép elérését lehetetleníti el, a gép valamely erőforrásának túlterhelésével. Univerzális hatékony védelem DDoS ellen nem áll rendelkezésünkre, éppen ezért ez a módszer mai napig előszeretettel alkalmazott. Olyannyira, hogy 2014-ben a statisztikák szerint óránként 28 DDoS támadásra derül fény [6].

4.3.1. Syn Flood

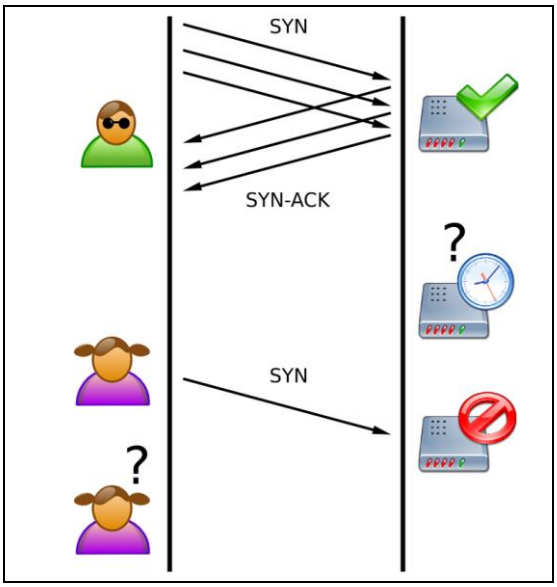
A módszer a TCP kapcsolat kiépülését, egészen pontosan a három-utas kézfogást támadja meg. Egy ilyen kapcsolat, ha minden jól megy, úgy néz ki, hogy a kliens küld a szervernek egy Syn csomagot, amire a szerver egy Syn-ack csomaggal válaszol. Ez jelenti azt, hogy a szerver tudja fogadni a beérkező kapcsolatot. A kliens mikor ez megérkezik hozzá, egy Ack csomaggal nyugtázza ezt, és a kapcsolat ez után épül ki a két szereplő között.



3. ábra: TCP kapcsolat kiépülése normál esetben (Forrás [7])

A szerver a Syn-ack csomagot értelemszerűen arra a címre küldi, ahonnan a Syn-t kapta, ezt az információt pedig onnan tudja meg, hogy mi van a Syn csomag fejlécébe írva. A támadók ezt kihasználva hamis IP címmel töltik ki ezt a részt, a szerver így olyan gépekhez akar csomagot küldeni, akik vagy nem is léteznek, vagy ha léteznek is, nem számítanak Syn-ack-ra és eldobják azt. A szerver egy előre beállított timeout értékig vár mielőtt bármi ilyen félig kiépült kapcsolatot bezárna, ez lehetőséget biztosít a támadónak arra, hogy a szervert ilyen hamis kapcsolódási kérésekkel túltöltse, ezáltal a szabályos kapcsolatok kiépülését jelentősen lelassítsa, vagy akár teljesen ellehetetlenítse.

Syn flood támadás ellen több ismert védekezési módszert használhatunk. [2] Egy bizonyos kapcsolódási szám után tilthatjuk az összes bejövő kérést egy adott routertől, vagy subnetből. Ezt a módszert filterelésnek nevezzük. Csökkenthetjük a timeoutot, ezáltal gyorsabban bezárhatjuk a félig nyitott kapcsolatokat, így megnehezítve a támadó kísérletét ennek túltöltésére. Jól alkalmazható módszer továbbá, hogy ha betelik az összes kapcsolatoknak fenntartott hely másnéven slot, akkor a legrégebb ideje kiépülés alatt lévő kapcsolat slotját használjuk fel, az új kapcsolathoz.



4. ábra: A Syn flood támadás menete (forrás [7])

Természetesen lehetőség van különböző proxy-k, csomagszűrők, vagy tűzfalak használatára akár az előbb említett módszerekkel egyidejűleg is.

