ITBH Technical White Paper
Italian Black Hats Association - Associazione Italiana Black Hats

Man in the Middle Attacks: cos'è, come ottenerlo, come prevenirlo, come sfruttarlo.

Versione 1.0

Marco Valleri (**NaGA**), laureando in informatica presso l'Università di Milano, lavora per Intesis gruppo Finmatica come Pen-Tester. Creatore del progetto "ettercap", partecipa all'iniziativa no-profit blackhats.it.

Alberto Ornaghi (**ALoR**), in tesi presso l'Universita' di Milano, dove sviluppa un protocollo alternativo ad ARP che previene gli attacchi "arp-spoofing".
Creatore del progetto "ettercap", partecipa all'iniziativa no-profit blackhats.it.

Entrambi sviluppatori in ambiente Free Software e particolarmente focalizzati sulle problematiche security related, sono membri di ITBH dal luglio 2002.

INDICE

Disclaimer

Le opinioni e le informazioni espresse nel presente documento appartengono agli autori e non ad aziende pubbliche o private: esse non rappresentano in alcun modo idee, politiche aziendali o servizi specifici se non il pensiero e l'esperienza degli autori stessi.

Il disclaimer standard si applica al presente documento, in particolare modo per la non responsabilità degli autori, verso qualunque tipo di danni - causati direttamente o indirettamente – conseguenti alla lettura del presente documento e/o all'utilizzo illegale o fraudolento delle informazioni e/o funzionalità ivi contenute.

Gli autori non si assumono alcuna responsabilità per i contenuti di questo documento – così come di eventuali errori od omissioni – o di qualunque documento, prodotto o servizio da esso derivati, indirettamente o meno.

Il presente documento può essere liberamente distribuito, pubblicato o copiato con ogni mezzo disponibile a patto che lo stesso non venga modificato in alcun modo e previa autorizzazione scritta degli autori.

E' assolutamente vietato "appropriarsi" della proprietà intellettuale dell'opera, ovverosia spacciarsi per gli autori, tradurlo in altre lingue appropriandosene la paternità o estrapolare singoli paragrafi spacciandosi per l'autore degli stessi.

Abstract

MAN IN THE MIDDLE ATTACKS.
(COS'E', COME OTTENERLO, COME SFRUTTARLO, COME PREVENIRLO)

In questo documento tecnico vengono esposte e commentate tecniche di attacco "mitm" riguardanti attacchi di tipo **Locale**, da **Locale a Remoto** e **Remoti**.

In particolare vengono trattati i seguenti argomenti:

- **LOCALE**

- arp poisoning
- dns spoofing

- **DA LOCALE A REMOTO**

- arp poisoning
- dns spoofing
- dhcp spoofing
- icmp redirection
- route mangling

- **REMOTO**

- dns poisoning
- traffic tunneling
- route mangling

Come tutti i Technical Paper dell'Associazione Italiana Black Hats, anche questo documento può essere letto dal punto di vista dell'attaccante o del gestore del sistema stesso: vale in ambo i casi la regola "conosci il tuo nemico prima di"…

Ogni qualvolta se ne presenterà la possibilità il presente documento verrà aggiornato, con le nuove scoperte o gli update rilevati da ITBH: ovviamente invitiamo i lettori a comunicarci eventuali errori, imperfezioni o aggiornamenti dei quali siano a conoscenza.

Buona lettura,

gli autori.

Introduzione

==========

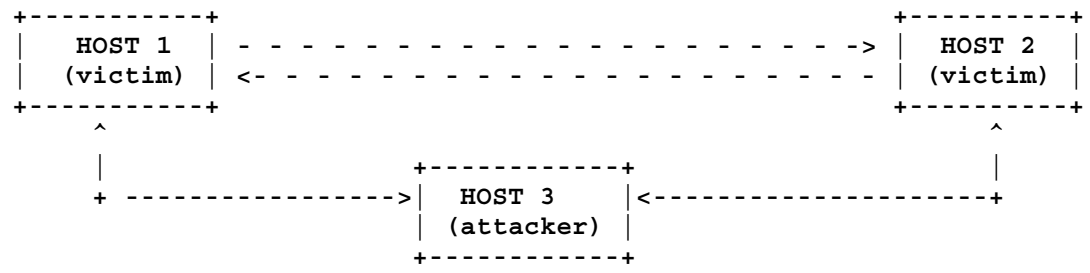
MAN-IN-THE-MIDDLE

* **cos'e'**
* **come ottenerlo**
* **come sfruttarlo**
* **come prevenirlo**

Alberto Ornaghi <alor@blackhats.it>
Marco Valleri <naga@blackhats.it>

==========

La tipologia di attacco che va sotto il nome di man-in-the-middle consiste nel dirottare il traffico generato durante la comunicazione tra due due host verso un terzo host (attaccante). Durante l'attacco e' necessario far credere ad entrambi gli end-point della comunicazione che l'host attaccante e' in realta' il loro interlocutore leggittimo.



LEGENDA

```
- - - ->    Connessione logica

------->    Connessione reale
```

L'host attaccante riceve quindi tutto il traffico generato dagli host 1 e 2 e si preoccupa di inoltrare correttamente il traffico verso l'effettiva destinazione dei pacchetti ricevuti.

==========

A seconda dello scenario in cui ci si trova ad operare l'attacco man in the middle assume forme diverse:

- **RETE LOCALE**
  - arp poisoning
  - stp mangling
  - traffic desync
  - dns spoofing

- **DA RETE LOCALE A REMOTO (attraverso un gateway)**
  - arp poisoning
  - traffic desync
  - dns spoofing
  - dhcp spoofing
  - icmp redirection
  - irdp spoofing
  - route mangling

- **REMOTO**
  - dns poisoning
  - traffic tunneling
  - route mangling

==========

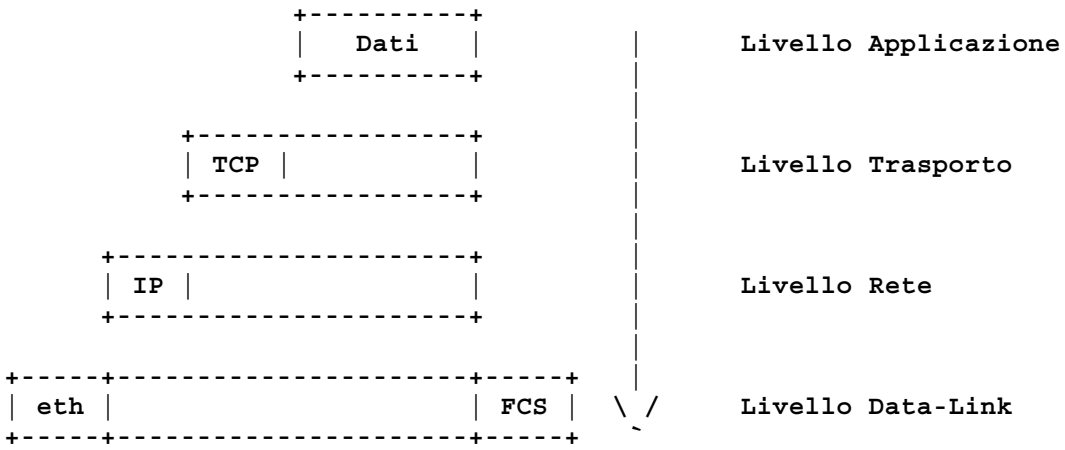
MAN-IN-THE-MIDDLE IN RETE LOCALE

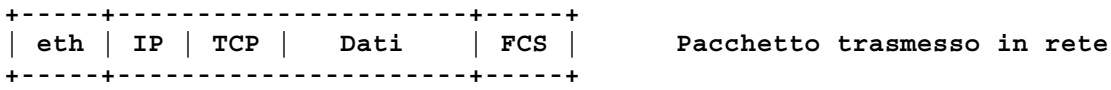
ARP POISONING

Il protocollo IP e' lo standard per le comunicazioni attraverso internet ed e' quindi utilizzato per la stragrande maggioranza delle comunicazioni locali e remote. Ethernet (802.3) e' lo standard piu' diffuso per la connettivita' locale per questo ci atterremo a questo standard negli esempi successivi.

Ethernet identifica le macchine non attraverso il loro indirizzo IP ma con un indirizzo univoco di 48 bit chiamato Media Access Controllesrs (MAC) address.

Una volta che un pacchetto IP arriva all'interno di una LAN deve essere convertito in un pacchetto che la LAN stessa possa gestire; per questo motivo il pacchetto IP viene incapsulato in un "frame" (ethernet, fddi etc).





Il protocollo ARP (Address Resolution Protocol) si occupa di "mappare" i 32 bit di indirizzo IP nei 48 bit di indirizzo MAC. Una volta che l'indirizzo MAC e' stato determinato, il pacchetto IP incapsulato puo' essere spedito all'host destinazione.

Nello standard Ethernet il protocollo ARP prevede due tipi di messaggi:

- ARP request: richiede il MAC address di un particolare indirizzo IP. Generalmente viene spedito in broadcast locale.

- ARP reply : viene spedito in risposta a una "request" dichiarando il MAC address dell'host che corrisponde all'indirizzo IP richiesto.

Un pacchetto ARP ha questa forma:

```
--- /usr/include/net/if_arp.h ---

struct arphdr
  {
    unsigned short int ar_hrd;      /* Format of hardware address.  */
    unsigned short int ar_pro;      /* Format of protocol address.  */
    unsigned char ar_hln;           /* Length of hardware address.  */
    unsigned char ar_pln;           /* Length of protocol address.  */
    unsigned short int ar_op;       /* ARP opcode (command).  */
#if 0
    /* Ethernet looks like this : This bit is variable sized
       however...  */
    unsigned char __ar_sha[ETH_ALEN];  /* Sender hardware address.  */
    unsigned char __ar_sip[4];         /* Sender IP address.  */
    unsigned char __ar_tha[ETH_ALEN];  /* Target hardware address.  */
    unsigned char __ar_tip[4];         /* Target IP address.  */
#endif
  };
```

I pacchetti arp sono di dimensione variabile a seconda del tipo di link su cui ci si trova, ma anche a senconda della versione IP che si utilizza.
La parte costante e' quella rappresentata dai primi 5 campi della strutturasopra citata.
Nel nostro caso IP (versione 4) + ethernet il pacchetto ha esattamente la questa struttura.

Ricapitolando dunque quando un host desidera comunicare con un altro il kernel si preoccupa di mandare una richiesta per poter mappare l'indirizzo IP destinazione su un indirizzo MAC.

Questa informazione viene memorizzata in una apposita tabella chiamata ARP CACHE.

COS'E' L'ARP CACHE

Essa e' stata progettata per ottimizzare le prestazioni di rete. Infatti, una volta richesto un MAC address relativo ad un IP questo rimane memorizzato in questa tabella (per un periodo di tempo variabile) e puo' essere riutilizzato dall'host per inviare altri pacchetti IP senza dover fare nuovamente una richiesta ARP.
Per incrementare ulteriormente le prestazioni sono stati introdotti altri stratagemmi di caching. Se infatti un host riceve una ARP reply (anche non sollecitata) esso la memorizzera' in cache.
Alcuni kernel (come quelli di linux) adottano anche un meccanismo di caching all'arrivo di una ARP request.

```
/usr/src/linux/net/ipv4/arp.c

/*
 *  Process entry.  The idea here is we want to send a reply if it is a
 *  request for us or if it is a request for someone else that we hold
 *  a proxy for.  We want to add an entry to our cache if it is a reply
 *  to us or if it is a request for our address.
 *  (The assumption for this last is that if someone is requesting our
 *  address, they are probably intending to talk to us, so it saves time
 *  if we cache their address.  Their address is also probably not in
```

```
 *  our cache, since ours is not in their cache.)
 *
 *  Putting this another way, we only care about replies if they are to
 *  us, in which case we add them to the cache.  For requests, we care
 *  about those for us and those for our proxies.  We reply to both,
 *  and in the case of requests for us we add the requester to the arp
 *  cache.
 */
```

Vediamo ora in dettagli cosa succede nei seguenti tre casi:

- Spedizione di un pacchetto IP

Il kernel controlla di avere in cache l'indirizzo MAC corrispondente all'IP destinazione. Se si, costruisce il fram ethernet e lo spedisce.
Se no, manda una ARP request e una volta ricevuta la risposta prepara e spedisce il frame.

- Arrivo di una ARP request

Il kernel risponde con una ARP reply.
Alcuni kernel inseriscono in MAC address sorgente della richieta nella cache nell'ipotesi che anche lui dovra' a breve comunicare con l'altro.

- Arrivo di una ARP reply

Se l'arp reply era stata sollecitata dal kernel viene inserita immediatamente in cache.
Alcuni kernel (windows, linux) inseriscono in cache anche arp reply non sollecite. Altri (solaris) non aggiornano l'arp cache solo se l'entry relativa a quell'ip era gia' presente in cache.

ATTACCO

Sfruttando il comportamento di un host in ricezione di una ARP reply e' possibile modificare le ARP cache degli host vittime.

Si mandano delle false risposte ARP ai due host che vogliamo attaccare.
Nelle risposte diremo al primo host che il MAC address del secondo host e' quello della nostra interfaccia di rete e lo stesso faremo col secondo host.
Da quel momento in poi tutti i pacchetti che dovrebbero viaggiare tra le vittime saranno in realta' spediti a noi.

ESEMPIO DI ATTACCO :

```
HOST 1:  mac: 01:01:01:01:01:01      ATTACKER HOST: mac: 03:03:03:03:03:03
          ip: 192.168.0.1                            ip: 192.168.0.3

HOST 2:  mac: 02:02:02:02:02:02
          ip: 192.168.0.2
```

l'ARP cache dell'HOST 1 sara' simile a questa:

```
[user@HOST1] $ arp -a
host2.victim.org (192.168.0.2) at 02:02:02:02:02:02 [ether] on eth0
```

mentre quella dell'HOST 2 :

```
[user@HOST2] $ arp -a
host1.victim.org (192.168.0.1) at 01:01:01:01:01:01 [ether] on eth0
```

A questo punto l'attacker (host 3) mandera' delle risposte ARP a:

```
HOST 1 dicendo che 192.168.0.2 ha come MAC 03:03:03:03:03:03
HOST 2 dicendo che 192.168.0.1 ha come MAC 03:03:03:03:03:03
```

Le arp cache si presentano ora cosi':

```
[user@HOST1] $ arp -a
host2.victim.org (192.168.0.2) at 03:03:03:03:03:03 [ether] on eth0

[user@HOST2] $ arp -a
host1.victim.org (192.168.0.1) at 03:03:03:03:03:03 [ether] on eth0
```

A questo punto i pacchetti da e verso 1 e 2 saranno mandati in un frame ethernet con indirizzo di destinazione 03:03:03:03:03:03 (che e' quello dell'attaccante)

Per portare a termine l'attacco e' necessario che i pacchetti siano inoltrati correttamente verso l'effettiva destinazione. Quindi l'attaccante non deve far altro che inoltrare :

```
a HOST 1 (01:01:01:01:01:01) i pacchetti indirizzati a 192.168.0.1
```

```
a HOST 2 (02:02:02:02:02:02) i pacchetti indirizzati a 192.168.0.2
```

La cache e' periodicamente cancellata dal kernel. quindi per poter portare a termine un attacco ARP poison a lung termine, l'attaccante dovra' continuamente rinfrescare il valore di timeout delle entries mandando le relative ARP replies ad intervalli regolari minori del timeout.

