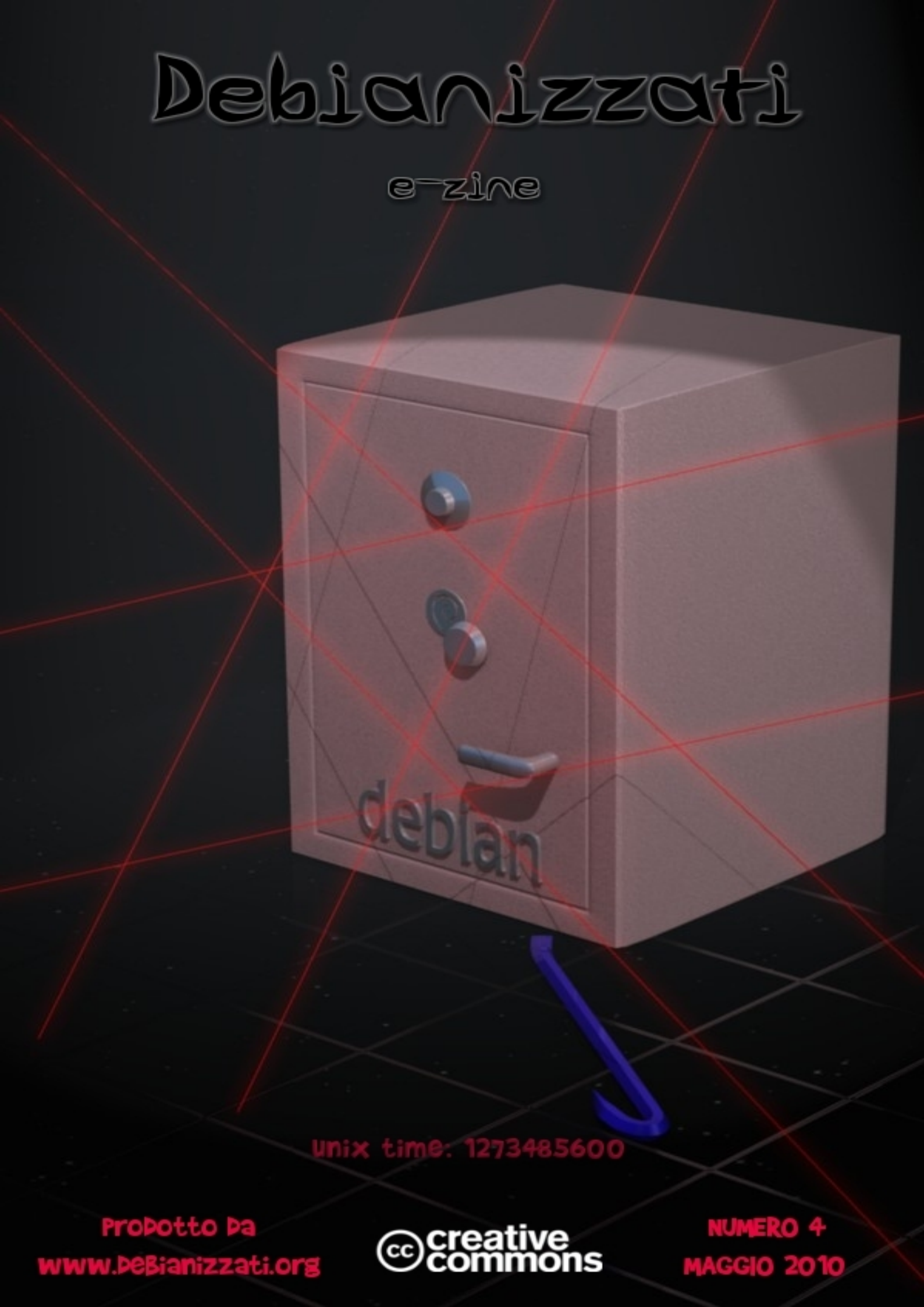
Debianizzati

e-zine



unix time: 1273485600

Prodotto da
www.debianizzati.org

Numero 4
Maggio 2010

Se l'ultimo numero giunse con l'arrivo dell'inverno, il numero 4 di Debianizzati vi saluta con la primavera (che stiamo diventando una rivista stagionale ;-) ?).

Novità? Molte, anche se non tutte sono palpabili allo stesso modo. Innanzi tutto è con una certa soddisfazione che posso annunciare l'allargamento ulteriore del team, passando da 7 a 12 membri. Non si possono però in ogni caso dimenticare vari spunti da diversi utenti, senza l'aiuto dei quali non sarebbe stato possibile pubblicare per intero quest'ultimo numero. Per quanto riguarda l'organizzazione del lavoro abbiamo cercato di applicare un diagramma di flusso strutturato, in particolare per quanto riguarda il lavoro di revisione. Il concetto di qualità al posto di quantità è stato ulteriormente fissato e anche se ciò è costato una qualche settimana di ritardo sui 3 mesi ai quali vi avevamo abituati, sono certo che non vi deluderemo per quanto riguarda i contenuti. E se la struttura principale del portale e del progetto e-zine vi sta annoiando, non dimenticatevi di seguire da vicino i movimenti di casa debianizzati: a partire dall'inizio di maggio è previsto il passaggio del portale da smf & Co. ad una nuova piattaforma! L'e-zine cambierà dunque come tutto il resto del portale, il suo abito da sera :-).

Continuando sullo slancio del numero 2, quando abbiamo incominciato a caratterizzare i numeri con un tema di base, dopo l'anatomia del sistema e il port, abbiamo deciso di orientare il numero 4 alla sicurezza. Un tema sicuramente importante ed avvincente che non evita di tralasciare dubbi, fra meraviglie e nubi nere all'orizzonte...

Buona lettura!

Brunitika

Indice

Editoriale

Occhi bendati e mani legate

Negli ultimi anni la diffusione sempre più capillare degli accessi al web, della banda larga e dei circuiti peer to peer ha spinto i grossi nomi dell'industria dell'intrattenimento (majors discografiche, cinematografiche, etc.) a tentare di rafforzare i propri diritti in materia di proprietà intellettuale per difendere i propri modelli di business all'interno di un mercato, come quello elettronico, in fortissima espansione.
La compiacenza della politica (sempre pronta ad intervenire a favore di chi è espressione di un potere forte come quello economico e mediatico piuttosto che in favore della cosa pubblica), ha fatto sì che si venisse a concretizzare uno scenario fatto di alleanze, accordi - per lo più segreti - e restrizioni in cui ingabbiare chi fruisce della rete come strumento di comunicazione, condivisione e di espressione della propria libertà ed autodeterminazione.
Nell'Ottobre del 2007 Comunità Europea, Stati Uniti, Giappone e Svizzera annunciarono al mondo che avrebbero negoziato un nuovo trattato contro la contraffazione, l'ACTA (Anti-Counterfeiting Trade Agreement).
Altre otto nazioni ben presto si unirono al gruppo: Australia, Canada, Emirati Arabi, Giordania, Marocco, Messico, Nuova Zelanda e Repubblica di Corea.
Anche se il titolo dell'accordo proposto avrebbe potuto suggerire che le trattative avrebbero riguardato i beni materiali (ad esempio medicine o alimenti), è risultato subito chiaro, sebbene si mantenessero segreti i temi discussi, che il vero obiettivo sarebbe stato il controllo di Internet e dell'Information Technology.
Nell'agosto del 2008 il GIP di Bergamo dispose il filtro dns sugli ip di thepiratebay.org, il popolare sito svedese di indicizzazione di file torrent.
Curiosamente, da quel momento in poi, chiunque tentava di accedere al sito veniva indirizzato ad una pagina appartenente a www.pro-music.org, sito di proprietà della

International Federation of the Phonographic Industry (IFPI), la quale, di fatto, ebbe la possibilità (fin quando il re-indirizzamento non fu modificato) di mettere in atto operazioni di tracking e di logging nei confronti dei visitatori.

Dopo un tira e molla durato quasi due anni, il Nucleo polizia tributaria della Guardia di Finanza di Bergamo, a partire dal mese di febbraio del 2010, ha iniziato a notificare agli Internet Provider italiani il provvedimento che li obbliga a rendere inaccessibile il sito dal territorio nazionale.

Un paio di mesi dopo, nell'aprile del 2010, a seguito delle richieste pressanti di quella parte della comunità digitale che vuole con forza difendere la libertà e l'indipendenza della rete, la bozza preliminare del testo dell'ACTA viene rilasciata al pubblico. Ciò che si era sempre temuto (cioè che i negoziati non riguardassero propriamente i beni fisici) trova conferma nella lettura del testo: l'ACTA riguarda il copyright, i brevetti software e la proprietà intellettuale in senso lato.

Riguarda Internet e i vincoli che si vogliono imporre agli utenti per limitare - di fatto - la comunicazione, la collaborazione e la condivisione attraverso una serie di obblighi gravanti sui provider, formalmente obbligati a presidiare il web (per mezzo dell'esercizio di tutta una serie di controlli sulle abitudini di navigazione degli utenti) e a porre in essere le limitazioni necessarie a parare **preventivamente** qualsiasi tipo di potenziale violazione della proprietà intellettuale.

Sono questi solo alcuni degli avvenimenti degli ultimi anni che danno il senso di come Internet, da strumento di comunicazione snobbato dai più, sia diventato un territorio ambito, un luogo da controllare rigidamente e da sfruttare per i propri tornaconti in barba alle libertà personali e ai diritti di privacy e di riservatezza.

Eric Schmidt, CEO di Google, durante un'intervista rilasciata alla CNBC nel dicembre 2009, affermò testualmente: If you have something that you don't want anyone to know, maybe you shouldn't be doing it in the first place che letteralmente si potrebbe tradurre: Se hai qualcosa che non vuoi far sapere a nessuno, probabilmente la prima cosa che dovresti fare è non farla ovvero: Se non hai niente da nascondere, non hai niente di cui preoccuparti. Insomma, il messaggio appare chiaro: Se non vuoi problemi, rinuncia alla tua privacy.

Il che ha un non troppo vago sapore di minaccia.

Se nei primi anni di vita (e per un buon periodo a seguire) il web è stato un laboratorio di sperimentazione per il mondo dell'information technology e un canale di comunicazio-

ne svincolato dalle distanze geografiche e dalle barriere politiche, oggi esso è diventato un vero e proprio mercato ed il suo controllo fa gola a tanti.

Chi detiene potere (politico o economico) sta tentando con qualsiasi mezzo di esercitare una stretta attività di controllo sulle informazioni che in grande quantità transitano per le maglie della rete globale ogni giorno.

Si tratta di mettere le mani su una base dati potenzialmente infinita di nomi, abitudini, gusti, contatti di tutti coloro i quali usano la rete quotidianamente.

È vero che la natura anarchica di Internet ha facilitato, soprattutto negli ultimi anni, l'esplosione di veri e propri fenomeni criminosi (pedopornografia, truffe online, etc.) ma è anche vero che si vuol far credere che la partita che si sta giocando sia *sicurezza contro privacy* quando invece essa riguarda *controllo contro libertà*.

La sicurezza di ognuno di noi è direttamente proporzionale al grado di libertà che siamo in grado di esercitare.

Più siamo liberi più siamo sicuri.

D'altronde, è proprio la privacy che ci cautela dagli abusi e rimanere lontani dagli abusi significa essere sicuri, significa poter vivere in libertà.

La sicurezza non deve prevedere intrusione.

Il CEO di Google, insomma, ha torto: la privacy è un valore e un diritto e questo diritto deve essere reclamato anche e soprattutto quando non si ha niente da nascondere.

Nel nostro Paese viviamo in una situazione che sta fra l'avanspettacolo di infimo ordine e la tragedia greca.

Abbiamo una magistratura che condanna in base alla possibilità che dei reati vengano commessi (tra l'altro) fuori dalla propria giurisdizione (si veda la vicenda The Pirate Bay per l'appunto), una classe politica incapace di legiferare al passo con i tempi e troppo coinvolta negli interessi economici in gioco, dei mezzi di informazione (radio, giornali, tv) che mantengono un ossequioso silenzio e, infine, una società civile che sembra disconoscere il problema e che, se lo riconosce, quanto meno lo sottovaluta.

Siamo disorganizzati, poco consapevoli, poco curiosi e per niente desiderosi di capire e di impegnarci a difendere i nostri diritti.

Insomma, si ha l'impressione di un pericoloso disinteresse generale.

Il 4 Maggio 2010 dai gruppi per la giustizia sociale e i diritti online è stata indetta la giornata internazionale contro i Digital Restrictions Management (DRM) ovvero contro i sistemi di gestione delle restrizioni digitali.

L'obiettivo è quello di far crescere la consapevolezza dell'opinione pubblica relativamente ai pericoli derivanti dall'utilizzo di tecnologie atte a limitare l'accesso ai dati digitali, relativamente all'ennesima forma di controllo che si cerca di imporre contro la libera diffusione del sapere.
Speriamo non venga sprecata l'ennesima occasione di cercare di strapparci la benda dagli occhi, di cercare di spezzare quella corda che ci immobilizza le mani.

pmate

Capitolo 1

La Pagina dei lettori

Dopo l'ultimo articolo sull'emacs abbiamo ispirato samiel ad approfondire le varie facce di questo - chiamiamolo - programma. La parte curata in dettaglio è rivolta a tutto ciò che si potrebbe definire come email; in poche parole, Emacs E-mail Essential.

1.1 Emacs E-mail Essential: parte prima

In ogni guida a Emacs è canonico l'*incipit* in cui si sottolineano le molteplici funzioni e potenzialità di questo programma, che lo fanno essere ben più di un semplice editor. In questa sede cercheremo di chiarire alcune (ma solo alcune) delle svariate modalità in cui è possibile gestire direttamente con Emacs la posta elettronica - mentre, com'è noto, esso può essere scelto come editor per un MUA (acronimo di Mail User Agent) come Mutt. In una parte successiva di questa guida prenderemo in considerazione altri programmi che possono affiancare Emacs sempre per gestire la posta (come VM, Wanderlust e Mew).
Per l'invio e la ricezione della posta Emacs si affida a dei programmi che agiscono in sottofondo, ma, a differenza di Vim, l'altro grande editor, riesce a gestire dal suo interno l'intero processo.
Faremo qui riferimento all'ultima versione attualmente disponibile di Emacs, la 23.1, che presenta significative modifiche rispetto alla versione precedente anche proprio in relazione alla gestione della posta elettronica. Aggiungiamo che questa guida è stata approntata su Debian Testing (Squeeze) e quindi verificata su Debian Unstable (Sid). Non vale invece per l'attuale versione stabile di Debian (ossia Lenny), che presenta Emacs 22.2.

Seguiremo la consuetudine tipografica per cui i comandi sono preceduti dal tasto C (= Control) e le funzioni da M (= il tasto Meta), che nelle nostre tastiere si ottiene premendo Alt.

1.1.1 Comporre posta

Comporre una mail

Presupponiamo al momento di affidarci alla modalità standard di Emacs per l'invio della posta elettronica, il cosiddetto MailMode. Per comporre un messaggio è necessario in primo luogo inizializzare il buffer della mail, il che si ottiene con C-x m. Due possibili varianti: C-x4m apre il buffer della mail dividendo la finestra e lasciando visibile il buffer da cui si è impartito il comando, mentre C-x5m apre il buffer della mail in un'altra finestra. In alternativa, è possibile digitare M-x mail. Nel buffer troveremo tre righe predefinite:

```
To:
Subject:
--text follows this line--
```

e sarà possibile iniziare a digitare subito il testo. Nel campo To: possono essere inclusi più indirizzi, separati da virgole senza spazi.
L'invio si effettua con C-c C-s che invia il messaggio restando nel buffer della mail, oppure con C-c C-c che invia il messaggio e chiude il buffer.
Altri comandi disponibili nel MailMode, in aggiunta a quelli già disponibili nel TextMode, sono:
C-c C-t: va all'inizio del testo del messaggio;
C-c C-w: inserisce alla fine del messaggio una firma traendola dal file apposito (a meno che non si sia attivata la firma automatica di tutti i messaggi. Vedi sotto);
C-c C-i ''file'' [Invio]: inserisce il contenuto di un file alla fine del messaggio;
M-x ispell-message: effettua la correzione ortografica del testo del messaggio (ma non del testo citato da altri messaggi);
M-x goto-address: richiama gli indirizzi di mail. Cliccando su un indirizzo si apre il buffer per un messaggio;
C-h m: ottiene l'intera gamma dei comandi.

Personalizzare gli header

Destinatario e soggetto della mail sono header predefiniti che il mittente compila manualmente. Altri header, come il mittente e la data, vengono creati automaticamente. Altri header ancora devono essere scritti a mano, prima del delimitatore costituto dalla riga:

```
--text follows this line--
```

Se non precisato altrimenti, il campo `From:` ripropone il valore del campo `user-mail-address`.
È possibile controllare il formato dello header `From:` con le seguenti variabili:
`nil`: usa il semplice indirizzo mail. Risulterà così ad es.:
`From: king@grassland.com;`
`parens`: usa dapprima l'indirizzo mail e quindi il nome completo ponendolo fra parentesi tonde. Risulterà così ad es.:
`From: king@grassland.com (Elvis Parsley);`
`angles`: usa dapprima il nome completo e quindi l'indirizzo mail ponendolo fra parentesi uncinate. Risulterà così ad es.:

```
From: Elvis Parsley <king@grassland.com>;
```

`system-default`: consente al sistema di inserire un campo `From:`.
Si avrà così ad esempio:

```
(setq user-full-name "Mario Rossi"
      user-mail-address "mario.rossi@gmail.com"
      mail-from-style 'angles)
```

Gli header aggiuntivi possono essere creati con una serie di comandi ulteriori (disponibili anche nel menu `Header` nella versione di Emacs per X):
`C-c C-f C-t`: va al campo `To:` (destinatario), creandolo se non esiste;
`C-c C-f C-s`: va al campo `Subject:` (argomento), creandolo se non esiste;
`C-c C-c C-c`: va al campo `Cc:` (copia per conoscenza), creandolo se non esiste;
`C-c C-f C-b`: va al campo `Bcc:` (copia cieca per conoscenza), creandolo se non esiste;

`C-s C-f C-f`: va al campo `Fcc:` (si tratta di un file a cui Emacs aggiunge il testo della mail, scrivendolo all'atto dell'invio. Se si desidera utilizzare questo campo per salvare le mail, si può precisare un campo `Fcc:` fisso mediante la variabile `mail-archive-file-name`), creandolo se non esiste.
Un altro dei campi utilizzabili è:
`Reply-to:`: indirizzo di replica che in genere soppianta il campo `From:`. Per impostare un campo `Reply-to:` fisso, si usa in `/.emacs` la variabile `mail-default-reply-to`.
L'inserimento automatico degli header desiderati si può ottenere grazie al seguente codice:

```
(setq mail-default-headers "Cc: \nBcc: \nFCC: \n")
```

Come abbiamo accennato sopra, non è possibile soltanto aggiungere di volta in volta una firma alla mail traendola dal file `/.signature`. Si può anche aggiungere automaticamente una firma a tutti i messaggi impostando la variabile `mail-signature` a `t`; se la si vorrà omettere da un particolare messaggio, bisognerà cancellarla manualmente. La firma può svilupparsi in più righe. È sufficiente separare le righe con l'operatore *n*. Si avrà pertanto un codice come il seguente:

```
(setq mail-signature t)
(setq mail-signature "Mario Rossi \nVia dei Sicomori, 23 \n00100 Roma
\ntel.: 066 6666666 \ncell.: 666 6666666 \nmail: mario.rossi@gmail.com")
```

Creare degli alias

È possibile creare una sorta di rubrica degli indirizzi tramite degli alias (esattamente come in Mutt), collocandola in un file denominato `/.mailrc`. Emacs espande questi alias, che possono comprendere semplici indirizzi o gruppi di indirizzi, quando si presentano nei campi
`To:`, `From:`, `Cc:`, `Bcc:` e `Reply-to:`.
La sintassi del file `/.mailrc` è la seguente:
`alias indirizzo_breve indirizzo_completo`
Indirizzi multipli devono essere separati da uno spazio. Se un indirizzo contiene spazi, esso dev'essere racchiuso fra virgolette doppie.

Inoltre, Emacs riconosce i comandi inclusi in `/.mailrc` (e ignora qualsiasi altra stringa). Ad esempio si potrebbe avere:
`source ''filename''`
Infatti, il file `/.mailrc` è usato innanzitutto da altri programmi di posta elettronica.
È tuttavia possibile definire gli alias anche all'interno di Emacs, ponendoli in `/.emacs`, mediante il comando `define-mail-alias`. Il programma richiede l'alias e quindi l'indirizzo completo. La sintassi in `/.emacs` è la seguente:
`(define-mail-alias ''alias indirizzo'')`
Invece del file di default, per l'appunto `/.mailrc`, si può indicare un altro file impostando opportunamente la variabile `mail-personal-alias-file`.
Di default, Emacs espande gli alias quando il messaggio viene spedito. Se però si preferisce che li espanda al momento della stesura del messaggio stesso (ad esempio per esigenze di controllo), si deve dare il comando: `C-c C-a` oppure `M-x expand-mail-aliases`.

1.1.2 Inviare posta

nvio col MailMode

Per l'invio della posta, presupporremo qui di utilizzare un server SMTP e ci serviremo di una funzione di MTA (ossia Mail Transfer Agent) integrata in Emacs, e precisamente di `smtpmail`. Si tratta della soluzione più semplice e funzionale.
È necessario inserire nel file di configurazione di Emacs, e cioè `/.emacs`, delle stringhe che precisano alcuni dati sugli header del messaggio e quindi le modalità di invio.
Tali righe di configurazione sono poche se abbiamo a che fare con un normale server SMTP, mentre si richiede qualche stringa in più se il server richiede autenticazione, com'è nel caso della diffusissima Gmail.

Server che non richiede autenticazione

Se il server non richiede autenticazione, le righe minimali o quasi da inserire sono le seguenti:

```
(setq user-full-name "Nome Cognome"
      user-mail-address "mittente@dominio"
      send-mail-function 'smtpmail-send-it
      smtpmail-smtp-server "server_smtp")
```

La variabile `smtpmail-smtp-server` controlla lo hostname del server, e può contenere uno hostname o un indirizzo IP. Se vuota, prende i dati dalla eventuale variabile `smtpmail-default-smtp-server`. Quest'ultima controlla lo hostname di default da usare; in genere si usa per impostare un file di inizializzazione per l'intero sistema, ha meno senso in un desktop con una sola connessione di questo tipo. Essa dev'essere posta prima che la libreria SMTP sia caricata. Non ha alcun effetto se si è già definita la variabile `smtpmail-smtp-server`. Usualmente, il servizio SMTP usa la porta TCP 25; alcuni servizi adottano invece la porta 587 o altre ancora. In questi casi, la porta viene definita attraverso la variabile `smtpmail-smtp-service`.
Una versione un po' più articolata è la seguente, che indica anche - nel caso servisse - la porta stessa:

```
(setq user-full-name "Nome Cognome"
      user-mail-address "mittente@dominio"
      send-mail-function 'smtpmail-send-it
      message-send-mail-function 'smtpmail-send-it
      smtpmail-default-smtp-server "server_smtp"
      smtpmail-smtp-server "server_smtp"
      smtpmail-smtp-service 587)
(require 'smtpmail)
```

In realtà il campo `(setq user-full-name ‘‘)` non è strettamente necessario, ma se non viene inserito il mittente della mail risulterà semplicemente l'indirizzo di posta elettronica. Se si vuole specificare precisamente il mittente, allora è necessario inserire anche questo campo.

Server che richiede autenticazione

Alcuni server richiedono autenticazione, via SSL oppure via TLS. Viene allora richiesto un pacchetto aggiuntivo, `starttls`, da installare separatamente. In questi casi le stringhe essenziali da inserire in `/.emacs` saranno (esemplifichiamo qui con i campi di Gmail):

```
(setq user-full-name "Nome Cognome"
      user-mail-address "mittente@gmail.com"
   send-mail-function 'smtpmail-send-it
   message-send-mail-function 'smtpmail-send-it
   smtpmail-starttls-credentials '(("smtp.gmail.com" 587 nil nil))
   smtpmail-auth-credentials (expand-file-name "~/.authinfo")
   smtpmail-default-smtp-server "smtp.gmail.com"
   smtpmail-smtp-server "smtp.gmail.com"
   smtpmail-smtp-service 587)
(require 'smtpmail)
(starttls-use-gnutls nil)
```

I dati per l'autenticazione possono essere inseriti direttamente in `/.emacs`, ma trattandosi di un file in lettura, ciò potrebbe compromettere la sicurezza. Per ovviare a tale inconveniente, i dati sensibili andranno inseriti in un altro file, `/.authinfo`, con la sintassi seguente:

```
machine smtp.gmail.com login mittente@gmail.com password SECRET port 587
```

Il file andra poi protetto chmodandolo a 600, in modo che soltanto l'utente cui appartiene possa accedervi.

Impostazioni aggiuntive

In `/.emacs` possono essere definite innumerevoli impostazioni aggiuntive, a seconda delle necessità individuali. Ne menzioniamo qui alcune fra le più semplici e meno specifiche.

```
(add-hook 'message-mode-hook 'toggle-input-method)
```

usa il metodo di immissione di default nella composizione dei messaggi, che si attiva automaticamente quando si apre il buffer per una mail;
`(smtpmail-debug-info t)`
riporta estesamente eventuali problemi di trasmissione con il server SMTP. Può essere utile almeno per le prime connessioni, e può venire in seguito eliminata;

```
(setq message-default-mail-headers "Cc: \ nBcc: \n")
```

aggiunge gli header `Cc:` e `Bcc:` nel buffer della mail;

```
(add-hook mail-mode-hook 'turn-on-auto-fill)
```

attiva l'*auto-fill*, ovvero il completamento automatico;

```
(setq mail-yank-prefix > )
```

imposta lo stile delle citazioni dei messaggi precedenti. In questo caso usa come simbolo la parentesi uncinata chiusa.

1.1.3 Il MessageMode

Il `MailMode` è da preferire senz'altro nel caso Emacs sia l'editor di un qualche programma di gestione della posta (come Mutt) - o anche di newsgroup (come slrn e tin). Tuttavia è possibile ricorrere ad altre modalità di invio della posta elettronica, anche più ricche di opzioni e potenzialità.
Il primo elemento discriminante è il `mail user agent` adottato per inviare via SMTP. Emacs usa la libreria SMTP. Il `mail user agent` di default usa la variabile `send-mail-function`. Per il `MessageMode` e per `Gnus` invece la variabile adottata è `message-send-mail-function`. Tale variabile può assumere differenti valori, a seconda dell'agente di trasmissione (cioè dell'MTA) che si intende adottare (in questa sede considereremo solo quelli essenziali per un'impostazione di base):

```
message-send-mail-with-sendmail
message-send-mail-with-mh
message-send-mail-with-qmail
message-smtpmail-send-it
smtpmail-send-it
feedmail-send-it
message-send-mail-with-mailclient
```

Ciascuna di queste modalità presente un diverso metodo di composizione e ha comandi diversi e un differente formato. Qui prendiamo in considerazione una sola alternativa: se `mail-user-agent` viene impostato a `sendmail-user-agent` (il default) significa che sarà adottato il `MailMode` per comporre le mail, laddove impostato a `message-user-agent` significa che sarà adottato il `MessageMode`.

Il MailMode (che è attivo di default) si imposta con la variabile:

```
(setq send-mail-function 'smtpmail-send-it)
```

mentre il MessageMode e Gnus con la variabile:

```
(setq message-send-mail-function 'message-user-agent)
```

a cui però va aggiunta l'invocazione a smtpmail.
Quando si lancia il buffer della mail col classico comando C-x m, esso si aprirà direttamente in MessageMode.
Il MessageMode, benché sia propriamente quello predefinito di Gnus, è da questo indipendente e può essere adottato in ogni caso. Ha un suo manuale disponibile in Emacs, attivabile col comando:

```
C-h i m message [Invio]
```

Si tratta - come abbiamo visto - di una modalità alternativa al MailMode, il cui principale vantaggio sta nel fatto che, a differenza dell'altro, può gestire gli allegati MIME. Per inserire l'allegato il comando è il seguente: C-c C-a.
Altri comandi di base, a volta identici e a volte diversi o aggiuntivi rispetto al MailMode, sono:
C-c C-c: spedisce il messaggio ed esce dal buffer;
C-c C-s: spedisce il messaggio ma non esce dal buffer;
C-c C-b: va all'inizio del messaggio;
C-c C-k: cancella il messaggio;
C-c C-z: cancella il testo dal point alla fine del buffer;
C-c C-a: aggiunge un allegato;
C-c C-d: pospone il messaggio;
C-h m: mostra l'intera gamma dei comandi.
Le impostazioni di base (come user-mail-address e user-full-name) sono le stesse del MailMode.
Si può precisare il server da adottare per l'invio:

```
(setq smtpmail-smtp-server "server_smtp")
(message-send-mail-function 'message-smtpmail-send-it)
```

Altre configurationi utili, fra le tante possibili, sono:

```
(add-hook 'message-mode-hook 'toggle-input-method)
```

per usare il metodo di default di immissione;

```
(setq smtpmail-debug-info t)
```

per ottenere informazioni su possibili problemi con server SMTP;

```
(setq message-default-mail-headers "Cc: \nBcc: \n")
```

per aggiungere automaticamente gli header `Cc:` e `Bcc:` al buffer del messaggio in composizione;

```
(setq message-auto-save-directory "~/Mail/drafts")
```

per posporre l'invio del messaggio collocandolo nel file indicato.

Differire l'invio

A volte, può essere utile differire l'invio della posta, soprattutto nel caso non si disponga di una connessione permanente. In `/.emacs` andrà allora inserita la stringa:

```
(setq smtpmail-queumail t)
```

e, una volta attivata la connessione, si dovrà lanciare il comando: `M-x smtpmail-send--queued-mail`.
Di default, la posta differita viene conservata in `/Mail/queued-mail`. Per modificare questo comportamento, si può impostare una diversa directory, ad esempio con la stringa:

```
(setq message-auto-save-directory "~/Mail/drafts")
```

Non bisogna dimenticare però che in questa modalità è possibile salvare solo un messaggio, che resterà nel suo buffer originario. Se esso risale a una sessione precedente di Emacs, è necessario porsi manualmente in `MessageMode` con la funzione `M-x message-mode`.
Se vi fosse la necessità di posporre più messaggi, è necessario ricorrere al `MessageMode` di `Gnus`.
Infine, è possibile richiamare gli indirizzi da qualsiasi buffer con la funzione `M-x goto-address`. Cliccando su un indirizzo si aprirà il buffer per comporre una mail.

Salvare la posta inviata

Questo sistema di invio non consente l'archiviazione automatica della posta elettronica. È comunque possibile tenere traccia della corrispondenza in due modi: o inviandosi automaticamente una copia mediante uno header `Bcc:` o conservandone una copia in un file in locale mediante uno header `Fcc:`.
Nel primo caso la procedura si può automatizzare inserendo in `/.emacs` la stringa:

```
(setq mail-self-blind t)
```

nel secondo caso con una stringa del genere:

```
(setq mail-archive-file-name (expand-file-name "~/outgoing"))
```

In quest'ultimo caso tutte le mail verranno collocate sequenzialmente in un unico file di testo.

1.1.4 Ricevere posta

La ricezione della posta implica un po' di lavoro in più, e soprattutto la scelta del programma che gira in background. Le alternative attuali sono diverse (oltre al fatto che Emacs può interfacciarsi con programmi classici come `Fetchmail` e `Procmail`). Si tratta di `Rmail`, `Gnus`, `VM`, `MH-E`, `Wanderlust` e `Mew`. `Rmail` è stato sempre tradizionalmente incluso in Emacs, anche se in Debian viene fornito in un pacchetto differente. `Gnus` è incluso in Emacs a partire dalla versione 23. Gli altri programmi dovranno essere installati separatamente.

Usare Rmail

`Rmail` è il tradizionale MTA di Emacs (che però anche agli utenti di Emacs più affezionati e smaliziati appare ormai un po' invecchiato. La gestione della posta in `Rmail` si attiva con la funzione `M-x rmail`.
Se si intende ricevere posta da un server POP3, la configurazione di `Rmail` è molto semplice, richiedendo solo poche stringhe. Quelle minimali possono assumere due diverse configurazioni. La prima consiste nel passare come indirizzo del server POP un URL contenente il nome del server e quello dell'utente:

```
(setenv "MAILHOST" "server\pop3 ")
(setq rmail-primary-inbox-list '("pop://nome_utente @nome_host" )
rmail-pop-password-required t)
```

La seconda sintassi, mantenuta per retrocompatibilità, adotta una forma forma leggermente diversa, e cioè `po:username:hostname`, che equivale comunque a `pop://username@hostname`.

```
(setenv "MAILHOST" "server_pop3 ")
(setq rmail-primary-inbox-list '("po:nome_utente ")
rmail-pop-password-required t)
```

dove si indica il server POP3 alla prima riga, il nome dell'utente alla seconda e la richiesta di password alla terza. `Rmail` usa il programma `Movemail` per scaricare la posta. Una versione piuttosto obsoleta è già predente in `Rmail`, oppure è possibile scaricare separatamente una versione più aggiornata e duttile: si tratta di un componente del pacchetto `Mailutils`. `Rmail` supporta non senza qualche occasionale problema l'autenticazione SSL/TLS, ma presenta un altro difetto - apparentemente banale ma decisivo - che finora non è stato risolto. `Movemail` interpreta gli indirizzi in modo da intendere tutto quello che trova dopo il (primo) *at* (@) come indirizzo (dominio) del server. Ciò comporta che non è in grado di intendere correttamente quei nomi utente che includono anche il dominio, come ormai accade per la più gran parte dei provider. Esso infatti lavora con una POP URL del tipo: `pop://username@hostname` oppure, se si vuole inserire anche la password, del tipo: `pop://username:password@hostname`.
Di conseguenza, se si cerca di accedere alla casella con un indirizzo comprensivo anche del dominio, l'operazione non andrà a buon fine e si otterrà un messaggio di errore come il seguente:

```
Loading /etc/emacs23/site-start.d/50vm-init.el (source)...done For information
about GNU Emacs and the GNU system, type C-h C-a. Empty Rmail file. Counting
messages...done Getting mail from the remote server ... movemail: mailbox
‘pop://mauro.sacchetto@gmail.com@pop.gmail.com’: cannot open: DNS name
resolution failed (No new mail has arrived) 0 new messages read No mail.
```

Infine, l'accesso a caselle di posta remote via IMAP è supportato solo dalla versione di `Movemail` presente nelle `Mailutils`. In questo caso, l'URL che indica l'indirizzo sarà nella forma seguente:

```
imap://username[:password]@hostname
```

L'indicazione della password è opzionale.

Usare Gnus

`Gnus` è un lettore di news, ma gestisce egregiamente anche la posta elettronica. Mentre fino alla versione 22 di Emacs il pacchetto `Gnus` andava installato separatamente, con la versione 23 esso è già presente all'interno di Emacs.
`Gnus` si apre ricorrendo alla funzione `M-x gnus`, e controlla automaticamente la posta all'avvio, oppure può controllarla in un secondo momento mediante il tasto `g` (= `get`).
Le configurazioni vanno inserite nel file `/.gnus`. Esse sono diverse a seconda che si scarichi la posta con IMAP oppure con POP3. Se in questo file sono presenti anche le password, anche in questo caso sarà opportuno chmodarlo a 600. Inoltre, bisogna impostare il backend a cui Gnus deve fare ricorso.
Per l'archiviazione delle mail, di default `Gnus` usa un archivio virtuale. La sua struttura è la seguente:

```
(nnfolder "archive"
          (nnfolder-directory   "~/Mail/archive")
          (nnfolder-active-file "~/Mail/archive/active")
          (nnfolder-get-new-mail nil)
          (nnfolder-inhibit-expiry t))
```

È però possibile adottare un diverso sistema. Il più diffuso e adottato è `nnml`. Esso archivia ogni mail in un singolo file e risulta pertanto molto veloce. Se però si preferisse avere un file per un gruppo di mail, per evitare il moltiplicarsi di numerosi piccoli file, allora si dovrà adottare `nnfolder`. Queste impostazioni di base possono essere ulteriormente affinate, ad esempio per avere con `nnfolder` un file che raccoglie mensilmente la posta per passare il mese successivo a un file nuovo.
Nei due casi avremo allora rispettivamente il codice che segue:

```
(setq gnus-select-method '(nnml ""))
```

e

```
(setq gnus-select-method '(nnfolder ""))
```

Volendo utilizzare `Gnus` esclusivamente come client di posta, al fine di inibire la ricerca di messaggi sui newsgroup, si può adottare il codice seguente:

```
(setq gnus-nntp-server nil
      gnus-read-active-file nil
      gnus-save-newsrc-file nil
      gnus-read-newsrc-file nil
      gnus-check-new-newsgroups nil)
```

Nel caso si adotti POP3 con una connessione cifrata, il codice minimo opportuno da inserire sarà il seguente (esemplifichiamo ancora una volta con i dati relativi a Gmail):

```
(setq gnus-select-method '(nnml "mail"))
(setq gnus-permanently-visible-groups "mail")
(setq gnus-posting-styles '((".*"
                             (name "''Nome Cognome")
                             (address "''indirizzo@gmail.com"))
                            ("^nnml.*gmail"
                             (address "''indirizzo@gmail.com"))))
(setq mail-sources '(pop :server "pop.gmail.com"
                         :port 995
                         :user "indirizzo@gmail.com"
                         :password password
                         :connection ssl
                         :leave t))
```

L'opzione `:leave t` non è in realtà necessaria per Gmail, ma andrà inserita almeno finché si sperimenta il corretto funzionamento del sistema, per evitare il *flushing* delle mail, cioè il loro prelievo e la loro cancellazione dal server. Se si è in possesso di più account, essi andranno inseriti di seguito. Ad esempio, potrebbe aversi:

```
(setq gnus-select-method '(nnml "gmail.com"))
(setq gnus-permanently-visible-groups "mail")
(setq gnus-posting-styles '((".*"
                             (name "Nome Cognome")
                             (address "indirizzo @gmail.com"))
                            ("^nnml.*gmail" (address "indirizzo@gmail.com"))))
(setq mail-sources '((pop :server "pop.gmail.com"
                          :port 995
                          :user "nome_utente@gmail.com"
                          :password "password"
                          :connection ssl
                          :leave t)
                     (pop :server "pop.istruzione.it"
                          :user "nome_utente"
                          :password "password")))
```

È inoltre possibile leggere la posta presente in altre locazioni:
1) da un tradizionale file di spool presente in locale:

```
(eval-after-load "mail-source"
 '(add-to-list 'mail-sources '(file :path "percorso_ del_file_di_spool")))
```

2) da una Maildir, con un file per mail, come usano ad esempio `Postfix`, `Qmail` e (se così impostato) `Fetchmail`, con eventuale indicazione delle subdirectory in cui prelevare le mail (in questo caso esemplifichiamo con la subdirectory `/news`):

```
(eval-after-load "mail-source" '(add-to-list
                                 'mail-sources '(maildir :path "percorso_alla_Maildir"
                                 :subdirs ("news"))))
```

3) da alcuni file presenti in una directory, nel caso, ad esempio, `Procmail` abbia già distribuito le mail:

```
(eval-after-load "mail-source"
     '(add-to-list 'mail-sources '(directory :path "percorso_alla_directory_di_procmail"
      :suffix ".prcml")))
```

dove `:suffix .prcml` suggerisce a `Gnus` di usare i file col suffisso `.prcml`.
Grazie alle direttive

```
(setq message-directory "~/Mail")
(setq message-auto-save-directory "~/Mail/drafts")
```

le mail verranno salvate in una pseudo-directory visibile nel buffer `Group` con la denominazione `nndrafts:drafts`. Una volta aperta, sarà sempre possibile ritornare in quel buffer col tasto `q`.
Nel caso si adotti IMAP, ci troviamo davanti a due alternative. Se intendiamo semplicemente usare IMAP come POP3, ossia fare in modo che le mail siano prelevate dal server IMAP e trasferite in locale, allora in `/.gnus` si dovrà inserire un codice come il seguente:

```
(add-to-list 'mail-sources '(imap :server "mail.mycorp.com"
                                  :user "username" :pass "password"
                                  :stream network
                                  :authentication login
                                  :mailbox "INBOX"
                                  :fetchflag " Seen"))
```

Se invece si intende usare IMAP nel senso in cui è propriamente inteso, allora di dovrà adottare un approccio differente e ricorrere a `nnimap` come backend.
Il codice minimo da inserire sarà il seguente:

```
(setq gnus-select-method '(nnimap "gmail"
                          (nnimap-address "imap.gmail.com")
                          (nnimap-server-port 993)
```

```
(nnimap-stream ssl)
(nnir-search-engine imap)
(nnimap-authinfo-file "~/.authinfo")
(nnimap-list-pattern "archive.*")))
```

In questo esempio, i dati sensibili non stanno in `/.gnus`, ma in `/.authinfo`. Naturalmente, `Gnus` può leggere sia le news sia le mail. In questo caso, tralasciando qui l'impostazione per ricevere le news, si dovrebbe adottare a proposito delle mail la stringa:

```
(add-to-list 'gnus-secondary-select-methods)
```

Le mail possono essere organizzate anche per vari sottogruppi, a seconda del mittente. Ciò si effettua ricorrendo alla variabile `nnmail-split-method`, che volendo consente un'organizzazione più razionale e ordinata della posta. Ogni regola filtro consiste in un gruppo e in un'espressione regolare che può ricorrere ai vari campi `From: Subject` ecc. Ad esempio potrebbe aversi:

```
(setq nnmail-split-method
     '(("junk" "^From:."Bill Gates")
      ("other" "")))
```

Usare Gnus anche per spedire

`Gnus` è in grado anche di inviare mail. Se ci si vale di un MTA come `sendmail` non è necessario fare nulla. Se invece si vuole inviare posta direttamente da `Gnus` a un server SMTP, allora è necessario specificare a quale `mail-user-agent` si intende fare ricorso.

```
(require 'smtpmail)
(setq mail-user-agent 'message-user-agent
       smtpmail-starttls-credentials '(("smtp.gmail.com" 587 nil nil))
       smtpmail-auth-credentials (expand-file-name "~/.authinfo")
       smtpmail-default-smtp-server "smtp.gmail.com"
       message-send-mail-function 'message-smtpmail-send-it
```

```
       smtpmail-smtp-server "smtp.gmail.com"
       smtpmail-smtp-service 587)
(setq message-signature "Nome Cognome \nIndirizzo \nCAP Città
\ntel.: Numero di telefono \ncell.: Numero di cellulare \nmail: mail")
(setq message-default-mail-headers "Cc: \nBcc: \nFCC: \nSENT \n")
```

Il buffer per comporre una nuova mail si apre mediante il tasto `m` (laddove il buffer per un nuovo messaggio a un newsgroup si aprirebbe col tasto `a`) e si invia sempre con `C-c C-c`. Infine `C-c C-d` pospone il messaggio nel caso non lo si voglia o possa spedire subito.

samiel

Capitolo 2

Storia e filosofia di Debian

In questa sezione verranno proposti articoli a riguardo la storia e la filosofia che sta dietro al sistema operativo debian.

