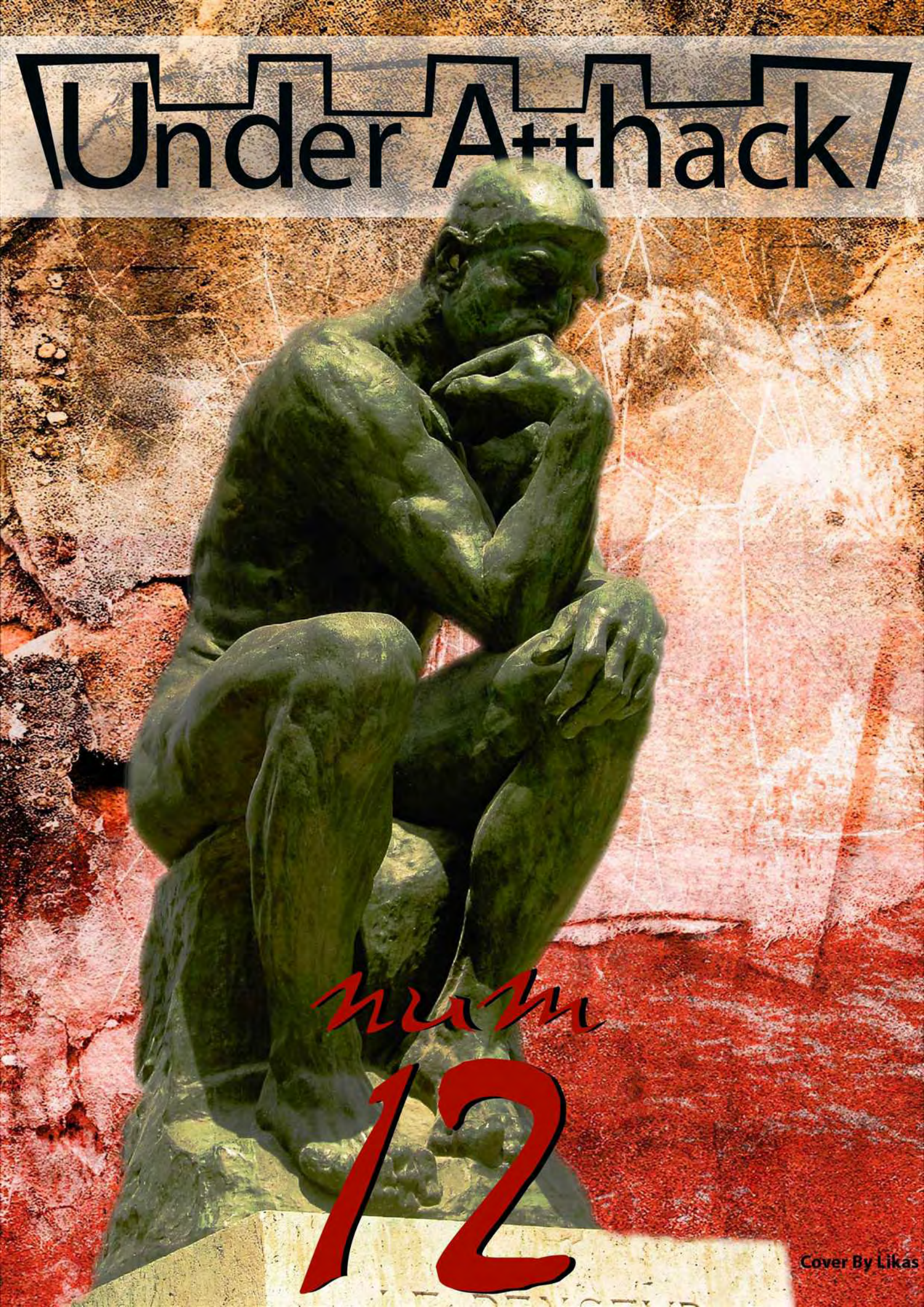
Under Attack



Issue 12

Cover By Likas

UNDERATTHACK

IN QUESTO NUMERO

N.12

Prefazione

Ciò che contraddistingue l'uomo è la sua razionalità, cioè la sua capacità di pensare, imparare e ragionare per causa/effetto.
La cosa che mi sono chiesto è: quando noi realmente pensiamo?
Ci svegliamo e facciamo tutto in modo meccanico. Al lavoro facciamo le stesse cose ogni giorno, poi la spesa comprando gli stessi prodotti, poi mangiamo alla stessa ora, guardiamo gli stessi programmi, andiamo negli stessi posti in vacanza.
Avete presente la massima del salutismo "ogni giorno dedica tot tempo alla cura del tuo corpo"? Avete mai sentito dire di prendersi un po' di tempo per pensare?
Sembrerebbe che sedersi a riflettere sia cosa secondaria se non inutile. Evidentemente è comodo organizzare una società sulla base del fatto che qualcuno o qualcosa deve pensare al posto degli individui: re, sacerdoti, generali, politici, scienziati, giornalisti, ecc.
Anche il solo fatto di ascoltare ed elaborare un'opinione articolata passa in secondo piano.
L'Università insorge contro la riforma? Posizione A: la Ministra criminale sta distruggendo il futuro dei giovani. Posizione B: studenti criminali, andate a lavorare. Discorso chiuso, ogni altro ragionamento non fa audience e non piace alla gente.
In ogni organizzazione la gestione della "responsabilità" è un aspetto fondamentale; purtroppo è un dato di fatto che nella prassi i responsabili hanno due ruoli principali in positivo e in negativo, in positivo se diventano partecipi dello sviluppo complessivo come esperti coordinatori del loro settore, in negativo quando fungono da capro espiatorio degli insuccessi generali. Forse avere dei forti (o presunti) gruppi di potere aiuta a vivere più sereni. Quando le cose vanno male è colpa della politica, della società, dei ricchi, dei poveri, dei giovani, dei vecchi. La cosa più importante è che questi soggetti siano quanto più distanti possibile da noi stessi. D'altra parte se quei gruppi in realtà fossimo semplicemente noi con i nostri mille interessi contrapposti, allora le responsabilità sarebbero solo e unicamente nostre, giusto? Abbiamo le nostre colf in nero ma non siamo evasori, costruiamo i nostri garage senza autorizzazione ma non siamo abusivi, chiediamo i favori al Sindaco ma non siamo corrotti, buttiamo le cartacce per terra ma non siamo inquinatori. Ci sono sempre sopra di noi fantomatiche categorie di persone che creano il danno, con le nostre piccole negligenze che diventano addirittura l'effetto del danno creato non da noi e contro cui nulla possiamo.
La cosa che realmente mi dà speranza è che scrivendo queste poche righe mi sono messo a pensare. E spero che voi abbiate fatto altrettanto.

Buona lettura
Floatman

Happy Two Years

UnderAttHack compie due anni. Possiamo dire che effettivamente ha imparato a camminare con le sue gambe quindi siamo in linea con i tempi della crescita. Siamo decisamente felici che UAH non sia diventato un contenitore pubblicitario e altrettanto si può dire per lo stile degli articoli rimasto improntato a dare spunti e non a fornire il sussidiario di scuola. Si può obbiettare che dalla nostra e-zine non c'è nulla da imparare; la cosa purtroppo accade perché in quel caso non si è imparato a leggerla. Se dai nostri articoli si dovesse imparare, allora noi dovremmo insegnare e personalmente non mi sento in grado di fare così tanto. Pensare insieme, noi della Redazione e voi Lettori, questo si può fare ed è una cosa utile: c'è già troppa gente che insegna. Buon compleanno UnderAttHack!

Floatman

Grazie a tutti coloro che hanno collaborato in questi due anni:

adsmanet	**vikkio88**	**Init0_**
MRK259	**BlackLight**	**DoMinO**
Floatman	**Sh3llc0d3r**	**Andrea Bertin**
Søny	**Syst3m Cr4sh**	**Nex**
Zizio|Kriminal	**TheCr0w**	**ultimoprofeta**
Elysia	**(giorgio) LostPassword**	**cercamon**
Slacer™	**!R~**	**Likas**
ptrace()	**Stoke**	

Under_NEWS_AttHack

Hacking a tutto campo!

Suppongo che come tutti i veri smanettoni anche voi abbiate le tipiche giornate in cui vi frullano mille idee in testa. Purtroppo tempo libero e idee non sempre vanno a braccetto e quindi a volte ci si trova in quello stato di limbo produttivo in cui si ha la voglia di lavorare in qualche modo su un qualche cosa, senza però avere dei passi precisi da seguire. A quel punto si aprono due possibilità: se siete appassionati del genere potete aiutarvi con le droghe psichedeliche, se invece come me preferite drogarvi a tagliatelle e carne al sangue allora potete rivolgervi a Google...

http://hackaday.com

"Hack a Day" è una maxi raccolta dove ogni pazzia mentale trova una propria forma nella realtà; comprende software, hardware, multimedialità, meccanica, elettronica e ogni altra diavoleria che possa smuovere il gusto dell'innovazione. Nel sito (in lingua inglese) potete trovare un comodo menù dei vari argomenti trattati, con una serie di post che rimandano ai testi originali degli autori che hanno inviato il loro hack; è anche possibile inviare i propri hack agli amministratori del portale in modo che siano pubblicati, quindi fatelo per dimostrare che siamo sempre italiani e l'ingegno non ci manca (il momento è anche buono per alzare un po' la testa). L'effetto collaterale è la facilità a passare la notte insonne perdendosi nei meandri del Paese delle Meraviglie, ma tanto ci siamo abituati, oltre al fatto che con le droghe psichedeliche credo che il problema sia il medesimo.

Digitale e Sostenibile

Anche se chi non ha cultura informatica non ci pensa minimamente, tutto preso dai pannelli solari e dalle auto elettriche, esiste da tempo anche la questione dell'informatica sostenibile. Un impegno dell'open source applicato è anche quello di ridurre il divario digitale e rendere l'accesso alla tecnologia una facoltà che sia realmente di tutti. Dall'hardware 'ecologico', al trashware, all'uso di sistemi aperti nelle aziende e nella pubblica amministrazione, la logica rimane la stessa: sfruttare le tecnologia con lo scopo di sfruttare al meglio... la tecnologia!

http://www.binarioetico.org

BinarioEtico è un'associazione che si impegna proprio in questo campo; il progetto è in espansione e punta su quell'unione tra informatica e sviluppo sociale sostenibile che tanto bene farebbe al nostro tempo e al nostro futuro. Le sua iniziative sono sempre ben documentate e possono sicuramente servire a coloro che intendano intraprendere percorsi analoghi, oltre al fatto di essere molto attivi nei campi della formazione e della consulenza. Un progetto che promette molto bene e che speriamo sia imitato il più possibile.

Fuoco alle polveri o polvere negli occhi

Giovane, esperto di informatica, voglia di intraprendere, disposto da subito a mettersi in gioco per progetti innovativi. Voi lo assumereste? In Italia no, sarebbe meglio qualcosa tipo: ragazzo offresi per lavori pesanti, minimo dieci ore per sei giorni a settimana, massimo 800 euro al mese. Ecco, questo è l'identikit del giovane a cui il Bel Paese offre opportunità per il domani. Chi come me lavora da un po'di anni e fortunatamente è fuori da questi problemi (finché dura) non può non aver notato come negli ultimi tempi il mondo del lavoro sia basato sul fatto di non dare uno stipendio alla gente. Che sia colpa delle imprese, della politica, dei Cinesi, della crisi, ecc., sta di fatto che chi ha la testa la può usare soltanto all'estero. Nel sito del Corriere leggo la storia di Augusto Marietti, Marco Palladino e Michele Zonca, tre giovani Milanesi con la passione per l'informatica e voglia di impresa:

http://www.corriere.it/cronache/11_gennaio_20/tre-milanesi-rifiutati-in-Italia-che-hanno-sfondato-nella-Silicon-Valley_1ef7b94c-246f-11e0-8269-00144f02aabc.shtml

Troppo giovani, troppe idee innovative, niente credito, niente investitori, niente clienti. Avete presente il 'dannati ragazzini' del manifesto hacker di Mentor? I ragazzini hanno fatto il fagottino e sono andati in USA, dove hanno ottenuto un finanziamento di 100.000 dollari dai signori di YouTube, che ora stanno investendo nei loro progetti. 100.000 dollari sono veramente due lire non solo per un colosso della new-economy ma anche per molte medie imprese del nostro paese. Era davvero necessario passare l'Atlantico? A quanto pare sì. Buona fortuna ragazzi, a voi e a tutti gli altri che resistono e cercano disperatamente un futuro.

Post - Office

Salve gente... Eccoci di nuovo con la nostra rubrica della posta. In questo numero abbiamo deciso di pubblicare solo due mail belle corpose, però mi sento in obbligo di farvi notare, dato che il numero scorso qualcuno si è lamentato della pubblicazione di un suo messaggio, che in fondo ad ogni numero è scritto questo:

> 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.

La parte finale è quella che maggiormente mi preme farvi notare. Chiunque invia mail ad UnderAttHack senza specificare che non vuole essere pubblicato, di fatto ci autorizza a pubblicare i messaggi che riteniamo di interesse generale o particolarmente curiosi o che ci danno l'occasione per spiegare meglio il nostro impegno e il nostro lavoro. Quindi ci dispiace per chi si è vista la mail pubblicata anche se non era sua intenzione e vi prego d'ora in poi di specificare i vostri desideri. Grazie e buona lettura .

Da: fuffo fuffoni
<c4p.h00k@gmail.com>

Salve a tutta la redazione,
sono un vostro appassionato lettore, e devo dire che la vostra rivista mi opiace molto, primo motivo è scritta in italiano e credo che sia unica nel panorama nazionale.
Non è l'unica enzine di hacking che leggo le altre (tra cui anche una di quelle storiche) ma sono tutte in inglese.
Apprezzo molto i vostri contenuti e la chiarezza di esposizione, alla vostra domenda se c'era bisogno di una nuova enzine inforamtica vorrei dire la mia: si, c'era bisogno!
Oggi invece vedo una certa contrazione della diffusione di conoscenza che ha animato l'underground di qualche anno fa.
Siti di hacking costantemente messi offline (a proposito che fine ha fatto www.rootikt.com?), progetti prima open che diventano a pagamento, forse qualcuno si è accorto che la sicurezza può diventare un bel business e comincia a fermare la conoscenza?
Questo andrebbe contro tutti i discorsi di hacking fatti negli anni passati e disegna uno scenario molto triste, come sempre i soldi vincono su tutto.
ho una domanda allo staff: ma avete anche voi messo offline il forum di hackingeasy che leggevo con molto interesse?
grazie per l'attenzione e buon lavoro a tutto lo staff.
buon natale.

