



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Arnot: un tool per il monitoraggio di una LAN basato su ARP poisoning

Laureando: Michele Massaro

Relatore: Prof. Paolo Bertasi

Anno Accademico 2011/2012

Indice

1	Introduzione	1
2	Protocollo ARP	3
2.1	Descrizione	3
2.2	Ottimizzazioni	4
2.3	Problemi	4
3	ARMOT	7
3.1	Linguaggi e librerie utilizzate	8
3.2	Interfaccia	8
3.2.1	Scoperta di altri host nella rete	9
3.3	Poisoning	11
3.4	Funzioni	12
3.4.1	Packet Reader	12
3.4.2	Real Time Communication	13
3.4.3	Broadcast Poisoning	14
3.4.4	Forward	14
3.5	Interfaccia Testuale	16
3.5.1	Comando "IPs"	16
3.5.2	Comando "Connections"	17
3.5.3	Comando "Broadcast"	18
3.5.4	Comando "Read"	18
3.5.5	Comando "Export"	20
4	Esempio di funzionamento	21
4.1	Cattura flusso VNC	21
4.1.1	Stato della rete	21
4.1.2	Utilizzo di Armot	22
4.1.3	Risultati	23
4.1.4	Manipolazione	25
4.2	Cattura flusso Internet	27
4.2.1	Stato della rete	27
4.2.2	Utilizzo di Armot	27
4.2.3	Risultati	27
4.2.4	Manipolazione	29
4.3	Conclusioni dalle prove effettuate	29
5	Conclusioni	31
	Bibliografia	33
	Elenco delle figure	35

Capitolo 1

Introduzione

L'obiettivo di questa tesi è lo sviluppo di una applicazione in Java che permetta di monitorare e controllare una propria rete locale con metodi più avanzati dei normali strumenti di diagnostica.

Il problema principale per questo tipo di applicazioni è che in una rete basata su switched ethernet i pacchetti non possono essere visti da nessuno ad esclusione del mittente ed il destinatario.

A meno che l'amministratore della rete non utilizzi switch e router professionali non c'è modo di raccogliere informazioni sul traffico dei vari host. Una rete LAN casalinga è un ottimo esempio di questa situazione: difficilmente troveremo "switch gestiti" o router professionali da centinaia di euro.

Proprio per risolvere questo problema è stato necessario scrivere un software che permetta di "forzare" il normale funzionamento della LAN, e per renderlo possibile sono state usate tecniche attinenti al mondo dell'hacking.

Prima di procedere alla descrizione e al test del software, verrà illustrato il protocollo ARP, cioè quello che ci ha permesso di raggiungere il nostro scopo.

Capitolo 2

Protocollo ARP

2.1 Descrizione

Attualmente lo standard più diffuso per le reti locali è Ethernet. Quest'ultimo, identifica gli host in base ad un indirizzo univoco chiamato indirizzo MAC (o fisico) a differenza di Internet dove ciascun host viene individuato grazie all'indirizzo IP o internet (nello specifico IPv4). Gli indirizzi IP sono assegnati indipendentemente dagli indirizzi fisici di una macchina, ma è bene sottolineare come due macchine qualsiasi possono comunicare solo se conoscono gli indirizzi fisici di rete; sorge quindi il problema di associare agli indirizzi IP quelli reali del livello fisico. La soluzione viene affidata al protocollo ARP (Address Resolution Protocol)¹, che ha quindi la funzione di mappare gli indirizzi internet a 32 bit in indirizzi hardware a 48 bit.

Quando un host vuole ricavare l'indirizzo MAC di un altro host di cui conosce l'IP, invia in broadcast una richiesta ARP contenente il proprio MAC address e l'indirizzo IP del destinatario. In ogni dispositivo che riceve la richiesta, il protocollo ARP verifica se viene richiesto il proprio MAC address (confrontando l'IP di destinazione con il proprio), e in caso affermativo provvederà ad inviare al richiedente una risposta ARP contenente il proprio indirizzo MAC.

Per comprendere il funzionamento passo-passo di questo protocollo prendiamo come esempio la figura sottostante (figura 2.1):

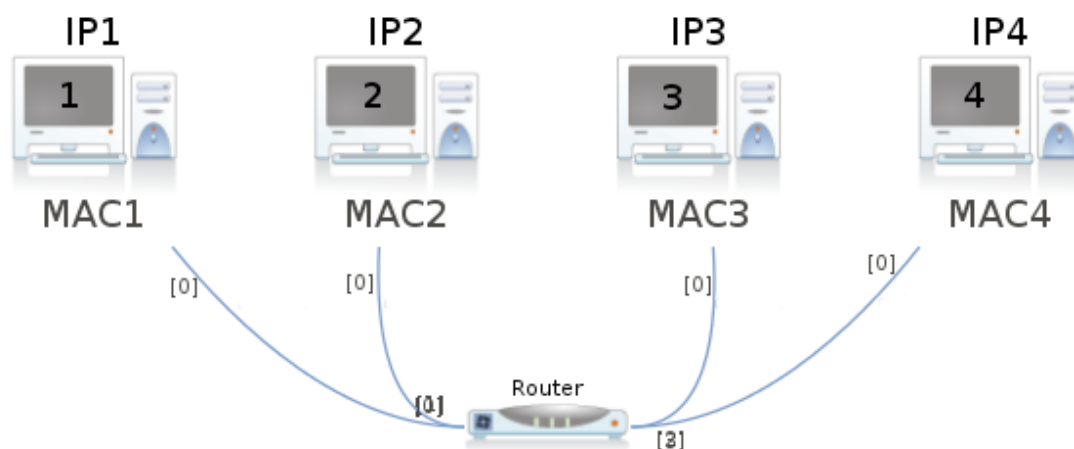


Figura 2.1: Esempio di rete locale

Supponiamo che l'host 1 voglia inviare dei pacchetti all'host 2, e di quest'ultimo conosca solo l'indirizzo IP. Per poter creare dei pacchetti del livello data link in uso (ethernet) è necessario conoscere anche il MAC address del destinatario. Per risolvere il problema si usa un approccio simile a quello di un automobilista che non conoscendo la strada deve chiedere informazioni. Di conseguenza l'host 1 invia in broadcast una ARP request, cioè un pacchetto speciale in cui è inserito il suo IP e MAC address, nel quale viene specificato come ricevente l'IP dell'host 2. Il pacchetto raggiungerà tutti gli host della sottorete, ma solo l'host con IP indicato come ricevente lo prenderà in considerazione. A questo punto l'host 2 può rispondere con un ARP reply (non più in broadcast) all'host 1, indicando il suo MAC Address. Ora entrambi possiedono gli indirizzi necessari per poter inviare pacchetti da 1 a 2 e viceversa.

Per evitare di dover eseguire questa operazione ad ogni connessione, ogni computer/router possiede una ARP Table, cioè una tabella in cui vengono registrate le associazioni IP - MAC conosciute. Ciascuna entry ha un timeout che dipende dal sistema operativo, necessario per non rendere obsoleti i dati contenuti.

2.2 Ottimizzazioni

Per migliorare il protocollo ARP sono state effettuate alcune ottimizzazioni, come il gratuitous ARP. Questo tipo di pacchetto ARP viene usato solitamente quando un nuovo host si connette alla rete (ad esempio all'accensione). L'host invia una ARP request in broadcast senza indicare uno specifico IP ricevente, in questo modo ogni host della rete lo legge e aggiunge l'entry del nuovo host alla sua ARP Table. Viene chiamato gratuitous ARP poiché l'host che lo invia non ha necessità di comunicare con altri, lo spedisce solo per favorire le connessioni successive ed evitare agli altri host di dover inviare ulteriori ARP request.

Una seconda ottimizzazione, è quella di accettare una ARP reply anche se non è stata effettuata alcuna ARP request.

2.3 Problemi

Purtroppo il protocollo ARP soffre di un grosso problema. Come si può notare dalla descrizione precedente, non viene effettuata alcuna autenticazione degli host: quando viene effettuata una richiesta un certo host risponde. Ma c'è la sicurezza che chi risponde sia veramente chi dice di essere?

Prendiamo in considerazione l'immagine sottostante (2.2):

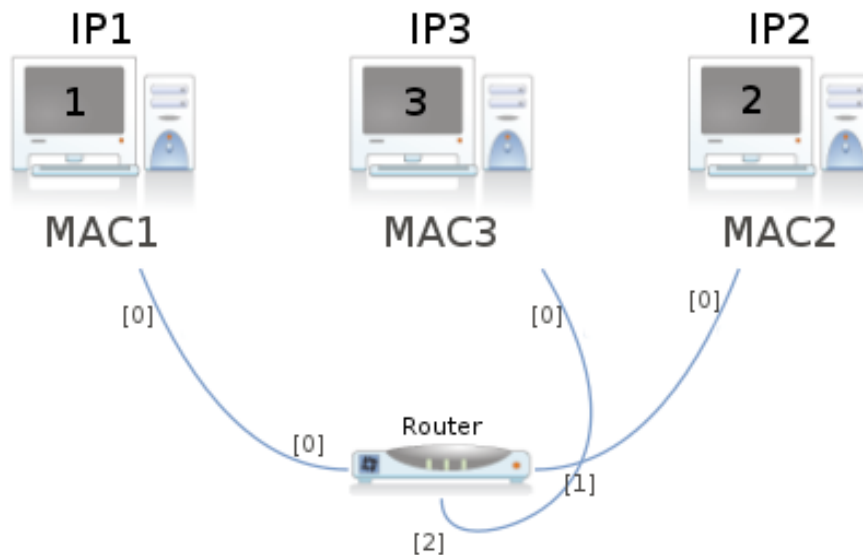


Figura 2.2: Intercettazione del flusso tra l'host 1 e 2 da parte del terzo

come nell'esempio precedente l'host 1 vuole comunicare con l'host 2. Quindi 1 invia una ARP request in broadcast sperando che l'host 2 gli risponda. Essendo un pacchetto in broadcast ogni host nella sottorete lo vede, e solitamente solo l'interessato lo legge, gli altri lo scartano. Nessuno però vieta all'host 3 di leggere la richiesta e di rispondere fingendo di essere l'host 2, cioè inserendo nella ARP Reply l'IP dell'host 2 ed il suo MAC Address. Dopo la risposta, nella tabella ARP dell'host 1 ci sarà una entry errata, che associa l'IP dell'host 2 al MAC Address dell'host 3. La tabella è stata "avvelenata", da cui il nome ARP poisoning. Da questo momento, ogni pacchetto inviato da 1 a 2 avrà come MAC Address di destinazione quello dell'host 3, che quindi riceverà i pacchetti e potrà decidere di bloccarli o inoltrarli, e nel secondo caso sarà molto difficile per gli altri host della rete rilevare questo doppio passaggio.