ATTACCO VERSO PARTICOLARI SISTEMI OPERATIVI

LINUX

Come visto in precedenza il kernel di linux mette in cache anche gli indirizzidegli host che effettuano una richiesta ARP. Sfruttando questa feature, possiamo portare a termine l'attacco di man-in-the-middle mandando all'host delle richieste arp spofate.

ESEMPIO:
(gli host e i rispettivi indirizzi sono gli stessi di prima)

HOST 3 mandera' una richesta ARP a HOST 2 dicendo di essere HOST 1.
viceversa mandera' una richesta ARP a HOST 1 dicendo di essere HOST 2.

le risposte che le vittime genereranno, non influiscono sul regolare andamento dell'attacco, poiche' la risposta sara' indirizzata verso l'attaccante, che e' quindi il solo a ricevere le vere risposte ARP.
(molti IDS controllano solo le ARP reply e non le arp request spoofate)

SOLARIS

Questo sistema operativo non mette in cache le risposte ARP se la rispettiva entry non e' gia' presente. Quindi se una vittima non ha ancora effettuato comunicazioni verso l'altro host, non e' possibile poisonare la cache in modo tradizionale.
Per portare a termine l'attacco dobbiamo in qualche modo forzare il kernel ad inserire in cache l'entry corrispondente. Per fare cio' si manda un ICMP echo-request spoofato con l'indirizzo dell'altra vittima.
Il kernel dovendo rispondere con un ICMP echo-reply, cerchera' il MAC address della vittima attarverso una ARP request. La reply sara' messa in cache (poiche' era sollecitata).
A questo punto possiamo iniziare a mandare le ARP replies per poisonare la cache della vittima.

SNIFFING SU RETE SWITCHATA

Questo e' possibile anche su rete Switchata poiche' i pacchetti hanno come MAC address destinazione il nostro reale Mac address.
Infatti uno switch una volta create le associazioni porta-MACaddress si limita a mandare tutti i frame che hanno come destinazione il mac relativo ad una determinata porta.
Lo switch e' totalmente ingnaro di quello che sta accadento sulla rete, poiche' tutti i frame che passano hanno MAC sorgente e destinazione legali. Lo switch non si preoccupa di cosa c'e' scritto nel payload delle richieste/riposte ARP, lui si limita solo a forwardare i frame sulla porta corretta.
Il poisoning avviene quindi a livello di host e non di switch come invece avverrebbe con un MAC flooding.
Se poi i pacchetti arp sono mandati all'interno di frame con MAC sorgente reale, anche il "port security" dello switch e' inefficace.

TOOLS

- ETTERCAP
  - poisoning
  - sniffing
  - hijacking
  - filtering
  - ssh sniffing (non proxed)

- DSNIFF
  - poisoning
  - sniffing
  - ssh sniffing (proxed)

- ARPSPOOF
  - poisoning

TRACCE LASCIATE

La traccia piu' evidente lasciata e' nelle arp cache delle vittime.
L'IP dell host con cui si voleva comunicare e' associato al mac address reale dell'attaccante.
Se lo switch non e' provvisto di port security (o meglio ancora su hub), si potrebbe spoofare il mac address utilizzato per l'attacco.

CONTROMISURE

- Port security (NO)
- Passive monitoring (arpcop arpwatch, lo rileva ma non lo evita)
- Active monitoring (ettercap, lo rileva ma non lo evita)
- IDS (lo rileva ma non lo evita)
- Static ARP (SI ma dove e' possibile... dhcp no, rete grandi dimensioni no.)

==========

MAN-IN-THE-MIDDLE IN RETE LOCALE

STP MANGLING

Questo NON e' un vero e' proprio attacco di mitm, ma ci permette di ricevere delle "copie" di pacchetti, che normalmente non riceveremmo, in ambienti con uno o piu' switches collegati fra loro. Il protocollo di spanning tree (802.1d) e' un protocollo di layer 2 appositamente progettato per evitare "loop" di pacchetti dove siano presenti percorsi ridondati, per rendere "fault tolerant" una rete degli switches.

Nel seguito prenderemo come modello di riferimento Cisco Catalyst. Ogni pacchetto che ha come destinazione un MAC address che non e' presente nelle tabelle del switch, cosi' come ogni pacchetto broadcast o multicast (tranne dove esplicitamente indicato dalla configurazione dell'apparato), viene inoltrato su tutte le porte dell'apparato (sia quelle collegate alle workstations sia quelle collegate ad altrgli switches). Questo potrebbe portare a dei loop.
La soluzione consiste nel mettere nello stato di "blocking" (e quindi non far partecipare al forwarding) tutte le porte che fanno parte di percorsi ridondati.

Per risalire alla topologia della rete, e costruire un albero privo di loop, gli switches che supportano 802.1d scambiano dei pacchetti di bridge protocol data unit (BPDUs) con i loro vicini.

Lo scambio si conclude con l'elezione di un root switch (che sara' la radice dell'albero di forwarding) e con la designazione, da parte di ogni switch, della porta corrispondente al percorso piu' breve per raggiungere la radice (root port) e della porta (o delle porte) per inoltrare i pacchetti ai rami inferiori dell'albero (designated port).
Questo accade per ogni VLAN presente nella topologia. Vediamo piu' nel dettaglio il funzionamento del protocollo.

Ogni pacchetto contiene l'identifcativo di priorita' del switch che lo sta trasmettendo, l'identificativo della porta di trasmissione, l'identificativo di priorita' del switch che il mittente crede essere la radice in quel momento, il costo del percorso per raggiungere la radice dalla sua posizione, piu' altre informazioni relative ai timeout e all'eta dei pacchetti (che esulano dalla nostra trattazione). L'identificativo di priorita' e' composto da due byte che rappresentano la priorita' vera e propria (32768 e' il valore di default) piu' il MAC address del switch (ricordando che il MAC address di ogni porta del switch e' formato dal MAC address del switch piu' l'identificativo di ogni porta).

Minore e' questo numero, maggiore e' la priorita'. In una configurazione di default, lo switch con priorita' maggiore (root switch) sara' quindi quello col MAC address piu' basso. I due byte aggiuntivi permettono di modificare manualmente questo comportamento, forzando l'elezione di una radice "piu' centrale" nella topologia di rete.

Inizialmente tutte le porte del switch passano attraverso uno stato di "listening" e poi di "learning". In questo stato lo switch comincia a trasmettere le BPDUs.

All'inizio ogni switch crede di essere root. Quando uno switch riceve un pacchetto che indica una radice con priorita' maggiore (un numero piu' basso) della sua, prende quella come root e identifica la porta da cui l'ha ricevuto come "root port"; quando riceve un pacchetto con un root ID uguale a quello che lui conosce al momento, viene scelto il percorso col costo minore; nel caso anche il costo sia uguale, viene guardato l'identificativo della porta.

Quindi lo switch inizia a trasmettere pacchetti contenenti il root ID "aggiornato". Alla fine della fase di learning, ogni switch sa quali porte sono coinvolte nello spanning tree (le porte da cui ha ricevuto BPDU), e ha identificato la sua root port (la porta relativa al percorso piu' breve per raggiungere la radice).
A questo punto, per ogni segmento di rete, lo switch piu' vicino alla radice (e con ID piu' basso) viene eletto "designated switch". Questo switch identifica la porta che lo collega al segmento, di cui e' stato eletto "designato", come designated port (tutte le porte del root switch coinvolte nello spanning tree diventano "designated"). A questo punto, ogni switch mette nello stato di blocking tutte le porte coinvolte nello spanning tree che non siano "root" o "designated".

Le altre vengono messe in stato "forwarding", raggiungendo una configurazione stabile e priva di loop. Le porte in stato forwarding continuano a ricevere e mandare BPDU (le porte blocking ricevono soltanto) per poter reagire all'inserimento di un nuovo root nella rete o alla caduta di un link.
Questo avviene quando uno switch non riceve piu' BPDU sulle porte root o designated (probabilmente uno switch e' andato "giu'") per un certo periodo di tempo, e lo switch forza un cambiamento di topologia, rimettendo in gioco le porte che erano state tenute in "blocking".

ATTACCO

Possiamo semplicemente forgiare e inviare pacchetti che identifichino noi come root, mettendo a zero i campi di priorita', ed eventualmente "abbassando" manulamente il nostro MAC address (nel caso sia gia' presente uno switch a priorita' zero).
In breve tempo tutti i siwtches "saluteranno" l'ingresso del nuovo root switch e ricostruiranno l'albero usando noi come radice. A questo punto tutto il traffico "unmanaged" verra' inoltrato, di switch in switch, verso di noi. Questo ci permette di ricevere traffico unicast non destinato a noi, nel caso in cui nessuno switch nel percorso fra l'apparato da cui il pacchetto parte e noi (la radice) abbia il MAC address destinazione del pacchetto nelle sue tabelle (cosa che sarebbe molto improbabile se fossimo una semplice foglia). In questo caso potrebbe essere utile tentare di "riempire" le tabelle del switch, qualora tali tabelle siano globali e non "per-port", visto che saremmo in grado di riempire solo le tabelle relative alle porte che ricevono i nostri pacchetti, e non quelle relative alla reale destinazione del pacchetto che vogliamo intercettare. Quando smetteremo di portare l'attaco, i switch reagiranno automaticamente alla nostra "caduta" e ridisegneranno l'albero senza di noi.

N.B.
Durante l'attacco, i pacchetti raggiungono comunque anche il legittimo destinatario. Per ottenere il m-i-t-m (se riusciamo a monitorare entrambi i versi di una connessione) potremmo tentare di "desincronizzare" le sessioni TCP (vedere TRAFFIC DESYNC)

TOOLS

- Ettercap lamia plugin

TRACCE LASCIATE

- Il nostro MAC address nello "stato" del switch (che puo' essere comunque spoofato).

CONTROMISURE

- Disabilitare STP sulle VLAN prive di loop (sconsgiliato da Cisco)

- Root Guard (se l'apparato lo supporta) impedisce che determinate porte diventino "root port".

- Settare le porte connesse a workstations come "portfast", e attivare BPDU Guard su queste porte (se l'apparato lo supporta). Il portfast da solo NON e' sufficiente.

- Alcuni switches permettono di attivare selettivamente STP sulle singole porte.

==========

MAN-IN-THE-MIDDLE IN RETE LOCALE

TRAFFIC DESYNC

Se siamo in grado di monitorare tutto il traffico di un segmento di rete (se siamo collegati ad un semplice HUB o a una porta in "monitor" di uno switch o con STP mangling) possiamo inserire, all'interno di una connessione TCP, dei paccchetti (forgiandoli ad hoc) in maniera tale da far incrementare il numero di sequenza atteso. In questo modo l'endpoint della connessione verso cui abbiamo "iniettato" questi pacchetti smettera' di accettare il traffico proveniente dalla sorgente legittima della connessione, poiche' lo riterra' obsoleto, e accettera' invece il traffico che noi avremo intercettato, eventualmente modificato, e rispedito (dopo aver aggiornato i numeri di sequenza). Questo comunque NON e' a tutti gli effetti un attacco di tipo m-i-t-m per cui le considerazioni fatte, precedentemente ed in seguito, vanno prese con le dovute restrizioni.

==========

MAN-IN-THE-MIDDLE IN RETE LOCALE

DNS SPOOFING

Il servizio DNS (Domain Name System) si occupa della traduzione dei nomi simbolici degli host (www.kernel.org) in indirizzi IP (204.152.189.113). In realta' il protocollo utizzato e' molto complesso ma ci limiteremo ad una trattazione minimale per gli scopi inerenti l'attacco.

Quando un host vuole contattare un server di cui conosce solamente il nome simbolico manda una richiesta di tipo A al suo DNS che rispondera' con l'indirizzo IP corrispondente.

Un pacchetto DNS si presenta cosi':

```
/*
 *                                 1  1  1  1  1  1
 *       0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |                      ID                       |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |                    QDCOUNT                    |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |                    ANCOUNT                    |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |                    NSCOUNT                    |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 *     |                    ARCOUNT                    |
 *     +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 */
```

L'unico campo che ci interessa per il momento e' il Campo ID (2 byte) per identificare le risposte.

(I vari tipi di richieste, la trattazione delle query ricorsive e non, le risposte autoritative e non, la generazione degli ID, verranno trattate in seguito nella sezione di DNS POISONING).

ATTACCO

Quello che ci interessa ottenere e' sostituire l'indirizzo IP della risposta con quello della macchina attaccante. Per fare cio' e' necessario creare delle risposte con il giusto valore nel campo ID, altrimenti il client non accettera' la risposta. Questo meccanismo di minima sicurezza garantisce che non si possano creare risposte in modo blind, ma se potessimo sniffare le richieste, il gioco e' fin troppo semplice. Una volta sniffato l'ID della richiesta si crea una riposta con lo stesso ID e la si manda al client prima che il vero DNS risponda.

Per completare l'opera e' buona norma anche cambiare le richieste in reverse (PTR) da IP a nome simbolico.

Una volta spoofata la riposta, il client mandera' tutti i pacchetti destinati verso quel nome simbolico all'ip dell'attaccante. L'attaccante si deve preoccupare di accettare la connessione e di crearne un altra verso il server reale. L'attaccante puo' fare da proxy tra il client e il server, oppure puo' non contattare mai il server reale e mostrare servizi fake al client.

TOOLS

- Ettercap (phantom plugin)

- Dsniff (dnsspoof)

- Zodiac

TRACCE LASCIATE

Usando un comando di risoluzione indirizzi, si notera' che al nome simbolico corrisponde un IP non reale (e' quello dell'attaccante).

CONTROMISURE

- Risposte multiple (non implementata o IDS?)

- Usare File lmhosts o hosts per la risoluzione statica di indirizzi

- DNSSec permette la firma digitale dei dati da parte del DNS. Ovviamente
  anche il resolver deve supportare questa estensione.

