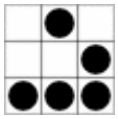
Come diventare un hacker

Eric Steven Raymond <esr@thyrsus.com>

Versione italiana del 3 Novembre 2006 presa da
http://www.saltatempo.org/hacker/hacker_howto_it.php

Adattamento di Marco Costanzo in pdf per http://www.espertointernet.com

Indice dei contenuti



Perché questo documento?

Come editore dello Jargon File e autore di altri documenti simili, spesso ricevo delle richieste tramite *mail* da parte di novellini della rete che mi chiedono, sostanzialmente, "come posso diventare un grande hacker?". Nel 1996 mi accorsi che sembrava non esserci alcuna FAQ o documento *web* che rispondesse a questa domanda fondamentale, così iniziai a scrivere questo. Adesso, molti *hacker* lo considerano un documento definitivo, e quindi suppongo che lo sia. Non ritengo di essere la principale autorità in materia, e se non vi piace ciò che leggete in questa sede scrivete qualcosa voi stessi.

Se state leggendo un'*istantanea* non in linea di questo documento, la versione corrente risiede all'indirizzo http://catb.org/~esr/faqs/hacker-howto.html.

Nota: c'è una lista di Frequently Asked Questions alla fine di questo documento. Per favore, leggetela - due volte magari - prima di scrivermi *mail* con domande riguardanti i contenuti di questo documento.

Numerose traduzioni di questo documento sono disponibili: Bulgaro, Catalano, Cinese (Semplificato), Cinese, Danese, Olandese, Persiano, Finlandese, Tedesco, Ebraico, Ungherese, Italiano, Giapponese, Polacco, Portoghese (Europeo), Spagnolo, Turco, e Svedese. Inoltre, poiché questo documento cambia periodicamente, tutte potrebbero essere non aggiornate a vari livelli.

Il diagramma dei cinque punti nei nove quadrati che decora questo documento è detto *glider*. E' un semplice disegno con delle proprietà sorprendenti in una simulazione matematica conosciuta come Life che ha affascinato gli *hacker* per molti anni. Penso che sia una buona rappresentazione del modo di essere degli

hacker - astratta, un po' misteriosa al primo impatto, ma allo stesso tempo una porta per un mondo vasto con una logica propria ed intricata. Per saperne di più, glider.

Cos'è un Hacker?

Lo Jargon File contiene un serie di definizioni del termine "hacker", la maggior parte delle quali hanno a che fare con abilità tecniche e con il piacere di risolvere problemi e superare i limiti. Se volete diventare un *hacker*, credo, soltanto due di queste sono rilevanti.

C'è una comunità , una *cultura condivisa* (shared culture), di esperti programmatori e maghi delle reti la cui storia risale a decadi addietro, ai primi *time-sharing minicomputer* ed ai primi esperimenti dell'ARPAnet. I membri di questa cultura hanno dato origine al termine *hacker*. Gli *hacker* hanno costruito Internet. Gli *hacker* hanno reso il sitema operativo UNIX quello che è attualmente. Gli *hacker* frequentano la Usenet. Gli *hacker* hanno reso funzionante il World Wide Web. Se siete parte di questa cultura, se avete contribuito alla sua crescita e altre persone conivolte sanno chi siete e vi chiamano *hacker*, allora siete un *hacker*.

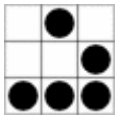
Ma la *forma mentis* di un *hacker* non è confinata alla cultura del *software*. Ci sono persone che applicano l'attitudine dell'*hacker* in altri campi, come l'elettronica e la musica - attualmente potete trovarli ai livelli più elevati della scienza o dell'arte. Gli hacker dei *software* riconoscono questi spiriti affini ovunque e possono chiamare anch'essi *hacker* - ed alcuni affermano che la natura di un *hacker* è pressoché indipendente dal particolare *media* su cui l'*hacker* lavora. Ma nel resto del documento concentreremo l'attenzione sulle capacità e le attitudini degli *hacker* dei software, e sulle tradizioni della *cultura condivisa* che ha dato origine al termine *hacker*.

C'è poi un altro gruppo di persone che si autodefiniscono *hacker* ma non lo sono. Persone (principalmente adolescenti di sesso maschile) che provano un gusto matto nel penetrare nei computer e a intasare il sistema telefonico. I veri *hacker* chiamano queste persone *cracker* e non hanno nulla a che vedere con loro. I veri

hacker pensano principalmente che i *cracker* siano pigri, irresponsabili e non molto brillanti, ed obiettano che essere in grado di compromettere la sicurezza di un sistema non rende *hacker* più di quanto manomettere una macchina renda ingegneri automobilistici. Sfortunatamente, molti giornalisti e scrittori sono stati fuorviati e utilizzano spesso il termine *hacker* per indicare il realtà i *cracker*, cosa che irrita terribilmente i veri *hacker*.