L'attacco descritto viene chiamato "Man in the Middle", proprio perché è presente un host "invisibile" che si interpone tra altri due, e che può leggere e manipolare il contenuto della comunicazione.

L'ARP Poisoning però, può essere usato anche per altri scopi ovvero come strumento avanzato per monitorare ed amministrare la propria rete LAN, come nel caso dell'applicazione sviluppata per questa tesi, Armot.

Capitolo 3

Armot

Armot (ARp MOnitoring Tool) è nato come tool per il monitoraggio e la gestione delle reti locali.

Le reti ethernet su cui possiamo operare possono essere di due tipi: ethernet normale o switched ethernet.

Il primo tipo è caratterizzato dalla presenza di uno o più hub come nodi della rete, da cui parte un cavo per ogni host; ogni hub ha solo il compito di inoltrare ogni pacchetto entrante in una porta, nelle altre, senza compiere alcuna azione di controllo. In questo modo i pacchetti inviati raggiungono tutti gli host della rete, e sarà compito di questi ultimi decidere di scartarli o accettarli.

Tuttavia questo approccio ha un difetto, in quanto l'hub connette elettricamente i vari host come se i loro cavi di rete fossero saldati tra loro, in questo modo durante la trasmissione tra due host viene occupata la banda anche di chi non sta trasmettendo in quel momento, cioè l'hub non aumenta la capacità totale della rete, ma la divide tra tutti gli host connessi. Se questo all'inizio non era un problema, lo è diventato quando il numero di dispositivi collegati alla rete è iniziato ad aumentare, e si è pertanto sentito il bisogno di trovare una soluzione.

L'espedito risolutivo è stato la sostituzione degli hub con gli switch (da cui il nome switched ethernet); quest'ultimo, a differenza del precedente, non inoltra ogni pacchetto ricevuto in tutte le altre porte, ma solo in quella corretta. Per farlo si basa sul MAC address indicato nel pacchetto. In questo modo la banda viene occupata solo a chi trasmette e riceve, permettendo a tutti gli host di poter usare la banda totale, e non una frazione come nell'ethernet normale.

Oltre a quanto appena esposto, un'altra positiva caratteristica determinata dall'utilizzo degli switch, è che mentre precedentemente ogni trasmissione veniva ricevuta da tutti gli host, ora gli unici che sono a conoscenza dell'esistenza di un pacchetto sono il mittente ed il ricevente; questo è sicuramente un ulteriore vantaggio per gli utilizzatori della rete in quanto viene garantita una maggiore privacy.

Ma, dal punto di vista di un amministratore della rete può essere un problema, perché risulta impossibile analizzare il traffico o capire quale host sta occupando la banda.

Visti i vantaggi offerti e la diminuzione del costo dei componenti elettronici, al giorno d'oggi quasi la totalità delle LAN è implementata su switched ethernet, e per questo motivo ci si è basati su questo modello per la scrittura del software.

Come si può immaginare, l'impossibilità di vedere il traffico altrui è un grosso handicap per una applicazione che si prefigge l'obiettivo di monitorare l'intera rete, e pertanto è stato necessario usare dei metodi alternativi per raggiungere lo scopo desiderato, e come si può notare dal significato del nome Armot, è stato utilizzato l'ARP poisoning.

Nella maggior parte dei casi viene adoperato con finalità di hacking, ma nel nostro caso viene impiegato per manipolare il traffico e spostare i flussi desiderati attraverso un certo host di controllo, avendo quindi la possibilità di controllare i servizi utilizzati e la banda usata, e qualora fosse necessario, poter bloccare un qualsiasi host della LAN.

3.1 Linguaggi e librerie utilizzate

Il software è stato scritto in java, che purtroppo è a un livello troppo alto per permettere la manipolazione dei pacchetti. Per ovviare a questo problema è stata utilizzata la libreria open source JPCap², che tramite il Java Native Interface (JNI) usa delle librerie esterne in C per la cattura e l'invio di pacchetti.

Purtroppo l'utilizzo di questa libreria non ha portato solo dei vantaggi: essendo una libreria esterna rende Armot meno "portable", cioè a differenza di un software puramente in Java non può essere esportato in un altro computer ed avviato, ma per funzionare richiede la preventiva installazione della libreria JPCap.

Inoltre, dato il basso livello richiesto dalla libreria, è necessario ottenere i privilegi di root per l'esecuzione.

Questi compromessi si sono resi necessari, in quanto non sarebbe stato possibile scrivere questo software esclusivamente in Java.

Fortunatamente questa libreria è disponibile per Linux, OS X, Windows e come codice sorgente.

Inoltre è stato utilizzato Eclipse³ come IDE per l'intero progetto.

3.2 Interfaccia

La prima cosa che viene presentata all'avvio, è una finestra in cui viene chiesto all'utente di selezionare l'interfaccia di rete che vuole utilizzare, sia essa ethernet o wireless.

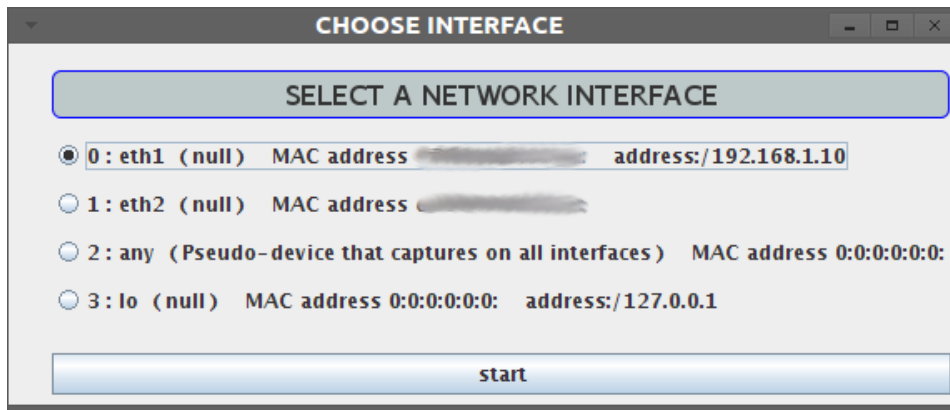


Figura 3.1: Scelta dell'interfaccia di rete

Effettuata la scelta, viene presentata la finestra principale (Figura 3.2) da cui si può accedere a tutte le funzioni del programma.

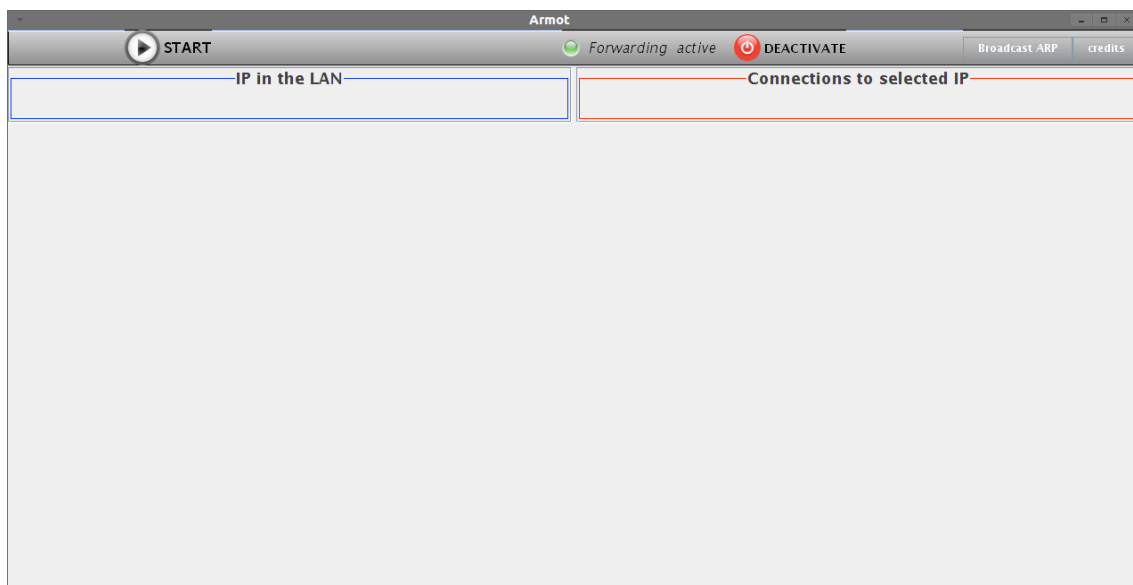


Figura 3.2: Finestra principale

3.2.1 Scoperta di altri host nella rete

Per iniziare la cattura dei pacchetti è necessario cliccare sul tasto start, in questo modo il sistema non scarnerà i pacchetti ARP non indirizzati a se stesso, e li utilizzerà per scoprire quali host stanno comunicando.