==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

ARP POISONING

Quando un host deve mandare pacchetti verso una destinazione che non e' all'interno della sua LAN, guarda nelle sue tabelle di routing alla ricerca del gateway per tale destinazione. Una volta conosciuto l'IP address del gateway, l'host fara' un ARP request per conoscere il suo MAC address. A questo punto i pacchetti potranno essere mandati "sul cavo" con l'effettivo IP address del destinatario e il MAC address del gateway come destinazione. Sara' poi il gateway a prendersi carico di inoltrare i pacchetti verso la destinazione scelta.

ATTACCO

Esattamente come nel caso precedente, facendo poisoning dell'indirizzo del default gateway su un host, saremo in grado di ricevere tutto il traffico che tale host fara' verso "internet". Valgono quindi tutte le considerazioni fatte precedentemente a riguardo.

N.B. Nel caso venga utilizzato il Proxy ARP (il gateway risponde alle ARP request per gli indirizzi all'esterno della LAN) lo scenario si riporta esattamente a quello visto in precedenza (anche gli host remoti vengono considerati come locali).

CONTROMISURE

- Inserendo ARP statiche per il gateway (o i gateway) impdiremo all'attaccante di intercettare la parte della connessione che va verso l'esterno.

==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

TRAFFIC DESYNC

Valgono le stesse considerazioni fatte nel caso della rete locale.

==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

DNS SPOOFING

Valgono le stesse considerazioni fatte nel caso della rete locale.

==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

DHCP SPOOFING

Il servizio DHCP e' utilizzato per l'assegnazione in maniera dinamica di una serie di parametri per la connettivita' di rete (indirizzo IP, dns, default route) ad un host che ne fa richiesta. Il protocollo usa UDP come layer di trasporto e non supporta alcun tipo di autenticazione.

Il protocollo dhcp e' in realta' abbastanza complesso ma ci limiteremo a una trattazione minimale inerente alle informazioni necessarie a portare a termine l'attacco.

ATTACCO

Intercattando una richiesta dhcp e' possibile rispondere prima del vero server assegnando ad esempio un default gateway fasullo o un falso dns.

- Default gateway: assegnando l'indirizzo IP dell'attaccante come default gateway, tutto il traffico verso l'esterno della lan passera' da esso

- DNS: assegnando l'indirizzo IP dell'attaccante come DNS, tutte le richieste di risoluzione dei nomi verranno fatte a lui e sara' quindi in grado di portare un attacco simile a quello visto in precedenza (DNS spoofing).

Le richieste dhcp (Discover e Request) sono mandate in broadcast locale quindi, al contrario dell'attacco di dns spoofing, e' possibile sfruttarlo anche nel caso non si possa sniffare l'intero segmento di rete. Bastera' essere all'interno dello stesso broadcast domain.

TOOLS

- Non e' necessario un tool apposito ma bastera' configurare la macchina attaccante come dhcp server, installando ad esempio dhcpd.

TRACCE LASCIATE

- L'IP address dell'attaccante nelle configurazioni del client. Tale indirizzo puo' essere comunque "spoofato"

CONTROMISURE

- In questo caso, la migliore contromisura e' l'attenzione dell'utente che si vedra' porbabilmente arrivare piu' di una risposta dhcp e potra' controllare "manualmente" alla ricerca di strane o inaspettate configurazioni assegnate.

- Non permettere l'assegnazione dinamica di DNS e gateway.

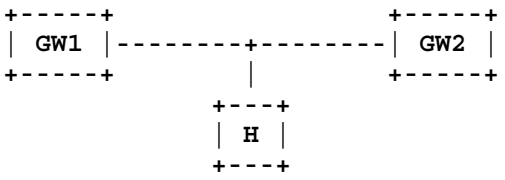
- Esistono alcune implementazioni proprietarie che supportano l'autenticazione.

==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

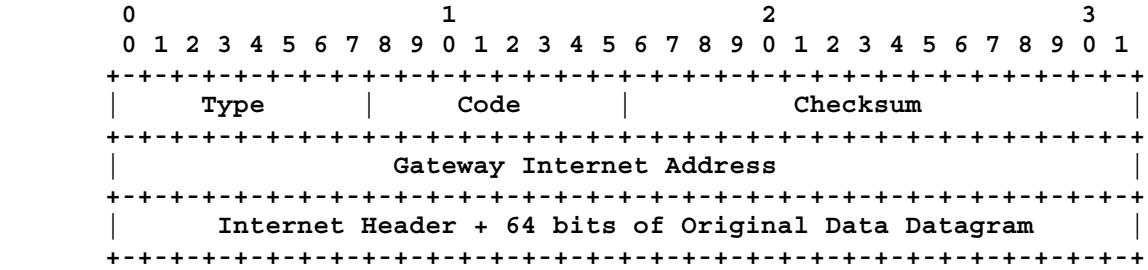
ICMP REDIRECTION

Il protocollo ICMP (Internet Control Message Protocol) e' utlizzato per una grande varieta' di scopi tra cui il famoso "ping". In questo caso esamineremo il comando REDIRECT.
Abbiamo gia' visto cos'e' e a che serve un gateway, vediamo ora uno scenario in cui ce ne sia piu' di uno.



Mettiamo che GW1 sia, per H, il gateway per una determinata serie di indirizzi. Se pero' GW1, per instradare un pacchetto ricevuto da H, e' costretto a passare da GW2, informera' l'host con un ICMP REDIRECT, dicendogli di instradare i pacchetti successivi, per la medesima destinazione, direttamente verso GW2, eliminando cosi' un "hop" dal tragitto.
Il trattamento di tale tipo di pacchetto varia comunque da OS a OS.

In generale un pacchetto di tipo REDIRECT e' fatto in questo modo:



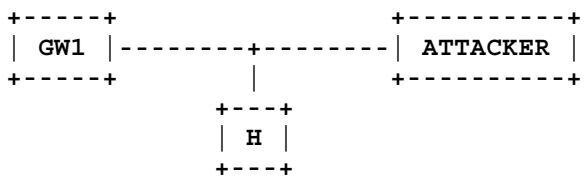

```
ICMP Fields:
 Type
    5

 Code
    0 = Redirect datagrams for the Network.
    1 = Redirect datagrams for the Host.
    2 = Redirect datagrams for the Type of Service and Network.
    3 = Redirect datagrams for the Type of Service and Host.
```

E' importante notare come nel REDIRECT sia presente una parte del pacchetto originale che GW1 e' stato costretto a inoltrare su GW2. Questo e' necessario perche' l'host sia in grado di capire a quale "flusso" fa riferimento il REDIRECT.

Si puo' vedere inoltre come sia possibile indicare una redirezione per un singolo host destinazione o per l'intera network a cui quell'host appartiene.

ATTACCO

Si puo' semplicemente forgiare un pacchetto di tipo REDIRECT per deviare il traffico che H inoltrerebbe attarverso GW1 per una data destinazione, e far agire ATTACKER come gateway "piu' comodo" per tale destinazione (host o network).



In questo caso manderemo ad H un pacchetto "spoofando" il nostro IP come se il pacchetto provenisse realmente da GW1. H comincera' a usare come gateway ATTACKER, il quale si preoccupera' di inoltrare i pacchetti attraverso GW1, avendo cura di non spedire REDIRECT "leciti" ad H che indichino GW1 come gateway preferenziale.

I pacchetti di tipo REDIRECT vengono presi in considerazione dagli host a seconda del loro sistema operativo:

- Windows9x accetta i REDIRECT di default e aggiunge una entry (di tipo host) nelle sue tabelle di routing.
- Linux accetta di default i REDIRECT in alcune distribuzioni (vedere in proposito /proc/sys/net/ipv4/<nome interfaccia>/accept_redirects)L e rotte redirette non vengono comunque mostrate nelle tabelle di routing.

etc.
Le rotte aggiunte sono comunque temporanee.

Per portare a termine questo tipo di attacco e', nella maggior parte dei casi, necessario poter sniffare il segmento di rete. Alcuni test effettuati su Linux hanno comunque riportato che tale sistema operativo sembra prendere in considerazione soltanto i campi source e destination IP del datagram originale, non facendo alcun tipo di check sulle porte TCP o UDP, sull'identificativo dei pacchetti IP etc.(contrariamente a quanto consigliato nell'RFC). In questo caso saremo quindi in grado di mandare dei REDIRECT "alla cieca". Linux permette comunque di attivare l'opzione "secure redirect", anche se questa non sembra una contromisura efficace.

TOOLS

- IRPAS icmp_redirect di Phenoelit
- icmp_redir di Yuri Volobuev

TRACCE LASCIATE

- Dove mostrato, il nostro indirizzo IP nelle rotte dell'host (anche se e' possibile spoofarlo e far comunque arrivare a noi i pacchetti rispondendo alle ARP request per l'indirizzo spoofato).

CONTROMISURE

- Si potrebbe consigliare di disabilitare i REDIRECT, ma cosi' facendo ci potebbero essere dei cali di prestazioni nella rete.

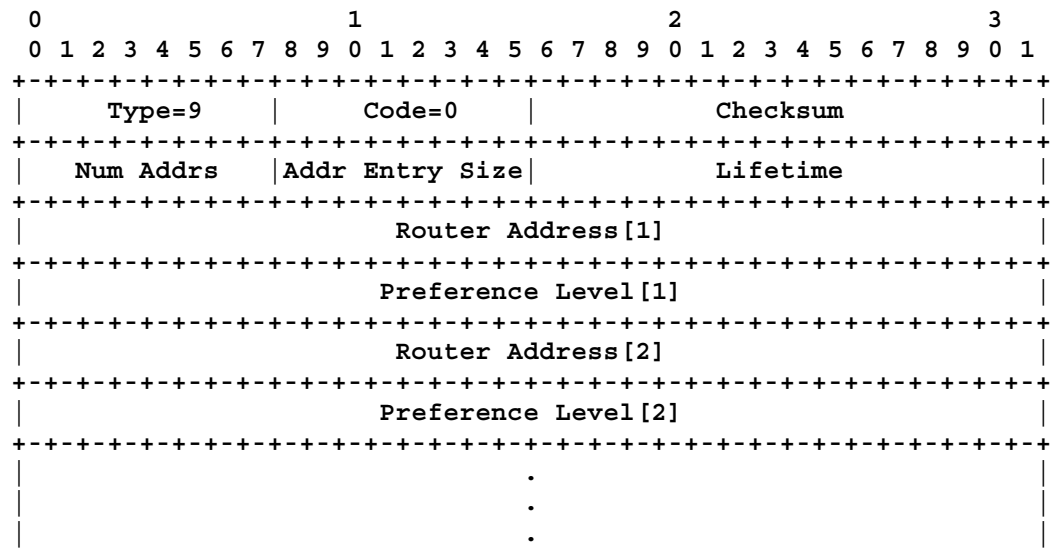
==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

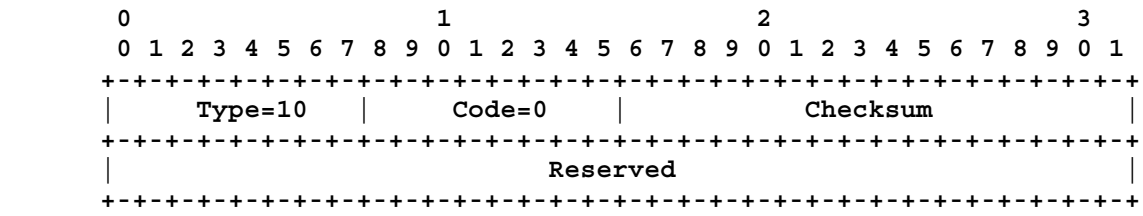
IRDP SPOOFING

IRDP (ICMP Router Discovery Protocol) e' un protocollo, sempre basato su ICMP, utilizzato per l'assegnazione automatica agli host di un gateway(router). Ci sono due tipi di messaggi previsti dal protocollo chiamati "Router Advertisements" e "Router Solicitations". Periodicamente, ogni router che supporta questo tipo di protocollo manda in multicast degli advertisment per annunciare il suo indirizzo IP (su ogni sua interfaccia di multicast). Gli host possono semplicemente aspettare un "advertisment" o mandare una "solicitation", per forzare la spedizione di un "advertisement" da parte del router. Questo protocollo non identifica quale sia il miglior gateway per una data destinazione, ma questa scelta e' lasciata all'host. Un host che accetta il primo "advertisment" e che prende il suo mittente come default gateway, potrebbe in seguito ricevere un ICMP redirect che gli indichi una rotta migliore. Ogni advertisment contiene comunque un "livello di preferenza". Nel caso riceva piu' advertisment da diverse sorgenti, l'host dovrebbe scegliere quello con il livello piu' elevato; il valore minimo per tale campo e' 0x80000000 (il valore e' signed) e indica che quel router non dovrebbe essere usato come default gateway. Il pacchetto contiene inoltre un campo "lifetime" che sta ad indicare il tempo per cui l'host debba conservare quella rotta.

Ecco quindi come appare un pacchetto di Advertisment:



E come appare un pacchetto di Solicitation:



ATTACCO

Possiamo semplicemente forgiare degli "advertisment" annunciandoci come router e settando i campi "livello di preferenza" e "lifetime" al massimo valore consentito (per il lifetime e' 18h:12min:15sec).

In aggiunta, si puo' rendere l'attacco piu' efficace forgiando degli ICMP Host Unreachable impersonando l'attuale router di default.

Questi advertisement verranno presi in considerazione dagli hosts a seconda del loro sistema operativo:

- Windows9x accetta IRDP
- WindowsNT usa IRDP al boot
- Windows200 ignora IRDP
- Linux ignora IRDP

TOOLS

-IRPAS suite di Phenoelit

TRACCE LASCIATE

-Il nostro indirizzo IP nelle rotte dell'host (anche se e' possibile spoofarlo e far comunque arrivare noi i pacchetti rispondendo alle ARP request per l'indirizzo spoofato).

CONTROMISURE

-Disabilitare IRDP sugli hosts se il sitema operativo lo permette.

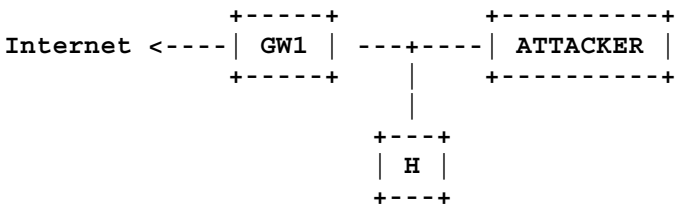
==========

MAN-IN-THE-MIDDLE DA LOCALE A REMOTO

ROUTE MANGLING

N.B. Prima di leggere questo paragrafo e' consigliabile consultare: APPENDICE A - PROTOCOLLI DI ROUTING DINAMICO E LORO DEBOLEZZE

Prendiamo il solito scenario:



e supponiamo che GW1 faccia girare un protocollo di routing dinamico (in questo caso e' molto probabile che si tratti di uno degli interior gateway protocols).