2.1 Zack: un esempio di sviluppatore italiano

Il 15 Aprile si sono concluse, con la vittoria di Stefano Zacchiroli, le elezioni per il nuovo *Debian Project Leader*. Prima di dare il meritato spazio a *Zack* e a quelle che saranno le scelte e i progetti che dovrà affrontare nel suo nuovo ruolo, può essere interessante focalizzare l'attenzione su alcuni aspetti che hanno reso questa campagna di voto per la nomina a DPL particolare e diversa dalle altre. Innanzitutto, un dato significativo è rappresentato dalla notevole partecipazione della comunità al dibattito suscitato dalla campagna di voto, partecipazione che si è espressa poi in una della più elevate percentuali di votanti degli ultimi anni. Gli aventi diritto, infatti, erano quest'anno 886: un numero inferiore rispetto agli anni passati perché sono stati eliminati svariati account di persone che non contribuivano più da anni, così da rendere più aderente alla realtà l'ammontare dei votanti. La percentuale di votanti si è attestata sul 49,21%: se si confrontano questi dati con quelli dell'anno scorso, quando gli aventi diritto erano 1013 e la percentuale di votanti fu del 35,63%, si nota una netta differenza nella partecipazione. (vedi tabella; fonte: http://lists.debian.org/debian-devel-announce/2010/04/msg00009.html)

Year	Num DDs	Quorum	Valid Votes	Unique Voters	Rejects	% Voting	Multiple of Quorum
1999	347	27,942		208		59,942	7,44399
2000	347	27,942		216		62,248	7,73030
2001	??	??		311			
2002	939	45,965	509	475	122	50,586	10,33395
2003	831	43,241	510	488	200	58,724	11,28559
2004	908	45,200	506	482	52	53,084	10,66372
2005	965	46,597	531	504	69	52,228	10,81615
2006	972	46,765	436	421	41	43,313	9,00246
2007	1036	48,280	521	482	267	46,525	9,98343
2008	1075	49,181	425	401	35	37,302	8,15356
2009	1013	47,741	366	361	43	35,636	7,56155
2010	886	44,648	459	436	88	49,210	9,76513

Questo è un dato estremamente incoraggiante, che indica come la comunità degli sviluppatori sia viva e presente, interessata a questioni vitali per il futuro di Debian, come quelle poste ai candidati durante la campagna di voto sulla mailing list debian-vote@lists.debian.org, tra cui: il tema del processo di rilascio delle versioni stabili [1], il tema relativo ai requisiti di licenza e copyright dei programmi [2], o ancora quello relativo

[1] http://lists.debian.org/debian-vote/2010/03/msg00071.html
[2] http://lists.debian.org/debian-vote/2010/03/msg00192.html

alla mancanza di interesse, da parte dei nuovi DD, per il mantenimento di pacchetti ed aree di importanza vitale per la struttura di Debian [3].

Infine un fatto nuovo e degno di nota è, senza dubbio, la candidatura di Margarita Manterola: la prima donna candidata come DPL. In un ambito, come quello dell'informatica in generale e del Free Software in particolare, caratterizzato da una cronica sotto rappresentazione del genere femminile, la candidatura di *Marga*, e il numero elevato di preferenze ottenute, è sicuramente un segnale di cambiamento.

2.1.1 Chi è *Zack*?

Nella vita di tutti i giorni, Stefano *Zack* Zacchiroli è un ricercatore universitario in informatica: svolge un post-dottorato nel laboratorio PPS (*Preuves, Programmes et Systèmes*) all'Università Paris Diderot. In particolare, lavora al progetto *mancoosi* [4] nell'ambito del quale vengono applicati metodi formali per lo studio delle distribuzioni FOSS (*Free and Open Source Software*): tale progetto fornisce regolarmente strumenti di controllo utili per la comunità come *edos-debcheck*, *edos.debian.net* e svariati servizi QA utilizzati da Debian e da altre distribuzioni. *Zack* diventa uno sviluppatore Debian (*Debian Developer, DD*) nel Marzo 2001. Inizialmente, la sua partecipazione al Progetto Debian è limitata al mero mantenimento dei suoi pacchetti, senza alcun interesse per le questioni relative alla comunità; nel corso di questi anni si occupa del mantenimento di svariati pacchetti tra cui *devscripts*, *vim*, *phyton-debian*, *turbogears-2*; inoltre diventa co-*Maintainer* del *Package Tracking System*[5], strumento che permette di seguire tutte le attività di un pacchetto. Si occupa a lungo di QA e contribuisce a formare la *Debian OCaml Task Force* [6]che gestisce circa 150 pacchetti relativi, appunto, al linguaggio OCaml[7]. Al LinuxTag del 2004, tuttavia, scocca la scintilla per la comunità Debian e da allora il suo coinvolgimento nell'aspetto più prettamente comunitario del Progetto aumenta fino, appunto, alle candidature del 2009 e 2010 come DPL.

[3] http://lists.debian.org/debian-vote/2010/03/msg00086.html
[4] http://www.mancoosi.org/
[5] http://packages.qa.debian.org/common/index.html
[6] http://wiki.debian.org/Teams/OCamlTaskForce
[7] http://caml.inria.fr/

2.1.2 *"Lather, rinse, repeat"*

Lather, rinse, repeat, scrive *Zack* nella mail con cui presenta la sua candidatura a DPL nel 2010 [8]: ovvero, in italiano, Insaponare, risciacquare, ripetere, frase che si trova in genere sul retro di una bottiglia di shampoo, e che si usa per indicare, ironicamente, chi prova e riprova fino ad ottenere il risultato desiderato. Non è infatti la prima volta che si candida: l'anno scorso fu lo sfidante dell'allora DPL in carica Steve McIntyre, finendo battuto per una manciata di voti. Un risultato più che dignitoso, e dunque: insaponare, risciacquare, ripetere.

Il programma

Oggi, come allora, *Zack* si è presentato con un programma estremamente completo e articolato, che mostra una visione chiara delle questioni più importanti da affrontare affinché Debian possa mantenere gli standard che le sono propri e confrontarsi con le altre distribuzioni senza per questo perdere gli elementi che ne costituiscono l'unicità: la peculiare struttura a base totalmente democratica, il ruolo fondamentale delle migliaia di volontari che agiscono all'interno della comunità e che determinano confini fluidi tra semplici utenti e sviluppatori e, soprattutto, la filosofia e l'etica del Free Software a cui aderisce strettamente. Il programma rimane sostanzialmente lo stesso da un anno all'altro: pochissimi i cambiamenti [9], che riguardano soprattutto il rapporto con le distribuzioni derivate (in particolare Ubuntu) e il ruolo fondamentale del sito web nel veicolare l'ideale alla base di Debian. In questo articolo ne verranno analizzate solo alcune parti, chi volesse approfondire può leggere il programma completo a questo indirizzo:
http://www.debian.org/vote/2010/platforms/zack

- Premessa: una *do-ocracy* imperfetta

La premessa fondamentale da cui parte *Zack* nell'elaborazione della sua piattaforma di voto è che il Progetto Debian, in virtù della base volontaria su cui si fondano i contributi che lo animano e costituiscono, può essere identificato sostanzialmente come una *do-ocracy* ovvero un sistema in cui gli individui si attivano direttamente per far funzionare le cose, e decidono loro stessi che compiti svolgere ed in che modo, senza alcun tipo

[8] http://lists.debian.org/debian-vote/2010/03/msg00001.htm
[9] http://www.debian.org/vote/2010/platforms/zack#sec:changelog

di imposizione.Il rischio, tuttavia, è quello che diventi una *do-ocracy* imperfetta, a causa delle dimensioni del progetto. Questi, in particolare, gli scenari paventati: potrebbero essere aggiunte delle restrizioni all'accesso per limitare il pericolo di interventi dannosi, cosa che però impedirebbe l'apporto spontaneo dei volontari e dunque il corretto funzionamento della *do-ocracy*; si potrebbe verificare il caso in cui nessuno sia abbastanza motivato da svolgere compiti sgradevoli o comunque non particolarmente interessanti, ma comunque necessari; e infine, esiste la possibilità che chi gode di posizioni ormai acquisite abbassi sensibilmente il livello del proprio contributo e non ammetta di non essere più in grado di svolgere i compiti richiesti. Il ruolo del DPL consisterebbe quindi, attraverso i compiti che gli sono propri in base alla Costituzione Debian [10], nel cercare di mitigare le possibili imperfezioni della *do-ocracy*: *Zack* si propone di farlo applicando una visione a tutto tondo della politica di Debian rivolta sia all'interno che all'esterno della struttura organizzativa del progetto.

- Rapporti interni

Per quanto riguarda i rapporti interni, *Zack* sembra proporre una razionalizzazione generale dei ruoli insieme ad una più efficace comunicazione circa lo svolgimento dei compiti: è il caso, ad esempio, della razionalizzazione delle delegazioni, come anche della spinta verso una maggiore trasparenza e comunicazione da parte dei *Core Teams*.
Un'altra tendenza significativa è quella all'apertura verso la comunità, che si esprime nelle riflessioni circa i *Debian Maintainer*[11]: la creazione dei *Debian Maintainer* ha rappresentato infatti, secondo *Zack*, un'innovazione positiva nella struttura del Progetto Debian, permettendo l'ingresso di un gran numero di volontari e allo stesso tempo fornendo una sorta di tirocinio per i volontari stessi affinché migliorino la propria capacità ed esperienza qualora desiderino diventare *Debian Developer*. È necessario quindi rafforzare questa tendenza, migliorando la comunicazione circa il processo di coinvolgimento dei volontari nel Progetto, ma garantendo allo stesso tempo un accesso controllato che permetta ai volontari di acquisire le abilità adeguate. In questa direzione è poi orientata la proposta verso il rafforzamento del mantenimento collaborativo dei pacchetti, che consenta anche una migliore qualità del pacchetto stesso. Sempre in materia di rapporti interni e di coesione c'è il proposito di *Zack* di favorire il più possibile, attraverso il

[10] http://www.debian.org/devel/constitution#5
[11] http://wiki.debian.org/DebianMaintainer

finanziamento con i fondi del Progetto Debian, l'organizzazione di meeting e incontri faccia a faccia tra i volontari.

- Rapporti con l'esterno

Particolarmente interessanti sono le considerazioni fatte da *Zack* in merito ai rapporti tra Debian e le distribuzioni derivate, ed in particolare Ubuntu. Il tema dei rapporti con Ubuntu è un tema piuttosto discusso, che coinvolge fortemente aspetti quali le politiche di rilascio e le strategie di coordinamento per la risoluzione dei bug. E la collaborazione con Canonical a questo proposito non sempre è ben vista dalla comunità o dagli stessi sviluppatori, che temono un abbassamento della qualità dei pacchetti Debian a causa delle pressioni e delle ingerenze di Canonical. Da un lato infatti *Zack* ribadisce che, così come Debian si impegna a restituire alla comunità del Free Software quanto le deve poiché è composta di pacchetti che fanno parte del mondo del Free Software, allo stesso modo le distribuzioni derivate devono comportarsi - sebbene non si possa forzarle a farlo - con Debian che rappresenta la fonte principale dei programmi che le compongono. Bisognerebbe quindi da un lato, fornire un esempio di come sia necessario sdebitarsi (ad esempio tracciando pubblicamente l'invio di *patch* ai pacchetti *upstream*); dall'altro, rendere loro più semplice possibile la collaborazione con Debian, magari partecipando a iniziative fra varie distribuzioni e favorendo l'upload di pacchetti da parte dei *Non-Maintainer* quando vengono fornite delle patch. Una riflessione a parte è dedicata ad Ubuntu, sicuramente la più diffusa tra le distribuzioni derivate, forte di una comunità di utenti probabilmente più ampia di quella di Debian stessa. Questo fatto, secondo *Zack*, non va ignorato, anzi va sfruttato in vari modi:

1. dal punto di vista *tecnico*, si dovrebbe permettere ai *Debian Developer* di interagire di più con la comunità di Ubuntu, proprio perché questo potrebbe garantire migliori risultati in termini di correzione di bug e migliorie varie al software;

2. dal punto di vista *politico*, bisognerebbe sottolineare come Ubuntu contenga il 70% di pacchetti Debian non modificati: dovrebbe quindi, in base alle regole proprie del mondo FOSS, attribuire il giusto riconoscimento a Debian

3. dal punto di vista della *comunicazione*, si deve sempre e comunque ricordare che esistono delle differenze fondamentali e importantissime tra Debian e Ubuntu che attengono all'etica stessa delle due distribuzioni e influiscono quindi su scopi e modi di agire e di prendere le decisioni.

- Progetti: il sito web

Proprio alla comunicazione è dedicato uno dei progetti specifici più interessanti del programma di *Zack*: il ripensamento e la ristrutturazione del sito ufficiale Debian. Il sito Debian è da molti anni croce e delizia degli utenti: la grafica retrò e poco accattivante è per alcuni un vessillo di purezza geek, e per altri un vero e proprio cruccio.
In prima battuta può sembrare una preoccupazione frivola, tuttavia è evidente che il sito web rappresenta lo strumento principe attraverso cui far conoscere gli ideali alla base di Debian e mostrare l'unicità di una distribuzione che è libera da cima a fondo, nel software e nelle infrastrutture, che è democratica nei processi decisionali, che non è guidata dall'interesse o dal denaro ma dall'etica hacker quindi dalla passione, dall'intelligenza, dall'impegno e dall'amore per la libertà e la condivisione della conoscenza. Tutti requisiti che poche altre distribuzioni possono vantare e il sito web è il mezzo attraverso cui mostrare al mondo queste caratteristiche. Ecco perché questa proposta di rinnovo del sito e il modo in cui *Zack* la esprime nel suo programma, sembra particolarmente benaugurante per il futuro di Debian: perché con essa si ribadisce con forza la natura e l'essenza stessa di un progetto che si fonda sugli ideali più potenti e rivoluzionari del Free Software e nel contempo questa identità la si comunica all'esterno, scartando a priori il teorema secondo cui gli utenti vanno attirati con ogni mezzo, quand'anche questo comporti scelte di dubbio valore etico. Da tempo il team WWW elabora proposte per la nuova veste del sito [12], tuttavia la questione è tutt'altro che risolta. Secondo Zack, per migliorare la situazione bisognerebbe concordare una serie di obiettivi tecnici da raggiungere, tra cui:

1. stabilire i requisiti minimi per un sito Web migliore in termini di: contenuti, struttura, accessibilità e work-flow, e renderli pubblici;

2. chiedere aiuto alla comunità per ottenere i requisiti antecedentemente elencati entro un certo lasso di tempo;

3. assicurarsi di (far) avere le risorse per svolgere i compiti prestabiliti.

Nel caso ciò non portasse ad ottenere i risultati desiderati, il piano d'emergenza consisterà nell'incaricare un collaboratore esterno di realizzare la pagina, sotto la supervisione del team WWW.

[12] http://wiki.debian.org/DebianWebsiteDiscussion

2.1.3 Conclusioni

Nelle sue prime *Pillole dal DPL* [13], inviate due giorni dopo l'elezione, *Zack* si propone come facilitatore: esorta i volontari dei team a parlare con lui di qualsiasi problema impedisca loro di svolgere al meglio i loro compiti e anticipa che rimetterà in moto, prima possibile, la *team review*: una sorta di analisi del metodo di lavoro e dell'efficienza dei vari team, inaugurata circa due anni fa da Steve McIntyre. Inoltre, dal momento che non nominerà un *Second in charge* (un vice, che collabori a stretto contatto con lui), come aveva già peraltro annunciato nel programma, *Zack* esorta coloro che se la sentono a proporsi come delegati: chi si sente in grado di migliorare un'area specifica del proprio progetto e intende prendersi l'onere di farlo, e ritiene che avere una delega diretta da parte del DPL possa essere necessario per la riuscita del progetto stesso, allora può farne richiesta. In questi giorni *Zack* si sta occupando anche di un altro aspetto evidenziato nel suo programma ovvero quello relativo alle sponsorizzazioni delle partecipazioni dei volontari alle Debconf: portando avanti un progetto già avviato dal precedente DPL Steve McIntyre, ha messo a disposizione dei fondi speciali per i *Debian Developer* e *Debian Maintainer* che non sono mai stati ad una DebConf e vorrebbero partecipare a quella di quest'anno (per maggiori informazioni si veda: http://wiki.debian.org/DebConfNewbies).

Sicuramente, dunque, *Zack* ha le idee estremamente chiare rispetto alle aree su cui intervenire per facilitare il lavoro della comunità e rendere migliore Debian: questo emerge prepotentemente da un programma che è tutto fuorché vago o nebuloso. Secondo l'autrice di questo articolo è importante osservare come, al di là delle proposte più pragmatiche e di carattere organizzativo della cui validità è in grado di giudicare solo chi si trova realmente all'interno dei meccanismi della struttura Debian e non i semplici utenti, *Zack* intenda riaffermare con forza l'identità di Debian cercando di valorizzarne gli aspetti etici e quelli comunitari. In tempi in cui si parla sempre più di Open Source e sempre meno di Free Software, in cui Philip van Hoof propone l'uscita di GNOME dalla GNU Foundation [14], in cui le ultime release di casa Ubuntu adottano un gestore di pacchetti, *Ubuntu Software Center* (inizialmente chiamato *Ubuntu Software Store*), che

[13] http://lists.debian.org/debian-devel-announce/2010/04/msg00011.htm

[14] http://mail.gnome.org/archives/foundation-list/2009-December/msg00054.html per un resoconto sulla questione si veda: http://idl3.wordpress.com/2009/12/14/stallman-sgrida-gnome-e-gnome-minaccia-di-uscire-da-gnu

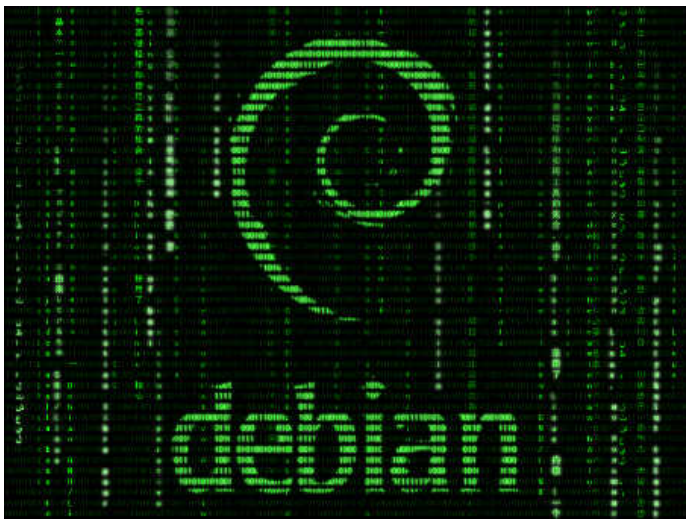
permetterà in futuro l'acquisto e l'installazione di software proprietario [15], è importante per il Progetto Debian non lasciarsi irretire dalle sirene del cambiamento, ma rimanere saldo sulle basi etiche che ne fanno un progetto esemplare e che si riflettono poi sulla qualità tecnica del codice prodotto.
In tal senso, da parte dell'autrice di questo articolo, dello Staff di Debianizzati e di tutta la comunità di Debianizzati, un grande *in bocca al lupo* a *Zack*!

MadameZou

[15] https://wiki.ubuntu.com/SoftwareCenter?action=show&redirect=SoftwareStore#October%202010

Capitolo 3

Il sistema operativo Debian



In questa sezione, tutto sul sistema operativo debian GNU/Linux (per debian GNU/hurd si segua la sezione dedicata).

3.1 Debian4Children

In questo articolo cercherò di spiegarvi la nascita di Debian4Children, una distribuzione liveCD e installabile, dedicata ai bambini e basata su Debian 5 Lenny.

3.1.1 Premessa

Questo progetto è nato un po' per caso, un po' per necessità.
Dopo le richieste del gruppo trashware con cui collaboro e le richieste di PC da parte della scuola elementare che frequenta mio figlio, ho pensato alla creazione di una distribuzione minimale con una serie di pacchetti pre-installati, rivolta ad un uso in ambiente Educational.
Grazie alla collaborazione del forum di Debianizzati, sono riuscito a muovermi dentro questa nuova esperienza, e ora il progetto si chiama: **Debian4Children**.
Il progetto è ancora in fase di sviluppo, però da questo articolo, è si può già capire come sia stato possibile realizzare una base su cui lavorare.
Il progetto è aperto a tutti coloro che vogliano collaborare.

Trashware

Trashware vuol dire promuovere una logica di riuso e non di spreco. Di fronte al disastro ambientale provocato dall'enorme mole di PC che ogni anno vengono dismessi (parliamo di centinaia di migliaia di macchine), il Trashware lavora per allungare la vita dei computer attraverso una gestione più efficiente del software.
I PC vengono dismessi troppo rapidamente (in media 2-3 anni) in obbedienza a logiche di mercato che finiscono per svantaggiare gli utenti finali. Abbiamo verificato che a rendere obsoleto l'hardware è in realtà il software che viene usato, spesso non ottimizzato e inutilmente sofisticato per gli utenti comuni. Questi, nella maggior parte dei casi, avrebbero bisogno di semplici programmi di scrittura, calcolo, navigazione in Internet e grafica. Invece la continua rincorsa tra hardware e software, fomentata dalle grandi multinazionali dell'informatica, costringe all'acquisto di macchine sempre più potenti con le quali si svolgono praticamente le stesse funzioni di dieci anni fa.
Il Software Libero, invece, in virtù della natura aperta del codice sorgente, consente ampi margini di ottimizzazione. Lavorandoci con attenzione, un vecchio computer dismesso può arrivare ad avere prestazioni paragonabili a quelle di un PC di ultima generazione

per determinati tipi di applicazioni. Tra le varie distribuzioni disponibili, per questo si è scelto di lavorare con la distribuzione GNU/Linux Debian, sia per ragioni tecniche (è indubbiamente la più modulare e tra le più performanti) sia politiche, in quanto si tratta di un progetto non commerciale basato su un contratto sociale con l'utente.
Riferimento:
http://www.binarioetico.org/index.php?option=com_content&task=view&id=26&Itemid=68
Per maggiori informazioni, potete cercare il Trashware più vicino a voi: lì potrete avere anche la possibilità di partecipare.

3.1.2 Introduzione

In questa guida analizzeremo i passaggi che abbiamo eseguito per costruire una distribuzione chiamata Debian4Children.
Debian4Children ha le seguenti caratteristiche:

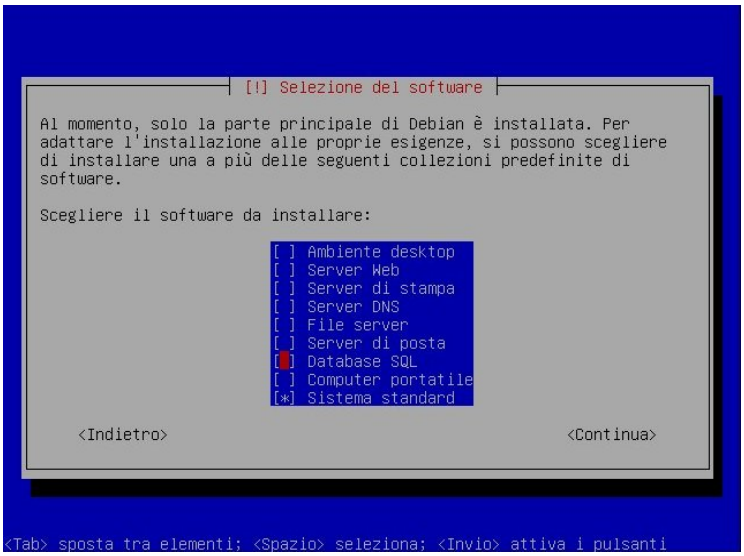
- basata su Debian 5.04 Lenny;
- versione Live e installabile. (Questo fa sì che sia possibile testare l'hardware prima dell'installazione);
- ha un peso di 700 MB e quindi è masterizzabile su un singolo CD;
- come desktop environment utilizza KDE 3.5.10;
- adatta per essere installata su computer datati o caratterizzati da poche risorse (va bene anche su macchine nuove: è pur sempre una Debian!);
- basata sul kernel 2.6.32-3 (in aggiunta al kernel 2.6.26);
- dotata di OpenOffice4Kids 0.9 in italiano.

Ricco parco software dal ramo educational, per lo più indicato per le scuole elementari.
Per la creazione di questa distribuzione è stato necessario utilizzare una Debian minimale usando la versione *net-install*, alla quale successivamente sono stati aggiunti i componenti essenziali per rendere il sistema idoneo alle nostre particolari esigenze. La versione *net-install* consente una prima installazione con lo stretto necessario al funzionamento del sistema; da qui, è poi possibile iniziare a comporre una versione personalizzata.
In seguito, grazie all'uso di *Remastersys* è stato possibile creare un'immagine *.iso* del sistema operativo, che ha il vantaggio di essere live e installabile in pochi passi.

3.1.3 Debian Lenny 5.04 con net-install

Come precedentemente annunciato, preoccupiamoci di avere a disposizione l'immagine *.iso* di Debian Lenny, che andremo a cercare alla pagina www.debian.org/distrib/netinst. Per l'installazione è possibile usare un PC oppure una macchina virtuale. Cominciamo con l'avviare il sistema da CD, impostando al boot l'avvio corretto se necessario.

Seguiamo i semplici passi che ci portano a una prima configurazione del sistema, fino ad arrivare al punto di dover scegliere alcune preconfigurazioni di pacchetti.



Dopo aver installato il sistema minimale della net-install, riavviamo e cominciamo a sporcarci le mani con la shell (*dò per scontato che la rete sia funzionante*).

```
Starting portmap daemon....
Starting NFS common utilities: statd.
Setting console screen modes and fonts.
INIT: Entering runlevel: 2
Starting enhanced syslogd: rsyslogd.
Starting ACPI services....
Starting MTA: exim4.
Starting NFS common utilities: statd.
Not starting internet superserver: no services enabled.
Starting deferred execution scheduler: atd.
Starting periodic command scheduler: crond.

Debian GNU/Linux 5.0 d4c tty1

d4c login: root
Password:
Linux d4c 2.6.26-2-686 #1 SMP Sat Dec 26 09:01:51 UTC 2009 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
d4c:~# _
```

Per prima cosa installiamo uno strumento per me fondamentale: MC, poi l'ambiente grafico e il nostro DE KDE3.5 in lingua italiana.

```
# aptitude update
# aptitude install mc xserver-xorg kdebase kde-i18n-it
```

```
  xserver-xorg-input-synaptics{a} xserver-xorg-input-wacom{a}
  xserver-xorg-video-all{a} xserver-xorg-video-apm{a}
  xserver-xorg-video-ark{a} xserver-xorg-video-ati{a}
  xserver-xorg-video-chips{a} xserver-xorg-video-cirrus{a}
  xserver-xorg-video-cyrix{a} xserver-xorg-video-dummy{a}
  xserver-xorg-video-fbdev{a} xserver-xorg-video-glint{a}
  xserver-xorg-video-i128{a} xserver-xorg-video-i740{a}
  xserver-xorg-video-imstt{a} xserver-xorg-video-intel{a}
  xserver-xorg-video-mach64{a} xserver-xorg-video-mga{a}
  xserver-xorg-video-neomagic{a} xserver-xorg-video-nsc{a}
  xserver-xorg-video-nv{a} xserver-xorg-video-openchrome{a}
  xserver-xorg-video-r128{a} xserver-xorg-video-radeon{a}
  xserver-xorg-video-radeonhd{a} xserver-xorg-video-rendition{a}
  xserver-xorg-video-s3{a} xserver-xorg-video-s3virge{a}
  xserver-xorg-video-savage{a} xserver-xorg-video-siliconmotion{a}
  xserver-xorg-video-sis{a} xserver-xorg-video-sisusb{a}
  xserver-xorg-video-tdfx{a} xserver-xorg-video-tga{a}
  xserver-xorg-video-trident{a} xserver-xorg-video-tseng{a}
  xserver-xorg-video-v4l{a} xserver-xorg-video-vesa{a}
  xserver-xorg-video-vga{a} xserver-xorg-video-vmware{a}
  xserver-xorg-video-voodoo{a}
0 pacchetti aggiornati, 248 installati, 0 da rimuovere e 0 non aggiornati.
È necessario prelevare 141MB di archivi. Dopo l'estrazione, verranno occupati 38
5MB.
Continuare? [Y/n/?] _
```

Prima di riavviare consiglio di rimuovere exim4 se non volete usare un MTA:

```
# aptitude remove exim4-config exim4-base
```

Ora riavviamo e cominciamo a personalizzare la nostra distro

```
# reboot
```

Configurazione

Al riavvio avremo KDM che ci aspetta. Dopo esserci autenticati accediamo al sistema.

A questo punto comincia la fase di configurazione della nostra distro personalizzata. Per installare dunque ulteriori pacchetti preferisco solitamente usare konsole e aiutarmi con una comoda GUI (kpackage) che non interferisce con aptitude da shell e mi permette di effettuare la ricerca e visualizzazione delle descrizioni e dipendenze relative ai pacchetti in modo più semplice e intuitivo. In alternativa, è possibile usare synaptic, che tuttavia richiede un discreto numero di dipendenze.
Configurate i vostri repository a seconda del tipo di pacchetti che desiderate.

```
deb http://ftp.it.debian.org/debian/ lenny main contrib non-free
deb-src http://ftp.it.debian.org/debian/ lenny main contrib non-free
```

```
deb http://security.debian.org/ lenny/updates main
deb-src http://security.debian.org/ lenny/updates main

deb http://volatile.debian.org/debian-volatile lenny/volatile main
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main

# Remastersys
#deb http://www.geekconnection.org/remastersys/repository debian/

deb http://www.backports.org/debian lenny-backports main contrib non-free
```

Per chi volesse ripetere l'installazione di D4C, qui trovate la lista dei pacchetti presenti in essa: http://dl.dropbox.com/u/3251342/d4c.txt. Per installarla eseguire:

```
# dpkg --set-selections < d4c_list.txt
# apt-get -u dselect-upgrade
```

Oppure date spazio alla vostra fantasia.
Warningbox | Se fate uso di repository non dello stesso ramo è consigliato il pinning. Per approfondimenti a riguardo
-> http://e-zine.debianizzati.org/web-zine/numero_2/?page=12:Pinning

3.1.4 Remastersys

Per una guida completa all'uso, potete fare riferimento alla documentazione presente nel wiki debianizzati, come ho fatto io
-> http://guide.debianizzati.org/index.php/Remastersys.
Remastersys [1] è uno strumento che ci permette di clonare un'installazione di un sistema Linux Debian based.
Sono possibili diverse soluzioni, tra le 2 più importanti:

- clonazione del sistema con le configurazioni;
- clonazione del sistema senza configurazioni.

[1] http://www.geekconnection.org/remastersys

Per dare più spazio alle possibili personalizzazioni, come nome utente e password, password di root e nome macchina, ho preferito la seconda soluzione, che però ci porta a dover effettuare alcune piccole operazioni di messa a punto del sistema dopo l'installazione.

Installazione

L'installazione di Remastersys presuppone l'abilitazione di un repository specifico. Per Debian (e derivate) bisogna inserire nel file */etc/apt/sources.list* la seguente riga:

```
deb http://www.geekconnection.org/remastersys/repository debian/
```

Procediamo quindi con il refresh dei pacchetti e l'installazione vera e propria del programma (tutto con privilegi di root):

```
# apt-get update
# apt-get install remastersys
```

Warningbox | If your kernel doesn't have the squashfs-modules and either the aufs-modules or unionfs-modules, you MUST use a different kernel.
Box | Nota |Per una corretta creazione dell'immagine iso, avviare un kernel che ha i moduli precedentemente indicati Per poter utilizzare la GUI di Remastersys occorre sia presente il pacchetto *zenity*, altrimenti si dovrà lavorare senza GUI con il comando *remastersys* da shell.
Tra gli altri pacchetti consiglio *gparted*, che vi permetterà dalla live creata di poter manipolare le partizioni per l'installazione.

Personalizzare il wallpaper

Possiamo inserire nel nostro LiveCD un wallpaper a piacere. In Debian4Children abbiamo pensato di utilizzare le immagini suggerite da wakko_kid. Per fare le cose per bene è meglio creare una immagine abbastanza grande (1600x1200 pixels può essere sufficiente) e posizionarla in /usr/share/wallpapers.
Successivamente clicchiamo su un punto a caso nel desktop, selezioniamo la voce Configura Desktop e scegliamo il nostro wallpaper, lo ritroveremo nel nostro LiveCD.

Personalizzazione Ksplash

Durante il caricamento del desktop di KDE, normalmente viene visualizzata una immagine con delle icone sottostanti che progressivamente si illuminano.
Questa immagine fa parte di una serie di temi (ksplash) che possono essere scelti dall'utente. Infatti, se clicchiamo su *menu K -> Impostazioni -> Aspetto e Temi -> Schermata d'avvio*, possiamo scegliere quale tema utilizzare. Il singolo tema può essere personalizzato. Per fare questo conviene duplicare la directory Default che troviamo in */usr/share/apps/ksplash/Themes* con un nome a nostra scelta, ad esempio d4c.
Entriamo nella cartella d4c (sempre con i permessi di root) e modifichiamo dapprima il file Theme.rc. Il file è un semplicissimo file di testo di facile lettura:

```
[KSplash Theme: d4c]
Name = KDE 3.5 Splash Screen D4C
Description = Debian4Children
Version = 1.2
Author = mm-barabba

# Theme behaviour settings.
Engine = Default

# Should icons blink ala KDE, or not?
Icons Flashing = true

# Show progress bar?
Always Show Progress = false

# Status text colour
Label Foreground = #FFFFFF
```

La prima parte del file riguarda le impostazioni generali del tema. Il primo importante accorgimento è quello di impostare il KSplash Theme contenuto nelle parentesi quadre con il nome esatto della subdirectory che abbiamo creato duplicando quella esistente (chiamata Default). Questo è importante altrimenti il nostro nuovo tema non funzionerà. Il resto della sezione è semplice da capire: nome, descrizione, versione e autore del tema. Possiamo scrivere quello che riteniamo opportuno.

Le altre sezioni del file riguardano la *Engine* (che è sempre meglio lasciare intatto), l'illuminazione delle icone durante il caricamento di KDE, ed eventualmente la comparsa di una *progress bar*, settata sempre a false, in quanto rovinerebbe il nostro nuovo tema.
Il *Label Foreground* indica il colore di fondo delle scritte che compaiono durante il boot di KDE. In questo caso possiamo lasciare il bianco di default (#FFFFFF), oppure inserire il colore che vogliamo in esadecimale.
Dopo aver apportato le modifiche salviamo il file.
Ora passiamo alle immagini, le immagini principali da creare sono quelle indicate:
Preview.png, splash_active_bar.png, splash_bottom.png, splash_inactive_bar.png e splash-_top.png.
Le immagini possono essere cambiate a piacimento. L'unica accortezza (intuibile) è quella di mantenere la risoluzione delle stesse (soprattutto la larghezza di 400 pixel) e l'estensione (*.png).
L'immagine Preview.png è soltanto dimostrativa del tema e può essere anche modificata per se stessa.In ogni caso, una volta creato il tema, possiamo spostarci su *menu K -> Impostazioni -> Aspetto e Temi -> Schermata d'avvio* e schiacciare il bottone Prova.
Così possiamo vedere se il tutto funziona.

Personalizzazione Grub

Nella directory */etc/remastersys/grub*, troviamo i file per personalizzare la nostra schermata di boot del LiveCD. Il file *menu.lst.debian* riprende esattamente il classico file *menu.lst* di grub.
Serve per impostare le indicazioni essenziali per il LiveCD: timeout, percorso della splash-image e le voci del menu. Possiamo editarlo (sempre con i permessi di root) e personalizzarlo come vogliamo.
Allo stesso modo possiamo creare una immagine di sfondo per il nostro bootloader. Per fare questo basterà prendere la nostra immagine preferita, ridimensionarla alla risoluzione 640x480 pixels,con una profondità di 14 colori. Possiamo ad esempio utilizzare Gimp.
Se abbiamo deciso di adoperare una immagine di sfondo, allora al precedente file *menu.lst.debian* dobbiamo forzatamente aggiungere due righe fondamentali, solitamente posizionate subito dopo la riga indicante il timeout, vale a dire:

```
foreground = ffffff
background = 000000
```

Questo servirà a indicare a grub che i testi delle varie voci avranno colore bianco (foreground ffffff) con colore di sfondo nero (background 000000). Si può provare a cambiare i colori come meglio si desidera, ma questa combinazione ha funzionato al meglio.
Preme sottolineare una cosa: se decidiamo di installare il sistema contenuto nel LiveCD, sappiate che il percorso della splash-image contenuto nel file *menu.lst.debian* verrà replicato nel sistema installato su hard-disk. Questo provocherà un errore iniziale, in quanto il bootloader installato non troverà l'immagine al percorso indicato.
Per ovviare a questo abbiamo due possibilità: o non utilizzare l'immagine di splash (basterà quindi nel file *menu.lst.debian* commentare la riga corrispondente con #), oppure copiare l'immagine di splash sull'hard-disk, nella posizione corretta indicata dal percorso.
In alternativa è possibile eliminare completamente tale immagine, rimuovendo il file sia da /boot/grub che da /etc/remastersys/grub ed eseguire:

```
#update-grub
```

Ancora una cosa...

Prima di lanciare Remastersys, sarebbe bene cancellare tutto quello che non è più necessario: tutti i file presenti nella nostra home (immagini, links, ecc).
Inoltre è utile installare il pacchetto gtkorphan (tramite apt-get install gtkorphan). Tale programma (lanciato da root) è una semplice GUI per deborphan e permette di individuare i pacchetti orfani ed eliminarli: utilissimo per creare spazio.
Altra cosa molto importante: quando siamo pronti riavviamo il sistema in modo da scaricare tutti i file temporanei presenti. Solo così saremo certi che il LiveCD che andremo a creare rispecchierà il sistema installato su hard-disk.

E ora... Lanciamo Remastersys!

Ora che abbiamo personalizzato il tutto possiamo occuparci di Remastersys. Prima di tutto entriamo in konsole (:-)) e autentichiamoci come root.

```
d4c:/home/d4c# remastersys dist
```

```
Distribution Mode Selected
Lettura della lista dei pacchetti in corso...
Generazione dell'albero delle dipendenze in corso...
Lettura informazioni sullo stato...
aufs-modules-2.6.26-2-686 è già alla versione più recente.
0 aggiornati, 0 installati, 0 da rimuovere e 0 non aggiornati.
Lettura della lista dei pacchetti in corso...
Generazione dell'albero delle dipendenze in corso...
Lettura informazioni sullo stato...
squashfs-modules-2.6.26-2-686 è già alla versione più recente.
0 aggiornati, 0 installati, 0 da rimuovere e 0 non aggiornati.
Checking if the /home/remastersys/remastersys folder has been created
Copying /var and /etc to temp area and excluding extra files
/usr/bin/remastersys: line 260: /home/remastersys/remastersys/dummysys/etc/gdm/  \
                                        gdm.conf-custom: No such file or directory
find:"/home/remastersys/remastersys/dummysys/var/crash": No such file or directory
cp: impossibile fare stat di `/etc/remastersys/grub/splash.xpm.gz': No such file \
                                                                    or directory
Setting up live options for dist mode
update-initramfs: Generating /boot/initrd.img-2.6.26-2-686
Copying your kernel and initrd for the livecd
Creating filesystem.squashfs  ... this will take a while so be patient
Adding stage 1 files/folders that the livecd requires.
Parallel mksquashfs: Using 1 processor
Creating little endian 3.1 filesystem on /home/remastersys/remastersys/ISOTMP/   \
                                    live/filesystem.squashfs, block size 131072.
[============================================================] 5274/5274 100%
Exportable Little endian filesystem, data block size 131072, compressed data,    \
                                        compressed metadata, compressed fragments,
 duplicates are not removed
Filesystem size 23559.44 Kbytes (23.01 Mbytes)
        32.00% of uncompressed filesystem size (73622.35 Kbytes)
Inode table size 73960 bytes (72.23 Kbytes)
        32.78% of uncompressed inode table size (225633 bytes)
Directory table size 62370 bytes (60.91 Kbytes)
        43.66% of uncompressed directory table size (142841 bytes)
No duplicate files removed
```

```
Number of inodes 6302
Number of files 4942
Number of fragments 224
Number of symbolic links  883
Number of device nodes 0
Number of fifo nodes 2
Number of socket nodes 6
Number of directories 469
Number of uids 9
        root (0)
        man (6)
        libuuid (100)
        statd (102)
        avahi (104)
        messagebus (103)
        haldaemon (105)
        d4c (1000)
        nobody (65534)
Number of gids 14
        lp (7)
        ssl-cert (111)
        root (0)
        games (60)
        libuuid (101)
        staff (50)
        adm (4)
        mail (8)
        avahi (107)
        lpadmin (112)
        messagebus (106)
        haldaemon (109)
        d4c (1000)
        crontab (102)
Adding stage 2 files/folders that the livecd requires.
Found a valid exportable little endian SQUASHFS superblock on /home/remastersys/ \
                                        remastersys/ISOTMP/live/filesystem.squashfs.
        Inodes are compressed
        Data is compressed
```

```
        Fragments are compressed
        Check data is not present in the filesystem
        Fragments are present in the filesystem
        Always_use_fragments option is specified
        Duplicates are not removed
        Filesystem size 23559.44 Kbytes (23.01 Mbytes)
        Block size 131072
        Number of fragments 224
        Number of inodes 6302
        Number of uids 9
        Number of gids 14

Parallel mksquashfs: Using 1 processor
Scanning existing filesystem...
Read existing filesystem, 6301 inodes scanned
Appending to existing little endian 3.1 filesystem on                          \
  /home/remastersys/remastersys/ISOTMP/live/filesystem.squashfs, block size 131072
All -be, -le, -b, -noI, -noD, -noF, -check_data, no-duplicates, no-fragments,    \
                            -always-use-fragments and -exportable options ignored

If appending is not wanted, please re-run with -noappend specified!

No recovery data option specified.
Skipping saving recovery file.

[============================================================] 73908/73908 100%
Exportable Little endian filesystem, data block size 131072, compressed data,    \
                                        compressed metadata, compressed fragments,
 duplicates are not removed
Filesystem size 683263.97 Kbytes (667.25 Mbytes)
        47.15% of uncompressed filesystem size (1449180.26 Kbytes)
Inode table size 875657 bytes (855.13 Kbytes)
        31.60% of uncompressed inode table size (2770963 bytes)
Directory table size 804843 bytes (785.98 Kbytes)
        48.19% of uncompressed directory table size (1670271 bytes)
No duplicate files removed
Number of inodes 85535
Number of files 73165
```

```
Number of fragments 6090
Number of symbolic links  4588
Number of device nodes 0
Number of fifo nodes 2
Number of socket nodes 6
Number of directories 7774
Number of uids 9
        root (0)
        man (6)
        libuuid (100)
        statd (102)
        avahi (104)
        messagebus (103)
        haldaemon (105)
        d4c (1000)
        nobody (65534)
Number of gids 21
        lp (7)
        ssl-cert (111)
        root (0)
        games (60)
        libuuid (101)
        staff (50)
        adm (4)
        mail (8)
        avahi (107)
        lpadmin (112)
        messagebus (106)
        haldaemon (109)
        d4c (1000)
        crontab (102)
        shadow (42)
        tty (5)
        nogroup (65534)
        mlocate (104)
        plugdev (46)
        ssh (105)
        src (40)
```

```
Creating debian4children_v1.iso in /home/remastersys/remastersys
Creating debian4children_v1.iso.md5 in /home/remastersys/remastersys
/home/remastersys/remastersys/debian4children_12.iso is ready to be burned or    \
                                                        tested in a virtual machine.
Check the size and if it is larger than 700MB you will need to burn it to a dvd
701M /home/remastersys/remastersys/debian4children_12.iso
It is recommended to run 'sudo remastersys clean' once you have burned and tested\
                                                          the debian4children_12.iso
```

Alla fine delle operazioni Remastersys ci indicherà che ha concluso. Inoltre ci avvertirà se l'immagine supera la capienza di un singolo CD.

Come sempre è utile controllare il checksum e provare la nuova iso su macchina virtuale.