La differenza fondamentale è questa: gli *hacker* costruiscono le cose, i *cracker* le rompono.

Se volete diventare *hacker*, continuate a leggere. Se volete diventare *cracker*, andate a leggere le *newsgroup* su alt.2600 e preparatevi a restare dalle 5 alle 10 in quella bolgia prima di accorgervi che non siete così cazzuti e fichi come credete di essere. E questo è tutto quel che dirò sui *cracker*.



L'attitudine degli hacker

1. Il mondo è pieno di problemi affascinanti che attendono di essere risolti
2. Nessun problema dovrebbe mai essere risolto due volte
3. Noia e monotonia sono il male
4. La libertà è fondamentale
5. L'attitudine non sostituisce la competenza

Gli *hacker* risolvono i problemi e costruiscono le cose, credono nella libertà e nell'aiuto volontario reciproco. Per essere accettati come *hacker*, dovete comportarvi come se aveste voi stessi questo tipo di attitudini. E per comportarvi come se aveste queste attitudini dovete realmente credere in quelle particolari attitudini.

Ma se pensate che coltivare le attitudini da *hacker* sia solo un modo per guadagnarsi l'accettazione in quella cultura sbagliate di grosso. Diventare il tipo di persona che crede in queste cose è importante per voi stessi - per aiutarvi ad

imparare e ad essere motivati a farlo. Come in tutte le arti creative, il modo migliore per diventare un maestro è imitare il modo di pensare dei maestri - non soltanto intellettualmente ma soprattutto emotivamente.

Come spiega questo poema Zen moderno:

> Per seguire il sentiero:
> guardare al maestro,
> seguire il maestro,
> camminare con il maestro,
> *guardare attraverso* il maestro,
> diventare il maestro.

Quindi, se volete essere un *hacker*, ripetete le cose che seguono finché non crederete sul serio ad esse:

1. Il mondo è pieno di problemi affascinanti che attendono di essere risolti.

Essere un *hacker* è molto divertente, ma è un tipo di divertimento che richiede molti sforzi e gli sforzi richiedono motivazione. Atleti di successo traggono motivazione da una sorta di piacere fisico che provano nel rendere i loro corpi performanti, nel lasciarsi alle spalle i propri limiti fisici. Analogamente, per essere un *hacker* dovrete provare un brivido di fondo nel risolvere i problemi, affinare le vostre capacità ed esercitare la vostra intelligenza.

Se non siete il tipo di persona che prova queste sensazioni in modo naturale, avrete bisogno di diventarlo prima di poterlo esercitare da *hacker*. In caso contrario vi accorgerete che la vostra energia nell'hacking verrà indebolita da distrazioni come il sesso, il denaro e l'approvazione sociale.

(Inoltre dovrete sviluppare una sorta di fede nelle vostre capacità di apprendimento. Il pensiero che sebbene potreste non conoscere tutto ciò che occorre per risolvere un determinato tipo di problema, dopo averne risolto soltanto un pezzo ed aver appreso da ciò, apprenderete il necessario per risolvere il pezzo successivo e così via, finché non avrete finito.)

2. Nessun problema dovrebbe mai essere risolto due volte.

Le menti creative sono una risorsa preziosa e limitata. Non dovrebbero essere sciupate nel *reinventare la ruota* quando ci sono abbastanza problemi nuovi che aspettano la fuori.

Essere un *hacker*, vuol dire rendersi conto che il tempo impiegato dagli *hacker* per pensare è prezioso - a tal punto che è quasi un dovere morale condividere informazioni, risolvere i problemi e mettere a disposizione le soluzioni in modo tale che altri *hacker* possano risolvere nuovi problemi anziché dover combattere perpetuamente con i vecchi.

(Non dovete pensare di essere obbligati a dar via *tutto* il prodotto della vostra creatività , sebbene gli *hacker* che lo fanno siano i più rispettati dagli altri *hacker*. E' comunque in linea con i valori degli *hacker* vendere buona parte del proprio lavoro per nutrirsi e pagare l'acquisto dei computer. E' bene utilizzare l'abilità nell'hacking per sostenere una famiglia o diventare ricchi, purché non dimentichiate la fedeltà alla vostra arte ed ai vostri compagni *hacker* mentre lo fate.)

