

# vulnerabilità delle reti

# reti e protocolli vulnerabili

- gran parte delle vulnerabilità delle reti sono in realtà **vulnerabilità dei protocolli**
- inserire la sicurezza in un protocollo significa costringere tutte le implementazioni a realizzare quelle funzionalità
  - costoso, inefficiente, non sempre necessario, ecc.

# vulnerabilità apparati

- gli apparati possono essere vulnerabili
  - errata implementazione dei protocolli
    - tutti gli standard lasciano un margine di decisione al progettista
  - vulnerabilità del firmware
    - es. buffer overflow
- ci concentriamo su vulnerabilità delle reti non legate all'implementazione o al software

# protocolli in chiaro e non autenticati

- è facile con uno sniffer ricostruire le sessioni tcp e trovare password
  - es. Wireshark/Ethereal
  - esistono strumenti più sofisticati
    - `sudo apt-get install ettercap`
    - `sudo apt-get install dsniff`
- è facile inserire pacchetti sulla rete facendo credere che li ha inviati un'altra macchina

# reti locali vs. Internet

- protocolli vulnerabili -> minaccia solo se l'hacker è “presente” su una lan per cui passa il traffico vulnerabile
- “presente” significa
  - presenza fisica (collegamento ethernet o wifi)
  - controllo remoto di una macchina (win, unix, apparato di rete)
- zone critiche:
  - la lan dell'utente
  - una lan di una server farm (es. web hosting)
  - una lan di un ISP intermedio

# fiducia e sniffabilità

- questa vulnerabilità sono minacce solo se...
  - non ci fidiamo dell'ambiente
  - il traffico è “sniffabile”
- sniffare su una lan
  - facile nelle reti vecchie: 10base2, hubs
  - leggermente più complesso per reti switched

# reti switched

- le tecnologie per reti locali nascono come inerentemente broadcast
  - legacy, ora le reti sono tutte switched
- credenza popolare
  - in una rete switched non si può sniffare quasi nulla
- purtroppo le reti switched sono molto vulnerabili

# mac flood

- quando uno switch satura la sua source address table si comporta come un hub
  - a questo punto lo sniffer vede tutto
- molto invasivo
  - alcuni switch vanno in crash
  - le spie dello switch segnalano traffico molto intenso