Ciao a te o fuffo :-)
Grazie tantissimo dei complimenti, devo dire che sei l'unico che ha capito che volevamo una risposta a quella domanda! Se non rispondevi il numero 12 era l'ultimo poi chiudevamo bracca e burattini (ovviamente scherzo :-)).
Il tuo discorso sulla contrazione della conoscenza che animava l'underground mi ha molto colpito, credo che tu abbia proprio ragione, il "business che ferma la conoscenza" non solo è un immagine trita e ritrita in qualsiasi campo, ma è purtroppo un dato di fatto, in quanto con la diffusione della conoscenza non si "porta pane a casa". Chi prova a farlo, come noi nel nostro piccolo con UnderAttHack, ha non pochi problemi, in quanto un progetto del genere ha bisogno di tantissimo tempo dedicato e non tutti nella redazione ne abbiamo a sufficienza ma con tanti sacrifici ci spingiamo l'un l'altro a partorire ogni due mesi un nuovo numero. I soldi non vincono su tutto, ma di sicuro chi fa soldi ha la strada spianata, se in redazione ci fosse qualcuno che ci paga sicuramente avremmo molti più articoli ogni mese non pensi? :-) L'avidità è nell'animo umano!
Personalmente io non ho messo offline nulla in quanto anche se ero admin non ero il founder, e così anche gli altri della redazione. Ci portavamo dietro HackingEasy come un peso arrivati ad un certo punto, perchè purtroppo nessuno di noi riusciva a dedicarsi ad entrambi i progetti ed UnderAttHack pianpiano è diventato più importante e seguito e abbiamo deciso di dividere le due cose, infatti adesso non troverai più nessuna citazione di hackingeasy su underatthack.org, Questo da circa 6 numeri. Poi un bel giorno mi sveglio mi faccio il solito giro e HE era sparito!
Sono rimasto perplesso quanto te anche perchè avevo dei topic su quel forum davvero interessanti e che ora non riuscirò più a recuperare, ma va bene lo stesso!
Grazie della bella chiacchierata e spero di avere risposto a tutto! :-)

vikkio88

Da: Koral
<koral78@gmail.com>

Ciao ragazzi,
mi chiedevo se voi poteste darmi una mano con del materiale per la mia tesi..
avrei da scegliere degli argomenti tra quelli di alcune slide che mi sono state date dal mio prof che, a quanto pare, è andato ad una specie di conferenza o corso di ethical hacking, CEH (Certified Ethical Hacker).
Dovrei scegliere 2 argomenti..e mi stavo orientando su
- botnet
- penetration testing

anche se ero indeciso anche su
- scanning
- sniffers
- VoIP hacking

Devo presentare un pò l'argomento (ognuno una 50ina di pagine) e spiegare il funzionamento di qualche tool open source

---

Ciao a te,
Non ho capito bene in cosa potremmo aiutarti noi :-), sulle botnet io stesso avevo scritto un post in giro sul circuito di forumcomunity diversi anni fa, ora non so che fine abbia fatto, viceversa sul pentesting aziendale abbiamo un articolo su uno dei primi numeri.
Scanning e sniffers non sono poi di così eccitante trattazione IMHO, quindi personalmente se potessi scegliere tra quello che gli argomenti elencati la mia risposta sarebbe di sicuro: BotNet.
E in giro per la rete troverai tantissimo materiale a riguardo...50 pagine sono parecchie ma si può allungare di molto magari parlano di tutti i metodi per crearle, tra cui phishing e vario altro tipo.
Un altro argomento che potresti integrare nella stessa discussione è il fatto che il phishing non è solo monopolio di Windows, e magari leggendo anche il mio articolo: Non accettare pacchetti dagli sconosciuti (UAH num.7), veder come con un pacchetto deb creato ad hoc anche un ubuntu in mano a qualcuno poco saggio può diventare parte integrante di una botnet globale :-)...
Beh spero di risentirti e per quanto possibile di averti aiutato :-)

Saluti
vikkio88
Redazione UnderAttHack

Prepariamoci per programmare il piccolo droide di casa Google

Intro

Oggi i dispositivi mobile hanno conquistato una grossa fetta di mercato. C'è chi li usa per lavoro, chi per svago e chi semplicemente li compra per il loro design.
In questi anni siamo stati bombardati da notizie su iPhone, Windows Mobile, Blackberry, ma poco spazio è stato dato ai dispositivi Android. Almeno fino a queste settimane dove il droide di casa Google si è fatto avanti prepotentemente sulle piattaforme smatphone e tablet.
Per chi non lo sapesse, Android è un progetto OpenSource scritto in Java e per questo motivo Google ha messo a disposizione l'intero codice sorgente del sistema operativo all'indirizzo:

http://source.android.com/source/download.html

(se volete sapere qualcosa di più su cosa è un sistema operativo vi invito a leggere i numeri 10 e 11 di UnderAttHack).

A differenza del suo principale concorrente (Apple), Google ha creato un market dove chiunque, previa registrazione e pagamento di circa 20 €, può pubblicare le sue applicazioni senza attendere che vengano approvate. Ma come si crea un'applicazione da pubblicare?

Strumenti di Lavoro

L'ambiente di sviluppo più utilizzato senza ombra di dubbio è Eclipse, uno degli IDE più famosi in circolazione grazie ai numerosi plugin disponibili in rete.
Proprio per questo IDE la stessa Google ha creato un plugin (ADT Plugin) che facilita la creazione dei pacchetti per Android (estensione APK).
Ecco cosa ci servirà per cominciare a lavorare:

1. Eclipse
2. SDK Android
3. Java SDK
4. ADT Plugin
5. Windows/Linux/MacOS

Procuriamoci il Necessario

Tralasciando il sistema operativo e l'SDK Java (che già dovreste avere se avete deciso di programmare per Android) andiamo a recuperare l'SDK ufficiale per lo sviluppo Android.
Da questo momento in poi io eseguirò la procedura di installazione su MacOS ma il tutto è applicabile anche a Windows e Linux con piccole varianti.

Eclipse

Per prima cosa scarichiamo, se già non lo abbiamo, Eclipse all'indirizzo:
http://www.eclipse.org/downloads/ e scegliamo la versione Classic. Al momento in cui scrivo l'ultima versione disponibile è la 3.6.1. Effettuiamo il download e installiamola sul nostro computer.
Se avete Linux basta il comando: `sudo apt-get install` eclipse per avere l'IDE completo installato.

SDK Android

Possiamo scaricare l'SDK dal sito ufficiale di Android all'indirizzo:
http://developer.android.com/sdk/index.html

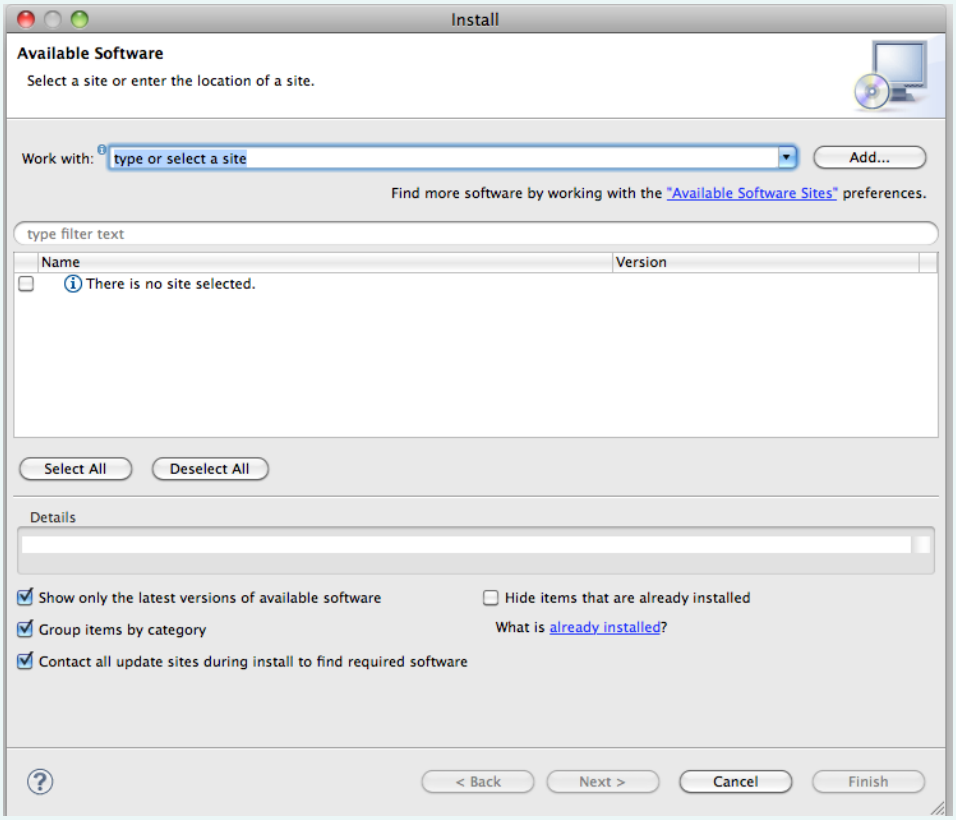
Ovviamente dovete scegliere la versione più adatta al vostro sistema operativo e il mio consiglio è quello di mettere la cartella scaricata (dopo che l'avrete estratta dall'archivio) in una zona del disco che al 100% non andrete MAI a cancellare.

ADT Plugin

A questo punto abbiamo scaricato quasi tutto il materiale che ci serve per iniziare a programmare. Ci manca solo l'**ADT Plugin** che ci darà una grossa mano nel creare un nuovo progetto già pronto per Android.

Per scaricarlo è necessario avviare Eclipse e, dopo aver impostato un **workspace**, andate nel menu **Help** e scegliete la voce **Install New Software** dal menu.

A questo punto vi troverete davanti a una schermata del genere:



Nel campo **Work with** non vi resta che aggiungere l'URL che punta al plugin, ovvero scrivete nel campo di testo:

```
https://dl-ssl.google.com/android/eclipse/
```

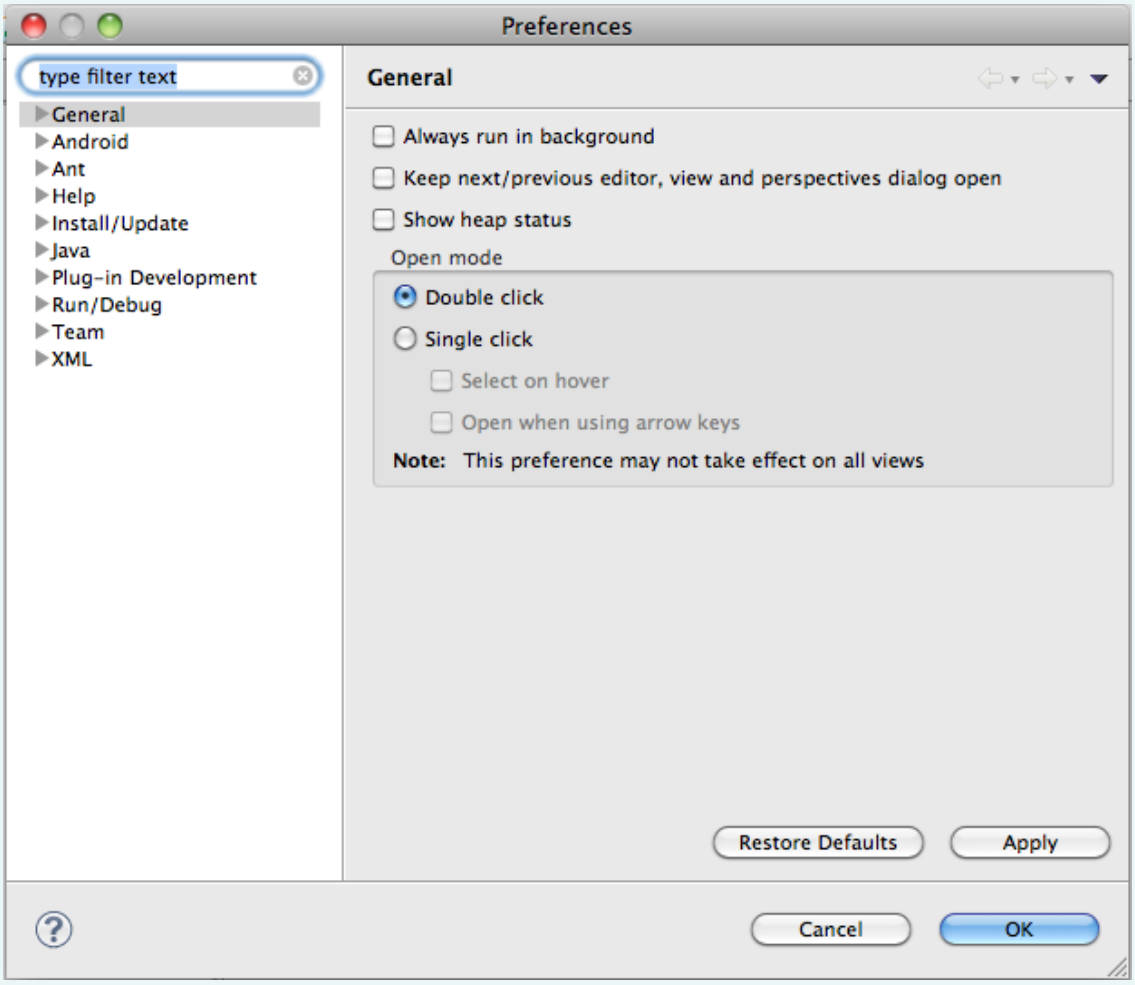
Se per qualche strano motivo andate incontro a qualche problema di connessione alla pagina usate il normale protocollo http al posto della connessione sicura https.

Selezionate i plugin che verranno elencati e installateli. A questo punto avete quasi finito, non resta che sistemare le ultime preferenze e creare un simulatore che servirà per testare le vostre applicazioni come se foste su un vero e proprio terminale Android.