Lascio i link dell'immagine della versione attualmente in fase di monitoraggio .

http://dl.dropbox.com/u/3251342/Debian4Children_11.iso

http://dl.dropbox.com/u/3251342/Debian4Children_11.iso.md5

E' disponibile un piccolo manuale che illustra i pochi e semplici passaggi che riguardano l'installazione e un'elenco delle applicazioni in dotazione .

http://dl.dropbox.com/u/3251342/D4C-manuale.pdf

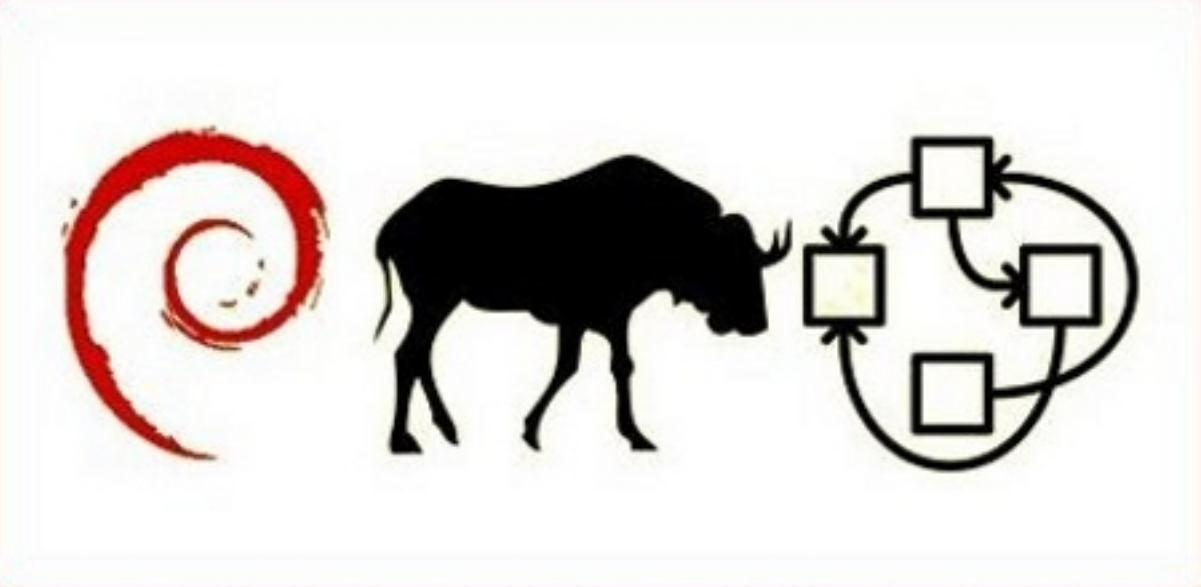
3.1.5 Note finali

Per ora sono state create 2 postazioni di prova. Si trovano installate in una scuola elementare del comune di Noceto (PR) e sono in 2 classi differenti, entrambe 1^a elementare. Sono stati necessari alcuni giorni perché i bambini si adattassero alle nuove macchine che si trovano nella classe , ma il livello di apprendimento di una mente vergine è certamente superiore nei confronti di un'adulto che ha utilizzato un sistema operativo di casa Microsoft per anni. Certi comportamenti residui nella memoria creano qualche problema agli utenti che non hanno mai usato un sistema operativo diverso da Windows , ma KDE 3.5 si fa notare per molte similitudini nelle operazioni più comuni da desktop.

Il progetto è ora sotto osservazione da parte mia, in questa prima fase mi sto interessando di raccogliere opinioni e problemi riscontrati nell'uso quotidiano da parte dei bambini e dalle maestre.
Happy Trashware con Debian :-)

Capitolo 4

Debian ports



Conoscevate solo debian GNU/Linux? Vi daremo la possibilità di studiare il vostro sistema operativo preferito sulla base di altri kernel.

Dopo i primi articoli su Debian GNU/Hurd una prima installazione di Debian GNU/kfreeBSD.

4.1 Aggiornamenti su Debian GNU/Hurd

Nel terzo numero dell'e-zine Debianizzati abbiamo elencato gli sviluppi ulteriori riguardo a debian GNU/Hurd. Il progetto, seppur a ritmo rilento, continua il suo percorso compiendo ogni mese piccoli passi.
Come nel numero precedente vogliamo illustrarvi in questo articolo i progressi ulteriori compiuti in questi ultimi sei mesi.

Pacchetti aggiornati

Riprendendo quanto citato nel numero due dell'e-zine, vediamo quali aggiornamenti sono stati portati ai pacchetti principali:

- crosshurd: passato alla versione 1.7.37 (rispetto alla 1.7.35 del numero 2). Dal *changelog*:
  - con la versione 1.7.36:
    - debian/control (Maintainer): Removed Jeff Bailey, added GNU Hurd Maintainers.
    - packages/common: Added initscripts and insserv.
    - native-install/native-install: Unpack insserv along required packages.
  - con la versione 1.7.37:
    - native-install/native-install: Explicitely use bash to invoke MAKEDEV.
    - debian/compat: Bumped to 7.
    - debian/control (Uploaders): Added Samuel Thibault.
- gnumach: passato alla versione 2:1.3.99.dfsg.git20091128-1 (rispetto alla 2:1.3.99.dfsg.cvs20090220-2 scorsa). Sempre dal *changelog*:
  - Update debian/copyright to point to the git repository.
  - Sync with upstream:
    - debian/patches/05_halt_on_panic_flag.patch: Remove, merged upstream.
    - debian/patches/12_sis900.patch: Likewise.
    - debian/patches/14_alloc_params.patch: Likewise.
    - debian/patches/15_mem_obj_proxy.patch: Likewise.

    * debian/patches/16_ide_multsect.patch: Likewise.
    * debian/patches/20_xmm_support.patch: Likewise.
    * debian/patches/05_halt_on_panic_flag.patch: Refresh.

  ** Switch to source format "3.0 (quilt)":

    * Remove quilt from Build-Depends.
    * Remove quilt.make include from debian/rules.
    * Remove patch and unpatch targets from debian/rules.
    * Remove now unneeded debian/README.source.

  – Do not include ChangeLog.0*, they are not shipped upstream anymore.
  – Remove Marcus Brinkmann from Uploaders. Closes: #503568.

GRUB2

La versione di GRUB del momento, 1.97 beta3-1, non era purtroppo in grado di digerire Hurd, citando l'articolo precedente ci eravamo lasciati così:

> Continueranno sicuramente gli esperimenti e speriamo di potervi mostrare, presto o tardi, l'avvio di GNU/Hurd utilizzando il nuovo GRUB2

Finalmente quel momento è arrivato. L'attuale versione su testing al momento della stesura dell'articolo, la 1.98-1, non solo è in grado di avviare Hurd, ma appoggiandosi su *os-prober* (1.35) ci darà grazie a *grub-mkconfig* (sempre 1.98-1, presente in *grub-common*) una configurazione del *grub.cfg* impeccabile e in modo assolutamente automatico! Nei prossimi capitoli sarà illustrato caso per caso come si può installare GRUB2 e come lo si può configurare.

Debian GNU/Hurd CD L1

Grazie a Philip Charles, come annunciato nel numero 2, dal 19. ottobre del 2009 è disponibile un nuovo CD d'installazione per questo sistema. La versione minimale (mini) conta 85Mb e contiene solamente i pacchetti essenziali. Qualora si intenda accedere a tutti i pacchetti attualmente disponibili per l'architettura GNU/Hurd occorre scaricare il DVD da 4.2Gb. Entrambe le immagini dei dischi sono scaricabili dal sito:
http://ftp.debian-ports.org/debian-cd/
Nel prossimo capitolo vedremo come installare il sistema a partire da questo supporto.

4.1.1 Installazione da CD/DVD

Possiamo scegliere se scaricare la versione mini (*debian-L1-hurd-i386-mini.iso*) o il DVD1 (*debian-L1-hurd-i386-DVD1.iso*), entrambe risalenti al 19 ottobre del 2009. Dopo aver masterizzato l'immagine disco su un CD (rispettivamente DVD) potremmo avviare il computer con quest'ultimo. L'installazione, almeno da un punto di vista pratico, è quasi uguale alla precedente K16. L'unica differenza palpabile è l'esecuzione dello script *native-install*: mentre nella K16 occorreva lanciarlo due volte, la seconda dopo un riavvio obbligatorio, con la L1 l'installazione il tutto avviene con un'esecuzione unica dello script. Un'ulteriore microscopica differenza sta nell'introduzione della *dash* come *shell* di default al posto della *bash*; alla sua configurazione ci verrà dunque proposto il suo utilizzo come *shell* di default, del resto succede analogamente con la sua installazione su Debian GNU/Linux. Infine, data anche l'introduzione di Grub2, a seconda di come abbiamo avviato il sistema avremo l'occasione di installarlo nel disco con GNU/Hurd, nel caso non lo avessimo già installato. Per la descrizione dettagliata dell'installazione vi rimandiamo all'articolo del numero uno. Al termine di quest'ultima verremo ancora invitati a riavviare in *multi-user-mode* e lanciare la console di Hurd.

Configurazione di Grub2

Dopo la configurazione di base del sistema tramite il CD saremo obbligati a riavviare la macchina. Per poi avviare il *Kernel mach*, così come il sistema operativo sarà necessario configurare Grub2. Se già lo abbiamo installato sulla macchina l'operazione è semplicissima: con un semplice *grub-mkconfig* e con *os-prober* installato, otterremo il tutto in modo automatico e senza sforzo. Il *grub.cfg* dovrà assomigliare a qualcosa di simile:

```
menuentry "GNU/Hurd (on /dev/hda4)" {
insmod ext2
set root='(hd0,4)'
search --no-floppy --fs-uuid --set 6fdea188-8766-4848-ada6-dce1bfc8f0e0
multiboot /boot/gnumach.gz root=device:hd0s4 -s
module /hurd/ext2fs.static ext2fs \
--multiboot-command-line='${kernel-command-line}' \
--host-priv-port='${host-port}' \
--device-master-port='${device-port}' \
--exec-server-task='${exec-task}' -T typed '${root}' \
```

```
'$(task-create)' '$(task-resume)'
module /lib/ld.so.1 exec /hurd/exec '$(exec-task=task-create)'
}
```

Vi ricordo che l'opzione -s dopo il *multiboot* serve ad avviare il sistema in *single-user-mode* dovremo ricordarci di toglierla, rispettivamente di creare un'altra *entry* per avviare il sistema in *multi-user-mode* (personalmente ho optato per questa seconda opzione).
Se invece non avessimo Grub2 già installato bisognerà creare un'immagine di Grub2 da avviare su un supporto esterno (floppy o CD) e poi avviare da quest'ultima il sistema. Per creare un'immagine si può utilizzare qualche trucco come *grub-mkrescue* e creare un *fileconfig* con i dati relativi all'avvio, o simili; in ogni caso potrete trovare in rete varie immagini prefabbricate, anche se il metodo che ci sentiamo di consigliarvi è quello di installare Grub2: tramite un sistema operativo *di appoggio* oppure installandolo nel MBR del disco tramite una *live*.

4.1.2 Installazione con crosshurd

L'installazione del sistema con crosshurd avviene esattamente come descritto nei numeri precedenti, attualmente con una piccola nota riguardo in particolare ad un *bug* riguardante il pacchetto *tar*. Come segnalato nel bug #577978 [1]all'estrazione del pacchetto *dash*, il sistema si blocca finendo in un *loop* infinito. Ciò è dovuto a *tar* bug #576876[2], la cui versione se superiore alla 1.22-2 (alias, se vi trovate dunque su testing o sid, dove attualmente trova posto la versione 1.23-1) non digerisce il fatto di trovare un *symlink* già preesistente */bin/sh* riferito alla *bash* mandando appunto in *loop* il sistema. Per risolvere il problema, attendendo che il bug venga risolto, basterà fare un *downgrade* del pacchetto alla versione inferiore (attualmente la 1.20.-1 da *lenny*).
Lanciato crosshurd i pacchetti verranno scaricati ed estratti nella partizione a loro riservata (v. numero 1, con l'unica differenza che il *symlink* a */usr* è stato abolito). Infine penserà ancora una volta GRUB2 (risp. os-prober) a darci una configurazione per avviare il sistema.
Al primo avvio del sistema, il device con GNU/Hurd verrà montato in *read-only* e dunque non sarà possibile eseguire in modo corretto il *native-install*. Basterà dunque un *fsck*

[1] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=577978
[2] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=576876

della partizione con GNU/Hurd (da un altro sistema dunque) e rilanciare il sistema. Si ricorda in ogni caso che la configurazione d'avvio creata da *GRUB* avrà un –readonly da togliere e dovrete in ogni caso aggiungere la solita -s per avviare ancora in *single-user mode*. Fatto tutto ciò, non ci saranno problemi a lanciare il native-install. Ciò nonostante l'installazione del sistema sarà bloccata a causa di un ulteriore bug[3].
Attualmente sembra dunque difficile utilizzare *crosshurd 1.7.37* in modo funzionale. In ogni caso, non tarderanno gli aggiornamenti e soprattutto l'installazione di Debian GNU/Hurd resta al momento possibile tramite i CD di Philip Charles.

4.1.3 Configurazione del sistema

Dopo aver installato il sistema dovremo fare qualche accorgimento, per lo più, in linea di massima con quanto già spiegato nei precedenti numeri. In breve:
* configurare /etc/fstab
Utilizzando ad esempio *nano*, inseriremo i seguenti dati:

```
# <file system> <mount point>   <type>    <options>  <dump>  <pass>
/dev/hd1s1      /               ext2      rw         0       1
/dev/hd1s2      none            swap      sw         0       0
/dev/hd2   /cdrom   iso9660fs ro,noauto  1      1
```

in ogni caso, adattando il nome dei dispositivi in funzione alla nostra situazione (vi ricordo che in questo caso, parlando ad esempio di *hd1s1* ci stiamo riferendo alla prima partizione del secondo disco).
* creare i dispositivi
Dalla directory */dev* e sempre riferito alla situazione sopra elencata:

```
hurd:/dev# MAKEDEV hd1s1 hd1s2 hd2
```

* Configurare la rete
Dapprima possiamo aggiungere un *hostname* in */etc/hostname* per togliere l'orrendo *(null)* di default. In seguito settiamo il server come di consueto (sempre facendo riferimento ai numeri precedenti):

[3]bug #557422 http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=557422

```
# settrans -fgap /servers/socket/2 /hurd/pfinet -i eth0 -a <IPInterfaccia>       \
                                        -g <IPGateway> -m <IPMascheraDiSottorete>
```

senza dimenticarci infine di inserire i *DNS* in */etc/resolv.conf*

- Configurare i repository

Sono sempre gli stessi usati nelle precedenti installazioni. Evidentemente la scelta del *mirror* è a discrezione dell'utente:

```
deb http://ftp.ch.debian.org/debian/ unstable main
deb http://ftp.debian-ports.org/debian unreleased main
```

- Configurare di default la console di Hurd

Andando ad abilitare i seguenti elementi in */etc/default/hurd-console*:

```
ENABLE='false' --> da impostare 'true'
KBD_REPEAT='--repeat=kbd' --> da decommentare
MOUSE='-d pc_mouse --protocol=ps/2' --> da decommentare
MOUSE_REPEAT='--repeat=mouse' --> da decommentare
SPEAKER='-d generic_speaker' --> da decommentare
```

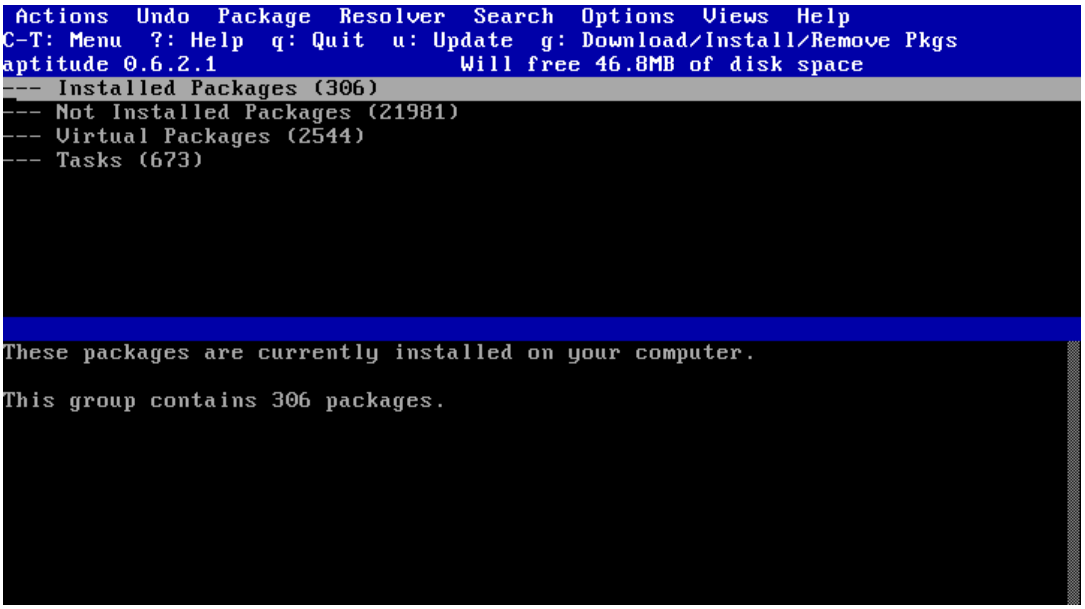
- Aggiornare il sistema

A questo punto abbiamo tutto il necessario e possiamo aggiornare il sistema con il classico:

```
# apt-get update && apt-get dist-upgrade
```

- Installare aptitude

Nonostante abbastanza lento e forse ancora poco adatto ad amministrare questo sistema, aptitude - a differenza delle versioni precedenti del sistema - si lascia installare senza problemi:




```
# apt-get install aptitude
```

- Installare X

Dopo vari tentativi a vuoto è attualmente possibile installare il fatidico server grafico X. Per fare ciò basta un semplice:

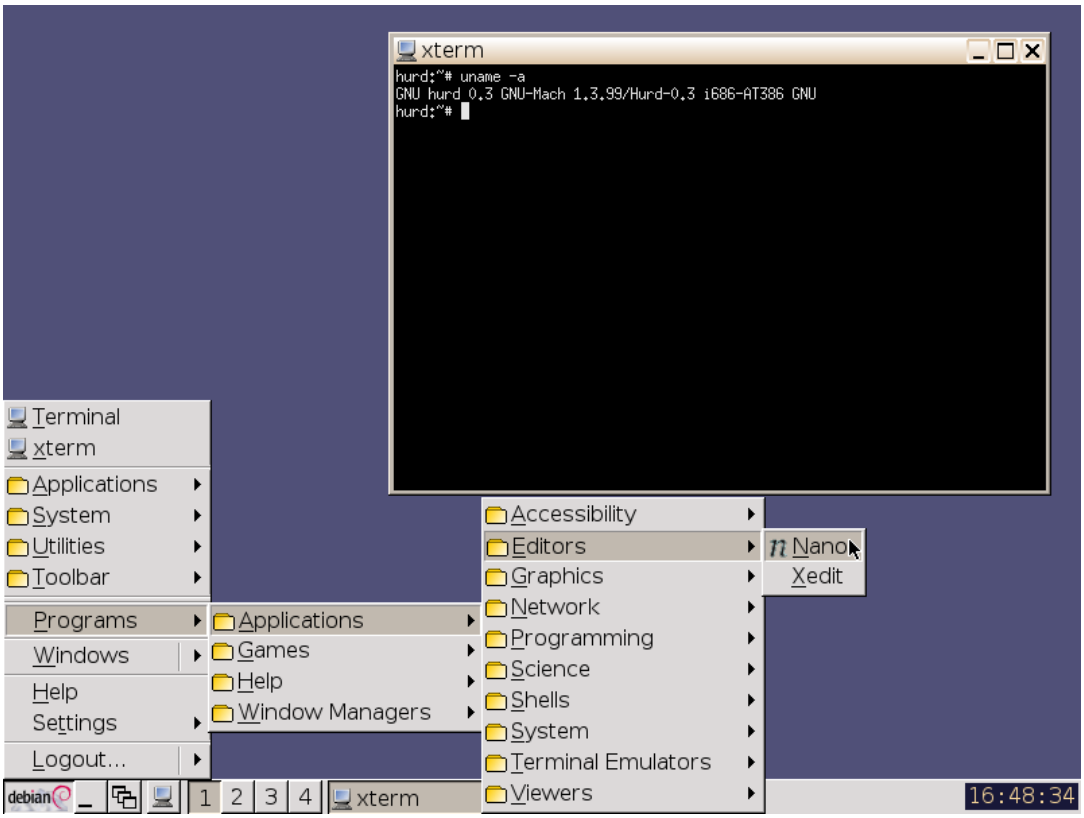
```
# apt-get install xorg
```

A questo punto basta installare un *window manager* qualsiasi (preferibilmente qualcosa di molto leggero, nell'esempio sottostante abbiamo utilizzato *icewm*) ed infine avviare il server con:

```
# startx
```

Al momento dei test, il server grafico fallisce l'avvio a causa di un bug[4]; sarà possibile ovviare al problema andando ad eliminare l'opzione MatchIsMouse dal file */usr/share/X11-/xorg.conf.d/10-mouse.conf*.

[4]bug #579130 http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=579130



QEMU

Per gli utilizzatori del noto simulatore *qemu*, oltre alle impostazioni di base (v.numero 0) sarà necessario tenere presente un paio di punti obbligatori:

- almeno con la versione 12.3 (attualmente in testing) la scheda di rete predefinita è una e1000. Quest'ultima non è compatibile con Hurd; si dovrà dunque passare ad una pcnet con accesso come user per poter utilizzare la rete nel sistema operativo. Per fare ciò bisognerà aggiungere all'avvio del simulatore le opzioni -net user -net nic,model=pcnet.
- Dal momento che la RAM di base viene impostata a 128Mb consiglio vivamente di alzare il valore (ad esempio a 1Gb, in ogni caso un minimo di 256Mb) utilizzando l'opzione -m.

Per lanciare dunque Debian GNU/Hurd su ad esempio un'immagine disco *debian-hurd.img* daremo il comando:

```
$ qemu debian-hurd.img -m 1024 -net user -net nic,model=pcnet
```

Per quanto riguarda il server grafico, dal momento che qemu utilizza come scheda grafica una Cirrus ed essendo questo driver (xserver-xorg-video-cirrus) al momento non propriamente funzionante, si sono ottenuti risultati migliori utilizzando il caro vecchio driver *vesa*. Si può dunque creare un file */etc/X11/xorg.conf* (di base non presente) ed impostare una sessione Device con il Driver vesa, oppure semplicemente eliminare il driver cirrus dal sistema, il quale andrà automaticamente ad utilizzare il vesa per avviare il server grafico.

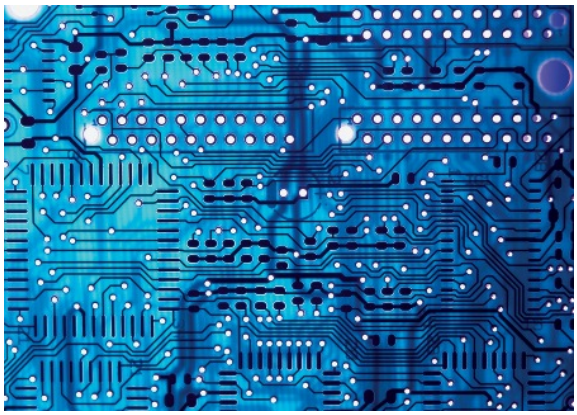
4.1.4 Conclusioni

Debian GNU/Hurd resta un sistema molto particolare, costantemente in aggiornamento. Per questo motivo, la sua installazione ex-novo è sempre una nuova storia da raccontare; a volte meglio *crosshurd* e altre volte, come attualmente, meglio i CD. Rispetto a qualche mese fa è ora possibile installare in modo relativamente semplice una moltitudine di pacchetti, fra tutti un server grafico quale Xorg. Il sistema resta purtroppo ancora lento ed instabile, così da rendere un utilizzo quotidiano tutt'ora un'utopia. È inoltre difficile stabilire se ci sarà un vero futuro per Debian GNU/Hurd; in ogni caso, resta un passatempo sicuramente interessante, al di fuori dalle regole dei classici *NIX e a differenza di forse molti altri progetti, il suo sviluppo è continuo e costante.
Mentre Debian GNU/kfreeBSD sembra oramai una vera nuova alternativa al Kernel Linux, per il Kernel Hurd (o meglio, il microkernel Mach) ci sarà ancora da lavorare. Come sempre, ulteriori informazioni si possono ottenere direttamente dalla *mailinglist* di *debian-hurd*; per il resto, alla prossima puntata ;-) !

brunitika

Capitolo 5

Hardware & Debian



Tutte le informazioni, esperienze, tuning sull'hardware e debian.
Debian, così come Linux, è una della distribuzioni più versatili adattandosi a quasi ogni tipo di interfaccia possibile.
Ogni tanto è però necessario penare un momento prima di riuscire a configurare il nostro sistema operativo per un certo tipo di hardware.
In questa sezione troverete i nostri esperimenti, così come consigli e hack.

5.1 PAM USB

Il controllo dell'accesso ai sistemi informatici è un'esigenza molto sentita in un'epoca, come la nostra, in cui alla facilità di trattamento dei dati deve corrispondere l'adozione di soluzione idonee a limitare il rischio della loro consultazione e diffusione involontaria o fraudolenta. Per contesti operativi nei quali l'utilizzo dei sistemi informatici è condiviso tra più soggetti e specialmente in aree in cui sono trattati dati di particolare delicatezza, può essere utile impiegare misure tecniche che garantiscano standard di controllo superiori rispetto a quelli comunemente adottati. In tal senso, l'evoluzione tecnologica ha messo a disposizione del responsabile della sicurezza informatica e dell'amministratore di sistema metodiche sempre più sofisticate.
Nel tempo, la tradizionale autenticazione con nome utente (*username*) e codice segreto (*password*) è stata progressivamente integrata con il riconoscimento di dati biometrici (impronte digitali, disegno dell'iride, morfologia del volto) e con la verifica del possesso di dispositivi fisici (ad esempio, tesserini con riconoscimento ottico, elettronico o in radiofrequenza).
A prescindere dalla specifica soluzione tecnologica, il principio generale adottato in ambiti che richiedano particolare attenzione nel controllo degli accessi è quello di associare almeno due modalità di autenticazione; ciò perché, in tal modo, si mitigano reciprocamente i punti deboli di ciascuna tecnica identificativa ottenendo una probabilità di accesso indesiderato più bassa rispetto a quella ottenibile con le singole tecniche.
Un aspetto da considerare in fase progettuale, inoltre, è certamente l'onere finanziario e organizzativo da sostenere per l'acquisto, l'integrazione e la manutenzione delle tecnologie di autenticazione che, nel caso di quelle più avanzate, è particolarmente elevato: per esse, infatti, si deve ricorrere a soluzioni proprietarie sia per l'hardware (brevettato) che per il software (*close source* e disponibile spesso solo per alcuni sistemi operativi).
Naturalmente, i dispositivi fisici che possono essere adoperati per l'identificazione sono i più disparati e la scelta è dettata dal rapporto desiderato dal progettista tra costi di implementazione ed efficacia ottenuta. Può sembrar strano ma anche dispositivi portatili largamente diffusi, spesso economici e standardizzati come le *pendrive* su porta USB, pur con alcuni limiti ed alcune accortezze d'uso, possono essere adoperati come componenti di una infrastruttura di autenticazione. Naturalmente, l' *hardware* è solo una parte della soluzione. I diversi sistemi operativi implementano architetture di autenticazione eterogenee, per cui è necessario, purtroppo, scrivere il *software* di autenticazione

specificamente per ciascuno di essi.
Noi appassionati di *free software* abbiamo però mille risorse e possiamo tranquillamente far fronte anche a questa particolare esigenza; nel caso di Debian GNU/Linux (come di altre distribuzioni GNU/Linux), il sistema di autenticazione si può avvalere di una libreria software chiamata *PAM* (*Pluggable Authentication Modules*). L'architettura di *PAM* permette di aggiungere di volta in volta moduli *software* specifici per la tipologia di autenticazione desiderata: *libpam-usb*, oggetto del presente articolo, appartiene a questa famiglia di soluzioni essendo un modulo *software* (in gergo tecnico, una libreria) specificamente disegnata per integrare l'uso di *pendrive* USB nell'infrastruttura di autenticazione fornita da *PAM*.

5.1.1 PAM

Cenni storici

Nel caso di sistemi unix-like, il meccanismo tradizionale di autenticazione prevede che l'utente confermi l'identità comunicando all'applicazione due dati (cosiddette credenziali): il nome utente (*username*) ed un codice segreto (*password*). L'utente è riconosciuto se la corrispondenza tra le credenziali trova conferma con quanto noto al sistema di autenticazione (ad esempio, quanto configurato nel *file* `/etc/passwd` oppure `/etc/shadow`). Una volta autenticato, l'utente può, nell'implementazione più semplice, accedere alle risorse del sistema associate al suo identificativo personale (*user ID* o UID) o all'identificativo del/dei gruppi di utenti (*group ID* o *GID*) cui appartiene; tale permesso è poi ulteriormente modulabile in funzione delle autorizzazioni per le attività di lettura, scrittura ed esecuzione di *file*.
Il meccanismo di autenticazione sopra indicato, variamente modificabile ed integrabile, è, naturalmente, quello più diffuso, ma non l'unico possibile. Ad esempio, un'azienda potrebbe volerlo integrare con altre e differenti tecnologie (come il riconoscimento delle impronte digitali oppure il contestuale utilizzo di un dispositivo univocamente identificabile). In tal caso però, se le istruzioni per implementare uno specifico meccanismo di autenticazione fossero inserite dal programmatore permanentemente all'interno delle applicazioni, allora ogni variazione del metodo di riconoscimento richiederebbe la riscrittura dei programmi interessati; in effetti, era quello che accadeva fino a non molto tempo fa e, in alcuni casi, accade ancora oggi.

Architettura

Per far fronte all'esigenza di rendere la scrittura delle applicazioni indipendenti dallo schema di autenticazione, nel 1995 la Open Software Foundation [1] propone uno standard che separa l'implementazione a basso livello dei singoli metodi di autenticazione dall'implementazione del programma che richiede tale servizio; ciò è realizzato attraverso la standardizzazione di uno strato *software* intermedio (in gergo tecnico definito *application program interface* o *API*) posto tra applicazione e metodo di autenticazione: nasce così il *Pluggable Authentication Modules* o *PAM*. In altri termini, ricorrendo a *PAM*, l'amministratore di sistema può scegliere il metodo utilizzato dalle applicazioni per autenticare gli utenti senza doverle ogni volta riscrivere. Resta inteso che l'opportunità di utilizzare *PAM* è facoltativa: il programmatore può comunque realizzare applicazioni che ne facciano a meno.
Per verificare se un'applicazione fa uso di *PAM* si può controllare se essa utilizza la libreria *software* chiamata `libpam.so`; ciò può essere ottenuto con il comando `/usr/bin/ldd` che, ad esempio riferito al comando `/bin/su`, potrà essere così impartito:

```
$ ldd /bin/su | grep libpam.so
```

che restituirà, in caso di esito positivo:

```
libpam.so.0 => /lib/libpam.so.0 (0xb7f15000)
```

PAM, inoltre, fornisce ulteriori funzionalità all'amministratore di sistema per modulare finemente la configurazione del sistema di autenticazione: per un dettagliato approfondimento di tale materia, che esula dalla trattazione del presente articolo, è possibile far riferimento alla documentazione ufficiale ed, in particolare, alla *The System Administrators' Guide*[2]. In questa sede, quindi, ci limiteremo a considerare solo quegli aspetti della configurazione funzionali alle finalità dell'articolo.
La configurazione specifica di *pam_usb* è contenuta nel *file* `/etc/pam.conf`, mentre la configurazione di *PAM* è contenuta nei *file* presenti nella *directory* `/etc/pam.d` di cui si riporta il contenuto dopo aver eseguito l'installazione di *libpam*:

[1] http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules
[2] http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html

```
$ ls /etc/pam.d/ -la
totale 104
drwxr-xr-x  2 root root 4096 12 apr 23:45 .
drwxr-xr-x 91 root root 4096 17 apr 12:25 ..
-rw-r--r--  1 root root  182 20 ott  2008 atd
-rw-r--r--  1 root root  384 14 nov 15:36 chfn
-rw-r--r--  1 root root  581 14 nov 15:36 chsh
-rw-r--r--  1 root root  392 13 mar 14:40 common-account
-rw-r--r--  1 root root  497 12 apr 23:45 common-auth
-rw-r--r--  1 root root  565 12 apr 21:56 common-auth.additional
-rw-r--r--  1 root root  574 12 apr 21:55 common-auth.alternative
-rw-r--r--  1 root root  436 12 apr 21:53 common-auth.old
-rw-r--r--  1 root root  610 12 apr 21:54 common-auth.unique
-rw-r--r--  1 root root 1212 13 mar 14:40 common-password
-rw-r--r--  1 root root  372 13 mar 14:40 common-session
-rw-r--r--  1 root root  289 28 set  2008 cron
-rw-r--r--  1 root root  267  3 set  2008 cvs
-rw-r--r--  1 root root  164  3 dic  2008 kcheckpass
-rw-r--r--  1 root root  345  3 dic  2008 kdm
-rw-r--r--  1 root root  389  3 dic  2008 kdm-np
-rw-r--r--  1 root root  168  3 dic  2008 kscreensaver
-rw-r--r--  1 root root 3217 14 nov 15:36 login
-rw-r--r--  1 root root  520 18 mar  2009 other
-rw-r--r--  1 root root   92 14 nov 15:36 passwd
-rw-r--r--  1 root root  168 28 nov  2008 ppp
-rw-r--r--  1 root root 2305 14 nov 15:36 su
-rw-r--r--  1 root root  108 14 dic 18:54 xscreensaver
```

Come si può notare sono presenti alcuni *file* i cui nomi ricordano quelli di specifiche applicazioni (ad esempio, *gdm*, *su*, *login*, etc.): essi contengono configurazioni specifiche per lo schema di autenticazione adoperato da PAM per ciascuno di essi e richiamano, al loro interno, le librerie di *PAM* (presenti solitamente in */lib/security* o in */lib64/security*) che svolgono il lavoro vero e proprio secondo lo schema di autenticazione previsto.
Il file `common-auth`, invece, contiene la configurazione relativa alla logica di autenticazione comune a tutti i programmi che utilizzano *PAM*, come avremo modo di approfondire nel proseguo.

5.1.2 pam_usb

pam_usb[3] è il nome del progetto all'interno del quale è sviluppata `libpam`, ovvero la libreria *software* che permette di aggiungere ulteriori funzionalità per l'autenticazione avvalendosi di comuni *pendrive* o altre periferiche di memoria di massa purché riconosciute dal sistema operativo come dispositivi collegati al bus USB. Poiché `libpam` è un modulo realizzato per *PAM*, esso si integra con le tutte le applicazioni (ad esempio, `/bin/su`) e i *login manager* (ad esempio, *gnome desktop manager* oppure *kde desktop manager*) che supportano tale architettura.

Finalità dell'articolo

Avvalendosi di una *pendrive* come dispositivo identificativo, si intende illustrare la realizzazione con *pam_usb* di tre schemi di autenticazione :

1. **unique**: l'autenticazione ha successo quando il nome utente (*username*) corrisponde alla *pendrive* collegata al sistema; non è mai richiesto l'inserimento del codice segreto (*password*);

2. **alternative**: l'autenticazione ha successo quando il nome utente (*username*) corrisponde al codice segreto (*password*) fornito (anche se la *pendrive* non è collegata); se, viceversa, la *pendrive* è collegata, non è richiesto l'inserimento del codice segreto (*password*);

3. **additional** : l'autenticazione ha successo quando il nome utente (*username*) corrisponde alla *pendrive* collegata al sistema e, contemporaneamente, il nome utente corrisponde alla codice segreto (*password*, che è chiesto obbligatoriamente); Inoltre, a scopo didattico, il sistema sarà configurato in modo che, una volta terminata con successo l'autenticazione, la successiva rimozione della *pendrive* attivi in automatico lo *screen saver* così come l'eventuale ulteriore re-inserimento ne causi la disattivazione.

Identificazione del dispositivo

Una *pendrive* può essere identificata in base ad alcuni dati (generalmente non modificabili) impostati dal produttore ed, in particolare: * codice e descrizione del produttore; *

[3] http://sourceforge.net/projects/pamusb/

codice e descrizione del prodotto; * numero seriale; * identificativo universale della partizione (*Universally Unique Identifier* o *UUID*). Si riporta, di seguito, un esempio ottenuto impartendo come amministratore di sistema (utente *root*) il comando:

```
# lsusb -v
```

che genera il seguente output:

```
[.. omissis.. ]
Bus 002 Device 002: ID 13fe:1f00 Kingston Technology Company Inc.
DataTraveler 2.0 4GB Flash Drive
[.. omissis ..]

Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x13fe Kingston Technology Company Inc.
  idProduct          0x1f00 DataTraveler 2.0 4GB Flash Drive [..omissis..]
  bcdDevice            1.10
  iManufacturer           1 Kingston
  iProduct                2 DataTraveler 2.0
  iSerial                 3 5B8509000178
[... omissis ...]
```

pam_usb utilizza il numero seriale (*iSerial*), la descrizione del produttore (*iManufacturer*) e la descrizione del prodotto (*iProduct*): tali dati saranno utili successivamente in fase di configurazione.
Ciò non di meno, questi elementi potrebbero essere considerati insufficienti per identificare univocamente la periferica; potrebbe essere sufficiente acquistare un secondo prodotto identico per superare il controllo (soprattutto se il numero seriale non è opportunamente valorizzato dal produttore).

Per migliorare, quindi, l'identificabilità della periferica, *pam_usb* genera una sequenza numerica casuale dopo ogni autenticazione terminata con successo e la registra sia nella *home directory* dell'utente interessato (nella *directory* `$HOME/.pamusb/`) che nella *pendrive* (sempre in una *directory* denominata `.pamusb/`). Tale sequenza è utilizzata come parte della successiva autenticazione a seguito della quale, in caso di successo, ne sarà generata una nuova: per tale motivo, tale sequenza è chiamata *one time pad* (è usata una volta sola).

Installazione

La libreria `libpam` è presente nei repository di Debian nel pacchetto `libpam-usb`. I comandi di seguito riportati devono essere eseguiti da una finestra di terminale di un computer dove GNU/Debian è già stato installato. Tale computer, inoltre, deve essere già stato configurato per eseguire il prelevamento dei pacchetti dei programmi dai *repository* della distribuzione attraverso un collegamento internet oppure da un *repository* locale (se presente).
Negli esempi di seguito riportati, inoltre, i comandi impartiti con i privilegi di utente ordinario sono preceduti dal carattere '$', mentre quelli impartiti come amministratore di sistema (utente *root*) sono preceduti dal carattere '#'; si è, inoltre, fatto ricorso al comando `su` (super user) per far acquisire temporaneamente i privilegi di accesso dell'utente *root* all'utente ordinario: è bene precisare che tale comando chiederà l'inserimento della password dell'amministratore di sistema.
Prima di procedere all'installazione è preferibile verificare lo stato di aggiornamento dei programmi installati nel sistema:

```
$ su -c "aptitude update"
$ su -c "aptitude safe-upgrade"
```

Se i comandi sopra indicati sono stati eseguiti con successo (senza errori) è possibile procedere con il comando successivo:

```
$ su -c "aptitude install libpam-usb pamusb-tools"
```

Per chi desiderasse ricompilare manualmente il codice sorgente di *pam_usb*, esso è disponibile nel sito principale di sviluppo:

1. come *file* `.tar.gz` all'indirizzo

   http://sourceforge.net/project/showfiles.php?group_id=127530

2. come snapshot dal *repository subversion* (dopo aver installato `subversion`) prelevabile con il comando:

```
$ svn co https://pamusb.svn.sourceforge.net/svnroot/pamusb/trunk/pam_usb
```

In entrambi i casi, una volta scaricato il codice sorgente e prima di procedere oltre, è necessario installare librerie e programmi richiesti; a tal fine, è possibile impartire i comandi:

```
$ su -c "aptitude install libpam-dev libxml2-dev pkg-config libhal-dev           \
                                                              libhal-storage-dev"
$ su -c "aptitude install pmount python python-celementtree python-gobject"
```

Quindi, nel caso al punto 1) (dove al posto di `<version>` sarà indicata la versione prelevata) si procederà alla compilazione ed installazione con i comandi:

```
$ tar xvfz pam_usb-<version>.tar.gz
$ cd pam_usb-<version>
$ make
$ su
password:
# make install
```

mentre nel caso al punto 2) si procederà alla compilazione ed installazione con i comandi:

```
$ cd pam_usb
$ make
$ su
password:
# make install
```

Resta, naturalmente, inteso che la ricompilazione potrà essere effettuata anche a partire dal pacchetto contenente il codice sorgente presente nei *repository* di Debian GNU/Linux.

Configurazione

La configurazione di *pam_usb*, come precedentemente accennato, è contenuta nel *file* in formato *XML* denominato `/etc/pamusb.conf` e segue la sintassi descritta in dettaglio nel *file* `/usr/share/doc/libpam-usb/CONFIGURATION.gz`. Sebbene eseguire manualmente la configurazione sia certamente possibile, sono disponibili programmi di supporto che semplificano ed automatizzano questo compito: `pamusb-conf` è uno di essi. Per ottenere maggiori dettagli sulla sua sintassi, è possibile consultare il manuale di sistema (comando `man`) oltre che visionare la documentazione nel percorso `/usr/share/doc/pamusb-tools/`.

- Configurazione dei dispositivi

Per istruire *pam_usb* a riconoscere i dispositivi che desideriamo utilizzare, è necessario prima di tutto attribuirgli un nome simbolico ed acquisirne i dati identificativi; a tal fine, è possibile impartire come amministratore di sistema (utente *root*) il seguente comando :

```
# pamusb-conf --add-device Kingston
```

dove l'argomento dell'opzione `-add-device` è il nome simbolico (anche di pura fantasia, `Kingston` nell'esempio) con cui quel dispositivo sarà successivamente riconosciuto da `pamusb-conf` e da *PAM* nell'interazione con l'utente. Impartendo il comando sopra indicato, quindi, il programma propone un elenco delle memorie di massa collegate in quel momento alle porte USB, dando la possibilità all'amministratore di sistema di sceglierne una. Qualora individui la presenza di una sola periferica, l'elenco conterrà solo essa e non sarà mostrata la richiesta di selezione. Di seguito, ad esempio, è riportato il risultato del comando sopra indicato nel caso di una sola periferica collegata:

```
Please select the device you wish to add.
* Using "Kingston DataTraveler 2.0 (Kingston_DataTraveler_2.0_5B7309B3C734-0:0)"
* (only option)

Which volume would you like to use for storing data ?
* Using "/dev/sda1 (UUID: A878-C033)" (only option)

Name : Kingston
```

```
Vendor          : Kingston
Model : DataTraveler 2.0
Serial : Kingston_DataTraveler_2.0_5B73XYZKW734-0:0
UUID : A878-C033

Save to /etc/pamusb.conf ?
[Y/n] Y
Done.
debian:~#
```

Sia il nome simbolico che gli altri identificativi indicati nell'output del comando, sono registrati all'interno del *file* `/etc/pamusb.conf`. Tale procedimento deve essere ripetuto per tutti i dispositivi (quindi, anche più d'uno) che si intendono utilizzare.

- Configurazione degli utenti

Per istruire *pam_usb* a riconoscere i singoli utenti ed associarne l'autenticazione ad uno specifico dispositivo tra quelli già configurati secondo quanto indicato nel paragrafo precedente, è possibile impartire come amministratore di sistema (utente *root*) il comando :

```
# pamusb-conf --add-user testuser
```

dove l'argomento dell'opzione `-add-user` è, in questo caso, il nome dell'utente *testuser*. In tal caso, si ottiene il seguente risultato:

```
Which device would you like to use for authentication ?
* Using "Kingston" (only option)

User            : testuser
Device          : Kingston

Save to /etc/pamusb.conf ?
[Y/n] Y
Done.
```

Come si può notare, il sistema riferisce che l'utente *testuser* è stato associato all'unico dispositivo al momento configurato per l'uso con *pam_usb*.
Tale procedimento deve essere ripetuto per tutti gli utenti che si desidera autenticare tramite *pam_usb* (quindi, anche più d'uno); ad esempio, per l'utente *root*, è possibile impartire il comando :

```
# pamusb-conf --add-user root
```

che genera il seguente risultato:

```
Which device would you like to use for authentication ?
* Using "Kingston" (only option)

User : root
Device         : Kingston

Save to /etc/pamusb.conf ?
[Y/n] Y
Done.
```

In tal modo, il *file* `/etc/pamusb.conf` è popolato con i dati identificativi degli utenti denominati *testuser* e *root* oltre che con i dati relativi alla associazione con il dispositivo la cui identificazione sarà usata per confermare la loro autenticazione.