4.3.2. Ping of Death

A Ping of Death (PoD) támadás lényege, hogy egy 64 Kbytenál nagyobb ICMP (Internet Control Message Protocol) vagyis hálózati diagnosztikai csomagot készítünk, és ezt küldjük el a célgépnek. Mivel a szabvány szerint IPv4 csomag mérete nem lehet 65535-nél nagyobb, egy ilyen méretű ping csomagot a rendszerek gyakran nem tudnak kezelni. Ez a Windows alapú rendszerekben kék halált, Linuxon kernelpánikot is eredményezhet.

Példa a PoD támadásra:

```
ping 127.0.0.1 -l 65540
```

Természetesen a tűzfalaknak nem kötelező reagálniuk a pingre, sőt ha nem akarunk komoly kockázati rést hagyni a rendszerben, akkor minden a működéshez nem esszenciálisan szükséges forgalmat tiltunk.

4.3.3. HTTP POST támadás

Először 2009-ben fedezték fel ezt a módszert. A támadás lényege, hogy a HTTP POST kérés fejlécében a Content-length vagyis a tartalom hossza mezőt a lehető legnagyobb értékkel (az Apache webszerveren ez alapértelmezettként 2GB) töltjük ki, majd a kérés törzsét rendkívül lassan mondjuk 1byte/sec sebességgel küldjük. A szerver, ha kifejezetten nincs felkészítve a támadás kivédésére, akkor a helyes fejléc és a folyamatosan fennálló kapcsolat eredményeképpen végig fogja várni, amíg a kérés teljes tartalma átér hozzá, ezzel jelentősen lassítva azt.

Célszerű a Content-length paraméter minimalizálása a szerver általános felhasználási igényeit figyelembe véve. Ha kifejezetten cél, hogy nagyméretű tartalmakat tudjunk POST kérésekben

elküldeni, akkor célszerű a kapcsolatok minimális sebességét is meghatározni, és az ezt huzamosabb ideig alulteljesítő kapcsolatokat bontani.

4.3.4. HTTP GET/POST flood

A HTTP GET vagy POST [4] flood is egy elárasztásos támadás, célja a szerver megbénítása számtalan különböző érvényesen összerakott kéréssel. A szerver alap esetben minden kérésre válaszol, attól függően, hogy milyen bonyolultságú feladat a kérés teljesítése, ez akár komoly erőforrás igényű is lehet. A rengeteg kéréssel elárasztott szerveren így minden esetben jelentős forgalmi és erőforrás igényű overhead jelentkezik.

Ahhoz képest, hogy a támadás nem egy bonyolult ötleten alapul, a legnagyobb gond vele, hogy nehezen, vagy egyáltalán nem különböztethető meg a jóindulatú felhasználók tevékenységétől, éppen ezért a védekezést sem erről az oldalról közelítjük meg. Alkalmazás szintű támadások esetén gyakorta alkalmazott módszer az úgynevezett „client puzzle” [1]. A kliens minden alkalommal mikor egy kérést küld a szervernek meg kell, hogy oldjon egy matematikai problémát, ki kell, hogy számoljon egy hash-t, stb. A lényeg hogy a kiszámítás műveletigénye változtatható, és az eredmény könnyen ellenőrizhető legyen.

A szerver ellenőrzi a megoldást, és csak akkor dolgozza fel annak tartalmát, ha az helyes, ellenkező esetben eldobja a kérést. A rosszindulatú felhasználók vagy rászánják az időt és erőforrást és így a másodpercenként elküldhető kérések száma drasztikusan lecsökken náluk, vagy nem törődnek vele. Vagy így vagy úgy dönt a támadó, „client puzzle” használatával az alkalmazásszintű flood támadások ellen jól lehet védekezni, úgy hogy a jóindulatú felhasználóknál sem keletkezik észrevehető lassulás.

4.3.5. Speciális XML-ek

A webes szolgáltatások szinte kivétel nélkül SOAP üzenetekkel kommunikálnak, amik törzsébe tetszőleges XML-ek ágyazhatóak. Ezt a lehetőséget több módon használhatják ki a rosszindulatú felhasználók.