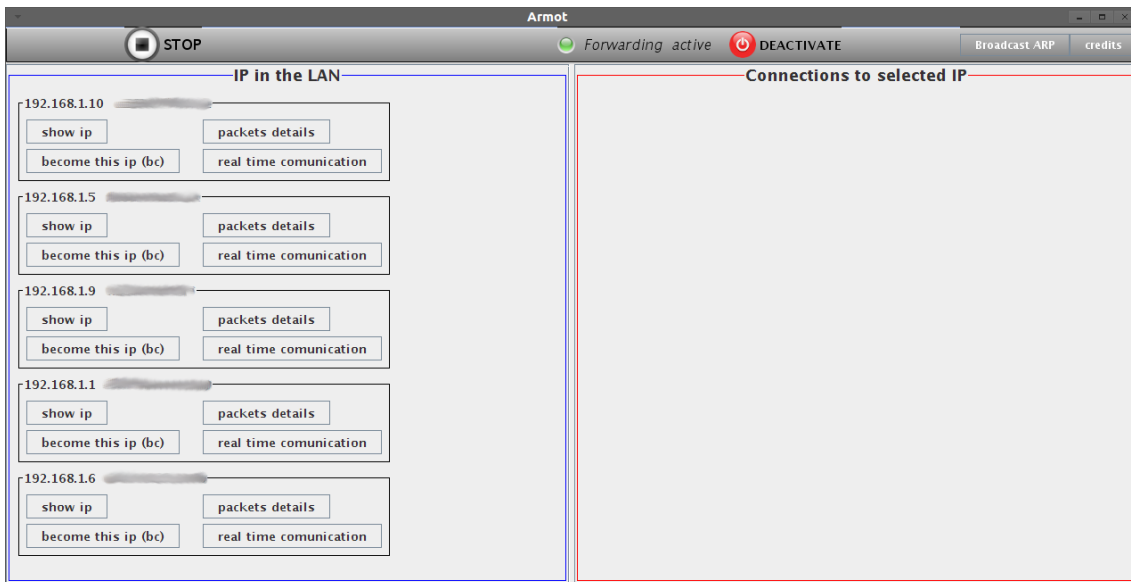


Figura 3.3: Popolamento della lista host

La lista degli IP si popolerà man mano che le tabelle ARP dei vari host scadranno: lo svantaggio di questo metodo è che può essere lento, infatti sono necessari diversi minuti per ottenere una visione chiara delle comunicazioni in corso; il vantaggio è che questo metodo è assolutamente passivo, quindi nessuno, nella rete, può notare la presenza del software in esecuzione.

Per visualizzare le connessioni effettuate da un certo IP è sufficiente usare il bottone “show IP”, ed immediatamente nella parte destra della finestra si potranno vedere gli IP cercati dai pacchetti ARP catturati. In questo modo possiamo capire con chi comunica un certo host.

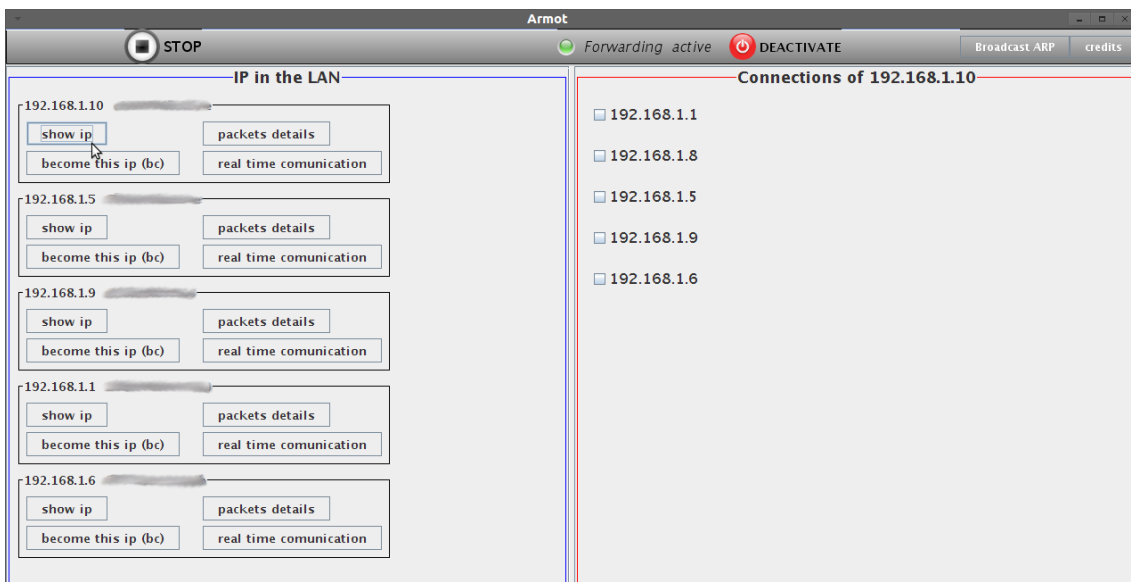


Figura 3.4: Lista di host contattati da 192.168.1.10

3.3 Poisoning

Come si vede dall'immagine precedente (Figura 3.4), accanto ad ogni IP con cui un host sta comunicando è presente un checkbox. Quello che sembra quasi un particolare è in realtà il fulcro del programma. Come è stato detto in precedenza, per riuscire a leggere un flusso di pacchetti è necessario usare un attacco "man in the middle", in modo da ridirezionare il traffico attraverso la propria scheda di rete. Per ottenere il flusso che va da un host A ad un host B è necessario "avvelenare" la tabella ARP di A; in questo modo ogni pacchetto che A spedisce a B conterrà il nostro MAC address, e verrà quindi inviato a noi. Per ottenere ciò si sfrutta l'ottimizzazione del gratuitous ARP, cioè si inviano delle ARP reply fasulle contenenti l'associazione tra l'IP di B ed il nostro MAC address ad A, il quale vedendo questa nuova associazione sovrascriverà quella esatta. Per evitare il timeout della tabella ARP viene inviata una ARP reply ogni due secondi; in questo modo ci assicuriamo di mantenere l'entry fasulla nella tabella di A. All'interno del programma basterà selezionare una delle checkbox presenti per far avvenire tutta questa operazione, e finché sarà selezionata verrà inviato un pacchetto ogni due secondi.

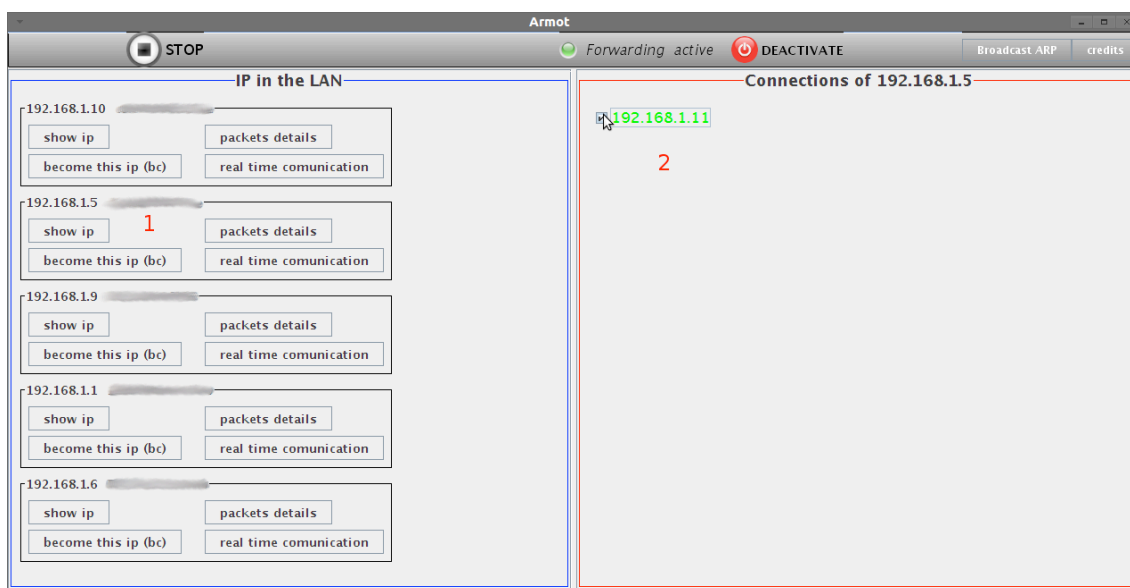


Figura 3.5: Poisoning dell'IP selezionato

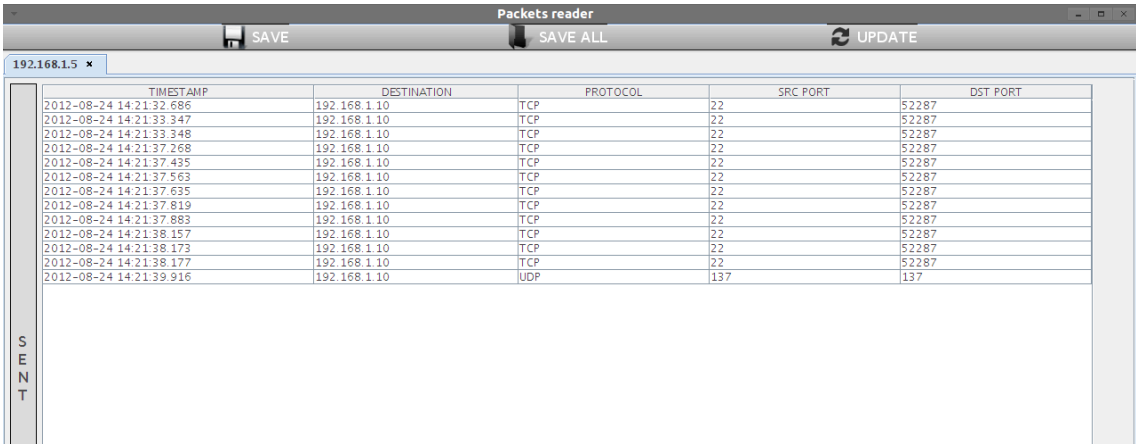
Come si può vedere nella figura 3.5, l'utente che utilizza Armot sta "spacciando" il suo IP per quello selezionato (2), nei confronti dell'host scelto sulla parte sinistra (1). Essendo le ARP reply unicast, solo l'host 1 avrà nella sua tabella l'associazione errata per l'IP 2.

Selezionando un altro IP nella parte sinistra è possibile avviare una ulteriore sessione di poisoning, senza alcun limite di host. Questo può essere utile quando si deve intercettare il traffico verso il router di tutti gli host presenti nella LAN; questi ultimi verranno selezionati sulla sinistra, e per ognuno di loro sarà sufficiente selezionare la checkbox in cui è indicato l'IP del router.

3.4 Funzioni

3.4.1 Packet Reader

Una volta abilitato il poisoner, i pacchetti verranno salvati dal software; per visualizzarli sarà sufficiente cliccare “show packet” nel riquadro dell’IP che ci interessa. Si aprirà una nuova finestra (Figura 3.6):



TIMESTAMP	DESTINATION	PROTOCOL	SRC PORT	DST PORT
2012-08-24 14:21:32.686	192.168.1.10	TCP	22	52287
2012-08-24 14:21:33.347	192.168.1.10	TCP	22	52287
2012-08-24 14:21:33.348	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.268	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.435	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.563	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.635	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.819	192.168.1.10	TCP	22	52287
2012-08-24 14:21:37.883	192.168.1.10	TCP	22	52287
2012-08-24 14:21:38.157	192.168.1.10	TCP	22	52287
2012-08-24 14:21:38.173	192.168.1.10	TCP	22	52287
2012-08-24 14:21:38.177	192.168.1.10	TCP	22	52287
2012-08-24 14:21:39.916	192.168.1.10	UDP	137	137

Figura 3.6: Esempio di cattura di alcuni pacchetti

Come si può vedere dall’immagine, per ogni pacchetto viene mostrato il timestamp, la porta sorgente, la porta di destinazione, l’IP di destinazione, e il tipo.