- Configurazione schema di autenticazione generale

La configurazione generale dello schema di autenticazione attuato da *PAM* è contenuta nel *file* `/etc/pam.d/common-auth` che ha il seguente contenuto predefinito (le righe precedute dal carattere '#' sono commenti):

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
```

```
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    required        pam_unix.so nullok_secure
```

La sua configurazione ha effetto su tutte le applicazioni che utilizzano *PAM*. Di seguito si riporta il contenuto di tale *file* per ottenere ciascuna configurazione; è utile, inoltre, ricordare che le configurazioni di seguito indicate sono mutualmente esclusive e che, tranne il caso in cui sia attivato un *auto-login*, l'applicazione interessata chiederà sempre il nome utente (*username*) .

Unique Mode

L'autenticazione ha successo quando la pendrive è univocamente identificata in relazione all'utente; non è richiesto l'inserimento del codice segreto (password). Il *file* `/etc/pam.d/common-auth` dovrà contenere:

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    required        pam_usb.so
```

Alternative Mode

L'autenticazione ha successo quando il nome utente (username) corrisponde al codice segreto (password) fornito (anche in assenza della pendrive); se, viceversa, la pendrive è collegata, non è richiesto l'inserimento del codice segreto (password). Il *file* `/etc/pam.d/common-auth` dovrà contenere:

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
```

```
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    sufficient      pam_usb.so
auth required        pam_unix.so nullok_secure
```

Qualora il *display manager*[4] sia configurato per eseguire l' *auto-login*, basterà inserire la *pendrive* prima di avviare il computer e *PAM*, in automatico, effettuerà il *login* all'interno dell'ambiente grafico per l'utente di default.

Additional Mode

L'autenticazione ha successo quando la pendrive è univocamente identificata in relazione all'utente e, contestualmente, il nome utente corrisponde al codice segreto (password); il *file* `/etc/pam.d/common-auth` dovrà contenere:

```
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    required  pam_usb.so
auth required  pam_unix.so nullok_secure
```

Resta inteso che questo è lo schema di autenticazione che offre il maggior livello di sicurezza in quanto sono applicati contestualmente due diversi metodi di identificazione dell'utente.

[4] http://en.wikipedia.org/wiki/GNOME_Display_Manager
http://it.wikipedia.org/wiki/KDE_Display_Manager

Verifica configurazione generale
Per eseguire la verifica della configurazione è possibile usare il comando `pamusb-check`. Esso accetta come parametro il nome utente (*username*) di cui si intende simulare l'autenticazione con 'PAM': naturalmente, durante la verifica, dovrà essere collegata la *pendrive* che, in fase di configurazione, è stata associata a tale nome utente. Ad esempio, per verificare l'utente *root*, è possibile impartire il comando:

```
# pamusb-check root
```

ottenendo il seguente risultato positivo:

```
* Authentication request for user "root" (pamusb-check)
* Device "Kingston" is connected (good).
* Performing one time pad verification...
* Regenerating new pads...
* Access granted.
```

mentre per l'utenza *testuser*, è possibile impartire il comando:

```
# pamusb-check testuser
```

ottenendo il seguente risultato, anch'esso positivo:

```
* Authentication request for user "testuser" (pamusb-check)
* Device "Kingston" is connected (good).
* Performing one time pad verification...
* Regenerating new pads...
* Access granted.
```

Volendo fornire un esempio di esito negativo, possiamo impartire lo stesso comando per un utente inesistente:

```
# pamusb-check utente_inesistente
```

ottenendo il seguente risultato, stavolta negativo:

```
* No device configured for user "utente_inesistente".
```

oppure, dopo aver disconnesso la *pendrive*, possiamo impartire il comando:

```
# pamusb-check testuser
```

ottenendo il seguente risultato, ancora una volta negativo:

```
* Authentication request for user "testuser" (pamusb-check)
* Device "Kingston" is not connected.
* Access denied.
```

Una volta terminata la verifica delle utenze configurate con esito positivo, è possibile passare alla configurazione dello schema di autenticazione desiderato.

- Configurazione per specifiche applicazioni

In alternativa alla configurazione dello schema generale di autenticazione (che ha effetto indifferentemente su tutte le applicazioni che utilizzano *PAM*), è possibile integrare le funzioni di *pam_usb* limitatamente a una o più specifiche applicazioni. A tal fine, sarà necessario riportare, se modificato, il contenuto di `/etc/pamusb.conf` al proprio contenuto originario, quindi modificare il *file* di configurazione specifico per l'applicazione di nostro interesse (contenuto nella *directory* `/etc/pam.d`).
Ad esempio, lo schema di autenticazione di *PAM* specifico per *gdm* (*gnome display manager*) è contenuto nel *file* `/etc/pam.d/gdm` il cui contenuto predefinito è:

```
#%PAM-1.0
auth       required     pam_env.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
```

```
session    required     pam_stack.so service=system-auth
session    optional     pam_console.so
```

Per abilitare l'uso di *pam_usb* solo per *gdm* secondo, ad esempio, lo schema *alternative mode*, è necessario modificare `/etc/pam.d/gdm` come di seguito indicato:

```
#%PAM-1.0
auth       sufficient pam_usb.so
auth       required     pam_env.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
session    optional     pam_console.so
```

Invece, lo schema di autenticazione di *PAM* specifico per *kdm* (*kde display manager*) è contenuto nel *file* `/etc/pam.d/kdm` il cui contenuto predefinito è:

```
#
# /etc/pam.d/kdm - specify the PAM behaviour of kdm
#
auth       required     pam_nologin.so
auth       required     pam_env.so readenv=1
auth       required     pam_env.so readenv=1 envfile=/etc/default/locale
@include common-auth
session    required     pam_limits.so
@include common-account
@include common-password
@include common-session
```

Per abilitare l'uso di *pam_usb* specificamente per *kdm* secondo, ad esempio, lo schema *alternative mode*, è necessario modificare `/etc/pam.d/kdm` come di seguito indicato:

```
#
```

```
# /etc/pam.d/kdm - specify the PAM behaviour of kdm
#
auth       sufficient   pam_usb.so
auth       required     pam_nologin.so
auth       required     pam_env.so readenv=1
auth       required     pam_env.so readenv=1 envfile=/etc/default/locale
@include common-auth
session    required     pam_limits.so
@include common-account
@include common-password
@include common-session
```

Note

* se per qualunque motivo si dovesse decidere di non configurare l'utente *root* per *pam_usb*, adottando gli schemi *unique mode* e *additional mode*, tale utenza si vedrà precluso qualunque accesso; l'unica soluzione, in tal caso, per permettere all'utente root di eseguire l'accesso al sistema è quella di avviarlo in *single-user mode*;
* nel caso in cui si volesse abilitare il *debugging* bisognerà aggiungere l'opzione `debug=1` alle righe relative al modulo *pam_usb.so* nel *file* `/etc/pam.d/common-auth`, ad esempio:

```
 auth    required  pam_usb.so    debug=1
```

In tal caso, i messaggi diagnostici saranno registrati nel *file* `/var/log/auth.log`;
* se invece volessimo implementare uno schema di autenticazione che effettui l' *autologin* in `gnome` a patto che la *pendrive* sia collegata, sarà sufficiente apportare una piccola modifica al *file* `/etc/pam.d/gdm-autologin`, sostituendo la riga:

```
auth required pam_permit.so
```

con la riga:

```
auth sufficient pam_usb.so
```

Interazione con altre applicazioni

Per far in modo che *PAM* interagisca con altre applicazioni (come, ad esempio, lo *screen saver*) al verificarsi di determinati eventi generati dal sistema di autenticazione, si può usare il programma `pamusb-agent` configurando manualmente le sezioni relative agli utenti direttamente nel file `/etc/pam.conf` di modo che risultino simili a questa:

```
<users>
<user id="testuser">
<device>Kingston</device>
<option name="quiet">true</option>
<agent event="lock">gnome-screensaver-command --lock</agent>
<agent event="unlock">gnome-screensaver-command --deactivate</agent>
</user>
```

In questo modo, scollegando la *pendrive* dal sistema, lo *screen saver* di `gnome` sarà attivato; re-inserendo la *pendrive*, invece, esso sarà disattivato.
Per configurare *KDE*, è possibile seguire un analogo procedimento, avendo l'accortezza di specificare l'applicazione che gestisce le funzioni di *screen saver* per questo *desktop environment*, come di seguito indicato:

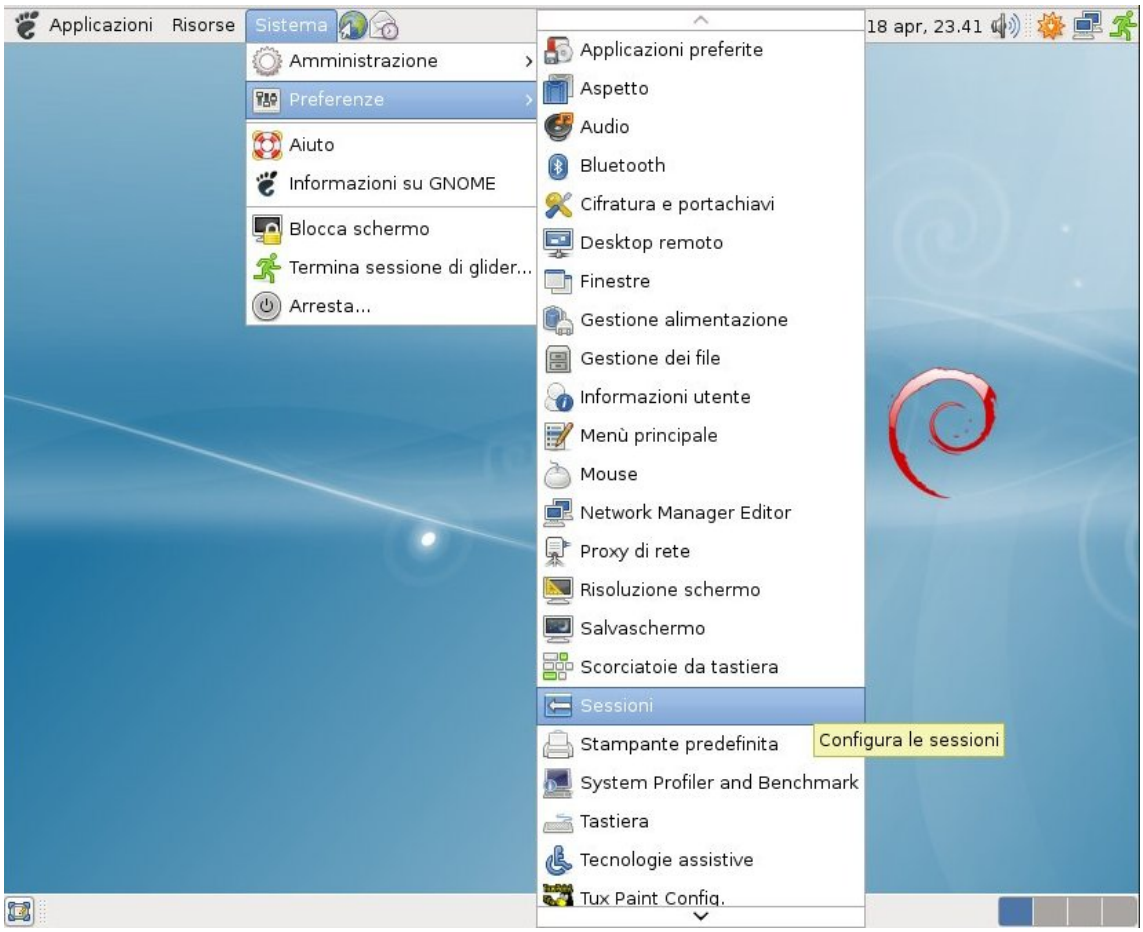
```
<users>
<user id="testuser">
<device>Kingston</device>
<option name="quiet">true</option>
<agent event="lock">dcop kdesktop KScreensaverIface lock</agent>
<agent event="unlock">dcop kdesktop KScreensaverIface quit</agent>
</user>
```

È importante ricordare che, affinché questa funzionalità sia attiva, `pamusb-agent` deve essere inserito nella lista delle applicazioni attivate automaticamente all'avvio del *desktop environment*. Ad esempio, per `gnome`, tale configurazione può essere eseguita dal menù di sistema:

```
Sistema -> Preferenze -> Sessione ..
      Seleziona programmi d'avvio (click su "Aggiungi" e aggiunta di pamusb-agent)
```

come da immagine di seguito indicata:
mentre, per `kde`, è possibile farlo impartendo il seguente comando:

```
$ ln -s /usr/bin/pamusb-agent ~/.kde/Autostart/pamusb-agent
```



5.1.3 Conclusioni

Abbiamo visto come sia facile con una spesa minima e poche configurazioni, integrare l'identificazione di un dispositivo (*pendrive* su porta USB) nell'architettura di autenticazione di Debian GNU/Linux. Nessun problema anche in caso di smarrimento o furto del dispositivo usato per l'autenticazione: è sufficiente riconfigurare `/etc/pam.conf` per sostituirne i dati identificativi con quelli del dispositivo in sostituzione.

pmate, Aki

Capitolo 6

Tips & Tricks

Che aggiungere, soffiate e trucchi ;-)
Tutto ciò di utile che non rientra nelle altre categorie.
In questa sezione troverete articoli che riguardano questioni settoriali e/o molto specifiche su di un argomento in particolare.

6.1 PGP: configurazione e utilizzo in Debian

Pretty Good Privacy (PGP) è un software crittografico estremamente diffuso e utilizzato in svariati ambiti.
Grazie ad esso è possibile cifrare documenti, directory, porzioni di disco rigido o di supporti removibili e può essere utilizzato per l'invio e la ricezione di email in modo sicuro.
Nasce da un'idea di Phil Zimmermann che volle implementare un protocollo di cifratura sicuro e allo stesso tempo di libera distribuzione.
Oggi, PGP è il software crittografico in assoluto più utilizzato.
Perché esso possa essere compreso appieno bisognerebbe possedere conoscenze matematiche non banali (complessità degli algoritmi, teoria dei numeri primi, etc.): in questo articolo, l'argomento sarà affrontato in maniera molto più semplice così da renderlo facilmente comprensibile anche a chi non ha quel tipo di preparazione teorica.
PGP è un software proprietario.
Nel 1999 si decise di passare alla realizzazione di un programma molto simile e compatibile con esso, utilizzando il modello di sviluppo Open Source. Nacque così in Germania il programma "GPG" acronimo di "GNU Privacy Guard", creato da Werner Koch: egli riuscì a realizzare un software del tutto compatibile con gli standard PGP oltre che con i sistemi operativi proprietari.
Presente dal 2000 nei sistemi *UNIX e BSD, GPG è diventato un must nella crittografia in ambienti di sviluppo UNIX like. L'utilizzo di GPG è identico a quello di PGP.

6.1.1 Cenni di funzionamento

PGP usa sia la crittografia simmetrica che quella asimmetrica (detta anche a chiave pubblica o RSA - acronimo di Rivest, Shamir e Adleman, che per primi ne pubblicarono le specifiche).
Il sistema a chiave simmetrica prevede che l'algoritmo di decodifica utilizzi la stessa chiave privata per cifrare e decifrare i messaggi da proteggere
Il sistema a chiave asimmetrica, invece, prevede l'utilizzo di una coppia di chiavi: pubblica e privata. Con la chiave pubblica del destinatario si cifra un messaggio. Il destinatario lo decifrerà con la propria chiave privata.
Questi concetti verranno ripresi ed ampliati nel corso dell'articolo.

6.1.2 Creazione delle chiavi

Il programma che useremo per creare le nostre chiavi è GnuPGP, presente nei repository di Debian.
Apriamo il nostro terminale e installiamolo

```
# apt-get install gnupg
```

Procediamo poi con la creazione della chiavi.
Sempre da terminale, ad installazione completata ma come utente comune, digitiamo:

```
$ gpg --gen-key
```

per accedere al pannello di creazione delle chiavi. Davanti a noi si presenteranno 4 opzioni:

```
Per favore scegli che tipo di chiave vuoi:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (firma solo)
   (4) RSA (firma solo)
```

Scegliamo l'opzione (1) per la creazione delle chiavi RSA. Successivamente, ci verrà richiesta la lunghezza della chiave da creare: l'inserimento di 1024 può andare più che bene; ovviamente maggiore è la lunghezza, maggiore sarà la sicurezza ma i tempi di cifratura e decifrazione saranno più lunghi.

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 1024
```

Dopo aver inserito la lunghezza della chiave RSA, procediamo premendo "invio". Nella schermata successiva, ci verrà chiesto per quanti giorni la chiave deve valere: le opzioni sono molte e noi sceglieremo la numero 0, ovvero nessuna scadenza:

```
Per favore specifica per quanto tempo la chiave sarà valida.
         0 = la chiave non scadrà
       <n>  = la chiave scadrà dopo n giorni
       <n>w = la chiave scadrà dopo n settimane
       <n>m = la chiave scadrà dopo n mesi
       <n>y = la chiave scadrà dopo n anni
Chiave valida per? (0) 0
```

Il sistema ci chiederà quindi di confermare la scelta effettuata:

```
Key does not expire at all
Is this correct? (y/N) y
```

Premendo ”invio”, giungeremo alla compilazione dei dati che precede la creazione della chiave pubblica e di quella privata.

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

Dovremo a questo punto inserire il nostro Nome e Cognome, un indirizzo email valido e un commento (facoltativo):

```
Nome e Cognome:
Indirizzo di Email:
Commento:
```

Una volta compilati i campi e data la conferma, ci verrà mostrato il riepilogo delle informazioni inserite con la possibilità di modificarle, di cancellarle o di accettarle:

```
Hai selezionato questo User Id:

Modifica (N)ome, (C)ommento, (E)mail oppure (O)kay/(Q)uit?
```

Ci verrà chiesto di inserire una passphrase che servirà per la protezione della chiave. Questo è un passaggio importante: bisogna cercare di scegliere una buona passphrase e di non dimenticarla.
Confermata la passphrase, comparirà una richiesta di esecuzione di altre operazioni per la generazione dei numeri primi (la creazione di RSA si basa proprio su questi numeri per aumentare l'entropia), per cui, quando leggeremo queste righe:

```
Dobbiamo generare un mucchio di byte casuali. È una buona idea eseguire
qualche altra azione (scrivere sulla tastiera, muovere il mouse, usare i
dischi) durante la generazione dei numeri primi; questo da al generatore di
numeri casuali migliori possibilità di raccogliere abbastanza entropia.
```

riduciamo il terminale e cominciamo a fare qualsiasi altra cosa: aprire documenti, digitare lettere a caso, giocare e ascoltare musica. L'importante è far lavorare il processore: nel caso ciò non accadesse, sullo schermo comparirà un avviso:

```
Non ci sono abbastanza byte casuali disponibili. Per favore fai qualche
altra cosa per dare all'OS la possibilità di raccogliere altra entropia!
(Servono altri 284 byte)
```

proseguiamo a lavorare, e quando i byte saranno stati accumulati, il computer automaticamente genererà le chiavi.
A lavoro completato si otterrà un messaggio simile a questo:

```
gpg: /home/pincopallino/.gnupg/trustdb.gpg: creato il trustdb
gpg: key 20ACD5A1 marked as ultimately trusted
chiavi pubbliche e segrete create e firmate.
```

```
gpg: controllo il trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   1024R/20ACD5A1 2010-02-11
      Key fingerprint = 10 gruppi da 4 cifre che comporranno la "key finger print"
uid                  Nome Cognome () <email>
sub   1024R/7DB807FC 2010-02-11
```

Le chiavi sono state create con successo!

> **Nota -** La scelta della passphrase è un passaggio fondamentale nel processo di creazione della coppia di chiavi. Chiunque dovesse riuscire ad impadronirsi della chiave privata dovrebbe riuscire ad oltrepassare la cifratura della chiave privata stessa. Una buona passphrase è quella facile da ricordare ma che altri difficilmente possono indovinare o violare tramite attacco brute force. Ecco perchè essa dovrebbe essere lunga e costituita da un insieme di caratteri alfabetici (maiuscoli e minuscoli), caratteri speciali e numeri.

6.1.3 Crittografia simmetrica

Questo sistema crittografico prevede l'utilizzo di una sola chiave per la cifratura e la decifrazione.
La sua affidabilità è basata esclusivamente sulla robustezza dell'algoritmo di cifratura (di solito DES - Data Encryption Standard -).
Facciamo un esempio pratico cifrando un messaggio presente nella nostra $HOME contenuto nel file "file-segreto.txt".
Prima di tutto leggiamo le chiavi presenti nel nostro keyring identificando i dati relativi a quella creata in precedenza:

```
$ gpg --list-keys
/home/glider/.gnupg/pubring.gpg
------------------------------
pub   1024D/3231A5D0 2008-07-26
uid                  Pinco Pallino (Chiave crittografica di Pinco Pallino)      \
                                                        <pinco.pallino@gmail.com>
```

```
sub   2048g/3B74F97C 2008-07-26
...
[cut]
...
```

Procediamo quindi a cifrare il messaggio:

```
$ gpg -r pinco.pallino@gmail.com --symmetric --encrypt file-segreto.txt
```

verrà richiesto l'inserimento e la conferma di una passphrase da utilizzare al momento della decifrazione del messaggio e sarà generato il file "file-segreto.txt.gpg", ossia il file cifrato.
Per decifrarlo:

```
$ gpg -o file-decifrato.txt --decrypt file-segreto.txt.gpg
```

e dopo l'inserimento della passphrase scelta in precedenza si otterrà il file "file-decifrato.txt".
La debolezza di questo approccio risiede nella necessità di condivisione della chiave di cifratura.
Per questo motivo, oggi, è preferibile affidarsi alla crittografia asimmetrica che garantisce una sicurezza maggiore.

6.1.4 Crittografia asimmetrica

Immaginiamo di disporre di un lucchetto elettronico e di due chiavi che lo aprono: una "pubblica" e una "privata".
Metteremo a disposizione di tutti il lucchetto e la chiave pubblica, in modo tale che ogni persona che vorrà inviarci qualcosa possa chiuderla in un contenitore qualsiasi che sigillerà con il nostro lucchetto elettronico.
Una volta chiuso il tutto (ovvero, dopo aver cifrato il contenuto con PGP), il contenitore ci potrà essere inviato sicuri che nessuno, durante il tragitto, sia in condizione di aprirlo e leggerne il contenuto.

A prima vista, un messaggio cifrato è illeggibile. Anche possedendo la chiave pubblica è impossibile decifrarlo.
È la nostra chiave privata che ci permetterà di riportare in chiaro il messaggio ricevuto.
Possediamo già una coppia di chiavi, una pubblica da distribuire e una privata da tenere, è il caso di inserirle all'interno di file di testo.
La procedura è molto semplice:

```
$ gpg --export -a pinco.pallino@gmail.com > chiavepubblica.txt
```

così facendo si esporta la chiave pubblica, contenuta nel keyring ed appartenente all'utente identificato da quella particolare email, in un file chiamato chiavepubblica.txt.
Procederemo in modo simile all'esportazione della chiave privata:

```
$ gpg --export-secret-key -a pinco.pallino@gmail.com > chiaveprivata.txt
```

e la chiave verrà esportata in un file chiamato chiaveprivata.txt.
Se proviamo ad aprire il file chiavepubblica.txt ci si presenterà davanti un file di questo genere:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.10 (GNU/Linux)

mI0ES3PwoAEEAM38UqWKnbthn1RAIj0EeIBbM0FoJ/UKgL0IyQxVz6JlVcFmWXr4
WzFMp025A0MyeWPj+vwGw42TBqJnS1jwu3cCff1OxsMYuX/afSWYM2tD7ey1nIXt
bLsJyr5Zy2YS0sM80KG77gBlLWJxX6YCjTG076plVI+PeyU2HuyuPdCFABEBAAG0
NUFuZHJlYSBQb3NzZW1hdG8gKHByb3ZhKSA8YW5kcmVhLnBvc3NlbWF0b0BnbWFp
bC5jb20+iLgEEwECACIFAktz8KACGwMGCwkIBwMCBhUIAgkKCwQWAgMBAh4BAheA
AAoJELE9UXEgrNWhjnsEALJc+6TAVd6P/S7fFsnWHpPnMX0r1VYvtp8uMmr2wZNa
BLJNY4MsgWAJDMwJbakzdL2DC8ThU9ydUTcBpw2OWQW/i8CAyUkS5GjJ1iQ40tLt
WfPTbAcCIbkJuep7XkFOMMsV/nJKbFMySKUCii3KFtj7nllxLQRbyhSsRr9LJ9Mo
.....................
.....................
-----END PGP PUBLIC KEY BLOCK-----
```

Questa è la chiave pubblica che potremo consegnare a chi vogliamo e che permetterà di comunicare in modo sicuro con noi.
Facciamo un esempio pratico.
Ipotizziamo che qualcuno voglia inviarci un file cifrato.
Questa persona dovrà aggiungere la nostra chiave pubblica al proprio keyring:

```
$ gpg --import chiavepubblica.txt
```

Se l'importazione è andata a buon fine, dando

```
$ gpg --list-key
```

vedrà nella lista anche la nuova chiave appena aggiunta.
Per la cifratura proseguirà ripetendo i passaggi su indicati inserendo nel campo UID la nostra email.
Completata la cifratura, ci invierà il messaggio cifrato che noi potremo decifrare inserendo la nostra passphrase, visto che quel file è stato cifrato con la nostra chiave pubblica!

6.1.5 Repository e GPG

Per garantire che un determinato repository sia fidato in Debian si utilizza una chiave di cifratura GPG.
L'autenticazione è di forma univoca, ovvero una chiave GPG può autenticare un solo repository. Per i repository ufficiali, l'acquisizione della chiave di cifratura è automatica, infatti non andiamo mai ad aggiungere una chiave GPG al primo avvio della nostra Debian, a meno che non usiamo già dal primo avvio repository non ufficiali. Ma se volessimo usare un repository non ufficiale che richiede l'autenticazione ed essa non viene fatta in automatico? Dobbiamo importare le chiavi GPG per questi repository non ufficiali. Prendiamo l'esempio di un repository backports:

> (esempio preso da http://www.e-pillole.com/ per quanto riguarda il backports)

```
##BACKPORTS
deb http://www.backports.org/debian/ lenny-backports main contrib non-free
deb-src http://www.backports.org/debian/ lenny-backports main contrib non-free
```

apriamo con il nostro editor di testo il file `/etc/apt/sources.list` e aggiungiamo questo repository. Salviamo e chiudiamo e cominciamo con i comandi.

```
# apt-get update
```

alla fine dell'update troveremo però un WARNING di questo tipo

```
W: GPG error: http://www.backports.org lenny-backports Release: Le seguenti firme \
           non sono state verificate perché la chiave pubblica non è disponibile: \
                                                        NO_PUBKEY EA8E8B2116BA136C
W: È consigliabile eseguire apt-get update per correggere questi problemi
```

analizziamo bene questo avvertimento: il sistema ci comunica che non è stato possibile verificare la firma del repository perché manca della chiave pubblica di riferimento.
La chiave che il sistema cerca è EA8E8B2116BA136C. Preleviamo dal keyserver predisposto la chiave dei nostri backports con questo comando:

```
# gpg --keyserver subkeys.pgp.net --recv EA8E8B2116BA136C
```

Una volta scaricata la chiave sul nostro computer, dobbiamo provvedere ad aggiungerla al nostro portachiavi.
Per far ciò utilizzeremo il comando export in pipe con il comando apt-key:

```
# gpg --armor --export EA8E8B2116BA136C | apt-key add -
```

Fatto questo, daremo nuovamente l'update con

```
# apt-get update
```

e se non otterremo altri Warning vorrà dire che avremo importato la chiave in modo corretto.
Saremo parimenti certi che, da quel momento in poi, i pacchetti scaricati proverranno da una fonte sicura.

Perché?
Il meccanismo di funzionamento della ”crittografia asimmetrica” dovrebbe ormai essere chiaro: la chiave pubblica (quella conosciuta da tutti) serve a cifrare un file, per leggere il quale sarà necessaria la chiave privata (che deve essere tenuta segreta).
È però possibile usare la chiave privata per ”’firmare”’ un file e non solo per decifrarlo: in questo caso, chiunque sia in possesso di chiave pubblica potrà verificare che quel file sia stato firmato da quella chiave privata.
Ogni repository Debian contiene un file `Release` che viene aggiornato ogni volta che un pacchetto dell'archivio cambia e che contiene anche i ”checksum” di altri file sempre contenuti in quel repository. Tra questi ci sono i file ”Packages.gz”.
Questo uno stralcio del file `Release` presente nei repository di Debian Lenny:

```
Origin: Debian
Label: Debian
Suite: stable
Version: 5.0.4
Codename: lenny
Date: Fri, 29 Jan 2010 23:18:16 UTC
Architectures: alpha amd64 arm armel hppa i386 ia64 mips mipsel powerpc s390 sparc
Components: main contrib non-free
Description: Debian 5.0.4 Released 29 January 2010
MD5Sum:
 5a5774342de8498b6c39666a287277cb 13513135 Contents-powerpc.gz
 31e526616dffa65cbd000aba82049fb8 13210302 Contents-armel.gz
 a663bc33ed6e79c0256053cd30226987 13157248 Contents-alpha.gz
 d8b6b3e5097edcda8d280211fa5df4ca 13824603 Contents-i386.gz
...
[cut]
...
 dda6fb2fc2e6ed6d41a6a0786c05e5bd 24354793 main/binary-i386/Packages
 56df8601bb6c9fc91e899e88cbd6fe3d  6724599 main/binary-i386/Packages.gz     <------
 9e32cd68e56b8ae0120b2e278e5a30d3  5196154 main/binary-i386/Packages.bz2
 6f8ee24cb11bb35bfff84442b5f23c02       95 main/binary-i386/Release
...
[cut]
...
```

Ogni file "Packages.gz", parimenti, contiene un checksum per ogni pacchetto in esso listato.
Eccone uno stralcio:

```
...
[cut]
...
Package: 3dchess
Priority: optional
Section: games
Installed-Size: 100
Maintainer: Debian Games Team <pkg-games-devel@lists.alioth.debian.org>
Architecture: i386
Version: 0.8.1-15
Depends: libc6 (>= 2.7-1), libx11-6, libxext6, libxmu6, libxpm4, libxt6,         \
                                                                  xaw3dg (>= 1.5+E-1)
Filename: pool/main/3/3dchess/3dchess_0.8.1-15_i386.deb
Size: 34526
MD5sum: 4605788bdc35ee2e7ff419162f86c126
SHA1: 27415ead5f8bf401d73bd9f0fb9ea24cd727e54e
SHA256: e1231555d343e141fbd803998537032e4af78695d523a0e8f65b2f6f4d38226f
Description: 3D chess for X11
 3 dimensional Chess game for X11R6.  There are three boards, stacked
 vertically; 96 pieces of which most are the traditional chess pieces with
 just a couple of additions; 26 possible directions in which to move.  The
 AI isn't wonderful, but provides a challenging enough game to all but the
 most highly skilled players.
Tag: game::board, game::board:chess, implemented-in::c, interface::x11,           \
                role::program, uitoolkit::xlib, use::gameplaying, x11::application
...
[cut]
...
```

Mediante il confronto del checksum relativo al file "Packages.gz" contenuto nel Release file e del checksum del file "Packages.gz" contenuto nei repository, apt capisce se la copia scaricata di "Package.gz" file è corretta.

Quando poi scarica un singolo pacchetto, apt esegue il check di questo con quanto contenuto nel file "Packages.gz".
La firma crittografica del file `Release` è costituita da un file chiamato "Release.gpg" che viene fornito insieme al file `Release`.
Ecco il contenuto di http://ftp.it.debian.org/debian/dists/lenny/Release.gpg:

```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (GNU/Linux)

iQIcBAABAgAGBQJLY2zLAAoJEJqjjc1VvjAr6E0QAJMm3HWcrMd3GPUxQ/tThtNz
PsIUH5Bkd5LwHGXnRntz/pF1eeQPZ9VvwTNc0q+AP5ZP21/q3kiTFBbaqTkqXsc0
QOiF2mKMI+tv4iJaLqMI/IBEAXqwyIdDGXzuyCG50RxajKmARb4Npab2OqSz/Dy3
IMROR9wy8tzPnCtyVnkOZ7I1Vz0AF5JQYBCCDscQJu4O7i6t5itqkgF0vA/FvXJ2
GHOQPSAVOAMhd1vudVjn9bu5mP6cAT6wQoJKa/uNs5pRlotC2qN4unZKleZ2NHvU
9E7pVleKb1ND07dFPQZjDdT7Ir7B4J2Ma7yHXn8nFdPRrOHb0/fXBC5js0XgenHH
ZASRQeZnjVh8PRAZDU47bhF9DIsaxnbds0DMMLFhMDQM2vGjQlTcWRoEPm1QMLv+
ttL7aD4rrq/S413cKodqaR9TnuXjydNoja8Od/cgtH9VvMKVtBbPdhjn749Fx5Hw
b3fnwmzTbgguzsleRyF9/fJIBSuVc9TQH42GmCa7OsCB7wRHlOEdyj/4FtFfP37R
0gYTVQeeV5l9wP/cun3boWyhGSg2cXwhPKFU+I7cDW2TobQ3I/eUrodeyr/ikrPe
iLqGsUXBIWmlAC7NGUBrITQw6rouc82qnDSKR+RLPBuXZeEVRy95d+LI+CvPa0sr
TiHtlcssVR+1v/d1G9Qk
=/CSh
-----END PGP SIGNATURE-----
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)

iEYEABECAAYFAktjbk0ACgkQTScNBvQlhOYVJgCgg1ry+a9GkDe5EDGy8M5ReE64
/TIAn1WqrgX49oOr6RVLXil2roYhCaX/
=003L
-----END PGP SIGNATURE-----
```

Se apt non riesce a scaricare `Release.gpg` o se la firma non è verificata, avvisa che il "Packages.gz" al quale il file `Release` fa riferimento (e ovviamente tutti i file in esso listati) provengono da una fonte non verificata.
Il meccanismo di sicurezza si basa sulla verifica dell'esistenza e della validità di `Release.gpg`, validità accertata eseguendo il controllo di quella firma.

Per eseguire tale controllo, apt deve conoscere la chiave pubblica della persona che l'ha firmato (come detto la firma avviene per mezzo della propria chiave privata).
Tali chiavi pubbliche sono contenute nel keyring `/etc/apt/trusted.gpg`:

```
# apt-key list
/etc/apt/trusted.gpg
--------------------
pub   1024D/6070D3A1 2006-11-20 [expired: 2009-07-01]
uid                  Debian Archive Automatic Signing Key (4.0/etch)             \
                                                      <ftpmaster@debian.org>

pub   1024D/ADB11277 2006-09-17
uid                  Etch Stable Release Key <debian-release@lists.debian.org>

pub   1024D/BBE55AB3 2007-03-31 [expired: 2010-03-30]
uid                  Debian-Volatile Archive Automatic Signing Key (4.0/etch)

pub   1024D/F42584E6 2008-04-06 [expires: 2012-05-15]
uid                  Lenny Stable Release Key <debian-release@lists.debian.org>
...
[cut]
...
```

È questa, per concludere, la ratio del funzionamento di GPG: una "catena di fiducia". Le chiavi, firmate e gestite come appena esposto, sono gli unici veri garanti dell'affidabilità del repository.

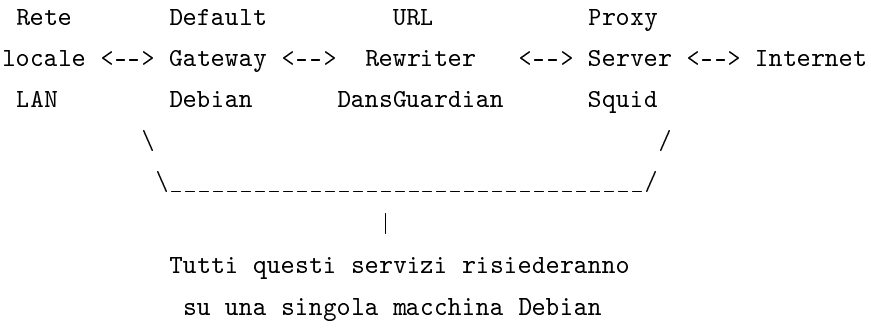
greyfox, pmate

6.2 Squid e DansGuardian: come costruire un proxy con filtro dei contenuti web

In diversi contesti di rete, si pensi ad esempio ad una rete LAN aziendale o ai laboratori informatici di scuole medie inferiori, può essere importante implementare un sistema il più possibile automatico per il controllo e il filtraggio di alcuni contenuti web considerati dall'amministratore inappropriati al contesto. Si considerino a titolo di esempio le seguenti necessità:

- proteggere i minori dal rischio di imbattersi in siti dedicati alla pornografia, alla violenza, al gioco d'azzardo e altro (il cosiddetto *parental control*);
- impedire al personale di un'azienda o di un ente di perdere tempo frequentando siti non connessi con l'attività lavorativa;
- migliorare l'utilizzo della banda disponibile, bloccando l'accesso a siti non connessi con l'attività lavorativa o la visione di filmati in streaming. Gli ingredienti che permettono di raggiungere gli obiettivi sopra elencati in una macchina Debian sono:
- un insieme di blacklist che elenchino sia i siti e le URL da bloccare o meno, sia frasi che, trovate in una pagina, facciano sì che essa sia bloccata;
- un software di filtraggio, o *url rewriter*, che, servendosi delle blacklist, possa dirottare una richiesta verso una pagina prestabilita, oppure lasciarla passare;
- un software *proxy*, che si occupi essenzialmente di ricevere le richieste dal *url rewriter* e ad inoltrarle verso l'esterno;
- le utility *iptables*, utilizzate per definire le regole che, in maniera trasparente, dirottano tutto il traffico web attraverso il proxy o il software di filtraggio, indipendentemente dalla volontà degli utenti;
- il supporto *Netfilter* nel kernel, necessario alle utility *iptables*. Per raggiungere tutti questi obiettivi è anche necessario che la macchina Debian che configureremo sia impostata come gateway predefinito della rete LAN che vogliamo sottoporre a filtraggio.

Lo schema logico di questa struttura di rete sarà quindi come il seguente:



Nei capitoli seguenti analizzeremo uno per uno tutti i punti della struttura da implementare.

6.2.1 Debian come gateway

Configurare la nostra Debian box come gateway per la rete LAN interna è il primo passo per raggiungere l'obiettivo di un firewall che filtri i contenuti indesiderati. Questo renderà inoltre la nostra rete molto più flessibile di quanto non sarebbe con l'utilizzo di un firewall hardware dedicato e ci permetterà, qualora ne avessimo la necessità, di aggiungere in futuro ulteriori servizi senza bisogno di nuovo hardware.
Per trasformare un PC Debian in un gateway abbiamo bisogno innanzitutto di due schede di rete:

- `eth0` sarà la scheda di rete connessa alla nostra LAN e in questo articolo avrà indirizzo IP 192.168.1.1;
- `eth1` sarà la scheda di rete connessa al modem/router del nostro ISP e andrà configurata in base ai parametri di rete forniti al momento della stipula del contratto ADSL. Le schede di rete andranno configurate nella classica maniera Debian, agendo cioè sul file di configurazione`/etc/network/interfaces`.

Perché la nostra Debian funzioni come gateway e instradi correttamente i pacchetti dalla nostra LAN verso internet e viceversa, abbiamo bisogno che siano abilitate le funzionalità di *IP forwarding* e che siano stabilite alcune regole di instradamento. Tutto questo può essere fatto grazie a *iptables*, il programma che consente la configurazione di *netfilter*, il componente del kernel Linux che permette il filtraggio (con e senza stati) dei

pacchetti, la traduzione degli indirizzi di rete e di porta e altre forme di manipolazione dei pacchetti IP.
Iptables dovrebbe già essere stato installato nell'installazione base di Debian, ma per scrupolo effettuiamo un controllo:

```
# apt-get install iptables
```

Abbiamo inoltre bisogno di stabilire un insieme di regole da dettare a *iptables*:

- disabilitare le connessioni entranti da internet sulla scheda`eth1`;
- permettere l'instradamento dei pacchetti in uscita dalla LAN (scheda`eth0`) verso internet;
- permettere alle connessioni stabilite di ricevere i pacchetti di ritorno.

Iptables si configura generalmente costruendo degli script di regole da attivare all'avvio della macchina gateway. Le regole logiche definite al paragrafo precedente conducono alla stesura di uno script simile a questo:

```
#!/bin/sh

PATH=/usr/sbin:/sbin:/bin:/usr/bin

#
# Elimino eventuali regole esistenti
#
iptables -F
iptables -t nat -F
iptables -t mangle -F
iptables -X

#
# Abilito il traffico di loopback
#
iptables -A INPUT -i lo -j ACCEPT

#
```

```
# Permetto le connessioni attive e quelle provenienti dalla LAN
#
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m state --state NEW -i ! eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Peretto connessioni dalla LAN a internet
#
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT

#
# Configuro la funzione di mascheramento
#
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

#
# Impedisco il forward da internet verso la LAN
#
iptables -A FORWARD -i eth1 -o eth1 -j REJECT

#
# Abilito il routing
#
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Il *mascheramento IP* di Linux utilizzato nello script è un metodo per eseguire una traduzione dell'Indirizzo di Rete (in breve NAT, Network Address Translation) affinché anche macchine alle quali non è stato assegnato un indirizzo internet possano utilizzare tutti i servizi disponibili in rete, grazie ad una workstation Linux che funzioni come gateway. Tutta internet vedrà le richieste dell'intera LAN come se provenissero dalla sola macchina gateway poiché gli indirizzi IP *originari* dei vari client saranno stati *mascherati* da *iptables*.
A questo punto non resta che salvare il file come`/etc/init.d/firewall`, renderlo eseguibile con il comando:

```
# chmod +x /etc/init.d/firewall
```

e automatizzarlo:

```
# update-rc.d firewall defaults
```

La nostra macchina Debian è ora un gateway per tutta la rete LAN.

6.2.2 Il proxy server Squid

Un *caching proxy server* come Squid è un software che si interpone tra un client ed un server web, inoltrando le richieste e le risposte dall'uno all'altro:

- il client si collega al proxy invece che al server web, e gli invia delle richieste;
- il proxy a sua volta si collega al server web e inoltra la richiesta del client;
- il proxy riceve poi la risposta e la inoltra al client. In altre parole il proxy agisce come mediatore tra un qualunque pc della rete locale e internet. Questa funzione comporta alcuni vantaggi, tra cui:
- possibilità di tenere traccia di tutte le operazioni effettuate (ad esempio di tutte le pagine web visitate), consentendo statistiche ed osservazioni dell'utilizzo della rete. Non ci si dimentichi però di operare in conformità alle norme in vigore riguardanti la privacy;
- possibilità di velocizzare la navigazione dei client, poiché il proxy server mantiene nella sua memoria cache le pagine visitate più di frequente, potendole così servire ai client in maniera più rapida. Faremo inoltre in modo che il proxy si comporti in maniera *trasparente*, sollevando l'amministratore di rete da qualsiasi configurazione sulle macchine client.

Il proxy server più utilizzato nel mondo Linux è Squid, che andremo adesso ad installare sulla nostra macchina gateway:

```
# apt-get install squid
```

Il proxy Squid si configura attraverso il suo file di configurazione`/etc/squid/squid.conf`, in cui dobbiamo introdurre alcune modifiche.
Innanzitutto dobbiamo configurarlo per restare in ascolto sulla sola interfaccia interna e sull'interfaccia di loopback, aggiungendo le righe:

```
http_port 127.0.0.1:8080
http_port 192.168.1.1:8080
```

Poi dobbiamo impostare l'`hostname` del proxy e l'indirizzo email dell'amministratore, che sarà mostrato in caso di errori:

```
visible_hostname gateway.lan.local
cache_mgr        amministratore@lan.local
```

Dovremo infine aggiungere a *Squid* le informazioni per supportare la modalità *transparent proxy*. A seconda della versione di Squid utilizzata, queste direttive sono differenti.