3. Noia e monotonia sono il male.

Gli *hacker* (e le persone creative in generale) non dovrebbero mai annoiarsi o fossilizzarsi in lavori stupidi e ripetitivi, perché quando questo accade vuol dire che non stanno facendo quel che potrebbero invece fare - risolvere nuovi problemi. Questo dispendio di energie non giova a nessuno. Quindi, attualmente, noia e monotonia non sono soltanto spiacevoli ma anche dannosi.

Per essere un *hacker* dovete credere il più possibile nella possibilità di liberarvi delle parti noiose rendendole automatiche non solo per voi stessi ma per chiunque altro (in particolare gli altri *hacker*).

(C'è una sola apparente eccezione a questo. Gli *hacker* a volte fanno delle *cose* come esercizi di pulizia mentale, o per acquisire nuove capacità o esperienze che non possono essere acquisite altrimenti, cose che possono sembrare ripetitive o noiose ad un osservatore estraneo. Ma questo è per scelta - nessun essere

dotato di intelletto potrà mai essere costretto a situazioni che lo annoiano.

4. La libertà è un bene.

Gli *hacker* sono per natura anti-autoritari. Nessuno ha il diritto di impartire degli ordini per impedirvi di risolvere un qualsiasi problema dal quale siete affascinati e, dato il modo in cui le menti autoritarie operano, siate certi che troverеranno sempre qualche ragione terribilmente stupida per farlo. Per questo l'autoritarismo dovrà essere combattuto ovunque vi capiterà di incotrarlo, per timore che soffochi voi ed altri *hacker*.

(Questa non è la stessa cosa che combattere ogni tipo di autorità . I bambini hanno bisogno di essere guidati ed i criminali di essere repressi. Un *hacker* può essere daccordo nell'accettare un qualche tipo di autorità al fine di ottenere qualcosa che desidera più del tempo perso nel seguire gli ordini. Ma questo è un limitato e consapevole compromesso; il tipo di resa personale che le persone autoritarie desiderano non è in offerta.)

Le persone autoritarie traggono profitto dalla censura e dal segreto e distruggono la cooperazione volontaria e la condivisione delle informazioni - gradiscono solo il genere di cooperazione che possono controllare. Quindi per essere un *hacker* dovrete sviluppare una ostilità istintiva verso la censura, verso il segreto e nei confronti dell'uso della forza o dell'inganno per obbligare adulti responsabili. E dovrete essere disposti ad operare in questa convinzione.

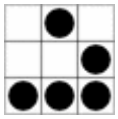
5. L'attitudine non sostituisce la compentenza.

Per essere un *hacker* dovrete sviluppare alcune di queste attitudini. Ma fare propria un'attitudine soltanto non vi renderà un *hacker* non più di quanto vi renderà un campione di atletica o una *rock star*. Per diventare un *hacker* occorre intelligenza, pratica, dedizione e duro lavoro.

Quindi, dovrete imparare diffidare dell'attitudine e rispettare invece la competenza di ogni tipo. Gli *hacker* non lasceranno che le *domande imbarazzanti* rubino loro il tempo, ma adoreranno la competenza - specialmente la competenza nell'hacking,

ma anche la competenza in genere. Competenza nel voler acquisire capacità che pochi possono insegnare e nell'affinare capacità che richiedono acutezza mentale, abilità e concetrazione.

Se perseguirete la competenza, proverete piacere nel sentirla crescere in voi stessi - il duro lavoro e la dedizione diverranno una sorta di gioco intenso piuttosto che monotonia. Questa attitudine è vitale per diventare un *hacker*.



Capacità base nell'hacking

1. Imparare a programmare.
2. Prendere uno dei sistemi Unix open-source ed imparare ad usarlo e farlo girare.
3. Imparare come usare il World Wide Web e a scrivere in HTML.
4. Se non conoscete un inglese funzionale imparatelo.

L'attitudine dell'*hacker* è vitale ma le capacità lo sono di più. L'attitudine non sostituisce la competenza, e c'è un insieme di capacità basilari che dovrete avere prima che un altro *hacker* si sogni di chiamarvi nello stesso modo.

Questo insieme cambia lentamente nel corso del tempo man mano che la tecnologia crea nuove *capacità* e rende le precedenti obsolete. Ad esempio, lo è stata la programmazione in linguaggio macchina mentre recentemente lo è stato l'HTML. Attualmente include le seguenti:

1. Imparare a programmare.

Naturalmente questa è la capacità fondamentale per l'hacking. Se non conoscete nessun linguaggio di programmazione vi suggerisco di iniziare con Python. E' progettato in modo chiaro, è ben documentato ed è relativamente agevole per i principianti. Nonostante sia un buon linguaggio per iniziare, non è un giocattolo; è molto potente e flessibile e ben adatto per grandi progetti. Ho scritto una

valutazione più dettagliata su Python e ottimi tutorial sono disponibili nel sito web di Python.