Nella parte superiore sono presenti i bottoni che permettono di salvare ed esportare i pacchetti salvati.

La differenza tra il bottone “save” e “save all” è che nel primo caso verranno salvati solo i pacchetti dell’IP visualizzato nella finestra, mentre nel secondo caso verranno salvati tutti i pacchetti raccolti da quando è stato premuto il tasto “start”.

Ci sono due possibilità di salvataggio, infatti si può mantenere solo una lista di pacchetti in formato TXT, oppure esportarli in formato pcap, che permette di analizzare il contenuto dei pacchetti tramite Wireshark⁴.

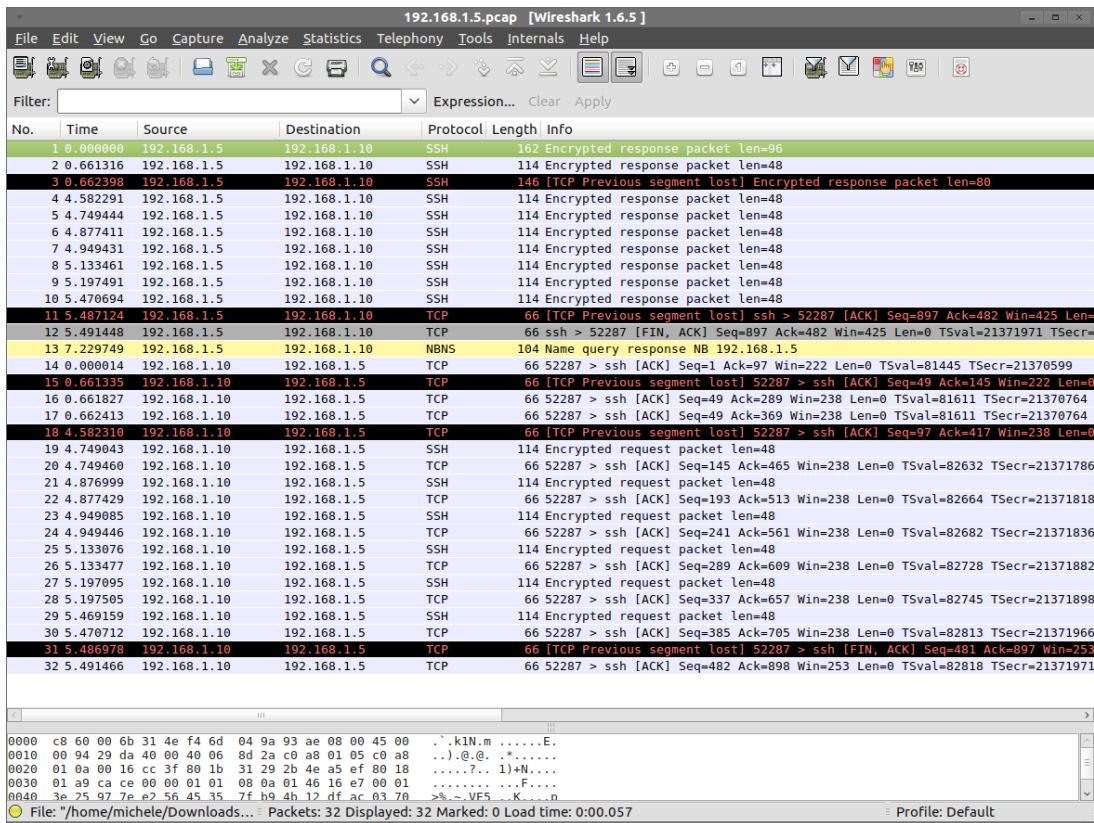


Figura 3.7: Finestra di Wireshark contenente l'esportazione di Armot

3.4.2 Real Time Communication

Oltre alla lettura dei pacchetti, è anche possibile visualizzarne in tempo reale la quantità di quanti ne sono stati inviati e ricevuti da un host. Per farlo è sufficiente utilizzare il bottone “real time communication”, che farà apparire una nuova finestra (Figura 3.8):

Real Time Connections						
192.168.1.1 connections:						
	IP	D port	S port	Protocol	Speed	Counter
SENT	239.255....	1900	49833	UDP	0	40
	239.255....	1900	37287	UDP	0	6
	192.168....	49395	53	UDP	0	1
	192.168....	60753	53	UDP	0	1
	192.168....	38371	53	UDP	0	1
	192.168....	56971	53	UDP	0	1
	192.168....	42579	53	UDP	0	1
	192.168....	34956	53	UDP	0	1
	192.168....	37787	53	UDP	0	1
	192.168....	60990	53	UDP	0	1
RECEIVED	192.168....	49395	53	UDP	0	1
	192.168....	60753	53	UDP	0	1
	192.168....	42579	53	UDP	0	1
	192.168....	37787	53	UDP	0	1
	192.168....	60990	53	UDP	0	1
	192.168....	35029	53	UDP	0	1
	192.168....	41290	53	UDP	0	1
	192.168....	35048	53	UDP	0	2
	192.168....	47104	53	UDP	0	1
	192.168....	35316	53	UDP	0	1

Figura 3.8: Finestra Real Time Connections

Come si può notare dall'immagine, si tratta di una tabella contenente tante righe quante sono le diverse porte utilizzate dall'host selezionato. Per ogni riga è indicato l'host di destinazione, la porta sorgente, la porta destinazione, il protocollo, il contatore di pacchetti e la velocità istantanea di comunicazione.

Tramite questa visualizzazione si può avere un'idea immediata sul traffico generato da un certo client, sui vari client con cui comunica, e sul servizio usato (se la porta è nota). In caso si voglia entrare nello specifico, è necessario tornare nel Packet reader ed esportare i pacchetti salvati in modo da analizzarli con Wireshark.

3.4.3 Broadcast Poisoning

Come abbiamo visto finora, il software ci permette di essere riconosciuti con un certo IP per un determinato host. In caso si voglia ottenere quell'identità in tutta la rete, è possibile inviare un ARP request in broadcast.

Per poterlo fare è necessario selezionare il bottone "become this IP" nel riquadro dell'host a cui siamo interessati, oppure è possibile farlo manualmente tramite il bottone "Broadcast ARP" in alto a destra.

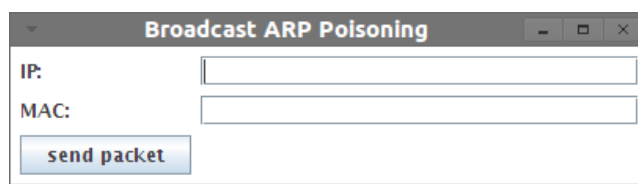


Figura 3.9: Finestra per il Broadcast ARP Poisoning

Il problema di questo metodo è che tutta la LAN vede arrivare il pacchetto, compreso il detentore dell'IP di cui ci vogliamo appropriare. Proprio per questo è probabile che compaiano avvisi di conflittualità di indirizzi, e questo porterebbe a diversi problemi per gli host. Inoltre si rinunciarebbe alla caratteristica di semi-invisibilità che si è mantenuta finora.

Proprio per questi motivi, quando viene selezionato il bottone per il broadcast ARP non viene avviato un thread di invio automatico come negli altri casi, ma ogni pressione del tasto corrisponde ad un solo pacchetto inviato.

Questa funzione è pensata per casi specifici e normalmente viene sconsigliata.

3.4.4 Forward

Quando un host della rete riceve un pacchetto che non è indirizzato a lui può agire in due modi: lo scarta (impostazione predefinita dal sistema operativo) o lo inoltra (solitamente questo è ciò che fa un router)

Quando un pacchetto viene scartato, può essere anche letto e salvato, ma non raggiungerà mai la sua destinazione; viceversa quando viene inoltrato, può essere letto e salvato, e poi inviato nuovamente attraverso la rete per raggiungere la sua destinazione.

Normalmente un host non ha il bisogno di inoltrare pacchetti, dato che il suo utilizzo della rete è quello di inviare o ricevere pacchetti. Il discorso è diverso per un router che divide una rete: dato che i pacchetti arriveranno a lui è necessario che li inoltri al vero destinatario.

Ciò che rende possibile il forward (inoltrare) non è una differenza di hardware tra la scheda di rete di un pc e quella di un router, viene semplicemente attuata una politica diversa nella gestione dei pacchetti.

In un sistema Linux è necessario solamente inserire il valore 1 nel file `/proc/sys/net/ipv4/ip_forward` per abilitare il forward, viceversa 0 per disabilitarlo.

In un software come Armot è molto importante poter far passare i pacchetti attraverso la propria scheda di rete, altrimenti ad ogni attacco “man in the middle” il destinatario dei pacchetti non riceverebbe più nulla, e otterremmo solo il blocco del flusso verso quell’host.

Proprio per questo motivo nella barra superiore dell’interfaccia principale di Armot è presente un indicatore di forward, che avvisa l’utente sullo stato attuale. Accanto è presente un bottone per attivare o disattivare l’inoltrare dei pacchetti.

Come è stato detto precedentemente questo è necessario al funzionamento del software, ma ci offre una funzione in più, infatti dopo aver selezionato un IP ed aver iniziato il poisoning, possiamo decidere di bloccare il flusso di dati che stiamo intercettando, e questo ci torna utile nel caso stia saturando la banda ed impedisca agli altri utenti di utilizzare al meglio la rete.

Inoltre, se intercettiamo il flusso tra un host ed il router connesso al modem, possiamo bloccare la sua connessione ad internet lasciando però la possibilità di utilizzare la rete locale.

