Digitális aláírással erősített DNS szolgáltatás (DNSSEC) eszközkészletek

Szerző:
PÁSZTOR János

Konzulens:
PÁSZTOR Miklós

Pázmány Péter Katolikus Egyetem
Információs technológiai Kar

2010.

Alulírott Pásztor János, a Pázmány Péter Katolikus Egyetem Információs Technológiai Karának hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomamunkában csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a diplomamunkát más szakon még nem nyújtottam be.

Tartalomjegyzék

Ábrák jegyzéke

Összefoglaló

1983-as kitalálása óta a DNS az internet szerves része. A felhasználók számára rejtetten működik, azonban az általuk használt programok változatos módon veszik igénybe a DNS szolgáltatásait: például az e-maileket az MX rekordok segítségével továbbítjuk a hálózaton, a spamszűrésben használjuk az SPF[1] rekordokat, a TXT rekordokban tárolhatjuk a víruskereső programok adatbázisának verzióját.

A dolgozatomban a DNS egy új, fontos biztonsági kiegészítésével, a DNSSEC-cel foglalkoztam.

A feladatmegoldás során Debian GNU/Linux alapú rendszereket üzemeltettem és a `perl` programozási nyelvet használtam.

Első gyakorlati lépésként aláírtam a *ppke.hu* és *itk.ppke.hu* tesztzónákat, majd üzembe helyeztem DNSSEC-et támogató autoritatív és rekurzív névszervereket a zónákban, ezek segítségével vizsgálatokat végeztem a konstrukció fölött. Kipróbáltam két DNSSEC-beli eszközkészletet: az egyik az ISC által készített BIND névszerverhez csomagolt segédprogramok összessége, a másik a RIPE és több ccTLD operátor által fejlesztett OpenDNSSEC Ezek az eszközök nagyban megkönnyítik a DNSSEC bevezetését és fönntartását. Megvizsgáltam ezen eszközkészletek közötti különbségeket, mind az üzemeltető, mind a fejlesztő szempontjából.

Tanulmányoztam a zonewalking effektust, „proof-of-concept" szintű implementációt írtam a jobb megértést elősegítendő.

Méréseket végeztem a DNSSEC által generált hálózati overhead-del kapcsolatban, különös tekintettel a különféle DNSSEC opciók (kulcsméret, NSEC vs. NSEC3 használata) hatását vizsgálva.

[1] **S**ender **P**olicy **F**ramework

Abstract

Since the 1983 invention of the DNS, it's an integral part of the internet. It works hidden from the aspect of the users, but the programs they use, make use of the DNS in diversified ways, for example we deliver the e-mails in the network with the help of the MX records, we use SPF[2] records in the spam filtering programs, in the TXT records we can store the database version of the antivirus programs.

In my thesis I've studied a new, important security extension of the DNS, the DNSSEC.

During the problem solving, I've operated Debian GNU/Linux based systems, and used the `perl` programming language.

As a first practical step, I've signed the *ppke.hu* and *itk.ppke.hu* test zones, after I setup authoritative and recursive name servers, which supports DNSSEC, and made investigations over the construction. I've tried two tools: the first was the utilities packaged to the the BIND nameserver, which has developed by the ISC, the other one was the OpenDNSSEC toolkit, which is developed by RIPE and other ccTLD operators. These tools are greatly helping the introduction and operation of the DNSSEC. I've examined the differences between these tools, from the point of the operator and the developer.

I've studied the zone-walking effect, wrote a „proof-of-concept" level implementation, to help gain deeper understanding.

I've made measures related to the network overhead, which was generated by DNSSEC, especially investigating the effect of the various DNSSEC options (key size, usage of NSEC vs. NSEC3).

[2] **S**ender **P**olicy **F**ramework

1. fejezet

Bevezetés

Az internethasználatban kulcsszerepe van a DNS-nek, hiszen minden alkalmazás használja, amely az interneten egy másik gépet szeretne elérni a neve alapján.

Amikor a DNS működését Paul Mockapetris RFC-kben definiálta (RFC1034[1], RFC1035[2]), nem volt szempont a tervezés során a biztonság. Ekkor az internet a kutatási és tudományos célokat segítette elő, ezekben az időkben az akadémiai körökre ma is jellemző bajtársiasság, önzetlenség, és segítőkészség uralkodott a hálózaton. Az üzleti felhasználás csak később került előtérbe, ennek a folyamatnak a mellékhatásaként jelentek meg a kéretlen reklámlevelek (spam), az adathalászat(phising) és a DNS cache poisoning. Ezek a „vívmányok" súlyosan veszélyeztetik a mai internethasználókat.

Ezen okok miatt hibának tekinthető, hogy a DNS még mindig ebben az állapotban van. Több olyan esetet is tudunk, amikor a meglévő hibákat kihasználva kéretlen reklámokat kapunk. Erre egy példa Verisign esete volt 2003-ban, aki visszaélt a helyzetével, és minden nemlétező .com vagy .net domain-t átirányított egy keresőoldalra, ezáltal nem lehetett tudni, hogy az a domain valójában nem létezik. Ezt a wildcard rekordot a közfelháborodás után törölték.

A DNSSEC a meglévő gyengeségeket próbálja kijavítani azáltal, hogy nyilvános kulcsú titkosítással írja alá a DNS rekordokat: így tudunk információt szerezni arról hogy a válaszként kapott DNS információ hiteles és sértetlen.

Ennek a megvalósításához a meglévő DNS protokollban több változtatást is kell alkalmazni, a meglévő maximális üzenethosszat megnövelni(EDNS0), mert 512 byte nem elég az új flagekhez és Resource Rekordokhoz. Emiatt új fizikai eszközökre, firmware frissítésekre is szükség lehet, mert nem mindegyik tudja ezt fejlesztést megfelelően kezelni.

2. fejezet

Feladatkiírás

Helyezzen üzembe NSEC3-al működő DNS szolgáltatást OpenDNSSEC eszközkészlet alapon.

Állítson fel egy kísérleti szolgáltatást például a ppke.hu domain DNSSEC-cel védett szolgáltatására. Hasonlítsa össze a régebben használt technológiával készült megoldást ezzel. Vizsgálja meg, hogy milyen előnyei vannak a szolgáltatás és üzemeltetés szempontjából az OpenDNSSEC-nek a régebben kipróbált eszközökhöz képest.

Foglalja össza, hogy milyen előnyei és kockázatai vannak általában a DNSSEC-nek a felhasználók és az üzemeltetők szempontjából.

3. fejezet

Előzmények

A DNS-ben már 1990-ben komoly biztonsági hibákat talált Steve Bellovin[3], ezzel egyidőben kutatások is indultak a DNS biztonságosabbá tételére. Az első DNSSEC-cel kapcsolatos RFC (RFC2065) 1997-ben jelent meg, majd ezt követte az RFC2535 1999-ben, ami már a teljes specifikációt tartalmazta, valamint terveket a DNSSEC bevezetésére.

Azonban ennek a megoldásnak komoly hibája volt: nem skálázódott jól az internet egészére, ezért ebben a formában nem lehetett bevezetni. A probléma abban rejlett, hogy az aláírásnál nagy mennyiségű adatot kellett az apa zónának elküldeni, amit ő aláírt és visszaküldött a delegált zónának, ez került volna be a SIG rekordba. A kulcsváltásnál hasonló problémák léptek föl, például ha a *.com* kulcsot cserélt akkor 80 millió rekordot kellett volna neki fölküldeni, amit ő aláírva visszaküldött. Ebből látszik, hogy ebben a megoldásban tervezési hibák találhatóak, így nem vezethető be.

Ekkor az IETF alapvető változásokat hajtott végre a protokollon, kitalálta a DS rekordot, ami nagyban megkönnyítette az új aláírásokat és a kulcsok cseréjét. A DS rekord segítségével kulcsváltáskor elég csak a nyilvános kulcsot eljutattni a delegáló zónába, ebből hash-t képez, ez kerül bele a zónába, így jön létre a kapcsolat a két zóna között. A protokollt innentől DNSSEC-bis-nek hívtuk, ezzel különböztetve meg a „klasszikus" DNSSEC-től. A dolgozatban mi már csak ezzel foglalkozunk, és ezt hívjuk DNSSEC-nek.

A DNSSEC bevezetése elkezdődött, az *.se, .br, .cz*, és a *.org* TLD-k már alá vannak írva, ebből csak a *.org* használ NSEC3-at, a többiek NSEC-et használnak. A root zónában is elkezdődött a bevezetés 2009. december 1-el, inkrementális módon, sorban vezették be a DNSSEC rekordokat a 13 root névszerverbe. Ezalatt az idő alatt alatt DURZ [1] kulcsok voltak a zónában, ami megakadályozta az éles használatot. A folyamat 2010. július 15-vel ért véget, ekkor az IANA aláírta az „éles" root zónát és a trust anchor-okat elérhetővé tette [4].

[1] **D**eliberately **U**nvalidatable **R**oot **Z**one

4. fejezet

A tervezés részletes leírása

4.1. A Kaminsky bug

4.1.1. A „klasszikus” cache poisoning

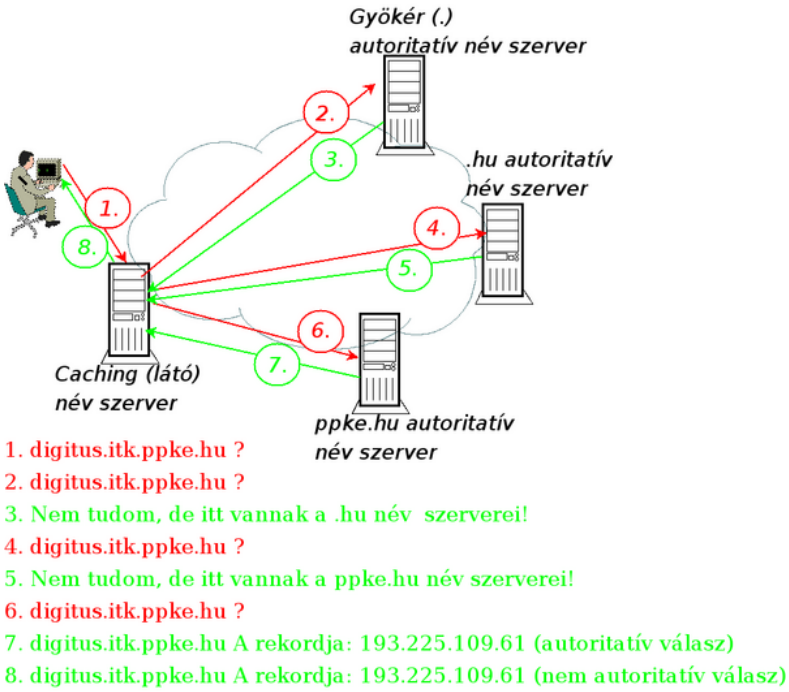
A cache poisoning megértéséhez célszerű fölidézni egyes DNS-beli alapismereteket: a rekurzív föloldást és a DNS csomagformátumát.

A rekurzív föloldás folyamatát a 4.1 képen tekinthetjük meg. Az ábrán egy, a mindennapokban gyakran előforduló kérdés jelenik meg, szeretnénk tudni a digitus.itk.ppke.hu géphez tartozó IP címet. Hogy ezt megtudhassuk, a gépünkbe bekonfigurált rekurzív névszerverek egyikéhez folyamodunk. A folyamat itt két irányba ágazik el: ha a szerver már tudja a választ (benne van a cache-ben), akkor onnan válaszol, ha nincs benne, akkor a root névszerverek egyikéhez fordul. A root névszerver egy referral segítségével továbbirányít minket a *.hu* névszervereihez, akik szintén továbbirányítanak minket, addig amíg el nem érünk az *itk.ppke.hu* domain egy autoritatív névszerveréhez, aki megválaszolja nekünk a kérést. A mi rekurzív névszerverünk elteszi ezt a cache-be, így következőleg már nem kell ezt az utat bejárnia.

Miután fölidéztük a föloldás menetét, vizsgáljuk meg a DNS kérésekben és válaszokban előforduló csomagokat. A kérdés és válasz mindössze egy bitben tér el egymástól, ami ténylegesen megmondja hogy az adott csomag kérdés vagy válasz-e. A 4.2. képen látható[5]. A sárgával jelölt mezők az igazán fontosak nekünk, mivel ezeknek a mezőknek a segítségével valósulhat meg a cache poisoning.

A támadás sikeres kivitelezéséhez a következő követelményeknek kell megfelelnünk:

- Arra a portra küldjük a választ, ahonnan a kérés indult.
- A Question szekció másolata a kérésben szereplő Question szekciónak.
- A Query ID egyezik.
- Az Additional és Authority szekcióban a kérdéses domain-ből származó rekordok szerepelnek.
- A válasz nem szerepel a névszerver cache-ében.



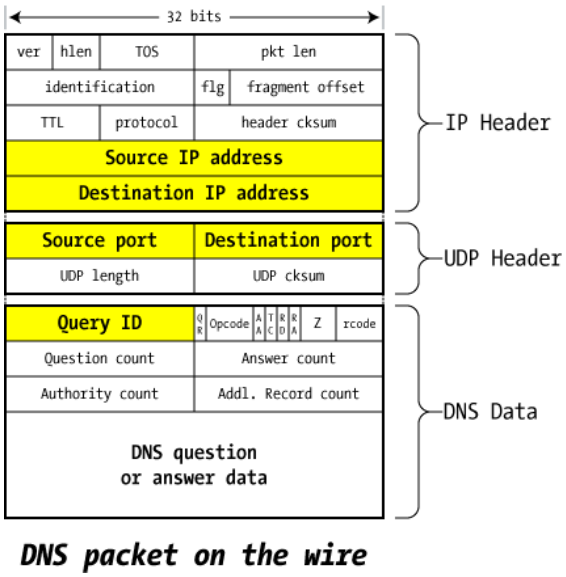
4.1. ábra. Rekurzív föloldás a DNS-ben

A 4.1. ábrán a támadást a 7-ik válasz pozíciójában kell elképzelni, ott egy 7.a. választ érkezik, ez valósítja meg a cache poisoning-et.

Mivel mind a Query ID mind az UDP port egy 16 bites szám, ezért esélyünk van arra hogy mindkettőt eltaláljuk, így TTL időre hamis adatot csempészünk a cache-be. Ez egészen egyszerű feladat, annak a fényében, hogy a régebbi implementációk mind a Query ID-t, mind a portot egyesével növelték a kiküldött kérdések után.