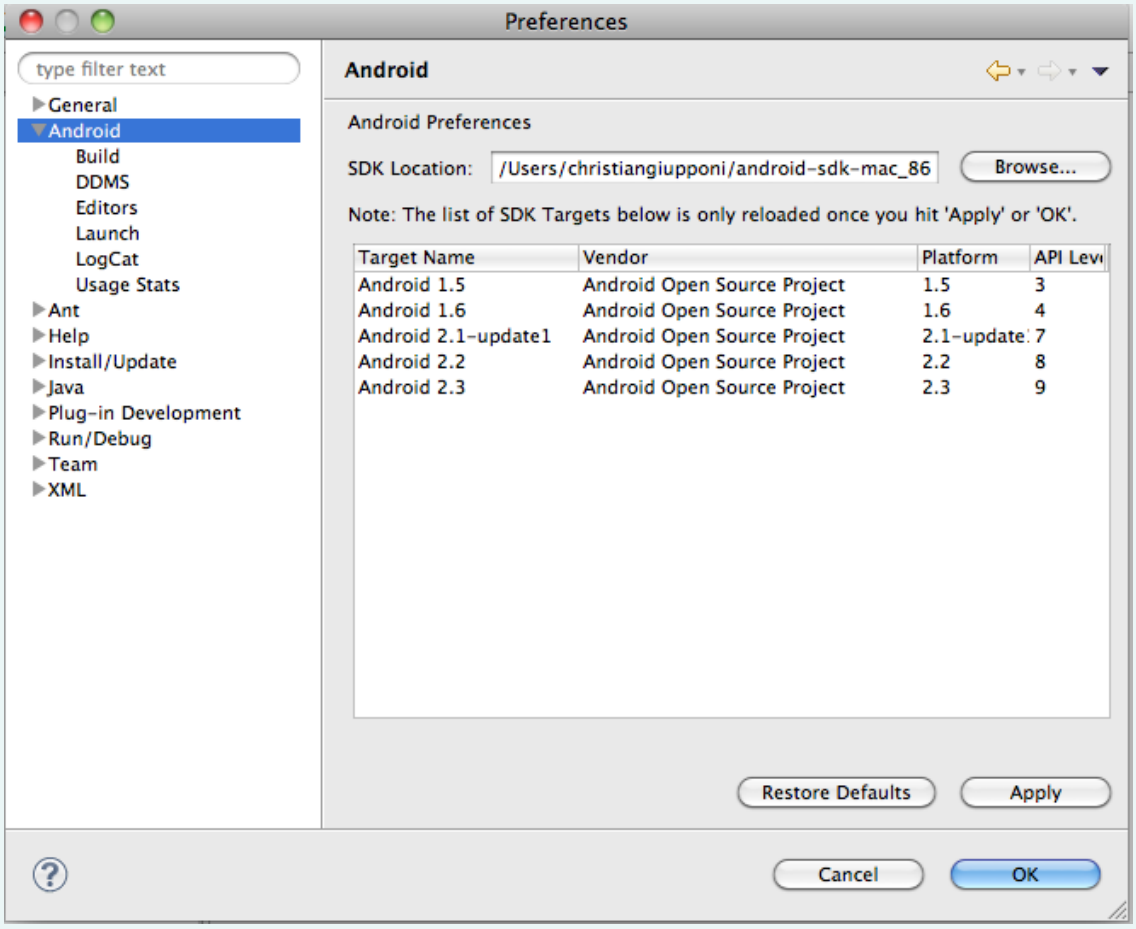
Configurazione

Per prima cosa dovete riavviare Eclipse in modo che le modifiche viste in precedenza diventino effettive. Per riavviarlo potete semplicemente andare nel menu **File** e cliccare sulla voce **Restart.**

Una volta riavviato andate nella voce **Preferences** e dovreste trovare una schermata come questa:



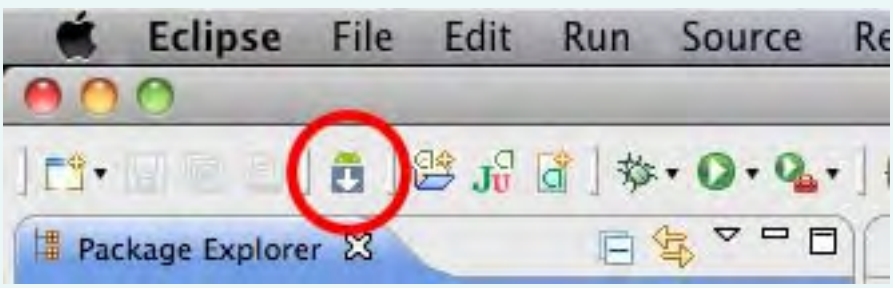
Selezionate la voce relativa ad Android e mettete il link relativo alla cartella dell'SDK che avete scaricato all'inizio della guida e che vi avevo detto di mettere in un posto sicuro.
Una volta fatto dovreste ottenere qualcosa di simile a questo:



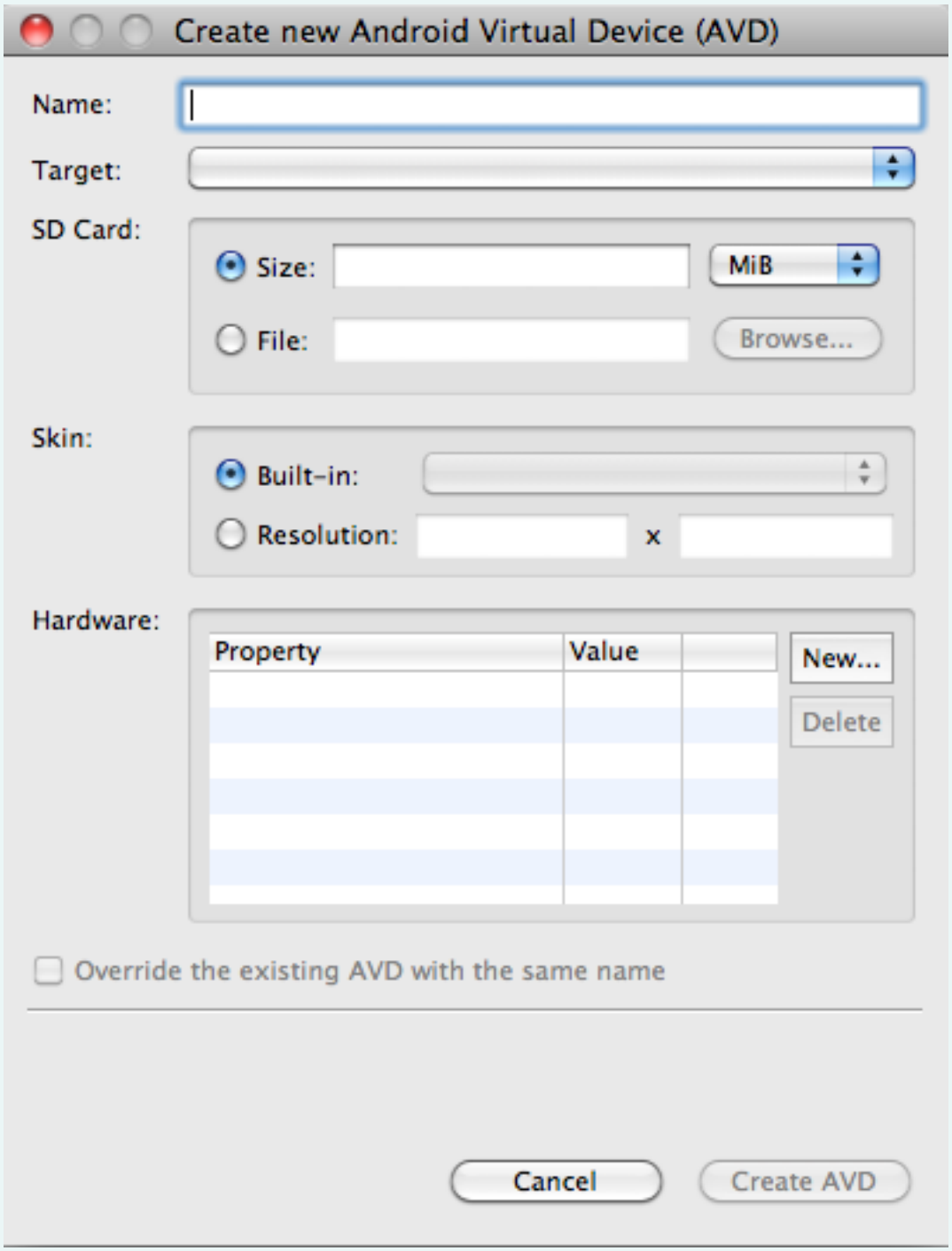
Nella finestra **Preferences** vi verranno elencati tutti i cosiddetti **target** disponibili che si riferiscono alle versioni di Android che potrete utilizzare.

N.B.: Se per qualche motivo non vedete nessun target (nemmeno dopo aver riavviato Eclipse), vi consiglio di rimuovere il plugin ADT installato in precedenza e reinstallare tutti i suoi componenti.

Una volta che avete l'elenco dei target disponibili, dovete creare il simulatore che vi permetterà di testare il frutto del vostro lavoro come se aveste un dispositivo fisico.
Per creare l'emulatore dovete cliccare sull'icona che il plugin ADT ha aggiunto al menu di Eclispe. Per essere più precisi dovete cliccare su questa icona:



Nella schermata che si aprirà cliccate prima sulla voce **Virtual Device** e poi sul pulsante **New**
A questo punto dovete semplicemente riempire i campi che appariranno:



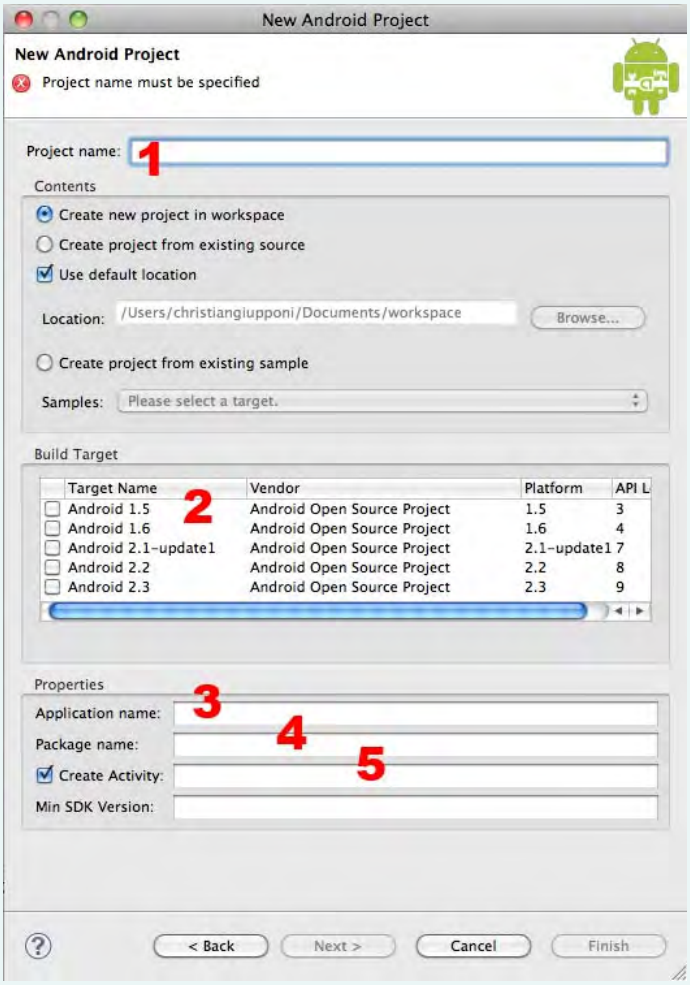
- **Name:** è il nome che verrà associato all'emulatore
- **Target:** è la versione di Android che volete utilizzare sull'emulatore
- **SDCard:** se inserita verrà associata all'emulatore un file che avrà funzione di simulare la presenza di una SDCard
- **Hardware:** permette di aggiungere e definire la potenza da simulare, per esempio la quantità di memoria, ecc.

Una volta completato il tutto cliccate su **Create AVD** ed Eclipse si occuperà del resto!
Se non sapete aspettare e volete vedere il simulatore già in azione vi basta premere sulla voce **Start.**

Ma potevo condurvi fino a questo punto e non farvi creare nemmeno un piccolo progetto? Potevo... ma mi è sembrato crudele, perciò... andiamo avanti! :)

Primo Progetto

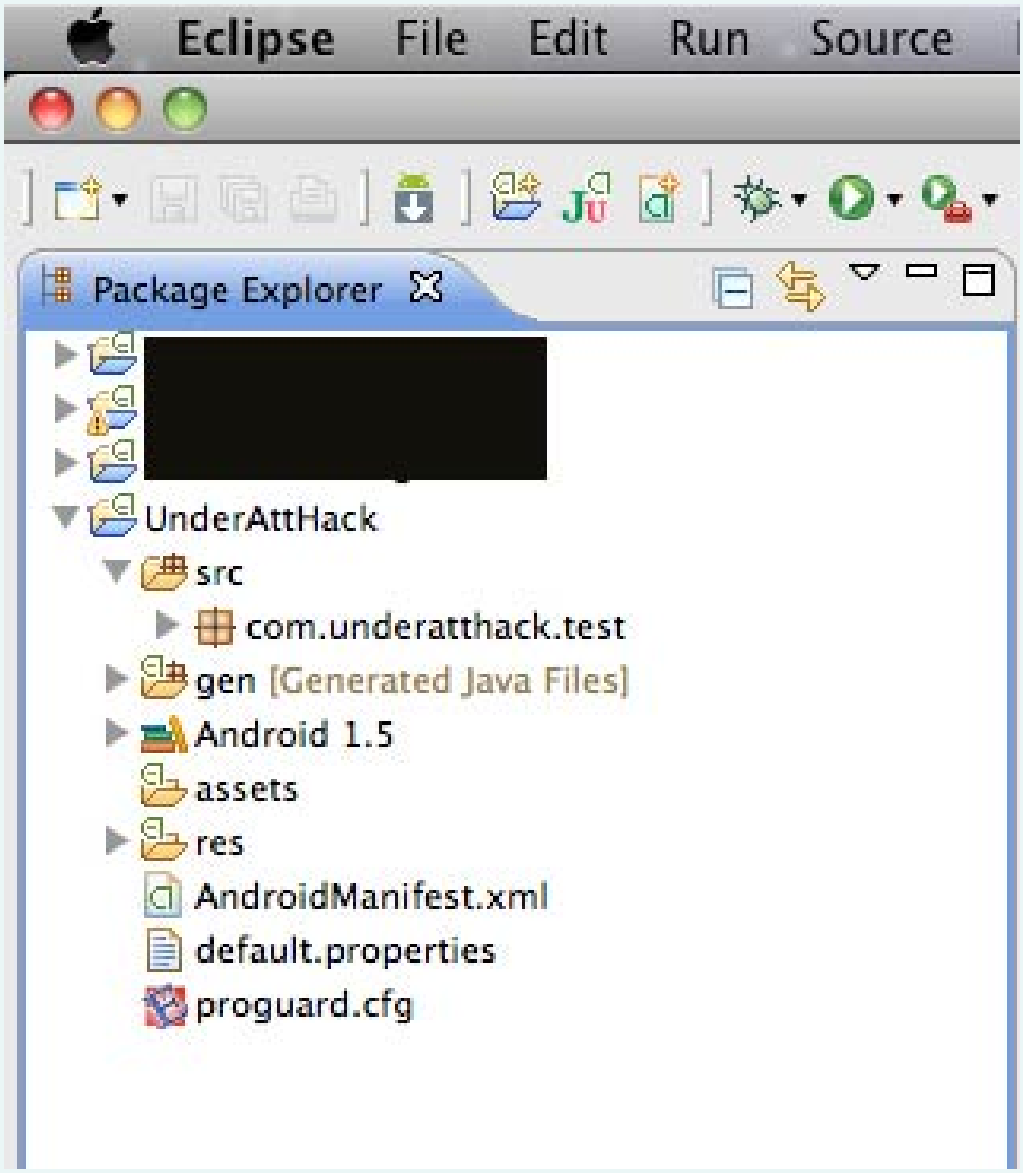
I giochi sono praticamente fatti: per creare un nuovo progetto non dovete fare altro che cliccare su **File > New > Android project** e riempire il form che si presenterà. Ecco come vi dovrebbe apparire il tutto:



Nella finestra:
1. Si riferisce al nome del nostro progetto.
2. Si riferisce all'SDK che vogliamo utilizzare, una SDK bassa garantisce più compatibilità anche su device più vecchi ma limita molto il dispositivo.
3. Si riferisce al nome che vogliamo dare alla nostra applicazione.
4. Indica come sarà il package (molto comune nei progetti Java) del progetto. Per poterlo accettare Android richiede che sia formato da almeno due voci separate da un punto:
(es: com.underatthack.test).
5. Si riferisce al nome che dovrà avere la nostra activity, ovvero la classe principale che verrà chiamata all'avvio dell'applicazione.