- Per *Squid* **2.5** o versioni più vecchie dovremo aggiungere le righe:

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

- Per *Squid* **2.6** o versioni più nuove dovremo aggiungere la riga:

```
http_port 8080 transparent
```

L'ultima modifica che ci rimane da fare è quella di comunicare a *Squid* quali *subnet* sono autorizzate a collegarsi al proxy senza vedersi rifiutare la connessione:

```
# Autorizziamo la nostra LAN
```

```
acl our_networks src 192.168.1.0/24
http_access allow our_networks
http_access allow localhost

# Impediamo l'accesso al proxy al resto del mondo
http_access deny all
```

Arrivati a questo punto, un riavvio di *Squid* completerà l'opera di configurazione:

```
# /etc/init.d/squid restart
```

Ci troviamo adesso con un *gateway* e un *proxy server* - in ascolto all'indirizzo 192.168.1.1:8080 - perfettamente funzionanti e possiamo già agire sulla configurazione del browser dei nostri client per iniziare ad utilizzare Squid. Vogliamo però risparmiarci la fatica di entrare su ogni client e vogliamo rendere trasparente agli utenti la presenza del nostro proxy.
Per ottenere questo risultato dobbiamo scrivere una regola di *iptables* che intercetti le richieste uscenti verso il web (sulla porta 80 del nostro gateway) e le rediriga verso il nostro proxy server, in ascolto sulla porta 8080.
Torniamo quindi a modificare il file`/etc/init.d/firewall` creato in precedenza e aggiungiamo alla fine del file questa ulteriore regola:

```
# Transparent proxy
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Riavviamo il nostro script e prepariamoci a verificarne il funzionamento. Se da un client della rete proviamo a navigare su internet, agendo sulla console del nostro gateway dovremmo vederne i log:

```
# tail -f /var/log/squid/access.log
1104854410.086    159 192.168.1.50 TCP_MISS/302 469 GET http://www.google.com/    \
                                                - DIRECT/216.239.59.104 text/html
1104854410.217    128 192.168.1.50 TCP_MISS/200 1459 GET http://www.google.it/    \
                                                 - DIRECT/216.239.59.99 text/html
```

```
1104854410.397    180 192.168.1.50 TCP_MISS/200 9022 GET http://www.google.it/intl\
                            /it/images/logo.gif - DIRECT/216.239.59.99 image/gif
1104854415.196    200 192.168.1.50 TCP_MISS/200 1459 GET http://www.google.it/    \
                                                 - DIRECT/216.239.59.99 text/html
1104854415.271    74 192.168.1.50 TCP_REFRESH_HIT/304 235 GET http://www.google.it\
                          /intl/it/images/logo.gif - DIRECT/216.239.59.99 text/html
```

Perfetto! Il nostro *proxy server* è entrato correttamente in funzione.

6.2.3 Prime conclusioni

Abbiamo fino ad ora impostato una macchina Debian per funzionare come *gateway* di una rete LAN e per far girare un *proxy server* che sveltisca la navigazione dei client e ne registri i log.
Pur essendo questa una configurazione funzionante e completa, in alcune situazioni potrebbe essere conveniente espanderla e installare anche un sistema per filtrare i contenuti della navigazione. È in questi casi che torna utile una categoria di programmi chiamata *URL rewriter*.

6.2.4 DansGuardian: l'URL rewriter

Un *URL rewriter* è un software che si occupa di riscrivere un URL, sostituendolo con un'altro ritenuto più opportuno sulla base delle regole e dei filtri impostati. Il risultato è il dirottamento di certe richieste verso una o più pagine predefinite. Un caso tipico è quello in cui l'utente viene deviato su una pagina recante un messaggio standard del tipo *Spiacente, il sito che volevi raggiungere non sembra collegato agli interessi dell'Azienda* o qualcosa del genere.
Tecnicamente la riscrittura delle richieste HTTP può avvenire in due modi:

1. l'*url rewriter* è invocato come sottoprocesso dal *proxy* stesso allo scopo di riscrivere l'URL: il proxy si limiterà quindi a puntare verso l'url rewriter, senza curarsi di altro;

2. l'*url rewriter* riceve direttamente la richiesta dallo stack TCP/IP, la riscrive, se lo ritiene necessario, e in ogni caso passa l'URL originale, o quello riscritto, al proxy affinché sia inoltrato all'esterno

Un esempio di *url rewriter* del primo tipo è *squidGuard*, un eseguibile che viene invocato direttamente dal proxy *Squid*.
Un esempio di *url rewriter* del secondo tipo è invece *DansGuardian*, un servizio di rete che si frappone fra il browser e il proxy.
In questa guida la scelta ricadrà su *DansGuardian* perché ritenuto un sistema molto capillare e con la possibilità di essere pesantemente personalizzato dall'utente.
Prima di addentrarci nell'installazione e nella configurazione di DansGuardian è bene però modificare il numero di porta su cui si pone in ascolto il proxy server Squid. Apriamo quindi il suo file di configurazione`/etc/squid/squid.conf` e modifichiamo la riga:

```
http_port 127.0.0.1:8080
http_port 192.168.1.1:8080
```

in:

```
http_port 127.0.0.1:3128
http_port 192.168.1.1:3128
```

e la riga:

```
http_port 8080 transparent
```

in:

```
http_port 3128 transparent
```

Lo scopo di questa modifica è fare in modo che la regola di *iptables* impostata all'inizio dell'articolo indirizzi ora i pacchetti verso il nostro *url rewriter DansGuardian* (che porremo quindi in ascolto sulla porta 8080), lasciando poi a quest'ultimo il compito di reindirizzare i pacchetti filtrati verso il proxy server.
Ora possiamo procedere con l'installazione di *DansGuardian*:

```
# apt-get install dansguardian
```

Il file principale di configurazione è`/etc/dansguardian/dansguardian.conf`. Apportiamovi alcune piccole modifiche:

- impostiamo la lingua italiana:

```
language = 'italian'
```

- impostiamo il numero di porta su cui si metterà in ascolto:

```
filterip =
filterport = 8080
```

- comunichiamo a *DansGuardian* dove si trova *Squid*:

```
proxyip = 127.0.0.1
proxyport = 3128
```

- commentiamo infine la linea

```
UNCONFIGURED
```

che si trova all'inizio del file, altrimenti *DansGuardian* non funzionerà.

I filtri di *DansGuardian*

I files che permettono di agire sui filtri di *DansGuardian*, che in questo articolo descriveremo solo brevemente, si trovano tutti nella directory`/etc/dansguardian` e seguono questa convenzione:

- i file che cominciano per *banned* si riferiscono all'azione di negare l'accesso;
- i file che iniziano per *exception* si riferiscono all'azione di consentire l'accesso.

L'ordine in cui i file sono esaminati dal programma è il seguente:

- *exceptioniplist* - contiene l'elenco degli indirizzi IP che devono saltare il controllo dei contenuti (ad.es. gli indirizzi dei PC degli amministratori);
- *exceptionuserlist* - contiene l'elenco degli utenti che saltano il controllo di dansguardian (ad. es. gli amministratori);
- *exceptionsitelist* - contiene la parte terminale dei domini le cui pagine non saranno filtrate (ad. es. linux.org);
- *exceptionurllist* - contiene l'indirizzo di pagine di siti che non sono filtrate;
- *blanket block* - permette lo sblocco totale o parziale dei siti che sono indicati nei file greysitelist e greyurllist, a differenza dei file exception
- il filtro è applicato;
- *bannediplist* - indirizzi IP dei PC che non devono avere accesso al web;
- *banneduserlist* - nomi degli utenti che non devono avere accesso al web;
- *bannedregexpurllist* - elenco delle espressioni regolari negli URL a cui negare l'accesso ;
- *bannedurllist* - indirizzi di pagine web a cui non deve essere consentito l'accesso (serve per non bloccare un sito intero, ma solo parti di esso);
- *blanket ip block* - consente il blocco degli URL basati su IP;
- *bannedsitelist* - contiene la lista dei siti (domini) a cui non è consentito l'accesso (ad. es. sex.com);
- *postupload* - blocco o limite delle operazioni di upload (da impostare nel file`/etc/dansguardian/dansguardian.conf`);
- *bannedmimetypelist* - contiene la lista dei tipi MIME che saranno bloccati (è un modo eccellente per bloccare certi tipi di filmati);

- *bannedextensionlist* - contiene la lista delle estensioni di file da bloccare, può essere utilizzato per impedire lo scaricamento di certi screen saver e hacking tools;

- *exceptionphraselist* - se in una pagina compare una frase indicata in questo file l'accesso è consentito (prudenza nell'utilizzare questa opzione);

- *bannedphraselist* - contiene la lista delle frasi negate, se in una pagina è presente una di queste frasi l'accesso è negato;

- *weightedphraselist* - a ogni frase è assegnato un valore positivo o negativo, nell'analisi di una pagina i valori sono sommati. Frasi che hanno a che vedere con buoni argomenti avranno valori negativi, con cattivi argomenti valori positivi. Se la somma raggiunge il naughtynesslimit (da impostare nel file di configurazione`/etc/dansguardian/dansguardian.conf`) l'accesso alla pagina è negato. In generale 50 è per i bambini, 100 i primi anni dell'adolescenza, 160 per i giovani adulti; i valori indicati possono servire da punto di riferimento, la sperimentazione fornirà quelli più indicati allo specifico caso.

Per mezzo del file *bannedextensionlist* è possibile bloccare lo scaricamento di file eseguibili (ad es. exe, mentre con il file *bannedmimetypelist* è possibile bloccare filmati indesiderati ad esempio video/mpeg). Per rendere attive eventuali modifiche è necessario riavviare il servizio:

```
/etc/init.d/dansguardian restart
```

Le blacklist

Per far funzionare al meglio il nostro *DansGuardian* abbiamo bisogno ancora di una cosa: un insieme di *blacklist* preconfezionato e diviso in categorie. Il migliore (e anche quello consigliato dagli autori di *DansGuardian*) è l'elenco scaricabile a questo indirizzo: http://urlblacklist.com/?sec=download. Si noti che il download non è gratuito e che, prima di sottoscrivere un abbonamento, abbiamo diritto ad un solo download di prova. Purtroppo non sono a conoscenza di altre backlist compatibili con *DansGuardian* e altrettanto valide.

Scompattiamo il file scaricato dentro la directory`/etc/dansguardian/lists`; verrà creata una sottodirectory chiamata`blacklists` contenente tutte le liste, immediatamente utilizzabili.
Per rendere attive le blacklist scaricate dobbiamo aprire i file`bannedsitelist` e`bannedurllist` e decommentare le categorie di siti che intendiamo filtrare.
Già che ci siamo, apportiamo anche alcune altre modifiche ad un paio di altri files, dato che le impostazioni di default potrebbero essere trovate troppo restrittive:

- `bannedextensionlist`: decidiamo per quali estensioni di files vogliamo consentire il download;
- `bannedmimetypelist`;
- `weightedphraselist`: la directory`/etc/dansguardian/phraselists/badwords/` contiene le liste di frasi da censurare, divise per idioma.

Adattate ognuno di questi file alle vostre esigenze e, a configurazione ultimata, riavviate *DansGuardian* per fargli digerire le modifiche.

Evitiamo il bypass di *DansGuardian*

Utenti esperti potrebbero presto scoprire un trucco per aggirare il filtro di *DansGuardian*: poiché la nostra rete ha impostato una regola trasparente di *iptables* che reindirizza il traffico di rete verso la porta 8080 presidiata da *DansGuardian*, basterebbe entrare nelle proprietà del browser e forzare l'impostazione di un proxy sulla porta 3128 per permettere a un client di arrivare direttamente a *Squid* senza passare per la censura dell'*url filter*.
Per fortuna con una piccola ulteriore regola di *iptables*, da aggiungere alla fine del solito file`/etc/init.d/firewall`, possiamo facilmente impedirlo:

```
# Impedisco il bypass di DansGuardian
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 3128 -j REDIRECT --to-port 8080
```

In questo modo, qualunque client che tentasse una connessione diretta a *Squid* verrebbe automaticamente ridiretto su *DansGuardian*.

Quest'ultima regola, in combinazione con eventuali altre regole *iptables* che filtrino il traffico in uscita su porte non espressamente autorizzate, è in grado di darci il controllo completo su tutto il traffico entrante e uscente dalla nostra rete LAN.

6.2.5 Analisi dei Log

Tutti i log di questo sistema di filtraggio dei contenuti saranno reperibili nelle directory `/var/log/squid/` e `/var/log/dansguardian/`.
Per una comoda lettura dei log generati da *Squid* e *DansGuardian* suggerisco di installare il pacchetto

```
# apt-get install calamaris
```

Calamaris è uno script Perl che genera belle statistiche a partire dai file di registro di Squid o Oops. Viene invocato giornalmente prima che il proxy faccia la rotazione dei propri file di registro e invia le statistiche via posta o le mette sul web. (Fonte: http://packages.debian.org/lenny/calamaris)
Il seguente esempio genera un report in formato *html*.

```
# calamaris -a -F html /var/log/squid/access.log > log_squid.html
```

6.2.6 Conclusioni

Abbiamo visto in questo articolo un sistema abbastanza completo per implementare un *proxy server* che funzioni anche come filtro dei contenuti web da veicolare agli utenti di una LAN.
Senza addentrarci in dettagli legali, vi ricordo che è necessario che una struttura che intenda installare un sistema di controllo della navigazione si doti anche di un *regolamento per la navigazione internet*, affisso nella bacheca degli annunci aziendali, allegato ad ogni contratto di assunzione e preventivamente discusso con gli organi sindacali della struttura stessa.
A tale scopo può essere utile considerare le linee guida del Garante per l'*utilizzo di posta elettronica e internet*, che possiamo così riasssumere:

- I datori di lavoro pubblici e privati non possono controllare la posta elettronica e la navigazione in Internet dei dipendenti, se non in casi eccezionali. Spetta al datore di lavoro definire le modalità d'uso di tali strumenti, tenendo conto dei diritti dei lavoratori e della disciplina in tema di relazioni sindacali.

- Il Garante privacy, con un provvedimento generale che sarà pubblicato sulla - Gazzetta Ufficiale, fornisce concrete indicazioni in ordine all'uso dei computer sul luogo di lavoro. La questione è particolarmente delicata - afferma il relatore Mauro Paissan - perché dall'analisi dei siti web visitati si possono trarre informazioni anche sensibili sui dipendenti e i messaggi di posta elettronica possono avere contenuti a carattere privato. Occorre prevenire usi arbitrari degli strumenti informatici aziendali e la lesione della riservatezza dei lavoratori.

- L'Autorità prescrive innanzitutto ai datori di lavoro di informare con chiarezza e in modo dettagliato i lavoratori sulle modalità di utilizzo di Internet e della posta elettronica e sulla possibilità che vengano effettuati controlli. Il Garante vieta poi la lettura e la registrazione sistematica delle e-mail così come il monitoraggio sistematico delle pagine web visualizzate dal lavoratore, perché ciò realizzerebbe un controllo a distanza dell'attività lavorativa vietato dallo Statuto dei lavoratori. Viene inoltre indicata tutta una serie di misure tecnologiche e organizzative per prevenire la possibilità, prevista solo in casi limitatissimi, dell'analisi del contenuto della navigazione in Internet e dell'apertura di alcuni messaggi di posta elettronica contenenti dati necessari all'azienda.

- Il provvedimento raccomanda l'adozione da parte delle aziende di un disciplinare interno, definito coinvolgendo anche le rappresentanze sindacali, nel quale siano chiaramente indicate le regole per l'uso di Internet e della posta elettronica.

- Il datore di lavoro è inoltre chiamato ad adottare ogni misura in grado di prevenire il rischio di utilizzi impropri, così da ridurre controlli successivi sui lavoratori. Per quanto riguarda Internet è opportuno ad esempio:

  - individuare preventivamente i siti considerati correlati o meno con la prestazione lavorativa;
  - utilizzare filtri che prevengano determinate operazioni, quali l'accesso a siti inseriti in una sorta di black list o il download di file musicali o multimediali.

ferdybassi

Capitolo 7

Softwares in analisi

Approfondimenti e test su tutto il softwares per la nostra debian.
Il mondo del software libero ha raggiunto dimensioni più che ammirevoli.
In questa sezione cercheremo di presentarvi delle applicazioni utilizzabili con la nostra debian, nel modo più esaustivo possibile.

7.1 Moblock

In questo articolo vorrei introdurvi ad uno strumento che non sempre viene considerato nella sicurezza.
Molti parlando di sicurezza pensano subito ad un firewall, ma questi ha una funzione di filtro sulle porte, mentre quello che andremo ora ad illustrare si potrebbe considerare un secondo filtro, infatti andrà ad agire sugli IP.
MoBlock nelle ultime versioni si è molto evoluto e da semplice applicazione per filtrare gli IP da un database, ora ha aumentato la sua potenza andando ad inserire le proprie regole direttamente in iptables.
MoBlock si occuperà di filtrare in ingresso e in uscita tutti gli IP che gli arriveranno da analizzare, autorizzando o negando loro l'accesso; si può intuire, quindi, quanto sia importante avere una valida lista di IP da aggiornare quotidianamente.

7.1.1 Note dal Sito

http://moblock-deb.sourceforge.net/

> MoBlock-deb, come PeerGuardian, controlla il traffico Internet sulla base di liste di indirizzi IP. Sono disponibili i pacchetti di MoBlock-deb per Debian (Lenny, Squeeze e Sid) e Ubuntu (Hardy e Intrepid).
>
> Moblock-deb è composta dai demoni MoBlock e NFBlock, che eseguono il blocco degli indirizzi IP, da blockcontrol, un'interfaccia a linea di comando progettata per svolgere tutti i compiti legati al blocco di IP dei demoni, e da mobloquer una GUI per blockcontrol.
>
> **ATTENZIONE**: MoBlock può bloccare completamente l'accesso alla rete/internet!
>
> Blockcontrol, di default, si avvia automaticamente all'avvio del sistema. Alcune blocklists preconfigurate sono aggiornate una volta al giorno. Bisogna, però, prestare molta attenzione perché non solo sono bloccati molti indirizzi IP indesiderati, ma nella maggior parte dei casi l'utilizzo di tali blocklist può comportare una limitata disponibilità della rete, inclusa la propria LAN ed il proprio router, molte pagine web, servizi come e-mail, messaggistica istantanea o le "meteo applet" e l'accessibilità da internet alla proprio pc. È possibile configurare opportunamente MoBlock per evitare tali comportamenti.

L'impostazione di default (basata su una whitelist) consente l'accesso alla LAN, al server DNS ed al device di loopback. Su una LAN pubblica può essere opportuno disabilitare questo comportamento.

Inoltre MoBlock (dalla versione 0.9) e NFBlock non vanno in conflitto con altri firewalls basati su regole di iptables. È necessario, in questo caso, prestare particolare attenzione al fine di evitare gravi conflitti, ed assicurarsi che siano soddisfatte le seguenti condizioni:

1. Il demone di blocco IP contrassegni i pacchetti non-abbinati (L'IP non è nella blocklist). (Come impostazione predefinita tale funzionalità è attiva.)
2. Altri firewalls non contrassegnino i pacchetti.
3. Blockcontrol è avviato dopo gli altri firewalls. Se altri firewalls sono avviati / ricaricati dopo blockcontrol, allora è necessario riavviare blockcontrol nuovamente. Le regole di iptables che inviano traffico alle catene di iptables (blockcontrol_in, blockcontrol_out e blockcontrol_fw), devono precedere tutte le altre regole di iptables che accettano il traffico.

Blockcontrol ha le seguenti caratteristiche:

- Avviare e fermare il demone di blocco IP. O permettere a init di farlo automaticamente
- Aggiornare la tua blocklist da sorgenti online e da blocklists locali. O permettere a cron di farlo automaticamente a intervalli regolari.
- Rimuovere le righe per parola chiave dalla blocklist.
- Gestire le regole di iptables: utilizzare una configurazione predefinita, consentire tutto il traffico su porte specifiche e utilizzare un elenco con le autorizzazioni, o aggiungere le proprie sofisticate regole di iptables.
- Consentire tutto il traffico LAN e il server DNS automaticamente. Se siete su una LAN pubblica, probabilmente si desidera disabilitare questa funzione.
- Controllare lo stato e testare il demone di blocco IP.
- Rilevare se i moduli del kernel sono necessari ed eventualmente li carica

- Impostare la verbosità and le opzioni di log.
- Fornire script di init compatibile con LSB 3.1.
- Rotazione giornaliera dei file di log.

Configurazione e utilizzo (da eseguire con i privilegi di root):

- blockcontrol start – inserisce le regole di iptable e avvia il demone di blocco IP. Se cambia la configurazione di blocklist è necessario ricostruire il blocklist master.
- blockcontrol stop – cancella le regole di iptable e arresta il demone di blocco IP.
- blockcontrol restart – riavvia il demone di blocco IP.
- blockcontrol reload – recostruisce il blocklist master e ricarica il demone di blocco IP se è in esecuzione.
- blockcontrol update – aggiorna le blocklists, riscostruisce il blocklist master e ricarica il demone di blocco IP.
- blockcontrol status – fornisce le impostazioni di iptables settings e lo stato del demone di blocco IP.
- blockcontrol test – esegue un semplice test per verificare se il demone di blocco IP è in funzione (pinga un IP a caso nella blocklist e verifica se tale IP è presente nel file di log del demone di blocco e se ha risposto).
- search PATTERN – visualizza le occorrenze di una parola chiave e i nomi delle singole blocklist.
- stats – riporta le statistiche di MoBlock
- reset_stats – resetta le statistiche di MoBlock.
- show_config – mostra le impostazioni della configurazione corrente.

7.1.2 Installazione (i386 and amd64)

Aggiungete al vostro */etc/apt/sources.list*

```
#Debian lenny (stable):
```

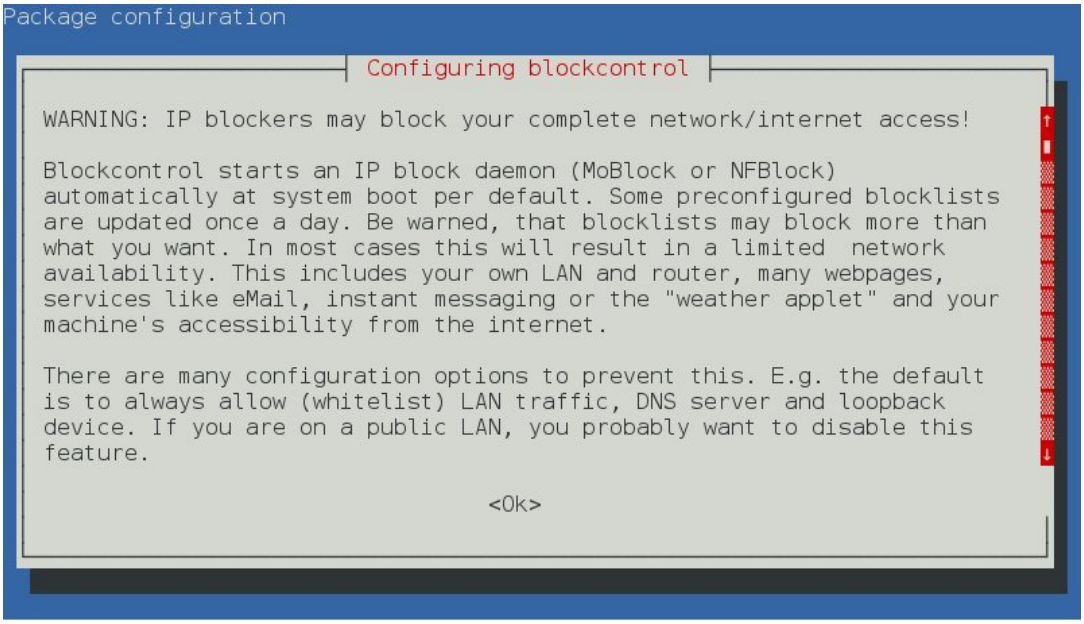
```
deb http://moblock-deb.sourceforge.net/debian lenny main
deb-src http://moblock-deb.sourceforge.net/debian lenny main


#Debian squeeze (testing):

deb http://moblock-deb.sourceforge.net/debian sid main
deb-src http://moblock-deb.sourceforge.net/debian sid main


#Debian sid (unstable):

deb http://moblock-deb.sourceforge.net/debian sid main
deb-src http://moblock-deb.sourceforge.net/debian sid main
```

Da root eseguite:

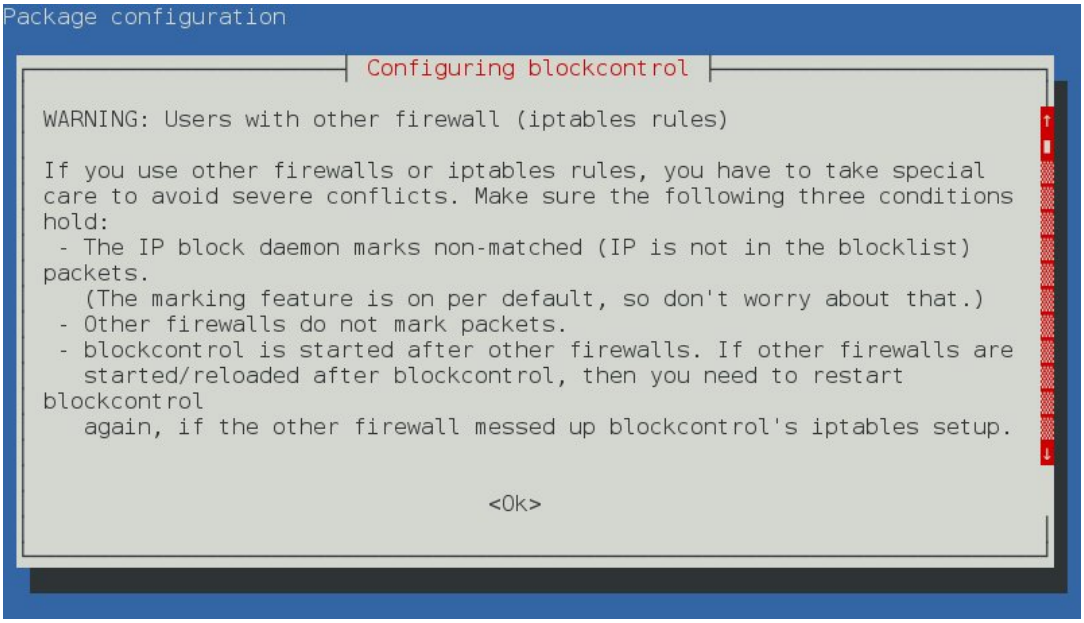
```
# gpg --keyserver wwwkeys.eu.pgp.net --recv-keys 58712F29
# gpg --export --armor 58712F29 | sudo apt-key add -

# aptitude update
# aptitude install moblock blockcontrol mobloquer
```

Oltre ai pacchetti citati verranno installate le seguenti dipendenze: *libnetfilter-queue1*, *libnfnetlink0*. In ogni caso, a seconda dei pacchetti già installati, potrebbero essercene altre.
Dopo aver scaricato tramite aptitude i pacchetti si passerà in modo automatico alla configurazione degli stessi. Le immagini che seguiranno vi guideranno nei passaggi successivi dell'installazione.

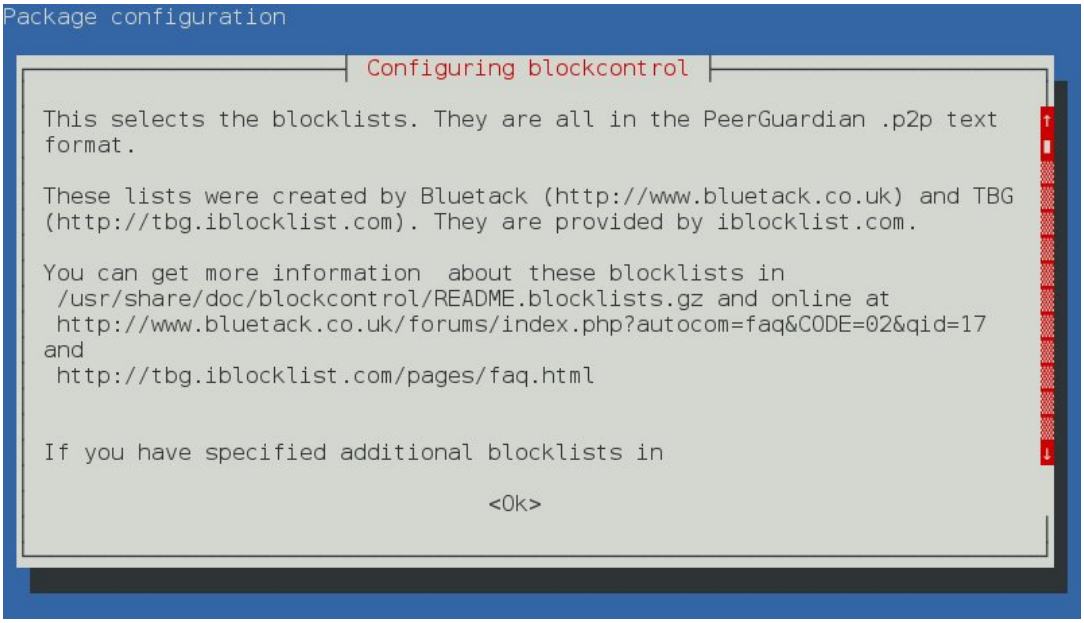
Si comincia così:



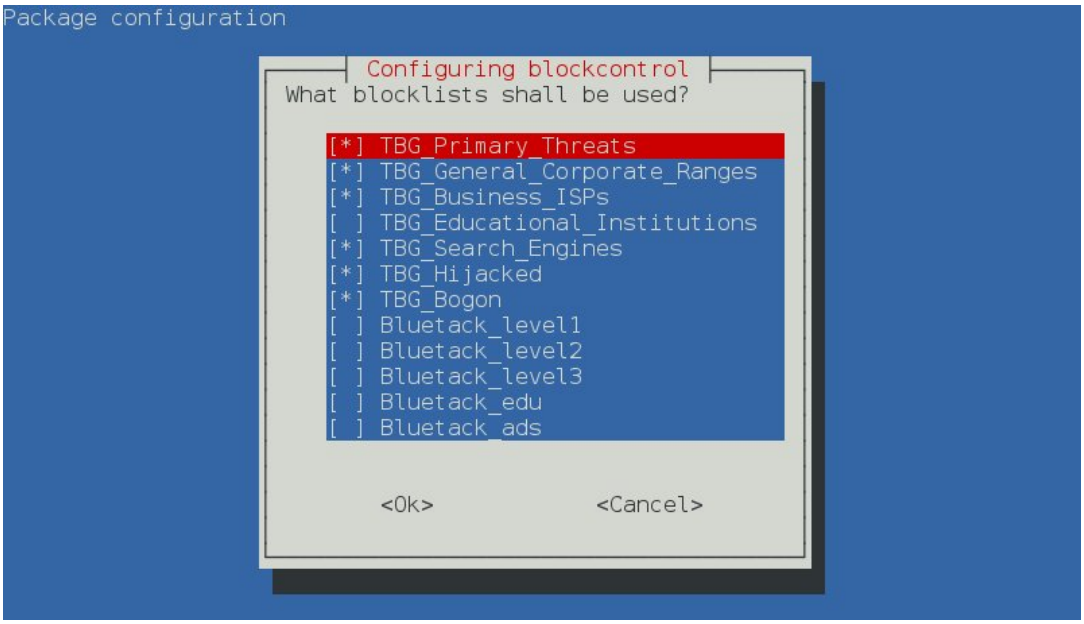
Ci viene poi fatto notare che MoBlock potrebbe avere conflitti con regole di altri firewall:



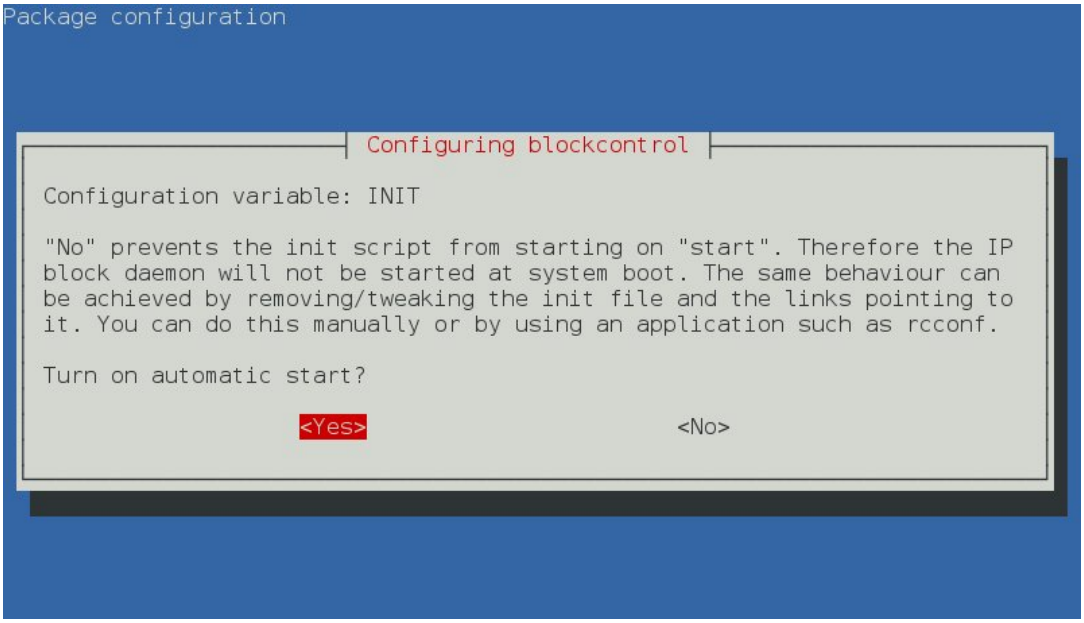
Seguono ulteriori informazioni riguardo le blacklist:



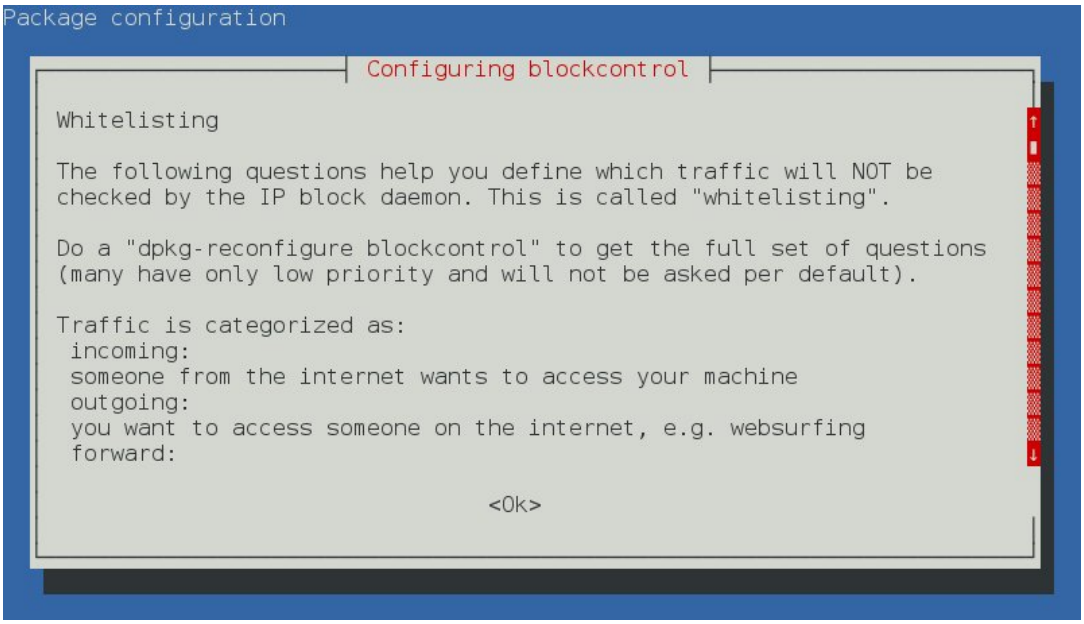
Qui abilitiamo alcuni dei database IP (le liste sopracitate) da usare come filtro; è possibile aggiungerli o rimuoverli anche in seguito (v. sotto).



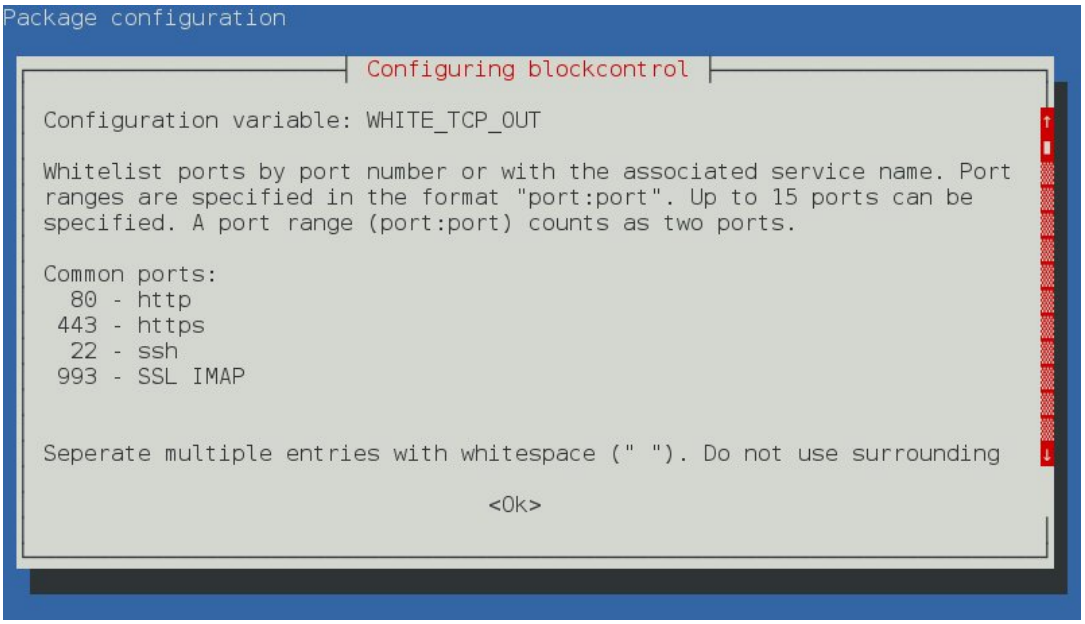
Possiamo poi scegliere di avviare il demone al boot del sistema tramite INIT:



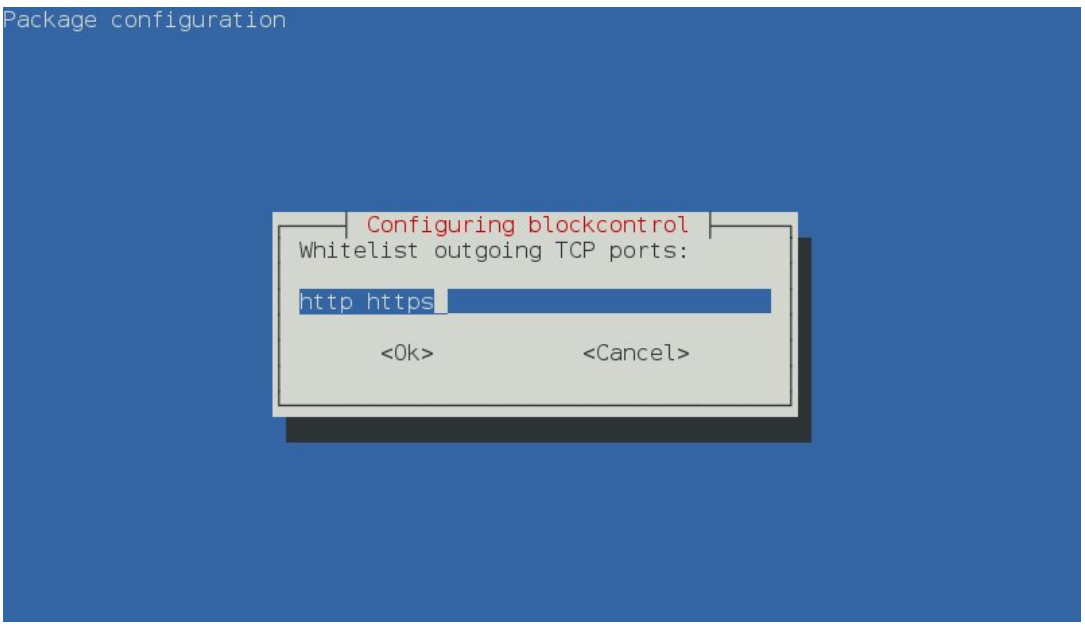
Seguono indicazioni sulle *whitelist*: le liste di indirizzi IP e porte TCP e UDP che non saranno soggette al blocco. Inoltre verremo informati che un *dpkg-reconfigure blockcontrol* ci darà ulteriori possibilità di configurazione non presentate nell'installazione di default.



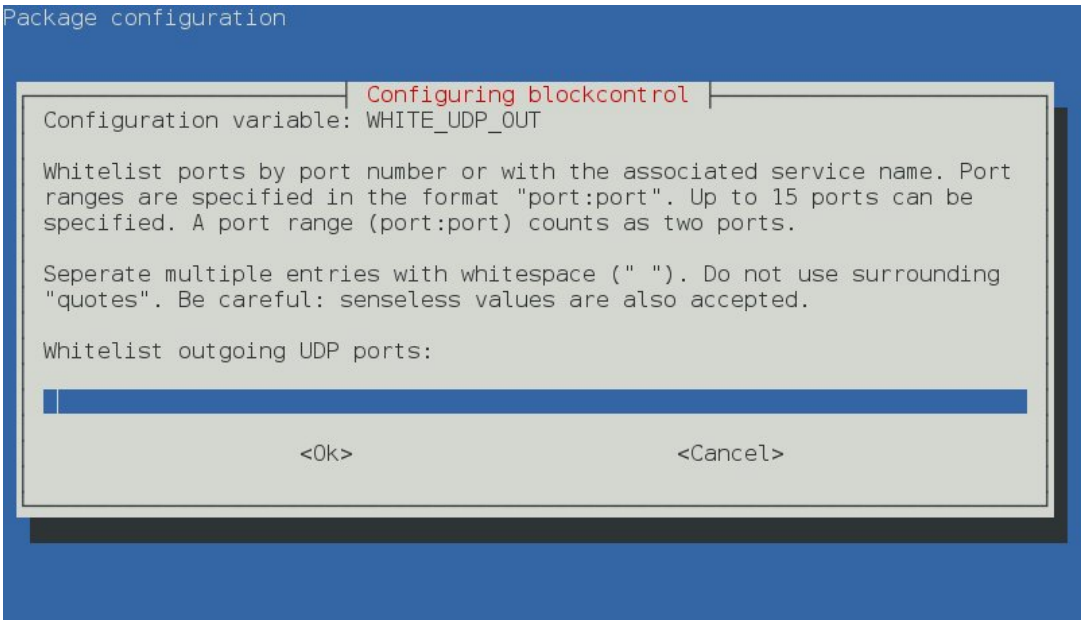
La prima lista è riferita alle porte in uscita del TCP; ci viene proposto qualche esempio:



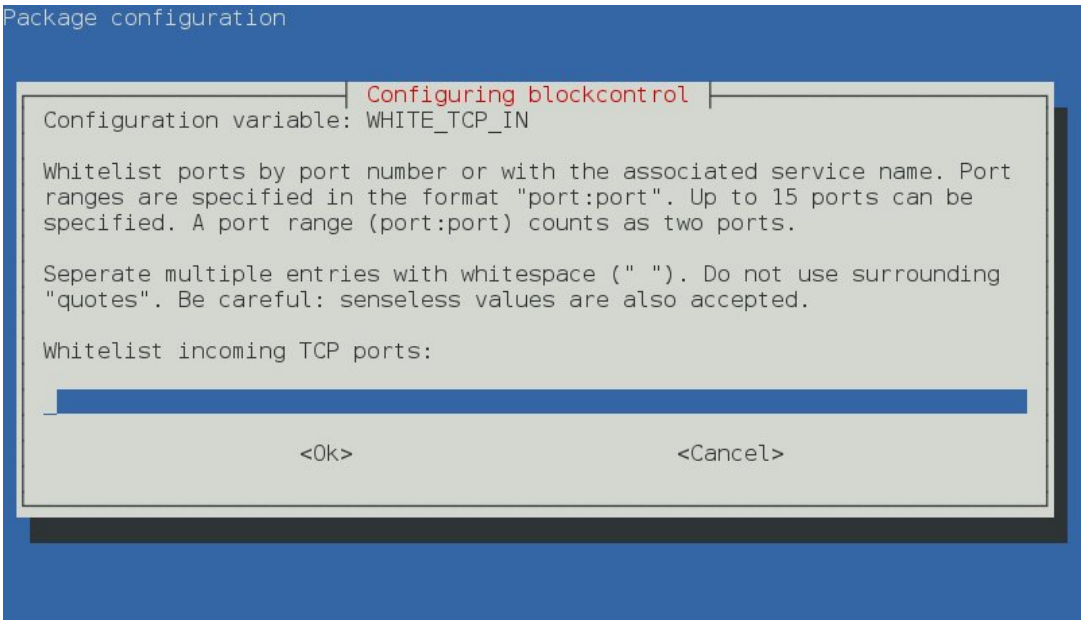
Ora se volete aprire alcune porte, inseritele qui separandole con uno spazio. Io, per esempio, ho inserito la porta 4662 per aMule e la 1720 per Ekiga. Una piccola nota va fatta sulla differenza fra TCP e UDP; non è lo scopo di quest'articolo spiegare ora le caratteristiche dei due protocolli ma sicuramente dovrà essere tenuta in conto a seconda del servizio che vogliamo attivare.



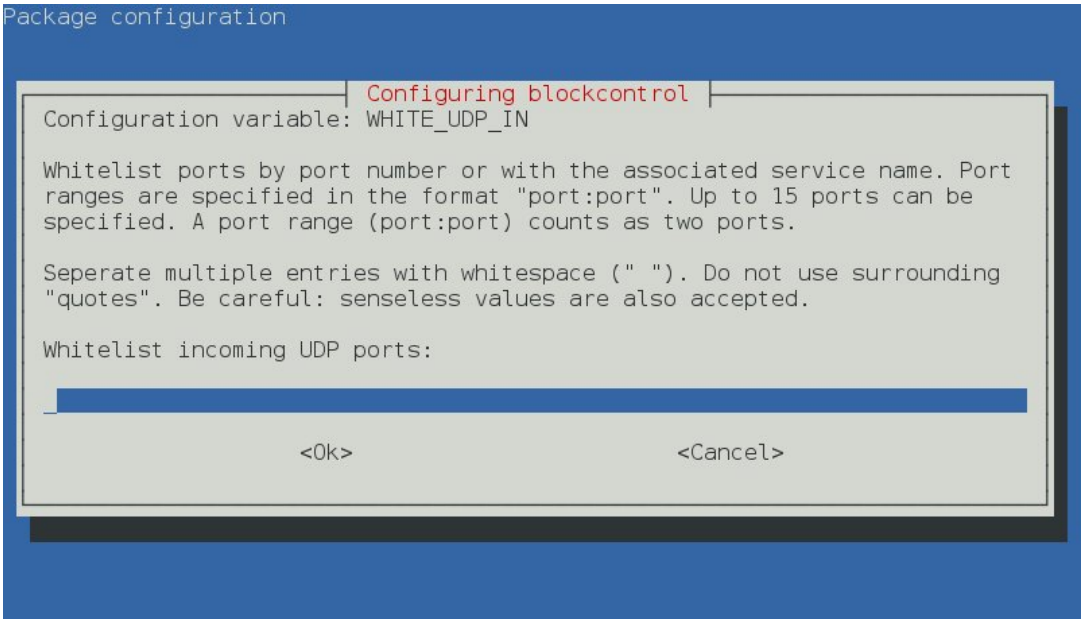
Qui (porte UDP in uscita) invece ho inserito 4672 per aMule e 3478, 3479, 5060 per Ekiga:



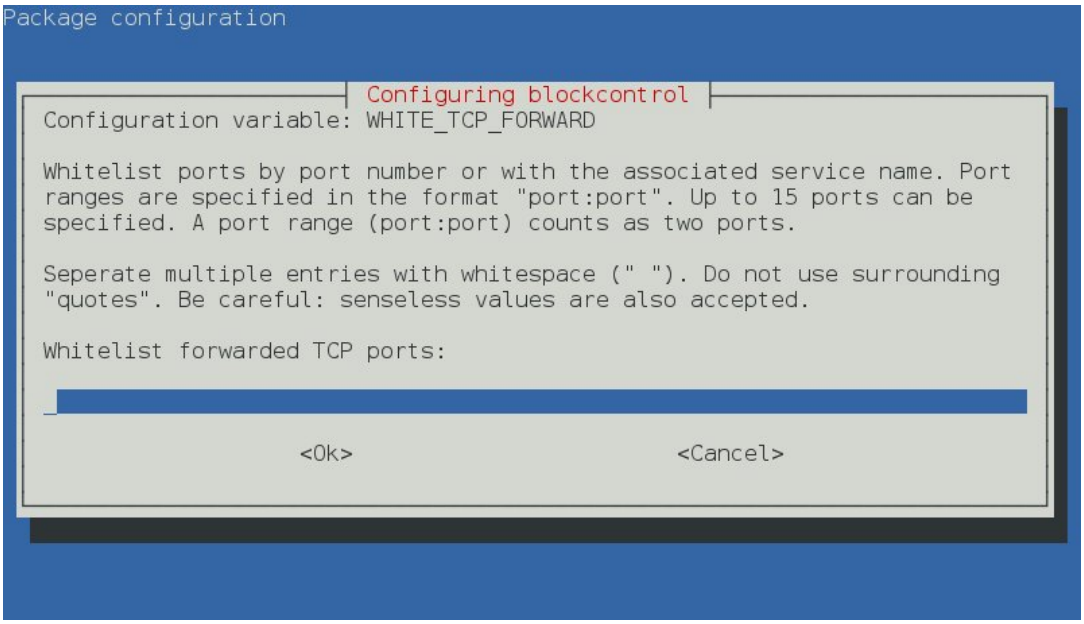
Qui (porte TCP in entrata) 4662 per aMule:

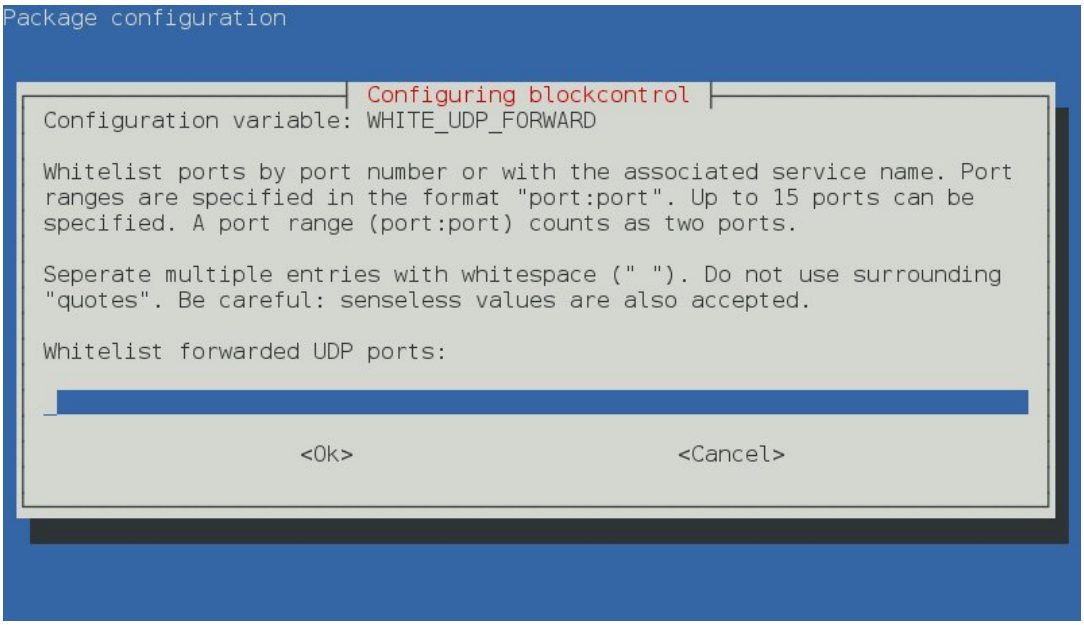


Qui (porte UDP in entrata) 4672 per aMule:

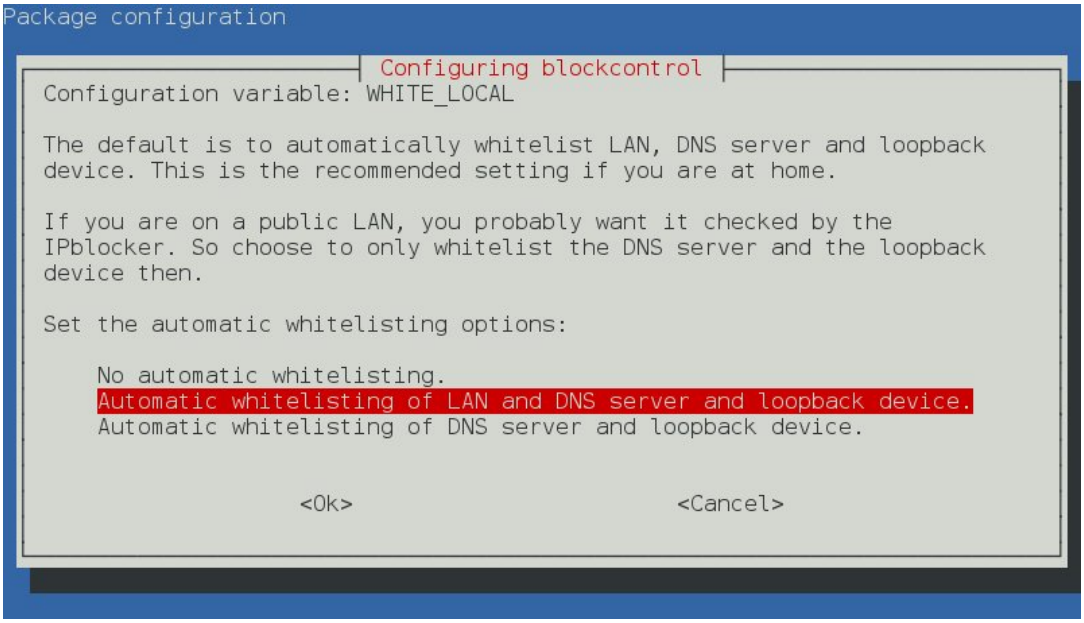


Seguono due schermate per la configurazione del *forward* sia per TCP che per UDP:

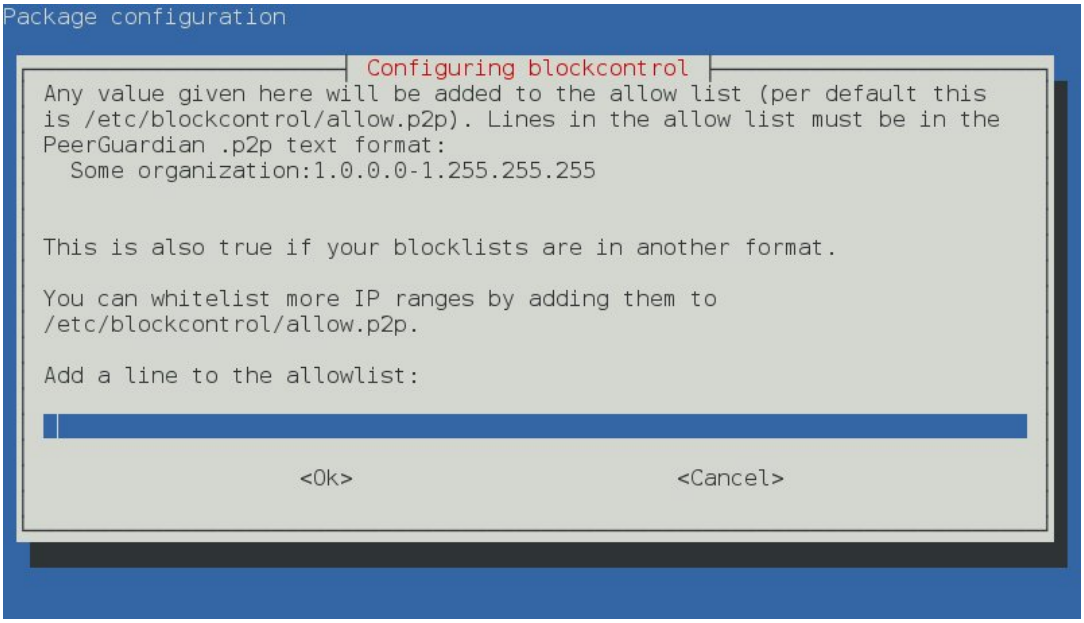




Qui impostate il criterio della *whitelist* (come ci viene indicato, l'opzione predefinita è un'ottima scelta se siete in una rete domestica):



Infine possiamo inserire gli indirizzi IP dei server che non vogliamo sottostiano alle regole di MoBlock (ad esempio il mail server al quale ci colleghiamo per leggere la posta):



ATTENZIONE - Potreste trovare problemi nel terminare l'installazione, questo accade spesso quando si aggiorna dalla vecchia versione con moblock-control. Il mio consiglio è di disinstallare tutto prima di procedere all'installazione. Aptitude è un'ottima scelta in quanto ci permetterà di eliminare anche le dipendenze relative ai pacchetti in un solo passaggio:

