Esercitazione di Laboratorio - NA04

Riccardo Persello

22 giugno 2021

DNS

Si eseguano le query DNS indicate nel seguito mentre è attiva la cattura di pacchetti tramite sniffer. Si analizzino i messaggi di query DNS e relative risposte.

```
C:> nslookup
> set type=A
> www.uniud.it
> set type=A
> uniud.it
> set type=MX
> uniud.it
> exit
```

1. **Si descrivano i messaggi intercettati relativi a tali interrogazioni.**

   Le query analizzate consistono in un messaggio UDP diretto al default gateway (alla porta 53, quella caratteristica del servizio DNS) e la relativa risposta.

   In un header DNS (sia query che risposta), troviamo all'inizio un campo denominato *Transaction ID*, il quale consente di abbinare una risposta alla relativa query (ricordare che si sta utilizzando un messaggio UDP, quindi non connesso, e alcune query e risposte potrebbero accavallarsi).

   Immediatamente dopo, troviamo due otteti di flag: da questi si può capire se il messaggio è una query o una risposta (`QR`), un *opcode* per identificare il tipo di operazione (in questo caso una *Standard query (0)*), opzioni per il troncamento, query ricorsive (unica opzione attiva nella cattura effettuata) ed autenticazione. Gli ultimi 4 bit sembrano essere inutilizzati nelle query, in quanto contengono un *Reply Code* inviato soltanto nelle risposte. Un valore pari a zero indica l'assenza di errori.

Altri due byte contengono il numero di query, successivamente, altri due indicano il numero di risposte (ovviamente impostato a zero nelle query).

Nella query, si vede del testo ASCII che riporta esattamente il nome del server che abbiamo indicato in `nslookup`, seguito da due byte per indicare il tipo di query (`0x01`, ovvero una query di tipo `A`, address), ed altri due byte riportanti la classe (`0x01`, classe `IN`, internet).

Nel pacchetto di risposta, oltre ad avere dei flag diversi ed un numero non nullo di risposte (quando l'operazione va a buon fine), a seguito del campo contenente la richiesta effettuata, riportato esattamente come nella query, troviamo il campo (i campi, quando una query è multipla) che riportano il risultato della richiesta.

Per prima cosa, è indicato il nome del server a cui si riferisce quella specifica risposta, ma visto che è già stato scritto nel pacchetto (nel campo relativo alla query), viene riportato come un puntatore (in questo caso `00c`, che se considerato in relazione all'inizio della trama DNS, corrisponde all'inizio della stringa `www.uniud.it`). Successivamente, come visto in precedenza, si trovano il tipo e la classe della query. Andando avanti, si trovano un valore di TTL (in questo caso 8 secondi), la lunghezza dei dati in risposta (4 byte), ed infine l'indirizzo IP corrispondente al nome richiesto (`9e 6e 03 2e`, 158.110.3.46).

La seconda query è sostanzialmente identica alla prima, cambia solo il nome contenuto nella richiesta (`uniud.it`), e l'indirizzo comunicato in risposta (158.110.3.47). Anche il *Transaction ID* è variato (ovviamente), così come il TTL.

Per quanto concerne la terza query, di tipo MX, varia soltanto il campo *Type*, ora impostato a `0x000F (15)`. Nella risposta, invece di trovare un indirizzo IP, si trova una stringa ASCII riportante il seguente nome: `uniud-it.mail.protection.outlook.com`. Ovviamente, come in precedenza, si assume ovvio che il *Transaction ID* ed il TTL possano essere differenti.

FTP

Si eseguano una connessione a un server FTP e alcuni comandi mentre è attiva la cattura di pacchetti tramite sniffer. Il client FTP può essere un browser, utilizzando come indirizzo FTP://allegro.diegm.uniud.it, un client dedicato (per esempio WinSCP) oppure un client a riga di comando (utilizzando il comando ftp in ambiente Windows o Linux).

Si analizzino i messaggi del protocollo.

Dati per l'accesso:
server: allegro.diegm.uniud.it
user: demo_reti
password: segreto!

1. **Si visualizzi la lista dei file contenuti nella home directory dell'utente `demo_reti` e si descrivano relativi i messaggi intercettati.**

   Innanzitutto, si nota che il protocollo si basa su una connessione TCP alla porta 21 del server. Si vede anche che tutti i pacchetti FTP sono in chiaro (nome utente e password visibili). I comandi inviati al server sono scritti in caratteri ASCII, terminati da `\r\n` e composti da un comando (scritto in maiuscolo), seguito da uno o più argomenti. Le risposte presentano un numero di tre cifre seguite da una spiegazione leggibile del messaggio rappresentato.

   Per prima cosa, al momento della connessione, il server FTP risponde con il codice 220 (*Service ready for new user.*), indicando alcuni dati tra cui versione ed autore del server.

   Segue un comando da parte del client: `USER demo_reti\r\n`, in cui si richiede al server di iniziare una sessione con il nome utente indicato.

   Il server risponde con il codice 331 (*User name okay, need password.*), al quale il client replica con `PASS segreto!\r\n`. La risposta successiva, 230, annuncia che l'utente è stato autenticato e può procedere.

   Dopo aver richiesto la lista dei file presenti nella cartella home dell'utente specificato, mediante il comando `ls`, il client richiede al server, mediante il comando `PORT 172,20,10,2,209,52\r\n`, di aprire una connessione TCP (per il flusso di dati) con l'host 172.20.10.2 (IP locale) alla porta 53556. La porta lato server per il flusso di dati FTP è la 20, quella immediatamente precedente rispetto a quella usata per l'invio di comandi. Il server risponde con il codice 200, indicando che l'azione richiesta è stata completata correttamente. Ora abbiamo una connessione TCP attiva anche sulla porta 21 del server.