Nell’immagine in basso si può notare l’indicatore nei due casi:



Figura 3.10: Indicatore di Forward ON e OFF

Per evitare dimenticanze da parte dell’utente del software, oltre all’indicatore è possibile controllare lo stato dell’inoltrare anche nell’IP sottostante, quando è verde indica il passaggio di pacchetti, viceversa rosso barrato indica il blocco del flusso.

3.5 Interfaccia Testuale

Tutto quello che abbiamo visto finora si può gestire tramite l'interfaccia principale del programma. Questo però causerebbe dei problemi se dovessimo usare un pc non provvisto di interfaccia grafica, come ad esempio un server.

Proprio per questo Arnot può essere avviato in modalità testuale tramite il parametro “-t”.

Per quanto la controparte grafica permetta un uso più semplice e flessibile grazie all'utilizzo del mouse, quella testuale cerca di riproporre le stesse funzioni.

Quando si avvia viene riproposta la scelta dell'interfaccia di rete da utilizzare:

```
Select the number of the nic you wish to use:
0 :eth1
  data link:EN10MB(Ethernet)
  MAC address: :
  address:/192.168.1.10 /255.255.255.0 /192.168.1.255
  address:/ /ffff:ffff:ffff:ffff:0:0:0:0 null
1 :eth2
  data link:EN10MB(Ethernet)
  MAC address: :
2 :any
  data link:LINUX_SLL(Linux cooked)
  MAC address:0:0:0:0:0:0:
3 :lo
  data link:EN10MB(Ethernet)
  MAC address:0:0:0:0:0:0:
  address:/127.0.0.1 /255.0.0.0 null
  address:/0:0:0:0:0:0:1 /ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff null
|
```

Figura 3.11: Scelta testuale dell'interfaccia di rete

Una volta effettuata la scelta ci si troverà nel prompt principale, e tramite il comando “help” si potrà ottenere la lista delle possibili opzioni.

Per mantenere più coerenza possibile tra le due interfacce, anche qui si dovrà usare il comando “start” per la cattura dei pacchetti, e “stop” per l'interruzione.

Ogni comando dato può agire in due modi, effettuare un'azione e tornare al prompt principale (come start e stop), o far entrare l'utente in una sezione specifica. Nel secondo caso il prompt indicherà il nome della sezione scelta, ed in tal caso il comando help si adeguerà alle nuove opzioni disponibili, mentre “back” permetterà di tornare indietro.

3.5.1 Comando “IPs”

Usando il comando IPs si entra in una sezione che permette di effettuare azioni simili all'interfaccia grafica principale, infatti è possibile visualizzare gli host della rete ed usare l'ARP poisoning.

Il comando “show” permette di visualizzare la lista di host conosciuti, mostrando IP e MAC address.

```

->ips
IP Selection (use help for more commands, back to return to the previous menu)
IPs ->show
List of IPs using ARP:
IP: 192.168.1.1 MAC: ██████████
IP: 192.168.1.136 MAC: ██████████
IP: 192.168.1.5 MAC: ██████████
IP: 192.168.1.58 MAC: ██████████
IP: 192.168.1.80 MAC: ██████████
IP: 192.168.1.10 MAC: ██████████
IPs->

```

Figura 3.12: Comando show della sezione IPs

Il comando “scan <IP>” riprende il bottone “show IP” che abbiamo incontrato nell’interfaccia grafica, e come quest’ultimo mostra tutti gli host a cui l’IP specificato nel comando ha inviato una ARP request.

```

IPs->scan 192.168.1.1
192.168.1.1 connections:
- 192.168.1.203
- 192.168.1.10
IPs->

```

Figura 3.13: Comando scan della sezione IPs

Infine il comando “poison” ricalca ciò che veniva effettuato tramite la selezione della checkbox, cioè avvia il thread di invio di finti pacchetti ARP e viene usato con la sintassi “poison <target> <IP>”, che indica al software di inviare ARP reply a <target> dicendo di avere l’IP <IP>.

```

IPs->poison 192.168.1.1 192.168.1.10
Poisoner thread started, gathering packets...
use connections command in the main menu to handle current threads

```

Figura 3.14: Comando poison della sezione IPs

Putroppo nell’interfaccia testuale non c’è la possibilità di utilizzare checkbox, quindi è stato necessario creare un’altra sezione, denominata “connections”, per la visualizzazione e l’arresto dei thread correnti.

3.5.2 Comando “Connections”

Usando il comando “connections” nel prompt principale si entra nell’omonima sezione, come indicato dal cursore.

Come anticipato precedentemente da qui si possono visualizzare e gestire i thread di invio di ARP reply avviati dalla sezione IPs.

Tramite il comando “show”, viene visualizzata una lista con i thread in corso, comprendente anche le informazioni quali l’host avvelenato (target) e la falsa identità con cui veniamo visti da quell’host (IP). Si può vedere dall’immagine sottostante:

```
Connections->show
|- target:192.168.1.5, with identity:192.168.1.10
- target:192.168.1.1, with identity:192.168.1.10
```

Figura 3.15: Comando show della sezione Connections

Con il comando “stop” si può fermare qualunque thread visualizzato precedentemente, tramite la sintassi “stop <target> <IP>”

```
Connections->stop 192.168.1.1 192.168.1.10
Poisoning 192.168.1.1 as 192.168.1.10 ended
```

Figura 3.16: Comando stop della sezione Connections

3.5.3 Comando “Broadcast”

Con il comando “bc” si entra nella sezione che permette di inviare pacchetti ARP broadcast. Come detto precedentemente non è una pratica consigliata, ma può tornare utile per alcuni casi specifici.

Per poterlo utilizzare è necessario impostare due parametri; il primo è l’IP che vogliamo utilizzare, e per farlo è necessario utilizzare il comando “setIP <IP>”.

Secondariamente è possibile impostare il MAC address: questa parte è facoltativa, infatti solitamente si usa il proprio indirizzo, in modo da far convergere i pacchetti al proprio pc, e proprio per questo motivo l’indirizzo di default è quello del computer su cui è in esecuzione il programma; in caso si voglia usare un MAC address diverso è possibile modificare l’indirizzo con il comando “setMAC <mac>”.

Essendo una pratica facilmente rilevabile da altri non viene avviato un thread per l’invio ogni 2 secondi (come nel caso del comando poison di IPs), ma per inviare un pacchetto si deve usare il comando “send”. Quest’ultimo invia un pacchetto ogni volta che viene chiamato, quindi sta a discrezione dell’utente utilizzare il comando in modo appropriato.

3.5.4 Comando “Read”

La controparte della finestra “packet reader” è data dalla sezione “read”, richiamabile dall’omonimo comando.

Una volta entrati, tramite il comando “list” è possibile visualizzare una lista di tutte le connessioni rilevate nella rete.

```

Read->list
|192.168.1.1 connections:
    192.168.1.203
    192.168.1.10
    192.168.1.2
    192.168.1.233
192.168.1.136 connections:
    192.168.1.1
192.168.1.5 connections:
    192.168.1.9
    192.168.1.10
192.168.1.58 connections:
    192.168.1.1
192.168.1.80 connections:
    192.168.1.1
192.168.1.10 connections:
    192.168.1.1
    192.168.1.8
    192.168.1.5
192.168.1.233 connections:
    192.168.1.1
    192.168.1.233
    169.254.255.255
192.168.1.13 connections:
    192.168.1.1
192.168.1.8 connections:
    192.168.1.203

```

Figura 3.17: Comando list della sezione Read

Tramite il comando “show” (la cui sintassi è “show <IP1> <IP2>”) è possibile selezionare una di queste connessioni e visualizzare alcuni dati dei pacchetti che sono transitati tra i due host.

```

Read->show 192.168.1.10 192.168.1.5
|packet sent:
from 192.168.1.10 to 192.168.1.5, P OUT null, P IN null
from 192.168.1.10 to 192.168.1.5, P OUT null, P IN null
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900
from 192.168.1.10 to 192.168.1.5, P OUT 55982, P IN 5900

```

Figura 3.18: Comando show della sezione Read

3.5.5 Comando “Export”

Nella sezione precedente, è possibile visualizzare alcune informazioni sui pacchetti catturati, ma per poterli salvare o analizzare è necessario eseguire il comando “export” per accedere all’omonima sezione.

Ora sarà possibile scegliere tra il comando “txt” e “pcap”.

Utilizzando il comando “txt” è possibile esportare un file di testo contenente una lista di pacchetti contenente per ognuno le informazioni generali (protocollo, destinazione, porte) ma non i dati contenuti, anche perché il formato esadecimale non sarebbe visualizzabile in un file di testo. Una volta lanciato il comando ci verrà chiesto di inserire l’indirizzo IP di cui vogliamo salvare le comunicazioni, oppure scrivere “all” per salvare tutto il contenuto catturato.

In caso volessimo analizzare i pacchetti, dovrà essere usato il comando “pcap”, che a differenza del precedente salva i pacchetti in un formato compatibile con Wireshark, contenente anche i dati presenti negli stessi. Anche in questo caso potremmo scegliere un determinato IP o salvare il tutto.

In entrambi i casi ci verrà chiesto di indicare il percorso di destinazione del file, che dovrà essere indicato come percorso assoluto (ad esempio: /home/utente/documenti/file).

```
Choose export file: txt or pcap
or type back te return to the main menu (or type help)
Export->pcap
Select an IP to save or type all
192.168.1.10
enter the absolute path to the file without extension
(for example: /home/user/Documents/file)
/home/michele/Documents/save
saved!
```

Figura 3.19: Sequenza di comandi per l’esportazione in formato “pcap”

Capitolo 4

Esempio di funzionamento

Per meglio dimostrare le potenzialità e la flessibilità del programma, di seguito verranno esposti due test, uno relativo all'intercettazione di un flusso dati di una connessione VNC, e l'altro riguardante il monitoraggio dell'utilizzo di Internet da parte di un host.

4.1 Cattura flusso VNC

L'obiettivo di questa prova è riuscire ad intercettare un flusso dati di una connessione VNC. Quest'ultimo utilizza il protocollo TCP, e quindi i pacchetti sono visibili solo al mittente e al destinatario.

Si vuole dimostrare che tramite Armot è possibile catturare i pacchetti e anche bloccare il flusso.

