SICUREZZA NELLE RETI a. a. 2012/13 – Riassunto DNS/DNSSEC

by Salvatore Fresta

Il **Domain Name System** (DNS) è un sistema per la risoluzione dei nomi di rete. Molto spesso, per diverse ragioni, si preferisce allestire due server DNS:

- **Esterno**: riceve query da utenti esterni per la risoluzione di nomi riguardandi gli host pubblicamente accessibili dalla rete interna.

- **Interno**: riceve query da utenti interni sia per la risoluzione di nomi riguardanti host interni sia per quelli su Internet. Per le query che non è in grado di risolvere, contatta altri server DNS mediante **query ricorsive**.

La separazione ha dei **vantaggi**. Per esempio in caso di **compromissione** del server esterno, la rete interna rimane protetta. Inoltre i due server contengono **informazioni differenti**, uno quelle pubbliche e l'altro quelle private.

Vediamo come avviene la **risoluzione** di una query DNS.

1. Il client vuole connettersi a **www.esempio.com** ed invia una richiesta al server DNS configurato.

2. Il server DNS controlla se contiene l'informazione richiesta ed in tal caso risponde. Se **non** la contiene, la query diventa ricorsiva. In questo caso consulterà l'elenco dei **root server** (domini di più alto livello, in questo caso .com) che conosce e fa partire la query ricorsiva.

3. Ovviamente il root server non può sapere la risoluzione di tutti i domini *.com* possibili ed immaginabili (saranno milioni), quindi quello che fa sarà rispondere con un record di tipo **NS** (Name Server) comunicando un elenco di server **Global Top Level Domain** che hanno maggiori informazioni su quel dominio.

4. Tutta la prassi al punto 3 si ripete fin quando non si trova il server **autoritativo**, cioé uno dei server che sa per certo qual'è l'indirizzo IP corrispondente a www.esempio.com. La risoluzione viene conservata nella **cache locale** del server DNS per un tempo pare al TTL comunicato.

CACHE POISONING

Le comunicazioni cui sopra avvengono utilizzando il protocollo **UDP** e la porta standard dei server DNS è la **53**. Di fatto però non esiste nessun meccanismo di autenticazione delle risposte di un server DNS, quindi come si fa ad essere certi che un pacchetto in arrivo è la risposta alla richiesta appena fatta? Si usano 4 regole per identificare la risposta:

1. La **porta sorgente** utilizzata nel pacchetto di richiesta dev'essere uguale in quello di risposta. Ovvero se io invio una richiesta di risoluzione DNS al mio server alla porta 53 utilizzando la

porta sorgente 2000, questo, quando mi risponderà, dovrà inserire come porta sorgente del suo pacchetto di risposta, il valore 2000.

2. Il campo **question** in entrambi i pacchetti deve coincidere.

3. Il campo **query ID** (o transaction ID) in entrambi i pacchetti deve coincidere.

4. La risposta deve contenere **dati veritieri**, nel senso che se richiedo la risoluzione di un dominio .net ad un server DNS per i .com e questo mi risponde, l'informazione sicuramente sarà falsa e dovrà essere scartata.

Se un attaccante riesce a prevedere **tutti** questi dati, può alterare la cache DNS.

Se il query ID è un semplice contatore, è facile prevederlo. Stessa cosa se si usa sempre la stessa porta sorgente. Quindi è normale intuire come l'imprevedibilità di questi valori è fondamentale.

Randomizzare il query ID è una delle difese principali. Anche se lungo solo **16 bit,** provare tutte le combinazioni significa provare 65536 valori (2^{16}). Bisogna provarne statisticamente la metà prima che il server autoritativo risponda.

ATTACCO DI DAM KAMINSKY

È **un'evoluzione** insidiosa del cache poisoning. L'idea è pressoché la stessa solo che l'attacco da poco probabile diventa molto probabile.

La differenza sostanziale è che la **richiesta** che fa partire il meccanismo di risoluzione del nome dominio parte dalla **macchina attaccante**. Generare 65K valori è difficile ma siccome questa volta è l'attaccante che genera le richieste, può generare tanti nomi casuali da risolvere e di conseguenza tantissime query una dietro l'altra:

123.esempio.com
1234.esempio.com
12345.esempio.com
ecc.. ma **tutti diversi** per non essere memorizzati nelle cache

Quindi siccome il processo di risoluzione è ripetuto tantissime volte, diventa probabile indovinare l'ID.

Per esempio, se riesce a fare solo 50 risposte finte prima di essere superato dal vero sever autoritativo, **può ripetere questo tentativo tante volte** con nomi diversi. In questo esempio la probabilità di riuscita è:

$$\frac{50}{65536}$$

ma con 10000 tentativi, la probabilità di successo sale a 99.9% !!!

Ovviamente in questo esempio abbiamo sottointeso che solo il query ID fosse random. Una difesa sarebbe quella di **randomizzare anche la porta UDP**, aumentando così il range di possibili valori.

DNSSEC

Per far fronte ai problemi nativi di DNS, si è deciso di realizzare un'estensione retrocompatibile che fornisse **integrità** e **autenticazione** alla comunicazione. Il nome di quest'estensione è **DNSSEC**.

In realtà, a causa di problematiche tecniche e logistiche, DNSSEC non ha riscontrato un grande successo e l'adozione di questo sistema ha subito un forte rallentamento.

L'idea alla base di DNSSEC è **firmare digitalmente le risposte dei server DNS autoritativi**. La chiave pubblica di una zona viene distribuita da quella gerarchicamente superiore fino ad arrivare ai root server. Il problema è che la gestione di tale sistema è tutt'altro che banale, soprattutto l'aggiornamento delle chiavi per evitare attacchi di replay.

Oltre ai problemi di gestione sussistono anche **problemi di efficienza**. È impensabile utilizzare la crittografia in ogni pacchetto.

Non c'è confidenzialità dei dati ma solo integrità e non esistono protezioni contro attacchi **DOS** (se un pacchetto viene modificato viene scartato, ma di fatto il client non riceve nessuna risposta, quindi DOS). Inoltre l'attacco potrebbe trasformarsi da DOS a **DDOS** in quanto una query di 78 byte si può trasformare in una risposta da 3113 byte!

Come già detto, l'integrità è fornita per le risposte dei server autoritativi, mentre **i glue record rimangono falsificabili**.