Una volta completata la compilazione dei campi clicchiamo su **Finish** e nell'interfaccia di Eclipse, sotto la voce **Package Explorer**, avremo il progetto appena creato!

Ecco cosa dovreste avere:



Già a questo punto il progetto è eseguibile e quindi potete testarlo con l'emulatore. Avviandolo dovreste ottenere questo:

Conclusione

Mi fermo qui, sperando di avervi dato una mano a preparare il vostro ambiente di sviluppo per addentrarvi nel fantastico mondo di Android!
Se provate a dare un'occhiata al codice generato vedrete che è praticamente identico a Java. Quindi, se volete iniziare a programmare seriamente per quest'ottima piattaforma, non vi resta che fare di Java il vostro migliore amico, oltre alle API Android ovviamente. :)

Christian "ultimoprofeta" Giupponi

Elettroche?

Una innocente (e priva di formalità) introduzione all'elettronica amatoriale grazie alla piattaforma opensource: Arduino

Qualsiasi individuo al mondo che abbia avuto a che fare con l'informatica, sia ad alti che a bassi livelli, ha dovuto (la maggior parte delle volte inconsapevolmente) utilizzare diverse leggi fisiche che ne regolano la logica di funzionamento nei sistemi più reali e concreti che stanno dietro a tutto questo meraviglioso mondo. Molti affrontano il tutto, come detto, in maniera inconsapevole, sia per questioni di carattere personale tendente a fregarsene di capire come funzionano le cose (basta che funzionino), sia per paura di affrontare sfide di (secondo l'ignorante) alto livello. Ma chiunque nella sua vita si sia mai chiesto:

(1) Come fa il mio PC a mandare su un foglio elettronico del mio programma di videoscrittura la lettera 'a' subito dopo che ho premuto il rispettivo tasto?

O più semplicemente 'come fa...?'

si sarà trovato disorientato in un mondo che sembra alquanto complesso e costruito solo ad uso e consumo di una ristretta élite, ma che con un minimo di ingegno e di apertura mentale può essere trasformato in qualcosa di davvero interessante da sperimentare con le proprie mani, mutando tutti i 'come' in 'aaaaaaaaaah!'.

Per continuare a leggere questo articolo non vi dico di studiare l'elettromagnetismo, le leggi di Ohm, di Gauss, il flusso del campo elettrico, le leggi di Coulomb, di Faraday, ecc. Ma di sicuro conoscerle e saperle adoperare anche solo per risolvere uno stupido esercizio di un circuito RC può farci comprendere in un qualsiasi circuito di qualsiasi apparecchio elettronico cosa sono tutti quei cosini lì incollati e quali sono le leggi che governano il flusso di elettroni che li attraversa. La teoria insomma ci può far scoprire alla fine che non è tutto così difficile come immaginavamo e magari farci diventare dei veri smanettoni dell'elettronica. E lo dico per esperienza personale: studiare che un condensatore accumula carica q, legata dalla legge:

$$C = q/V \rightarrow q = CV$$

oppure applicare nella realtà una batteria da 4,5V ad un condensatore e poi fare accendere un led per qualche secondo, per verificare effettivamente la presenza di carica, sono due approcci completamente diversi. Sicuramente, per chi ama sporcarsi le mani, il secondo approccio è ancora più interessante ed istruttivo del primo.

Ora rispondiamo al quesito (1), anche se in maniera semplicistica. Ai tasti del nostro PC sono applicate delle differenze di potenziale. Se schematizziamo i nostri tasti come dei condensatori a facce piane, la formula della Capacità, ovvero della quantità di carica che possono accogliere, fratto il voltaggio applicato alle armature varia, in funzione della geometria delle armature e della distanza tra le stesse.
Secondo la formula:

$$C = \frac{\varepsilon_o A}{d}$$

dove *d* è la distanza tra le armature. Se questa distanza diminuisce (pressione del tasto) allora la Capacità (C) cambia (in particolare aumenta) e quindi la carica sulle armature a parità di differenza di potenziale applicata aumenta di conseguenza.

Questo aumento è riconosciuto attraverso un determinato circuito e questa informazione, viaggia attraverso vari circuiti per essere riconosciuta infine dal nostro sistema operativo, che a sua volta identificherà da dove viene, cosa significa e cosa ci deve fare col segnale, in funzione di quello che sta eseguendo.

La parte più complessa nel realizzare un sistema del genere per chi si avvicina al mondo dell'elettronica è operare in funzione della variazione di stati. Ovvero, come si può far capire al mio circuito che c'è questo segnale? Com'è possibile farglielo interpretare correttamente e come in definitiva si può procedere dal punto di vista pratico?

A queste domande non darò una risposta teorica, anche perché non ne ho la facoltà, il tempo di andarmi a ricreare da zero un circuito che deve fare tutto quello che abbiamo detto sopra o quantomeno studiarmi un iter operativo per ottenere qualcosa del genere smanettando da solo.

Questo passo spiazza molti di quelli che come me hanno sempre avuto un approccio software agli apparecchi elettronici.

Quelli che trattano l'hardware ad un livello così logicamente astratto da fare dimenticare che in realtà sono pezzi di plastica e metallo attraverso i quali "scorre" (i puristi mi perdonino il termine incorretto) corrente elettrica.

Chi si sarà spinto oltre avrà trovato roba complicatissima, come crearsi da solo attraverso componenti spesso anche cari, apparecchi utili a decodificare diversi segnali per trasformarli in altri a loro volta da decodificare e con i quali operare. Beh, quando ci ho provato, ho quasi subito lasciato perdere. Preferisco il software, le istruzioni in semi-inglese, un controllo totale e la prevedibilità dei risultati.

In realtà non mi ero neppure posto la domanda se ci fosse una qualche via di mezzo. Programmare hardware sembra una cosa assurda da dire per chi come me è un profano dell'hardware e che probabilmente lo sarà sempre e comunque. Ma da qualche tempo avevo cominciato ad osservare un piccolo fenomeno in crescita, un piccolo componente elettronico che fa quello che gli dici di fare eseguendo un programma binario compilato da un linguaggio simil-Java (Processing). Un microcontrollore, una board, un "coso": Arduino.

Arduino

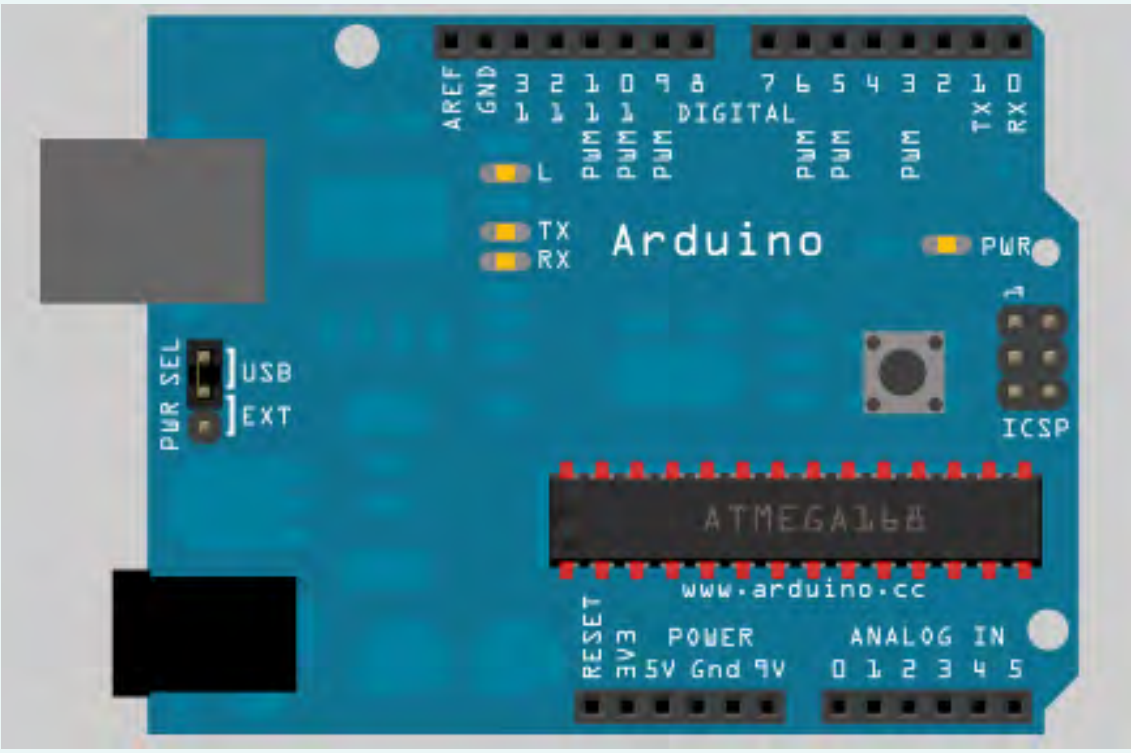
Milioni di anni fa qualcuno scrisse alla redazione di UAH: "Perché non parlate di Arduino? Perché sempre e solo software? Mi avete stracciato gli gnocchi!"

Bene, chissà dov'è finito quel lettore dopo tutto questo tempo! Adesso, grazie a quella sua richiesta sedimentata nel mio cervello, cercherò di accontentarlo e di avvicinare quanti di voi hanno avuto, come me, paura di cominciare a smanettare nel misterioso mondo dell'elettronica. Paura di non capire questo mondo, che poi così tanto lontano dalla programmazione non è!

Cominciamo introducendo l'oggetto del nostro piccolo studio.

Arduino cos'è?
Arduino è un piccolo strumento elettronico ricco di un fascino a metà tra l'oggetto ignoto e misterioso e il giocattolino stupido.

Arduino com'è?
E' fatto così:



(vi sfido a rispondere a questa domanda senza l'ausilio di un'immagine :D).
Dall'alto a sinistra abbiamo quel rettangolo grigio, che è l'interfaccia USB con la quale colleghiamo la nostra scheda Arduino al PC; le due striscioline forate nere alla destra sono i pin digitali, nel senso puro del termine, ovvero pin che possono dare/ricevere solo due segnali: 0/1 (Acceso/Spento, Bianco/Nero, Si/No...). Attraverso questi piccoli pin faremo i primi interessanti esperimenti, se li si può chiamare così. Vi sono in tutto 14 pin, numerati da 0 a 13.

Il blocchetto/chip/integrato con su scritto "ATMEGA" è il processore di Arduino, il suo cervello, la sua centrale operativa, quella che fondamentalmente esegue quello che gli dite di eseguire e quello che comanda tutta la board.

Sotto abbiamo due striscioline nere. La prima con la nomenclatura "POWER" racchiude dei pin utili per verificare la presenza di corrente elettrica pura e cruda sia in ingresso che in uscita. Notate che un pin fornisce 3V, uno 5V, ecc. Il pin che riporta 9V è l'ingresso per un eventuale pila da 9 volt al polo positivo, utile per lavorare con Arduino senza doverlo tenere attaccato al computer via USB o tramite il cavetto di alimentazione da rete con trasformatore.

Attenzione! Arduino lavora esclusivamente a 5V e anche se ha un circuito interno per il controllo e la regolazione di tensioni più grandi, è sempre consigliabile non eccedere con l'alimentazione. Insomma attaccare Arduino alla rete elettrica si può, ma poi sarà solo buono da bollire, con un contorno di insalata verde rinfrescante, con frutta di stagione e un buon vino bianco Chardonnay ad accompagnare il sapore duro e provocante della plastica sbruciacchiata :D.

La seconda strisciolina è quella dei pin analogici. Essi leggono, attraverso semplici collegamenti, i valori della tensione in ingresso interpretando il valore ricevuto come un intero, con valore massimo 1023 e minimo 0.

Arduino è un progetto totalmente italiano e già solo per questo meriterebbe di essere osservato anche da chi non si è mai avvicinato a questo tipo di argomenti. All'inizio era un semplice esperimento di hacking puro ma poi è diventato uno dei fenomeni di hacking italiano più diffusi in tutto il mondo. Basta fare una ricerca su YouTube per vedere l'infinità di progetti esistenti, facilmente documentabili e riproducibili avendo a disposizione una buona dose di componenti semplici da trovare e poco costosi, in caso qualcuno voglia crearsi da zero un arsenale elettronico :D.

Il creatore di Arduino, Massimo Banzi, ha costruito quasi per gioco questa board, che pian piano, senza alcuna previsione, ha avuto una larghissima diffusione diventando lo standard per nel suo genere. L'intero progetto è stato rilasciato con licenza open-source, quindi anche lo schema hardware e questo significa che chiunque può, avendo i giusti componenti (e una dose massiccia di preparazione per l'elettronica) crearsi da sé il proprio Arduino. Il suo prezzo di acquisto è più che accessibile: circa 26 €. Ottimi tutorial su come iniziare si possono trovare sul sito ufficiale del progetto:

http://arduino.cc

Navigando qua e là nel sito potete trovare il forum in lingua italiana, dove lo stesso Massimo Banzi risponde ai vostri dubbi, un wiki, la documentazione del software per programmarlo e una sezione dove potete comprare la board e tanto altro ancora... Beh, cliccateci su! Che aspettate?

Devo precisare per chi me lo ha chiesto in diversi forum/chat/altro che Arduino al costo di 26 € non prevede componenti, ma solo la board nuda e cruda, senza cavo USB, senza jumper senza breadboard, senza accessori. Se volete un insieme di componenti base per fare i primi esperimenti basta acquistarli a parte oppure acquistate lo starter kit al costo di circa il doppio (~ 50€).

Personalmente vi consiglio di comprare la board da sola, perchè i componenti comprati separatamente costano meno.

Per i primi esperimenti vi consiglio di prendere assieme alla board (magari nello stesso negozio online, per risparmiare sulle spedizioni):

10x Led 5mm/3mm
20x Resistenze assortite da 100Ω a 10KΩ
2x Fotoaccoppiatori
1x Display 7-segment
1x Sensore di temperature
10x PushButton
1x Potenziometro
1x breadboard con n punti (con n>300, n + ∞ :D) (spiegazione sotto)
1x set jumper (i fili di collegamento nella breadboard)

Per il resto gli altri componenti acquistateli solo se avete soldi da investire in questo hobby, perché fondamentalmente sono facilmente riciclabili da altre fonti.