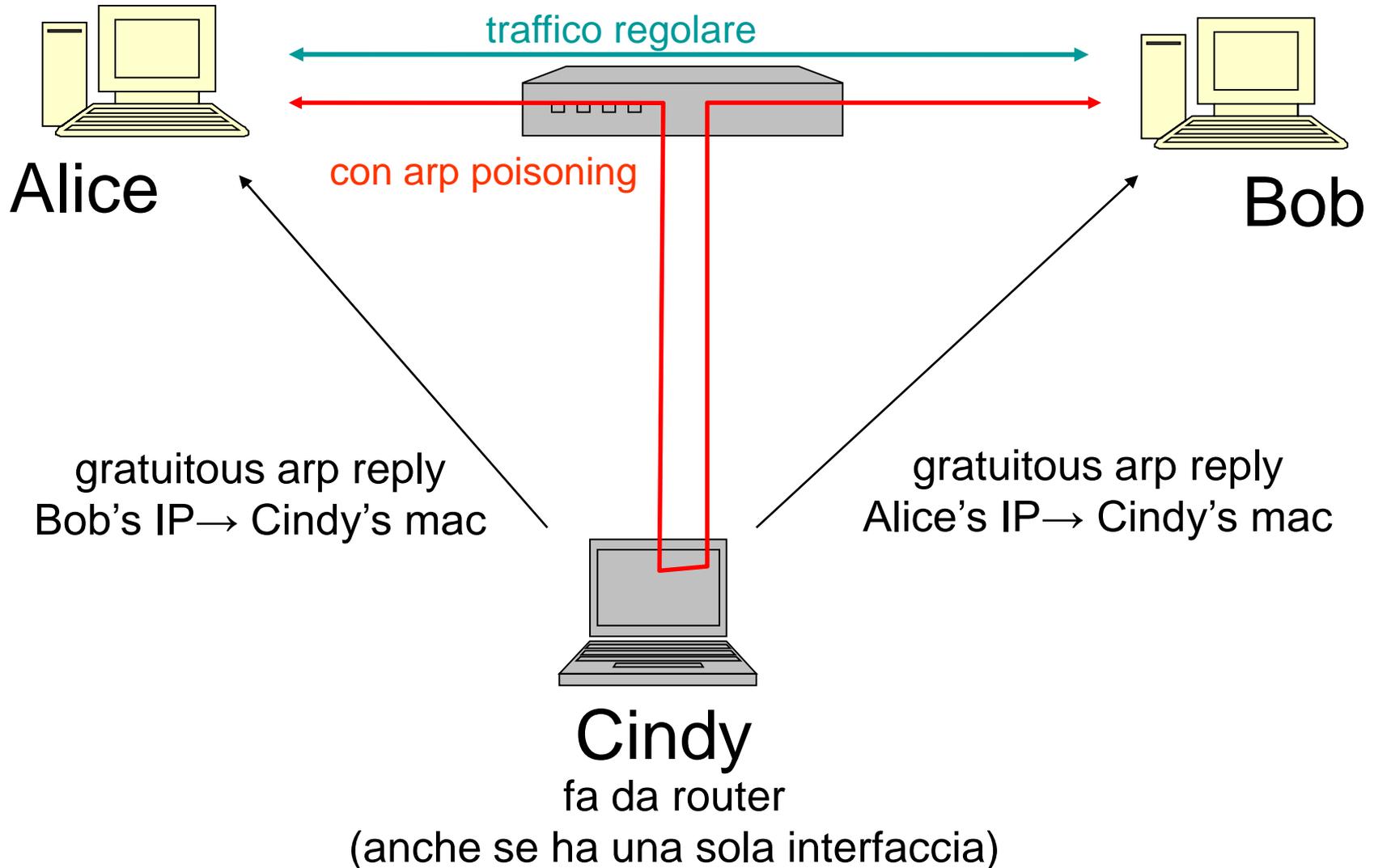
# terminologia

- *spoofing o forgery*: i termini sono usati quasi indifferentemente per denotare cambiamenti illeciti di dati
  - tipicamente in rete
  - header/campi di pacchetti o di messaggi
    - es. l'indirizzo sorgente
- *poisoning*: alterazione di una cache o altro stato

# arp poisoning (o spoofing)

- le implementazioni di arp sono stateless
  - da standard, praticamente tutte
  - aggiornano la arp cache ogni volta che ricevono un'arp reply... anche se non hanno inviato alcuna arp request!
- si può “avvelenare” la arp cache inviando delle arp reply “gratuite”
  - è visibile dalla macchina avvelenata (arp -a)
- le entry statiche risolvono il problema
  - ma rendono la vita impossibile!