4.1.1 Stato della rete

Della rete LAN che verrà utilizzata sono noti tre host:

- dexter (192.168.1.5), pc fisso connesso tramite ethernet, il server VNC
- deede (192.168.1.59), portatile connesso in WiFi, client VNC
- mandark (192.168.1.10), pc fisso connesso tramite ethernet, in cui è stato avviato Armot

Inizialmente, la tabella ARP di deede (192.168.1.5) risulta essere:



Figura 4.1: Tabella ARP di deedee prima del poisoning

4.1.2 Utilizzo di Armot

Una volta avviato Armot su mandark (192.168.1.10), la situazione visibile è:



Figura 4.2: Stato della rete rilevato da Armot

Dato che vogliamo deviare il flusso tra deedee (192.168.1.59) e dexter (192.168.1.5) attraverso mandark (192.168.1.10), dopo aver usato “show ip” nel riquadro di 192.168.1.59 dovremo attivare la checkbox 192.168.1.5 come risulta nella figura che segue:



Figura 4.3: Avvio dell'ARP poisoning sull'host 192.168.1.59

Qualche istante dopo, controllando la tabella ARP di deedee noteremo l'effetto dell'ARP poisoning: accanto all'hostname dexter è presente il MAC address di mandark, quindi la nuova associazione è stata sovrascritta a quella corretta.

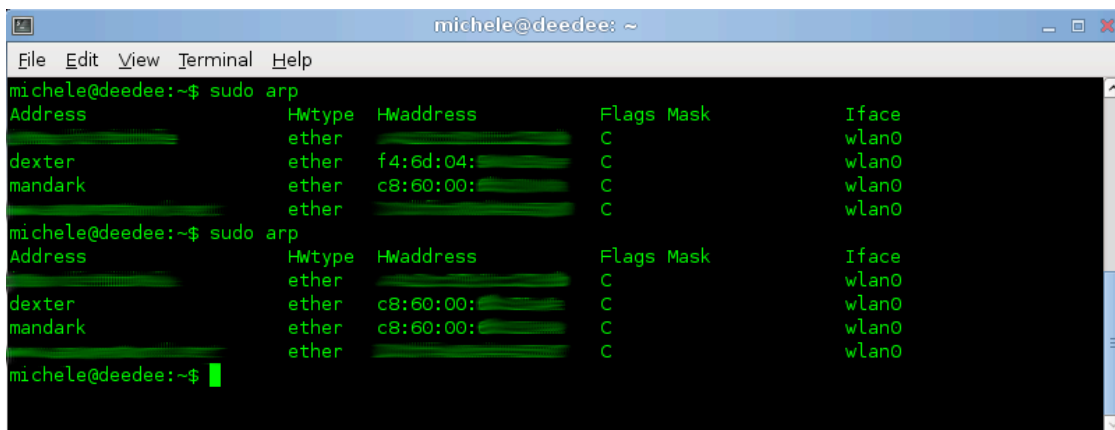


Figura 4.4: Tabella ARP di deedee dopo l'azione del poisoning

4.1.3 Risultati

Da questo momento in poi, in ogni pacchetto da deedee a dexter conterrà il MAC address di mandark, e quindi tutte le connessioni dal livello data link in giù considereranno quest'ultimo come ricevente.

Utilizzando la funzione "real time communication" di Armot sull'indirizzo di deedee (192.168.1.59), si può notare che risulta visibile il traffico di dati, e come si vede

dall'ultima riga della tabella "sent" (Figura 4.5) abbiamo ottenuto ciò che volevamo, cioè il flusso di pacchetti del VNC, riconoscibile dalla porta 5900.

The screenshot shows a window titled "Real Time Connections" with a sub-header "192.168.1.59 connections:". It contains two tables. The top table, labeled "SENT", lists outgoing connections. The bottom table, labeled "RECEIVED", lists incoming connections. Both tables have columns for IP, D port, S port, Protocol, Speed, and Counter.

SENT						
IP	D port	S port	Protocol	Speed	Counter	
224.0.0.251	5555	5555	UDP	0	70	
255.255.25...	17500	17500	UDP	0	70	
192.168.1.2...	17500	17500	UDP	0	70	
192.168.1.10	49176	17500	TCP	0	61	
192.168.1.10	49177	17500	TCP	0	7	
192.168.1.10	49178	17500	TCP	0	7	
192.168.1.10	49179	17500	TCP	0	55	
192.168.1.10	17500	38872	TCP	0	258	
192.168.1.10	17500	38874	TCP	0	127	
192.168.1.10	17500	38876	TCP	0	209	
192.168.1.5	5900	49076	TCP	108	3539	

RECEIVED						
IP	S port	D port	Protocol	Speed	Counter	
192.168.1.10	49177	17500	TCP	0	6	
192.168.1.10	49176	17500	TCP	0	48	
192.168.1.10	49178	17500	TCP	0	6	
192.168.1.10	49179	17500	TCP	0	44	
192.168.1.10	17500	38872	TCP	0	121	
192.168.1.10	17500	38874	TCP	0	54	
192.168.1.10	17500	38876	TCP	0	80	

Figura 4.5: Visualizzazione del flusso VNC attraverso la finestra Real Time Connections

Per esserne sicuri si può anche controllare dalla finestra "packets detail", da cui sono visibili i singoli pacchetti spediti ed inviati. Ci viene confermato che il programma ha catturato i suddetti pacchetti, come visibile in figura 4.6:

The screenshot shows a list of captured packets in a table format. The table has columns for time, IP, protocol, and port. A vertical label "SENT" is on the left side of the table.

Time	IP	Protocol	Port
2012-08-27 09:44:53.96	192.168.1.5	TCP	49076
2012-08-27 09:44:53.963	192.168.1.5	TCP	49076
2012-08-27 09:44:53.978	192.168.1.5	TCP	49076
2012-08-27 09:44:53.991	192.168.1.5	TCP	49076
2012-08-27 09:44:53.992	192.168.1.5	TCP	49076
2012-08-27 09:44:54.004	192.168.1.5	TCP	49076
2012-08-27 09:44:54.006	192.168.1.5	TCP	49076
2012-08-27 09:44:54.019	192.168.1.5	TCP	49076
2012-08-27 09:44:54.02	192.168.1.5	TCP	49076
2012-08-27 09:44:54.031	192.168.1.5	TCP	49076
2012-08-27 09:44:54.032	192.168.1.5	TCP	49076
2012-08-27 09:44:54.034	192.168.1.5	TCP	49076
2012-08-27 09:44:54.036	192.168.1.5	TCP	49076
2012-08-27 09:44:54.037	192.168.1.5	TCP	49076
2012-08-27 09:44:54.051	192.168.1.5	TCP	49076
2012-08-27 09:44:54.052	192.168.1.5	TCP	49076
2012-08-27 09:44:54.053	192.168.1.5	TCP	49076
2012-08-27 09:44:54.056	192.168.1.5	TCP	49076
2012-08-27 09:44:54.066	192.168.1.5	TCP	49076
2012-08-27 09:44:54.067	192.168.1.5	TCP	49076

Figura 4.6: Visualizzazione dei pacchetti tramite il Packet Reader

Se invece vogliamo vedere i dettagli e i dati contenuti nei pacchetti catturati è sempre possibile usare il tasto esporta nella finestra "packets detail" ed ottenere un file .pcap da analizzare con Wireshark (Figura 4.7).

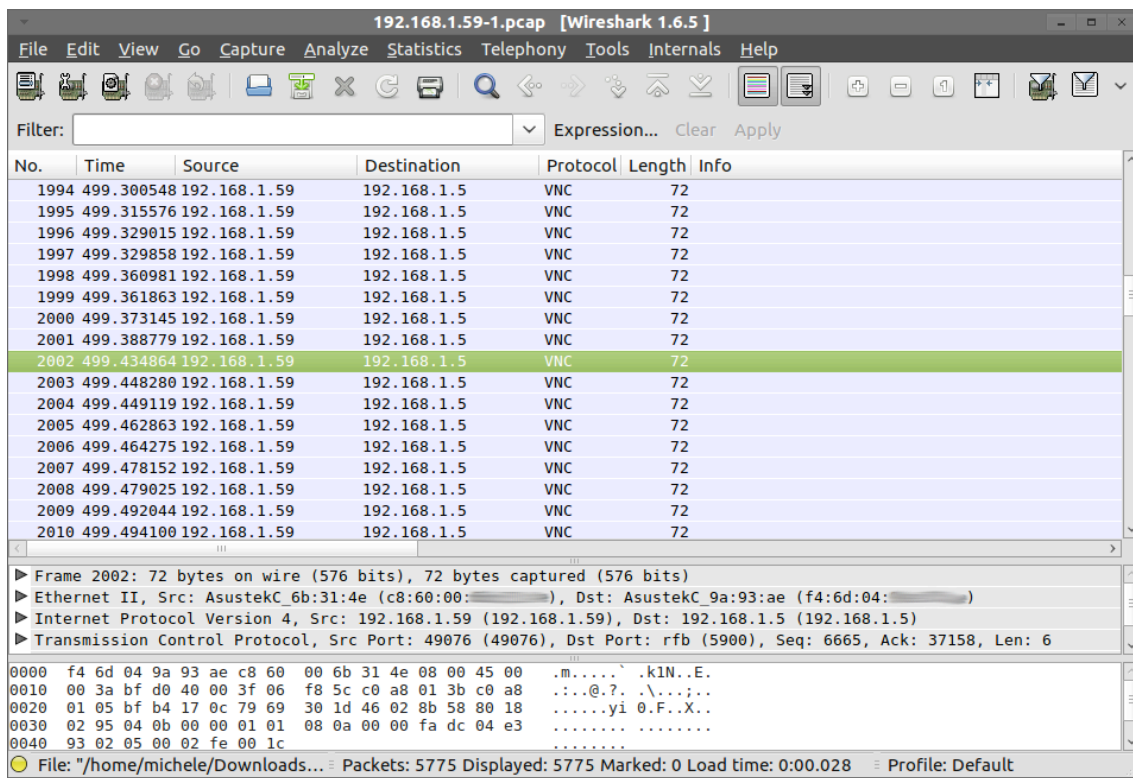


Figura 4.7: Visualizzazione dei pacchetti dopo l'esportazione su Wireshark

4.1.4 Manipolazione

Come è stato spiegato nella descrizione del software, con Armot non è solo possibile deviare un flusso, è possibile anche bloccarlo.

Per farlo sarà sufficiente disattivare il forward con l'apposito bottone situato nella barra superiore. Il programma ci avviserà barrando i flussi condizionati.