Visto che GW1 e' il ruoter di una "foglia" di internet, le sue tabelle di routing saranno molto semplici e settate in maniera statica nella configurazione. Una di queste entries rappresentera' la "candidate default", la rotta che verra' intrapresa se nessun'altra viene "matchata". Questa rotta avra' per forza di cose una netmask piccola o nulla. Se un router ha piu' rotte possibili per una stessa destinazione, scegliera' quella col "peso" maggiore. Il peso viene determinato in base a vari fattori come le metriche proposte dai vari protocolli di routing, e il tipo di protocollo stesso che ha proposto la rotta. Piu' il protocollo e' "fidato", maggiore sara' il peso delle rotte che propone. Le rotte statiche hanno generalmente un "grosso" peso. La prima cosa che pero' viene guardata nella scelta di una rotta e' (in generale) la netmask. Questo perche' piu' la netmask e' grande e piu' vuol dire che quella rotta e' specifica per l'host che vogliamo raggiungere e non per una piu' generica subnet. Se non fosse cosi', ad esempio, tutti i pacchetti "matcharebbero" la default route.

ATTACCO

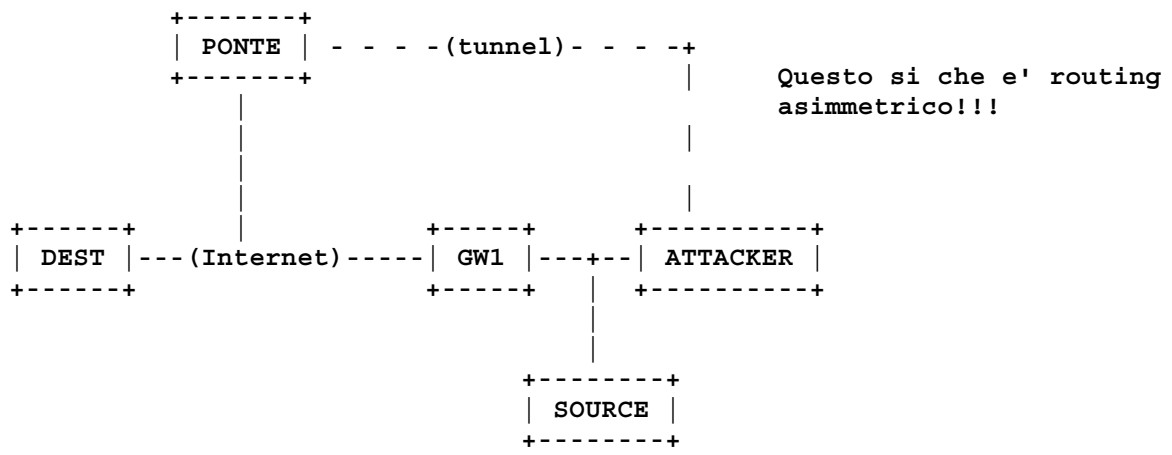
Possiamo forgiare dei pacchetti per GW1 annunciandoci come router con un ottima metrica per un determinato host (o classe o subnet). Dobbiamo solo stare attenti a non settare delle netmask troppo piccole (nei protocolli che le supportano) nelle rotte che proponiamo, per riuscire a "battere" le rotte statiche con il subnetting.

Quando H mandera' dei pacchetti per l'host per cui ci siamo annunciati come "strada" preferenziale, questi arriveranno a GW1. GW1 sara' convinto che il modo piu' veloce per raggiungere quell'host e' attraverso ATTACKER, e mandera' a noi questi pacchetti. Inoltre, vedendo che siamo sulla stessa interfaccia, GW1 mandera' ad H degli ICMP redirect dicendogli di usare noi (ATTACKER) come gateway per i pacchetti successivi. H, con molta probabilita', non accettera' questi redirect (se li accettasse ci converrebbe fare direttamente quel tipo di attacco che e', nella maggior parte dei casi, piu' semplice).

Il problema a questo punto e' trovare un modo per far arrivare i pacchetti al legittimo destinatario. A GW1, infatti, abbiamo clamorosamente "mentito" dichiarando di poter raggiungere facilmente tale

destinazione. Se provassimo a inoltrare i pacchetti verso GW1 (l'unico gateway che abbiamo per uscire dalla nostra LAN) questo, con molta probabilita', ce li rispedirebbe indietro.
Per ovviare a questo problema abbiamo due soluzioni possibili:

- Se possediamo una qualsiasi altra macchina nel mondo con un IP address pubblico o comunque raggiungibile da internet, potremo instaurare con lei un tunnel unidirezionale (con GRE ad esempio) in cui manderemo i pacchetti ricevuti, delegando al nostro "ponte" il compito di routarli.



Legenda:

`---------` **connessioni reali**

`- - - - -` **connessioni logiche**

- Se il router supporta il Multipath Routing o una qualche forma di QoS possiamo modificare il ToS del pacchetto ricevuto (generalmente settato a 0) e rispedirlo, sperando che non ci ritorni indietro. Questa e' una soluzione MOLTO aleatoria.

Non prendiamo neanche in considerazione l'ipotesi di usare il source routing che ormai e' da anni "bloccato" su praticamente tutti i router su internet.

TOOLS

- IRPAS
- Nemesis

TRACCE LASCIATE

- Il nostro indirizzo IP nelle tabelle del router che sono comunque temporanee (possiamo spoofarlo come nei casi precedenti).

CONTROMISURE

- Disabilitare i protocolli di routing dinamico che sono inutili in uno scenario di questo tipo o, quantomeno, mettere delle ACL esplicite sull'interfaccia interna del router che blocchino gli "update" indesiderati.

- Abilitare l'autenticazione MD5 nei protocolli che la supportano

==========

MAN-IN-THE-MIDDLE REMOTO

DNS POISONING

Abbiamo gia' visto cosa succede quando un host deve "risolvere" un nome simbolico in un indirizzo IP. Vediamo ora come si comporta il DNS quando riceve una richiesta del tipo visto in precedenza.

Quando il DNS riceve una richiesta ci possono essere 3 possibilita':

- Il DNS e' "autoritativo" per il dominio (o zona) a cui il nome richiesto appartiene.
  Es: se un DNS e' autoritativo per il dominio "kernel.org" ha registrati tutti gli indirizzi IP corrispondenti a i nomi di tipo *.kernel.org.

  In questo caso il DNS fornisce direttamente la risposta.

- Il DNS non e' "autoritativo" per il dominio e non conosce l'indirizzo IP della macchina richiesta.

  In questo caso si aprono altre due possibilita:

  - La query e' di tipo ricorsivo:

Questo genere di query viene generalmente fatto dagli host al proprio DNS. Il DNS si prende carico di risolvere "completamente" il nome. Se la richiesta e' per "www.kernel.org" il DNS chiedera' all'autoritativo per il dominio *.org di risolvere il nome. Tale DNS fara' la stessa
richiesta a un DNS autoritativo per *.kernel.org (di cui conosce l'indirizzo) e propaghera' indietro la risposta. L'host si aspettera' quindi di ricevere una risposta dal DNS a cui ha fatto la prima interrogazione.

  - La query e' di tipo iterativo:
    Il DNS restituisce all'host l'indirizzo del DNS (o dei DNS) autoritativo per *.org in una risposta di tipo "referral". L'host stesso a questo punto chiedera' a quest'ultimo l'indirizzo del DNS autoritativo per *.kernel.org a cui chiedera' infine di risolvere il nome www.kernel.org.
    L'host in questo caso si aspettera' di ricevere una risposta dal DNS autoritativo per il dominio kernel.org.

E' plausibile che le richieste seguano un tragitto misto. L'host fa una richiesta di tipo ricorsivo al suo DNS (da cui si aspettera' quindi diricevere la risposta). Il DNS, dal canto suo, completa la richiesta in maniera non ricorsiva (si aspetta quindi una risposta dal DNS autoritativo per il dominio richiesto).

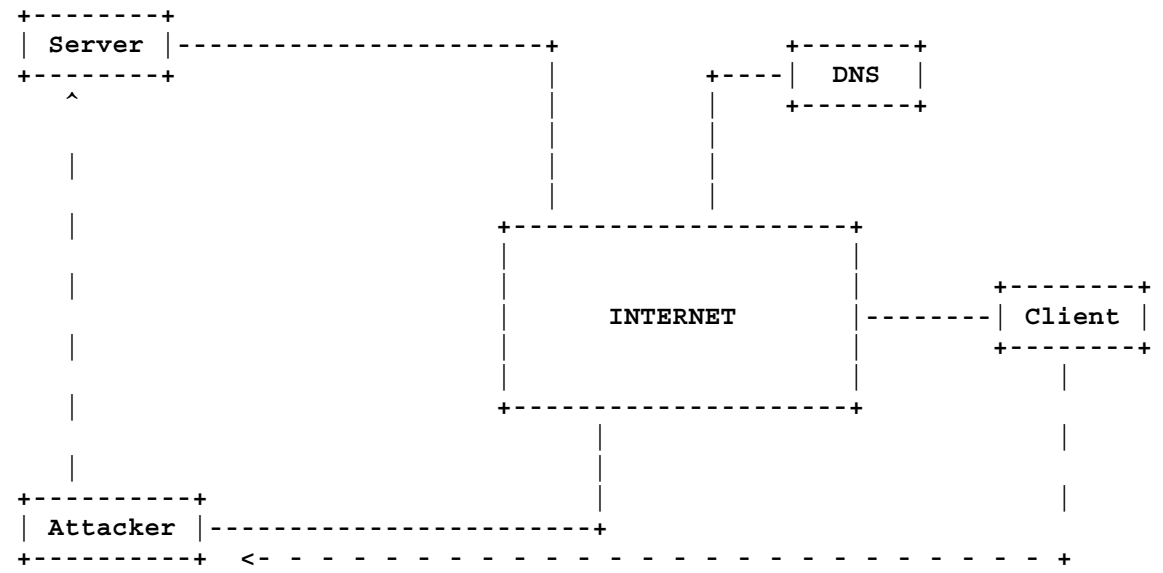
- Il DNS non e' "autoritativo" per il dominio richiesto ma ha cachato la coppia <nome simbolico, indirizzo IP> in seguito ad una precedente richiesta. Per migliorare le prestazioni un DNS

conserva i risultati delle richieste in una cache interna per un tempo specificato all'interno della risposta stessa

In questo caso il DNS fornisce direttamente una risposta di tipo "non autoritativo".

ATTACCO

ATTACCO 1:
L'idea di base dell'attacco consiste nel "cachare" in un DNS remoto delle coppie <nome simbolico, indirizzo IP> dove "nome simbolico" e' un server plausibilmente molto contattato dalle macchine che sfruttano quel DNS e "indirizzo IP" e' l'indirizzo della macchina attaccante. Riuscendo a cachare ad esempio www.kernel.org, quando una macchina fara' richiesta per tale nome, il DNS rispondera' con l'indirizzo della macchina attaccante. Sara' poi l'attaccante a prendersi carico di "girare" la connessione verso il vero server.
L'host sara' quindi convinto di parlare direttamente con il vero server



Legenda:

--------- connessioni reali

- - - - - connessioni logiche

Per perpetrare l'attacco faremo una richiesta al DNS che vogliamo "poisonare" richiedendo l'indirizzo della macchina che vogliamo "cachare". Il DNS, in seguito alla sua richiesta, si aspettera' di ricevere una risposta dal DNS autoritativo per il dominio in questione. Noi forgeremo e invieremo tale risposta "spoofando" l'IP sorgente e inserendo nel pacchetto l'IP della macchina attaccante. Tale spoof e' possibile poiche' si tratta di pacchetti UDP (TCP viene utilizzato solo nel caso le risposte siano troppo lunghe per essere gestite in un singolo pacchetto UDP).

A questo punto e' necessario fare attenzione a due cose:

- Il DNS non deve avere gia' nella cache il nome che chiediamo di risolvere, altrimentici spedira' indietro direttamente la risposta e non fara' alcuna richiesta ai DNS autoritativi (ignorera' quindi le risposte forgiate da noi)

- Nella risposta da noi forgiata dobbiamo inserire l'ID corretto della transazione iniziata dal DNS in maniera che la nostra venga riconosciuta come risposta "valida".

Tale ID e' composto di due byte = 2^16 possibili ID diversi.

Per "guessare" tale ID possiamo tentare due tipi di approccio:

- Brute force: Invieremo una grossa quantita' di risposte con ID casuali sperando che uno di questi corrisponda a quello cercato.

- SemiBlind guessing: Alcuni DNS hanno una generazione sequanziale dei transaction ID.
  Se una richiesta viene fatta con ID 10 la successiva avra' ID 11.

Potendo quindi intercettare una di queste richieste saremo in grado di predire con una certa accuratezza l'ID delle transazioni successive. Per fare cio' dobbiamo essere in grado di "sniffare" un segmento di rete dove sia presente un DNS autoritativo per un qualsiasi dominio (evilhacker.com)
Prima di tutto richiederemo al DNS che vogliamo attaccare di risolvere ad esempio il nome nohost.evilhacker.com. Il DNS vittima dovra' fare una richiesta all'autoritativo per evilhacker.com. Noi saremo in grado di sniffare tale richiesta e leggere l'ID corrispondente. Una volta ottenuto tale ID faremo al DNS la richiesta per il nome che vogliamo cachare e forgeremo le nostre risposte con gli ID successivi a quello sniffato. Se il DNS non ha un carico eccessivo saremo in grado di "indovinare" l'ID anche con un numero relativamente basso di risposte.

N.B. Un DNS potrebbe non accettare la risoluzione di nomi che non appartangano alle sue zone.

ATTACCO 2:
Un'altro possibile approccio al problema e' il dynamic update. Questo non e' un vero e proprio attacco, ma piu' una "feature" di alcuni DNS. Inviando delle semplici richieste e' possibile cancellare e aggiungere entries (non solo di tipo 'A' ma anche MX etc.) alle zone per cui il DNS e' autoritativo.
La pericolosita' di questo "attacco" e' elevata poiche' queste entries rimarranno permanentemente modificate nel db del DNS ed, essendo questo autoritativo, verranno propagate anche alle cache degli altri DNS che ne faranno richiesta. Restringere la possibilita' di fare dynamic update ad un range di IP e' poco efficace come soluzione dato che il protocollo utilizzato e' UDP e potremo quindi facilmente "spoofare" le nostre richieste.