   Il client ora invia il comando `LIST\r\n`, con cui richiede di ricevere la lista dei file del direttorio selezionato. Il server risponde con il codice 150, avvisando il client che il comando richiesto è in esecuzione e di attendere altre risposte dal server.

   Successivamente, sulla connessione FTP-DATA precedentemente stabilita tra le porte 53556 del client e 20 del server, quest'ultimo invia la stringa `-r--r--r-- 1 ftp ftp 80 Jun 05 2012 welcome.txt\r\n`, con cui indica la presenza di un file chiamato

"welcome.txt", con i relativi permessi, dimensione e data di creazione. Questa stringa, così come le risposte precedenti, viene riportata integralmente sul terminale del client. Alla fine di questa trasmissione, sulla connessione di controllo stabilita sulla porta 21, il server invia una risposta con codice 226, con cui indica che il trasferimento è avvenuto con successo.

2. **Si effettui la copia sul vostro disco locale del file che si trova in tale direttorio e si descrivano relativi i messaggi intercettati.**

   Dopo aver eseguito il comando `get welcome.txt` dal client, questo invia al server un nuovo comando `PORT`, simile al precedente, con cui richiede l'avvio di una nuova connessione per il trasferimento dati (sempre sulla porta 20 del server). Come prima, il server risponde affermativamente, avvisando che la connessione è stata attivata.

   Il client invia il comando `RETR welcome.txt\r\n` (retrieve). Il server risponde con il codice 150, che in questo contesto indica la validità della richiesta e di restare in attesa di comunicazioni successive.

   Sulla connessione per il flusso di dati, viene inviata dal server la seguente stringa: `\r\nBenvenuti nel sito FTP allestito per l'esercitazione di Reti di Calcolatori.\r\n`, ovvero il contenuto del file richiesto. Segue conferma di trasferimento sulla connessione di controllo (codice 226).

   Al momento della disconnessione, il client invia il comando `QUIT`, a cui il server risponde con 221 (*Goodbye.*).

HTTP

Si apra una pagina web accessibile con protocollo http (non https) mentre è attiva la cattura di pacchetti tramite sniffer e si descrivano relativi i messaggi intercettati. Esempio di sito raggiungibile in http: `www.montessoro.it`.

Si avvia una cattura Wireshark, e si naviga con un browser al sito indicato.

Per prima cosa, si notano dei pacchetti TCP atti all'apertura della connessione con il server all'indirizzo 109.168.121.241. Successivamente, si nota una richiesta HTTP.

Questa richiesta è contenuta nel payload di un segmento TCP. È chiaramente espressa in caratteri ASCII:

```
GET / HTTP/1.1
Host: www.montessoro.it
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
            AppleWebKit/> 605.1.15 (KHTML, like Gecko)
            Version/14.1.1 Safari/605.1.15
Accept-Language: fur-latn-it
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Si può notare che nella prima riga vi è indicato il tipo di richiesta (`GET`), il file desiderato (`/`, corrispondente all'indice), l'host, la preferenza del client riguardo all'uso di connessioni crittografate (`Upgrade-Insecure-Requests`), i tipi di contenuto che il client è in grado di interpretare (`Accept`), il tipo di client, browser, piattaforma ed estensioni (`User-Agent`), la lingua preferita (`Accept-Language`), le tecniche di (de)compressione supportate dal client (`Accept-Encoding`), e la preferenza da parte del client di riutilizzare o meno la connessione TCP già aperta anche per le richieste e risposte successive (`Connection`), assieme ad eventuali altri parametri come *timeout* o numero massimo di richieste per quella connessione. Segue un ACK da parte del server.

Il server invia quattro segmenti TCP, che una volta riassemblati, presentano una risposta alla precedentemente descritta richiesta HTTP:

```
HTTP/1.1 200 OK
Content-Length: 4895
Content-Encoding: gzip
Server: Apache
ETag: "6dc15-41cc-84919fc0"
Accept-Ranges: bytes
Content-Type: text/html
Age: 224
Date: Tue, 22 Jun 2021 15:20:07 GMT
Last-Modified: Wed, 08 Jul 2020 13:59:35 GMT
Connection: keep-alive

--- SEGUE SORGENTE HTML ---
```

In questa risposta possiamo notare un codice (200) indicante il successo dell'operazione. Nell'header HTTP sono riportati numerosi

dettagli sul contenuto, tra cui lunghezza, codifica, tipo di server, tipo di contenuto, data, data di ultima modifica, ...

Come per l'header della richiesta, si tratta di coppie chiave-valore separate da due punti. A seguire, si trova il sorgente HTML della pagina, non riportato a causa della sua lunghezza. Questo è un file di markup che definisce i contenuti della pagina.

Una pagina web non è solo un documento HTML: può essere composta da script, fogli di stile, immagini ed altri contenuti multimediali. Svuotando la cache del browser e ricaricando la pagina, si nota che viene effettuata una richiesta HTTP per ogni file presente nella pagina. Per prima viene scaricato il file HTML, e successivamente, una volta trovati dei contenuti "esterni", il browser procede ad effettuare le richieste HTTP per l'ottenimento deldi tutte le immagini contenute nella pagina.

Ad ogni risposta HTTP segue un ACK del client.

La sorgente del documento è disponibile al seguente indirizzo: https://github.com/persello/esercizi-rdc. Documento generato con pandoc.