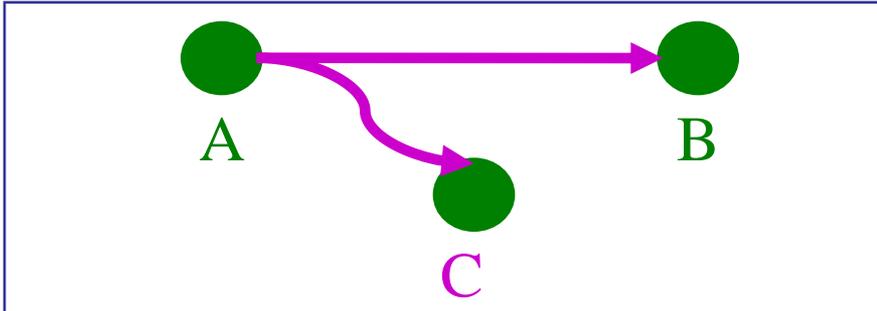
# arp poisoning



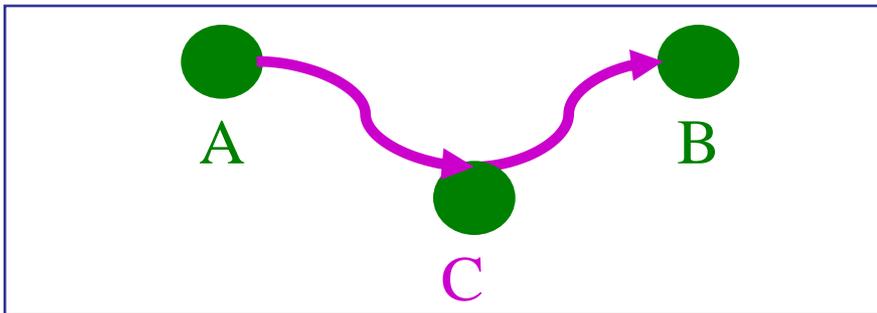
# moving to ipv6

- well... sooner or later ARP will be a legacy protocol
- ipv6 use Neighbor Discovery (ND)
  - rfc 4861
- same behavior
  - Neighbor solicitation  $\equiv$  ARP request
  - Neighbor advertisement  $\equiv$  ARP reply
- cached, stateless request
- same security problems of ARP
- SeND – **secure** neighbor discovery

# attacchi Man in the Middle (MitM)



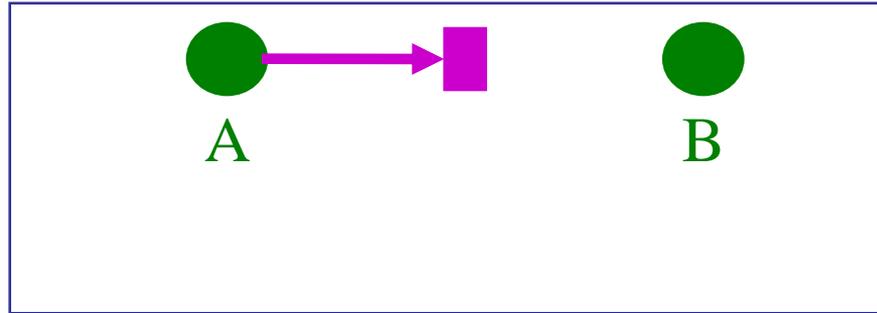
MitM passivo  
arp poisoning + sniffer



MitM attivo  
arp poisoning + sniffer  
+ altro

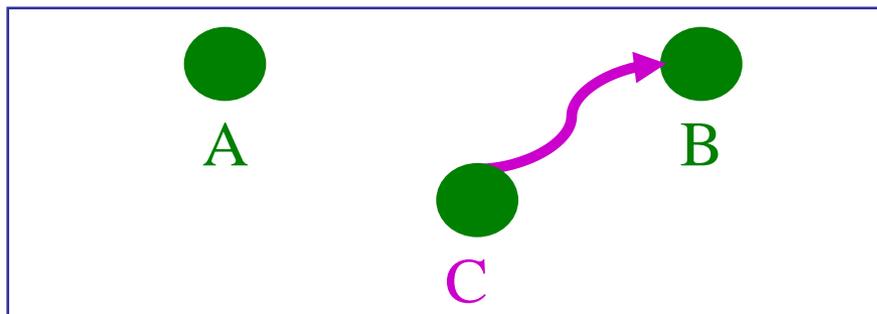
MitM attivo è semplice per protocolli udp based  
richiede un lavoro complesso su protocolli tcp based  
(gestione dei numeri di sequenza)

# denial of service (DoS)



- DoS su tutta la LAN
  - raramente è fatto saturando la rete
  - più facile saturare risorse di calcolatori
  - broadcast storm
    - per le macchine ogni broadcast ricevuto è un interrupt
- si può inibire una singola macchina con arp poisoning
  - es. dirottare tutto il traffico senza instradarlo a destinazione
  - o instradandolo selettivamente

# ip address spoofing

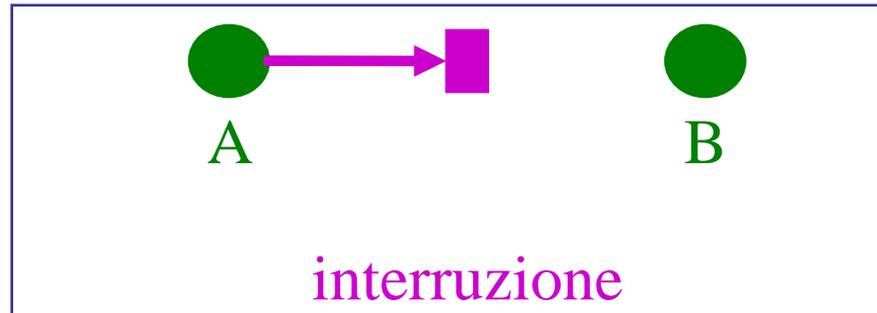


- l'indirizzo ip è facile da modificare in modo da impersonare una altra macchina
  - per mezzo di arp poisoning la macchina proprietaria dell'ip può essere neutralizzata (vedi DoS di una singola macchina)
- non riceveremo mai la risposta
  - a meno di arp poisoning di C su B
- anche su Internet
  - ma per ricevere la risposta bisogna attaccare il routing