Ennek egy módja a mély XML dokumentum, ilyenkor a támadó célja hogy minél több egymásba ágyazást írjon a fájlba, ezáltal megnövelve az elemzéséhez szükséges memória mennyiségét. Ugyanilyen memória túltöltő támadási típus a „billion laughs” vagyis milliárd nevetés. Itt egy olyan üzenetet kreálunk, amiben úgy csinálunk entitásokat, hogy minden következő entitás legyen az előző 10szer egymás mellé írva. A dokumentumtörzsbe pedig csak az utolsó entitást értékeltetjük ki. Könnyen belátható hogy 10^{10} legalacsonyabb szintű entitás lesz a kiértékelés eredménye, ami könnyen memóriatúlcsordulást eredményezhet.

Példa 5^5 bonyolultságú „billion laughs” típusú XML-re:

```
<?xml version="1.0"?>
  <!DOCTYPE lolz [
    <!ENTITY lol "lol">
    <!ELEMENT lolz (#PCDATA)>
    <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;">
    <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
    <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
    <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
    <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  ]>
  <lolz>&lol5;</lolz>
```

Másik hasonló, bár elvben különböző XML támadási fajta, a WSS (web service standards) specifikáció azt mondja ki, hogy készíthető olyan üzenet, ami egy wsse:Security fej blokkot, de rengeteg ds:Signature blokkot tartalmaz. Ezek a blokkok felelősek a titkosításokért. A szerver, amikor azt látja, hogy van wsse:Security blokk, akkor az összes aláírás hitelességét leellenőrzi, ami egy nyilvános kulcsú kriptográfiai feladat. Ez, mivel számtalan aláírás elemet használhatunk, komoly CPU terheléssel, ennek megfelelően rengeteg idővel járhat.

5. Összefoglalás

Az Interneten, vagy céges hálózatokban elérhető szolgáltatásokat többnyire valamilyen cloud szolgáltatás biztosítja. Ismertettük ezen szolgáltatások különböző egymásra épülő fajtáit, és bemutattuk hogy ezeknek kik a potenciális felhasználói. Fontos gondoskodni az adataink megfelelő védelméről, mert a támadások egyik fajtájának pont ezek megszerzése a célja.

Szó volt a különböző szolgáltatáskieséssel járó támadásokról, amiket már több évtizede sikeresen alkalmaznak a rosszindulatú felhasználók a szolgáltatók ellen. A DDoS támadások rendkívül sokfélék lehetnek, ezért nincs is ellenük egy univerzális védekezési módszer, a teljesség igénye nélkül bemutattuk a DDoS-ok különböző fajtáit, és a hozzájuk tartozó védekezési lehetőségeket részletesen szemléltettük.

Irodalom

1. Suriadi, S.; Stebila, D.; Clark, A.; Hua Liu, "Defending Web Services against Denial of Service Attacks Using Client Puzzles," Web Services (ICWS), 2011 IEEE International Conference on , vol., no., pp.25,32, 4-9 July 2011
2. Suriadi, S.; Clark, A.; Schmidt, D., "Validating Denial of Service Vulnerabilities in Web Services," Network and System Security (NSS), 2010 4th International Conference on , vol., no., pp.175,182, 1-3 Sept. 2010
3. Bakshi, A.; Yogesh, B., "Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machine," Communication Software and Networks, 2010. ICCSN '10. Second International Conference on , vol., no., pp.260,264, 26-28 Feb. 2010
4. Choi, Junho, et al. "A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment." Soft Computing(2014): 1-7.
5. *Wikipedia* intrusion detection systems (2014) http://en.wikipedia.org/wiki/Intrusion_detection_system
6. *Wikipedia* denial of service attack (2014) http://en.wikipedia.org/wiki/Denial-of-service_attack
7. *Wikipedia* Syn flood (2014) http://hu.wikipedia.org/wiki/SYN_flood
8. *Cisco* Distributed Denial of Service Attacks (2014) http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-4/dos_attacks.html