Ha a rekurzív szerverünk nyílt rekurzív szerver (ami konfigurációs hibának minősül), akkor a feladat tovább egyszerűsödik, a támadó kérdést intéz a szerver felé, ami egy általa kontrollált domain-ra vonatkozik, ebből jó becslést bír készíteni a portra és a Query ID-ra. Továbbá a támadó nem csak egyszer próbálkozhat, hanem elárasztható minket csomagokkal, és elég ha csak egy ér célba a helyes válasz előtt, a cache poisoning megvalósult.

Ezek fényében két megoldás mutatkozott eddig a probléma megoldására: a Query ID legyen véletlenszerű, valamint a TTL értékét növeljük nagyra, például egy napra. Így a támadó csak napjában egyszer próbálkozhat, és annak kicsi az esélye hogy abban az időintervallumban eltalálja a Query ID-t és a portot is. Ezen megoldások azonban Kaminsky fölfedezése után haszontalannak bizonyultak.



4.2. ábra. A DNS és az alatta elhelyezkedő protokollok csomagformátuma

4.1.2. A Kaminsky-féle cache poisoning

Ahhoz hogy jól megértsük az „előrelépést” a cache poisoning témakörben, tüzetes vizsgálat alá vetjük magát a DNS csomagot. Példánkban ismételten a digitus.itk.ppke.hu fog szerepelni, itt azonban egy való életből vett lekérdezésen mutatjuk be a problémát.

```
pasja@sechu:~$ drill digitus.itk.ppke.hu
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 55343
;; flags: qr rd ra ; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;; digitus.itk.ppke.hu. IN      A

;; ANSWER SECTION:
digitus.itk.ppke.hu.    86400   IN      A       193.225.109.61

;; AUTHORITY SECTION:
itk.ppke.hu.    214424  IN      NS      ns.ppke.hu.
itk.ppke.hu.    214424  IN      NS      apa.btk.ppke.hu.

;; ADDITIONAL SECTION:
ns.ppke.hu.     38024   IN      A       193.225.195.195
apa.btk.ppke.hu.        41624   IN      A       193.225.194.100

;; Query time: 2 msec
;; SERVER: 193.239.149.6
;; WHEN: Sat Dec  4 21:29:11 2010
;; MSG SIZE  rcvd: 124
```

4.3. ábra. A digitus.itk.ppke.hu-ra vonakozó lekérdezés

A számunkra érdekes dolgok most nem az Answer szekcióban vannak, hanem az Additional és Authority részben. Amint láthatjuk, kérdésünkkel nem csak a választ, hanem információkat is megtudtunk az adott zónáról: az alatta szereplő NS rekordok listáját és a hozzájuk tartozó A rekordokat. Ez segítséget nyújt a rezolvernek, ha a közeljövőben ugyanebből a domain-ból kérdezünk rá valamire, akkor nem kell az egész fát bejárnia a válaszhoz.

A probléma is itt rejtőzik: míg az Authority szekcióra alkalmazni tudjuk a „bailiwick

checking"-et, az Additional szekcióban szereplő glue rekordok helyességéről semmit nem tudunk mondani!

Egy tipikus támadás a következő sémát követi:

- Elárasztjuk kérdésekkel a rekurzív névszervert az elkötendő zónából, olyanokkal amik biztosan nem szerepelnek a cache-ben. A fönti példánál maradva pl.: www1231234.digitus.itk.ppke.hu
- Mi válaszolunk a kérdésekre, ahol az Additional szekcióban elhitetjük, hogy a miáltalunk megadott IP-n üzemel a digitus.itk.ppke.hu autoritatív névszervere.
- Ha ez sikerült, akkor megszereztük az egész zónát, nem csak a kérdésben szereplő bejegyzést!

„Fejlődés" az is az előző támadáshoz képest, hogy míg ott csak a legutolsó lépcsőben tudtunk bekapcsolódni, itt gyakorlatilag tetszőleges helyen be bírjuk csapni a kérdezőt. Elköthetjük pl.: a *.hu* zónát, az összes kérdésre a választ egy DNS proxyból szolgáltatjuk, kivéve az általunk támadni kívánt zónát, ahol a gyanútlan netezőt például egy, az eredetire megtévesztésig hasonlító, adathalász oldal várja. Mivel nem csak egy rekordot kötöttünk el, így a DNS teljes tárháza a rendelkezésünkre áll: eltéríthetjük a leveleit hamis MX rekordokkal, elérhetetlenné tehetjük a szolgáltatásokat, etc, ...

Ideiglenes megoldást az UDP port véletlen kiválasztása jelenti, ezzel milliós nagyságrendre növeltük a rendelkezésre álló teret, ahonnan a Query ID, UDP port párost kiválasztjuk. Az előző probléma alapján azonban ez is csak átmeneti megoldás, a számítási kapacitás növekedésével ez a mennyiség is kevésnek bizonyulhat.

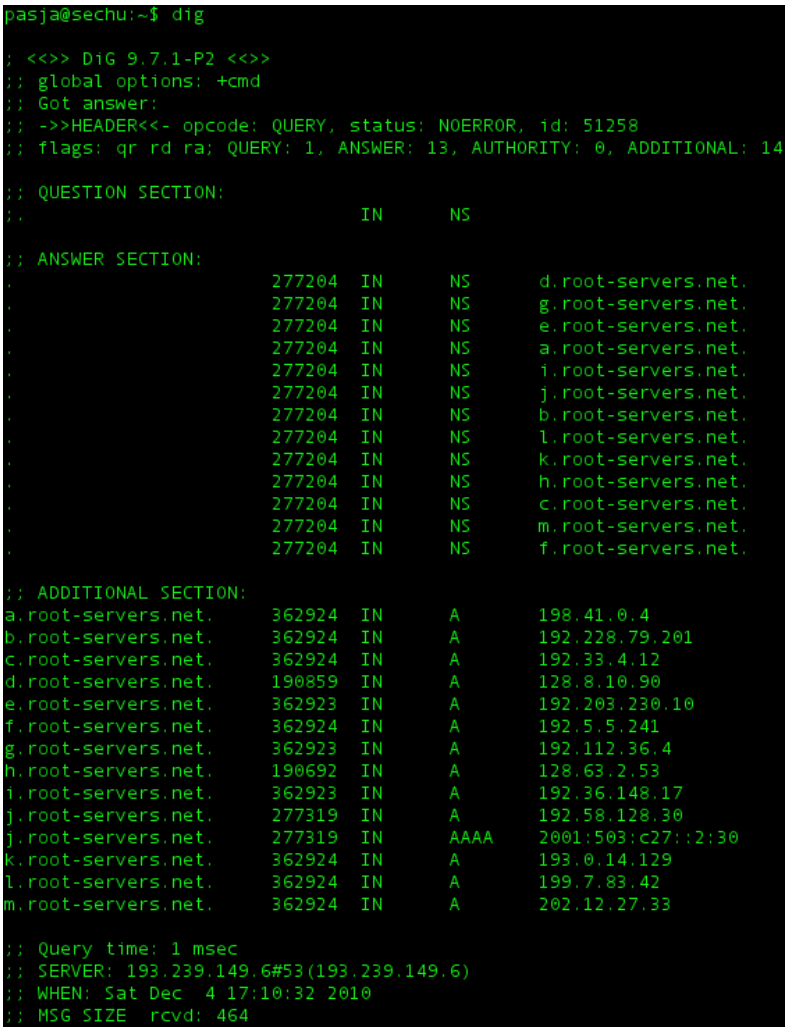
Az egyik lehetséges megoldás a cache poisoning kiküszöbölésére a DNSSEC bevezetése.

4.2. Elméleti áttekintés

4.2.1. Új Resource Rekordok

A DNSSEC négy új resource rekordot vezet be az adatbázisba: Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS), Next Secure (NSEC). Ezen felül bevezet 3 új bitet a headerben is. Ezek a Checking Disabled(CD), Authenticated Data (AD) és a DNSSEC OK (DO) . Végül a DNSSEC-hez szükséges hogy a szerver és a kliens is tudja kezelni az EDNS0 protokollkiterjesztést a megnövekedett üzenethossz miatt.

Mielőtt elkezdjük elemezni ezeket az új RR-eket, tisztázni kell az RRset fogalmát[6], mert ez az egyik elemi egység a DNSSEC-ben. Azoknak az RR-eknek az összeségét hívjuk RRset-nek, amelyekben azonos a label-class-type hármas. Erre jó példa a UNIX-os `dig` parancs alapértelmezett kimenetével visszakapott root szerverek NS rekordjai, amit a 4.5. képen láthatunk. Ha kérés érkezik egy adott class-ra, label-re vagy type-ra, akkor az egész RRset-et kell visszaadni, ha nem fér bele egy csomagba, akkor a truncated bitet be kell állítani, amire a 4.4. kép mutat egy példát.



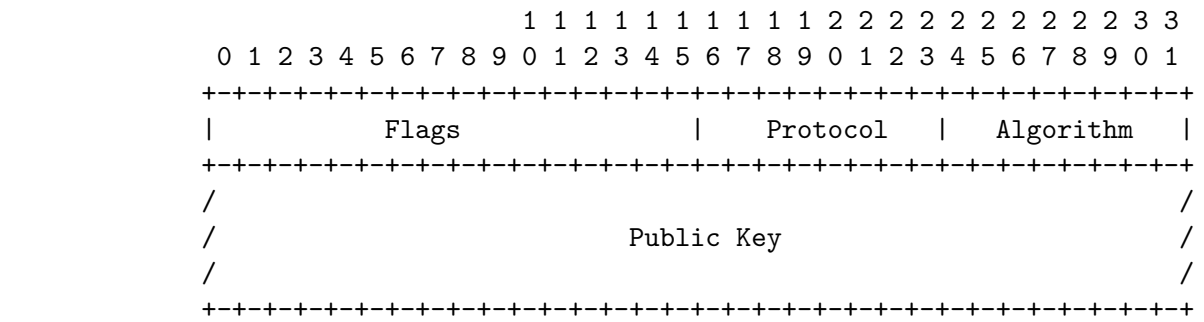
4.4. ábra. A dig parancs alapértelmezett kimenete

Az alábbi rekordelemzés erősen támaszkodik az RFC4034-ben[7] és az RFC5155-ben[8] leírtakra, a dolgozat megértéséhez fontos ezeket az alapvető fogalmakat átvenni.

```
pasja@sechu:~$ drill -D dnskey se -b 512
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 38271
;; flags: qr tc rd ra ad ; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;; se.  IN      DNSKEY

;; ANSWER SECTION:

;; AUTHORITY SECTION:

;; ADDITIONAL SECTION:

;; Query time: 0 msec
;; EDNS: version 0; flags: do ; udp: 4096
;; SERVER: 193.239.149.6
;; WHEN: Sat Dec  4 17:01:47 2010
;; MSG SIZE  rcvd: 31

;; WARNING: The answer packet was truncated; you might want to
;; query again with TCP (-t argument), or EDNS0 (-b for buffer size)
```

4.5. ábra. Truncated válasz a .se zónából

A DNSKEY rekord



4.6. ábra. A DNSKEY rekord csomagformátuma

Flags: Két értéket használunk belőle, a 7-ik bitet, ami jelzi hogy DNSSEC-ben használatos kulcsot tartalmaz a rekord, valamint a 15-ik bitet (SEP flag), ami jelzi, hogy a kulcsunk KSK vagy ZSK.

Protocol: A protokoll mezőnek 3-as értéket kell fölvennie konstrukció szerint, ha más értéket vesz föl, akkor nem vesszük figyelembe ezt a rekordot.

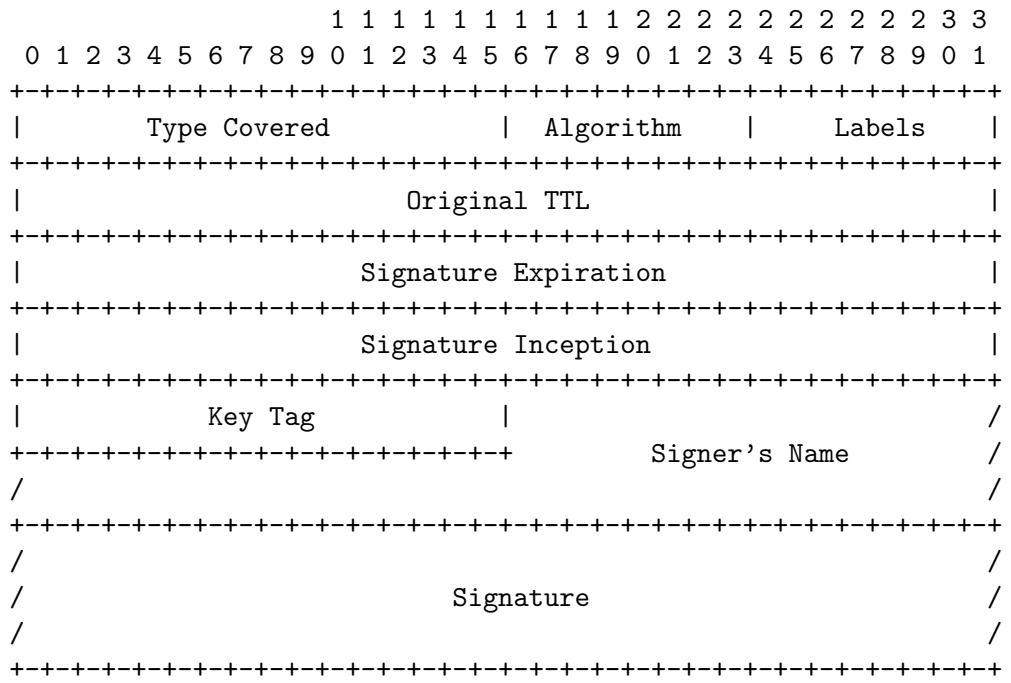
Algorithm: Ez jelzi, hogy milyen algoritmussal generáltuk a kulcsot, ezért csak fix értékeket vehet föl. A kurrens értékeket az aktuális RFC-kben találhatjuk meg.

Public Key: Itt tárolódik maga a kulcs Base64 kódolással

Az RRSIG rekord

Type Covered: Ez a mező mondja meg, hogy milyen RRset-et írunk alá ezzel az aláírással.

Algorithm: Megadja, hogy milyen algoritmussal generáltuk az aláírást.