# DoS con ping smurf

- interesse storico
- esempio di sfruttamento di IP spoofing per DoS
- C vuol fare DoS su A
- C invia un echo\_request con sorgente A (spoofed) e destinazione bcast (cioè “a tutte le macchine della sottorete”)
- tutte le macchine della sottorete risponderanno ad A con echo\_reply
  - non tutte le macchine rispondono ai ping bcast
  - la frequenza di echo reply ricevuti da A è (echo request inviati da C) \* (numero pc in subnet che rispondono al bcast)

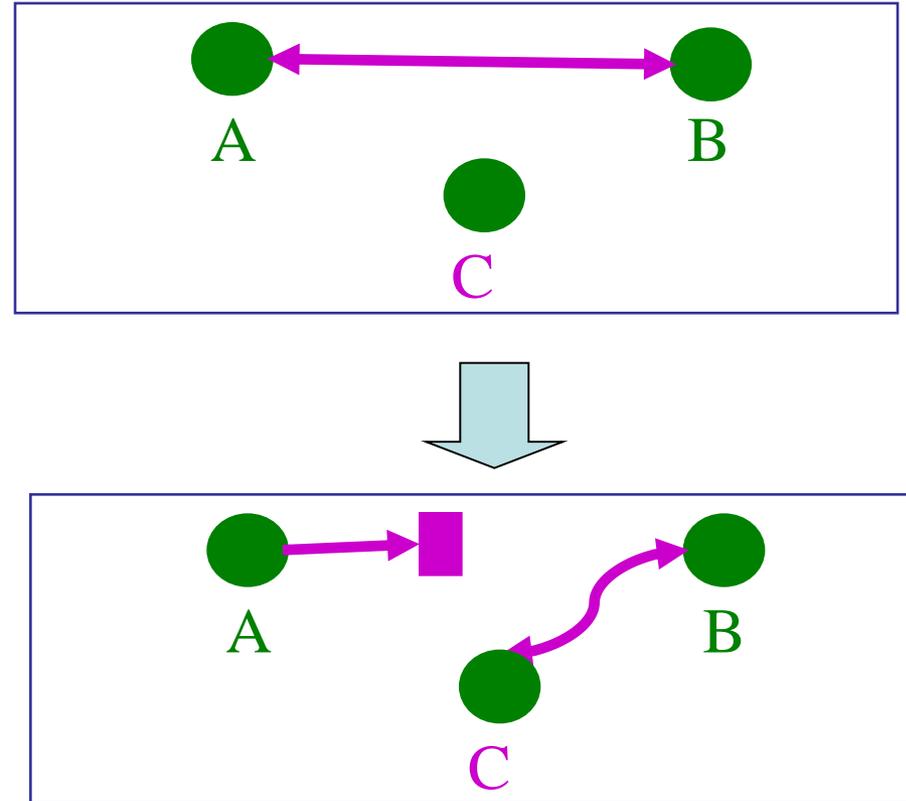
# tcp DoS (tcp reset)



- una sessione tcp attiva può essere “buttata giù”:
  - una delle due parti può essere “resettata”
- ecco come...
  - pacchetto tcp “forgiato” con la corretta quadrupla <ip, porta, ip, porta>
  - flag RST attivo
  - numero di sequenza scelto opportunamente
  - **funziona anche su Internet** poiché non ha bisogno di risposta