ATTACCO 3:
Una variante meno "tecnica" di questo attacco e' possibile abusando del sistema di assegnazione dei domini (ad esempio richiedendo un spostamento di dominio a Network Solutions Inc.)

TOOLS

- ADMIdPack

- Zodiac

- Net::DNS::Update permette di effettuare il dynamic update con un semplice script perl

---

TRACCE LASCIATE

Come nel caso del DNS spoofing, gli host manderanno tutti i pacchetti della connessione verso l'indirizzo IP della macchina attaccante. Inoltre il DNS conservera' traccia di tale IP per tutto il tempo di latenza dell'entry nella cache.

CONTROMISURE

- Aggiornare i DNS con generazione sequenziale dei transaction ID(Es: Bind v9)

- DNSSec (anch'esso implementato in Bind v9) permette la firma digitale dei dati. Ovviamente anche il resolver deve supportare questa estensione.

- Evitare di rendere pubblico il DNS che si utilizza "in casa" per la risoluzione di nomi non appartenenti alla propria zona.

- Eliminare il dynamic update o rendere pubblico un semplice forwarder che inoltra le richieste ad un altro name server con questa feature, non raggiungibile dall'esterno.

==========

MAN-IN-THE-MIDDLE REMOTO

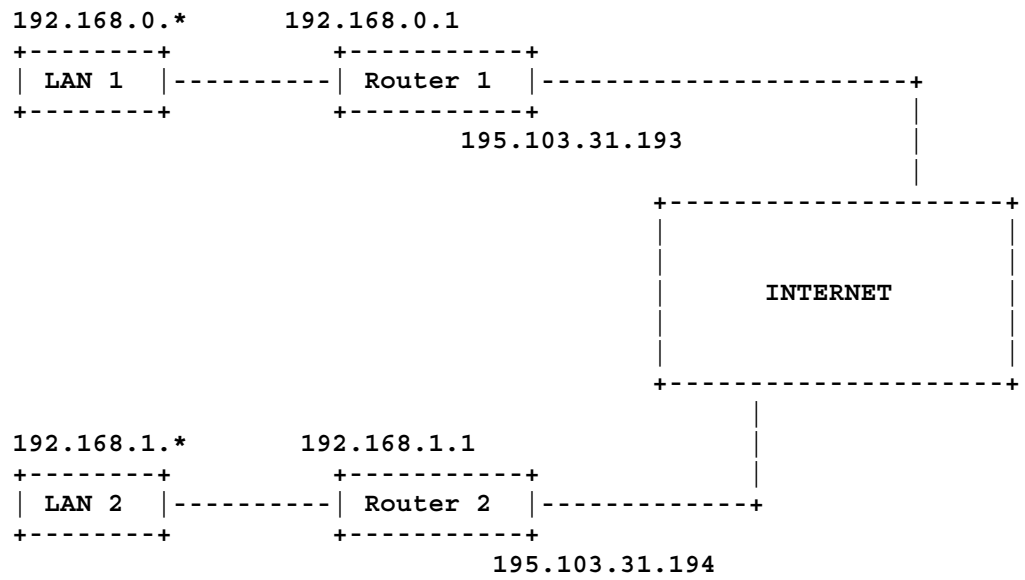
TRAFFIC TUNNELING

L'obbiettivo di questo tipo di attacco e' dirottare, verso l'attaccante, il traffico in entrata di un router remoto, sfruttando la protocol encapsulation e il traffic tunneling. Nel seguito della spiegazione faremo riferimento ai router CiscoIOS e al metodo di incapsulamento IP su GRE (la teoria e' comunque applicabile anche in altri casi).

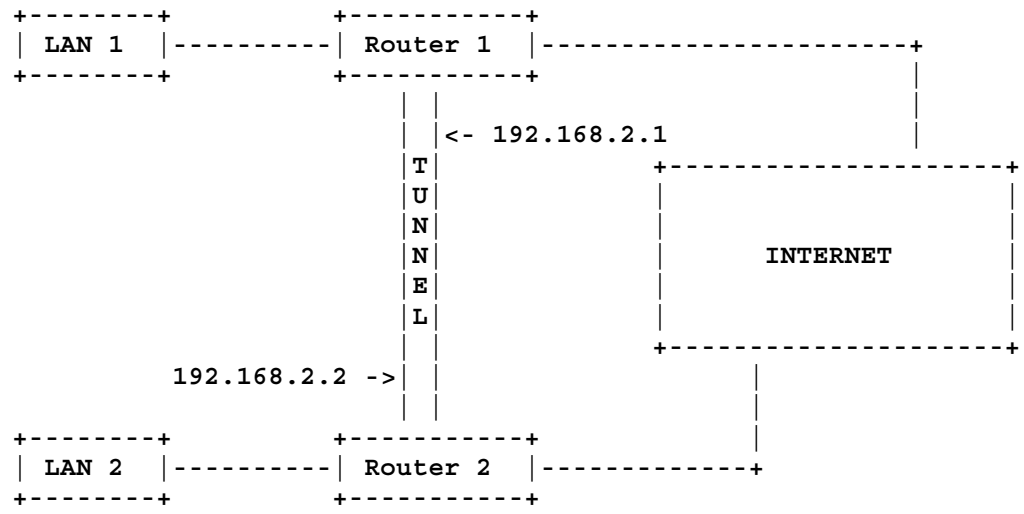
Un tunnel e' instaurato tra due "end point" detti tunnel broker. Su ognuno dei due endpoint, il tunnel e' associato ad un interfaccia virtuale con un suo indirizzo (ad esempio Tunnel0), che deve essere legata ad un interfaccia reale (ad esempio Serial0).

Ogni volta che il router deve inoltrare un pacchetto IP che ha come "next hop" un indirizzo che cade nella subnet dell'IP address dell'interfaccia di tunnel, il pacchetto viene incapsulato in un altro che avra' come indirizzo sorgente quello dell'interfaccia reale a cui il tunnel e' legato, e come indirizzo destinazione quello dell'interfaccia reale a cui e' legato il tunnel sull' altro end point.

Ad esempio:
Supponiamo di avere due LAN, una con indirizzamento 192.168.0.0/24, l'altra con indirizzamento 192.168.1.0/24. I default gateway delle due reti siano rispettivamente le interfacce ethernet di due router 192.168.0.1 e 192.168.1.1 Supponiamo inoltre che gli indirizzi delle interfacce seriali dei router verso internet siano rispettivamente 195.103.31.193 e 195.103.31.194 Entrambi i router inoltrano tutto cio' che ricevono dall'interfaccia ethernet sulla loro interfaccia seriale. Supponiamo a questo punto di voler far parlare fra di loro macchine della prima lan con macchine della seconda, senza operare nessun tipo di NATing.
Cio' risulterebbe impossibile poiche' indirizzi del tipo 192.168.*.* non vengono inoltrati su internet.

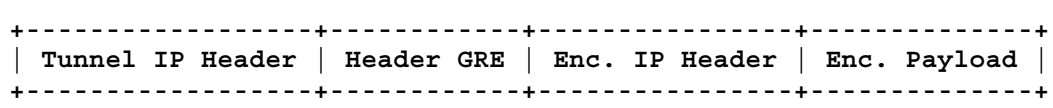


Per fare cio' possiamo creare due interfacce di tunneling sui due tunnel broker (i due router). Diamo alle due interfacce rispettivamente gli indirizzi 192.168.2.1 e 192.168.2.2. A questo punto definiamo gli end point del tunnel. Sul primo router definiamo come sorgente del tunnel l'interfaccia seriale 195.103.31.193 (diciamo cioe' al router su quale interfaccia fisica i pacchetti vanno inoltrati) e come destinazione l'indirizzo ip dell'interfaccia reale a cui e' associato l'altro tunnel endpoint (in questo caso l'interfaccia seriale dell'altro router 195.103.31.194). Sul secondo router facciamo esattamente l'inverso. Definiamo a questo punto, sulla seriale del primo router, una route map che definisce 192.168.2.2 come "next hop" per tutti i pacchetti che hanno come destinazione indirizzi del tipo 192.168.1.*. Stessa operazione anche per l'altro router.



A questo punto, quando il primo router ricevera' un pacchetto per 192.168.1.10 tentera' di inoltrarlo sull'interfaccia Tunnel0. Questa pero' e' un'interfaccia virtuale che non fara' altro che incapsulare questo pacchetto in un altro, avente come sorgente 195.103.31.193 e come destinazione 195.103.31.194. Sorgente e destinazione del pacchetto "payload" sono lasciati intatti. Quando il pacchetto arrivera' al secondo router, potra' essere decapsulato. Sara' poi il secondo router a preoccuparsi di inoltrare il pacchetto a destinazione (192.168.1.10) sulla sua interfaccia ethernet.

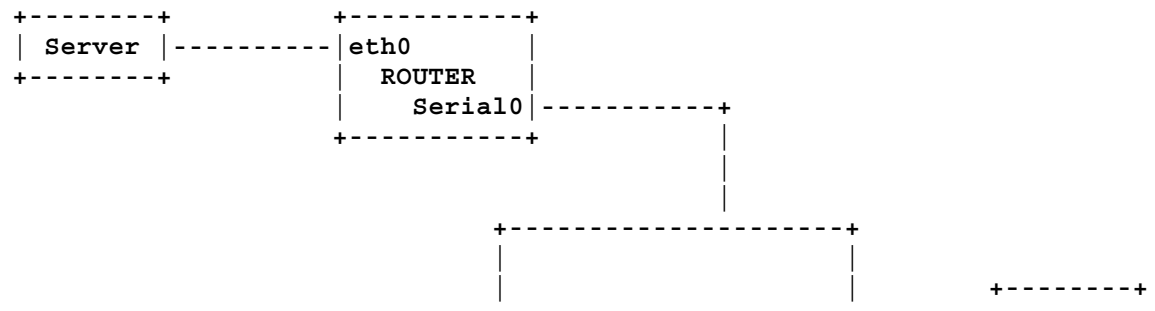
I pacchetti che usciranno dall'interfaccia seriale del router avranno quindi questa struttra :

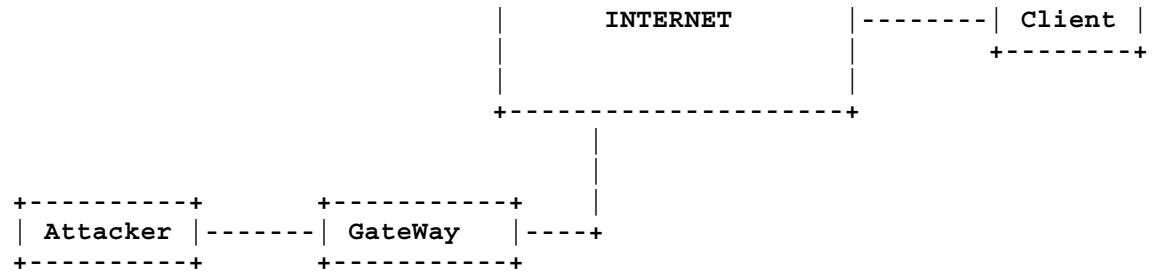


Come e' facile intuire ogni tunnel e' identificato anche dal tipo di incapsulamento utilizzato. In questo caso IP su GRE torna particolarmente utile data la natura completamente "stateless" del protocollo.

ATTACCO

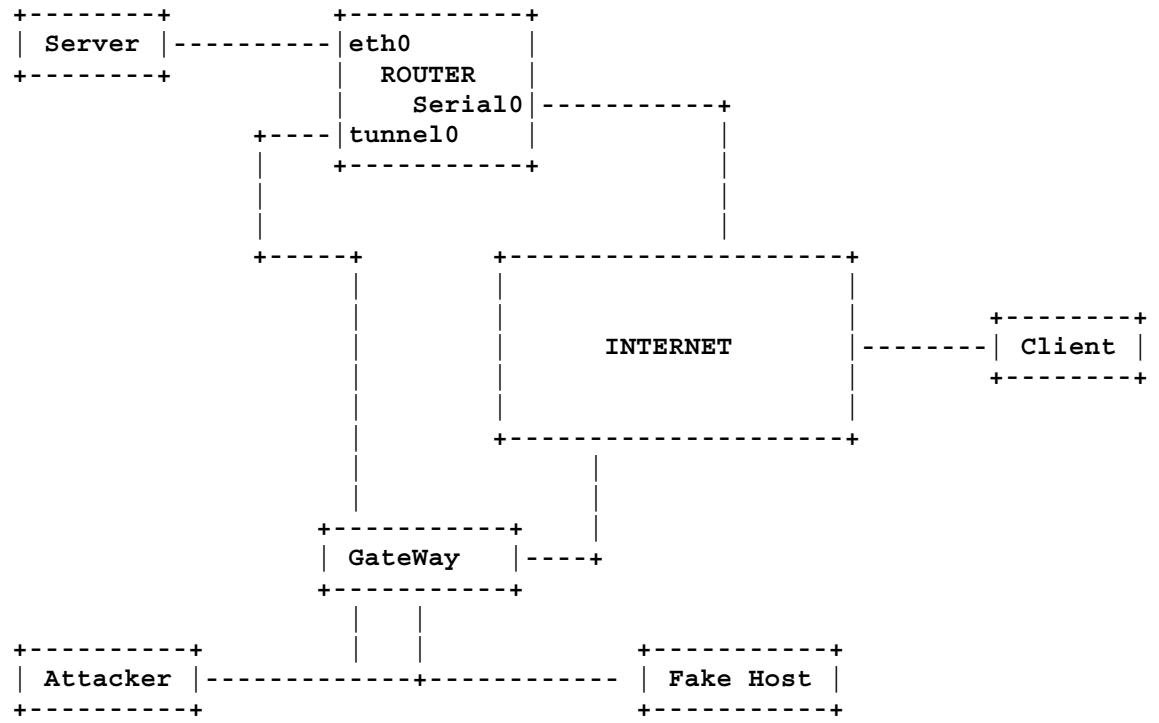
Prendiamo uno scenario di questo tipo:





Quello che a noi interessa ricevere, dalla nostra posizione (attaccante), e' il traffico dal client (o dai clients) al server e viceversa. Per fare cio' dovremo innanzitutto avere accesso a livello di amministrazione su un router a monte del server. (il metodo per avere accessi non autorizzati su di un router e' un argomento che esula da questa trattazione).