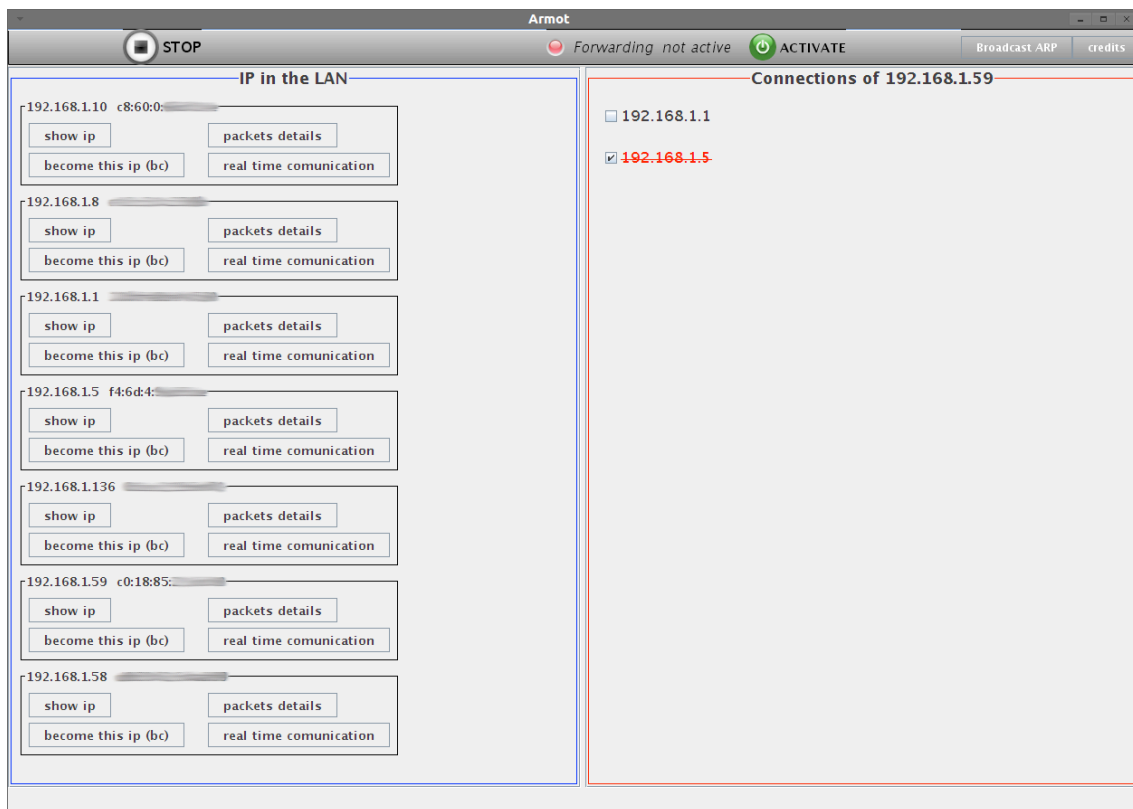


Figura 4.8: Blocco del Forward dei pacchetti provenienti da deedee

Istantaneamente su deedee il VNC viene bloccato, come anche ogni altra comunicazione con dexter. Ogni altro pacchetto indirizzato ad altri host non subirà conseguenze. Come si può vedere dall'immagine sottostante l'effetto è uguale alla disconnessione di dexter dalla rete, ma solo per deedee.

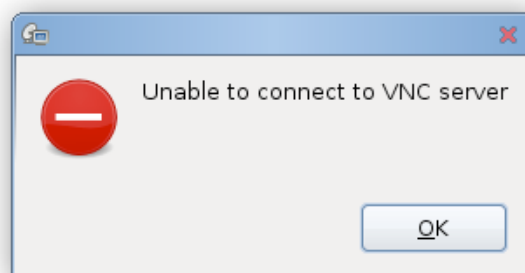


Figura 4.9: Risultato del blocco dei pacchetti

4.2 Cattura flusso internet

Utilizzando la stessa rete di prima, si cercherà di usare Armot per ottenere informazioni sul traffico internet di un host.

4.2.1 Stato della rete

In questa prova gli host interessati saranno:

- deedee (192.168.1.59), su cui un utente sta navigando in internet
- mandark (192.168.1.10), su cui è stato avviato Armot
- router (192.168.1.1), gateway verso l'esterno

4.2.2 Utilizzo di Armot

Ogni pagina web richiesta da un host della LAN passa obbligatoriamente dal router, quindi il flusso di dati da intercettare sarà quello da deedee al router.

Per ottenere questo risultato è necessario visualizzare gli IP contattati da deedee (192.168.1.59) e selezionare la checkbox del router (192.168.1.1), come in figura 4.10:

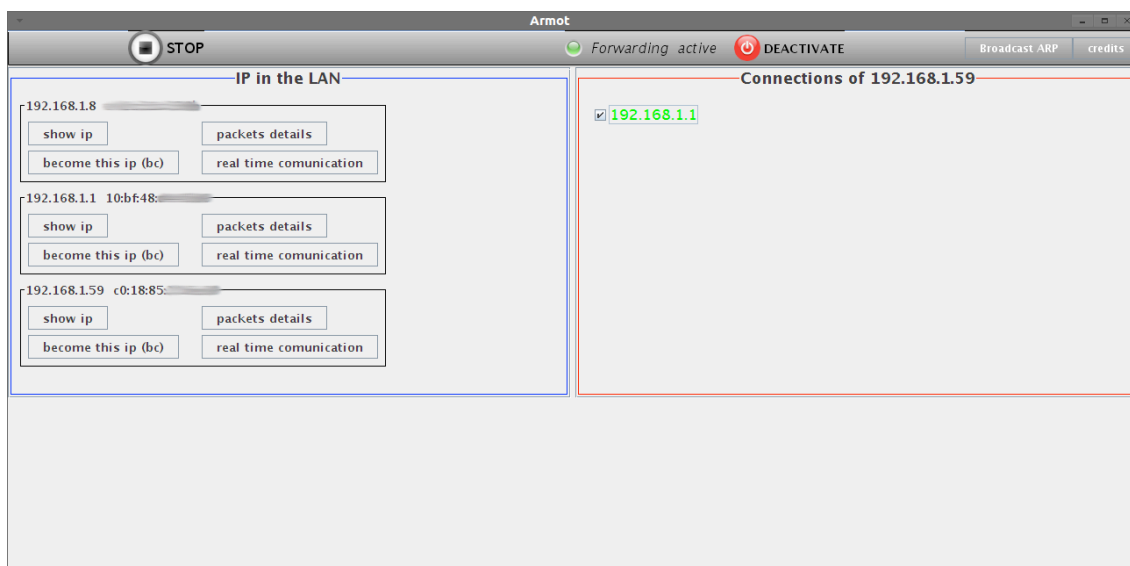


Figura 4.10: Intercettazione del flusso diretto da deedee al router

4.2.3 Risultati

Come precedentemente il flusso è visibile tramite la finestra “real time communication” ed i pacchetti tramite “packets detail”.

In questo caso però, non ci interessa solo avere una lista di pacchetti, sarebbe molto più interessante avere informazioni sui siti visitati, pensiamo ad esempio a scuole o uffici in cui l'accesso ad alcuni siti è normalmente vietato.

Per ottenere questo risultato è necessario esportare i pacchetti raccolti da Armot su Wireshark.

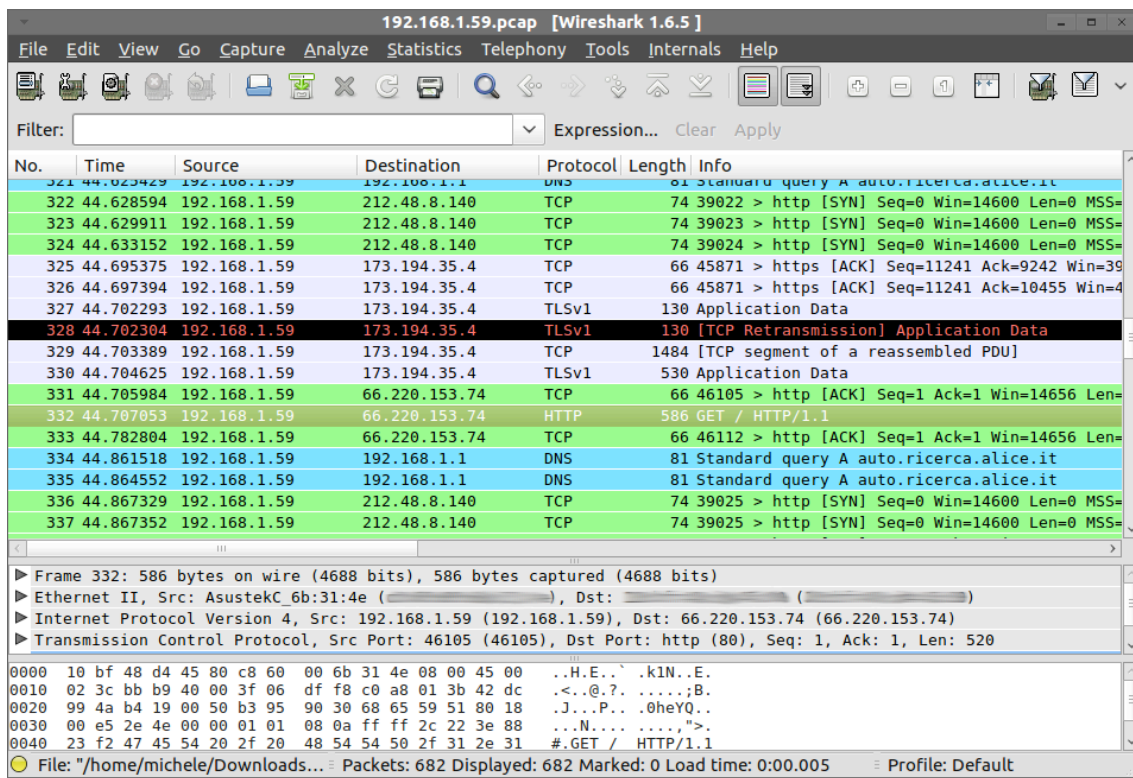


Figura 4.11: Visualizzazione dei pacchetti raccolti tramite Wireshark

Da qui è semplice visualizzare gli IP contattati da deede, cosa impossibile se non avessimo intercettato il flusso di pacchetti tramite ARP poisoning.