4.7. ábra. Az RRSIG rekord csomagformátuma

Labels: Ez a mező megmondja, hogy hány címkéből áll az eredeti RRSIG birtokosának a neve, és hogy van-e benne joker karakter. Pl: www.example.com, a label mező értéke 3, *.example.com, a label mező értéke 2. Ez azért kell, hogy az ellenőrzés során helyre tudjuk állítani az eredeti birtokos nevét.

Original TTL: Megmondja az eredeti TTL-jét az aláírt RRset-nek.

Signature interception és expiration: Ez a két mező adja meg, hogy az aláírás mettől meddig érvényes.

Key tag: Ez tartalmazza a key tag értéket, aminek segítségével hatékonyan tudunk publikus kulcsot választani az aláírás ellenőrzéséhez.

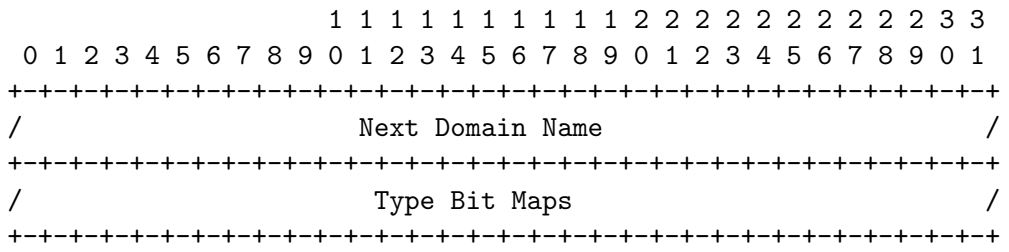
Signer's Name: Ez a mező tartalmazza annak a nevét, aki aláírta ezt a rekordot, ővele kell hitelesíttetni az aláírást.

Signature: Itt található maga az aláírás Base64 kódolással.

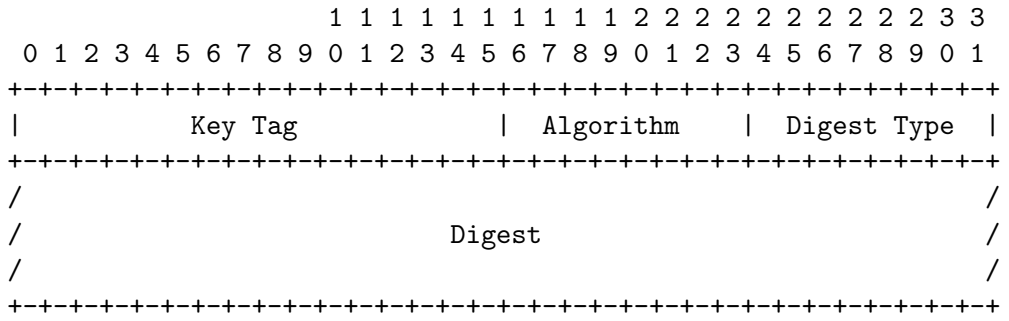
Az NSEC rekord

Next Domain: Ez a mező tartalmazza a kanonikusan következő autoritatív nevet a zónában.

Type Bit Maps: Ez a mező azt tartalmazza, hogy ezen a címen milyen RR-ek találhatóak (A, MX, RRSIG, NSEC . . .)



4.8. ábra. Az NSEC rekord csomagformátuma



4.9. ábra. A DS rekord csomagformátuma

A DS rekord

Key Tag: Ez a mező tartalmazza a key tag-et, amivel hatékonyan tudjuk a DS rekordhoz tartózó DNSKEY rekordot kiválasztani.

Algorithm: Megadja, hogy milyen algoritmussal generáltuk a DNSKEY rekordot ami ehhez a DS rekordhoz kapcsolódik.

Digest Type: Megadja hogy milyen algoritmussal generáltuk a digest mező tartalmát

Digest: Ezt a mezőt a következőféleképpen generáljuk:

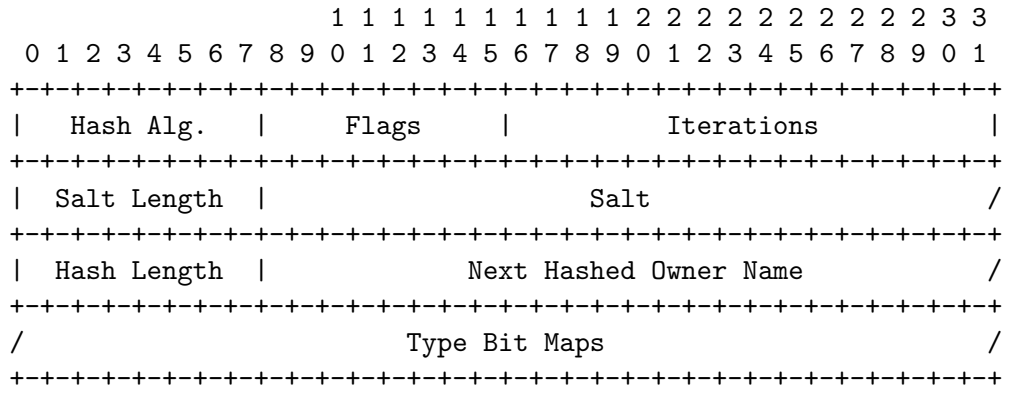
```
digest = digest_algorithm (DNSKEY owner name | DNSKEY RDATA);
```

ahol a | konkatenációt jelöl.

Az NSEC3 rekord

Hash Algorithm: Ez az érték mondja meg, hogy milyen algoritmust használtunk az NSEC3 hash előállításához. A kurrens értékeket az aktuális RFC-kben találhatjuk meg.

Flags: Jelenleg mind 0, kivéve az Opt-Out flaget ami, az LSB helyen található, az lehet 1 is. Ha az Opt-out flag be van billentve, akkor az NSEC3 rekord 0 vagy több aláíratlan delegálást „kihagy” az NSEC3 láncból (nem mond semmit a létezésükről).



4.10. ábra. A NSEC3 rekord csomagformátuma

Iterations: Ez a mező határozza meg, hogy a hash-elést hányszor végezzük el. A magasabb érték nagyobb biztonságot eredményez, cserébe nagyobb terhet ró a processzorra.

Salt Length: A salt mező hosszát mondja meg.

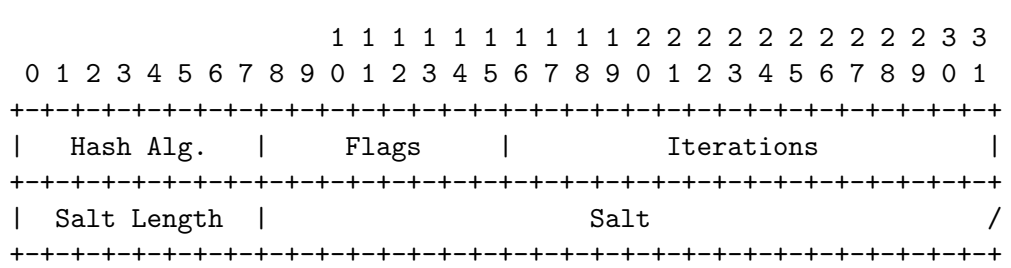
Salt: Ez a mező tartalmazza a salt-ot, amit a hash-elés során használunk.

Hash Length: Ez a mező tartalmazza a Next Hashed Owner Name hosszát.

Next Hashed Owner Name: Ez a mező tartalmazza a kanonikusan következő autoritatív név hash-ét a zónában.

Type Bit Maps: Ez a mező azt tartalmazza, hogy ezen a címen milyen RR-ek találhatóak (A, MX, RRSIG, NSEC, ...)

Az NSEC3PARAM rekord



4.11. ábra. A NSEC3PARAM rekord csomagformátuma

Hash Algorithm: Ez az érték mondja meg, hogy milyen algoritmust használtunk az NSEC3 hash előállításához. Az értékei ugyanazon halmazból kerülnek ki mint az NSEC3 azonos nevű mezőjének értékei.

Flags: Jelenleg nincs használatban, csupa 0 az értéke.

Iterations: Ez a mező határozza meg, hogy a hash-elést hányszor végezzük el. Az értékei ugyanazon halmazból kerülnek ki mint az NSEC3 azonos nevű mezőjének értékei.

Salt Length: A salt mező hosszát mondja meg.

Salt: Ez a mező tartalmazza a salt-ot, amit a hash-elés során használunk.

4.2.2. A DNSSEC konstrukció által nyújtott szolgáltatások

A DNSSEC biztosítja az adatok hitelességét és sértetlenségét nyilvános kulcsú aláírások segítségével.

Ezek az aláírások az RRSIG rekordban vannak eltárolva. A kulcs, amivel aláírunk, a zónához tartozik és nem annak az autoritatív névszerveréhez. A nyilvános kulcs az DNSKEY rekordban van eltárolva, innen tudja meg aki ellenőrizni akarja az aláírás hitelességét. A hozzátartozó privát kulcsot célszerű offline tartani. A publikus kulcsot is ellenőrizni kell, hogy hiteles-e, ezt úgy tehetjük meg, hogy föllépünk egyet a hierarchiában, és ott kérdezük meg, hogy hiteles-e a kulcs. Ahhoz hogy ez a hitelesítési lánc fölépüljön, kell lenni legalább egy trust anchor-nak, ahonnan ez el tud indulni.

A DS rekord adminisztratív célokat szolgál, megkönnyíti a delegálásokkal kapcsolatos esetek lekezelését. A delegálások között is van hitelesítés, ami a következőképpen néz ki: DNSKEY $\rightarrow$ [DS $\rightarrow$ DNSKEY]* $\rightarrow$ RRset, ahol a * iterációt jelöl. Az ennél komplexebb beágyazásokat nem lehet használni. Ezt a láncot alapértelmezetten a root-tól indítjuk, de lokális szabályok fölülírhatják. Ezen felül nem csak azt kell tudnunk hitelesen mondani, hogy ő tényleg ő, hanem azt is, ha ilyen név nincs a hálózaton. Erre szolgál az NSEC vagy NSEC3 rekord, hogy hitelesen tudjuk a negatív választ.

A hitelesítés négyféle válasszal térhet vissza:

Secure: A kliensnek van trust anchor-ja, fölépült a hitelesítési lánc, és minden aláírást ellenőrizni tud.

Insecure: A kliensnek van trust anchor-ja, fölépült a hitelesítési lánc és tudja aláírással bizonyítani, hogy a fa ezen ágához nincs DS rekord, vagyis megszakad a bizalmi lánc, tehát ez a részfa nincs DNSSEC-cel védve. Lokális szabályokkal is szabályozhatjuk, hogy mit tekintünk biztonságosnak és mit nem.

Bogus: A kliensnek van trust anchor-ja, de nem tudja a láncban az aláírásokat hitelesíteni. Ennek több oka lehet: nem támogatott algoritmus, hiányzik az aláírás, nincs aláírás, lejárt az aláírás érvényessége, etc...

Indeterminate: A kliensnek nincs trust anchor-ja a DNS fa azon ága fölött, ahol a szóban forgó név van.

4.2.3. Az új Resource Rekordok fölvétele a zónába, a zóna aláírása

Az új Resource Rekordok bevezetését az RFC4035[9] alapján mutatjuk be.

A DNSKEY fölvétele

A zóna adminisztrátora az aláíráshoz először generál egy vagy több kulcspárt. Azokat a publikus kulcsokat kell bevenni a DNSKEY RRset-be amelyek titkos kulcsával aláírtuk a többi RR-t. Ezeknek a kulcsoknak a zone key flag-jét be kell állítani, ez jelzi, hogy zónát írunk alá velük. Más publikus kulcsokat is tárolhatunk itt, pl a KSK-t, ezeket a zone key flag segítségével különböztetjük meg egymástól.

Az RRSIG fölvétele

Minden autoritatív RRset-hez kell tartoznia minimum egy RRSIG-nek ami teljesíti az alábbi föltételeket:

- Az RRSIG owner name mező egyenlő az RRset gazdájának nevével
- Az RRSIG class mező egyenlő az RRset osztályával
- Az RRSIG type covered mező egyenlő az RRset típusával
- Az RRSIG original TTL mező egyenlő az RRset TTL-jével
- Az RRSIG RR TTL-je egyenlő az RRset TTL-jével
- Az RRSIG label mező egyenlő az RRset gazdájának nevében lévő címkék számával, nem számolva a root címkét, és baloldali címkét, ha az joker karakter
- Az RRSIG signer's name mező egyenlő az annak a zónának a nevével, ami tartalmazza az RRset-et
- Az RRSIG algorithm, key tag és signer's name mezők azonosítják a zóna kulcsát jelentő DNSKEY rekordot a gyerek oldalon.

Az RRSIG rekordok a többi rekordtól eltérően nem formálnak RRset-ket, mert szoros kapcsolatban vannak az általuk aláírt rekordokkal. Egy RRset-nek lehet több RRSIG rekordja is. Egy RRSIG-et tilos aláírni, mert ez végtelen aláírási ciklushoz vezetne.

Az NSEC fölvétele

Minden névhez a zónában, ami autoritatív adatot szolgáltat, vagy delegálási pont, tartozni kell egy NSEC rekordnak, ami a következő létező névre mutat a zónában. A delegálási pontoknál nagy figyelmet kell fordítani az NSEC rekord bitmap-jára, csak azokat szabad beállítani, amire az apuka zóna autoritatív, a gyerek zóna részét üresen kell hagyni.

A DS fölvétele

A DS rekord állítja föl a kapcsolatot az aláírt apa és gyerek zóna között. Ezek a rekordok csak az apa oldalán jelennek meg. Ennek a rekordnak egy gyerek oldali DNSKEY RR-re kell mutatnia, a gyerek oldali DNSKEY RRset-et pedig az RR-hez tartozó titkos kulccsal

kell aláírni. Ha ezeket a kritériumokat nem teljesíti, akkor nem használható hitelesítésre, de ideiglenes eltérések előfordulhatnak a DNS nem midig konzisztens volta miatt.

Az RRset TTL-je megegyezik a delegáló NS rekord TTL-jével.

Hogy a DS-t meg tudjuk konstruálni, ahhoz szükséges a gyerek oldalon lévő DNSKEY rekord ismerete, ami megköveteli, hogy ezek a zónák kommunikáljanak egymással.

4.2.4. A válaszok hitelesítése

Két példán keresztül mutatjuk be, hogy ha a fönti feltételek teljesülnek, akkor hogyan történik a válaszok hitelesítése.