Possiamo ora creare un interfaccia di tunneling che abbia come sorgente Serial0 del router, come destinazione un indirizzo pubblico (o static NATed) non assegnato (FakeIP) della nostra subnet (per evitare degli spiacevoli protocol unreachable), e come indirizzo IP ad esempio 192.168.2.1. Definiamo una ACL che faccia il matching di tutto il traffico che ci interessa sniffare. Associamo questa ACL a una route map che definisce 192.168.2.2 come "next hop". Associamo la route map cosi' creata ad entrambe le interfacce reali del router (ethernet e seriale). Ora, quindi, tutto il traffico che arrivera' al router verra' inoltrato sull' interfaccia di tunneling che ha noi (FakeIP) come end point. A questo punto dovremo solo prenderci cura di rispondere alle ARP request che il nostro gateway mandera' per risolvere l'inidirzzo inesistente (FakeIP) e potremo ricevere tutto il traffico da e verso il server, e decapsularlo.



N.B.
Lo stesso effetto puo' essere ottenuto senza l'ausilio di un gateway, definendo come end point direttamente l'inidirzzo della macchina attaccante, dove inseriremo una manciata di regole di firewalling per evitare che il nostro kernel risponda ai pacchetti GRE; o ancora abilitando un interfaccia di tunneling anche sulla nostra macchina e poi inibendo il forwarding dei pacchetti decapsulati...insomma le possibilita' sono tante.

A questo punto ci dobbiamo preoccupare di re-inoltrare il traffico verso la destinazione originale dei pacchetti. Si puo' operare in due modi:

- Rimandiamo i pacchetti all'interno del tunnel e lasciamo che sia il ruoter a inoltrarli per noi. Questa scelta e' consigliabile poiche' saremo anche invisibili ai traceroute. Infatti nel tragitto fra il router e noi (e viceversa) il ttl che viene decrementato e' quello del pacchetto GRE e non quello del pacchetto originale.

- Inoltriamo direttamente i pacchetti decapsulati facendo pero' attenzione a "marchiarli" in qualche modo (cambiando ad esempio il TOS) in maniera tale che l'ACL sul router (da cui dovra' comunque ripassare il traffico che era originariamente diretto al server) riconosca che si tratta di pacchetti rispediti da noi e non li reinserisca nel tunnel (creando un ciclo infinito) In questo caso e' conveniente incrementare manualmente il ttl del pacchetto prima di rispedirlo, per far "sparire" dal traceroute gli hop fra noi e il router. Questa tecnica ci potra' tornare utile in alcune situazioni usando il route mangling.

N.B. Piu' il router che compromettiamo e' vicino al server di cui vogliamo sniffare il traffico, piu' possibilita' avremo che da li' passi tutto il traffico da e verso il server.

TOOLS

- Ettercap

- TunnelX

TRACCE LASCIATE

Supponendo di rimanere invisibili al traceroute, la traccia piu' evidente e' senza dubbio la riconfigurazione del router.

CONTROMISURE

- Password forti sul router per l'accesso a qualsiasi livello.

- Disabilitare o proteggere con community forti l'accesso in scrittura via snmp.

==========

MAN-IN-THE-MIDDLE REMOTO

ROUTE MANGLING

N.B. Prima di leggere questo paragrafo e' consigliabile consultare:
APPENDICE A - PROTOCOLLI DI ROUTING DINAMICO E LORO DEBOLEZZE e MAN-IN-THE-MIDDLE DA LOCALE A REMOTO - ROUTE MANGLING

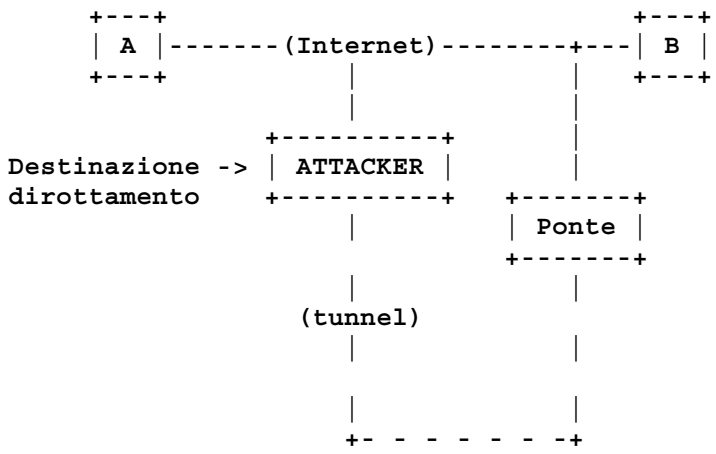
Rimane valido quanto dettto in precedenza per le netmask e il subnetting. Questa volta, pero', avremo anche bisogno di mandare degli update spoofati. Alcuni apparati (es. Cisco IOS) hanno la possibilita' di abilitare una funzione chiamata validate-update-source (disponibile per RIP e IGRP). Con questa funzione attivata, il router, prima di accettare un "update", controllera' che l'indirizzo sorgente dei pacchetti cada nella netmask dell'indirizzo dell'interfaccia fisica da cui ha ricvuto il pacchetto stesso. Questo limitera' le nostre possibilita' di "spoofare" gli update ma, come vedremo in seguito, non rappresentera' un grosso problema.

ATTACCO

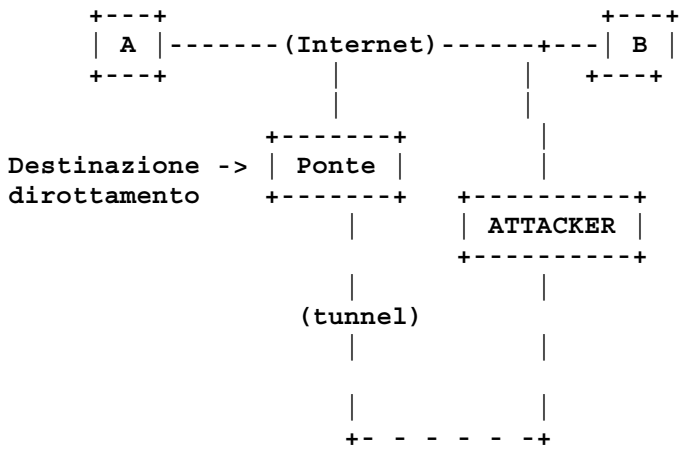
Premettiamo che questo tipo di attacco non e' per nulla deterministico. Infatti tutto e' molto dipendente dallo scenario, dai protocolli usati, dalla nostra posizione, dalla possibilita' che abbiamo di raccogliere informazioni riguardanti il routing e, nel caso di scenari molto complessi, dalle routing policies. In alcuni di questi scenari, le effettive possibilita' di attacco non solo a livello di m-i-t-m, ma anche ad esempio di DoS, sono ancora oggetto di studio.

Volendo intercettare il traffico da A a B, l'attacco consiste nel tracciare una rotta ideale fra A e B che passi attraverso di noi. Gli approcci possibili sono due: dirottare effettivamente il traffico verso la nostra posizione di "attaccante" o dirottarlo verso un punto della rete da cui siamo in grado di "raccoglierlo" (ad esempio un router su cui abbiamo creato un tunnel verso di noi). Anche in questo caso avremo il problema di rispedire il traffico al legittimo destinatario. Se abbiamo scelto di dirottare il traffico verso un router su cui abbiamo un tunnel, e' conveniente re-inoltrare il traffico semplicemente dalla nostra posizione di attaccante (ammesso che siamo abbastanza "lontani" da tale router). Se abbiamo scelto di far arrivare il traffico "a casa nostra", ci converra' re-inoltrarlo (come visto prima) attraverso un tunnel su di un router abbastanza lontano da noi, in posizione "strategica". Purtroppo non sara' sempre possibile trovare un percorso alternativo per reinstradare i pacchetti.

Soluzione 1:



Soluzione 2:



In seguito indicheremo come ATTACKER indifferentemente la box dell'attaccante o il router su cui ha un tunnel (Destinazione dirottamento).

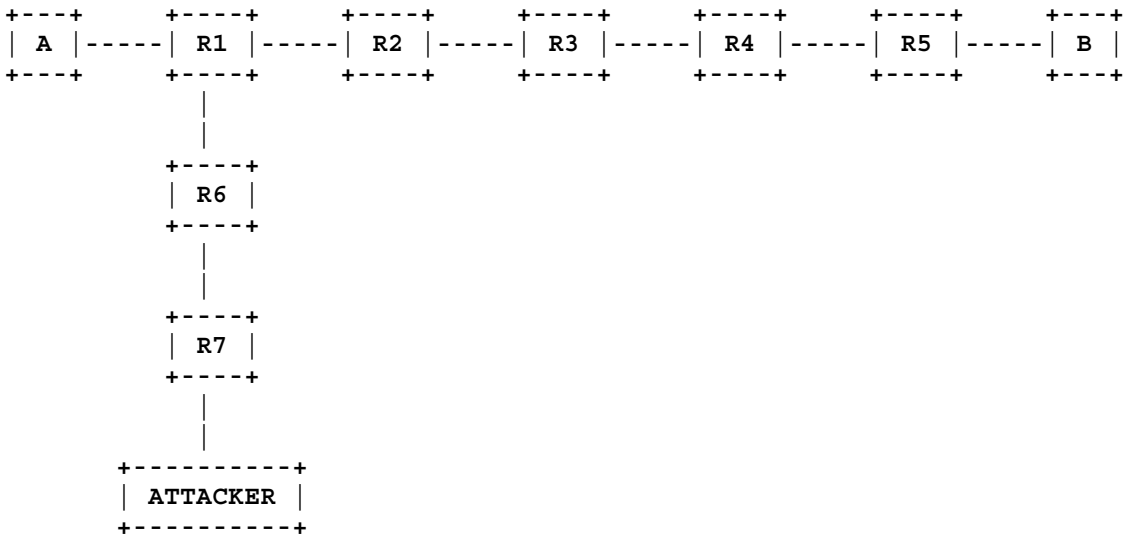
La prima cosa da fare, nell'intercettazione del traffico fra A e B, e' eseguire dei traceroute dalla nostra posizione di ATTACKER verso A e verso B, cercando di capire da quale punto del tragitto fra i due e' piu' conveniente cominciare a deviare il traffico (plausibilmente quello piu' vicino a noi e che ci permetta di avere una buona posizione per il reinstradamento dei pacchetti) e quale rotta dovremo far seguire al traffico per arrivare verso di noi. Eseguendo in seguito dei portscan e dei protoscan sulle macchine del tragitto, potremo avere un'idea piu' precisa dello scenario davanti a cui ci troviamo. A seconda del protocollo utilizzato esistono poi varie possibilita' per farsi "inviare" le tabelle di routing dei vari apparati (o parte di esse). Questo argomento verra' trattato in una versione futura di questo documento (o in uno piu' specifico esplicitamente dedicato al routing).

Vediamo ora dei tipici scenari (semplificati) davanti a cui ci potremo trovare.

- Il traffico inizia, finisce o tranista all'interno del nostro AS

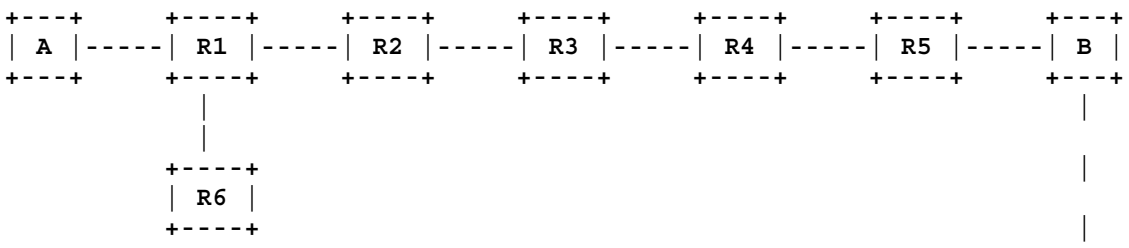
  In questo caso e' molto probabile che la gestione del routing dinamico sia affidata ad un unico IGP. Se piu' IGP vengono usati in combinazione all'interno dello stesso AS, andremo ad attaccare quello su cui sappiamo di avere piu' possibilita' di successo (come gia' visto avremo piu' probabilita' di successo attaccando ad esmepio RIP piuttosto che OSPF). Supponiamo che si tratti di IGRP.

Vediamo un primo scenario:



Dove:
A e' il punto in cui il traffico viene originato o transita per arrivare a B.
B e' il punto in cui il traffico arriva o transita per raggiungere la destinazione.
R* sono routers.

In questo caso bastera' semplicemente annunciarci a R7 come gateway per B.



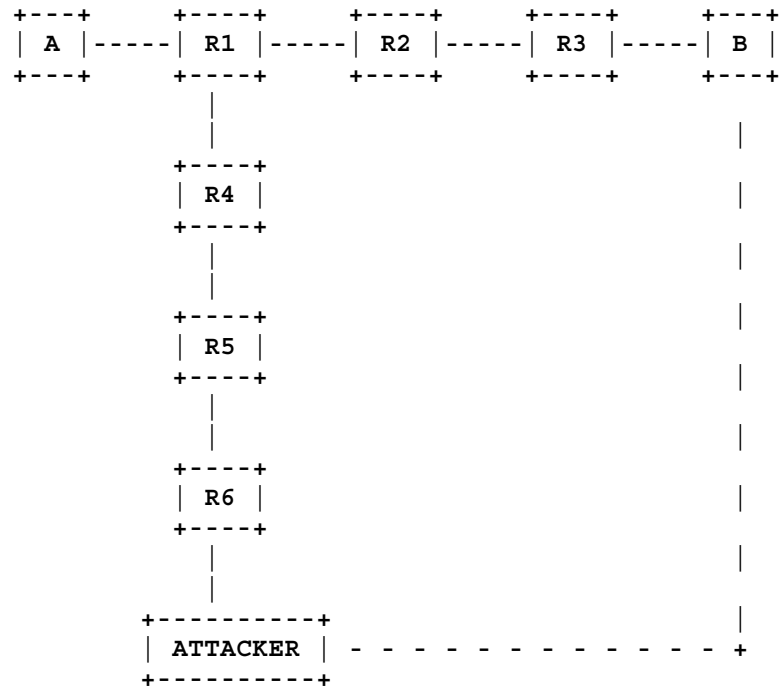
Legenda:

`---------` **connessioni reali**

`- - - - -` **connessioni fittizie**

Sara' R7 a propagare la nostra rotta "fittizia" a R6 che la propaghera' fino a R1. A questo punto R1 sapra' di poter raggiungere B attraverso R2 con 4 hop e "credera'" di poterlo raggiungere attraverso R6 con 3 hop. R1 scegliera' quindi la rotta con costo minore che passera' attraverso ATTACKER.

Vediamo ora un caso un po' piu' complesso:



Se ci annunciamo a R6 come gateway per B (come nel caso precedente), R1 vedra' di poter raggiungere B con 2 hop attraverso R2, e attraverso R4 con 4 hop. R1 continuera' a scegliere la rotta che passa da R2.