Per ottenere più informazioni è sufficiente analizzare il traffico in maniera più dettagliata, che consentirà di scoprire istantaneamente i siti visualizzati, come possiamo vedere nell'immagine sottostante.

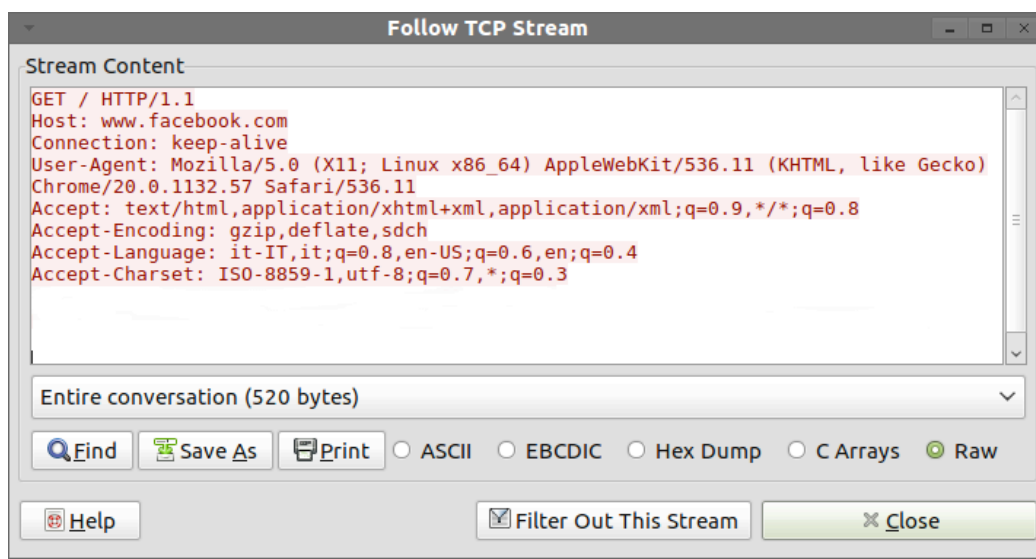


Figura 4.12: Dettagli sul flusso di pacchetti selezionato, il sito visitato è visibile in seconda riga

Come si può leggere nella seconda riga, è chiaramente indicato il sito visitato da deedee, e scorrendo la lista dei pacchetti è possibile rintracciare tutta la cronologia.

4.2.4 Manipolazione

Come nella prova precedente, semplicemente utilizzando il bottone per bloccare il forward è possibile interrompere il traffico tra deedee e il router.

Come conseguenza immediata si può notare l'interruzione di ogni connessione ad Internet:

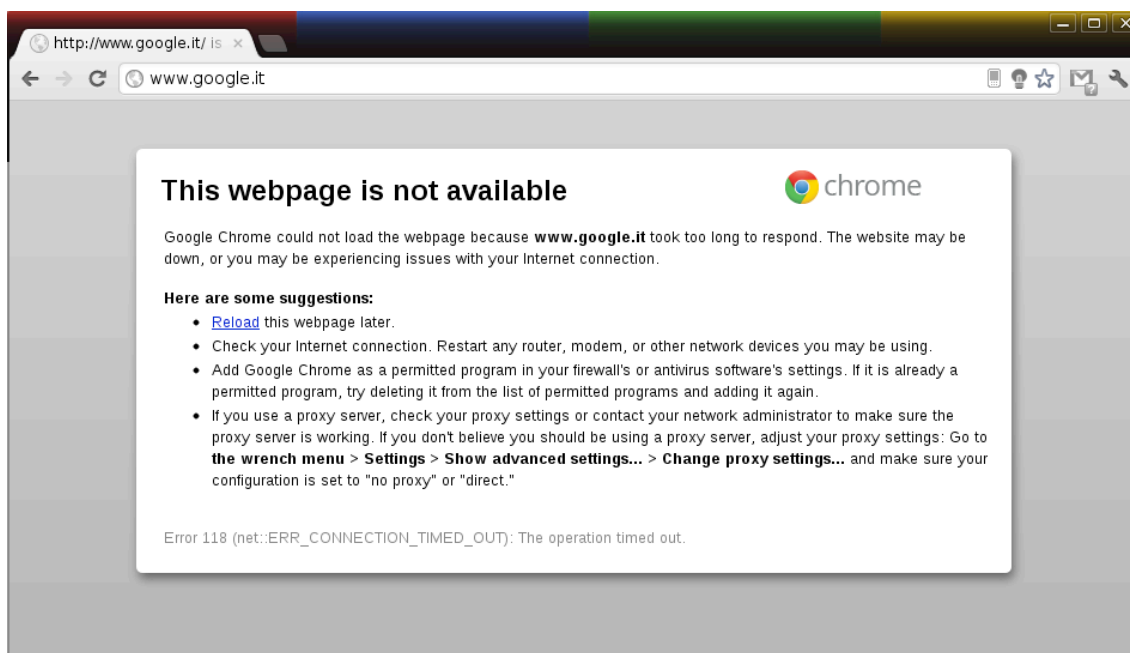


Figura 4.13: Il blocco di Forward di Armot fa credere al PC di non avere una connessione ad Internet, come mostrato da Google Chrome

4.3 Conclusioni dalle prove effettuate

Da quello che abbiamo visto nelle due prove precedenti, il software funziona come descritto, permette di deviare i flussi di pacchetti ed in caso bloccarli.

E' importante notare che in entrambi i casi, nel pc in cui è stato utilizzato Armot non è stata effettuata alcuna modifica alla connessione, non è stato dato alcun potere di amministrazione speciale, tutto ciò che è necessario al funzionamento è semplicemente un computer connesso alla rete LAN.

Inoltre l'azione di questo software non è legata solamente ai pc, ogni host connesso alla rete può essere controllato, dagli smartphone ai router, nessuno escluso.

Capitolo 5

Conclusioni

Lo scopo di questa tesi è stato lo sviluppo di un'applicazione che permettesse ad ogni utente (con background adeguato) di poter monitorare e controllare la propria rete locale. Per renderlo possibile è stato usato l'ARP poisoning, tecnica non ordinaria ma necessaria per ottenere informazioni altrimenti non accessibili. E' stata utilizzata la libreria JPCap per permettere a Java di catturare ed inviare pacchetti tramite l'interfaccia di rete; tutto il resto del software è stato scritto in Java, ed è stata usata la libreria Swing per l'implementazione dell'interfaccia. Quest'ultima è stata resa più minimale possibile, cercando di far risaltare le funzioni principali del programma. Inoltre è stata realizzata una seconda interfaccia, solo testuale, che permette l'uso del software anche in remoto su host non dotati di ambiente grafico.

Per quanto sia stato reso semplice il funzionamento di Arnot, resta sempre un tool destinato ad utenti con le conoscenze necessarie per comprendere le azioni del software, anche perché, come detto precedentemente, il programma lavora al di fuori del normale funzionamento della rete.

Un aspetto importante che deriva dallo sviluppo di questo software, è quello di evidenziare la poca sicurezza offerta dalla rete. Infatti, anche con un programma come Arnot, nato come strumento per aiutare nella gestione della propria rete, sarebbe piuttosto facile raccogliere informazioni sulle attività degli altri utenti se, ad esempio, il suddetto programma fosse avviato in un Internet Point pubblico.

Purtroppo la tecnica usata da Arnot, l'ARP poisoning, è molto difficile da combattere, infatti sfrutta un problema insito nel protocollo ARP. Come si può facilmente immaginare, sarebbe improponibile pensare di modificare il protocollo a tutti i dispositivi che ne fanno uso in tutto il mondo.

In rete è possibile trovare dei software che permettono di difendere il proprio PC da questo attacco, ma il problema principale è che la maggior parte degli utenti non è a conoscenza del pericolo.

Bibliografia

- [1] Tanenbaum A., Wetherall D.: Reti Di Calcolatori, Pearson, (2011)
- [2] Libreria open-source JPCap, download al sito <http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/download.html>
- [3] Eclipse IDE, download al sito <http://www.eclipse.org/downloads/>
- [4] Wireshark, download al sito <http://www.wireshark.org/download.html>

Elenco delle figure

- 2.1 Esempio di rete locale
- 2.2 Intercettazione del flusso tra l'host 1 e 2 da parte del terzo
- 3.1 Scelta dell'interfaccia di rete
- 3.2 Finestra principale
- 3.3 Popolamento della lista host
- 3.4 Lista di host contattati da 192.168.1.10
- 3.5 Poisoning dell'IP selezionato
- 3.6 Esempio di cattura di alcuni pacchetti
- 3.7 Finestra di Wireshark contenente l'esportazione di Armot
- 3.8 Finestra Real Time Connections
- 3.9 Finestra per il Broadcast ARP Poisoning
- 3.10 Indicatore di Forward ON e OFF
- 3.11 Scelta testuale dell'interfaccia di rete
- 3.12 Comando show della sezione IPs
- 3.13 Comando scan della sezione IPs
- 3.14 Comando poison della sezione IPs
- 3.15 Comando show della sezione Connections
- 3.16 Comando stop della sezione Connections
- 3.17 Comando list della sezione Read
- 3.18 Comando show della sezione Read
- 3.19 Sequenza di comandi per l'esportazione in formato "pcap"
- 4.1 Tabella ARP di deedee prima del poisoning
- 4.2 Stato della rete rilevato da Armot
- 4.3 Avvio dell'ARP poisoning sull'host 192.168.1.59
- 4.4 Tabella ARP di deedee dopo l'azione del poisoning
- 4.5 Visualizzazione del flusso VNC attraverso la finestra Real Time Connections
- 4.6 Visualizzazione dei traccetti tramite il Packet Reader
- 4.7 Visualizzazione dei pacchetti dopo l'esportazione su Wireshark
- 4.8 Blocco del Forward dei pacchetti provenienti da deedee
- 4.9 Risultato del blocco dei pacchetti
- 4.10 Intercettazione del flusso diretto da deedee al router
- 4.11 Visualizzazione dei pacchetti raccolti tramite Wireshark
- 4.12 Dettagli sul flusso di pacchetti
- 4.13 Risultato del blocco dei pacchetti