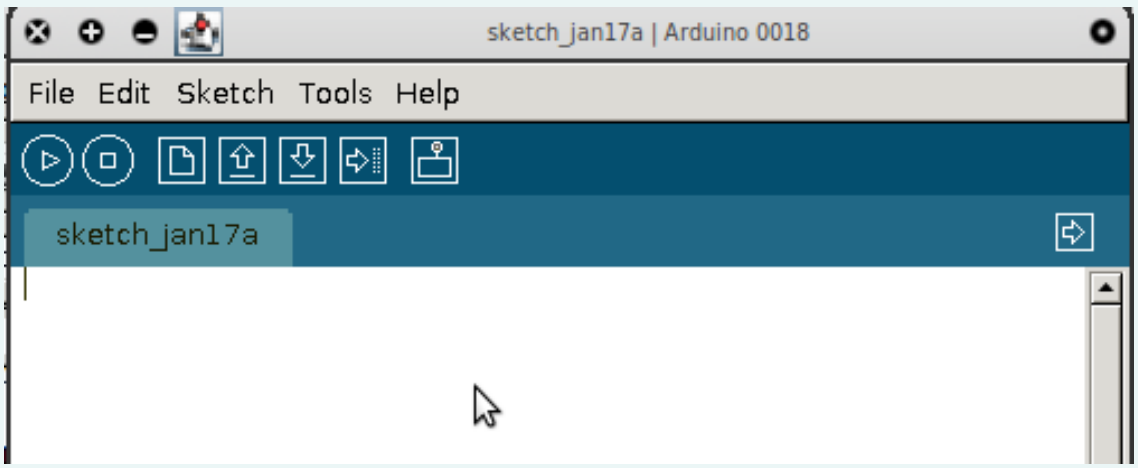
In un vecchio lettore CD ad esempio vi sono quasi sempre due motorini DC, in una vecchia cassa da PC vi è un altoparlante da 16 Ω. Personalmente ho riciclato quasi tutto, anche degli interruttori, oppure, cosa ancora più interessante, da un vecchissimo videoregistratore due Led IR (infrarossi), che ho usato per un progetto molto interessante che potete osservare nel blog che vi linko alla fine dell'articolo.

Aggiungo un piccolo parere personale e disinteressato: non per fare pubblicità, ma mi sento di consigliarvi di fare i vostri acquisti sul sito: http://www.robot-italy.com. Sono veloci a spedire e davvero professionali nell'imballaggio e nel packaging dei componenti.

Parliamo di software!

Sul sito di Arduino sopralinkato vi è anche la sezione software, dove potete scaricare l'IDE utile a compilare/uploadare i vostri software, leggere i dati che Arduino invia, ecc. L'IDE è scritto in Java ed è basato su un linguaggio chiamato Processing. La sintassi è semplice per chi ha familiarità con linguaggi C/C++ e con linguaggi di programmazione in genere. Chi non conoscesse nulla di tutto questo, beh vada a studiare prima di acquistare Arduino!

L'IDE ha un interfaccia semplice ed intuitiva:



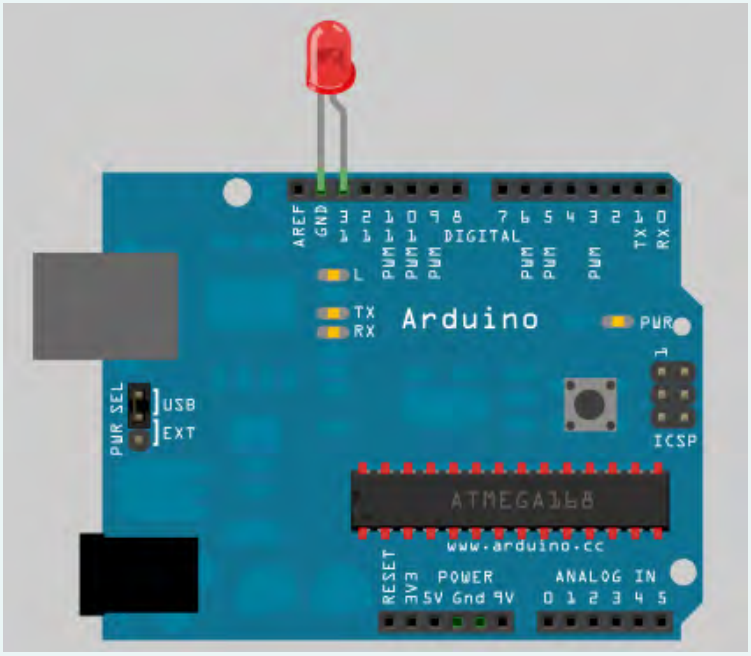
I tasti sopra la barra dei Tab, in cui verranno scritti i programmi veri e propri, sono di semantica semplice, ma ve li spiegherò lo stesso. Il primo da sinistra serve per compilare il sorgente in locale, per vedere se ci sono errori e fare un test locale puramente sintattico del listato appena codato. Il secondo, il terzo, il quarto e il quinto sono di facile lettura (stop all'esecuzione/apri/salva/ecc.).

Gli ultimi due sono i più importanti: il penultimo serve per caricare su Arduino il programma scritto dopo averlo compilato, l'ultimo si chiama Serial Monitor ed è un programma utile per osservare cosa ritorna Arduino all'IDE, attraverso la funzione: Serial.print().

Sperando di non aver dimenticato nulla, (se sono stato poco esaustivo vi ricordo che sopra è linkato il sito ufficiale), passiamo alla fase operativa con un paio di esempi pratici.

Arduino in pratica: Led Blink-Switching

Anche se questo esempio è davvero banale, è grazie ad esso che ho capito in realtà la potenza operativa di questo piccolo ed affascinante tool :D.
Il tutorial più semplice che trovate sul sito ufficiale, vi illustra, in maniera fin troppo dettagliata, come far accendere un led e come farlo blinkare ossia farlo accendere ad intermittenza... lampeggiare insomma. Se aveste a disposizione solo una batteria da 4,5V (e un led ovviamente :D) mettereste i piedini del led nel senso corretto (per i più niubbi dove c'è la lancia all'interno si attacca al polo positivo) e senza un interruttore otterremmo solo un'emissione luminosa... ma noi vogliamo che blinki! Non deve restare fisso e se vi scoccia staccare e attaccare un interruttore con intervalli definiti dalla vostra pigrizia, Arduino può fare questo pesante compito per voi.
Schemino:



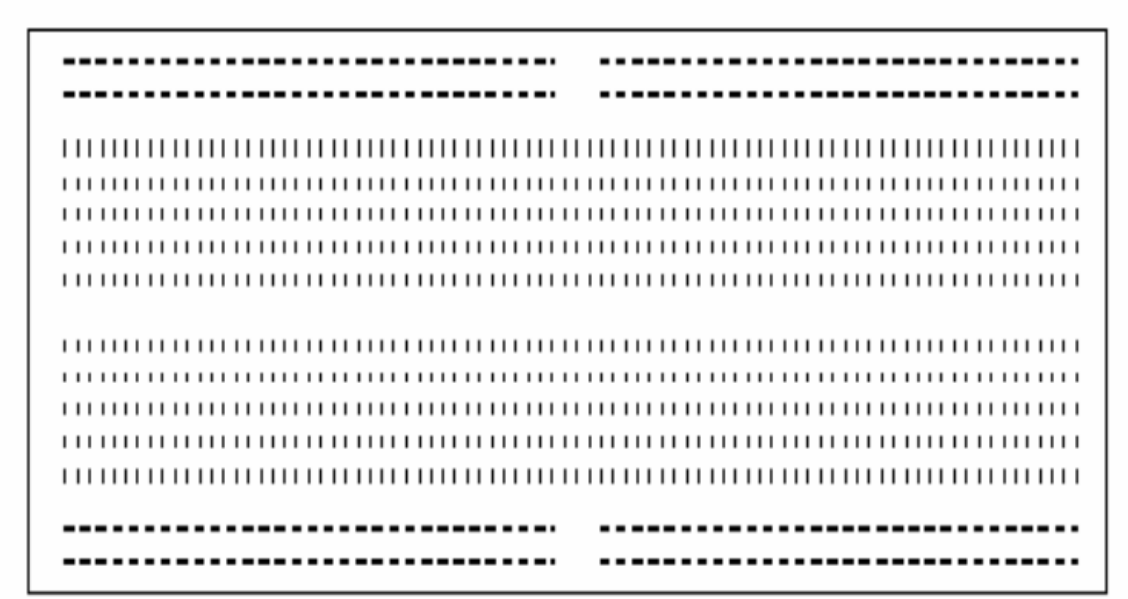
Come potete notare, il piedino che ho denominato "con la lancia" sta nel pin 13 di Arduino mentre quello negativo sta nel pin con la dicitura GND => Ground →Terra →Massa, che in poche parole è il polo negativo. Tutti i pin con scritto GND sono collegati e possono essere usati indifferentemente. Se viceversa mettessimo entrambi i piedini in due pin numerati, il led non si accenderebbe perché non essendoci differenza di potenziale tra i due poli non vi è alcuna corrente elettrica nel led e quindi nessuna emissione di luce.
Ma avendo l'Arduino puro senza alcun programma caricato su, avrete lo stesso esatto risultato di un led collegato male... quindi non demoralizzatevi. Collegate il led come in figura e poi scrivete il seguente codice nell'IDE:

```
int ledPin =  13;    // LED connesso al pin 13 inizializziamo la variabile a 13

// La funzione setup si avvia solo la prima volta

void setup()   {
  // inizializziamo il pin 13 ad OUTPUT, ovvero ad emettere o meno un segnale
 // essendo un pin digitale non potete far altro che questo con il pin 1 o 0
  pinMode(ledPin, OUTPUT);
}

// La seguente funzione è il cuore di tutti i programmi, essa
//verrà eseguita da arduino finchè avrà corrente

void loop()
{
  digitalWrite(ledPin, HIGH);   // scrive HIGH (1, ovvero c'è
                                // corrente)
  delay(1000);                  // aspetta un secondo
  digitalWrite(ledPin, LOW);    // scrive LOW (0 assenza di
                                // corrente)
  delay(1000);                  // aspetta un secondo
}
```

Una volta scritte queste righe di codice, che ho preso dall'esempio di default e di cui ho grossolanamente tradotto i commenti in inglese, dovete premete quel tasto PLAY visto in precedenza per assicurarvi della correttezza del codice... Bene, ora collegando Arduino al PC, dovete premete il pulsante: Upload To Board... e dopo qualche secondo BOOM!!! (scherzo, spero di no). Niente esplosioni ma si dovrebbe accendere e spegnere il led ad intervalli di 1 secondo... Se variate i valori di delay, mettendo, ad esempio, delay(10000) il led sta acceso 10mila millisecondi, ossia 10 secondi...
Quest'esempio è tanto banale quanto istruttivo perché vi fa capire che per collegare già due led, la sola board non vi basta, in quanto dovreste mettere tutti i piedini negativi in collegamento col ground e questo può essere un problema! Ma ecco che viene in nostro aiuto uno dei componenti che vi ho invitato a comprare: la BreadBoard.

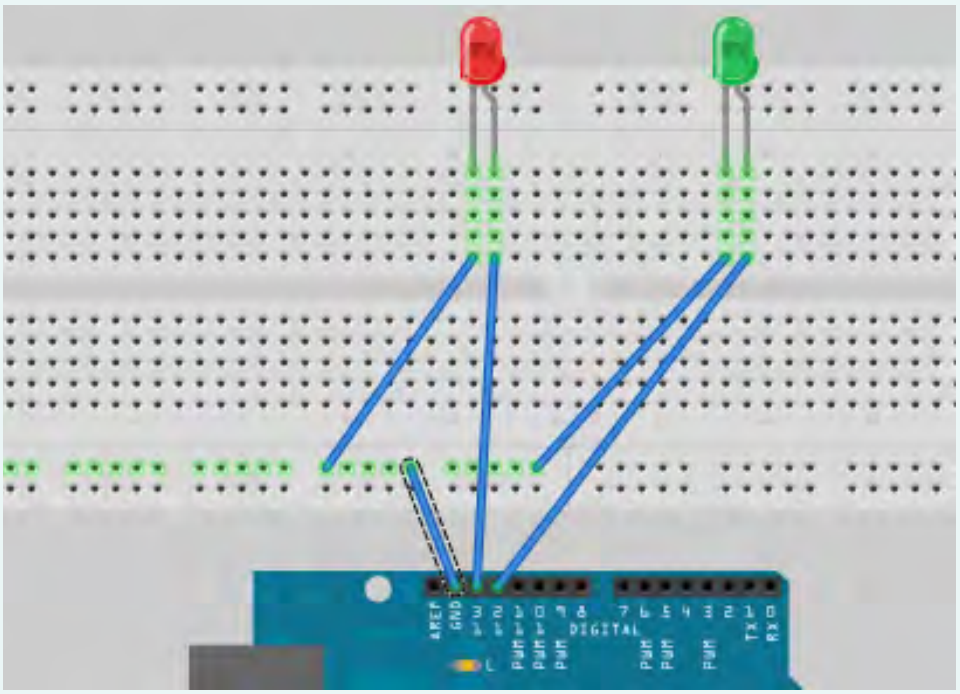
Breadboard

Non è una tavolozza per tagliare il pane (come il nome suggerisce) anche se lo stesso Banzi nei tutorial su YouTube la descrive come "una ciabatta" in cui si moltiplicano i pin. Nella ciabattale varie strisce sono connesse secondo questo schema:



Le strisce laterali sono tutte connesse tra di loro lungo il nostro asse x, invece quelle centrali sono connesse tra di loro lungo l'asse y.