Egy általános válasz hitelesítése

Egy általános válasz hitelesítéséhez szükség van az aláírásához tartozó DNSKEY rekordhoz. Ezt az információt, hogy melyik DNSKEY rekord tartozik az RRSIG rekordhoz, magából az aláírásból nyerhetjük ki, ennek a rekordnak az algorithm és key tag field-je segítségével. Ha több egyező DNSKEY rekord is van, akkor minddel végig kell ellenőrizni, és ha bármelyikkel sikerül az ellenőrzés, akkor hitelesnek tekintjük az adott RR-t.

A DNSKEY rekord hitelesítése

Ebben az esetben egy lépést mutatunk be a fában, a többit ez alapján lehet rekurzívan elvégezni. A kliensnek, aki ezt a rekordot hitelesíteni akarja, legalább egy bekonfigurált trust anchor-ral, (ami egy DNSKEY vagy DS rekord) kell rendelkeznie. Amint ez megvan, meg kell nézni, hogy ez a rekord szerepel-e a delegáló zóna a DNSKEY RRset-jében és ezzel hitelesíteni tudjuk-e a delegáló zóna DNSKEY RRset-jét. Ha ez sikerült, akkor a DNSKEY RRset-et hitelesnek mondjuk. Ezekkel a kulcsokkal hitelesítjük a DS RRset-et ami a gyerek zónához tartozik. Ha ez sikeresen megtörtént, akkor megkeressük azt a DNSKEY-t a gyerek zónában, amire ez a DS rekord mutat. Amint ezt megtaláltuk, ellenőrizzük az aláírás érvényességét, és azt hogy ezzel a kulccsal alá van írva a DNSKEY RRset. Ha ez érvényes és aláírt, akkor a gyerek zóna DNSKEY RRset-jét érvényesnek tekintjük. Ezután az előző pontban leírtak alapján tudjuk ellenőrizni a hitelességét a gyerek zónában szereplő adatoknak a DNSKEY RRset segítségével. A 4.12. képen egy példát látunk erre a folyamatra, a svéd zóna DNSKEY RRset-jét ellenőrizzük, a trust anchor a root zóna KSK-ja.

4.12. ábra. A svéd DNSKEY RRset ellenőrzése

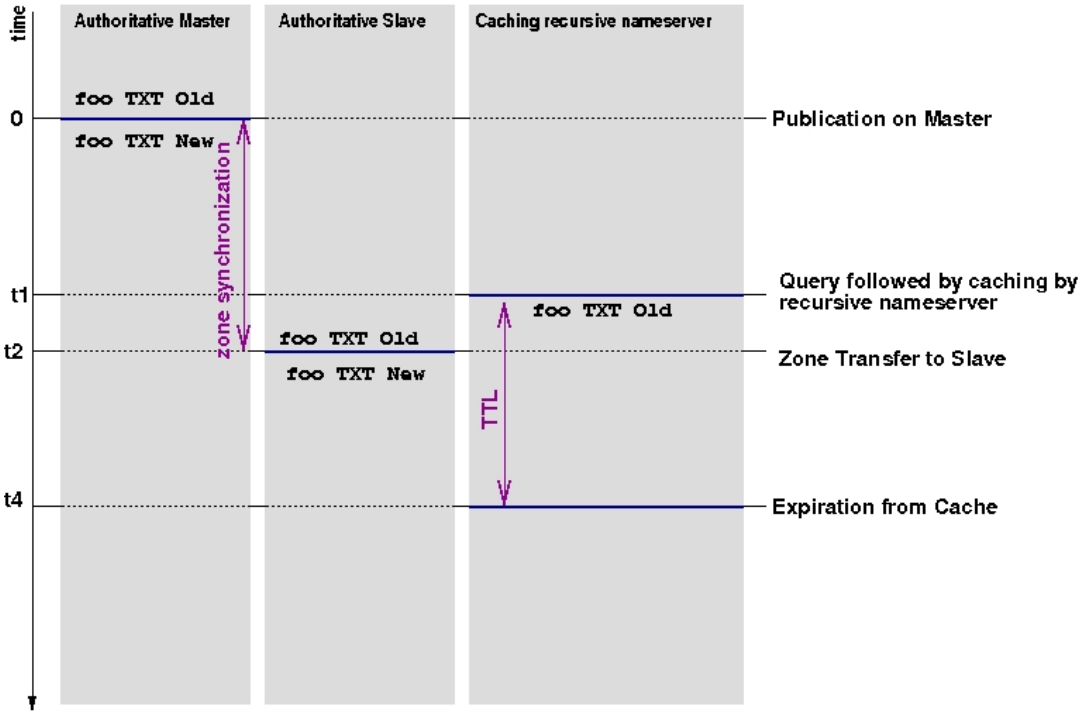
4.2.5. Key-rollover

Bevezetés

A key-rollover[10] (kulcs-csere) egy olyan folyamat, amikor a régi kulcsainkat újra cseréljük le. A DNSSEC-ben a kulcsok élettartamának nincsenek technikai korlátai, azonban célszerű őket rendszeresen cserélni, mert ahogy telik az idő, egyre nagyobb az esély hogy a kulcs titkos része nyilvánosságra kerül, pl.: óvatlanságból vagy emberi mulasztásból. Több típusú key-rollover történhet. Az egyik fajta mikor vagy a KSK-t vagy a ZSK-t cseréljük le. Megkülönböztetjük aszerint is, hogy ez előre eltervezett csere volt, avagy a titkos kulcsunk nyilvánosságra került és amiatt vált szükségessé a csere.

A bizalmi lánc egy pillanatra sem törhet meg, ezért a rollover-t különös figyelemmel kell végrehajtani. Ügyelni kell az adatoknak a DNS infrastruktúrában történő propagálására, ami hagyományos esetben szinte észrevehetetlen, itt nagy jelentőséggel bír. Az 4.13. ábra[11] egy olyan esetet mutat be, amikor a változás még nem terjed el a caching szerverig, ő még a cache-ből válaszol, mialatt az autoritatív szerverben már más

adat szerepel.



4.13. ábra. Az adatok frissülése a DNS-ben

A rollover-t többféleképpen is végre lehet hajtani. Mi két esetet vizsgálunk meg, a pre-published és a double-signature módszert. Általában az első módszert a ZSK, a második módszert a KSK cseréjénél szoktuk használni.

Pre-Published key-rollover

Ebben az esetben a zóna aláírásakor nem egy, hanem két kulcsot használunk, egy aktívat, amivel az aláírás történik, és egy passzívat, amit csak publikálunk, de nem írunk vele alá. Ezzel a zónánkat előkészítettük a key-rollover-re. A SOA rekord serial number-jét minden lépésben növelni kell, különben a változtatások nem fognak elterjedni. A módszer gyengesége, hogy a bevetendő új kulcs már ismert, lehet adatokat gyűjteni róla. Ez egyben előnye is, mert így a bizalmi lánc nem törik meg.

- **Első fázis:** Ebben a fázisban megcseréljük az aktív és a passzív kulcsot és így aláírjuk a zónát. A következő lépés nagyon fontos - minden esetben - meg kell várni, míg a változtatás végigterjed a DNS-ben. Erre egy jó felső korlát az aláírás élettartama és a TTL maximuma, ugyanis ha ezek egyike lejár, akkor a rekurzív szervernek az autoritatív szerverhez kell fordulnia, ekkor frissül a cache a rekurzív szerverben, vagy kiesik az adat.

- **Második fázis:** Ebben a fázisban a zónánkhoz új passzív kulcspárt kell generálni, és ezt bevezetni a zónába. A bevezetés után a zónát újra alá kell írni. A régi passzív kulcspárt ekkor el lehet távolítani, de érdemes biztonsági másolatot készíteni róla.

Double-Signature key-rollover

Ennél a módszernél nincsen előre publikált kulcs, így nem lehet adatokat gyűjteni a kulcsról előre. Hátránya a módszernek, hogy a bizalmi lánc könnyen megtörhet, ha nem figyelünk arra, hogy a változás végigterjedjen a DNS-ben.

- **Első fázis:** Ebben a fázisban kreálunk egy új kulcsot, és ezt bevezetjük a zónafájlba. Ezután aláírjuk a zónát mind az új, mind a régi kulccsal. Ha KSK rollover történik, akkor az új kulcsot be kell vezetni az apánál is, az új DNSKEY vagy DS rekordot el kell neki küldeni, valamint aki a régi kulcsot használja trust anchor-nak, azoknak is frissíteni kell. Ezután megvárjuk, míg ez a változás szétterjed a DNS-ben.
- **Második fázis:** Ebben a fázisban a régi kulcsot eltávolítjuk a zónából, és a zónát már csak az új kulccsal írjuk alá.

5. fejezet

Megvalósítás

5.1. DNSSEC a BIND által nyújtott eszközök segítségével

5.1.1. Az elsődleges autoritatív névszerver fölállítása - a zóna aláírása

A DNSSEC bevezetéséhez[12] az első lépés az, hogy keresünk egy olyan autoritatív névszervert, ami támogatja ezt a konstrukciót. Ezekből több is létezik pl.: BIND9, NSD. Jelen példában mi a BIND-ot választottuk. Fordításkor meg kell neki adni a gépen szereplő *OpenSSL* könyvtárat, mert a kriptográfiai műveleteket ennek segítségével végzi a szerver. A névszerverrel együtt telepítésre kerül két segédprogram is, a `dnssec-keygen` és a `dnssec-signzone`, amivel a kulcsok generálása és a zóna aláírása fog történni.

Jelen esetben a *ppke.hu* és az *itk.ppke.hu* zóna szolgált a teszt alanyául. Ezen két zónát zónatranszferrel le lehet tölteni a tesztkörnyezetként szolgáló virtuális gépre, és ott lehet vele dolgozni.

Első lépésként kulcspárokat kell generálni. Már itt el kell döntenünk, hogy a zónánkban NSEC-et vagy NSEC3-at akarunk használni, mivel más lesz az algorithm mező a kulcsban, és ez problémákhoz vezet az aláíráskor. A

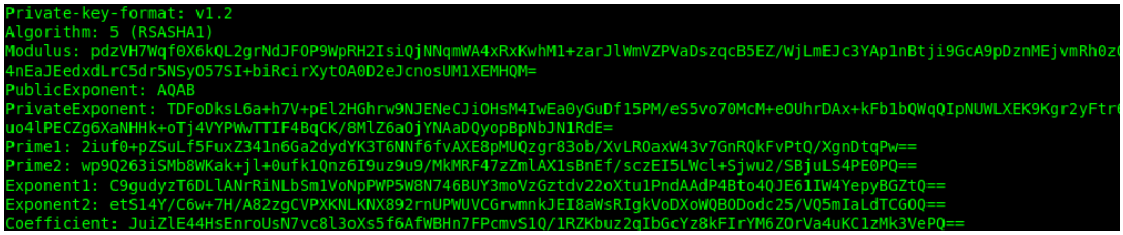
```
dnssec-keygen -r/dev/random -a RSASHA1 -b 1024 -n ZONE itk.ppke.hu
```

parancs kiadásával egy 1024 bites, SHA1-es ZSK kulcspár jön létre az *itk.ppke.hu* zónához ami NSEC-et fog használni. Ha NSEC3-at szeretnénk bevezetni a zónánkban, akkor a `-a` kapcsoló után az RSASHA1-NSEC3-SHA1-et kell megadni, ezzel NSEC3-as ZSK jön létre. Következő lépésként le kell generálni a KSK-t a zónához, amit az alábbi parancs segítségével teszünk meg:

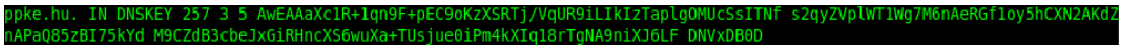
```
dnssec-keygen -r/dev/random -f KSK -a RSASHA1 -b 2048 \
-n ZONE itk.ppke.hu.
```

A parancs kiadása után létrejön egy 1280 bites SHA1-es KSK kulcspár. A `-f` kapcsoló mondja meg, hogy KSK-t generálunk. Ha NSEC3-at szeretnénk használni, akkor ugyanúgy kell eljárni, mint a ZSK generálásánál. A generált kulcsfájlok elnevezése az alábbi konvenció szerint történik: `Kzoname+algorithm_field+key_id`. Emiatt egy kulcsról ránézésre el lehet dönteni, hogy KSK vagy ZSK, valamint az id segít a kulcsok

megkülönböztetésében. Ha ez a két kulcspár sikeresen létrejött, akkor a zónatranszfer eredményeképpen megkapott zónafájlba be kell hivatkozni a két kulcs nyilvános részét. Ezzel a zónafájlunkat fölkészítettük az aláírásra. A generált kulcsokat az 5.1. és az 5.2. ábrákon láthatjuk.



5.1. ábra. A generált titkos kulcs



5.2. ábra. A generált nyilvános kulcs

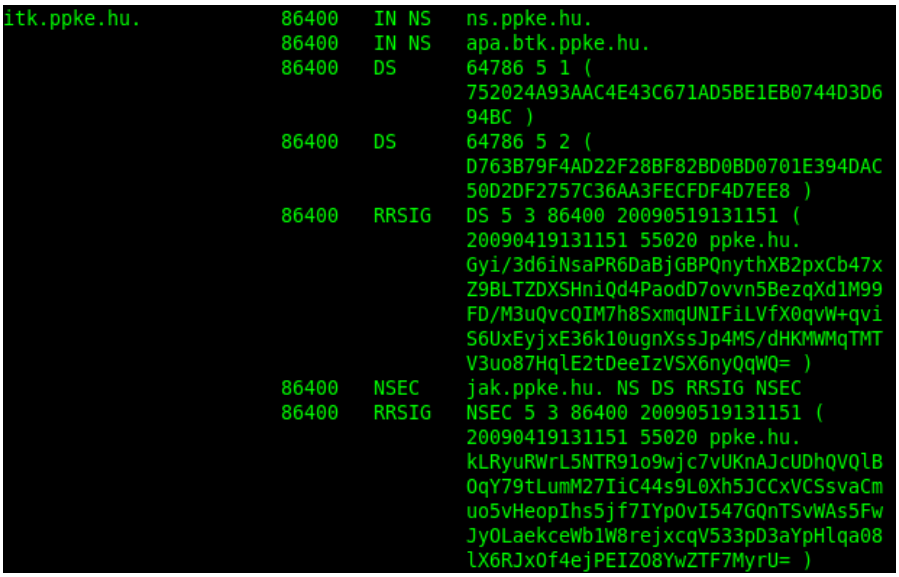
Az aláíráshoz a következő parancsot kell kiadni(a kulcsnevek értelemszerűen változhatnak):

```
dnssec-signzone -o itk.ppke.hu -k Kitk.ppke.hu.+005+49656 \
db.itk.ppke.hu Kitk.ppke.hu.+005+17000.key
```

Ez a parancs sokat változtat a zónán, ezeket most részletesen fölsoroljuk:

- A zónát „abc sorrendbe” rendezi.
- Minden címke megkapja az NSEC vagy NSEC3 rekordját.
- A DNSKEY rekordoknál a kommentben megjelenik a kulcs azonosítója.
- A DNSKEY RRset-et aláírja mind a ZSK-val, mind a KSK-val.
- Az összes többi RRset-et aláírja a ZSK-val.
- Létrejön két fájl, a *dsset-itk.ppke.hu* és a *keyset-itk.ppke.hu*, ezeknek a segítségével tudjuk majd a bizalmi láncot (chain of trust) fölépíteni.
- Létrejön az aláírt zónafájl *db.itk.ppke.hu.signed* néven egy részletet láthatunk a zónából a 5.3. képen.

