VPN

23/05/2023

Le VPN oggi hanno cambiato la visione del mondo in ambito lavorativo.

Si pensi ad esempio ad un'azienda con più sedi dislocate nel territorio, oppure ai dipendenti che lavorano comodamente in smart working comodamente dalla propria abitazione.
In tutti questi casi è bene garantire la sicurezza di tutte queste sottoreti.

Si parla di *Virtual Private Network* poiché la rete fisicamente non è privata ma virtualmente si.

Allo stesso tempo però, si vuole permettere traffico verso l'esterno.
Inoltre sarebbe bello farlo in maniera efficiente (senza ritardi) e trasparente

Noi useremo **IPsec**: uno dei possibili modi per realizzare una VPN.
[RFC 1825] é uno standard che lavora a livello IP che si pone in mezzo tra IP e i protocolli sopra.

Come si fa? Inserendo nell'header del pacchetto dei bit appositi.

Complessità: si aggiunge cifratura ed autenticazione del PDU (Protocol Data Unit)...che però non intaccano l'header IP che é leggibile e quindi legittimo/gestibile dalla rete.

In alcune versioni la cifratura riguarda anche l'header (tra cui IP sorgente/destinazione) e quindi ciò comporta il mascherare il traffico.

IPsec può funzionare in due modalità:

- ***Transport mode:*** cifratura/autenticazione su computer a livello locale
  - Il pacchetto viaggia da end-point ad un altro
  - Richiede una speciale configurazione dell'host (non trasparente) :(
  - Protegge in locale :)
- ***Tunnel mode:*** cifratura/autenticazione su Firewall o Router; si parla di *Terminatore VPN*
  - Il pacchetto prima di raggiungere l'end-point di destinazione, raggiunge (in chiaro) il Termimatore VPN che successivamente lo cifra.
  - É trasparente :)
  - Non mi protegge in locale :(

Le VPN possono essere annidate per una maggiore sicurezza (ad esempio combinando *Transport* e *Tunnel*)

IPsec prevede due standard:

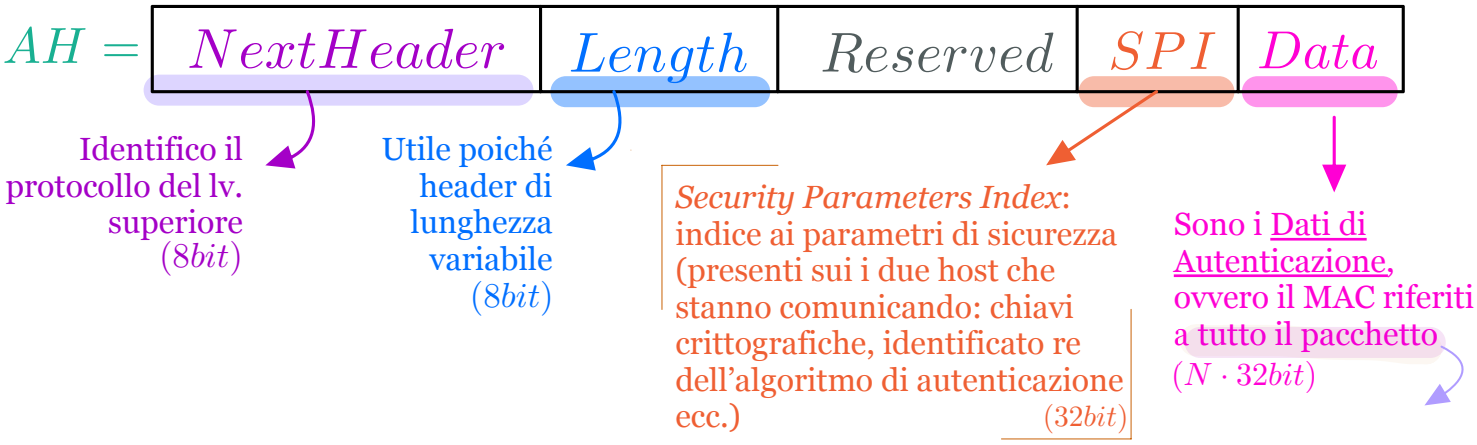
- **Authentication Header (AH)** [RFC 1826] che svolge l'autenticazione
- **Encapsulation Security Payload (ESP)** [RFC 1827] che svolge la cifratura (e autenticazione)

Cifratura e Autenticazione sono simmetrici

Authentication Header (AH)

Si inserisce tra il livello IP e il livello di trasporto.

Il suo header contiene:



La domanda sorge spontanea: **cosa autentico?**

N.B.

- Ci sono dei valori all'interno dell'header IP che cambiano ad ogni hop come ad esempio il TTL.
- Il campo '*Data*' dell'AH non é settato (ovviamente) prima dell'autenticazione

L'autenticazione deve quindi tenere conto di questi campi.

Soluzione: ai soli fini dell'autenticazione tali campi vengono settati a 0…per poi ripristinarne i valori.

Encapsulation Security Payload (ESP)

Il suo header contiene:

$$ESP = SPI + PDU + NextHeader + Autenticazione$$

In CHIARO

Che il destinatario utilizzerà per decifrare il pacchetto andando a vedere i dati dell'algoritmo memorizzati in loco

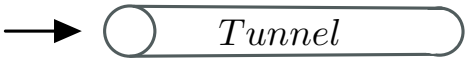
In cui é presente una *Tabella delle associazioni* che associa la VPN al particolare algoritmo e alla particolare chiave

CIFRATO!

Nel *tunnel mode* si aggiunge e cifra anche un così detto HeaderIP incapsulato...

ESP permette anche l'autenticazione (volendo potremmo fare a meno dell'AH!

MA questo non si fa poiché ESP autentica solo header ESP ed il suo payload.
Mentre AH autentica anche l'header IP



Questo è necessario poiché il Tunneling svolge una operazione molto simile al NAT: nasconde l'IP interno, all'interno di una cifratura.

Mentre l'headerIP esterno all'EIP rappresenta l'IP pubblico (non cifrato)

Extra: quando NAT & VPN sono entrambe in esecuzione, nascono dei problemi che é bene tenere a mente... ma che oggigiorno sono stati ampiamente risolti

---

Sia in AH che in ESP é presente anche il *Sequence Number*.
Questo potrebbe sembrare strano visto che siamo a livello IP (e non TCP!).

La risposta é presto detta: si vuole evitare la *duplicazione di pacchetti*, poiché non posso affidarmi a quello che (eventualmente) c'è sopra.
Si parla dunque di **IPSec anti-replay.** Seq relativi al numero di pacchetti (e non i byte come in TCP)!

Più nello specifico si utilizza una *Sliding Windows W* di un certo numero di bit (e quindi con *bit vector* in cui segnare i pacchetti ricevuti della finestra)...

Idea:
- Se arriva un pacchetto antecedenti alla finestra: lo butto, sperando che non sia più rilevante
- Se arriva una pacchetto della finestra con bit già settato: lo scarto; replay!
- Se arriva un pacchetto successivo alla finestra: faccio uno shift della finestra!