Ed ecco così risolto il problema di collegamento di due led:



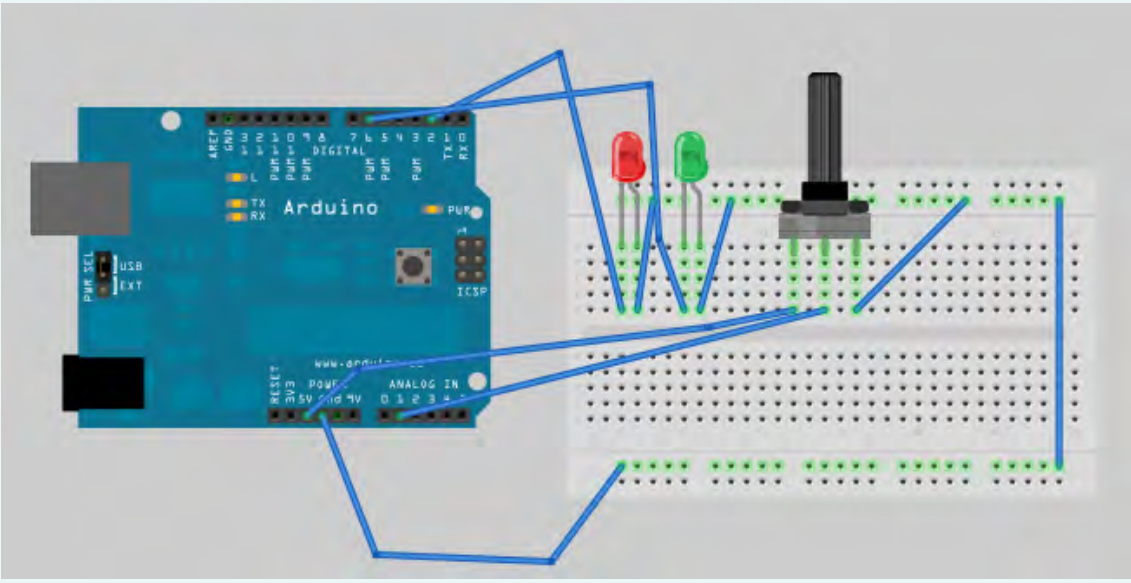
Come potete osservare ho collegato in questa immagine due pin (il 12 e il 13) a due striscioline dove sono posizionati i led nel polo positivo, e i pin negativi vanno entrambi connessi al ground... Semplice no? Potremmo riadattare il programma per farlo diventare un semaforo:

```
int redled = 13;
int greenled = 12;

void setup(){
     pinMode(redled, OUTPUT);
     pinMode(greenled, OUTPUT);
}

void loop(){
     digitalWrite(greenled, LOW);
     digitalWrite(redled, HIGH);
     delay(5000);
     digitalWrite(redled, LOW);
     digitalWrite(greenled,HIGH);
     delay(5000);
}
```

Semplice ed intuitivo... Ora basta compilare e caricare e... avrete un semaforo per formiche! Molto utile per ridurre la mortalità delle stesse negli incroci di notte dopo essere state in disco dai fuchi, ubriache di afidi... Lo so, questa faceva schifo :-(.

Torniamo ora a ciò di cui parlavo in questo titolone: switching led!
Ovvero, avendo un potenziometro, posso decidere quale led tra i due accendere? Primi elementi di lettura dei pin Analogici.

Lo schema di collegamento è lo stesso, si aggiunge solo un particolare:



Se questo schema vi sembra complesso e brutto le cose sono due:

1. Cambiate Hobby
2. Andate a cagare :D

In elettronica è difficile fare cose belle da vedere (punti di vista :D).

Ora leggiamo il codice della libreria di esempi standard tradotto in italiano da me:

```
int sensorPin = 1;          // il pin dove collego il piedino del
                            //potenziometro che misura la resistenza
int redPin = 2;             // pin del led rosso
int greenPin = 6;           // pin del led verde
int sensorValue = 0;        // variabile che conterrà il valore del
                            //sensore

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop() {
  // ecco come leggo la variazione di resistenza
  sensorValue = analogRead(sensorPin);

  //se il valore del sensore è > 1020 allora accendo il led rosso
  if (sensorValue>1020){
     digitalWrite(redPin, HIGH);
     digitalWrite(greenPin, LOW);
  }else{
//altrimenti accendo quello verde
     digitalWrite(redPin, LOW);
     digitalWrite(greenPin, HIGH);
  }
}
```

Così, con un potenziometro potrete semplicemente switchare quale led accendere... Una stupidata, ma molto istruttiva! Arduino, come detto, dai pin analogici legge la variazione di tensione e in questo esempio la provochiamo con un potenziometro, che ha una resistenza variabile grazie ad una manopola girevole. Tutti i sensori di temperatura, gli accelerometri, ecc. funzionano con lo stesso principio del potenziometro nei vari circuiti elettrici.

Che altro dire? Vi sono migliaia di altri esempi in giro per il web, sia su YouTube che altrove. Se mi contattate privatamente vi dò anche tanti altri link, ma qui volevo citare almeno i seguenti:

http://isnotnull.hellospace.net
Un mio blog
Ci trovate oltre a questo, altri esempi con video e documentazione adatta.

http://fritzing.org
Fritzing
Un programma utilissimo a prototipare circuiti elettronici con disegni e collegamenti (tutte le immagini di questo tutorial sono fatte con fritzing).

http://gioblu.com
Gioblu Robotics
Blog molto interessante dove qualcuno posta spesso articoli su Arduino con tanto di schema dei circuiti e spiegazione passo passo.

http://google.it
Google
Sito molto interessante per riprenderti dopo qualche cortocircuito e cercare se qualcuno prima di te ha pensato di risolvere quel problema elettronico in una qualche maniera, sicuramente più intelligente di te.

Piccola menata finale

Concludo sperando di aver fatto breccia in qualche cuore spavaldo che smetterà di pensare all'elettronica come l'insieme di quei pezzi inanimati governati dal software, ma comincerà ad immaginarli come quell'importante anello di congiunzione tra fisica tangibile dell'elaborazione dell'informazione e della filosofia informatica software, sempre più lontana dalle proprie radici.

vikkio88

Sharepoint Designer 2010
DLL HIJACKING

Cos'è Sharepoint Designer?

Microsoft SharePoint Designer (conosciuto anche con il nome Office SharePoint Designer) è un software pensato per essere impiegato per la progettazione di siti web e per l'editing di contenuti HTML. In connessione con la piattaforma Microsoft SharePoint Server consente la creazione e la modifica di portali e siti web in modo condiviso e con diverse modalità di accesso e di operatività. Con la piattaforma SharePoint è possibile creare liste, repository di documenti, calendari sincronizzati con Outlook e molto altro ancora.
La parte client della soluzione, denominata SharePoint Designer, era parte della famiglia di prodotti Microsoft Office 2007, ma non è stata inclusa nelle versioni successive della suite. Assieme a Microsoft Expression Web, SharePoint Designer è il successore della soluzione Microsoft FrontPage, anch'esso in precedenza distribuito con la famiglia di prodotti Office. Molte delle caratteristiche di FrontPage sono infatti presenti in SharePoint Designer come molti componenti web, database objects, applet, barre di navigazione, inserimento mappe, ecc.
Uno dei punti di forza di SharePoint Designer è la condivisione dell'accesso alle risorse e ai contenuti di un sito web. Due o più utenti, infatti, possono connettersi contemporaneamente e visionare e lavorare sullo stesso documento, attraverso le operazioni definite "check-out" (estrazione) e "check-in" (archiviazione), per evitare conflitti.

Cos'è il DLL Hijacking?

Con questa espressione (letteralmente "dirottamento DLL") si intende una tecnica di hacking che sfrutta una vulnerabilità software che s'innesca quando un file eseguibile o di libreria viene aperto all'interno di una directory controllata da un attaccante. Nella maggior parte dei casi l'utente vittima deve materialmente aprire uno specifico programma o un suo file accessorio affinché l'exploit funzioni correttamente e permetta all'attaccante di penetrare il computer bersaglio. Il file o programma che viene aperto dall'utente vittima può essere all'apparenza completamente innocuo, ma il problema è che l'applicazione eseguita caricherà una libreria DLL (Dynamic Link Library) appositamente creata e depositata nella directory di lavoro dell'applicazione o del file che è associato all'applicazione.

Dal punto di vista pratico, questa vulnerabilità può essere sfruttata spedendo all'utente bersaglio un link ad una condivisione di rete che viene considerata dal bersaglio come sicura. Nel momento in cui l'utente cerca di aprire un file presente nella directory, l'applicazione corrispondente viene eseguita e se l'attaccante ha avuto modo di depositare una libreria DLL che contiene malware, shellcode o altro codice maligno, questa DLL verrà caricata nel processo dell'applicazione al posto di quella originale. Questo farà sì che l'exploit entri in funzione e consentirà all'attaccante di prendere possesso della macchina dell'utente vittima o di sfruttarne le risorse in qualche modo. La tecnica DLL Hijacking è stata scoperta da HD Moore, il quale ha pubblicato un exploit per Metasploit, la piattaforma software open-source di penetration testing.

Perché proprio SharePoint Designer 2010?

Semplicemente perché, al momento, è vulnerabile ad attacchi di tipo DLL Hijacking. Il funzionamento di un exploit per SharePoint Designer è pressoché banale e rientra perfettamente in questa classe di vulnerabilità. In sostanza, come detto, viene sfruttato il meccanismo di caricamento delle librerie DLL di Windows. Le applicazioni Windows, come è noto, sono costituite da uno o più file eseguibili **.exe** che possono richiamare, secondo necessità, le librerie di collegamento dinamiche DLL su percorsi (path) predefiniti. Se una certa libreria non viene trovata in uno dei percorsi predefiniti, solitamente l'applicazione arresta la sua esecuzione e rilascia un errore che segnala la mancanza della DLL necessaria.

In queste condizioni, la prima operazione che viene compiuta dal processo dell'applicazione è quella di ricercare la DLL mancante all'interno dello stesso percorso di esecuzione, quindi nella stessa directory di lavoro del file eseguibile oppure nella stessa cartella in cui si è tentato di aprire un file associato all'applicazione. Successivamente la ricerca viene effettuata anche su altri percorsi standard di Windows, ad esempio su **c:\programmi** e su **c:\windows\system32.** L'exploit consiste nel depositare nella stessa directory dell'eseguibile o del file associato una libreria DLL di cui il programma fa uso, ma modificata opportunamente in modo che contenga uno shellcode, un malware o altre istruzioni utili per attaccare le risorse della macchina. La libreria DLL "fasulla" viene caricata immediatamente da SharePoint Designer, senza che questo modifichi o acceda in alcun modo ai percorsi c:\programmi o c:\windows\system32 (in effetti non sempre il processo dell'applicazione ha accesso a quei percorsi).

Il risultato finale è che il processo che esegue SharePoint Designer carica una libreria DLL che ne consente la continuità di esecuzione, ma poiché essa contiene altro codice, questo verrà eseguito, permettendo di accedere a tutte le risorse cui lo stesso processo di SharePoint Designer ha accesso. E se l'utente del processo è di tipo Administrator, allora...

Come si può costruire una DLL modificata per l'hijacking?

Quella che segue è una semplice DLL. Nei commenti al codice è spiegato cosa fanno le funzioni che vi sono state inserite. Per compilarla si può utilizzare Dev-C++, un ambiente di sviluppo / compilatore gratuito per Windows. Lo trovate qui:
http://bloodshed-dev-c.softonic.it/

Codice:

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
/*
Questo è un semplice metodo che esegue una MessageBox e poi
richiama l'eseguibile calc.exe dalla directory system32 di Windows
*/
```