Ezek után be kell tölteni a BIND-dal ezt a zónafájlt, előtte azonban *named.conf.options*-ban engedélyezni kell a DNSSEC-et a *dnssec-enable yes;* beállítással. A sikeres betöltést ellenőrizni tudjuk a logok között, valamint egy megfelelően paraméterezett lekérdezés segítségével.



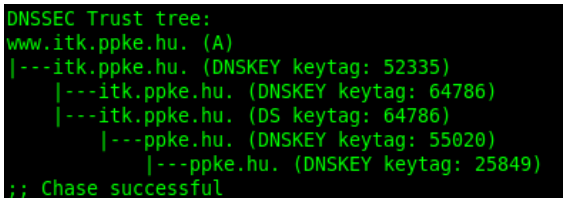
5.3. ábra. Az aláírt *ppke.hu* zóna egy részlete

5.1.2. A chain of trust fölépítése – az apa zóna aláírása

A következőkben a konstrukció egyik leglényegesebb részéről lesz szó, arról hogy hogyan épül föl a chain of trust (bizalmi lánc) a DNSSEC architektúrában, ami a válaszok hitelesítését teszi lehetővé. Az apa zóna aláírása hasonlóan megy végbe a gyerek zónáéhoz, csak a különbségekre térünk ki.

A kulcsok létrehozása és a zónafájlba való fölvétele ugyanúgy történik, mint az előző esetben. Azonban ezután be kell hivatkozni a zónafájl az előzőleg létrejött *dsset-itk.ppke.hu* fájlt. Ezzel mutatunk rá az *itk.ppke.hu*-ra, hogy ott mi a KSK rekord. Az aláírás ezek után ugyanúgy történik, mint a gyerek zónában.

A kész zónafájlt ezután be kell tölteni a BIND-al, majd ha ez megtörtént, és a gyerek zóna is be van töltve, akkor a bizalmi lánc kiépül, ezt az 5.4. ábrán láthatjuk valamint ellenőrizhetjük egy jól paraméterezett `drill` parancs segítségével.

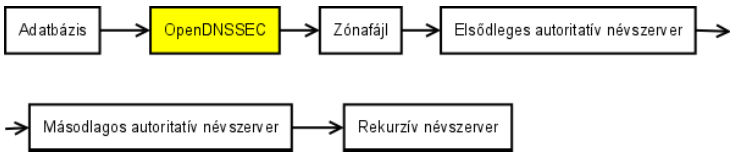


5.4. ábra. A kiépült chain of trust

5.2. DNSSEC az OpenDNSSEC segítségével

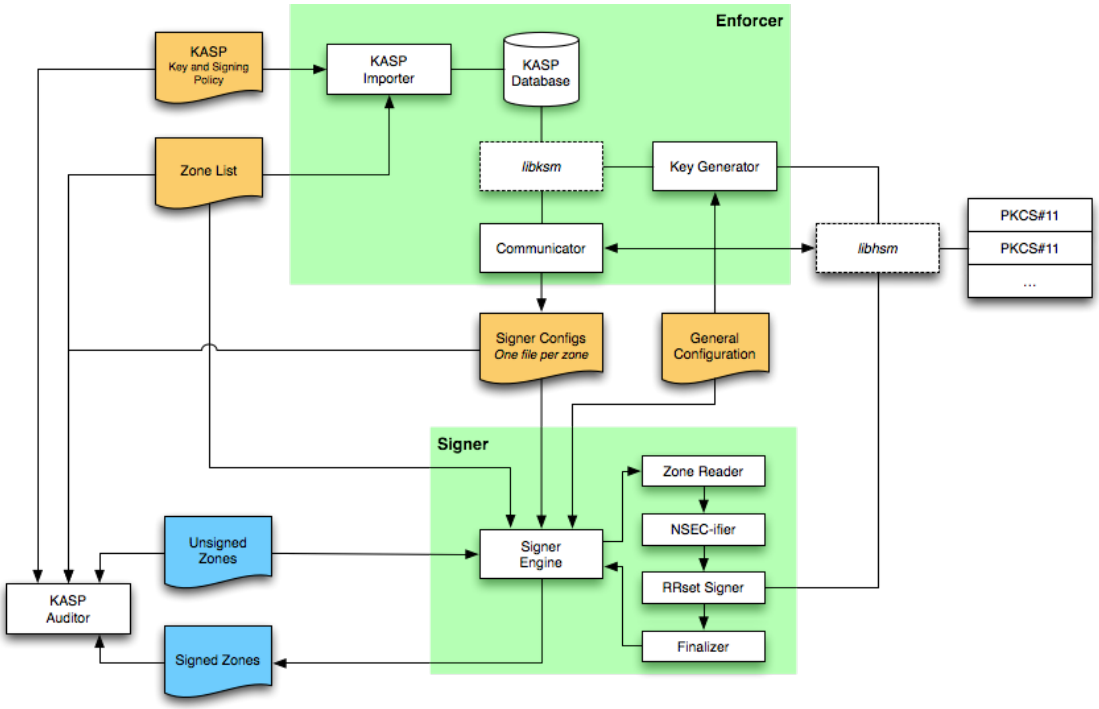
5.2.1. Bevezetés

A DNS az elmúlt 20 évben szervesen beépült a hálózatba, ezért az üzemeltetéshez is jól bevált, kipróbált eszközök állnak rendelkezésre. A DNSSEC ezen a folyamaton sokat változtat, az eddig bevált „best practice” jellegű eszközök nem biztos hogy kezelni tudják a változást. Az OpenDNSSEC[13] ezen próbál segíteni. A 5.5. ábra egy lehetséges architektúrát vázol az OpenDNSSEC beépülését illusztrálva az eddigi munkafolyamatba. Az adatbázisból az aláíratlan zóna bekerül az OpenDNSSEC-be, onnantól kezdve végig az aláírt zóna terjed tovább.



5.5. ábra. Az OpenDNSSEC beépülése a meglévő infrastruktúrába

Az OpenDNSSEC eszközkészlet részletes fölépítése az 5.6. képen[14] látható, a fontosabb részeket az alábbiakban ismertetjük.



5.6. ábra. Az OpenDNSSEC eszközkészlet részletes fölépítése

Enforcer: Ez a modul biztosítja a zónákra definiált policy „betartását”, valamint ezen keresztül tudunk az OpenDNSSEC-cel kommunikálni.

Signer: A policy alapján ez a modul végzi el a zóna tényleges aláírását.

KASP Auditor: [1] Ez a modul automatikus teszteket hajt végre az aláírt zónán, ezzel is elősegítve a problémák elkerülését.

libhsm: Ezen az interface-en keresztül tudjuk a HSM-et elérni.

5.2.2. Az OpenDNSSEC használata

Miután sikeresen telepítettük az OpenDNSSEC-et, végre kell hajtani az alapvető konfigurálást. Ezt a `/etc/opendnssec` mappában található 4 xml fájl segítségével tudjuk megtenni. Ezek szerepét az alábbiakban mutatjuk be.

conf.xml: Ez a fájl szolgál az általános beállítások megadására, itt állítjuk be hogy milyen HSM modult használunk, hol található a többi konfigurációs fájl, melyik könyvtárban dolgozzon a program, milyen SQL backend-et használunk, valamint ha a zónafájl megváltozik, akkor a szervert tudjuk erről értesíteni, aki újra betölti a friss változatot.

kasp.xml: Ez a fájl a program egyil legfontosabb része, itt definiáljuk a policy-ket, amik az egyes zónákra vonatkoznak. A mellékletek között megtalálható az általunk használt policy, valamint a policy írását részletesen bemutatjuk a következő pontban.

zonelist.xml: Ebben a fájlban adjuk meg a zónáinkat amit OpenDNSSEC-cel szeretnénk kezelni, és az egyes zónákra vonatkozó policy-t is definiáljuk.

zonefetch.xml: Az OpenDNSSEC nem csak zónafájlokon képes dolgozni, hanem zónatranszfer eredményeképpen megkapott zónán is. Ebben a fájlban a zónatranszferre vonatkozó paramétereket állíthatjuk be. Az általunk tesztelt konfigurációban nem használjuk.

A konfigurálás után az `ods-control start` parancs kiadásával indíthatjuk el az OpenDNSSEC-cet. A program démonként fut a rendszerünkön (`ods-enforcerd`), az indításhoz hasonlóan az `ods-control` segítségével tudunk interakcióba lépni vele. Néhány gyakran használt parancsot az alábbiakban mutatunk be:

Zóna hozzáadása: `ods-ksmutil zone add --zone example.com.`

KSK exportálása: `ods-ksmutil key export --zone example.com`

Policy frissítése: `ods-ksmutil update kasp`

5.2.3. OpenDNSSEC policy írása

Az OpenDNSSEC-ben az egyes zónákhoz policy-t definiálunk, ez tartalmazza a zónára vonatkozó szabályok összességét. Itt határozzuk meg a kulcsaink méretét, a key-rollover gyakoriságát, az aláírások élettartamát, a zóna frissülésének gyakoriságát, a delegáló zónára vonatkozó időbeli paramétereket, etc... Ezeket a paramétereket az alábbiakban

[1] **K**ey **A**nd **S**igning **P**olicy Auditor

részletesen bemutatjuk, és kitérünk a gyakorlati megfontolásokra is, amit a megíráskor figyelembe vettünk.

Signatures/Resign: Ez határozza meg hogy milyen gyakran néz rá a program az aláíratlan zónára, változásokat keresve. Ezt mi 3 órában definiáltuk, a *ppke.hu* zóna ritkán frissül.

Signatures/Refresh: Itt állítjuk be azt hogy milyen gyakran generáljuk újra az aláírásokat. Ezt 7 napra konfiguráltuk, ez elég a ritka változások miatt.

Signatures/Validity: Az aláírások érvényességét határozza meg. Mi 14 napot adtunk meg itt, így ha valami baj történik a Refresh-nél, akkor is lesz érvényes aláírása a zónánknak.

Signatures/Jitter: Véletlen paraméter, amit az aláírások érvényességének az idejéhez adunk hozzá, ezáltal biztosítva hogy nem egyszerre jár le az összes. Itt az alapértelmezett 12 órát adtuk meg.

Signatures/InceptionOffset: Ezt a paramétert kivonjuk az aláírás megkezdésének idejéből, így biztosítva azt hogy ha a kliens órája nincs teljesen szinkronban a mienkkel, akkor se kapjon aláírt adatokat a „jövőből".

Denial: Ebben a szekcióban definiáljuk hogy NSEC-et vagy NSEC3-at használunk a zónánkban. Az NSEC3-hoz extra paraméterek is tartoznak: meg kell adni a használni kívánt hash algoritmust, az iterációk számát, a salt hosszát, és hogy milyen gyakran cseréljük le a salt-ot. Jeleleg SHA-1 algoritmust és 8 hosszú salt-ot használunk, új salt-ot az alárásokkal együtt, 2 hét után generálunk.

Keys: Itt adjuk meg a KSK és a ZSK közös paramétereit. Ezek közé tartozik a kulcsok TTL-je és a régi kulcsok törlésének időpontja. Két hét után töröljük a használaton kívüli kulcsokat a HSM-ből, a DNSKEY rekordok TTL-jét pedig egy órában állapítjuk meg, így lehetővé válik a kulcsok gyors lecserélése, ha véletlen a titkos része nyilvánosságra jutna.

Keys/KSK: Itt adjuk meg a KSK paramétereit: jelenleg egy 2048 bites, RSASHA512 típusú kulcsot generálunk, amit egy év után cserélünk le.

Keys/ZSK: Itt adjuk meg a ZSK paramétereit: jelenleg egy 1024 bites, RSASHA512 típusú kulcsot generálunk, amit egy hónap után cserélünk le.

Zone: Ebben a részben adjuk meg a zónára vonatkozó általános paramétereket: az alapértelmezett TTL-t, az NSEC vagy NSEC3 rekordok TTL-jét, valamint a serial típusát. A konfigurációban 1 nap szerepel a default TTL-hez, ezzel kíméljük a szervereket, és 1 óra a az NSEC* rekordokhoz, így ha szükség van rá, gyorsan be tudunk vezetni új rekordokat. A serial típusa „datecounter", ezt az aláíratlan zónából örököljük meg.

Parent: Ebben a szekcióban találhatóak a delegáló zónára és a DS rekordra vonatkozó időbeli paraméterek. Ezeket a kapcsolattartás miatt fontos definiálni. Jelen esetben a PropagationDelay alapértelmezett értékét használtuk, a SOA értékét a *.hu* zónából szereztük be, így 1 napra állítottuk. A DS rekord TTL-jét is 1 napra állítottuk, mivel a KSK rollover ideális esetben évente egyszer valósul meg.

Audit: Itt állítjuk be, hogy szeretnénk-e auditálni a zónánkat. Ez hasznos funkció, hiszen a hibákra még a zóna publikálása előtt fény derülhet. Nagy zónáknál célszerű a „Partial" funkciót használni, ami mintát vesz a zónából, és csak azt ellenőrzi, így időt spóról meg nekünk. A *ppke.hu* kis zónának számít, teljes auditálást kértünk rá.

5.2.4. Az OpenDNSSEC előnyei

Az OpenDNSSEC számos előnnyel rendelkezik a BIND féle módszerhez képest. Az első, és talán legfontosabb az hogy nem foglakozunk közvetlen a kulcsok generálásával, tárolásával, etc..., hanem policy-t írunk, amit az enforcer segítségével tartatunk be, ezáltal magasabb absztrakciós szinten, sokkal egyszerűbben kezeljük a problémát.