Per dirottare il traffico verso di noi saremo costretti a mandare pacchetti ai router del tragitto spoofandoci per il router successivo, "correggendo" le visioni. Ci annunceremo a R6 come gateway per B.

R6 dira' a R5 di poter raggiungere B in un hop.
R5 dira' a R4 di poter raggiungere B in 2 hop.
Da questo punto in poi, la rotta che noi abbiamo proposto comincera' a diventare piu' sconveniente rispetto a quella reale che proviene da R3. Noi provvederemo quindi a:

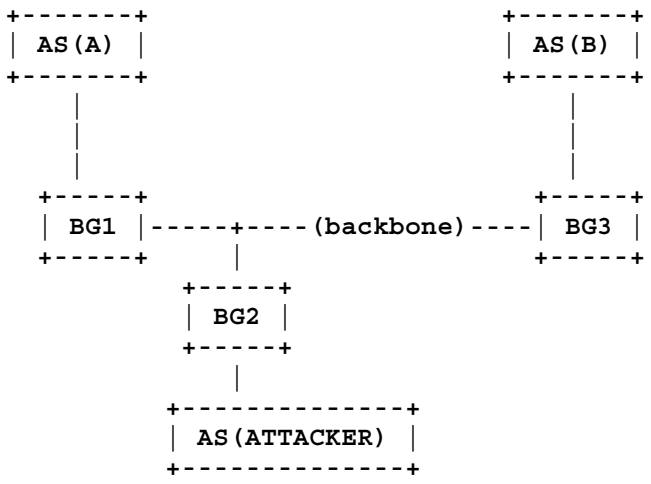
- Mandare degli update a R4 (fingendoci R5) e dicendo di poter raggiungere B in un hop.
- Mandare degli update a R1 (fingendoci R4) e dicendo di poter raggiungere B in un hop.

R1 scegliera' la rotta che passa da R4, credendo di poter raggiungere B in un solo hop; R4 scegliera' la rotta che passa da R5, credendo di poter raggiungere B in un solo hop (se lo reinoltrasse su R1 si creerebbe un loop) e cosi' via fino ad ATTACKER. Come e' facile intuire, in questo scenario l'abilitazione di validate-update-source non influenzera' minimamente la nostre possibilita' di spoofing.

N.B.
Le nostre visioni "spoofate" verranno periodicamente "contestate" dalle visioni reali che R5 mandera' a R4, e che R4 mandera' a R1 della nostra rotta. In questo caso ci potra' convenire inviare a R1 degli update fingendoci R2, e dicendo di poter raggiungere B con un alto costo in hop (ad esempio 16 in RIP per indicare un link down). Questo e' possibile solo nel caso in cui R1 non effettui il validate-update-source (R1 vedrebbe arrivare un pacchetto con sorgente R2 dalla linea su cui si aspetta di trovare R4) o se R1, R2 e R4 si trovano su di uno stesso canale condiviso. Comunque, con un po' di tempismo nell'invio degli update, riusciremo a convincere R1 (e anche R4, R5 e R6) che la nostra e' la rotta migliore in ogni caso.

N.B.2 Anche in questo caso valgono le considerazioni fatte precedentemente riguardo le netmasks (anche se nell'esempio abbiamo supposto si trattasse di IGRP che non permette di usarle).

- Il traffico non si trova mai a passare dal nostro AS



Dove:
AS(x) e' l'AS contenente x
BG* sono border gateways

Supponiamo che all'interno del nostro AS giri un IGP (ad esempio RIP) che siamo in grado di attaccare, e che sulla backbone giri BGP. Supponiamo anche

che BG2 sia configurato per ridistribuire su BGP le rotte ricevute da RIP. In questo caso potremo proporre a BG2 (via RIP) una rotta per B e lasciare che sia lo stesso BG2 a propagare la nostra rotta su BGP.

Ma....

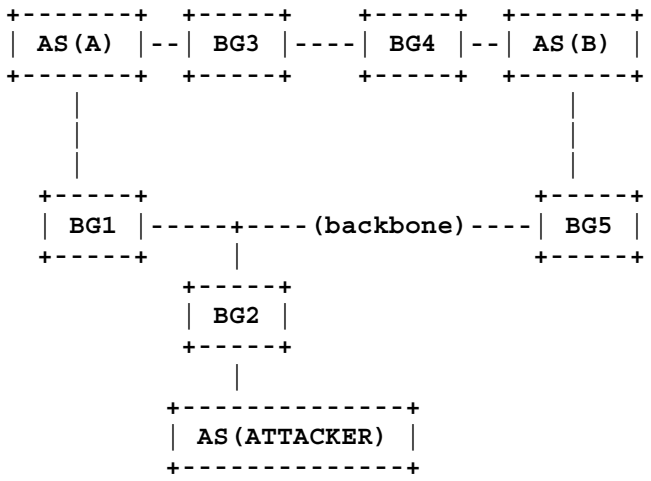
Una volta fuori dal nostro AS non potremo fare niente per "spingere" la nostra rotta.

Infatti:

- Non possiamo sfruttare le metriche visto che verranno "dimenticate" nel passaggio su BGP (a causa del sistema totalmente diverso usato da BGP per discriminare le rotte)
- Anche l'utilizzo di netmask grandi potrebbe non portare i risultati sperati. E' possibile infatti che il router sia configurato per fare "aggregate" delle rotte (per diminuire il traffico in rete e il volume delle tabelle di routing). E' possibile inoltre che BGP non scelga necessariamente la rotta con la netmask piu' specifica.

La cosa migliore che potremo fare sara' proporre la nostra rotta e sperare nella nostra buona stella.

Vediamo un altro caso:



Qui le cose si fanno piu' complesse, visto che, probabilmente, dovremo prima di tutto "intervenire" sull'IGP dell'AS di A, per far inoltrare il traffico per B verso BG1 (aumentando le nostre possibilita' di intercettazione). Una volta fatto questo, agiremo come nel caso precedente. Se in AS(A) viene usato IBGP, alcuni attributi della rotta BGP potranno essere propagati all'interno dell'AS , influenzando la scelta del gateway per uscire dall'AS.

Tutti questi ovviamente sono scenari d'esempio....la situazione puo' essere molto piu' complessa.

In tutti questi casi siamo stati in grado di intercettare il traffico da una sorgente verso una destinazione (HALF DUPLEX). Per ottenere un mitm FULL DUPLEX dovremo tentare anche il "tracciamento" della rotta di ritorno.

TOOLS

- IRPAS
- Nemesis
- rprobe

TRACCE LASCIATE

Non ci sono evidenti tracce dell'attacco.

CONTROMISURE

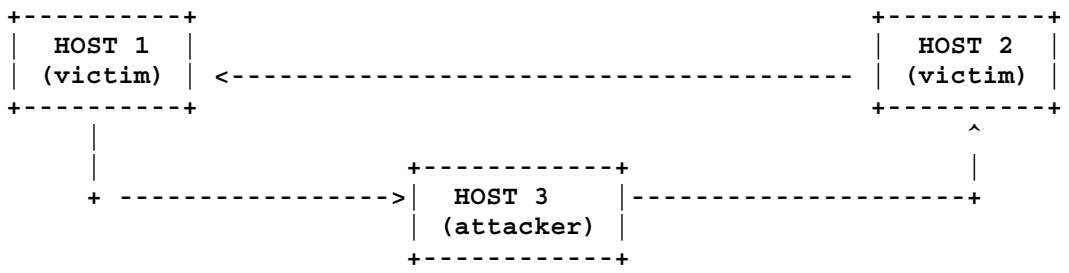
- La contromisura piu' efficace e' senza dubbio abilitare un autenticazione forte nei protocolli che la supportano

==========

FULL DUPLEX VS HALF DUPLEX

A seconda della capacita' di riuscire a dirottare solo uno o entrambi i versi della connessione l'attacco verra' chiamato mitm half duplex o mitm full duplex

HALF DUPLEX



FULL DUPLEX

E possibile ottenere un mitm full duplex con i seguenti metodi

- arp poisoning bidiriezionale
- traffic tunneling

E' possibile comunque monitorare entrambi i lati della comunicazione anche con gli altri metodi, se siamo in grado di intercettiare una conessione dall'inizio o nel caso di protocolli stateless.
Abbiamo due possibilita':

- Inoltrare i pacchetti rimpiazzando l'indirizzo sorgente con il nostro. Il destinatario del pacchetto rispondera' a quello che lui crede sia il mittente (noi); noi ci prenderemo carico di reinoltrare le risposte, dopo aver "sistemaro" gli indirizzi. Se il pacchetto ha come IP address destinazione quello della macchina attaccante (nel caso ad esempio si usi DNS spoofing), dovremo prenderci cura di modificare anche quello prima della rispedizione.
- Se si tratta di connessioni TCP possiamo accettare la connessione sulla nostra macchina attaccante e crearne un'altra col reale destinatario. In questo caso, se l'IP address destinazione dei pacchetti e' quello del reale destinatario, la nostra macchina agira' da "transparent proxy". Se l'indirizzo destinazione e' quello della nostra macchina, agiremo da "proxy" vero e proprio.

In questo caso pero' l'IP address con cui la macchina si presenta verso il server sara' quello dell'attaccante. Questo portera' a lasciare ulteriori tracce dell'attacco e risultera' in un insuccesso nel caso sul server siano presenti delle discriminazioni d'accesso (ACL) basate su indirizzi IP.

Questa distinzione e' importante perche' alcuni degli attacchi di seguito elencati possono essere portati a termine solo avendo il controllo di entrambi i versi della comunicazione.

==========

TIPOLOGIE DI ATTACCO

Il risultato piu' importante ottenibile con la tecnica del man in the middle e' la capacita' da parte di un attaccante di modificare il flusso di dati della connessione.

```
vediamo in dettaglio:
```

- SNIFFING

La cosa piu' ovvia che possiamo fare una volta ottenuto il "middle" e' sniffare i pacchetti che transitano attraverso in nostro host (attaccante). Sfruttare le tecniche di m-i-t-m locale (o da locale a remoto) puo' essere molto utile per sniffare il traffico anche in contesti di LAN commutate (come spiegato nel paragrafo relativo all'arp poisoning)

- HIJACKING

Un altro tipo di attacco che possiamo portare a termine e' l'Hijacking di una connessione. Poiche' tutto il traffico passa attraverso il nostro host, non c'e' niente di piu' facile. non dobbiamo preoccuparci di indovinare i numeri di sequenza della connessione, poiche' li vediamo passare. E' estremamente facile droppare la connessione da un lato e impossessarci dell'altro end point.

- INJECTING (solo con FULL DUPLEX)
La cosa che pero' ci permetter il man-in-the-middle rispetto ad un Hijack "normale" e' il poter aggiungere alla connessione quanti pacchetti vogliamo, pur mantenendo la connessione perfettamente sincronizzata. Per fare cio' sara' sufficiente modificare i numeri di sequenza dei pacchetti tcp ricevuti prima di forwardarli verso l'altro host. Memorizzando quanti byte sono stati inviati da un lato e dall'altro, con semplici addizzioni e sottrazioni si mantiene la connessione sincronizzata.

Nel caso si scelga una soluzione di tipo "proxy" questi problemi non sono rilevanti poiche' vengono instaurate due connessioni separate client-attaccante e attaccante-server.

- Command Injection
E' utile nel caso di sessioni di tipo login, telnet o ssh autenticate con metodi non riproducibili (es one time passwords). Infatti anche riuscendo a sniffare la password non e' possibile riutilizzarla per loggarsi successivamente sul server. Sfruttando la tcp injection si possono forgiare pacchetti simulando comandi del client o anche risposte del server, mentre la connessione e' attiva.

- Malicious code injection
E' possibile inserire ad esempio nelle pagine web lecitamente richieste script maligni o quan'altro (virus, etc.) vedi exploits per explorer 5.5 6)

- FILTERING

Cosi come possiamo modificare i numeri di sequenza dei pacchetti tcp, nulla ci vieta di modificarne anche il payload. Per portare a termine questo tipo di attacco e' necessario stare attenti a ricalcolare il checksum del pacchetto modificato da forwardare. In questo modo possiamo creare dei filtri a run-time che trasformeranno determinate stringhe in altre, il tutto mentre la connessione e' in corso.

Per come e' strutturato il protocollo un attaccante che ha il controllo su un solo senso della connessione sara' in grado di modificare il contenuto dei pacchetti senza pero' andare a cambiare la lunghezza degli stessi. (pena la desincronizzazione dei numeri di sequenza, dato che non saremo in grado di modificare gli ACK). Se invece si e' in ambito di FULL-DUPLEX valgono le ipotesi fatte per l'injecting.

- SSH v1 (solo con FULL DUPLEX)

Quando una connessione SSH v1 comincia, il server propone la sua chiava chiave pubblica al client.
Poiche' possiamo cambiare il contenuto dei pacchetti on-the-fly, possiamo sostituire la chiave pubblica del server con una generata al volo. In questo modo il client riceve una chiave pubblica di cui noi abbiamo la rispettiva chiave privata. Quando il client manda il pacchetto contenente la chiave di sessione cifrata con la chiave pubblica (fake) del server, noi saremo in grado di decifrarla e di sniffare la chiave 3DES di sessione. A questo punto noi criptiamo il pacchetto con la chiave pubblica corretta del server (precedentemente salvata) e la manderemo al demone SSH. La connessione si stabilisce normalmente, ma noi possediamo la chiave di sessione. A questo punto possiamo decifrare tutto il traffico semplicemente usando la chiave 3DES sniffata. La connessione rimane attiva anche se noi smettiamo di fare arp poisoning poiche' non stiamo facendo da proxy per la connessione, ma solo decifrandola con la chiave di sessione. Sono comunque possibili soluzioni di tipo "proxy" ottenibili con tutte le altre tecniche di m-i-t-m. In questo caso si avranno due scambi di chiavi separati, uno fra client e attaccante, l'altro fra attaccante e server.

La TEORIA alla base di questo attacco e' utilizzabile anche contro altri canali cifrati come ad esempio SSL.

- Parameters and banner substitution
E' possibile sostituire i parametri che il serve e il client si scambiano per modificare il comportamento di entrambi. Ad esempio e' possibile far si che una connessione che richiede SSH2 sia stabilita in SSH1. Se il client non e' settato a "Strict" vedendo la risposta del server decide quale versione di SSH utilizzare.
Risposte del server:

SSH-1.99 -- il server supporta ssh1 e ssh2
SSH-1.51 -- il server supporta SOLO ssh1

quindi se noi vogliamo forzare un client ad usare SSH version 1, possiamo creare un filtro che sostituisce il banner SSH-1.99 con SSH-1.51. In questo modo la connessione sara' instaurata usando il protocollo 1 facilmente vulnerabile al m-i-t-m (vedi sopra).