```
# aptitude purge moblock mobloquer blockcontrol
```

quindi ritentare l'installazione.
A volte si può bloccare al termine dell'installazione perché non riesce a connettersi al server per scaricare le liste.

7.1.3 Database lista IP

Per avere un filtro efficiente occorre avere un database aggiornato quotidianamente e con i giusti filtri, per questo vi consiglio due siti che vi offrono un archivio fornitissimo e per tutti i gusti:

- B.I.S.S. : http://www.bluetack.co.uk/forums/index.php
- blocklist : http://iblocklist.com/lists.php

Per inserire nuovi filtri o rimuoverli manualmente dalla nostra lista, il file da modificare è: */etc/blockcontrol/blocklists.list*.
Questa è una lista aggiornata che potete copiare in */etc/blockcontrol/blocklists.list*:

```
##Comprende la lista completa aggiornata al 17.02.2010
##La parte commentata riporta il nome della lista e l'autore della lista
##Sotto al nome trovale il link della lista, decommentare a piacere
##more info : http://iblocklist.com/lists.php

#level1 Bluetack
#http://list.iblocklist.com/?list=bt_level1
#level2 Bluetack
#http://list.iblocklist.com/?list=bt_level2
#level3 Bluetack
#http://list.iblocklist.com/?list=bt_level3
#edu Bluetack
#http://list.iblocklist.com/?list=bt_edu
#rangetest Bluetack
#http://list.iblocklist.com/?list=bt_rangetest
#bogon Bluetack
#http://list.iblocklist.com/?list=bt_bogon
#ads Bluetack
#http://list.iblocklist.com/?list=bt_ads
#spyware Bluetack
#http://list.iblocklist.com/?list=bt_spyware
#proxy Bluetack
#http://list.iblocklist.com/?list=bt_proxy
#badpeers Bluetack
#http://list.iblocklist.com/?list=bt_templist
#Microsoft Bluetack
#http://list.iblocklist.com/?list=bt_microsoft
#spider Bluetack
#http://list.iblocklist.com/?list=bt_spider
#hijacked Bluetack
```

```
#http://list.iblocklist.com/?list=bt_hijacked
#dshield Bluetack
#http://list.iblocklist.com/?list=bt_dshield
#Webexploit Forumspam Bluetack
#http://list.iblocklist.com/?list=bimsvyvtgxeelunveyal
#iana-reserved Bluetack
#http://list.iblocklist.com/?list=bcoepfyewziejvcqyhqo
#iana-private Bluetack
#http://list.iblocklist.com/?list=cslpybexmxyuacbyuvib
#iana-multicast Bluetack
#http://list.iblocklist.com/?list=pwqnlynprfgtjbgqoizj
#fornonlancomputers Bluetack
#http://list.iblocklist.com/?list=jhaoawihmfxgnvmaqffp
#Primary Threats TBG
#http://list.iblocklist.com/?list=ijfqtofzixtwayqovmxn
#General Corporate Ranges TBG
#http://list.iblocklist.com/?list=ecqbsykllnadihkdirsh
#Business ISPs TBG
#http://list.iblocklist.com/?list=jcjfaxgyyshvdbceroxf
#Educational Institutions TBG
#http://list.iblocklist.com/?list=lljggjrpmefcwqknpalp
#Search Engines TBG
#http://list.iblocklist.com/?list=pfefqteoxlfzopecdtyw
#Hijacked TBG
#http://list.iblocklist.com/?list=tbnuqfclfkemqivekikv
#Bogon TBG
#http://list.iblocklist.com/?list=ewqglwibdgjttwttrinl
#ipfilterX Nexus23
#http://list.iblocklist.com/?list=nxs23_ipfilterx
#DROP Spamhaus
#http://list.iblocklist.com/?list=sh_drop
#Pedophiles DCHubAd
#http://list.iblocklist.com/?list=dcha_pedophiles
#Spammer DCHubAd
#http://list.iblocklist.com/?list=dcha_spammer
#Hacker DCHubAd
#http://list.iblocklist.com/?list=dcha_hacker
#Faker DCHubAd
```

```
#http://list.iblocklist.com/?list=dcha_faker
#Atma Atma
#http://list.iblocklist.com/?list=tzmtqbbsgbtfxainogvm
#RapidShare PeerBlock
#http://list.iblocklist.com/?list=zfucwtjkfwkalytktyiw
#bogon cidr-report
#http://list.iblocklist.com/?list=cr_bogon
#LogMeIn I-Blocklist
#http://list.iblocklist.com/?list=logmein
#Steam I-Blocklist
#http://list.iblocklist.com/?list=steam
#Xfire I-Blocklist
#http://list.iblocklist.com/?list=xfire
#Blizzard I-Blocklist
#http://list.iblocklist.com/?list=blizzard
#Ubisoft I-Blocklist
#http://list.iblocklist.com/?list=ubisoft
#Nintendo I-Blocklist
#http://list.iblocklist.com/?list=nintendo
#Activision I-Blocklist
#http://list.iblocklist.com/?list=activision
#Sony Online Entertainment I-Blocklist
#http://list.iblocklist.com/?list=soe
#Crowd Control Productions I-Blocklist
#http://list.iblocklist.com/?list=ccp
#Linden Lab I-Blocklist
#http://list.iblocklist.com/?list=lindenlab
#Electronic Arts I-Blocklist
#http://list.iblocklist.com/?list=electronicarts
#Square Enix I-Blocklist
#http://list.iblocklist.com/?list=squareenix
#NCsoft I-Blocklist
#http://list.iblocklist.com/?list=ncsoft
#PunkBuster I-Blocklist
#http://list.iblocklist.com/?list=punkbuster
#Joost I-Blocklist
#http://list.iblocklist.com/?list=joost
#Pandora I-Blocklist
```

```
#http://list.iblocklist.com/?list=aevzidimyvwybzkletsg
#The Pirate Bay I-Blocklist
#http://list.iblocklist.com/?list=nzldzlpkgrcncdomnttb
#Apple I-Blocklist
#http://list.iblocklist.com/?list=aphcqvpxuqgrkgufjruj
#The Onion Router I-Blocklist
#http://list.iblocklist.com/?list=tor
#AOL I-Blocklist
#http://list.iblocklist.com/?list=aol
#Comcast I-Blocklist
#http://list.iblocklist.com/?list=comcast
#Cablevision I-Blocklist
#http://list.iblocklist.com/?list=cablevision
#Verizon I-Blocklist
#http://list.iblocklist.com/?list=verizon
#AT&T I-Blocklist
#http://list.iblocklist.com/?list=att
#Cox I-Blocklist
#http://list.iblocklist.com/?list=cox
#Road Runner I-Blocklist
#http://list.iblocklist.com/?list=roadrunner
#Charter I-Blocklist
#http://list.iblocklist.com/?list=charter
#Qwest I-Blocklist
#http://list.iblocklist.com/?list=qwest
#Embarq I-Blocklist
#http://list.iblocklist.com/?list=embarq
#Suddenlink I-Blocklist
#http://list.iblocklist.com/?list=suddenlink
#Australia I-Blocklist
#http://list.iblocklist.com/?list=au
#Brazil I-Blocklist
#http://list.iblocklist.com/?list=br
#Canada I-Blocklist
#http://list.iblocklist.com/?list=ca
#China I-Blocklist
#http://list.iblocklist.com/?list=cn
#Germany I-Blocklist
```

```
#http://list.iblocklist.com/?list=de
#Spain I-Blocklist
#http://list.iblocklist.com/?list=es
#European Union I-Blocklist
#http://list.iblocklist.com/?list=eu
#France I-Blocklist
#http://list.iblocklist.com/?list=fr
#United Kingdom I-Blocklist
#http://list.iblocklist.com/?list=gb
#Italy I-Blocklist
#http://list.iblocklist.com/?list=it
#Japan I-Blocklist
#http://list.iblocklist.com/?list=jp
#Republic of Korea I-Blocklist
#http://list.iblocklist.com/?list=kr
#Mexico I-Blocklist
#http://list.iblocklist.com/?list=mx
#Netherlands I-Blocklist
#http://list.iblocklist.com/?list=nl
#Russia I-Blocklist
#http://list.iblocklist.com/?list=ru
#Sweden I-Blocklist
#http://list.iblocklist.com/?list=se
#Taiwan I-Blocklist
#http://list.iblocklist.com/?list=tw
#United States I-Blocklist
#http://list.iblocklist.com/?list=us
```

Se avete uno o più IP che sfuggono al normale controllo, potete farvi una vostra lista e inserire all'interno gli indirizzi da bloccare; basterà poi far caricare la lista personalizzata. Creeremo allora un file con un nome a piacimento che termini in *.p2p*, ad esempio *barabba_list.p2p* e inseriremo all'interno:

```
##lista personale Ip da bloccare

amico indesiderato : xxx.xxx.xxx.xxx-xxx.xxx.xxx.xxx
```

Potete inserirla manualmente con la posizione del file all'interno di */etc/blockcontrol/blocklists.list*, oppure caricarla da Mobloquer (v. sotto).

Consiglio di prestare molta attenzione alla configurazione delle liste IP da utilizzare perché una configurazione errata potrebbe compromettere l'uso della rete sia in locale che sul web. Per esempio con la lista *'bogon'* troverete molti problemi nell'utilizzo della rete locale rendendovi samba inefficace; la lista *'road runner'* inibirà i servizi di applicazioni come quick time. Consiglio di provare ad aggiungere le liste una alla volta e verificare prima se tutto va bene perché nel caso fossero tutte attive, cercare di capire qual'è quella che non permette di usare la rete vi farà perdere molto tempo.

7.1.4 Utilizzo

MoBlock è comodamente utilizzabile da shell e i comandi sono semplici e intuitivi; loggati come root basta eseguire:

```
# blockcontrol -h
Usage: blockcontrol OPTION
Options:
        start
        stop
        restart
        reload
        update
        status
        test
        stats
        reset_stats
        show_config
        search
```

Nota - Consiglio di aggiornare la lista con MoBlock spento e poi avviarlo dato che a volte ha problemi a scaricare le liste. Se aggiornate le liste con MoBlock attivo, al termine dovrete fare un reload delle nuove liste appena aggiornate.

Aggiungere/Rimuovere IP e/o porte TCP/UDP

Sia gli IP (in entrata e in uscita), che le porte TCP/UDP (sempre sia in entrata che in uscita) vengono memorizzate nel file */etc/blockcontrol/blockcontrol.conf*. Si potrà dunque agire direttamente su questo file per aggiungere/rimuovere porte e/o IP. Ecco come si presenta dopo le configurazioni sopra illustrate:

```
# blockcontrol.conf - configuration file for blockcontrol

# This file is sourced by a shell script. Any line which starts with a # (hash)
# is a comment and is ignored. If you set the same variable several times,
# then only the last line will be used.

# Refer to blockcontrol.defaults (/usr/lib/blockcontrol/blockcontrol.defaults)
# for the complete set of possible configuration variables with comments.

# Do a "blockcontrol restart" (sometimes even "reload" is enough) when you have
# edited this file.

WHITE_TCP_OUT="http https 4662 1720"
WHITE_UDP_OUT="4672 3478 3479 5060"
WHITE_TCP_IN="4662"
WHITE_UDP_IN="4672"
```

Per aggiungere *forward* a TCP/UDP, così come dei numeri IP da lasciar passare, basterà aggiungere le seguenti linee:

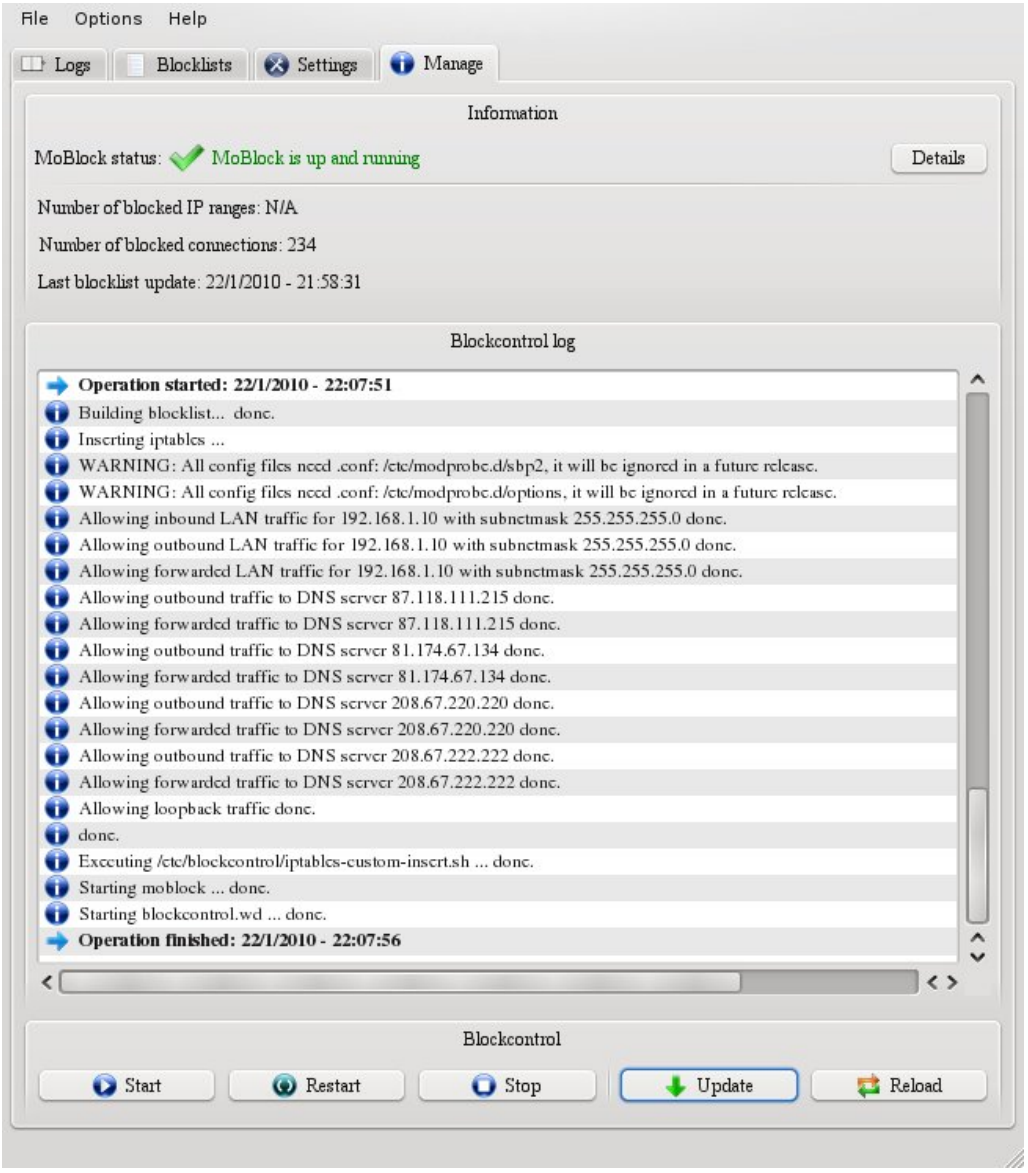
```
WHITE_TCP_FORWARD=""
WHITE_UDP_FORWARD=""
WHITE_IP_IN=""
WHITE_IP_OUT=""
```

inserendo evidentemente i valori relativi separati da uno spazio.

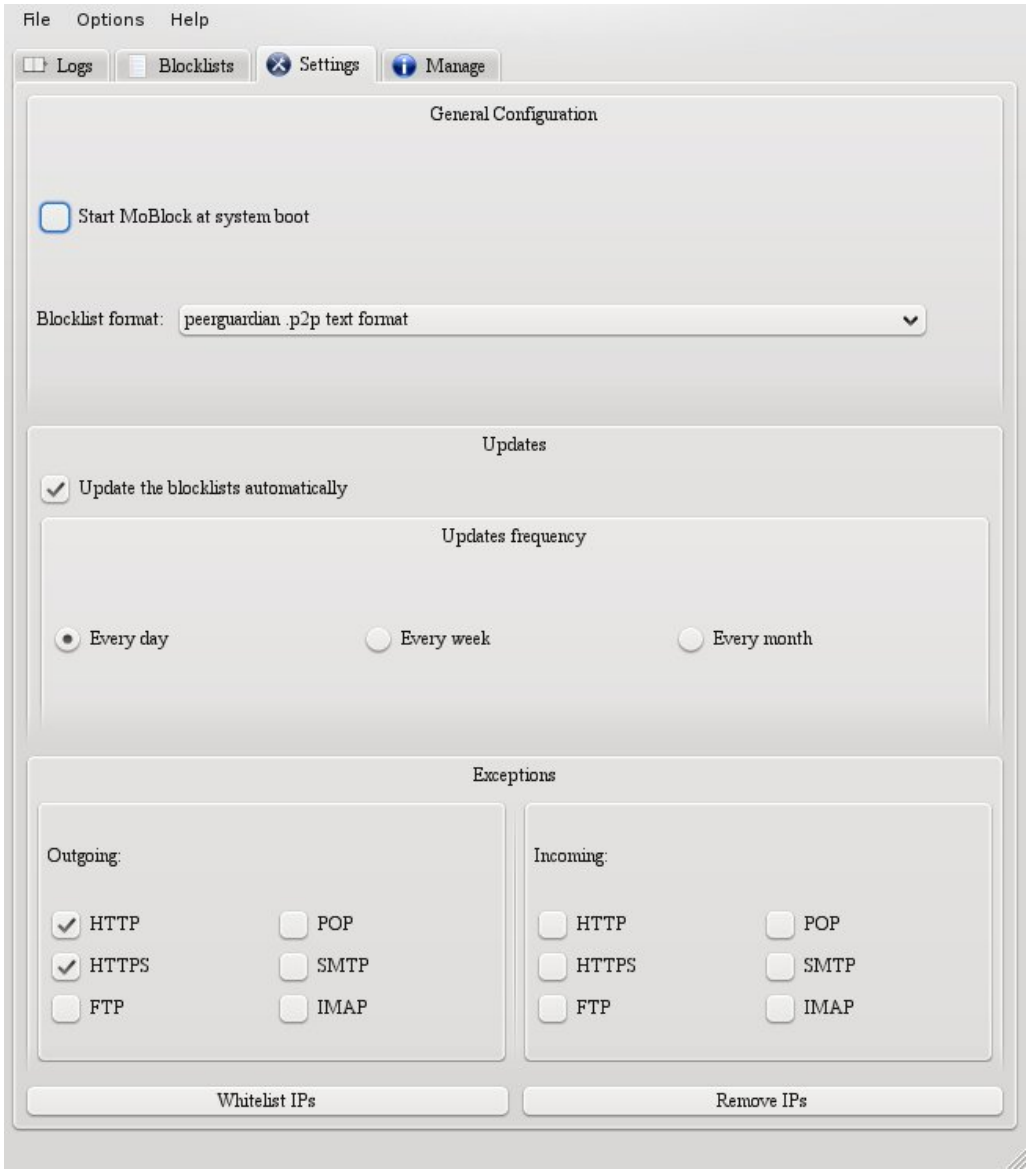
7.2 Mobloquer

Mobloquer non è altro che la comoda GUI per gestire Moblock, semplice e intuitiva. Dopo averlo eseguito troverete una comoda try-icon nella barra che vi segnalerà il suo funzionamento.

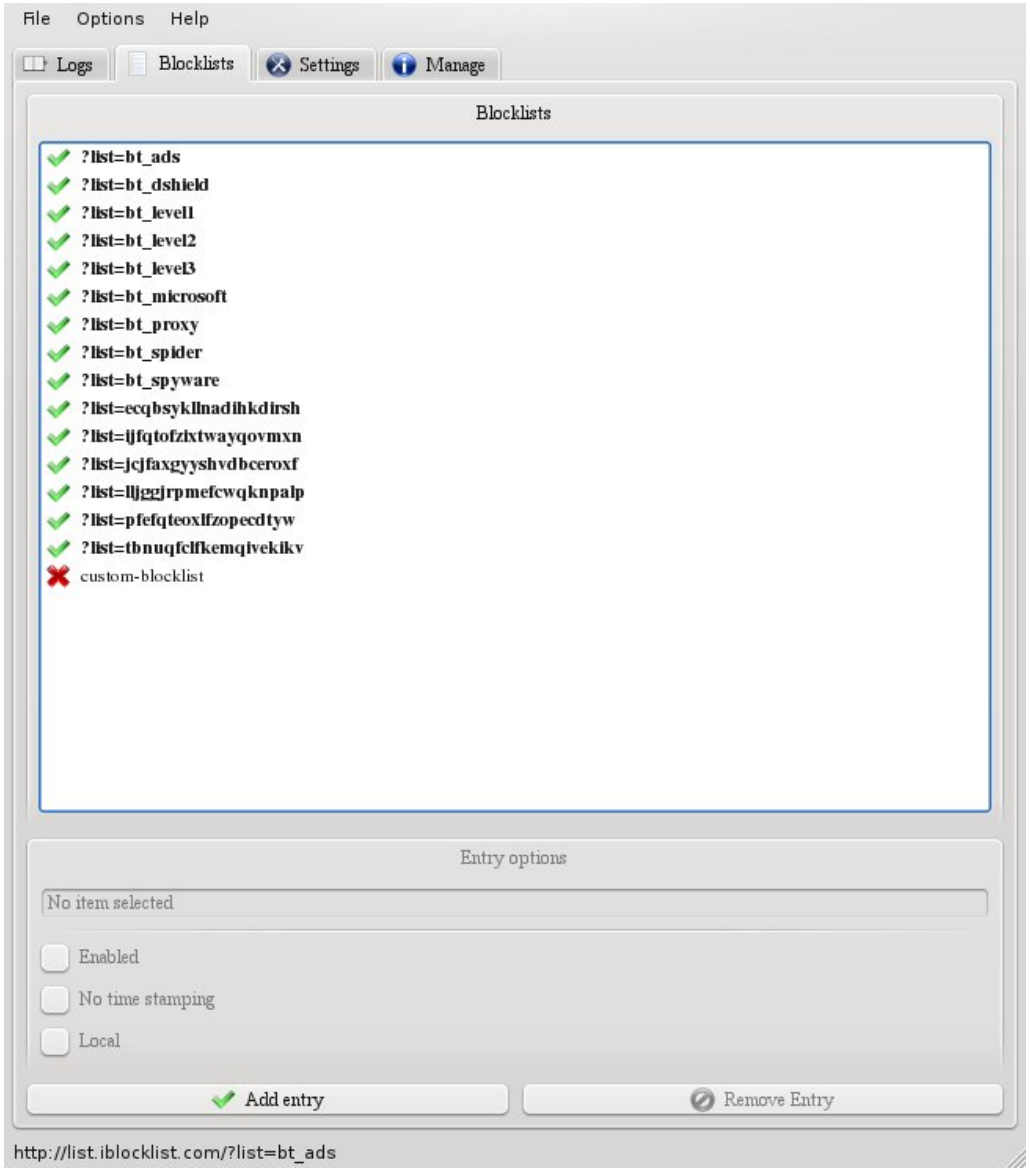
Gestione:



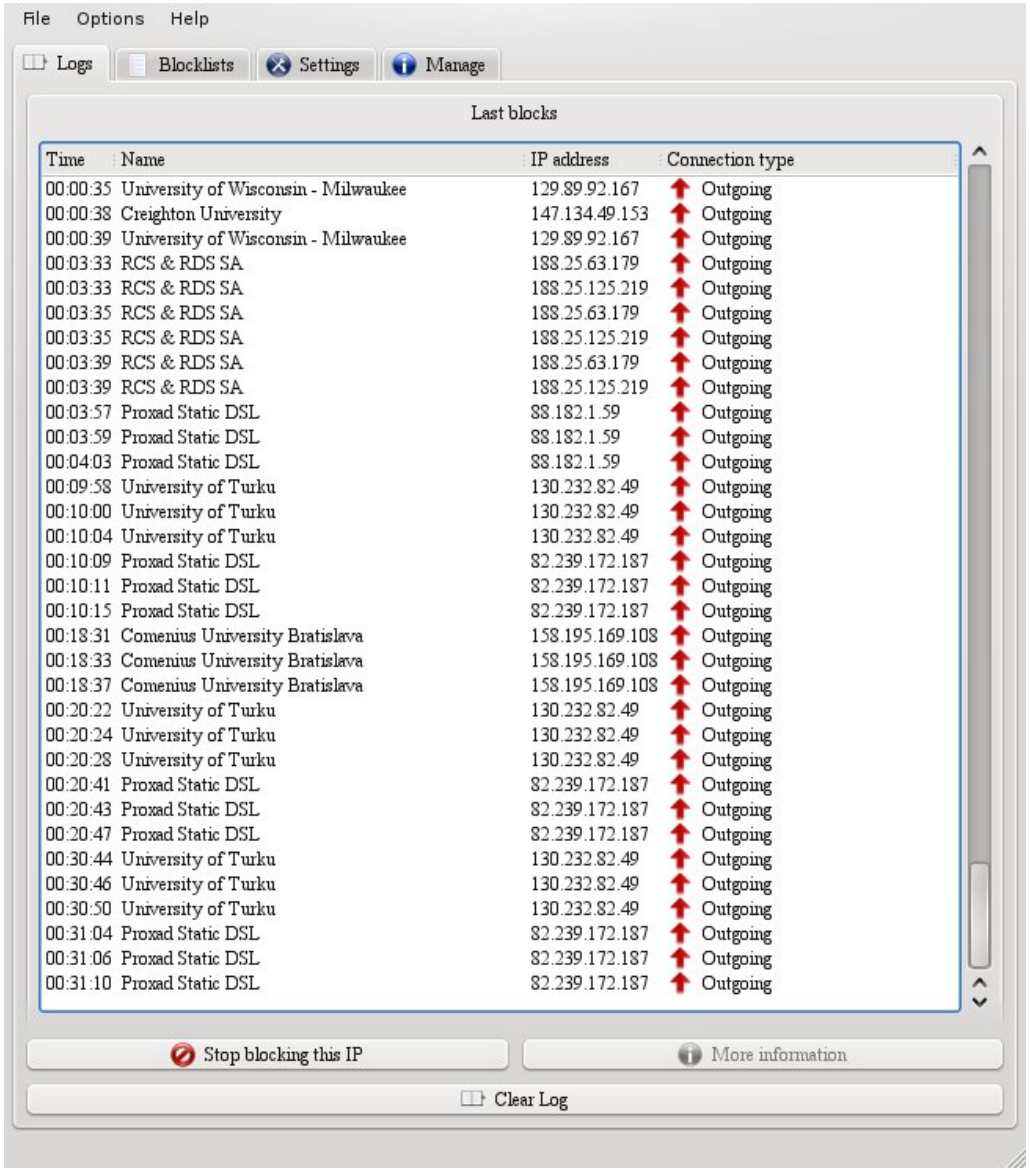
Configurazione:



Gestione liste (un po' scomoda, meglio farlo a mano):



Log (nell'immagine riporto l'azione di MoBlock con il solo skype attivo e il filtro edu):



Come potete notare, il vostro IP riceve parecchie richieste, forse più di quante immaginavate.

Dal punto di vista della sicurezza trovo MoBlock e la sua GUI, Mobloquer, molto efficaci.

7.2.1 Uso Privato

Nell'uso privato, nella normale navigazione, difficilmente entrerà in azione: occorre usare applicazioni come skype oppure per applicazioni per il P2P (tra cui amule, ktorrent e altre). Ho notato che vengono filtrati anche alcuni server FTP.
Se volete potete, dopo aver identificato l'IP corretto, aggiungerlo comodamente alla whitelist.
Nella pagina dei log, usate il pulsante Stop blocking this IP e potrete ristabilire la connessione.

7.2.2 Uso Server

Ho notato parecchi benefici usando liste specifiche per nazione, inserendo le liste in MoBlock. Alcune nazioni a parer mio pericolose e di poca utilità per il mio servizio hanno cessato la comunicazione con me, compresi parecchi tentativi di intromissione.
La lista proxy l'ho verificata personalmente e blocca tutto il traffico da proxy mentre la lista ISP nega alcune connessioni da Vodafone per quanto ho potuto provare.
Se è abilitato come servizio ed è presente sul server un servizio di MTA, automaticamente MoBlock si aggiornerà con cadenza giornaliera, inviando una mail con i risultati dell'aggiornamento, come da esempio:

```
"Updating blocklists and reloading IP block daemon: moblock.