Az OpenDNSSEC könnyűvé teszi a menedzselést, az xml alapú policy egyszerűen kezelhető, így akár egyszerre sok zónát is módosíthatunk egy változtatással. Automatikusan, beavatkozás nélkül megoldja a ZSK rollover-t, a KSK rollover-hez pedig támgatást nyújt. A működtetéshez további támogatást jelent, hogy az OpenDNSSEC adatbázist tart fönt a kulcsaink állapotáról[15], ezáltal például kevésbé valószínű, hogy lejárt kulcsot publikálunk, vagy a key-rollover-ben hibázunk. Támogatja a legújabb fejlesztéseket, draft-ot implementál, több DNS operátor is részt vesz a fejlesztésében.

Az OpenDNSSEC BSD licenc alatt került publikálásra, ezáltal szabadon fölhasználhatjuk a saját szoftvereinkben, nem köti meg a kezünket a licencelés, akár fizetős programcsomag részeként is terjeszthetjük.

5.3. Zonewalking

5.3.1. Bevezetés

A DNSSEC konstrukció egyik alapja, hogy nem csak a létező bejegyzéseket kell hitelesíteni, hanem arra is aláírt választ kell adni, ha az adott bejegyzés nem létezik. Ezzel például el lehet kerülni azokat az eseteket, hogy az ISP a nem létező domaineket reklámoldalakra irányítja. Ezt a tulajdonságát a zónának az NSEC rekordok garantálják. Definíció szerint minden, a domainban található RRset-hez tartozik egy NSEC rekord, ami a rákövetkező rekordra mutat. Az aláíráskor a domain „abc szerinti" sorrendbe kerül, és ekkor kapja meg az NSEC rekordokat is, ezáltal válik értelmezhetővé a rákövetkező elem. Az NSEC rekordok az egész domainon keresztülhúzódnak, ezáltal egy láncot alkotnak fölötte. Ha egy nemlétező bejegyzésre érkezik kérés, akkor a válasz egy NXDOMAIN lesz és a válasz tartalmaz egy NSEC rekordot is, ami a következő létezőre mutat, ezáltal hitelesítve azt, hogy amit kérdeztünk az nem létezik.

A probléma ezzel az, hogy így a zónánkat le lehet tölteni egy alternatív módszerrel: mindig a nemlétező rekordokra kérdezünk rá, ezzel visszakapunk egy bejegyzett rekordot, amivel tudunk tippelni egy következő nemlétezőre. Így akkor is hozzáférhetővé válik az egész zóna, ha a zónatranszfer tiltva van. Ez ahhoz hasonlatos, ha valaki ahelyett, hogy a titkárságot hívná egy cégnél, és a titkárnő kapcsolná az adott kollegát, megszerezné a céges telefonkönyvet és hívhatna bárkit.

5.3.2. Tapasztalatok a megírt programmal

A programunkat először a saját magunk által létrehozott tesztzónán próbáltuk ki, itt sikeresen le is futott, az *itk.ppke.hu* és a *ppke.hu* zóna tartalmát meg bírtuk vele szerezni. A következő tesztalany a *.hu* zóna volt, ami a *sechu.iszt.hu* tesztszerveren található meg. Ezen a zónán is végigfutott a program, így pl.: megtudtuk, hogy a *.hu* alatt mennyi bejegyzés szerepel pontosan.

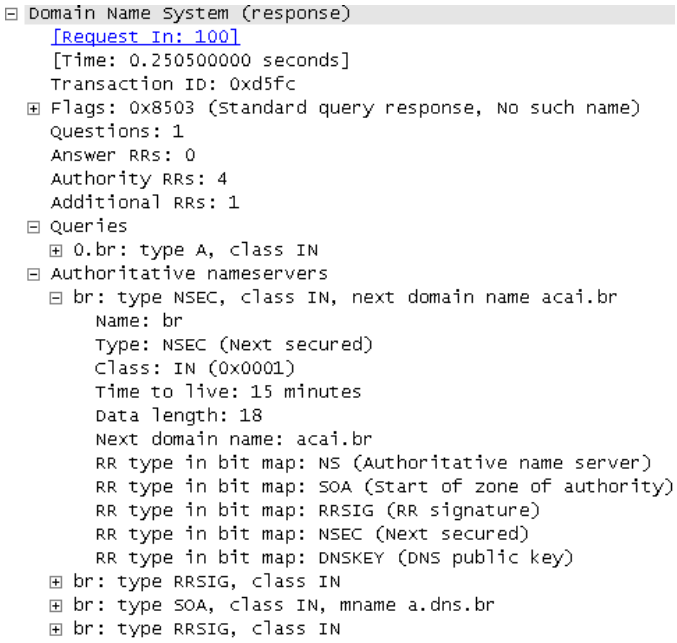
A *.cz* és a *.se* zónáknál már problémákba futottunk. Az RFC1035[2] 2.3.1-es bekezdése megkötést tesz a címkék hosszára a DNS-en belül, 63 karakterben maximalizálja őket. A svéd zónában található egy bejegyzés, ami 63 darab „a” betűből áll, amihez ha hozzárakunk egy 0-át akkor érvénytelenné válik a lekérdezés. A cseh zónában ehhez hasonlóan található egy „hy” sorozat, ami miatt hasonlóképpen nem lehet a zónát végigjárni.

Ideiglenes megoldásként a programot módosítottuk, hogy egy bizonyos ponttól kezdve tudja a zónát bejárni, így a problémás domain-t átugorjuk. Az elegánsabb megoldás a problémára valami jobb tippelés módszer, ami kezelni tudja az ilyen extrém eseteket.

A brazil zónában ennél is furcsább hibákba ütköztünk. Első megközelítésként megnéztünk egy válaszcsomagot, amit a brazil autoritatív szerver küld, és láttuk, hogy a rekordok sorrendje teljesen más, mint amit eddig megszoktunk. Ezután elkezdtük vizsgálni a szervert, hogy vajon miért viselkedik így. A vizsgálat végére arra jutottunk, hogy a brazilok nem a BIND9-et, hanem az NSD-t használják névszerverként, ebből adódhatnak a különbségek. Ezt megerősíteni látszik az NLnet Labs honlapján található dokumentáció[16] is, ami a BIND9 és az NSD különbségeit taglalja, többször külön kitérve a DNSSEC-re.

A rekordok sorrendjének eltérése azért okoz problémát, mivel a szkriptben ez fix, így ha az NSEC rekord nem ott van, ahova várjuk, akkor kifagy a program. Valamint a BIND által adott válaszban, amint az 5.8 képen is látjuk, két NSEC rekord is van, egyik a szülő zónára vonatkozik, és csak a másik az adott kérdésre. Ebből csak az adott kérésre küldött válasz NSEC rekordja a hasznos, a másikat el kell dobni. Ezen problémákat a válasz részletesebb vizsgálatával ki lehet küszöbölni, amire idő hiányában nem került sor.

A szkript fejlesztéséhez és a képek elkészítéséhez a `wireshark`[17] program biztosított segítséget.



5.7. ábra. A brazil (NSD) névszerver által adott válasz

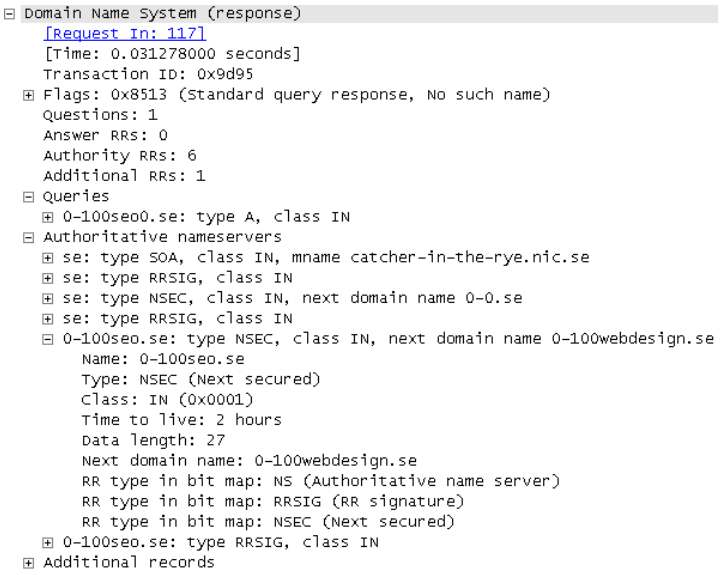
5.4. Az aláírások ellenőrzésének időigénye

A DNSSEC két, kölcsönösen kizáró módszert alkalmaz az NXDOMAIN válaszokban a nem bejegyzett domain-nek nemlétének bizonyítására. A különbség a kettő között az, hogy az NSEC-ben a zónában szereplő aldomain-ek neve szerepel, míg NSEC3-nál az utolsó label hash-e szerepel, ezzel megakadályozva a zonewalking létrejöttét. A 5.9. képen NSEC-es zónarészletet, a 5.10. képen pedig egy NSEC3-as zónarészletet láthatunk, ahol jól látszik a különbség a kétféle aláírási technika között.

Azonban az NSEC3-nak ára van, amit az aláíráskor, és az NXDOMAIN válaszban érkező NSEC vagy NSEC3 rekordok hitelesítésekor fizetünk meg. Mivel az aláírás offline történik meg, ezért annak az idejével nem foglalkozunk. Az érdekes kérdés az, hogy ellenőrzéskor mi történik, mivel ez sokkal nagyobb terhet ró a szerverekre és gyakrabban kell egy-egy aláírást ellenőriznünk, mint zónát aláírnunk, mivel ellenőrzés történik minden olyan esetben, amikor nekünk a DNS-ből szükségünk van valami információra, pl.: levélküldésnél, weblapok megtekintésénél, távoli gépekre való csatlakozáskor.

Vizsgálatunk a különböző kulcsméretekre vonatkozik és a két fajta aláírási metódust hasonlítja össze azonos kulcsméretekre. A másik szempont ami alapján vizsgálunk az az, hogy a mérés a szerveren történik, vagy egy távoli gépen. Ez azért fontos, mert ha a mérőszkriptet a szerveren futtatjuk, akkor a hálózati késleltetés kiküszöbölhető, ezáltal pontosabb eredményeket kapunk. A valóságban azonban számolnunk kell a a hálózati késleltetéssel, ezért a méréseket egy távoli gépen is végrehajtjuk.

A kétféle kulcs közül (KSK és ZSK), a ZSK kulcsméretét érdemes megvizsgálni, mivel a KSK-t csak a zónába való „belépéskor” használjuk, míg a ZSK-val történik a tényleges ellenőrzés.



5.8. ábra. A svéd (BIND9) névszerver által adott válasz

5.4.1. Az első mérés

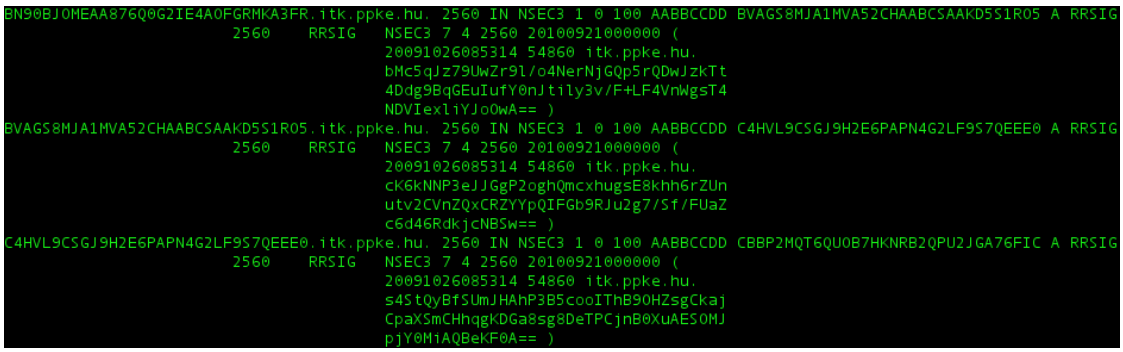
Először generáltunk mindegyik típusból egy-egy kulcsot, aláírtuk vele az *itk.ppke.hu* zónát, majd a mérőszkriptet lefuttattuk rá. A mérőszkript a következőképpen működik: egy preparált zónafájl segítségével végigmegyünk a zónán 30-szor, végig nem létező rekordokra kérdezünk rá, ezáltal NSEC-es vagy NSEC3-as válaszokat kapunk vissza. A preparált zónafájlra azért van szükség hogy NSEC3 esetben tudjuk szimulálni a zonewalking effektust, ezáltal az egész zónát meg tudjuk vizsgálni. Ezen válaszok aláírásait a `perl Net::DNS::SEC` könyvtárának `Net::DNS::RR::RRSIG` osztályának *verify* metódusával ellenőrizzük. A szkriptnek meg kell adni az *itk.ppke.hu* KSK-ját, a preparált zónafájlt és a névszerver ip-jét.

A mérési eredményeket a 5.11. képen látható táblázat és grafikon tartalmazza.

A vízszintes tengelyen a ZSK kulcs mérete (512,1024,2048,4096), a függőleges tengelyen az idő szerepel másodpercben.