```
DLLIMPORT void HelloWorld ()
{
char message[] = "Ciao sono una DLL farlocca!\nVengo eseguita automaticamente da Sha-
repoint Designer\nEssendo una dll sono soggetta a code injection o ad overwrite!\nOra
eseguo calc.exe (perchè sono una DLL injection non maligna) ma potrei eseguire anche
altro!...\nPotrei anche non avvisare quando mi eseguo :p";
MessageBox (0, message, "Sharepoint Designer bacato!", MB_ICONINFORMATION);
(void)system("%windir%\\system32\\calc.exe");
}
/*
Nel costruttore della dll non faccio altro che controllare la reason... se mi trovo in
process attach richiamo il metodo dichiarato sopra
*/
BOOL APIENTRY DllMain (HINSTANCE hInst, DWORD reason, LPVOID reserved)
{
switch (reason)
{
  case DLL_PROCESS_ATTACH:
    HelloWorld();
    break;
  case DLL_PROCESS_DETACH: break;
  case DLL_THREAD_ATTACH: break;
  case DLL_THREAD_DETACH: break;
}
  return TRUE;
}
/* EOF */
```

Come si esegue l'exploit?

Per costruire la libreria DLL "fasulla" da dare in pasto a SharePoint Designer 2010 è sufficiente compilarla con Dev-C++ e poi rinominarla mfc90eng.dll o con uno qualunque degli altri nomi di librerie DLL che vengono richiamate dall'applicativo di casa Microsoft.

Per effettuare un test con la libreria (innocua) del nostro esempio, è meglio procedere con la creazione di una cartella di nome sd_exploit in cui inserire un file di tipo .ascx, ad esempio sd_exploit.ascx. Il file può anche essere vuoto, l'importante è che abbia l'estensione .ascx. Nella cartella sd_exploit occorre inserire la libreria DLL modificata. Quando l'utente Windows cliccherà sul file sd_exploit.ascx nel tentativo di aprirlo, l'applicativo SharePoint Designer verrà caricato e con esso la nostra libreria DLL, il cui codice sarà quindi eseguito automaticamente, dando vita all'exploit.

Ovviamente nello scenario di un vero e proprio attacco ad una macchina Windows in cui sia installato SharePoint Designer 2010, occorre fare in modo che l'utente abbia accesso ad una cartella in cui si trovino sia il file .ascx che la libreria DLL contenente il codice (più o meno) maligno.

cercamon
exploit by Giuseppe "cyberdude" Ferraro

FreeCompany()

Il tema dell'uso di software open-source in ambito aziendale è ormai un vecchio ritornello.
Questa analisi punta ad avere un approccio diverso da parecchia documentazione che si trova in rete per due caratteristiche principali:
– vuole essere dedicata all'impresa e non agli informatici; troppe volte la visione del problema risulta totalmente distaccata dalle logiche economico-organizzative e quasi politicizzata agli interessi dei tecnici.
– non sarà dedicata alla pubblicità dell'open-source, come accade in molte pubblicazioni in cui l'etica del software libero sembra sovrastare il fine ultimo dell'impresa che è commerciale e non politico/umanitaria.
Dalla trattazione saranno necessariamente esclusi i casi di utilizzo di software pirata, cosa che a dire il vero dovrebbe essere presa in considerazione.
Togliamoci dalla testa i formalismi e pensiamo alla realtà: ad esclusione delle imprese legate intensamente al mezzo informatico per la loro attività produttiva, nessuno presta particolare attenzione alle licenze software come i sistemi OEM o i programmi 'freeware per uso personale'.
Ricordo in questo caso che ad un eventuale controllo sul software l'azienda risulterebbe non a norma riguardo l'uso dei prodotti nel caso il pc sia intestato alla persona giuridica e non all'utilizzatore; per andare al sodo nel pc aziendale AVG-free non è permesso, mentre lo è nel pc acquistato a nome dell'agente (stessa cosa accade ad esempio per VisualStudio Express nel pc dello sviluppatore).
Per quanto mi riguarda preferisco sempre e comunque informare la direzione su questa serie di problematiche lasciando a loro la decisione; è vero che molte volte la scelta di trasgredire le regole arriva proprio dall'alto ma lo considero un atto dovuto per motivi di professionalità visto che l'ambito informatico è nostra competenza e non dell'amministrazione. Opinioni personali, più o meno apprezzabili.
In questo documento si cercherà di rispondere ad alcune domande fondamentali: conviene (e a quali condizioni) il passaggio parziale o totale all'open-source per la PMI? Con quali modalità questo processo può essere svolto?
Le risposte verranno dalle esperienze professionali di chi scrive e sarà data particolare attenzione al 'come' fare, anziché indicare il 'perché' farlo.

Tips

La prima cosa che va definita è cosa si intende per utilizzo di software open in azienda, problema assai più complesso di quanto si può immaginare.
Una struttura basata totalmente su software libero implica l'adozione di tale modello a livello di sistema operativo e delle applicazioni in tutte le macchine presenti; cosa che ordinata ai tecnici aziendali potrebbe gettare un certo sconforto.
La maggior parte della letteratura che analizza questi problemi tende a considerare poco o nulla alcuni elementi pratici che non possono essere ignorati:

1. La volontà di passaggio all'open-source avviene al 90% per la riduzione dei costi che comporta. Le altre valutazioni di tipo tecnico sono probabilmente più importanti ma difficilmente concepibili da chi non si occupa direttamente del ramo informativo.

2. La ricerca di riduzione dei costi prevede che i costi siano presenti al momento o nell'immediato futuro. In altre parole esisterà quasi sicuramente una struttura non basata su software libero, oppure un progetto attuabile con software proprietario in tempi ridotti e costi perfettamente quantificabili.

3. Sebbene l'efficienza possa essere messa in discussione, la struttura presente in un dato momento garantisce il risultato attuale. Il principio secondo cui 'la produzione non si deve fermare' è la base primaria dell'azione direzionale ed è una questione inconfutabile; ogni migrazione è possibile se e quando garantisca il successo e la riduzione dei costi.

Una volta fatte le dovute premesse è arrivato il momento di vedere all'atto pratico quali sono le possibilità offerte da strutture open rispetto alle controparti commerciali.
È disponibile in rete una vasta documentazione sulla struttura di costo del software aziendale;
spesso tende a specificare complesse forme di valutazione che forniscono una precisione puntuale ma finiscono con il perdere quel fattore di intuitività che è utile nell'ambiente delle PMI.
Se dovessi spiegare di persona quali sono i vantaggi di piattaforme aziendali open-source mi soffermerei su tre caratteristiche principali:

1. minore costo di acquisto del software: il software open è gratuito o ha costo decisamente minore di quello proprietario (sicuramente come costo complessivo, non sempre sul singolo prodotto).

2. minor costo di acquisto dell'hardware utilizzato: il software open su macchine a sistema proprietario consumano generalmente meno risorse; se poi anche il sistema risulta open la durata di vita della macchina si allunga notevolmente, anche in relazione della personalizzazione del sistema in base alle caratteristiche hardware.

3. minore necessità di assistenza: questa affermazione so che può suscitare scalpore, però è un dato di fatto che gli addetti ai lavori potranno confermare senza problemi. I client possono non essere aggiornati ad un rischio nettamente minore di quanto accade su client Windows, ogni modifica attuata può essere facilmente clonabile per portare uno stato di default in un tempo irrisorio e senza reinstallare.

Si parla spesso del fattore assistenza considerando che nei software proprietari questa sia migliore 'per principio', nella mia esperienza ho notato che il servizio è generalmente uguale purché quel fattore venga preso in esame al momento dell'acquisto (non meno di quanto accade per il software proprietario), oltretutto è rilevante il fatto che sul software open si trova una folta letteratura e molti problemi sono risolvibili internamente, incluse eventuali modifiche e/o implementazioni interne che in ambienti proprietari probabilmente violerebbero licenze e condizioni di garanzia.
Letti questi vantaggi tutto sembrerebbe rose e fiori e la documentazione 'filosofica' sull'argomento tende a fermarsi qui, cosa che io non voglio assolutamente fare.
Una migrazione è un progetto vero e proprio come molti altri progetti che ogni azienda implementa nella sua vita e come tale va trattato, non è corretto né prenderlo alla leggera come fosse l'acquisto di un nuovo pc né scartarlo a priori perché (a priori) è inutile.

Quali sono quindi i problemi generati da un programma di migrazione all'open-source?
Anche in questo caso esiste parecchia documentazione che il più delle volte risulta viziata da motivazioni puramente commerciali o da testimonianze di fallimenti annunciati per scarsa o nulla programmazione.
Anche in questo caso segnalerei tre fattori di criticità da valutare:

1. tempi del cambiamento: è incredibile come in pochi anni tutti imparino ad usare telefoni che fanno tutto, mentre da decenni spostare un tasto da destra a sinistra dello schermo manda nel panico chiunque... Il principale fattore contrario all'adozione di software open è strettamente umano, provate a prendere una comune segretaria e sostituire il suo Word con OpenOffice per rendervi conto della realtà. È fondamentale per chi progetta il passaggio occuparsi di come renderlo il più indolore possibile, ricordando che il tempo è di primaria importanza; se la migrazione è troppo rapida le problematiche generate si accumulano in modo da diventare irrisolvibili, se è troppo lenta gli effetti diluiti non ne mostrano gli eventuali vantaggi. La risposta è programmazione (nel senso organizzativo...se non si era capito...)

2. gestione della multi-struttura: la migrazione prevede un tempo più o meno lungo (molte volte eterno) di compresenza di più software e più sistemi, con problemi di compatibilità e di moltiplicazione dei disagi. Pensate solo al tipico caso di presenze proprietarie di WindowsXP + Vista + Seven e ampliatele di conseguenza.

3. valutazione dei costi del cambiamento: questo fattore è comprensibile solo alla luce dei primi due anche se dovrebbe essere il primo punto, ed è importante più per la direzione che per il settore tecnico. Se il costo del software è rilevante allora altri fattori non ben quantificabili passano in secondo piano; molto diverso è il caso in cui i costi predominanti (punti 1 e 2) non siano valutabili, non fosse altro che per motivi strettamente fiscali. Chiudendo il ciclo torniamo nuovamente a parlare del *tempo*, cioè l'eventuale risparmio è valutabile al meglio come differenza dei costi iniziali e finali ammortizzata nel periodo di migrazione, confrontando i risparmi con i costi sostenuti nei periodi di ammortamento. Tempi più stretti, risparmi maggiori come detto prima; *programmazione*; collaborazione tra direzione e tecnici.

Dalle precedenti valutazioni è stato escluso però l'argomento principale, cioè l'aspetto che riguarda il capitale umano.
Se è innegabile il fatto che la competenza informatica del personale tecnico è il fattore principale di riduzione dei costi indipendentemente da qualunque struttura sia presente, è altrettanto vero che le infinite possibilità generate dall'adozione dell'open-source trovano piena manifestazione soltanto con un personale preparato.
Nel lungo periodo i risultati più apprezzabili ricavati dall'adozione di sistemi open è rappresentato dalla possibilità di sviluppo e modifica di applicazioni ad un costo quasi irrisorio; il fatto di avere a disposizione codici sorgenti anche a livello di sistema permette di lavorare su materiali già pronti anziché sviluppare da principio software utile all'attività aziendale. Progetti che potrebbero richiedere mesi di lavoro vengono invece realizzati in tempi ridottissimi, anche grazie ai linguaggi di scripting presenti di default nelle varie distribuzioni.
È chiaro che non tutte le aziende hanno a disposizione un CED con sviluppatori esperti di sistemi UNIX, però la programmazione non è una necessità e di codice già pronto se ne trova senza alcun problema.

Dal punto di vista del semplice utilizzo di un sistema GNU/Linux si potrebbe obbiettare il fatto che pochi addetti tecnici siano pratici di quella piattaforma, va però valutato il fatto che oggi le interfacce grafiche sono decisamente evolute e non è più necessaria una competenza di alto livello per gestire una macchina. In un breve periodo di apprendimento è un fatto che qualunque tecnico sia in grado di destreggiarsi senza problemi, anche per la vasta documentazione esistente;
personalmente non considero un problema il tempo di formazione dei sistemisti.
Se poi si valuta l'inserimento di una nuova figura che si occupi dell'aspetto informativo d'azienda stiamo pure terra terra... un esperto di sistemi liberi e un comune tecnico Windows lo pagate sempre la stessa cifra.

Dopo la parte teorica precedente, necessaria per una corretta comprensione del problema, possiamo passare a qualcosa di più pratico che tracci le linee generali del processo di migrazione all'open-source in azienda.
Se da un lato si può dire che gli aspetti pratici sono spesso trascurati, è anche da notare come ogni realtà abbia le proprie caratteristiche e sia molto difficile individuare un metodo univoco.
Quanto segue è frutto delle mie esperienze personali dopo anni di lavoro in ambiente open e un certo numero di esperienze di migrazione, a dire il vero non mi ero mai posto il problema di trarre una sorta di vademecum da quello che ho tratto nel tempo; meglio così.

Tricks

Se dovessi indicare un percorso schematico di una migrazione identificherei tre fasi principali:

1. L'analisi della struttura informativa presenta in azienda
2. La fase progettuale e preparatoria
3. La fase applicativa

Ogni fase dovrebbe essere un contenitore stagno, che nasce sulla base dei dati acquisiti nella fase precedente e si concretizza in modo completo prima della sua chiusura.
Il vero pericolo che origina eventuali fallimenti è infatti legato all'eccessiva corsa al risultato finale, con un prolungamento eccessivo dei tempi di realizzo per la necessità di tornare sui propri passi nel momento in cui il traguardo sembra a portata di mano.

Analisi strutturale

La prima cosa da fare è l'analisi funzionale di tutte le macchine presenti, difficilmente in azienda si utilizza hardware estremamente innovativo (a livello di case) quindi è difficile che vi siano problemi di compatibilità. Non altrettanto accade per l'hardware legato alle varie postazioni (scanner, stampanti, impiantistica disparata) a cui si aggiunge il software utilizzato.

Tale analisi deve rispondere alle seguenti domande:

1. Quali e quanti modelli di postazione sono presenti nella struttura aziendale?
2. Quale hardware viene gestito dalle postazioni?
3. Quale software è utilizzato dagli operatori?

Questa prima fase è fondamentale e se svolta in maniera accurata permette una buona programmazione dei lavori successivi, un considerevole risparmio di lavoro e una migliore identificazione e gestione dei costi e dei risparmi ottenibili.
È buona norma da parte dei tecnici raccogliere anche le impressioni degli addetti alle postazioni, in modo da poter gestire anche eventuali richieste di miglioramenti che potranno poi essere prese in considerazione nella fase di rinnovamento che seguirà.

Fase preparatoria

La seconda fase è più progettuale ed è quella che maggiormente si basa sulla competenza e la professionalità del personale tecnico.
Per comprendere il tutto è bene tornare ad enunciare dei principi detti in precedenza: esiste già una struttura funzionante su software proprietario, esiste un rifiuto di fondo alla migrazione, un processo troppo rapido crea sicuramente più problemi di quanti ne risolva.
A meno che non esistano elementi di novità tali da poter essere gestiti immediatamente su sistemi liberi (nuovi modelli di postazione), è fondamentale che soltanto i tecnici lavorino su tali sistemi (probabilmente con il tipico dual-boot o tramite virtualizzazione) a scopo di test, con uguale priorità di questi per l'utilizzo di software libero su sistema proprietario.
È necessario infatti creare un modello informativo nuovo che sia in grado di gestire più o meno totalmente il modello attuale, l'operazione va effettuata su due ordini di lavoro:

1. identificare i software disponibili in multi-piattaforma e/o i sostitutivi liberi alle applicazioni proprietarie.
2. impostare e verificare il porting degli applicativi su piattaforme libere.

Il primo punto riguarda la gestione dei sistemi propietari esistenti, sarà cioè necessario un primo passo in cui rimanendo invariata la piattaforma saranno sostituiti i software, in modo da slegarli dal sistema usato.
Dalla mia esperienza ho maturato l'idea che i 'pionieri' (o cavie...) di questo processo è bene siano i capi-ufficio e i capi-reparto per motivi più psicologici che tecnici: lo stretto rapporto che si crea con i tecnici può farli sentire scavalcati se vengono presi soggetti diversi da loro, 'sentire il progetto' li coinvolge maggiormente con una profiqua reazione a cascata sugli altri membri, al termine del percorso saranno loro ad educare i membri inferiori con un senso di potere che invoglia. Queste sono solo idee personali e non legge.

Il secondo punto è destinato al personale tecnico e riguarda invece la nuova piattaforma che andrà in essere.
Tra i vari modelli di postazioni identificate nella prima fase dovranno essere individuate e testate le modalità di gestione da parte di un sistema libero, sia in ambito hardware che software (ERP/CRM in primis); gli esperti di GNU/Linux sanno che oramai quasi tutto l'hardware è supportato e la letteratura in merito ricavabile anche con una semplice ricerca in rete è decisamente folta, i casi di impossibilità dovrebbero essere minimi, dato di fatto che serve come verifica del raggiungimento del traguardo di compatibilità (cioè se siete al 50% è bene mettersi al lavoro, se siete al 90% forse è il caso di gettare la spugna).