The following lists were updated:
TBG_Primary_Threats (last modified: 2010-03-04 06:00)
TBG_General_Corporate_Ranges (last modified: 2010-03-05 06:00)
TBG_Business_ISPs (last modified: 2010-03-05 09:00)
TBG_Educational_Institutions (last modified: 2010-03-03 09:00)
TBG_Bogon (last modified: 2010-03-02 09:00)
list.iblocklist.com/?list=cn (last modified: 2010-03-08 06:28)

For the following lists there was no update available:
TBG_Search_Engines (last modified: 2010-03-02 06:26)
TBG_Hijacked (last modified: 2010-03-02 06:00)
Bluetack_level1 (last modified: 2010-03-02 06:00)
Bluetack_level2 (last modified: 2010-03-01 06:27)
Bluetack_level3 (last modified: 2010-03-02 06:00)
```

```
Bluetack_edu (last modified: 2010-03-02 06:00)
Bluetack_ads (last modified: 2010-03-01 06:24)
Bluetack_bogon (last modified: 2010-03-01 06:00)
Bluetack_spyware (last modified: 2010-03-02 06:00)
Bluetack_spider (last modified: 2010-03-01 06:00)
Bluetack_Microsoft (last modified: 2010-03-02 06:28)
Bluetack_proxy (last modified: 2010-03-01 06:00)
Bluetack_hijacked (last modified: 2010-03-02 06:00)
Bluetack_dshield (last modified: 2010-03-02 06:00)"
```

In modo che potrà essere chiara la condizione di aggiornamento delle liste e provvedere in caso di problemi.

mm-barabba

7.3 Tiger: uno strumento per l'audit di sicurezza

7.3.1 Politica di sicurezza e audit

Definizioni

Il concetto di sicurezza applicato ad un sistema informatico spazia attraverso ambiti della conoscenza molto diversi che si estendono, solo per citarne alcuni, dalle soluzioni per la protezione fisica dei dispositivi, alle politiche di gestione del personale addetto, alle norme di tutela della *privacy* e dell'integrità dei dati, fino alla regole di configurazione e manutenzione dell'hardware o del software. Queste conoscenze, nel loro insieme, possono essere utilizzate per definire un elenco di regole comportamentali e procedure a cui ci si riferisce generalmente con il termine di *politica di sicurezza* (per gli anglosassoni, *security policy*). Il significato di tale termine è, naturalmente, piuttosto generico e deve essere necessariamente calato nel contesto del sistema informatico a cui ci si riferisce, delle funzioni di controllo da esso fornite e dall'assetto organizzativo di chi ne fa uso. Qualunque sistema informatico, comunque, in linea di principio applica di fatto una politica di sicurezza per semplice ed implicita che possa essere. La valutazione organizzativa e tecnica della conformità a tale politica è generalmente definita *audit*[1].
Di seguito si riportano, senza alcuna velleità di completezza, alcuni cenni sull'evoluzione storica del concetto di politica di sicurezza che si ritiene possano essere utili a chi si avvicina a questi concetti per la prima volta in qualità di utente amatoriale; nelle poche righe sotto riportate, per brevità, si è scelto di non trattare in dettaglio l'evoluzione storica delle minacce ai sistemi informatici e della normativa nazionale di riferimento in tema di sicurezza dei sistemi informatici.

Cenni storici

L'esistenza di politiche di sicurezza, in senso lato, nell'accesso a risorse preziose e limitate è connaturata all'essere umano. Nel tempo, ha trovato diversa applicazione in funzione dell'epoca, degli ideali dominanti e delle caratteristiche tecniche dello strumento, in particolare della sua condivisibilità. Ad esempio, riferendoci all'evoluzione storica dell'informatica, nel secondo dopoguerra la valenza strategico-militare delle prime applicazioni ha imposto politiche di sicurezza estremamente restrittive nell'accesso

[1] http://it.wikipedia.org/wiki/Audit

alle risorse condivise. Allo stesso modo, al tempo, la natura sperimentale di questi dispositivi ne ha relegato l'accesso ad una cerchia estremamente ristretta di utilizzatori. Quando, a cavallo della fine degli anni sessanta e dei primi anni settanta, iniziano a diffondersi le primi applicazioni commerciali, comincia a maturare un certo interesse per questa tecnologia; ciò non di meno, le risorse informatiche sono ancora per lo più accessibili a personale estremamente qualificato, spesso operante nell'ambito della ricerca accademica: l'esigenza di limitarne l'uso è, quindi, almeno nella fase iniziale, poco sentita.

Riferendoci, ad esempio, alle architetture multi-utente in *time-sharing* dei primi anni settanta, alcuni dei protagonisti del movimento del free software che si formano usando questa tecnologia, ci raccontano che tra di loro all'epoca era implicitamente adottata un politica di sicurezza estremamente permissiva, dettata dallo spirito di ricerca e collaborazione e volta alla massima condivisione della conoscenza sul funzionamento hardware e software. Ad esempio, raccontano che l'accesso ai terminali non prevedeva l'inserimento delle credenziali utente e chiunque poteva visionare o modificare il codice sorgente realizzato da altri purché ne rimanesse traccia e contribuisse alla crescita collettiva[2].

Se è vero che la limitazione nell'accesso alla conoscenza informatica per motivi commerciali e la negazione della libertà d'uso di tale conoscenza sono considerate negativamente dal movimento del free software, è altrettanto vero che esso non ha potuto farvi fronte ai suoi esordi. Come racconta Richard Stallman, a partire dai primi anni settanta, l'evoluzione tecnica e commerciale subita dai sistemi informatici ha condotto, comunque, all'introduzione di strumenti volti a controllarne la condivisione: questa tendenza, iniziata anche nei centri di calcolo del MIT frequentati da Stallman, persiste tutt'oggi. Paradossalmente, proprio l'esistenza di tali restrizioni ha stimolato molti cultori della materia a cercare, spesso per pura sfida intellettuale, soluzioni di elusione sempre più ingegnose; l'industria cinematografica americana non ha mancato di cogliere questi aspetti, seppur drammaticizzati, in produzioni di grande impatto come Tron[3] e War Games[4].

A partire dalla fine degli anni settanta, la creazione del mercato del *personal computer* ha avvicinato all'informatica una nuova generazione di utenti. Ciò si è accompagnato

[2] http://www.theopencd.it/live/extras/books/CodiceLibero/index.html
[3] http://it.wikipedia.org/wiki/Tron
[4] http://it.wikipedia.org/wiki/Wargames_-_Giochi_di_guerra

alla diffusione di una cultura della sicurezza volta essenzialmente al controllo fisico dei dispositivi. I computer dell'epoca, infatti, nella maggior parte dei casi non erano interconnessi, molti dei primi sistemi operativi non prevedevano la *login* dell'utente, alcuni caricavano il sistema operativo da memoria a sola lettura (ROM). Infine, la diffusione su larga scala del codice sorgente dei programmi non di rado al tempo avveniva su supporto cartaceo in riviste specializzate.

A partire dalla seconda metà degli anni ottanta, parallelamente alla convergenza delle soluzioni hardware per l'interconnessione di rete, i sistemi operativi per personal computer iniziano ad ampliare le funzionalità di sicurezza mutuando, in varia misura, quelle già sperimentate nei sistemi operativi utilizzati nei centri di ricerca. In particolare, compaiono il controllo dell'accesso alla postazione, al *file system* ed alla condivisione dei dati, seppur inizialmente mediante soluzioni proprietarie.

Proprio in questo periodo, ed in particolare il 14 marzo 1994, si assiste anche al rilascio della versione 1.0 del kernel Linux[5] congiuntamente alle applicazioni fornite dal progetto GNU ereditando, di fatto, le logiche di controllo già implementate e, almeno dal punto di vista logico, collaudate nei sistemi operativi *unix-like*.

A partire dagli anni novanta, l'affermazione di internet come strumento di collegamento informatico su scala geografica ha prepotentemente riproposto le tematiche legate alle politiche di sicurezza. L'entità delle conseguenza di una minaccia su larga scala ha stimolato da una parte l'esigenza di documentare le vulnerabilità conosciute (ad esempio, attraverso database dedicati come il *National Vulnerability Database* (NVB)[6] o il *Common Vulnerabilities and Exposures* (CVE)[7]), e dall'altra quella di ben utilizzare le funzionalità di controllo esistenti per mitigare i rischi.

È proprio a partire dagli anni novanta, inoltre, che centri di ricerca e aziende iniziano a sviluppare soluzioni per automatizzare la verifica delle politiche di sicurezza e delle vulnerabilità nei sistemi informatici: *tiger* appartiene a questa famiglia di applicazioni.

[5] http://it.wikipedia.org/wiki/Linux
[6] http://nvd.nist.gov/home.cfm
[7] http://cve.mitre.org/index.html

7.3.2 Tiger

Introduzione

Lo sviluppo di *tiger* è iniziato nel 1993 al *Texas A&M University Supercomputer Center*[8], transitato attraverso almeno tre *fork* (di cui uno specifico di Debian) determinati da uno stallo della versione originaria e successivamente confluito in un unica linea di sviluppo (supportata dai repository del progetto GNU [9]). L'applicazione è distribuita come *free software* con licenza *GNU General Public License* versione 2 ed è indirizzata specificamente a sistemi *unix-like*. Al momento della stesura di questo articolo, è presente la versione 3.2.2-6 (del 9 settembre 2008) nei repository Debian per il ramo stabile (versioni più recenti sono disponibili per altri rami). È utile far presente che il manutentore del pacchetto Debian di *tiger* (*Javier Fernández-Sanguino Peña*) è anche che tra coloro che curano il ramo principale di sviluppo del programma (*upstream maintainer*)[10] oltre che essere, dal 2002, anche curatore del *Securing Debian Manual*[11].

Finalità

In linea di principio, *tiger* è stato concepito come uno strumento per l'audit di sicurezza. In particolare, esegue ciò che in gergo tecnico è definito un *internal audit* (definito anche *white box*[12]) cioè un controllo dall'interno dello stesso sistema sottoposto a verifica e del quale si presuppone di conoscere la teorica corretta configurazione; in tal senso, si distingue da altri strumenti di audit che eseguono il controllo dall'esterno del sistema controllato. È implicito che la validità dell'esito di un *internal audit* può essere pregiudicato in caso di già avvenuta *compromissione del sistema* (potenziale perdita di controllo a causa di un attacco portato a termine con successo).
tiger, in particolare, esegue il proprio compito segnalando le configurazioni del sistema operativo che possono essere fattori di rischio per un accesso indesiderato al sistema e per un suo uso con finalità diverse da quelle previste dal proprietario facendo riferimento, a tal fine, alle vulnerabilità documentate dalle autorità di certificazione sulla sicurezza informatica o segnalate dagli istituti di ricerca specializzati.

[8] http://nis.tamu.edu/Home/Security/Security_Tools/Tiger.php
[9] http://savannah.nongnu.org/projects/tiger
[10] http://savannah.nongnu.org/users/jfs
[11] http://www.debian.org/doc/manuals/securing-debian-howto/
[12] http://en.wikipedia.org/wiki/Penetration_test

Nel caso specifico, il dettaglio di tali fonti è riportato nella documentazione a corredo del pacchetto (nel file `/usr/share/doc/tiger/README.sources.gz`). Nel percorso `/usr/share/doc/tiger/`, inoltre, è disponibile un'estesa documentazione che fornisce approfondimenti sui singoli aspetti dell'utilizzo e della configurazione dell'applicazione ed alla quale, necessariamente, si è fatto riferimento anche in fase di redazione del presente articolo.
Per quanti volessero approfondire ulteriormente l'argomento, inoltre, è possibile far riferimento alla *UNIX Security Checklist* realizzata per conto del *Department of Defence* (DOD) statunitense e reperibile all'indirizzo internet
http://web.nvd.nist.gov/view/ncp/repository/checklistDetail?id=282 .

Architettura

L'architettura dell'applicazione riflette l'impostazione multi-piattaforma del progetto, ed in particolare, l'esigenza di permetterne la portabilità sul maggior numero possibile di sistemi *unix-like* conformi al Portable Operating System Interface (POSIX) (collateralmente, a titolo di mera curiosità, può essere utile segnalare che, secondo quanto indicato da Wikipedia[13], l'acronimo POSIX è stato coniato da Richard Stallman [14]). Attualmente sono supportate versioni per i seguenti sistemi operativi: AIX [15], HPUX [16], IRIX [17], Linux, NeXT[18], SunOS[19], UNICOS/UNICOSMK[20]): nel file `/usr/share/doc/tiger/README-.unsupported.gz` è possibile consultare l'elenco completo con suggerimenti su come usare *tiger* con sistemi operativi ancora non supportati.
Dal punto di vista architetturale, l'applicazione principale è stata interamente realizzata come *shell script* evitando di adottare eventuali comandi specifici per una determinata piattaforma. Tale applicazione, sulla base della configurazione impostata e della modalità di avvio, richiama, uno per volta, i moduli software incaricati di eseguire i singoli test, anch'essi realizzati come *shell script*. In tali moduli, le funzioni che non potevano essere implementate come *shell script* sono state realizzate attraverso programmi scrit-

[13] http://it.wikipedia.org/wiki/POSIX
[14] http://it.wikipedia.org/wiki/Richard_Stallman
[15] http://it.wikipedia.org/wiki/AIX_%28sistema_operativo%29
[16] http://it.wikipedia.org/wiki/HP-UX
[17] http://it.wikipedia.org/wiki/IRIX
[18] http://it.wikipedia.org/wiki/NeXT
[19] http://en.wikipedia.org/wiki/SunOS
[20] http://it.wikipedia.org/wiki/Unicos

ti in linguaggio C il cui compilatore è universalmente utilizzato per i sistemi *unix-like*. È utile far presente, inoltre, che all'interno di ogni modulo è effettuata la verifica del parametro di configurazione che ne abilita o meno l'esecuzione.
Per un elenco dettagliato delle verifiche attualmente previste nella versione di *tiger* oggetto della presente trattazione si rimanda al manuale di sistema consultabile con il comando:

```
$ man tiger
```

La parte modulare di *tiger* in GNU/Debian è installata nel percorso `/usr/lib/tiger/`, come di seguito riportato:

```
$ ls /usr/lib/tiger/ -la
totale 136
drwxr-xr-x  9 root root  4096 14 mar 11:48 .
drwxr-xr-x 88 root root 61440 14 mar 11:48 ..
drwxr-xr-x  2 root root  4096 14 mar 11:48 bin
drwxr-xr-x  2 root root  4096 14 mar 11:48 check.d
-rw-r--r--  1 root root 15260  9 set  2008 config
drwxr-xr-x  2 root root  4096 14 mar 18:49 doc
drwxr-xr-x  2 root root  4096 14 mar 11:48 html
-rw-r--r--  1 root root 12829 22 mar  2005 initdefs
drwxr-xr-x  3 root root  4096 14 mar 11:48 scripts
-rwxr-xr-x  1 root root  1918 21 apr  2003 syslist
drwxr-xr-x  4 root root  4096 14 mar 11:48 systems
lrwxrwxrwx  1 root root    17 14 mar 11:48 tigexp -> ../../sbin/tigexp
drwxr-xr-x  2 root root  4096 14 mar 11:48 util
-rw-r--r--  1 root root    24  9 set  2008 version.h
```

Può essere utile notare che tra i *file* sopra elencati:

- lo *shell script* `/usr/lib/tiger/config` contiene la configurazione dei percorsi in cui sono installati i *file* di configurazione ed i componenti del programma: è, quindi, richiamato dai singoli moduli;
- la *directory* `/usr/lib/tiger/scripts/` contiene i moduli per i controlli comuni ai diversi sistemi operativi supportati;

- la *directory* `/usr/lib/tiger/systems` contiene, al proprio interno, i moduli specifici per i diversi sistemi operativi secondo una gerarchia in cui le *sub-directory* rappresentano il nome e la versione supportata del sistema operativo come nell'esempio di seguito indicato:

```
/usr/lib/tiger
   |
   [... omissis ...]
   |
   |-systems
   |---default
   |---Linux
   |-----0
   |-------0.99.12
   |-----1
   |-------0.99.12
   |-----2
```

L'architettura modulare di *tiger*, inoltre, ne favorisce la manutenibilità (ad esempio, ciascun modulo può anche essere avviato singolarmente) e l'estensibilità: per aggiungere alla *suite* un nuovo test, è sufficiente scrivere un nuovo modulo a partire dal modello (*template*) già incluso nella documentazione (per maggiori dettagli, consultare il *file* `/usr/share/doc/tiger/README.writemodules.gz`).

Installazione

- Comandi

I comandi di seguito riportati devono essere eseguiti da una finestra di terminale di un computer dove GNU/Debian è già stato installato. Tale computer, inoltre, deve essere già stato configurato [21] per eseguire il prelevamento dei pacchetti dei programmi dai *repository* della distribuzione attraverso un collegamento internet oppure da un *repository locale* (se presente).

[21] http://people.debian.org/ osamu/pub/getwiki/html/ch02.en.html

Negli esempi di seguito riportati, i comandi impartiti con i privilegi di utente ordinario sono preceduti dal carattere '$', mentre quelli impartiti come amministratore di sistema (utente *root*) sono preceduti dal carattere '#'; si è, inoltre, fatto ricorso al comando `su` (super user[22]) per far acquisire temporaneamente i privilegi di accesso dell'utente *root* all'utente ordinario: è bene precisare che tale comando chiederà l'inserimento della *password* dell'amministratore di sistema.
Prima di procedere all'installazione è preferibile verificare lo stato di aggiornamento dei programmi installati nel sistema:

```
$ su -c "aptitude update"
$ su -c "aptitude safe-upgrade"
```

Se i comandi sopra indicati sono eseguiti con successo (senza errori) è possibile procedere con il comando successivo:

```
$ su -c "aptitude install tiger"
```

L'installazione genera il seguente output:

```
[... omissis ...]
I seguenti pacchetti NUOVI (NEW) saranno installati:
  chkrootkit{a} john{a} john-data{a} tiger
0 pacchetti aggiornati, 4 installati, 0 da rimuovere e 0 non aggiornati.
È necessario prelevare 1829kB di archivi. Dopo l'estrazione, verranno occupati 5853kB.
Continuare? [Y/n/?]
Scrittura delle informazioni sullo stato esteso... Fatto
Get:1 http://debian.fastweb.it lenny/main chkrootkit 0.48-8 [293kB]
Get:2 http://debian.fastweb.it lenny/main john-data 1.7.2-3 [648kB]
Get:3 http://debian.fastweb.it lenny/main john 1.7.2-3 [251kB]
Get:4 http://debian.fastweb.it lenny/main tiger 1:3.2.2-6 [637kB]
Scaricato 1829kB in 18s (97,2kB/s)
Preconfigurazione dei pacchetti in corso
Selezionato il pacchetto chkrootkit, che non lo era.
(Lettura del database ... 93065 file e directory attualmente installati.)
```

[22] http://en.wikipedia.org/wiki/Su_(Unix)

```
Spacchetto chkrootkit (da .../chkrootkit_0.48-8_i386.deb) ...
Selezionato il pacchetto john-data, che non lo era.
Spacchetto john-data (da .../john-data_1.7.2-3_all.deb) ...
Selezionato il pacchetto john, che non lo era.
Spacchetto john (da .../archives/john_1.7.2-3_i386.deb) ...
Selezionato il pacchetto tiger, che non lo era.
Spacchetto tiger (da .../tiger_1%3a3.2.2-6_i386.deb) ...
Processing triggers for man-db ...
Configuro chkrootkit (0.48-8) ...
Configuro john-data (1.7.2-3) ...
Configuro john (1.7.2-3) ...
il modo di '/var/run/john' è diventato 0700 (rwx------)
Configuro tiger (1:3.2.2-6) ...
[... omissis ...]
Creating config file /etc/tiger/tigerrc with new version
[... omissis ...]
Lettura delle descrizioni dei task... Fatto
```

È utile precisare che al momento della stesura del presente articolo, durante la fase di installazione del programma con GNU/Debian versione 5.04, compare un avviso relativo alla sua configurazione, che però non ne inficia l'installazione. Tale anomalia di installazione, già segnalata nel *Debian BUG Tracking System* con i numeri #517798[23] e #521620[24], risulta essere stata risolta nelle versioni successive di *tiger* attualmente disponibili per le versioni di Debian seguenti alla 5.04.

Come si può notare nell'output sopra riportato, inoltre, il pacchetto *tiger* raccomanda l'installazione di ulteriori programmi ed, in particolare, di `chkrootkit` (un rilevatore di *rootkit*[http://it.wikipedia.org/wiki/Rootkit]) e `john` (uno strumento per verificare la robustezza delle password degli utenti), il quale, a sua volta, richiede il pacchetto `john-data`. Si precisa che tali ulteriori programmi non saranno in questa sede oggetto di approfondimento, rinviando, a tal fine, il lettore alla documentazione installata nei percorsi `/usr/share/doc/rootkit` e `/usr/share/doc/john` oltre che alle rispettive pagine del manuale di sistema (comando `man`).

A completamento, può essere, utile ricordare che *tiger* prevede tra le proprie *dipendenze*[25]

[23] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=517798

[24] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=521620

[25] http://people.debian.org/ osamu/pub/getwiki/html/ch02.en.html#_package_dependencies

anche il pacchetto `sendmail` o, in alternativa, `mail-transport-agent`. La ragione di ciò va ricercata nella funzionalità di inviare i resoconti delle verifiche effettuate per posta elettronica; nel caso dell'installazione base della versione 5.04 di GNU/Debian è però predisposta in automatico l'installazione del pacchetto `exim4-daemon-light` che installa il *mail trasfer agent* (*MTA*) chiamato `exim4` il quale, a sua volta, soddisfa la dipendenza del *meta-pacchetto* `mail-transport-agent`.
Si ricorda, inoltre, che a seguito dei comandi precedentemente impartiti, sono installati solo i moduli per le verifiche specifiche di GNU/Linux; è però possibile installare il pacchetto `tiger-otheros` qualora si desiderassero aggiungere anche i moduli per gli altri sistemi operativi supportati da *tiger*.
Infine, a causa di un comportamento anomalo della procedura di installazione della versione di *tiger* oggetto del presente articolo, per completare l'installazione è necessario impartire il comando:

```
$ su -c "/usr/lib/tiger/util/genmsgidx"
```

allo scopo di generare il *file* `/usr/lib/tiger/util/genmsgidx` (che indicizza le descrizioni dettagliate delle anomalie segnalate nei resoconti) che potrà essere così utilizzato successivamente ed internamente dal programma `tigexp`. Tale anomalia, segnalata nel Debian Bug Tracking System con il numero #507028[26], è stata già risolta nella versione attualmente disponibile per il ramo *unstable* di Debian.

- File installati

Una volta completati i passaggi sopra indicati risulteranno installati i seguenti programmi per il pacchetto *tiger*:

- `/usr/sbin/tiger`: il programma da utilizzare per l'avvio manuale;
- `/usr/sbin/tigercron`: il programma usato per l'avvio attraverso lo *scheduler* di sistema;
- `/usr/sbin/tigexp`: il programma usato per ottenere informazioni di dettaglio sui resoconti generati. Inoltre, risulteranno installati i seguenti *file* di configurazione:

[26] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=507028

- `/etc/tiger/tigerrc`: il *file* di configurazione in cui sono specificati i controlli da eseguire (come variabili di *shell script* che sono successivamente utilizzate all'interno dei singoli moduli per decidere se effettuare o meno la verifica) ed altri parametri di funzionamento dell'applicazione;
- `/etc/tiger/cronrc`: il *file* di configurazione in cui sono specificati gli orari ai quali eseguire i moduli di controllo dell programma quando esso è avviato dallo *scheduler* di sistema (in particolare, attraverso il *crontab file* `/etc/cron.d/tiger`);
- `/etc/tiger/tiger.ignore`: un elenco di espressioni regolari[27] (dove ogni riga contiene una sola espressione) per disabilitare la segnalazione delle anomalie ricorrenti e ritenute non rilevanti;
- `/etc/tiger/templates`: *directory* all'interno della quale è possibile collocare i resoconti di precedenti analisi (prelevati dalla *directory* `/var/log/tiger`) che possano costituire un riferimento per le verifiche successive della stessa tipologia quando *tiger* è eseguito attraverso lo scheduler di sistema;
- `/etc/default/tiger`: *shell script* all'interno del quale è possibile definire variabili di sistema (per l'interprete della *shell*) che sono interpretate dopo quelle definite da *tiger* e che quindi possono essere usate per l' *override* di quelle di default.

Come si può desumere dall'elenco dei *file* di configurazione sopra indicato, al termine dell'installazione *tiger* è già configurato per essere avviato in automatico dallo *scheduler* di sistema e, quindi, è da subito in attività secondo la configurazione di default prevista per GNU/Debian.

Configurazione

La configurazione di *tiger* consiste essenzialmente nella attivazione o disattivazione dei moduli di verifica secondo quanto indicato nel *file* `/etc/tiger/tigerrc` del quale si riporta uno stralcio a scopo esemplificativo (un esempio della configurazione di default è consultabile nel percorso `/usr/share/doc/tiger/examples/tigerrc.gz`):

```
#
# 'rc' file for tiger.  This file is preprocessed, and thus
```

[27] http://people.debian.org/ osamu/pub/getwiki/html/ch01.en.html#_regular_expressions

```
# can *only* contain variable assignments and comments.
#
#--------------------------------------------------------------------------
#
# Select checks to perform.  Specify 'N' (uppercase) for checks
# you don't want performed.
#
[... omissis ...]
#
TigerNoBuild=Y                  # C files are corrupted (ouch.)
Tiger_Check_PASSWD=Y            # Fast
Tiger_Check_PASSWD_FORMAT=N     # Fast - not needed if on systems with pwck
Tiger_Check_PASSWD_SHADOW=Y     # Time varies on # of users
Tiger_Check_PASSWD_NIS=N        # Time varies on # of users
Tiger_Check_GROUP=Y             # Fast
Tiger_Check_ACCOUNTS=Y          # Time varies on # of users
[... omissis ...]
#
# Who gets output from 'tigercron'?
#
Tiger_Mail_RCPT="root"
[... omissis ...]
```

Come si può facilmente intuire, risultano abilitate tutte le opzioni contrassegnate con `=Y`, mentre sono disabilitate quelle contrassegnate con `=N`.
Inoltre, qualora *tiger* sia avviato automaticamente tramite lo *scheduler* di sistema, la configurazione degli orari ai quali eseguire le diverse tipologie di verifiche è contenuta nel *file* `/etc/tiger/cronrc` del quale si riporta un breve stralcio:

```
#
# Default 'tigercron' cronrc file...
#
# You can run the different checks in stages, without having to
# clutter up the crontab for root. You can do all the checks in one
# step or (like this file does) separate all the checks in different
# stages.
#
```

```
#-----------------------------------------------------------------------
#
# Field 1:  Comma separated list of hours (0-23) or '*' when this should
#           be run.
#
# Field 2:  Comma separated list of days of month or '*' when this should
#           be run.
#
# Field 3:  Comma separated list of days of week or '*' when this should
#           be run.  Days of week must be exactly as 'date' prints.
#
# Remaining fields:  Scripts to execute (no newlines or \ allowed)
#
# ----------------------------------------------------------------------
[... omissis ...]
# ----------------------------------------------------------------------
# Check for known intrusion signs every 8 hours
#
0,8,16 * *      check_known check_rootkit check_logfiles check_runprocs \
                                              check_rootdir check_root
[... omissis ...]
```

Come si può notare, ad esempio, nella configurazione di *default* le verifiche relative ai moduli `check_known`, `check_rootkit`, `check_logfiles`, `check_runprocs`, `check_rootdir`, `check_root` sono eseguite tutti i giorni del mese e della settimana alle ore 0, 8 e 16. Inoltre, quando eseguito attraverso lo scheduler, *tiger* è configurato di *default* per notificare le anomalie riscontrate tramite posta elettronica all'indirizzo dell'amministratore di sistema (Tiger_Mail_RCPT=root).

Warningbox |È utile ricordare che il *mail transfer agent* `exim` dirotta, per motivi di sicurezza, i messaggi di posta elettronica indirizzati all'amministratore del sistema locale (l'utente *root*) ad un'altra utenza che è specificata in `/etc/aliases` (consultare `/usr/share/doc/exim4/README.Debian.gz` per maggiori dettagli).

Modalità di utilizzo

tiger è generalmente utilizzabile immediatamente dopo l'installazione. Al termine di essa, infatti, lo *scheduler* di sistema (attraverso il file `/etc/cron.d/tiger`) è istruito dagli script di installazione ad eseguire automaticamente lo *shell script* `textbf/usr/sbin/tiger-cron` (la versione di *tiger* specifica per l'avvio automatico) secondo gli orari e le frequenze di avvio specificate in `/etc/tiger/cronrc` e prelevando la configurazione sulle tipologie di verifiche dal file `/etc/tiger/cronrc`. Quando avviato secondo tale modalità, *tiger* distribuisce il carico computazionale delle diverse tipologie di controlli nel tempo, avviandone generalmente uno per volta e lasciando trascorre del tempo tra l'uno e l'altro.
Esiste, inoltre, la possibilità di utilizzare manualmente *tiger*, cioé di avviare, quando ritenuto necessario, il programma (lo *shell script* `textbf/usr/sbin/tiger`) che, in tal caso, esegue in sequenza i moduli di controllo indicati come attivi nel file di configurazione `/etc/tiger/tigerrc`. Questa tipologia di utilizzo prevede che il programma sia eseguito con i privilegi dell'amministratore di sistema (l'utente *root*).
Stante la natura modulare di *tiger*, infine, c'é la possibilità di avviare manualmente, essendo *shell script*, anche i singoli moduli di controllo indipendentemente dal programma principale.

Generazione dei resoconti

Man mano che esegue i controlli *tiger* registra nel percorso `textbf/var/log/tiger` un resoconto delle segnalazioni in *formato testo* consentendone la visione solo all'amministratore di sistema. Per ciascuna segnalazione all'interno dei resoconti segue una struttura predefinita che comprende i seguenti campi separati da uno spazio:

- la gravità della segnalazione espressa nei seguenti livelli di importanza decrescente (tratto dalla pagina di manuale di `tigexp`):
  - `-ALERT-`: possibile tentativo di intrusione o un grave errore di configurazione che può esporre l'intero sistema ad attacchi;
  - `-FAIL-`: violazione di un politica di sicurezza generica o possibile intrusione;
  - `-WARN-`: problema di sicurezza che dovrebbe essere verificato e potrebbe essere correlato ad una possibile vulnerabilità o ad un'esposizione (la maggior parte dei messaggi di *tiger* appaiono in questa categoria);

  - `-INFO-`: informazioni che non sono necessariamente una violazione della politica di sicurezza, ma che sarebbe utile l'amministratore conoscesse (la segnalazione è attivata attraverso il parametro `Tiger_Show_INFO_Msgs` nel file di configurazione `/etc/tiger/tigerrc`);
  - `-ERROR-`: messaggi relativi ad errori nella esecuzione di *tiger* (o una dei suoi *script*) probabilmente causato da un errore di configurazione del programma, da un problema di installazione oppure perché un *file* necessario è mancante;
  - `-CONFIG-`: informazioni sulla configurazione del programma;

- un codice alfanumerico identificativo del tipo di anomalia (ad esempio, `[lin002i]`) e che la ricollega all'elenco delle descrizioni dettagliate delle possibili anomalie (contenute nei *file* installati nel percorso `/usr/lib/tiger/doc` e che possono essere richiamate attraverso il comando `tigexp`); da notare che l'ultima lettera del codice identificativo corrisponde alla prima del livello di gravità;

- la descrizione sintetica dell'anomalia (ad esempio, `The process 'lisa' is listening on socket 7741 (TCP) on every interface.`)

Ecco un esempio di segnalazione:

```
--WARN-- [lin002i] The process 'lisa' is listening on socket 7741 (TCP) on every  \
                                                                          interface.
```

È, inoltre, possibile

- consultare un resoconto chiedendo al programma `tigexp` di visualizzare la descrizione dettagliata di ogni anomalia segnalata utilizzando il comando:

  ```
  # tigexp -f file_contenente_il_resoconto
  ```

- analizzare i resoconti ottenendone un estratto nel quale ciascuna tipologia di anomalia è riportata, se ripetuta, una sola volta con il comando:

  ```
  # tigexp -F file_contenente_il_resoconto
  ```

- formattare un resoconto in formato HTML impostando l'opzione `-H` del programma `/usb/sbin/tiger`.

È utile in questa sede far presente che è possibile escludere sistematicamente dai resoconti le segnalazioni che sono ritenute non rilevanti; le anomalie riscontrate, infatti, sono scartate se corrispondenti almeno ad una delle espressione regolari[28] contenute in un *file* a tal fine predisposto (`/etc/tiger/tiger.ignore`).

- Avvio manuale

Quando avviato manualmente con il programma `/usr/sbin/tiger`, *tiger* genera un unico resoconto nel percorso `/var/log/tiger` il cui nome è composto dai seguenti elementi corrisponde al *pattern*:
security.report.host-name.date.time
dove gli ultimi tre campi corrispondono rispettivamente al nome assegnato al computer (*host-name*), alla data (formato YYMMDD) ed ora (formato HH:MM) in cui è iniziato il controllo; ad esempio, nel caso di avvio il 14 marzo 2010 alle ore 19:39 su un computer chiamato *debian-lenny*:

```
$ su
# cd /var/log/tiger
# ls -la security*
-rw------- 1 root root 9772 14 mar 20:15 security.report.debian-lenny.100314-19:39
```

- Avvio automatico

Quando avviato attraverso lo *scheduler* di sistema, il programma `/usr/sbin/tigercron` utilizza il seguente algoritmo:

1. nel percorso `/var/log/tiger/` è creato un resoconto per ogni modulo di controllo eseguito; esso reca il nome dello *script* che lo ha generato seguito dall'estensione `.out` e da un numero progressivo separato da un punto ('.') espressione di un meccanismo di rotazione dei log che prevede, di default, fino ad un massimo dieci log consecutivi prima di cancellare il resoconto più vecchio (nota: il numero dei livelli è configurabile attraverso il parametro `TigerCron_Log_Keep_Max` presente nei *file* di configurazione `/etc/tiger/tigerrc`);

[28] http://it.wikipedia.org/wiki/Espressione_regolare

2. è generato un *file* di confronto in cui sono evidenziate le differenze tra il resoconto sopra indicato ed uno precedente; quest'ultimo è uno dei due di seguito indicati (sono mutualmente esclusivi) in funzione di quanto specificato nel *file* di configurazione:

   (a) un precedente resoconto scelto come riferimento e creato manualmente dall'amministratore di sistema nella *directory* `/etc/tiger/templates`; questo passaggio è configurabile attraverso il parametro `Tiger_Cron_Template` del *file* `/etc/tiger/tigerrc` ed è disabilitato di *default* nella configurazione della versione qui esaminata (per maggiori dettagli è possibile consultare il *file* `/usr/share/doc/tiger/README.hostids.gz`); per ogni modulo di controllo è possibile predisporre un solo *file template* ottenibile copiando il resoconto di riferimento dal percorso `/var/log/tiger` a quello `/etc/tiger/template` sostituendone l'estensione (ad esempio `.out.1`) con `.template`;

   (b) il resoconto cronologicamente precedente; questo passaggio è attivabile attraverso il parametro `Tiger_Cron_CheckPrev` del *file* `/etc/tiger/tigerrc` ed è abilitato di *default* nella configurazione della versione qui esaminata;

3. i *file* di confronto, infine, è inviato per posta elettronica.

Si riporta di seguito il frammento di codice di `/usr/sbin/tigercron` corrispondente ai passaggi sopra indicati (i numeri di linea sono riportati per maggior chiarezza):

```
[... omissis ...]
240 do
241   newfile=$filename.1
242   previousfile=$filename.2
243   templatefile=$filename.template
244 #  etctemplatefile=‘echo $templatefile | $SED ’s/\/var\/log\/tiger\///’‘
245   etctemplatefile=‘$BASENAME $templatefile‘
246
247   if [ "$Tiger_Cron_Template" = "Y" -a -s "$TEMPLATEDIR/$etctemplatefile
247 " ]; then
248     $BASEDIR/util/difflogs $TEMPLATEDIR/$etctemplatefile $newfile
249   elif [ "$Tiger_Cron_Template" = "Y" -a -s "$templatefile" ]; then
250     $BASEDIR/util/difflogs $templatefile $newfile
251   elif [ "$Tiger_Cron_CheckPrev" = "Y" -a -s "$previousfile" ]; then
```

```
252     $BASEDIR/util/difflogs $previousfile $newfile
253   else
254     $CAT $newfile
255   fi
256 done >> $WORKDIR/tigcron.diff.$$
[... omissis ...]
273   haveallcmds MAILER && [ "$send" = "Y" ] &&  {
274         # Mail header (so it does not just say it's root
275         echo "From: \"Tiger automatic auditor at $(hostname)\"           \
                                                          <$Tiger_Mail_FROM>"
276         echo "To: $Tiger_Mail_RCPT"
277         echo "Subject: Tiger Auditing Report for $(hostname)"
278         echo
279         cat  $WORKDIR/tigcron.diff.$$
280     } | $MAILER $Tiger_Mail_RCPT
[... omissis ...]
```

7.3.3 Casi d'uso

Ambiente di test

Si è scelto di creare il seguente ambiente di test virtualizzato:

- processore intel 386;
- 512 MByte di RAM;
- scheda grafica VGA;
- un disco rigido collegato ad un controller IDE di capienza pari ad 8 GByte
- tastiera e mouse PS/2;
- scheda di rete ethernet collegata ad una rete locale con server DHCP.

Debian GNU/Linux versione 5.04 è stato installato al suo interno usando un CD-ROM recante l'immagine per *netinst* disponibile al momento della redazione e prelevata all'indirizzo:
http://cdimage.debian.org/debian-cd/5.0.4/i386/iso-cd/debian-504-i386-netinst.iso

Per completezza, si precisa che, terminata l'installazione da CD-ROM e riavviato il sistema, sono stati installati gli ulteriori seguenti pacchetti completi delle loro *dipendenze*:

- `kde` : per disporre dell'omonimo *desktop environment*;
- `build-essential` e `linux-headers-2.6.26-2-686`: per la compilazione delle estensioni dell'ambiente virtualizzato;
- `tripwire`: per tener traccia delle modifiche introdotte da *tiger* in fase di installazione) il cui database di riferimento è stato creato con il comando

```
$ su -c "tripwire -m i"
```

.

Caso 1: Avvio manuale

tiger è avviato nell'ambiente di test con il seguente comando:

```
$ su -c "tiger"
```

con il seguente risultato:

```
Password:
Tiger UN*X security checking system
   Developed by Texas A&M University, 1994
   Updated by the Advanced Research Corporation, 1999-2002
   Further updated by Javier Fernandez-Sanguino, 2001-2007
   Covered by the GNU General Public License (GPL)

Configuring...

Will try to check using config for 'i686' running Linux 2.6.26-2-686...
--CONFIG-- [con005c] Using configuration files for Linux 2.6.26-2-686. Using
           configuration files for generic Linux 2.
Tiger security scripts *** 3.2.2, 2007.08.28.00.00 ***
20:22> Beginning security report for debian-lenny.
20:22> Starting file systems scans in background...
```

```
20:22> Checking password files...
20:23> Checking group files...
20:23> Checking user accounts...
20:23> Checking .rhosts files...
20:23> Checking .netrc files...
20:23> Checking ttytab, securetty, and login configuration files...
20:23> Checking PATH settings...
20:23> Checking anonymous ftp setup...
20:23> Checking mail aliases...
20:23> Checking cron entries...
20:23> Checking 'inetd' configuration...
20:23> Checking 'tcpd' configuration...
20:23> Checking 'services' configuration...
20:24> Checking NFS export entries...
20:24> Checking permissions and ownership of system files...
20:24> Checking for indications of break-in...
20:24> Performing rootkit checks...
20:25> Performing system specific checks...
20:56> Performing root directory checks...
20:56> Checking for secure backup devices...
20:56> Checking for the presence of log files...
20:56> Checking for the setting of user's umask...
20:56> Checking for listening processes...
20:56> Checking SSHD's configuration...
20:56> Checking the printers control file...
20:56> Checking ftpusers configuration...
20:56> Checking NTP configuration...
20:56> Waiting for filesystems scans to complete...
20:56> Filesystems scans completed...
20:56> Performing check of embedded pathnames...
20:58> Security report completed for debian-lenny.
Security report is in `/var/log/tiger/security.report.debian-lenny.100322-20:22'.
```

Volendo limitare l'analisi alle sole segnalazioni di maggior ipotetica gravità contenute nel resoconto `/var/log/tiger/security.report.debian-lenny.100322-20:22` si riportano di seguito quelle con livello di gravità *failure*:

```
# Checking boot loader file permissions...
--FAIL-- [boot02] The configuration file /boot/grub/menu.lst has world
         permissions. Should be 0600
[... omissis ...]
# Checking for vulnerabilities in inittab configuration...
--FAIL-- [lin007w] Normal users can reboot the system through ctrl+alt+del in
         runlevels 12345
[... omissis ...]
# Checking network configuration
--FAIL-- [lin013f] The system is not protected against Syn flooding attacks
--FAIL-- [lin014f] The system permits the transmission of IP packets with
         invalid addresses
--FAIL-- [lin016f] The system permits source routing from incoming packets
--FAIL-- [lin019f] The system does not have any local firewall rules
         configured
[... omissis ...]
# Checking md5sums of installed files
--FAIL-- [lin005f] Installed file '/var/lib/aspell/it.compat' checksum differs
         from installed package 'aspell-it'.
--FAIL-- [lin005f] Installed file '/sbin/start-stop-daemon' checksum differs
         from installed package 'dpkg'.
[... omissis ...]
# Checking device permissions...
--FAIL-- [dev002f] /dev/log has world permissions
--FAIL-- [dev002f] /dev/vboxuser has world permissions
[... omissis ...]
# Checking for existence of log files...
--FAIL-- [logf005f] Log file /var/log/btmp permission should be 660
[... omissis ...]
# Checking sshd_config configuration files...
--FAIL-- [ssh005w] Cannot find a configuration file for SSH.
[... omissis ...]
# Performing common access checks for root...
--FAIL-- [netw020f] There is no /etc/ftpusers file.
```

Ciascuna segnalazione può e deve, naturalmente, essere ulteriormente analizzata; ad esempio, per indagare la prima segnalazione (codice identificativo univoco `boot02`) è possibile impartire il comando:

```
$ su -c "tigexp boot02"
```

che genera il seguente risultato:

```
The grub configuration file (/boot/grub/grub.conf) should have permissions
limiting access to only the owner (usually root).
```

La descrizione sopra indicata e fornita da `tigexp` è riferita al *file* `/boot/grub/grub.conf` (non specifico per la versione di grub installata di default con GNU/Debian usato in ambiente di test). In realtà, *tiger* si riferisce al *file* `/boot/grub/menu.lst`, come riportato nella segnalazione nel *file* di log. Verificando i premessi di quest'ultimo, si può controllare se la segnalazione di *tiger* è veritiera:

```
$ su -c "ls -la /boot/grub/menu.lst"
-rw-r--r-- 1 root root 3925 13 mar 15:23 /boot/grub/menu.lst
```

L'anomalia segnalata da *tiger*, quindi, può essere ulteriormente approfondita per comprenderne la motivazione e valutare l'opportunità di porvi rimedio: a tal fine, è utile far riferimento ai testi indicati nella documentazione di *tiger* come, ad esempio, alla UNIX Security Checklist[29] precedentemente citata.
Una volta deciso di correggere l'anomalia segnalata, è possibile farlo impartendo il comando:

```
$ su -c "chmod 0600 /boot/grub/menu.lst"
```

Eseguita la correzione sopra indicata, possiamo eseguire un ulteriore controllo con *tiger* secondo le modalità già esposte; in alternativa, è possibile avviare selettivamente il modulo espressamente utilizzato per tale verifica (`/usr/lib/tiger/systems/Linux/2/check_lilo`), impartendo il comando:

[29] http://web.nvd.nist.gov/view/ncp/repository/checklistDetail?id=282

```
$ su -c "/usr/lib/tiger/systems/Linux/2/check_lilo"
```

che genera il seguente risultato:

```
Configuring...

Will try to check using config for 'i686' running Linux 2.6.26-2-686...
--CONFIG-- [con005c] Using configuration files for Linux 2.6.26-2-686. Using
           configuration files for generic Linux 2.

# Checking boot loader file permissions...
--WARN-- [boot06] The Grub bootloader does not have a password configured.
```

Si può notare come la precedente segnalazione non sia più presente (pur essendo presenti, però, altre segnalazioni che, essendo di gravità inferiore di quella trattata, avevamo precedentemente escluso a priori). Anche quest'ultima segnalazione, naturalmente, è meritevole, insieme alle altre, di approfondimento.

Caso 2: Avvio automatico

Come già precedentemente riferito, *tiger* è configurato di default per essere eseguito automaticamente. I resoconti prodotti sono inviati per posta elettronica; a scopo esemplificativo, si riporta di seguito la copia dello schermo del terminale su cui è visualizzato un messaggio all'interno del *client* di posta elettronica `mutt`:
[[Immagine:mail-delivery.png]]
nel quale le anomalie riscontrate negli ultimi due controlli sono contrassegnate:

- con il prefisso *OLD* se presenti nel penultimo controllo (oppure nel *template* eventualmente specificato), ma assenti nel controllo or ora eseguito;
- con il prefisso *NEW* se presenti nel controllo or ora eseguito ed assenti nel penultimo controllo (oppure nel *template* eventualmente specificato).

Gli stessi resoconti sono registrati nel percorso `/var/log/tiger` del quale si riporta il contenuto nell'installazione di prova:

```
$ su -c "ls -la /var/log/tiger"
Password:
totale 244
drwxr-xr-x 2 root root 4096 30 mar 00:00 .
drwxr-xr-x 8 root root 4096 30 mar 20:36 ..
-rw------- 1 root root  182 30 mar 00:00 check_known.out.1
[... omissis ...]
-rw------- 1 root root  182 14 mar 16:00 check_known.out.7
-rw------- 1 root root 1207 30 mar 00:00 check_listeningprocs.out.1
[... omissis ...]
-rw------- 1 root root 1303 14 mar 18:00 check_listeningprocs.out.10
-rw------- 1 root root  111 30 mar 00:00 check_logfiles.out.1
[... omissis ...]
-rw------- 1 root root  111 14 mar 16:00 check_logfiles.out.7
-rw------- 1 root root   41 30 mar 00:00 check_rootdir.out.1
[... omissis ...]
-rw------- 1 root root   41 14 mar 16:00 check_rootdir.out.7
-rw------- 1 root root  108 30 mar 00:00 check_rootkit.out.1
[... omissis ...]
-rw------- 1 root root  108 14 mar 16:00 check_rootkit.out.7
-rw------- 1 root root   99 30 mar 00:00 check_root.out.1
[... omissis ...]
-rw------- 1 root root   99 14 mar 16:00 check_root.out.7
-rw------- 1 root root  217 30 mar 00:00 check_runprocs.out.1
[... omissis ...]
-rw------- 1 root root  217 14 mar 16:00 check_runprocs.out.7
-rw------- 1 root root 3119 15 mar 01:32 check_system.out.1
```

Impartendo i comandi:

```
$ su
# for x in $(ls /var/log/tiger/*.out.1); do echo -e '=======>' $x; cat $x;       \
                                                    echo -e '\n'; done
# exit
```

è possibile consultare il contenuto dei resoconti derivanti dall'ultima verifica eseguita che, nell'installazione del nostro caso d'uso, risulta essere:

```
=======> /var/log/tiger/check_known.out.1

# Checking for known intrusion signs...
# Testing for promiscuous interfaces with /bin/ip
# Testing for backdoors in inetd.conf

# Performing check of files in system mail spool...


=======> /var/log/tiger/check_listeningprocs.out.1

# Checking listening processes
--WARN-- [lin003w] The process 'avahi-daemon' is listening on socket 5353        \
                                          (UDP on every interface) is run by avahi.
--WARN-- [lin003w] The process 'avahi-daemon' is listening on socket 54833       \
                                          (UDP on every interface) is run by avahi.
--WARN-- [lin002i] The process 'inetd' is listening on socket 517 (UDP) on every \
                                                                        interface.
--WARN-- [lin002i] The process 'inetd' is listening on socket 518 (UDP) on every \
                                                                        interface.
--WARN-- [lin002i] The process 'lisa' is listening on socket 7741 (TCP) on every \
                                                                        interface.
--WARN-- [lin002i] The process 'lisa' is listening on socket 7741 (UDP) on every \
                                                                        interface.
--WARN-- [lin003w] The process 'portmap' is listening on socket 111 (TCP on every\
                                                    interface) is run by daemon.
--WARN-- [lin003w] The process 'portmap' is listening on socket 111 (UDP on every\
                                                    interface) is run by daemon.
--WARN-- [lin003w] The process 'rpc.statd' is listening on socket 36600 (TCP on  \
                                                 every interface) is run by statd.
--WARN-- [lin003w] The process 'rpc.statd' is listening on socket 44997 (UDP on  \
                                                 every interface) is run by statd.
--WARN-- [lin003w] The process 'rpc.statd' is listening on socket 861 (UDP on    \
                                                 every interface) is run by statd.
```

```
=======> /var/log/tiger/check_logfiles.out.1

# Checking for existence of log files...
--FAIL-- [logf005f] Log file /var/log/btmp permission should be 660


=======> /var/log/tiger/check_rootdir.out.1

# Performing check of root directory...


=======> /var/log/tiger/check_rootkit.out.1

# Performing check for rookits...
# Running chkrootkit (/usr/sbin/chkrootkit) to perform further checks...


=======> /var/log/tiger/check_root.out.1

# Performing common access checks for root (in /etc/default/login, /securetty,   \
                                                                  and /etc/ttytab...