A mérési eredményeket a következőképpen értelmezzük: NSEC local esetben a vártnak megfelelő növekedést tapasztaltunk a kulcsmérettel arányban. Mindkét esetben 2048 bitnél érdekes csökkenést tapasztaltunk, ami további vizsgálatokat igényelt. Az alapvető különbség NSEC és NSEC3 között a rekordok hossza miatt van. Az NSEC3 remote eset érdekes még a számunkra, mivel ott az utolsó teszt (a 4096 bites kulccsal) nem futott le. Emiatt ezt az esetet ledumpoltuk, és érdekes módon válaszként egy ICMP csomagot kaptunk `host unreachable` üzenettel. Amint a konzultáció során ez kiderült, a küldő gép az *itk.ppke.hu* tűzfala (null.itk.ppke.hu) volt. Az ok amiért küldte az volt, hogy az 53-as porton a TCP forgalom tiltás alá esett. AZ UDP csomagok szabadon mehettek át ugyanezen a porton, ez a null.itk.ppke.hu-n futó, OpenBSD-ben található `pf` tűzfalkonfiguráció furcsasága. Amiért érdekes ez a dolog az az, hogy itt tértünk át először UDP-ről TCP-re. Ezt az áttérést az UDP csomagban megtalálható `tc` flag is bizonyítja. Ezt azért fontos tudni, mivel a TCP nagyobb terhet ró a hálózatra mint az

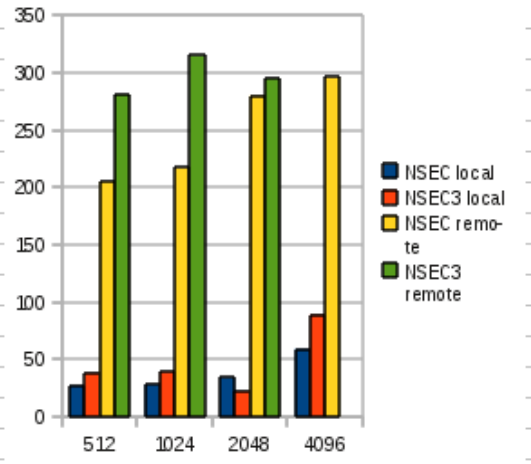
5.9. ábra. Az NSEC-es zóna egy részlete



5.10. ábra. Az NSEC3-es zóna egy részlete

UDP. Csak NSEC3 esetben kellett áttérni, NSEC esetben még belefértek a válaszok.

	NSEC local	NSEC3 local	NSEC remote	NSEC3 remote
512	27	38	205	281
1024	28	39	217	315
2048	34	22	280	295
4096	59	89	296	



5.11. ábra. Az első mérés eredménye

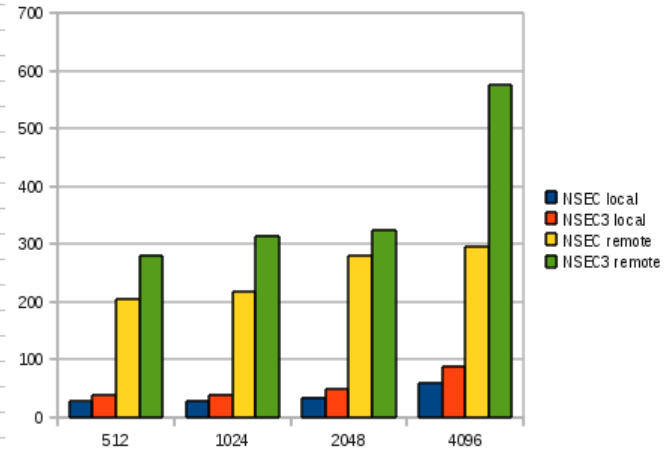
5.4.2. A második mérés

Miután a tűzfalon engedélyezve lett a TCP 53 porton zajló forgalom, a mérést megismételtük, ugyanazon peremfeltételek mellett. A 2048 NSEC3 esetet egy kicsit jobban megvizsgáltuk, új kulcsokat generáltunk hozzá, és arra is lefuttattuk a mérést.

A mérési eredményeket a 5.12. képen látható táblázat és grafikon tartalmazza.

A mérési eredményeket a következőképpen magyarázzuk: 4096 NSEC3 esetben nagyságrendbeli eltérést tapasztalunk az NSEC-hez képest. Ez a TCP-re történő váltás következménye, sokkal nagyobb lett az egyes DNS kéréseken az overhead. Ebből az látszik, hogy a TCP-t meg kell próbálni elkerülni, amennyire csak lehet. A másik problémás esethez (2048 bit) új kulcsokat generáltunk, mivel az volt a sejtés, hogy valami oknál fogva a generáláskor történt valami, azért tudtunk vele hatékonyan számolni. Ez a sejtés be is igazolódott, a fönti képen már a három mérés átlaga szerepel, ami szépen be is illeszkedik a sorozatba, ahogy azt mi elvártuk. Ez az eredmény arra sarkallt minket, hogy a kulcsgenerálást jobban megvizsgáljuk, ehhez a méréshez sok kulcsot generáljunk, és mindegyikre lefuttassuk a mérőszkriptet.

	NSEC local	NSEC3 local	NSEC remote	NSEC3 remote	2048v2 local (urandom)	2048v3 local (random)
512	27	38	205	281	61	56
1024	28	39	217	315	2048v2 remote (urandom)	2048v3 remote (urandom)
2048	34	48	280	324	335	340
4096	59	89	296	575		

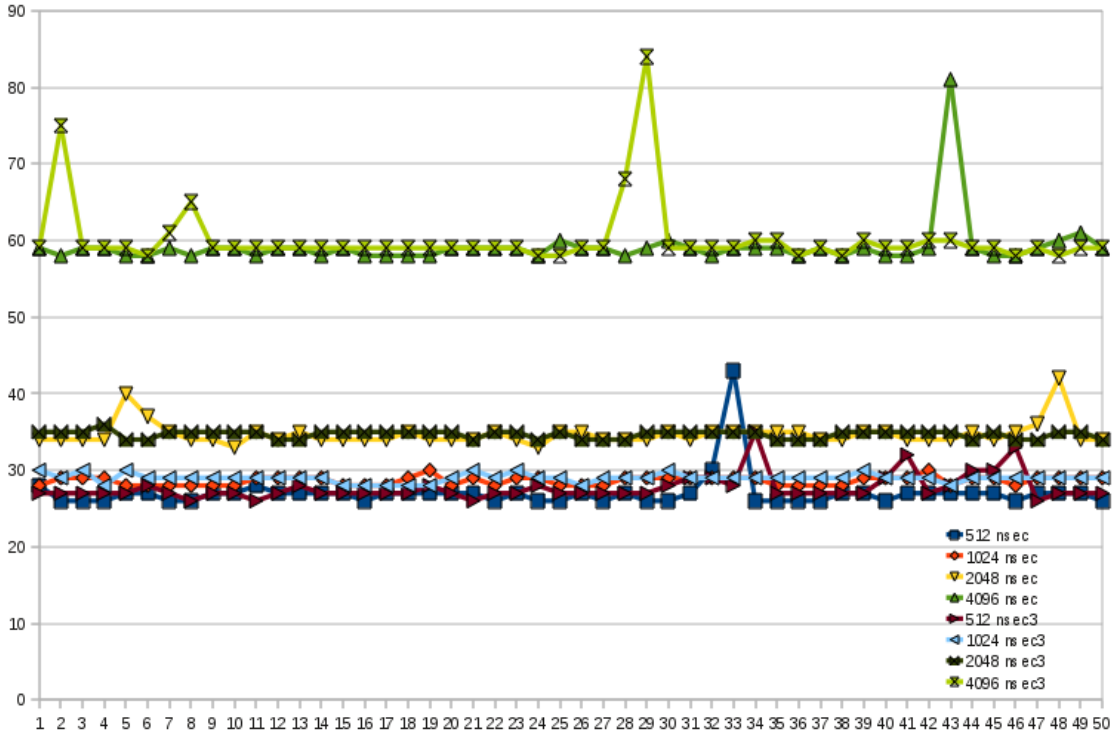


5.12. ábra. A második mérés eredménye

5.4.3. A harmadik mérés

A harmadik mérés során generáltunk 50 db KSK-t és hozzá 50 db ZSK-t. Ez KSK-ból 50 évre, zsk-ból pedig 4 évre elegendő kulcsmennyiség. A generált kulcspárokkal aláírtuk az *itk.ppke.hu* zónát, és mindegyikre lefuttattuk a mérőszkriptet.

A 5.13. diagramon látható mind az 50 mérés NSEC és NSEC3 esetre, különböző kulcsméretekre. Itt látszik jól, hogy tényleges időbeli különbség nincs NSEC és NSEC3 között. Az egyes kiugrások okát nem tudjuk pontosan megmagyarázni, az a sejtésünk, hogy a kulcsgenerálásnál történt valami, pl.: nem volt megfelelő az entrópia, ezért generálódott az átlagostól eltérő kulcs. Ennek a mérésnek az eredménye hozta azt, amit elvártunk tőle, nem okozott meglepetést. Sokkal érdekesebb azonban az az eset amikor egy távoli gépről mérünk, mivel az a valóságban előforduló eset. A szemléltetést elősegítendő, ezek átlagaiból készítettünk egy diagramot, ami a 5.14. képen látható.

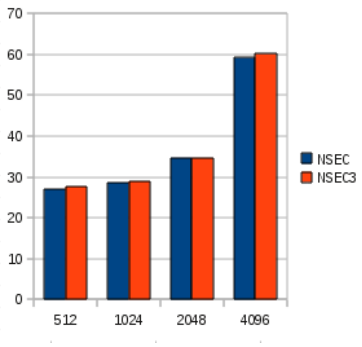


5.13. ábra. A harmadik mérés eredménye

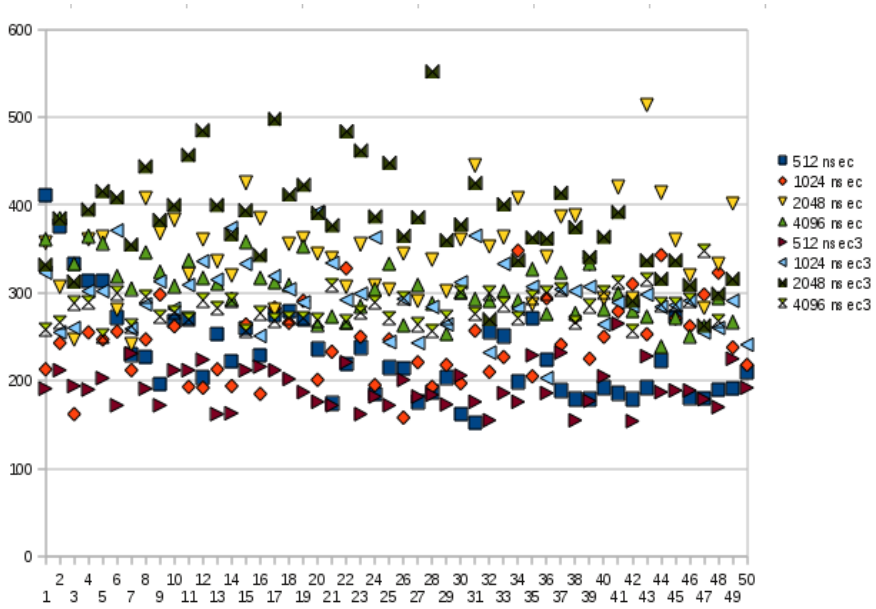
5.4.4. A negyedik mérés

Ez a mérés az előzőleg generált kulcsokkal és aláírt zónákkal történik, a különbség mindössze annyi, hogy most nem a szerveren fut a szkript, hanem egy távoli gépről. Emiatt a hálózati késleltetés megjelenik a mérés eredményében. A kapott eredményeket az alábbiakban ismertetjük.

	NSEC	NSEC3
512	27,04	27,68
1024	28,62	28,98
2048	34,68	34,7
4096	59,16	60,12



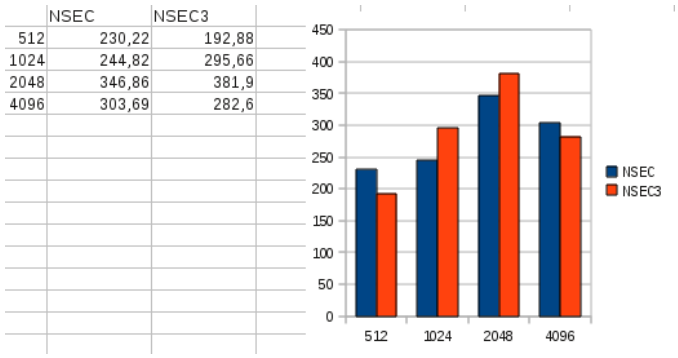
5.14. ábra. Az 5.13. képen található mérés kiátlagolt eredménye



5.15. ábra. A negyedik mérés eredménye

A 5.15. grafikonról leolvasható, hogy a hálózat késleltetése nagyban befolyásolja a mérést. Ebben az esetben is célszerű kiátlagolni az eredményeket.

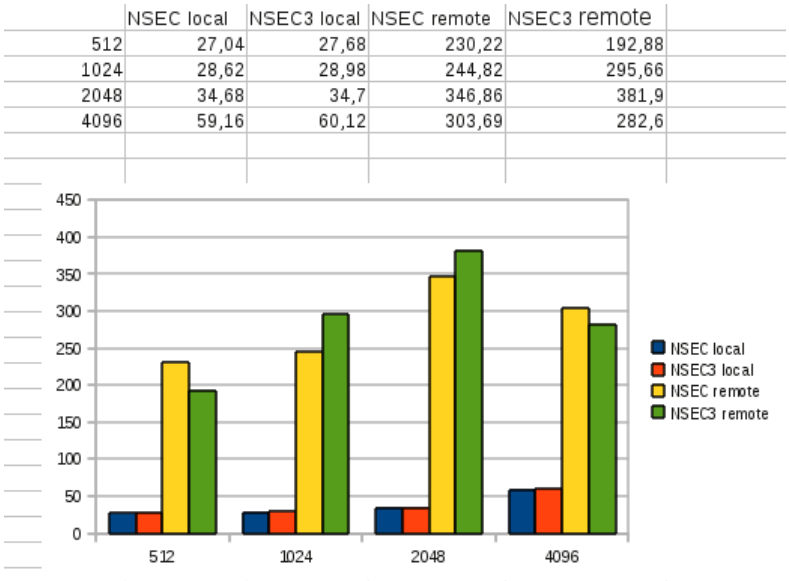
Az átlagolt eredmények a 5.16. képen a várakozásunknak megfeleltek. Az ingadozásokat az okozhatta, hogy a mérés több mint 2 napon keresztül futott, így a „forgalmas órák” is beleestek, ami nagyban befolyásolta a hálózat válaszidejét.



5.16. ábra. A negyedik mérés kiátlagolt eredménye

5.4.5. Konklúzió

A konklúziót az alábbi diagram (5.17.) tartalmazza, ami az utolsó két mérés átlagait jeleníti meg egyben.



5.17. ábra. Az utolsó két mérés eredménye egy diagramon

A különbség NSEC és NSEC3 között elhanyagolható minden esetben. A validáláshoz szükséges időt a hálózati késleltetés határozza meg. Igyekeznünk kell a forgalmat UDP fölött megvalósítani, mivel még a TCP-re való áttérés okozhat lassulást az aláírások ellenőrzésénél.