In questa fase sconsiglio di ricorrere alla mera sostituzione dell'hardware con modelli compatibili o del software commerciale per piattaforma diversa, è una cosa da valutare solo quando il processo è completo per evitare costi inutili.

Per andare agli aspetti strettamente pratici di questa fase vorrei soffermarmi su qualche argomento che merita attenzione.
Il primo 'amico' è un software su cui quasi sicuramente si andrà a sbattere la testa, cioè la suite OpenOffice. Probabilmente sarà la prima mossa per rispondere al primo punto di questa fase, la suite ha una buona manualistica e fornisce normalmente tutte le funzioni di MS Office; è necessario un certo periodo di ambientazione per gestire bene le funzioni degli applicativi (soprattutto per le funzioni dei fogli di calcolo e i database) ma se la procedura viene svolta dal settore tecnico prima che dagli uffici comuni il passaggio risulta piuttosto rapido e indolore.
Di default il programma non dispone di un tool analogo ad Outlook, richiede quindi l'integrazione di un programma come Thunderbird per gestire messaggistica e promemoria; l'abbinamento permette anche di giocare su estensioni di OO e plugin di TB con ottime soluzioni, decisamente superiori a quelle della suite di Microsoft.

Una seconda coppia di software testata nella mia esperienza va trattata insieme anche se gli utilizzi sono diversi: WINE e Cygwin.
I due modelli di porting rappresentano i due volti di congiunzione tra sistemi e permettono alla struttura informativa di risultare molto più omogenea di quanto sarebbe in realtà; in questa fase le due applicazioni sono dedicate chiaramente la prima al settore tecnico e la seconda ai vari client aziendali.

WINE (porting di applicazioni Windows sotto GNU/Linux) verrà testato per far girare applicativi non compatibili con sistemi diversi da Windows, in particolar modo per i client e server dei sistemi ERP (se non dotati di interfaccia web) ed eventualmente per dispositivi non supportati nativamente.
Ricordo che WINE non permette di natura di utilizzare driver Windows, a meno che non esistano sul sistema Linux appositi driver a cui interfacciarsi tramite le proprie librerie.
All'oggi WINE è molto ben supportato e fornisce una piattaforma decisamente ottima per una varietà molto estesa di software; ricordo in ogni caso che deve essere utilizzato solo qualora risulti insostituibile, non per duplicare un sistema Windows.
Un grande vantaggio è la possibilità di riportare la configurazione alle impostazioni di default semplicemente rinominando (o cancellando) la directory .wine della propria home, in maniera tale da garantire la perfetta replicabilità delle impostazioni su qualunque macchina GNU/Linux inserendo la versione definitiva della directory correttamente settata.

Diversa utilità ha invece Cygwin (porting di applicazioni GNU/Linux sotto Windows) che assume un ruolo diverso nella sua modalità su console o su interfaccia Cygwin-X.
La funzione principale di Cygwin in modalità minima è quella di permettere l'adozione di scripting per UNIX anche su sistemi Windows, sia direttamente tramite l'adozione di bash/bc/sed/awk, sia per via indiretta con l'utilizzo di interpreti installati da Cygwin che adottano impostazioni comuni per GNU/Linux (in altre parole, ad esempio Perl installato da Cygwin risponde in /usr/bin/perl e non in C:\Perl\perl.exe). Le possibilità che si aprono, anche in relazione al volume enorme di script disponibili in rete è enorme e immediatamente comprensibile agli addetti ai lavori (provare per credere).
Cygwin-X sarà invece utile ai semplici utilizzatori delle macchine, installando applicativi non presenti per Windows ma su cui potranno un giorno lavorare al termine della migrazione (applicazioni Gnome/Gtk e KDE/Qt).

Come WINE, anche Cygwin è replicabile semplicemente installando C:\cygwin in un'altra macchina e modificando il nome della home utente.
Un aspetto generale degno di nota è anche il fatto che l'utilizzo di Cygwin sia benefico anche per i tecnici, cioè aiuti a 'pensare UNIX' anche quando il personale informatico sia professionalmente legato a sistemi Windows per prassi o competenze.

Come ultime considerazioni flash vorrei dire che...
Non sono a favore in questa fase per il modello di 'corsa al server Samba'; il supporto lato client alle reti Windows è ormai un default delle maggiori distribuzioni e i client GNU/Linux saranno al momento soltanto sperimentali. Non è buono perdere tempo in cose non strettamente necessarie IMHO.
Sempre da esperienza segnalo due oggetti che mi hanno semplificato molto il lavoro: il primo è lo script 'winetricks' che installa in modo automatico parecchi applicativi Microsoft in pochi click; il secondo è una serie di programmi free (per Windows) in grado di convertire database Access in altri formati come mysql o postgree che aprono possibilità interessanti in questa fase progettuale.

Qualcuno avrà notato che non ho parlato di una questione molto dibattuta in documenti analoghi a questo, cioè la risposta alla domanda 'quale distro scegliere?'.
Sinceramente non mi sento capace di dare una risposta precisa che sia slegata dai gusti personali;
la risposta sta nella capacità dei tecnici di gestire al meglio un dato sistema e non può ridursi ad una supposizione di semplicità se poi gli addetti devono prima imparare a gestire una nuova piattaforma.
Va da sé che nell'informatica aziendale e soprattutto in un progetto del genere semplicità e velocità di realizzazione sono fattori determinanti, quindi opterei per una delle grandi distro ben supportate, con installazione e configurazione rapida e indolore, evitando il più possibile la necessità di patch e macchinosi settaggi.
Credo di esporre un'idea condivisa se dico che in azienda incompetenti e smanettoni provocano gli stessi danni.

Questa seconda fase progettuale dovrebbe essere conteggiata come tempistica a livello di settimane o di qualche mese, in relazione all'hardware/software presente e alla competenza dei tecnici (dove competenza è la conoscenza di GNU/Linux e della migrazione, non la bravura). L'uso di GNU/Linux solo dagli addetti per istruirsi, così come l'adozione di Cygwin per acquisire manualità è fondamentale per preparare il personale anche se non molto specializzato, con lo scopo di rendere quanto più indolore possibile la fase successiva.
Dal punto di vista 'visivo' il termine della fase porterà all'uso di un sistema comune, con qualche applicazione multi-piattaforma e qualche software emulato su Cygwin, piccoli problemi tipici dell'uso di applicativi diversi di facile soluzione (se i tecnici lavorano precedentemente sui nuovi software) e minimizzati in brevissimo tempo.
La modifica enorme è quella che rimane dietro le quinte e che non deve entrare per ora nella gestione corrente dell'attività aziendale, il cambiamento deve essere un dato di fatto potenziale da analizzare nelle procedure di azione ma non più nella base programmatica.

Fase applicativa

La terza fase è invece quella che implica la migrazione vera e propria, un momento piuttosto complesso perché deve essere un percorso e non una sostituzione pura e semplice.
L'inizio della programmazione finale deve avvenire solamente nel momento in cui l'attuazione delle prime due fasi hanno raggiunto quella stabilità tale da potersi dire concluse (acquisizione di competenze da parte dei tecnici e di manualità degli operatori); se il percorso è stato eseguito in maniera scrupolosa la capacità di programmare gli eventi futuri dovrebbe essere determinabile senza sforzi particolari.

Ricordo che in questo stato la migrazione non è ancora iniziata a livello pratico, oltre al fatto che non sono ancora stati analizzate le possibilità di supporto commerciale per le piattaforme libere (in parole povere: ancora di spese non si è parlato). Si partirà quindi da una simile situazione:

1. Una quota del sistema informativo hardware e software, che dovrebbe essere abbastanza ingente se il lavoro è stato svolto a dovere, risulterà essere pronta per la migrazione.
2. Probabilmente alcuni apparati hardware non risulteranno compatibili con sistemi open.
3. Probabilmente alcuni software non risulteranno compatibili con sistemi open.
4. La struttura attuale è già funzionante su sistemi proprietari di cui le licenze sono un costo già sostenuto, quindi ormai non recuperabile.

Prima di analizzare le implicazioni di questi punti vorrei far notare una profonda differenza tra questo documento e alto materiale presente in rete, che probabilmente andrà consultato dagli interessati.
La maggior parte della letteratura sulla migrazione fa riferimento al settore delle amministrazioni pubbliche, il motivo non è che in tali ambienti la pratica sia più diffusa ma semplicemente la pubblicità del cambiamento è maggiore (=presente) oltre al fatto che in azienda una migrazione diventa vantaggio competitivo che non è bene condividere più di tanto (prego di lasciare l'etica nel suo ruolo). E' evidente che un progetto simile nelle PA comporta una sostituzione semplice degli apparati per la maggiore disponibilità di risorse investite, cosa non certo replicabile nella realtà aziendale.

La difficoltà maggiore sarà quella di gestire le applicazioni non sostituibili legate necessariamente a sistemi proprietari, chiaramente non trasferibili su WINE. A rigor di logica tra applicazioni sostitutive e possibilità di emulazione i casi dovrebbero essere abbastanza rari, oltre al fatto che l'aggiornamento del programma è decisamente rapido e molte incompatibilità possono venire risolte in tempi brevi (durante la seconda fase di test WINE farà passi da gigante).
La prima cosa da vedere è se realmente l'applicazione è insostituibile, o meglio se non sia possibile una sostituzione parziale della stessa.
Da mia esperienza posso citare il caso di sviluppatori di grafica fortemente legati ad Adobe Photoshop (emulabile su WINE), che diventa necessario solamente nella gestione di file psd (in alcune caratteristiche) che possono essere trattati solo in alcune postazioni mentre il grosso del lavoro viene svolto tramite GIMP, con notevole risparmio sia dei costi di licenza che di risorse hardware; un'eventuale uso del controllo remoto su postazioni che agiscono da 'server Photoshop ne garantisce un utilizzo massimizzato nella più totale semplicità.

Il caso di hardware non compatibile deve invece avere una trattazione particolare; ricordo che non stiamo parlando di hardware a livello di case delle macchine visto che in azienda dovrebbe essere presente hardware comune ben supportato anche da GNU/Linux ma di apparati esterni legati all'uso della postazione modello.
Quella definizione di 'hardware comune' è appunto all'origine di eventuali problemi, in altre parole le stampanti degli uffici difficilmente danno problemi di supporto mentre le stampanti termiche della produzione potrebbero non trovare adeguato supporto.
Dopo aver verificato la mancanza di software adeguato è bene richiedere alle case produttrici per verificare la possibilità di utilizzo; spesso anche se non pubblicizzata la portabilità esiste perché le richieste sui grandi numeri sono molto più elevate di quanto potrebbe sembrare.

A volte viene citato il caso dell'emulazione completa di un sistema Windows sotto GNU/linux per risolvere i problemi specificati sopra. Poiché tale pratica prevede comunque l'acquisizione di una licenza del sistema operativo emulato, quindi apparirebbe totalmente inutile, è bene trattare l'argomento dando qualche spiegazione.
La virtualizzazione a determinate condizioni può essere conveniente; chiaramente qui si parla di semplice emulazione di un sistema operativo, non di virtualizzazione di server e/o soluzioni di cluod-computing che sono comunque inseribili nel contesto aziendale ma in maniera separata da qualunque concetto di migrazione all'open-source.
Il vantaggio generale dell'emulazione è legato a due fattori:

1. il vantaggio gestionale legato all'eliminazione dei problemi dati dalla compresenza di più sistemi all'interno dell'azienda. Virtualizzare tutte le macchine dove Windows è indispensabile porta automaticamente ad un livello di migrazione molto elevato in un colpo solo, con l'acquisizione di alcuni vantaggi quali la sicurezza (cioè stop al problema virus per sempre, ovvero riattivazione immediata del sistema tramite backup sulla stessa macchina).
2. l'utilizzo di software e sistemi obsoleti su macchine dell'ultima generazione, ad esempio software compatibile con versioni 9x di Windows su hardware moderno non più compatibile con un sistema di cui si possiede la licenza nel cassetto dei vecchi cd-rom.

Oggi che XP sta invecchiando piuttosto rapidamente (processo che a breve sarà spinto dalla stessa Microsoft) non è più così immediato trovare macchine compatibili a livello hardware con tale sistema, mentre il software è totalmente compatibile.
Vorrei citare il caso personale di una macchina da 4 GB di RAM e 200 GB di HD acquistata senza sistema operativo e compatibile con Vista, mentre il software utilizzato dava ancora problemi con il nuovo sistema. La macchina è stata settata con interfaccia basata su TinyWM che esegue Windows XP tramite Qemu, ovviamente non mostro una schermata di esempio perché si vedrebbe una comunissima immagine di Windows.
Da considerare anche il fatto che un sistema emulato è perfettamente clonabile copiando semplicemente l'immagine in un'altra macchina, sempre e comunque all'interno dell'EULA Microsoft (a meno di non voler fare i furbi).
Considerato il fatto che comunque una quota di licenze acquisite saranno sicuramente a disposizione dell'impresa, la valutazione sull'adozione o meno della virtualizzazione andrà fatta in collaborazione tra direzione e tecnici.

Escluse le precedenti valutazioni il processo pratico di migrazione deve essere basato sui meccanismi di implementazione e sostituzione.
Ogni eventuale nuova postazione dovrà essere montata con sistema open e relativi software, nel caso si necessiti di applicazioni fortemente legate a sistemi proprietari si provvederà alla conversione di una postazione diversa in maniera da disporre di una licenza libera (eventualmente tramite sistema virtuale); dato l'uso di risorse minore sui sistemi liberi sarebbe bene utilizzare tali sistemi sulla macchina meno performante.
Un discorso analogo può essere fatto per i dispositivi hardware nell'eventualità che non esista supporto per GNU/Linux, anche in considerazione del fatto che non esiste rapporto tra costo dell'hardware e compatibilità ad esclusione di apparati di scarsissimo costo e qualità che si spera non entrino nell'impresa, a volte è sufficiente acquistare un modello diverso della stessa marca per avere pieno supporto.

Il passaggio alla nuova piattaforma deve avvenire in linea generale in base alle normali sostituzioni dettate dall'obsolescenza programmata dell'impiantistica in modo da non gravare inutilmente sui costi aziendali, che si suppongono sempre essere meglio dedicati all'attività caratteristica d'impresa.

Oltre alla gestione del costo esiste un secondo motivo di tipo tecnico per gestire la migrazione attraverso la sostituzione periodica dell'hardware, cioè la possibilità di sfruttare gli sviluppi futuri di applicativi per la gestione dei device oggi incompatibili; l'hardware di vecchio tipo che vengono tolti dalla produzione attuale e tenuti a deposito diverranno a breve supportati (non è matematico ma è vero al 90%) e continueranno quindi a svolgere la propria funzione sostitutiva in caso di eventuali guasti.

Conclusione

Mi pare evidente che un articolo di poche pagine non può essere esaustivo dell'argomento, quindi eventuali considerazioni vanno fatte in ambito generale.
Un dato di fatto che spero sia deducibile dalla trattazione è che nel 2011 l'utilizzo di software libero nell'attività aziendale è cosa generalmente sia fattibile che conveniente; i dati di diffusione di 'aziende open' sono attualmente molto ridotti ma in fortissima espansione quanto lo è la diffusione degli utenti di sistemi open.
Distribuzioni sempre più user-friendly e un supporto hardware ormai capillare fanno diventare appetibili progetti di migrazione anche per realtà piccole e medie che spesso si considerano per principio incapaci di svolgere un passo simile, cosa per altro più diffusa nel nostro paese che altrove.

Per quanto l'ambito etico sia stato escluso, la reale causa del disinteresse per i sistemi liberi è da ricercare nella diffusione del software pirata che raggiunge nel nostro paese livelli inaccettabili, basta pensare alla sua diffusione all'interno delle stesse strutture pubbliche.
Sta di fatto che un sistema proprietario 'statico' e blindato nasce e si sviluppa per fare moderatamente bene determinate funzione, non è plasmabile in modo da ottenere eccellenze in un dato ambito come invece si richiede alle piattaforme usate in azienda; da qui si originano i tipici problemi del driver che non funziona bene o dell'aggiornamento che causa problemi.
Se la penetrazione dei sistemi open in Italia è molto inferiore a quella di altri grandi paesi industrializzati (la distanza con i paesi emergenti è ancora maggiore), questo rispecchia la situazione per cui il gap tecnologico delle imprese tra noi e il resto del mondo si sta ampliando velocemente; tecnologia vuol dire riduzione dei costi e riduzione dei costi vuol dire maggior competitività, e questa non è Informatica ma sono Affari.

Si è deciso di non soffermarsi su aspetti amministrativi-legali (ad esempio i finanziamenti agevolati per progetti open) o strettamente tecnici (ad esempio la gestione di ERP open-source) per puntare sul fattore organizzativo e progettuale.
È importante sottolineare nuovamente come sia la programmazione la vera fonte del successo di simili progetti, quella cooperazione tra direzione e ramo tecnico che diventa motore del percorso e causa del risultato finale.

Floatman

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 del prossimo numero (il num. 13) è prevista alla data di:

Venerdì 25 Marzo 2011

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