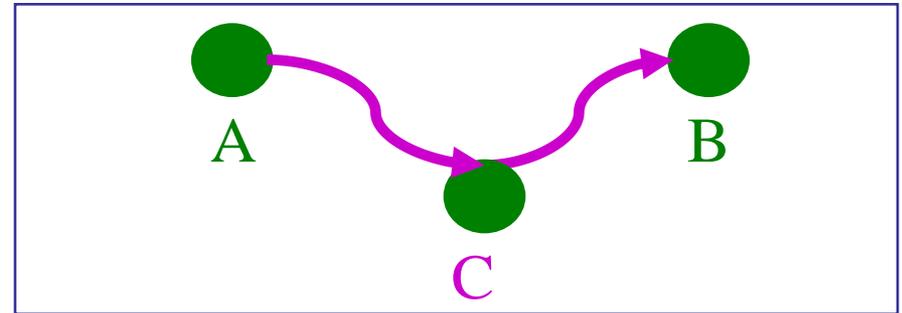
# tcp session hijacking

- l'obiettivo è di trasformare una sessione tcp tra A e B in una sessione tcp tra C e B senza che B se ne accorga

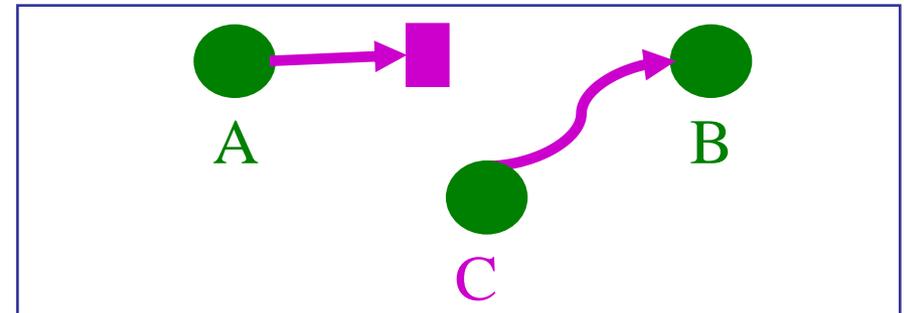


# tcp session hijacking

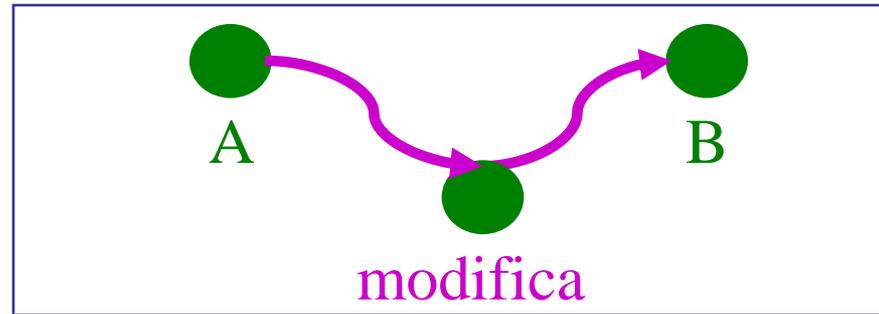
- fase1
  - MitM passivo
  - arp poisoning



- fase 2 (rubare la sessione)
  - A riceve un reset (tcp DoS su A)
  - C continua a fare arp poisoning su B
  - C continua la sessione tcp al posto di A continuando con i numeri di sequenza di A
  - B non si accorge del cambiamento di soggetto



# tcp MitM attivo



- il MitM può modificare singoli bytes dei pacchetti tcp (no inserimenti e cancellazioni)
  - nessun problema con i numeri di sequenza
- inserire nuovi bytes nel flusso tcp
  - B: manda ack per numeri di sequenza che la sorgente non ha inviato
  - gli ack possono essere droppati dal MitM
  - i numeri di sequenza nei pacchetti successivi possono essere slittati dal MitM (tecnica già usata da NAT per FTP)
- risincronizzazione
  - obiettivo: terminare l'arp poisoning ma mantenere la sessione  $A \leftrightarrow B$  attiva
  - modo1: perdendo alcuni caratteri della sorgente
    - dipende dal protocollo (ad esempio è possibile per telnet: l'utente non vede l'eco di alcuni tasti digitati, ma il problema è temporaneo)
  - modo2: inviando un TCP SYN
    - dipende dall'implementazione, oramai considerata una vulnerabilità

# vulnerabilità del DNS

- DNS non è autenticato
- spoofing: rispondere prima del server
  - tipicamente il dns molto lento
  - facile intercettare le richieste (ad es. con arp poisoning)
  - solo in rete locale
- è una delle minacce più pericolose per il web se non ci possiamo fidare del DNS...
- ...è molto importante autenticare il (web) server
  - vedi ssl nel seguito
  - oppure autenticare il DNS
    - DNSSEC: ancora poco diffuso