6. fejezet

Összefoglalás

6.1. Elért eredmények

A diplomamunka elkészítése során sikeresen áttekintettem a DNS-re és a DNSSEC-re vonatkozó RFC-ket, ezáltal mély elméleti ismeretekre tettem szert.

Kipróbáltam és megvizsgáltam több DNSSEC-et támogató eszközt: autoritatív és rekurzív szervereket, valamint az aláírást támogató programkönyvtárakat és a rájuk épülő programokat. Ezeket többféle szempont szerint összehasonlítottam, üzemeltetési oldalról megismertem. Az *ppke.hu* és az *itk.ppke.hu* zónákon a konstrukciót üzembe helyeztem, a kulcsméretekkel, a negatív válaszokat hitelesítő rekordokkal, és a megnövekedett forgalommal kapcsolatos méréseket hajtottam végre rajtuk. Konzulensem jóvoltából hozzáférhettem a *.hu* zónához is, ami különleges abból a szempontból hogy szinte teljes egészében delegálásokból áll. Itt ki tudtam próbálni a DNSSEC-ben szereplő delegálási mechanizmust támogató rekordok hatását a zónára nézve.

Részletes vizsgálat alá vetettem a zonewalking effektust, és a kivédésére tett megoldás előnyeit és hátrányait is tanulmányoztam.

Összefoglalásként ismeretetem a DNSSEC bevezetésének előnyeit és hátrányait, valamint a megoldásra váró problémák közül is bemutatok egyet.

6.2. A DNSSEC előnyei és hátrányai

6.2.1. Előnyök

A DNSSEC konstrukció fő előnye a zónák aláírása, így a benne szereplő rekordok hitelesíthetőek. Ennek számos előnye van:

Spam: Könnyebbé válik a spam elleni védekezés, például helyi policy-vel szabályozzuk hogy melyik domainekből fogadunk el leveleket, ehhez a trust anchor-ok nyújtanak segítséget.

Cache poisoning: A cache poisoning veszélye megszűnik, mivel a becsempészett rekordhoz nem tud a támadó aláírást generálni.

Man in the middle veszélyek: A DNS csomagokat az egyes hálózatok üzemeltetői minden nehézség nélkül meghamisíthatják. A DNSSEC-cel ez nem lehetséges, az aláírások megvédik a rekordokat.

Új rekordok fölvétele: Csak a titkos kulcs ismeretében vehetünk föl új rekordokat a zónába, így a munkatársak sem tudják egyszerűen módosítani a zónát.

Üzleti okok: Egy banknak fontos lehet ez a technológia, például a netbank szolgáltatás biztonságosabbá tételéhez.

CERT rekord: CERT rekordokat[18] tudunk bevezetni és aláírni, ami által az SSL certificate-ek könnyen terjeszthetővé és karbantarthatóvá válnak, nem kell föltétlen a böngészőbe integrálni őket.

6.2.2. Hátrányok

A DNSSEC nyújtotta előnyök mellé hátrányok is társulnak:

Zónaméret: A kulcsok és aláírások megnövelik a zóna méretét, az elvégzett kísérletek alapján 7×-es növekedést várhatunk.

Kulcsmenedzselés: A kulcsainkat továbbítani kell a delegáló zóna felé, figyelni kell a bizalmi lánc megőrzésére.

Csomagméret: Megnövekedik a DNS csomagok mérete az aláírások és a kulcsok miatt, a régebbi eszközök, tűzfalak ezt nem biztos hogy kezelni tudják.

Támogatottság: Mivel a DNSSEC még gyerekcipőben jár, probléma lehet olyan szoftvereket, hardvereket, valamint szakértőket találni a piacon akik jól kezelik és ismerik a konstrukciót.

Algoritmusok: Figyelemmel kell kísérni a technológia aktuális állását, a régi, nem biztonságos algoritmusokat újabbakra kell cserélni.

OS szintű megoldás: A böngészőben használt SSL tanúsítványokkal ellentétben a DNS-t operációs rendszer szinten kell kezelni, például az e-mail kliensnek is jelezni kell, ha nem tudta hitelesíteni a rekordokat a levélküldés folyamán.

6.3. Kurrens problémák a DNSSEC-ben

A DNSSEC fejlesztése napjainkban is folytatódik, a jelenleg fölmerülő problémákra újabb és újabb RFC-k születnek. Egyik jelentős feladat a „best parctice” jellegű dokumentumok elkészítése az üzemeltetőknek, a nagy ISP-ktől kezdve egészen a hétköznapi fölhasználókig. Jelenleg több dokumentum is létezik[10][11], a bennük szereplő adatok elévülő volta miatt egyre újabbak keletkeznek.

Egy másik megoldandó feladat a témában a „last mile” kommunikáció biztosítása, ami a stub rezolver és a caching névszerver között zajló adatok biztonságát garantálja.

Erre jelenleg több ötlet is van: az egyik megoldás a már jól ismert és széles körben használt TSIG használata. A TSIG egy szimmetrikus kulcsú titkosítást alkalmazó eljárás, manapság a primary és a secondary autoritatív névszerverek közötti kommunikációt biztosítja. Egy lehetséges megoldás lehet az hogy a DHCP szerver osztja ki a szimmetrikus kulcsokat a klienseknek, az IP címekhez hasonlóan.

Egy másik ötlet az IPSEC alkalmazása erre a célra, így a DNSSEC-től független titkosítási és authentikálási eljárás védi a csomagjainkat a hálózaton.

A harmadik megoldás az lehet, ha a titkosítás ellenőrzését a stub rezolverekre bízzuk, így a probléma megszűnik, azonban ez jelenleg nem szabványos megoldásnak minősül.

7. fejezet

Fogalomtár

DNSSEC: Domain Name System Security Extensions, a meglévő DNS protokoll kiterjesztése biztonsági funkciókkal. Az adatok hitelességét és integritását garantálja.

KSK: Key Signing Key, ezt a kulcsot használjuk arra hogy a zóna DNSKEY rekordját aláírjuk vele, ezzel a kulccsal történik meg a bizalmi lánc fölépítése.

ZSK: Zone Signing Key, ezzel a kulccsal írjuk alá a zónában szereplő összes rekordot.

AD: Authenticated Data, ez a bit jelzi, hogy a válasz hitelesítve lett a válaszoló DNS szerver által.

CD: Checking Disabled, ez a bit jelzi a kérésben, hogy én szeretném hitelesíteni a választ. Kölcsönösen kizáró kapcsolatban áll az AD bittel.

DO: DNSSEC OK, ezzel a bittel jelezzük, mind a kérésben, mind a válaszban azt, hogy használjuk a DNSSEC-et.

DNSKEY: Ebben a rekordban van a nyilvános kulcs, amivel az aláírásokat ellenőrizzük.

NSEC: Next Secure, ez a rekord tartalmazza a következő nevet a domainban, az aktuálishoz képest.

NSEC3: Ugyanazt a szerepet látja el mint az NSEC, csak itt a domain neve helyett egy hash-t kapunk vissza, ezzel akadályozza meg a zone-walking-ot.

RRSIG: Resource Rekord Signature, ez a rekord tartalmazza az aláírást, ami hitelesíti az adott rekordot.

DS: Delegation Signer, ez a rekord tartalmazza a gyerek zónájának KSK-ból képzett hash-t, ennek a rekordnak a segítségével épül fel a bizalmi lánc.

Chain of trust, (Bizalmi lánc): Az a lánc, amelyen végighaladva tudjuk hitelesíteni a választ. A lánc csúcsa egy trsut anchor.

Trust anchor: A névszerverbe bekonfigurált DNSKEY vagy DS rekord, amiben mi megbízunk, és ennek segítségével további válaszokat is tudunk hitelesíteni. Ideális

esetben ez egy root névszerver, így innen kiindulva minden választ tudunk hitelesíteni, de ezt helyi szabályok fölülírhatják.

Zonewalking: A DNSSEC kontrukciós nemkívánatos mellékhatása, ami lehetővé teszi a teljes zóna letöltését. Ellenszere ennek az NSEC3 rekord használata NSEC helyett.

Key-rollover: Az a folyamat amikor a DNSSEC-beli kulcsainkat újakra cseréljük.

BIND: Berkeley Internet Name Daemon, a legelterjedtebb interneten használatos névszerverimplemetáció. Jeleneg az ISC fejleszti, tartja karban. `http://www.isc.org/software/bind`

NSD: Name Server Daemon, az NlNet labs és RIPE által közösen fejlesztett autoritatív névszerver. `http://www.nlnetlabs.nl/projects/nsd/`

Bailiwick checking: Az Additional és Authority szekcióra vonatkozó ellenőrzés, ami azt biztosítja, hogy az itt szereplő nevek tényleg a kérdéses zónából származnak.

HSM: Hardware Security Module, ezzel az eszközzel tudunk generálni kulcsokat, valamint a kulcsok tárolása is itt történik.

8. fejezet

Köszönet

Köszönöm mindenkinek a segítséget, akiktől diplomamunkám elkészítése során szakmai tudást, türelmet, bizalmat, technikai és emberi segítséget kaptam.

Külön köszönet konzulensemnek, Pásztor Miklósnak, aki a diplomamunka felé vezető úton szakmailag és emberileg mindvégig támogatott.

Irodalomjegyzék

[1] P. Mockapetris, „Domain names - concepts and facilities." RFC 1034 (Standard), Nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, `http://www.ietf.org/rfc/rfc1034.txt`.

[2] P. Mockapetris, „Domain names - implementation and specification." RFC 1035 (Standard), Nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, `http://www.ietf.org/rfc/rfc1035.txt`.

[3] S. M. Bellovin, „Using the domain name system for system break-ins," in *Proceedings of the Fifth Usenix Unix Security Symposium*, (Salt Lake City, UT), pp. 199–208, June 1995. `http://www.cs.columbia.edu/~smb/papers/dnshack.pdf`.

[4] „Information about DNSSEC for the Root Zone." `http://www.root-dnssec.org/`.

[5] S. Friedl, „An Illustrated Guide to the Kaminsky DNS Vulnerability." `http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html`.

[6] R. Elz and R. Bush, „Clarifications to the DNS Specification." RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452, `http://www.ietf.org/rfc/rfc2181.txt`.

[7] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, „Resource Records for the DNS Security Extensions." RFC 4034 (Proposed Standard), Mar. 2005. Updated by RFCs 4470, 6014, `http://www.ietf.org/rfc/rfc4034.txt`.

[8] B. Laurie, G. Sisson, R. Arends, and D. Blacka, „DNS Security (DNSSEC) Hashed Authenticated Denial of Existence." RFC 5155 (Proposed Standard), Mar. 2008. `http://www.ietf.org/rfc/rfc5155.txt`.

[9] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, „Protocol Modifications for the DNS Security Extensions." RFC 4035 (Proposed Standard), Mar. 2005. Updated by RFCs 4470, 6014, `http://www.ietf.org/rfc/rfc4035.txt`.

[10] O. Kolkman and R. Gieben, „DNSSEC Operational Practices." RFC 4641 (Informational), Sept. 2006. `http://www.ietf.org/rfc/rfc4641.txt`.

[11] O. Kolkman, „DNSSEC HOWTO, a tutorial in disguise." http://www.nlnetlabs.nl/publications/dnssec_howto/.

[12] C. Liu and P. Albitz, *DNS and BIND*. fifth ed., 2006. pp. 322–347, http://oreilly.com/catalog/9780596100575.

[13] „OpenDNSSEC, an open-source turn-key solution for DNSSEC." http://www.opendnssec.org/.

[14] „OpenDNSSEC architecture." http://trac.opendnssec.org/wiki/Signer/Architecture/.

[15] S. Morris, J. Ihren, and J. Dickinson, „DNSSEC Key Timing Considerations," Oct. 2010. http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing-01.

[16] J. Jansen and W. Wijngaards, „Response differences between nsd and other dns servers," Sept. 2006. http://www.nlnetlabs.nl/downloads/nsd/differences.pdf.

[17] „Wireshark, the world's foremost protocol analyzer." http://www.wireshark.org/.

[18] S. Josefsson, „Storing Certificates in the Domain Name System (DNS)." RFC 4398 (Proposed Standard), Mar. 2006. http://www.ietf.org/rfc/rfc4398.txt.

9. fejezet

Mellékletek

9.1. kasp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<KASP>
                <Policy name="ppke.hu">
                <Description>The default signing policy
     for ppke.hu zone</Description>
                <Signatures>
                        <Resign>PT3H</Resign>
                        <Refresh>P7D</Refresh>
                        <Validity>
                                <Default>P14D</Default>
                                <Denial>P14D</Denial>
                        </Validity>
                        <Jitter>PT12H</Jitter>
                        <InceptionOffset>PT3600S</InceptionOffset>
                </Signatures>

                <Denial>
                        <NSEC3>
                                <OptOut/>
                                <Resalt>P14D</Resalt>
                                <Hash>
                                        <Algorithm>1</Algorithm>
                                        <Iterations>5</Iterations>
                                        <Salt length="8"/>
                                </Hash>
                        </NSEC3>
                </Denial>

                <Keys>
```

```
        <!-- Parameters for both KSK and ZSK -->
        <TTL>PT3600S</TTL>
        <RetireSafety>PT3600S</RetireSafety>
        <PublishSafety>PT3600S</PublishSafety>
        <ShareKeys/>
        <Purge>P14D</Purge>

        <!-- Parameters for KSK only -->
        <KSK>
                <Algorithm length="2048">10</Algorithm>
                <Lifetime>P1Y</Lifetime>
                <Repository>SoftHSM</Repository>
                <Standby>0</Standby>
        </KSK>

        <!-- Parameters for ZSK only -->
        <ZSK>
                <Algorithm length="1024">10</Algorithm>
                <Lifetime>P1M</Lifetime>
                <Repository>SoftHSM</Repository>
                <Standby>0</Standby>
                <!-- <ManualRollover/> -->
        </ZSK>
</Keys>

<Zone>
        <PropagationDelay>PT9999S</PropagationDelay>
        <SOA>
                <TTL>P1D</TTL>
                <Minimum>PT3600S</Minimum>
                <Serial>datecounter</Serial>
        </SOA>
</Zone>


<Parent>
        <PropagationDelay>PT9999S</PropagationDelay>
        <DS>
                <TTL>P1D</TTL>
        </DS>
        <SOA>
                <TTL>PT10800S</TTL>
```

```
                                <Minimum>PT10800S</Minimum>
                        </SOA>
                </Parent>

                <Audit>
                         <Partial />
                </Audit>

        </Policy>
</KASP>
```