Questo puo' portare anche ad aggirare il problema della coppia <server, chiave pubblica> salvata su file dal lato client (ad esempio known_hosts). Se il server supporta entrambe le versioni del protocollo, il client scegliera' molto probabilmente la SSH v2 ad ogni connessione, salvando la coppia <server, chiave> relativa a questa versione del protocollo. Supponiamo a questo punto che un client si connetta ad un server a cui si era gia' connesso in precedenza e sia vittima di un attacco di questo tipo. Il client non trovera' la coppia relativa a quel server per la versione 1 del protocollo, dato che non era mai stata scelta in precedenza per quel server, non potra' quindi accorgersi che si tratta di una chiave "fake", e non stampera' il famoso banner di warning.

- IP SEC FAILURE (solo con FULL DUPLEX a seconda della configurazione)
Per iniziare il "dialogo" tra due macchine IPSec utilizza la porta 500 UDP per effettuare gli scambi ISAKMP/IKE e generare il KeyMaterial con cui in seguito verranno cifrati/autenticati i pacchetti. Trovandoci "nel mezzo" e possibile "droppare" tutti i pacchetti per la porta 500 e mandare indietro dei "port unreachable". Ognuno degli estermi della comunicazione credera' quindi che l'altro non sia in grado di "parlare" ipsec. Se l'iniziatore della connessione (generalmente il client) e' configurato per fare il fall-back su connessioni non ipsec in caso di fallimenti del protocollo, e il responder (il server) accetta connessioni non securizzate, le due macchine cominceranno a "parlare" in chiaro in maniera trasparente all'utente (a meno che anche lui non stia sniffando la rete), e noi saremo in grado di monitorare/modificare il traffico che non sara' piu' ne' cifrato ne' autenticato. Questo procedimento chiaramente fallisce nel caso ipsec sia utilizzato in tunnel mode (ad esempio su un VPN gateway) per raggiungere indirizzi non routati o comunque non accessibili direttamente dall'esterno (internet) , visto che bloccando la creazione delle SecurityAssociations, impediremo anche la creazione del tunnel rendendo impossibile la comunicazione fra le due macchine.

==========

CONCLUSIONI

E' importante sottolineare come la sicurezza di una nostra comunicazione sia affidata non soltanto ad una corretta configurazione del client (per evitare ad esempio ICMP Redirect, ARP Poisoning etc.) e dell'infrastruttura del nostro interlocutore (es. DNS dynamic update), ma anche alla robustezza di apparati di "terzi", su cui non possiamo avere nessun tipo di controllo e di garanzia (es. Tunnelling e Route Mangling).

Il modo migliore per proteggere i nostri dati che viaggiano in rete e' quindi l'utilizzo corretto di suite crittografiche sicure (ovviamente sia dal lato client sia dal lato server), per la protezione a livello rete (es. IPSec), a livello trasporto (es. SSLv3) o a livello applicativo (es. PGP).

==========

APPENDICE A - PROTOCOLLI DI ROUTING DINAMICO E LORO DEBOLEZZE

Gli algoritmi di routing e i protocolli dinamici permettono ai router di modificare le proprie decisioni a seconda dei cambiamenti di topologia della rete o in base a stime o misurazioni del traffico. Dal punto di vista del routing, internet puo' essere vista come una collezione di sottoreti o di sistemi autonomi (Autonomous System, o AS, intesi generalmente sotto un'unica amministrazione tecnica e con un'unica routing-policy) connessi insieme da canali ad alta capacita' chiamati dorsali (backbone). Generalmente un router conosce tutti i dettagli su come instradare pacchetti all'interno della sua "regione", ma non sa nulla della struttura interna delle altre regioni. Quando deve instradare un pacchetto al di fuori del suo AS, lo inoltra verso il ruoter che connette il suo AS alla backbone (Border Gateway). I router della backbone vedranno ogni AS come un'unica entita' rappresentata dal suo border gateway; in questo modo saranno in grado di inoltrare il pacchetto verso il bordergateway dell'AS destinazione; a questo punto potra' facilmente essere inoltrato verso la LAN o l'host d'arrivo. Questa struttura prende il nome di routing gerarchico (in questo caso con una gerarchia di due livelli). Nella realta' i livelli possono essere molti di piu', anche se non esiste nessuna struttura o regola fissa, infatti gli AS possono essere connessi a piu' di una backbone, o anche direttamente connessi fra loro in peering, etc.

A seconda del modo in cui i protocolli dinamici scambiano le informazioni relative al routing, alcuni saranno piu' adatti per l'utilizzo all'interno di un AS (Interior Gateway Protocols), mentre altri si adatteranno meglio all'utilizzo sulle backbone e al routing fra gli AS (Exterior Gateway Protocols). I protocolli di routing dinamico si distinguono inoltre per il tipo di approccio con cui permettono di arrivare alla determinazione dei cammini minimi.
I metodi piu' utilizzati sono:

- Distance Vector: questi protocolli funzionano facendo in modo che ogni router mantenga una tabella contenente la migliore distanza conosciuta per ogni destinazione e quale canale utilizzare per raggiungerla. Queste tabelle sono aggiornate scambiando le informazioni coi vicini. Questi tipi di algoritmi hanno dei problemi come la scarsa velocita' di convergenza e il problema del "conteggio all'infinito" (in parte ovviabile con lo split-horizon).

- Link State: Generalmente piu' performanti dei precedenti, il funzionamento dei protocolli di tipo linkstate si puo' suddividere in 3 punti.

  Ogni router deve:
  - Scoprire i propri vicini e il ritardo o il costo per ognuno di essi con dei pacchetti di HELLO
  - Costruire e spedire a tutti gli altri router coinvolti un pacchetto con tutto quello che ha appena scoperto
  - Una volta ottenute informazioni sullo stato di ogni canale, costruire il disegno di rete e calcolare il cammino minimo per le varie destinazioni.

  A differenza del Distance Vector Routing, in questo caso i pacchetti conterranno solo informazioni sui vicini e non sulle varie destinazioni, mentre OGNI router conosce l'intera topologia di rete o almeno ne ha una sua visione. Dato che i pacchetti vengono spediti a tutti i router che partecipano al Link State Routing, il protocollo prevede dei controlli per evitare loops, basati sull'utilizzo di campi come "numero di sequenza" e "eta'".

L'utilizzo di questi protocolli puo' essere rilevato per mezzo di un "port scanner", per quelli che girano sopra il layer di trasporto, o con dei "protocol scanner" appositi per gli altri. Se ci troviamo in LAN con uno di questi apparati, potremo semplicemente "sniffare" il segmento di rete (gli update vengono generalmente mandati in broadcast o multicast) per accorgerci della loro presenza e catturare numerose informazioni che ci possono tornare utili, come l'identificativo dell'AS o le password in cleartext. Per i protocolli che usano multicast e' anche possibile che i pacchetti vengano inoltrati da un segmento ad un altro.

Entriamo ora nello specifico dei protocolli piu' diffusi. Non ci soffermeremo su tutti gli aspetti implementativi di questi protocolli, ma ci limiteremo ad esaminare gli aspetti che interessano ai fini dell'attacco, i punti di forza, le debolezze e le possibilita' che abbiamo di sfruttarle.

- IGRP (Interior Gateway Routing Protocol):
  Protocollo creato da Cisco. Ogni pacchetto contiene l'identificativo dell'AS a cui le rotte fanno riferimento. Sono possibili 65535 AS diversi ed e' necessario conoscere questo valore per fornire degli "update" validi. Non supporta alcun tipo di autenticazione. Supporta diverse metriche: delay, bandwidth, reliability, load e hop count.
  Non supporta le netmask. No host route. Non appoggiandosi ad alcun protocollo di trasporto e non supportando il sequencing e' possibile forgiare e mandare dei singoli pacchetti di update e spoofarne l'indirizzo sorgente. Essendo un protocollo di tipo distance vector, i pacchetti possono contenere anche solo la rotta che vogliamo "aggiungere" o "modificare". Possiamo mandare degli update a intervalli regolari per mantenere "vive" le nostre rotte. E' un protocollo che non e' piu' praticamente usato.

- EIGRP (Extended Interior Gateway Routing Protocol)
  Come dice il nome stesso e' un estensione del precedente, piu' potente e flessibile. Pur essendo un protocollo di tipo Distance-Vector implementa delle neighbor relationships coi vicini e un topology database. Manda soltanto i cambiamenti di topologia nei suoi updates. Puo' richiedere esplicitamente una rotta (QUERY) ai vicini, che possono inoltrare la richiesta. Supporta le netmasks. Utilizza degli speciali pacchetti di tipo HELLO (mandati in multicast) per permettere a un router di annunciarsi come "neighbor" ai suoi vicini. Se non intendiamo mandare updates spoofando l'indirizzo di uno degli apparati gia' riconosciuti dal router destinazione come vicino, dovremo preoccuparci di annunciarci come neighbor. Dal punto di vista della sicurezza, questa estensione implementa il sequencing e permette l'utilizzo di una autenticazione crittografica delle rotte (MD5).

- RIPv1 e RIPv2 (Routing Information Protocol)
  E' anch'esso un IGP. L'unica metrica supportata e' l'hop count. Non e' adatto a reti di grandi dimensioni poiche' la massima distanza misurabile sono 15 hop (16 vuol dire "linea morta"). A differenza dei precedenti non supporta la distinzione degli AS. La v1 non permette di specificare la netmask delle rotte (netmask fissata dalla classe). La v2 supporta l'autenticazione con password in chiaro (16 caratteri max) o le signature MD5 (solo per Cisco). Con RIPv1 ogni router puo' comunicare unicamente la sua visione delle rotte. RIPv2 puo' inoltrare anche informazioni per "conto terzi" inserendo nel pacchetto un campo "gateway". RIPv2 supporta anche un campo "ruote tag" con l'identificativo dell'AS , per essere usato in combinazione con un EGP (ad esempio BGP4). Il protocollo gira su UDP (porta 520) e, come i precedenti, e' di tipo Distance Vector.
  Gli attacchi sono quindi in tutto simili a quelli possibili con IGRP. OSPF avrebbe dovuto rimpiazzare RIP, ma quest'ultimo risulta ancora abbastanza usato.

- OSPF (Opens Shortest Path First)
  E' un protocollo di tipo link state. Puo' essere usato come IGP o anche come EGP. Non si appoggia a nessun protocollo di trasporto. Ogni router comunica con gli altri mandando in "flooding" (multicast) dei pacchetti di tipo LSA (Link State Advertisment) contenenti anche i campi "age" e "sequence".
  Supporta le subnet netmasks e il ToS. Ogni router viene riconosciuto per mezzo di pacchetti di HELLO che vengono mandati sulla rete ogni circa 10 secondi. Quando un router riceve un pacchetto di HELLO comincia un processo per la sincronizzazione del suo database con quello degli altri router. OSPF puo' essere configurato per usare nessuna autenticazione o autenticazioni in clear-text (alcune implementazioni con MD5). La difficolta' di un attacco a OSPF risiede nel tipo di approccio utilizzato dal protocollo stesso (link state). Non possiamo, infatti, come nei casi precedenti, "iniettare" delle rotte, ma dovremo forgiare pacchetti con una visione completa dei canali del router. Inoltre, un router che riceve informazioni su di un canale, si aspetta di riceverle da ENTRAMBI gli end-point del canale, ed eseguira' generalmente una media delle metriche ricevuti dai due end-point. Dei pacchetti forgiati in "blind" porteranno generalmente ad inconsistenze nelle visioni della topologia, creando loops o broken-links, o semplicemente la non convergenza dell'algoritmo. L'attacco diventerebbe piu' fattibile nel caso fossimo in grado di intercattare e modificare gli LSA che transitano su di un segmento di rete. In questo caso potremo creare delle visioni "plausibili" dei canali, con age e sequence corrette. In questo modo potremo anche "sniffare" le password, qualora venga utilizzata un'autenticazione di tipo clear-text.Un attacco di questo tipo, seppur possibile in teoria, risulta comunque molto complesso. Inoltre, se siamo in grado di monitorare un segmento di rete, possiamo trovare altri modi piu' semplici per intercettare il traffico.

- BGP4 (Border Gateway Protocol)
  E' un protocollo usato principalmente per il routing fra gli AS come rimpiazzo per il piu' vecchio EGP. La versione 4 del protocollo e' la piu' recente (1998) e gran parte di internet sfrutta questo tipo di protocollo. Si appoggia a TCP come protocollo di trasporto (porta 179). BGP e' un protocollo complesso ed essendo pensato per il routing su larga scala permette di prendere delle decisioni basandosi anche su fattori di carattere "politico". Infatti, pur essendo un protocollo di tipo distance vector, permette l'enumerazione di tutti gli AS che un pacchetto deve attraversare per raggiungere ogni destinazione. In questo caso gli AS sono suddivisi in Stub(unica connessione ad un altro AS), Multihomed(connesso a piu' AS), Transit(connesso a piu' AS e si prende carico di inoltrare traffico di transito per altri AS). Permette inoltre la configurazione e la trasmissione di una o piu' community (identificate dal nome e dai suoi membri) che possono essere utilizzate nelle routing policies. Non usa le metriche per le rotte, ma una serie di attributi per discriminarle. Supporta le netmasks. Comprende un gran numero di messaggi, quello che piu' ci interessa e' senza dubbio "UPDATE". Alcune implementazioni di BGP non utilizzano alcuno schema di autenticazione come default. Possono comunque essere configurati, per le sessioni (sul messaggio di OPEN), gli schemi di autenticazione visti in precedenza, con tutte le conseguenze del caso. BGP non ha un suo metodo di sequencing, ma si appoggia a quello del TCP.

  Le problematiche relative all'injection di pacchetti o allo spoofing della sorgente sono quindi le stesse che si hanno nel caso di attacchi di tipo session-hijacking e blind-ip-spoofing (numeri di sequenza predicibili o meno, etc). Uno schema di autenticazione forte e l'utilizzo del TCP rendono l'injection remota di rotte via BGP molto complessa. Anche in questo caso la possibilita' di monitorare un segmento di rete dove "gira" BGP rende possibile l'utilizzo di tools come ettercap (o dsniff insieme ad hunt) per sniffare le password, e iniettare o modificare gli UPDATEs per modificare le rotte a nostro piacimento. Un altro approccio possibile per

attaccare BGP e' quello di iniettare delle rotte (ad esempio via RIP) sul BorderGateway, e lasciare che sia lui a ridistribuirle su BGP.

- IBGP, IPX RIP, CIDR supernetting etc.
  Prossime versioni del documento.....

```
================================================================================
vim:ts=3:expandtab:textwidth=80:wrap
================================================================================
```