=======> /var/log/tiger/check_runprocs.out.1

# Checking running processes
--FAIL-- [misc020f] The process 'syslogd' has not been found running in the      \
                                                                    processes table.
--FAIL-- [misc020f] The process 'klogd' has not been found running in the        \
                                                                    processes table.


=======> /var/log/tiger/check_system.out.1

# Performing system specific checks...
# Performing checks for Linux/2...
```

```
# Checking for single user-mode password...

# Checking boot loader file permissions...
--WARN-- [boot02] The configuration file /boot/grub/menu.lst has group permissions.\
                                                                    Should be 0600
--FAIL-- [boot02] The configuration file /boot/grub/menu.lst has world permissions.\
                                                                    Should be 0600
--WARN-- [boot06] The Grub bootloader does not have a password configured.

# Checking for vulnerabilities in inittab configuration...
--FAIL-- [lin007w] Normal users can reboot the system through ctrl+alt+del in     \
                                                                  runlevels 12345

# Checking for correct umask settings for init scripts...
--WARN-- [misc021w] There are no umask entries in /etc/init.d/rcS

# Checking Logins not used on the system ...

# Checking network configuration
--WARN-- [lin012w] The system accepts ICMP redirection messages
--FAIL-- [lin013f] The system is not protected against Syn flooding attacks
--FAIL-- [lin014f] The system permits the transmission of IP packets with invalid \
                                                                       addresses
--FAIL-- [lin016f] The system permits source routing from incoming packets
--WARN-- [lin017w] The system is not configured to log suspicious (martian) packets
--FAIL-- [lin019f] The system does not have any local firewall rules configured

# Verifying system specific password checks...

# Checking OS release...

# Checking installed packages vs Debian Security Advisories...

# Checking md5sums of installed files
--FAIL-- [lin005f] Installed file `/var/lib/aspell/it.compat' checksum differs    \
                                              from installed package 'aspell-it'.
--FAIL-- [lin005f] Installed file `/sbin/start-stop-daemon' checksum differs from \
                                                    installed package 'dpkg'.
```

```
# Checking installed files against packages...
--WARN-- [lin001w] File ‘/lib/init/rw/.ramfs’ does not belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.symbols’ does not      \
                                                              belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.ieee1394map’ does not  \
                                                              belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.pcimap’ does not belong\
                                                                  to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.isapnpmap’ does not    \
                                                              belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.seriomap’ does not     \
                                                              belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.usbmap’ does not belong\
                                                                  to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.inputmap’ does not     \
                                                              belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.ccwmap’ does not belong\
                                                                  to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.alias’ does not belong \
                                    to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.dep’ does not belong to\
                                                                     any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/modules.ofmap’ does not belong \
                                                                  to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/misc/vboxguest.ko’ does not    \
                                                               belong to any package.
--WARN-- [lin001w] File ‘/lib/modules/2.6.26-2-686/misc/vboxvfs.ko’ does not      \
                                                               belong to any package.
```

Caso 3: Disabilitazione di una segnalazione

Qualora si desiderasse disabilitare una tipologia di segnalazione tra quelle prodotte da *tiger*, è possibile farlo aggiungendo al *file* `/etc/tiger/tiger.ignore` un riga contenente un' *espressione regolare*[30] corrispondente ad essa. Ad esempio, qualora si desiderassero escludere le segnalazioni relative al programma *portmap* di seguito elencate:

[30] http://it.wikipedia.org/wiki/Espressione_regolare

```
--WARN-- [lin003w] The process ‘portmap’ is listening on socket sunrpc (TCP on
         every interface) is run by daemon.
--WARN-- [lin003w] The process ‘portmap’ is listening on socket sunrpc (UDP on
         every interface) is run by daemon.
```

è possibile aggiungere al *file* `/etc/tiger/tiger.ignore` una riga contenente l'espressione:

```
The process ‘portmap’ is listening on socket
```

Qualora si desiderasse una selezione ancora più mirata, è possibile usare espressioni corrispondenti al contenuto dell'intera segnalazione purché sia rispettata la sintassi delle *espressioni regolari*; ad esempio, l'espressione per escludere le segnalazioni sopra riportate può anche essere così scritta:

```
The process .portmap. is listening on socket [A-Za-z0-9]+ \(TCP|UDP on every      \
                                                        interface\) is run by daemon.
```

dove i caratteri ‘ e ’ sono sostituiti con il `.` , la parentesi tonda aperta con `(` e quella chiusa con `)`, l'identificazione del *socket* (che può essere reso in formato numerico o alfanumerico) con l'espressione `[A-Za-z0-9]+` ed, infine, l'uso mutualmente esclusivo del protocollo TCP e UDP con l'espressione `TCP|UDP`.
Si ricorda che per modificare il *file* `/etc/tiger/tiger.ignore` è necessario disporre dei privilegi dell'amministratore di sistema (l'utente *root*).
Nel nostro esempio, dopo aver aggiunto una delle due espressioni sopra riportate al *file* `/etc/tiger/tiger.ignore`, è possibile eseguire, per ulteriore verifica, il modulo che si occupa specificamente di tale controllo con il comando:

```
# /usr/lib/tiger/scripts/check_listeningprocs
```

che, dopo la variazione sopra riportata, produce nella nostro caso d'uso il seguente resoconto:

```
Configuring...

Will try to check using config for 'i686' running Linux 2.6.26-2-686...
--CONFIG-- [con005c] Using configuration files for Linux 2.6.26-2-686. Using
           configuration files for generic Linux 2.

# Checking listening processes
--WARN-- [lin003w] The process `avahi-dae' is listening on socket 37446 (UDP
         on every interface) is run by avahi.
--WARN-- [lin003w] The process `avahi-dae' is listening on socket 45302 (UDP
         on every interface) is run by avahi.
--WARN-- [lin003w] The process `avahi-dae' is listening on socket mdns (UDP on
         every interface) is run by avahi.
--WARN-- [lin003w] The process `inetd' is listening on socket ntalk (UDP on
         every interface) is run by root.
--WARN-- [lin003w] The process `inetd' is listening on socket talk (UDP on
         every interface) is run by root.
--WARN-- [lin003w] The process `lisa' is listening on socket 7741 (TCP on
         every interface) is run by root.
--WARN-- [lin003w] The process `lisa' is listening on socket 7741 (UDP on
         every interface) is run by root.
--WARN-- [lin003w] The process `lisa' is listening on socket (hex) 0001 (raw
         on every interface) is run by root.
--WARN-- [lin003w] The process `rpc.statd' is listening on socket 51193 (TCP
         on every interface) is run by statd.
--WARN-- [lin003w] The process `rpc.statd' is listening on socket 53263 (UDP
         on every interface) is run by statd.
--WARN-- [lin003w] The process `rpc.statd' is listening on socket 854 (UDP on
         every interface) is run by statd.
```

nel quale, come si può notare, è assente la segnalazione che intendeva escludere. Warningbox | anche una sola riga vuota nel *file* `/etc/tiger/tiger.ignore` disabilita tutte le possibili segnalazioni di *tiger*; è quindi importante evitare di farlo

7.3.4 Discussione

Come accennato in premessa, la sicurezza di un sistema informatico è ottenuta con un approccio multidisciplinare: è il risultato dell'applicazione di una politica di sicurezza che gli utenti sono tenuti a rispettare e su cui gli amministratori di sistema sono tenuti a vigilare. Poiché la politica di sicurezza definisce regole che si estendono ad ambiti molto diversi della conoscenza, gli strumenti a disposizione dell'amministratore di sistema sono necessariamente eterogenei, ciascuno orientato a specifici campi di applicazione.
tiger è, quindi, solo uno di questi strumenti ed espleta il proprio compito verificando il rispetto di alcune regole di configurazione basilari pur potendo estendere il proprio raggio d'azione avvalendosi di altre applicazioni (come i rilevatori di *rootkit*, i sistemi di cracking pro-attivo delle *password* oppure i sistemi di monitoraggio delle alterazioni indesiderate del *filesystem*).
tiger è uno strumento diagnostico e, come tale, deve essere usato con raziocinio sapendo che i risultati dei test sono da analizzare alla luce della politica di sicurezza adottata e delle eventuali altre verifiche eseguite. Ad esempio, *tiger* è istruito a segnalare l'eventuale mancanza della password di accesso alla configurazione del *boot-loader* che, di per sé, costituisce la violazione di una regola di sicurezza semplice, ma che può avere effetti importanti quando l'accesso alla *console di sistema* è condiviso; ciò nonostante, la politica di sicurezza del sistema potrebbe ignorare questo controllo in quanto soddisfatto da altri è più stringenti controlli. Inoltre, i singoli resoconti di *tiger* dovrebbero essere analizzati alla luce della documentazione esistente per il sistema informatico analizzato: nel caso di Debian, a tal fine, si può certamente far riferimento ai contenuti della URL http://www.debian.org/security/ ed, in particolare, al già citato manuale *Securing Debian*[31].
Inoltre, è bene ricordare che qualunque strumento diagnostico può segnalare falsi positivi (situazioni indicate come degne di interesse che in realtà non lo sono) e falsi negativi (situazioni che meriterebbero attenzione che però non sono né rilevate né segnalate). Per quanto attiene ai falsi positivi, si richiama quanto precedentemente indicato: le segnalazioni devono essere analizzate ed interpretate alla luce della documentazione esistente in tema di sicurezza; ad esempio, nel caso del resoconto del caso d'uso sopra riportato, sono presenti alcune segnalazioni (come quelle di alcuni processi attivi in ascolto sulle porte di rete o la presenza di alcuni *file* che apparentemente non appartengono ad alcun

[31] http://www.debian.org/doc/user-manuals#securing

pacchetto) che, di per sé, potrebbero essere considerati falsi positivi, ma che in realtà richiedono un ulteriore approfondimento (ad esempio, se è segnalato attivo un processo che è stato precedentemente disabilitato manualmente potrebbe essere indicativo della presenza di codice malevolo che cerca di mettersi in comunicazione con l'esterno). Analogo discorso può essere fatto per i falsi negativi; l'assenza di segnalazioni non implica che necessariamente il sistema sia in assoluta sicurezza: come riportato nella pagina di manuale di *tiger*, il suo principale limite è che si tratta di un programma in continua evoluzione così come le situazioni a rischio che intende rilevare.
Quindi, rispetto al caso d'uso precedentemente descritto, si lascia al lettore l'esercizio di valutare le segnalazioni del programma, giudicare se costituiscono un rischio per il proprio sistema e, in caso affermativo, introdurre i necessari correttivi.
Un ultimo aspetto, infine, di cui tener conto sono le prestazioni del programma: se eseguito manualmente, infatti, *tiger* assorbe le risorse di sistema in maniera consistente e richiede un tempo di esecuzione rilevante (come è possibile vedere nei log riportati nei precedenti paragrafi) che è funzione, naturalmente, delle prestazioni del sistema interessato; qualora utilizzato attraverso lo *scheduler* di sistema, invece, l'impatto sulle prestazioni generali del sistema è più contenuto in quanto i test sono eseguiti ad una certa distanza di tempo l'uno dall'altro.

7.3.5 Conclusione

tiger è uno strumento molto utile per eseguire sistematicamente l'audit di sicurezza della configurazione di base di un'installazione Debian. Offre, per questo motivo, anche l'occasione di analizzare in dettaglio il funzionamento del sistema studiato ed, in tal senso, è indirettamente un'interessante risorsa didattica per l'uso amatoriale. È sempre bene, però, ricordare che è solo uno degli strumenti diagnostici tra quelli a disposizione dell'amministratore di sistema. Prima di intraprendere azioni correttive, inoltre, i resoconti prodotti devono essere analizzati alla luce di una metodologia diagnostica che affonda le proprie radici sia nella conoscenza della documentazione specifica del programma che in quella più generale della distribuzione Debian GNU/Linux:in tal senso, si raccomanda la lettura del manuale *Securing Debian*[32] prima di procedere all'utilizzo di *tiger* per la messa in sicurezza di un'installazione Debian.

Aki

[32] http://www.debian.org/doc/user-manuals#securing

Capitolo 8

Il kernel GNU/Linux

Il Kernel Linux viene inventato da Linus Torvalds nel 1991. Visti i ritardi nello sviluppo del sistema Hurd (basato sul mikrokernel mach), Torvalds, come dice lui stesso "per divertirsi", programmò il suo kernel monolitico "quasi per caso". Il punto di forza del Kernel è sicuramente la comunità che si creò. Ogni appassionato diede il suo contributo allo sviluppo di quello che stava per diventare un sistema operativo a tutti gli effetti. Nel 1992 Linus Torvalds decise di distribuire il suo progetto con licenza GPL. La free software foundation, che proprio non riusciva a rendere stabile il suo Hurd, adottò il kernel Linux per il suo sistema operativo GNU: nacque GNU/Linux.

8.1 Introduzione ai Kernel: prima parte

8.1.1 Introduzione

Al giorno d'oggi il Personal Computer è una costante quotidiana della nostra vita, lo utilizziamo per lavorare, comunicare, per ottenere informazioni o anche solo per svago. Molti dimenticano che un PC consiste in un complesso macchinario con uno scopo ben preciso nonché assai complicato: coprire una moltitudine di ruoli molto differenti gli uni dagli altri.

Con la televisione otteniamo notizie e svago in formato audiovisivo, ascoltiamo la musica con l'ausilio di lettori di vario tipo e utilizziamo il cellulare per comunicare a grandi distanze; il computer è in grado di svolgere tutte queste mansioni spesso con una profondità maggiore di quella dei dispositivi sopracitati. Il Personal Computer è una macchina *General Purpose* che deve offrire ottime prestazioni in tutti i suoi campi con tempi di risposta mediamente buoni e con una estrema facilità d'uso.

Questo articolo tratterà in modo abbastanza approfondito, cercando di non sacrificare la semplicità del linguaggio, la progettazione di un Sistema Operativo in quest'ottica, mostrando le principali difficoltà che occorre affrontare nella realizzazione di uno strumento così complesso; nel prossimo numero dell'e-zine Debianizzati l'attenzione verrà invece focalizzata sui kernel, strumenti essenziali di cui si parla spesso (in ambiente GNU/Linux) ma di cui si conosce ben poco.

Per trattare un argomento così vasto e complesso occorre introdurre alcuni metodi di progettazione tipici dell'Ingegneria Informatica, metodi non molto complicati a dire il vero, ma che spesso sono dati per scontati ignorandone le meccaniche e soprattutto, l'efficacia. Vedremo insieme vari concetti cercando di affacciarci ad essi non dal punto di vista del profano, ma dalla prospettiva di un progettista in erba che vuole costruire un Sistema Operativo.

8.1.2 Scienza dell'astrazione

Il cervello umano ha capacità limitate per quanto concerne la gestione di problemi con un elevato numero di variabili in gioco. Si pensi ad un'automobile moderna, se il guidatore dovesse controllare ogni singola componente del motore (carburazione, iniezione, apertura e chiusura delle valvole) difficilmente riuscirebbe ad avviare il veicolo.

Occorre perciò introdurre uno strumento in grado di semplificare il sistema rendendolo umanamente trattabile. Si analizza approfonditamente il problema e si elabora un modello in grado di automatizzare gran parte delle procedure interne, dopodichè si restitiusce all'utente una interfaccia semplificata che gli consenta il controllo dell'intero sistema. Per tornare all'esempio dell'automobile, l'automazione delle procedure è affidata ad una centralina elettronica mentre l'interfaccia è l'insieme a noi ben noto: frizione, freno, accelleratore, volante e leva del cambio.
I vantaggi di questo approccio sono molteplici. Innanzitutto al guidatore non è necessario conoscere bene il motore del proprio veicolo per poterlo controllare, questo svincola l'utente da molte complicazioni semplificandogli la vita e permettendogli di concentrarsi esclusivamente sul proprio compito. La conseguenza è l'introduzione di un approccio modulare il quale permette agli sviluppatori di concentrarsi solamente nel proprio ambito; otterremo così piloti eccellenti e ottimi meccanici. Diviene possibile introdurre meccanismi di protezione e mascheramento che impediscano all'utente di accedere direttamente alle variabili del sistema causando gravi danni (immaginate di aprire una valvola nel cilindro al momento sbagliato). Infine l'evoluzione di una tecnologia non costringe necessariamente l'utenza ad un aggiornamento, i motori cambiano ma le automobili si guidano sempre nello stesso modo, lo sviluppo dell'interfaccia è del tutto svincolato dallo sviluppo del sistema.
Questo metodo noto come astrazione o modello è applicabile infinite volte. Si può quindi strutturare un problema molto complesso come la gestione di un computer in un insieme di strati o moduli i quali sfruttano questo strumento per trarne il massimo vantaggio. Ogni strato utilizzerà l'interfaccia sottostante per produrre servizi utili allo strato superiore sempre tramite una apposita interfaccia. Vi sarà quindi uno strato 0, composto da software a basso livello (generalmente linguaggio *assembly*) che si appoggerà direttamente sull'hardware della macchina e lo strato N composto dai software applicativi di cui facciamo uso tutti i giorni; gli strati intermedi sono costituiti da software e librerie atte a semplificare via via l'architettura del calcolatore.
Il principale svantaggio di questa strategia è l'introduzione di un notevole *overhead* nel caso in cui ci si trovi ad un certo livello della stratificazione e si voglia utilizzare un servizio offerto da uno strato molto più in basso; se un applicativo che risiede nel livello N richiede un servizio a livello 0 dovrà accedere al livello N-1 tramite l'apposita interfaccia, il livello N-1 farà lo stesso con lo strato inferiore e così via. Ciò implica che, qualora ci occorra usufruire di un servizio residente in M strati più in basso, sarà necessario

passare per tutti gli M strati intermedi.
La seconda problematica risiede nella definizione dei moduli stessi e nella loro collocazione logica. Ad esempio, come collochiamo la gestione della memoria rispetto a quella del processore? Potremmo identificare il processore come entità primitiva di un calcolatore; a questo punto il gestore della memoria si dovrebbe appoggiare al gestore del processore per poter eseguire le proprie *routine* (allocazione degli spazi e relativo indirizzamento). Se però ci ritrovassimo in una situazione in cui un programma risiedente all'interno della CPU dovesse essere sospeso forzatamente per fare spazio ad uno più urgente, il gestore del processore dovrebbe far riferimento al gestore della memoria per poter mettere il programma da rimuovere in una locazione temporanea; ciò non è però possibile poichè uno strato non può essere a conoscenza degli strati a lui superiori. Contraddizioni come queste sono molto frequenti, esse fanno parte delle prime difficoltà che si incontrano nella progettazione di architetture così complesse.
Questo è il principio che sta alla base della strutturazione di un Sistema Operativo. È grazie a questo strumento, l'astrazione, che è possibile passare da un insieme di fili percorsi da corrente ai programmi che popolano il nostro quotidiano. Se ci guardiamo attorno abbiamo moltissimi esempi di astrazione di un problema, si pensi al televisore e al relativo telecomando, al telefono e alla sua tastiera o al lettore di CD.

8.1.3 Il Sistema Operativo

Ora che ci siamo appropriati degli strumenti adatti per proseguire nel nostro cammino dobbiamo fare un passo indietro. In qualunque progetto prima di sviluppare i meccanismi e i criteri giusti da adottare è necessario definire bene ambiti e obiettivi. La domanda che non ci siamo ancora posti è: cosa vogliamo fare? Questo quesito nel nostro contesto è traducibile in: che cos'è un Sistema Operativo?

8.1.4 Definizione di Sistema Operativo

Come accade in molti settori non esiste una definizione generale, completa ed esauriente di Sistema Operativo. Prendiamo come esempio una macchina a controllo numerico e un computer palmare. Essi avranno Sistemi Operativi del tutto imparagonabili. Lo stesso discorso è fattibile riducendo il campo ai soli *Personal Computer*: c'è chi ne fa un uso prevalentemente da ufficio, chi utilizza intensamente programmi multimediali, chi ama giocare ai videogiochi; ovviamente potremmo portare altri innumerevoli esempi.

Risulta difficile persino definire cosa faccia parte di un Sistema Operativo e cosa vada identificato come software opzionale, molti limitano la definizione al solo kernel, altri includono l'intero pacchetto di software applicativi forniti dal rivenditore al momento dell'acquisto della macchina. La questione ha preso rilevanza anche dal punto di vista legale; si pensi all'azione promossa dal Dipartimento di Giustizia degli Stati Uniti nei confronti di Microsoft: l'accusa era di includere troppi software applicativi di default nel Sistema Operativo Windows (Internet Explorer per la navigazione web ad esempio) applicando di fatto una forma di concorrenza sleale nei confronti di molte *software house*. [1]

Ci riduciamo quindi a dare una definizione quanto più generica e completa possibile di Sistema Operativo senza poter scendere nei particolari: Il Sistema Operativo è un insieme di software applicati sulla macchina che hanno lo scopo di estenderne l'architettura e le funzioni, di gestirne le risorse e nel contempo, di proteggerle da un uso scorretto da parte dell'utente.

8.1.5 Obiettivi di un Sistema Operativo

Possiamo osservare un Sistema Operativo sotto diversi punti di vista; distinguiamo in questo articolo tre aspetti fondamentali nella riuscita di un buon progetto.

- Interfaccia utente-macchina

Vediamo per cominciare che percezione di Sistema Operativo ha un normale utente.
Se analizzassimo la struttura di un microprocessore noteremmo che esso è in grado di eseguire solamente poche operazioni logiche ed aritmetiche, inoltre ci accorgeremmo che la sua architettura risulta composta da un elevato numero di registri specifici e generici molto complessi da utilizzare. Il principale scopo del Sistema Operativo è di astrarre le varie componenti del calcolatore restituendoci una vera e propria macchina virtuale estesa, molto più semplice e con molte più funzionalità della macchina originale. Utenti e programmatori ad alto livello vengono sollevati da tutti gli aspetti gestionali della macchina; con essa infatti possono solo comunicare utilizzando il Sistema Operativo come tramite. I meccanismi con cui utente e calcolatore interagiscono sono noti come

[1] Original Complaint Against Microsoft
http://usgovinfo.about.com/gi/dynamic/offsite.htm?site=http://www.usdoj.gov/atr/cases/f1700/1763.htm

procedure di *Input* e *Output* (più comunemente I/O). I termini informatici *Input* e *Output* sono oramai noti a tutti, non è quindi necessario dare spiegazioni o esempi; vediamo invece quali sono le complicazioni che insorgono dal punto di vista progettuale.
Stiamo mettendo in comunicazione un essere umano con una macchina elettronica. Queste due entità presentano velocità di elaborazione incredibilmente differenti; inoltre l'essere umano è in grado di fare molti errori (addirittura troppi). Occorre dunque tener ben presente questi due aspetti. La frequenza con cui utente e macchina comunicano è importante: abbiamo *I/O bounded job* dove gli utenti svolgono mansioni in cui è necessaria una elevata comunicazione (un videogioco o la visione di un film) e *CPU bounded job* dove invece sono necessarie poche direttive e molto tempo di calcolo (ad esempio la conversione di un filmato in Divx). In questo ambito sono molto importanti i tempi di risposta (anche noti come latenza di I/O). Immaginate quanto possa essere fastidiosa la visione di un film a scatti, mentre se convertiamo un file mp3 i tempi di risposta al segnale di inizio della conversione sono del tutto relativi; ben più importante sarà il tempo impiegato a concludere l'operazione. L'ultimo aspetto riguarda le diverse tipologie di dispositivi di I/O: ognuno di essi necessita di software specifico per funzionare e implementa una interfaccia diversa fra utente e macchina. Ciascuno dei dispositivi deve lavorare in maniera indipendente dagli altri e problematica molto importante, è possibile che più processi cerchino di avere accesso allo stesso dispositivo contemporaneamente.
Unix astrae tali dispositivi in files differenti. Alla creazione di un processo in Unix vegono aperti di default 3 files:

1. stdin: (standard input), la tastiera
2. stout: (standard output), lo schermo
3. sterr: (standard error), può essere lo schermo oppure un file residente su disco fisso.

- Gestione delle risorse

Approciamoci ora al Sistema Operativo dal punto di vista della macchina.
Visto dal calcolatore il Sistema Operativo è il software a lui più vicino, più strettamente correlato ai dispositivi che lo compongono; in tale contesto possiamo quindi considerarlo come un assegnatore di risorse. Ogni componente della macchina può infatti essere

visto come una risorsa da assegnare ai vari *task* in esecuzione; abbiamo quindi la risorsa tempo processore, la risorsa memoria centrale, più risorse memoria di massa e via discorrendo.
È compito del Sistema Operativo gestire nel migliore dei modi tali risorse, cercando di massimizzare le operazioni eseguibili in parallelo minimizzandone nel contempo la durata. Questo è chiaramente l'aspetto più complesso e interessante della progettazione di un Sistema Operativo. È richiesta una profonda conoscenza dell'architettura su cui ci si va ad appoggiare oltre che una certa dimestichezza con la programmazione.
Vediamo insieme quali strumenti ci offrono le moderne architetture hardware.
La CPU oltre a svolgere il compito di elaboratore generico ricopre un ruolo essenziale all'interno del calcolatore. Grazie alla teoria nota come *Sistema a Transizione degli Stati* (a volte definita *Macchina a Stati*) il processore è in grado di prendere decisioni basandosi sulle informazioni di cui è in possesso e di quelle che gli pervengono dall'esterno. Tutto ciò si traduce nel meccanismo di *interrupt*. Una *interrupt* (dall'inglese interruzione) è un segnale software o hardware diretto alla CPU.
I moderni Sistemi Operativi fanno ampio uso delle *interrupt*: a ogni pressione del tasto invio su terminale viene generata un' *interrupt software*, quando scollegate il cavo d'alimentazione del vostro portatile il relativo dispositivo genera un' *interrupt hardware*, tutte le procedure di I/O sono regolate interamente da *interrupt*. In questo modo il Sistema Operativo è in grado di operare i provvedimenti corretti al presentarsi di un qualsiasi evento. In un'area specifica (protetta) della memoria sono salvate le istruzioni da eseguire (*Interrupt Service Routine*) per ogni relativo segnale; all'arrivo di un' *interrupt* la CPU interrompe il *task* in esecuzione ponendolo temporaneamente in memoria, dopodichè interpreta il segnale ed esegue le operazioni necessarie. Terminata la procedura di *interrupt handling* il controllo del processore viene restituito al *task* sospeso precedentemente.
Ecco un esempio di espansione delle capacità di un sistema, da un dispositivo in grado di eseguire semplici calcoli e di ricevere segnali dall'esterno otteniamo un'entità in grado di prendere diverse decisioni. In questo modo si aumentano le potenzialità di uno strumento senza intervenire sulla sua struttura.
Ovviamente non è tutto oro quello che luccica. Questo meccanismo introduce parecchie difficoltà: innanzitutto occorre assegnare a ciascun segnale un'azione da eseguire e questo insieme di *routine* va protetto dall'accesso dell'utente per evitare malfunzionamenti dovuti a errate sovrascritture delle procedure. All'arrivo di un' *interrupt* occorre operare

un cambio di contesto, ovvero salvare i dati del processo in esecuzione in una locazione temporanea di memoria e caricare nella CPU dati e istruzioni relative al segnale ricevuto. Questo introduce un *overhead* non indifferente altresì noto come *latenza di interrupt* (tempo che intercorre dall'arrivo dell' *interrupt* all'esecuzione dell'azione ad essa legata). Infine cosa accadrebbe se durante l'esecuzione di una *Interrupt Service Routine* arrivasse un'altra *interrupt*? Le *interrupt* possono interrompere qualsiasi *task* in esecuzione? Vi sono processi che hanno una relativa importanza come ad esempio la schedulazione del processore. Non tutti questi processi possono essere interrotti; occorre perciò inibire la gestione delle *interrupt* in alcune occasioni. Questo implica che dobbiamo operare in due modalità differenti: una in grado di accettare le *interrupt* e l'altra sorda per quanto riguarda questo aspetto.

- Protezione dei dispositivi

Quando lavorate al PC non siete soli; un silenzioso compagno lavora assieme a voi sulla stessa macchina svolgendo un compito ben preciso: permetterne il corretto funzionamento. Utente e Sistema Operativo svolgono mansioni del tutto differenti sullo stesso calcolatore; è quindi necessario operare una distinzione fra il codice eseguito da un'entità e quello eseguito dall'altra.
Alcune operazioni svolte dal Sistema Operativo hanno la necessità di non essere interrotte; per fare un esempio vediamo rapidamente la schedulazione di un processore. Unix è un ambiente multiprogrammato, ovvero è in grado di eseguire più *task* in parallelo dando l'impressione all'utente di poter eseguire più operazioni contemporaneamente; lo *scheduler* del processore è uno degli strumenti che implementano questa peculiarità: esso esamina ciclicamente i processi in attesa della risorsa tempo processore e in base a diversi criteri, la assegna ad uno di essi. Risulta perciò essenziale che lo stato dei processi non cambi nel momento in cui lo *scheduler* sta operando la scelta: se un processo terminasse in quel momento sarebbe necessario riesaminare tutto l'insieme altrimenti si correrebbe il rischio di affidare al processore un *task* esaurito.
È chiaro che procedure come lo *scheduling* del processore necessitano di una posizione privilegiata rispetto al resto del codice; tali operazioni non vanno interrotte, perciò devono avere durata breve e una particolare robustezza. Impedire l'interruzione di un processo vuol dire che, qualora si blocchi a causa di un errore, esso non potrà rilasciare la CPU causando probabilmente il blocco dell'intero sistema. La tendenza è quindi quel-

la di evitare questa possibilità limitandosi alle meccaniche essenziali al funzionamento del Sistema Operativo.
Infine, come accennato in precedenza, occorre assicurare sufficiente protezione ai meccanismi interni della macchina e del Sistema Operativo stesso.
Immaginate un programma utente in grado di avere accesso alle aree dedicate all'avvio della macchina, oppure semplicemente ad aree dedicate ad altri programmi; il risultato sarebbe disastroso: un programma viziato da errori potrebbe sovrascrivere sezioni di codice critiche compromettendo dati utili per il corretto funzionamento del calcolatore. Lo stesso discorso è valido per i dispositivi di I/O: si potrebbe scrivere un programma che modifichi il *buffer* di una stampante durante la sua attività; il risultato sarebbe sicuramente curioso, se non fosse che l'utente con la stampante ci lavora.
L'hardware moderno è in grado di rilevare gran parte di questi errori. Nelle moderne CPU è introdotto un *flag* di dimensioni pari ad uno o due bit definito bit di modo che garantisce una duplice modalità di funzionamento: distinguiamo così una modalità utente se il bit è settato a 1 e una modalità detta di sistema (detta anche modalità privilegiata, supervisore, o modalità kernel) se il bit è settato a 0. In questo modo se un segmento di codice residente nel processore cerca di accedere ad un'area di memoria riservata e il bit di modo è settato a 1, l'hardware impedisce l'esecuzione del comando sollevando un'eccezione di sicurezza (una trap, gestita come un' *interrupt*) avvisando il Sistema Operativo che agirà di conseguenza.
In conclusione quando avremo bisogno di accedere a spazi di memoria al di fuori del nostro dominio (ad esempio la lettura di un file) dovremo fare richiesta a entità che possiedono maggiori privilegi tramite l'uso di interfacce apposite (approfondiremo nel prossimo articolo trattando le *system call*).
Occorre aggiungere una precisazione: quanto detto non ha nulla a che vedere con i meccanismi di protezione per gli utenti introdotti da Unix, anche se principi e realizzazioni sono molto simili. Le modalità utente e superutente (o root) alle quali si è abituati presentano obiettivi similari e adottano meccanismi analoghi, ma risiedono a livelli di astrazione ben più elevati; quando vi autenticate come root non impostate nessun bit del processore, inoltre non è possibile accedere direttamente a nessuno spazio di memoria (essa è infatti del tutto trasparente all'utente). È utile osservare come gli stessi meccanismi possono risiedere in più livelli nella realizzazione di un sistema complesso; questa è un'altra peculiarità del metodo di astrazione dovuta proprio al fatto che ogni modulo, pur dipendendo dall'interfaccia del livello sottostante, è completamente celato

agli altri.

8.1.6 Organizzazione di un Sistema Operativo

Abbiamo analizzato i principi di progetto, definito gli obiettivi e a grandi linee, esaminato gli strumenti messi a disposizione. Ora siamo in grado di tracciare un'ipotetica organizzazione del nostro Sistema Operativo.
Per cominciare cerchiamo di suddividere l'intera struttura in un preciso numero di livelli o strati. In questo modo sarà possibile dividere meglio il lavoro; la teoria non ci fornisce un numero fisso di moduli, esso dipende da molti fattori (numero di sviluppatori, tempo a disposizione, dimensione complessiva del progetto). Occorre tenere presente che applicando una frammentazione eccessiva si corre il rischio di risultare troppo dispersivi. D'altro canto moduli di dimensioni elevate comportano complessità nello sviluppo e tempi di *debug* in grado di compromettere il progetto.
Ecco come si potrebbe presentare la struttura del nostro Sistema Operativo; identifichiamo un totale di sei livelli.

- Livello 0 - Astrazione dell'hardware e virtualizzazione della CPU

In questo modulo prepariamo l'infrastruttura per la gestione delle *interrupt*, diamo vita alla nozione di processo e estendiamo l'architettura della CPU ottenendo uno o più processori virtuali. Parte della RAM viene riservata a tali procedure e risulta quindi opportuno (ma non strettamente necessario) oscurare questi settori in modo tale da evitare pericolosi accessi in scrittura da parte dei livelli superiori.

- Livello 1 - Virtualizzazione della memoria

L'esaurimento della memoria è sempre stato uno dei peggiori incubi del programmatore. A tutt'oggi non esiste una soluzione ottimale a tale problema. Per ovviare a tutto ciò, in passato si è giunti all'adozione di un valido espediente: oltre alla memoria centrale si dedica allo stesso scopo un'area della memoria secondaria (residente nei dischi fissi). Tale area in Unix è nota come area di swap. Lo *swapping* (dall'inglese scambio, avvicendamento) consiste nel salvare parte dei dati necessari nell'area apposita in modo tale da poterli richiamare rapidamente in memoria centrale qualora siano necessari. In questo modo il programma ha l'illusione di avere sempre memoria sufficiente a disposizione. I programmatori sono perciò sollevati da una problematica non indifferente.

Ultimamente, visti i costi attuali delle memorie RAM, tali meccanismi hanno perso molta della loro importanza. Lo scopo principale di questo livello è di astrarre completamente la gestione della memoria. I livelli superiori non avranno accesso a tale risorsa in modo diretto; semplicemente essi richiederanno a questo modulo la quantità di memoria necessaria, senza doversi preoccupare di come o dove tale memoria sia immagazzinata.

- Livello 2 - Gestione dei dispositivi

In questo livello sviluppiamo i *driver* dei dispositivi che vogliamo supportare nel nostro progetto; ovviamente la dimensione di tale modulo varia dalla quantità di macchine su cui vorremo far correre il Sistema Operativo. Ciascun dispositivo viene semplificato il più possibile automatizzando al meglio il suo funzionamento e limitando quanto possibile il numero di istruzioni che è necessario impartire per poterlo dirigere.

- Livello 3 - Astrazione dell'I/O nel File System

Il *File System* è l'insieme di meccanismi tramite i quali si astrae la gestione dei dati. Ciascuna collezione di informazioni viene racchiusa all'interno di un'entità nota come *file*. È compito di questo livello gestire l'organizzazione e la collocazione di tutti i *file* presenti all'interno della macchina.
Unix organizza il *File System* tramite una struttura ad albero assegnando ai vari rami ambiti e privilegi diversi. Inoltre l'interfaccia di ciascun dispositivo viene astratta in uno o più *file*: se si accede a tali *file* in scrittura è infatti possibile agire parzialmente sui dispositivi presenti all'interno della macchina. Generalmente le interfacce dei dispositivi sono racchiuse nei sottorami `/proc/</code` o `/sys/</code` in base alla versione del kernel in uso. Se si prova ad esplorare tali sottorami ci si accorge che sono presenti i direttori relativi a tutti i dispositivi di I/O della macchina: all'interno di tali direttori vi saranno un certo numero di *file*. Agendo correttamente e con la dovuta cautela è possibile impartire le disposizioni volute semplicemente sovrascrivendo il contenuto di tali *file*.

- Livello 4 - Application Programming Interface (API)

Giunti a questo livello possiamo finalmente allontanarci dalla macchina; essa è infatti completamente astratta dai moduli sottostanti i quali ci restituiscono un dispositivo virtuale esteso ma non ancora completo, almeno per quanto riguarda le aspettative dell'utente.

Occorre introdurre il software di base che permetta il passaggio dal supporto creato ai programmi noti all'utenza generale. Possiamo già includere programmi veri e propri come ad esempio l'interprete dei comandi o shell, oppure scrivere librerie che agevolino la programmazione al livello superiore. In questo modulo inseriamo tutte le librerie avanzate che riteniamo necessarie, dalla libreria standard del linguaggio C (Glibc all'interno del progetto GNU), ai *toolkit* per lo sviluppo di software a interfaccia grafica (ad esempio le librerie Qt o GTK+). I programmatori del livello superiore faranno uso di tali librerie per scrivere i programmi che utilizziamo quotidianamente.

- Livello 5 - Software applicativo

Questo è il modulo finale, ovvero l'interfaccia fra l'utente e l'intero sistema che abbiamo visitato in questo articolo. Utilizzando le librerie di Interfaccia alla Programmazione di un'Applicazione (note come API) i programmatori sono in grado di creare tutto ciò che vediamo sul monitor, dai singoli applicativi come gli elaboratori di testo o i *browser Web* all'intero *Desktop Manager*.

Questa struttura non ha la pretesa di essere un modello di riferimento per i Sistemi Operativi Desktop Oriented ma rimane comunque un buon esempio da utilizzare per avere una visione globale senza scendere troppo nei dettagli.
Ciò che risulta utile comprendere è l'utilizzo che il progettista fa di uno strumento come l'astrazione: ciascuno strato va a coprire quello precedente mascherando ed espandendo via via l'architettura che si va a formare. Ogni modulo ha accesso solamente all'interfaccia del livello sottostante; a esso non è neccessario conoscere altro, nemmeno quale strato andrà a sua volta a coprirlo. La comunicazione è verticale, orientata dall'alto verso il basso, il livello superiore impartirà le direttive a quello inferiore il quale potrà comunicare solamente il proprio stato.
Tornando all'esempio dell'automobile vediamo come il sistema sia analogo: il guidatore non sa nulla di ciò che avviene dietro al volante, egli comunica col veicolo impartendo le direttive di guida tramite l'apposita interfaccia. Il veicolo è in grado solamente di comunicargli il proprio stato tramite le spie presenti sul quadro dei comandi. Inoltre il motore non è in grado di capire se a guidarlo è un pilota professionista o un'anziana signora dalla guida sportiva; occorre perciò prevedere in fase di progettazione del modulo l'eventuale uso che ne verrà fatto dai livelli superiori.

8.1.7 Commento dell'autore

Arrivati a questo punto è lecito chiedersi quale sia lo scopo di questa lunga introduzione; l'articolo è incentrato sui kernels ma fino ad ora si è parlato di tutt'altro.
Spesso alcuni concetti ci sembrano talmente complicati da risultare impossibili da comprendere e analizzare, in realtà la difficoltà di una nozione è strettamente legata al modo in cui una persona tende ad approciarvisi. Cercare di capire cos'è un kernel partendo da concetti generali ed esempi è controproducente tanto quanto mettersi di fronte all'intera architettura con la pretesa di analizzarla nella sua interezza.
Lo scopo primo di una scienza orientata alla tecnologia è identificare uno o più concetti primitivi, acquisirne la completa padronanza vantando sulla loro semplicità (vi siete mai chiesti perchè nell'informatica si usa il sistema binario e non quello decimale?) e impiegarli in strutture ben più complesse. Le case son fatte di mattoni, i milioni di centesimi e i calcolatori di microscopici interruttori.
In questo articolo si prova ad applicare al processo cognitivo lo stesso principio: anzichè spiegare cos'è un kernel illustrandone le carattistiche si è partiti introducendo alcuni dei concetti che stanno alla base della progettazione di un Sistema Operativo, dopodichè si è passati alla progettazione vera e propria. Invece di chiedersi Che cosa è un kernel? ci siamo chiesti Dati i presupposti, dovessimo progettare un Sistema Operativo, come agiremmo?.
L'articolo che uscirà sul prossimo numero dell'e-zine Debianizzati partirà da questi concetti, sviluppando su di essi la nozione di kernel. La speranza è che il tutto sembri più semplice analizzandolo in quest'ottica: una volta sviluppato tale argomento si studieranno i principali approcci in uso per la progettazione di un kernel.

NoxDaFox

Impressum

Redazione

brunitika, mm-barabba, pmate, xtow, borlongioffei, Simone, bel.gio, Aki, ferdybassi, fr4nc3sco, NoxDaFox, MadameZou

Redattori articoli

Pagina dei lettori (Emacs E-Mail Essential [prima parte]) - *samiel*
Zack - *MadameZou*
Debian4Children - *mm-barabba, furly*
Debian GNU/Hurd: aggiornamenti - *brunitika*
pam_usb - *pmate, Aki*
PGP: configurazione e utilizzo in Debian - *greyfox, pmate*
Squid e DansGuardian - *ferdybassi*
MoBlock (Mobloquer) - *mm-barabba*
Tiger: uno strumento per l'audit di sicurezza - *Aki*
Introduzione ai Kernel: prima parte - *NoxDaFox*

Copertina

Simone

Impaginazione

borlongioffei (versione stampa), *brunitika* (web-zine)

Contatto

Tutti i membri del progetto sono reperibili sul forum del portale www.debianizzati.org, dove è possibile trovarci cercando l'utente relativo nel forum.
I sorgenti LaTeX di quesa versione e delle precedenti, sono disponibili all'indirizzo:

http://e-zine.debianizzati.org/source/

`Happy Debian, Happy hacking`