# Internet: vulnerabilità del DNS

- DNS cache poisoning o DNS pharming
  - obiettivo: modificare la cache di un DNS server, da un punto qualsiasi di Internet
  - target: sia DNS con open recursive service che stubs (only local recursive service)
- esempio
  - alcuni dns “creduloni” mettono in cache i record della “additional section” qualunque essi siano
  - applicabilità limitata a vecchissime installazioni bacate

# Internet: vulnerabilità del DNS

- un DNS riconosce la risposta quando....
  - IP sorgente corretto
    - spoofing
  - porta destinazione (e sorgente corretta)
    - random o fissa
  - ID della richiesta corretta
    - casuale, ma quanto casuale?
  - resource record corretto
    - richieste sollecitate dall'hacker
- l'hacker deve indovinare tutto
  - è un problema probabilistico
- l'attacco funziona se la probabilità di indovinare tutto è alta
  - normalmente è bassa ma....

# Internet: vulnerabilità del DNS

- Dan Kaminsky Febbraio-Luglio 2008
  - serie di osservazioni/tecniche che rendono l'attacco a qualsiasi DNS ricorsivo facile
- in dettaglio
  - ID della richiesta predicibili
    - vedi generazione di numeri casuali nella parte di crittografia
    - spazio degli ID piccolo, solo 16 bit
  - porta sorgente fissa
    - per alcune implementazioni
    - usare una porta random e considerarla ID è la contromisura standard, non sempre fattibile
  - annuncio: <http://www.kb.cert.org/vuls/id/800113>
  - l'analisi probabilistica è una variante del birthday attack (vedi parte di crittografia)

# Internet: vulnerabilità del DNS

- documentazione dell'impatto ad ottobre 2008
  - prodotti noti come vulnerabili 39%
  - prodotti noti come non vulnerabili 11%
  - prodotti per cui non è noto lo stato 50%

# Internet: Distributed DoS

- obiettivo: saturare le risorse del server costringendolo ad allocare un gran numero di connessioni tcp
- tipicamente tramite syn flood
- ip sorgente spoofed (casuali)
  - la sorgente non ha bisogno di ricevere il syn+ack
- alla ricezione del syn il server...
  - alloca i buffer
  - risponde SIN+ACK
  - attende un ACK
  - mantiene allocato lo spazio fino al timeout
- difficile da evitare
  - esistono delle tecniche interessanti per contenere il problema: fanno uso accorto delle risorse quando arriva un SYN

# Internet: tcp reset

- se le sessioni tcp sono critiche può essere molto pericoloso
- le sessioni più critiche sono quelle BGP
  - intere porzioni di internet diventano irraggiungibili
  - estensione MD5 cisco
    - sostanzialmente una estensione di tcp per autenticare l'header
    - metodo generale ma usata solo per BGP

# Internet: route hijacking

- annunciare nel protocollo di routing una rotta non propria
- se la rotta è 0.0.0.0/0 è detto black hole
- intra-dominio: OSPF, RIP, ecc.
  - un problema di sicurezza dell'ISP
- inter-dominio: BGP
  - nessuna autorità centrale (ICANN, regional IRR?)
  - fatto spessissimo con indirizzi non assegnati o rotte molto generiche, usate per spam e poi ritirate

# Internet: “biggest security hole”

- quando si fa routing hijacking non si può fare MitM
  - perché non si possono inviare pacchetti all’obiettivo, tornerebbero all’attaccante!
- ipotesi: route hijacking con BGP (interdominio)
  - bisogna controllare un router BGP collegato ad Internet
- l’idea: proteggiamo i router nel percorso dall’attaccante all’attacato dall’hijacking che stiamo facendo
  - Tony Kapela e Alex Pilosov (agosto 2008)

# Internet: the “biggest security hole”

- procedura
  1. traceroute verso il target
  2. trovare tutti gli AS dagli indirizzi ip del traceroute (facile con la RIB del router)
  3. fare l'hijacking inserendo nell'AS path tutti gli AS calcolati al punto 2
    - BGP loop avoidance fa si che tali AS non ricevano (o ignorino l'annuncio)
  4. Inserire una rotta statica verso il primo hop per il target
- “tutta Internet” manda il traffico all'attaccante ma l'attaccante può rigirarlo all'obiettivo