Anche il Java è un buon linguaggio per imparare a programmare. E' più difficile di Python, ma produce un codice più veloce. Penso che possa essere un eccellente secondo linguaggio. Sfortunatamente, l'implementazione di riferimento della Sun è ancora proprietaria. Questo non è tanto un problema con linguaggio Java in se, visto che interpreti *open-source* Java di alta qualità sono già disponibili; il problema sono le *librerie di classe* che vanno di pari passo con il linguaggio. Infatti, le *librerie di classe open-source* sono, per forza di cose, in ritardo rispetto a quelle della Sun. Quindi, se scegliete di imparare il Java, fatelo con una delle implementazioni *open-source* pittosto che diventare dipendenti del codice proprietario della Sun.

Ma fate attenzione al fatto che non raggiungerete le capacità di un *hacker*, o più semplicemente di un programmatore, se imparerete soltanto uno o due linguaggi - avrete bisogno di imparare a pensare ai problemi di programmazione in senso generale, in modo indipendente dal particolare linguaggio. Per essere realmente un *hacker* dovrete arrivare al punto di poter apprendere un nuovo linguaggio in pochi giorni, mettendo in relazione quello che c'è nel manuale di riferimento con quanto già conoscete. Questo vuol dire che dovreste imparare svariati linguaggi differenti.

Se entrerete nella programmazione seriamente dovrete imparare il C, il linguaggio centrale di Unix. Il C++ è strettamente collegato al C e se conoscete uno dei due imparare l'altro non sarà difficile. Comunque non è un linguaggio da provare ad imparare come primo e, allo stato delle cose, più eviterete di programmare in C e più sarete produttivi.

Il C è molto efficente e risparmia molto le risorse della vostra macchina ma sfortunatamente il C raggiunge questa efficenza grazie ad una gestione *manuale* di basso livello delle risorse (come la memoria). Tutto questo codice di basso livello è complesso e incline ai *bug*, e succhierà grosse quantità del vostro tempo per il *debugging*. Con le potenti macchine di oggi questo è un pessimo scambio - è preferibile usare linguaggi che usano il tempo macchina *meno* efficentemente

ma il vostro tempo molto *più* efficentemente. Quindi, Python.

Altri linguaggi di particolare importanza per gli *hacker* includono Perl e LISP. Il Perl vale la pena di impararlo per ragioni pratiche; è usato moltissimo nelle pagine *web* dinamiche e per l'amministrazione di sistema, quindi anche se non scriverete mai in Perl dovreste almeno imparare a leggerlo. Molte persone usano il Perl nel modo in cui io vi ho suggerito dovreste usare Python, per evitare la programmazione in C in lavori che non richiedono l'efficenza macchina che quest'ultimo offre. Dovrete essere in grado di capire il loro codice.

Il LISP è importante per una ragione differente: la profonda ed illuminante esperienza che avrete quando lo avrete appreso. Questa esperienza farà di voi un programmatore migliore per il resto dei vostri giorni, anche se attualmente non usate molto LISP.
(Potete acquisire una esperienza base con il LIPS semplicemente scrivendo e modificando i modi di editing per Emacs).

Attualmente è meglio imparare tutti e cinque i linguaggi, Python, C/C++, Java, Perl, e LISP. Oltre ad essere i linguaggi più importanti per l'hacking rappresentano approcci molto differenti alla programmazione e ognuno di essi vi educherà in modo diverso e considerevole.

In questa sede non posso dare istruzioni complete sul come imparare a programmare, è un problema complesso. Ma posso suggerirvi quei libri e quei corsi che certamente *non* lo faranno (molti, e forse la maggior parte dei migliori *hacker* sono autodidatti). Potete imparare le caratteristiche di un linguaggio - pezzetti di conoscenze - dai libri, ma la mentalità che fa vivere questa conoscenza nelle capacità di ognuno può essere appresa soltanto attraverso la pratica e l'apprendistato. Ciò che renderà possibile tutto ciò è (a) *leggere codice*, (b) *scrivere codice*.

Imparare a programmare è come imparare a scrivere bene ed in modo naturale. Il modo migliore di farlo è leggere qualcosa scritto da un *maestro* della forma, scrivere alcune cose da soli, leggere ancora qualcosa in più, scrivere ancora, leggere dell'altro, scrivere dell'altro...e ripetere tutto questo finché il vostro modo di

scrivere non inizia a sviluppare il tipo di efficacia ed economia che apprezzate ed ammirate nei vostri modelli.

Trovare del buon codice da leggere era difficile un tempo, per i *piccoli* hacker c'erano pochi grossi programmi di cui era disponibile il sorgente per la lettura e con cui armeggiare. Tutto ciò è cambiato in modo drammatico; il software *open-source*, gli srumenti di programmazione e i sistemi operativi (tutto costruito dagli *hacker*) sono ora ampiamente disponibili. Il che mi porta ad introdurre il prossimo argomento.

2. Prendere uno dei sistemi Unix open-source ed imparare ad usarlo e farlo girare.

Presumo che abbiate un personal computer o che possiate accedere ad un pc (i ragazzi di oggi hanno tutto così facilmente :-)). L'unico e più importante passo che un novellino può muovere verso l'acquisizione delle capacità di un *hacker* è reperire una copia di Linux o di uno degli Unix *liberi* della BSD, installarlo su una macchina personale e farlo *girare*.

E' vero, ci sono altri sistemi operativi nel mondo oltre a Unix, ma sono distribuiti in forma binaria - non potete leggerne il codice, e non potete modificarlo. Provare ad imparare l'hacking su una macchina Windows o MacOS oppure ogni altro sistema *non-open* è come cercare di impare a ballare indossando un'armatura.

Sotto Mac OS X è possibile, ma soltanto una parte del sistema è *open-source* - presumibilmente vi scontrerete contro parecchi muri, e dovrete fare attenzione a non sviluppare cattive abitudini che dipendono dal codice proprietario della Apple. Se vi getterete a capofitto su Unix potrete imparare molte cose utili.

Unix è il sistema operativo di Internet. Mentre potete imparare ad utilzzare Internet senza conoscere Unix, non potete essere un Internet *hacker* senza capire Unix. Per questa ragione, la cultura degli *hacker* oggi è molto incentrata su Unix. (Questo non è sempre stato vero, ed alcuni *hacker* dei tempi passati non ne sono felici, ma la simbiosi tra Unix ed Internet è diventata forte a tal punto che nemmeno i muscoli di Microsoft sembrano in grado di scalfirla seriamente.)

Perciò, procuratevi un sistema Unix - personalmente mi piace Linux ma ce ne sono altri (e potete anche far girare Linux e Windows sulla stessa macchina). Imparatelo. Fatelo girare. Armeggiateci. Leggetene il codice. Modificatene il codice. Avrete a disposizione strumenti di programmazione (incluso C, LISP, Python e Perl) che nessun sistema Microsoft potrà mai sognarsi di poter ospitare, vi divertirete, e acquisirete molta più conoscenza di quanto pensiate finché non la guarderete dall'alto da maestri dell'hacking.

Per saperne di più sull'apprendimento di Unix visitate The Loginataka. Vorrete dare un'occhiata anche a The Art Of Unix Programming.

Per mettere le mani su un sistema Linux visitate il sito Linux Online! ; potete scaricarlo da li o (idea migliore) trovare un gruppo di utenti Linux vicino a voi che possa aiutarvi nell'installazione. Dal punto di vista di un nuovo utente, tutte le distribuzioni Linux sono pressoché equivalenti.

Potrete trovare aiuto e risorse per sistemi Unix BSD all'indirizzo www.bsd.org.

Ho scritto un manuale elementare sulle basi di Unix e di Internet.

(Nota: raccomando di non installare Linux o BSD da soli se siete dei principianti. Per Linux, cercate un gruppo di utenti locali e chiedete aiuto.)

3. Imparare a come usare il World Wide Web e a scrivere HTML.

Molte delle cose che la cultura *hacker* ha costruito lavorano dietro le quinte, aiutano a mandare avanti fabbriche, uffici e università senza nessun impatto evidente sullo stile di vita dei *non hacker*. Il Web è la sola grande eccezione, questo immenso e splendente giocattolo degli *hacker* che anche i *politici* ammettono stia cambiando il mondo. Per questa sola ragione (e molte altre, altrettanto buone) avrete bisogno di imparare a come lavorare sul Web.

Questo non significa solo imparare a utilizzare un *browser* (tutti possono farlo), ma imparare a scrivere codice HTML, il linguaggio *markup* del Web. Se non sapete come programmare, scrivere in HTML vi insegnerà alcune abitudini mentali che vi aiuteranno ad imparare. Costruite un *home page*. Orientatevi verso

l'XHTML, che è un linguaggio più pulito del classico HTML. (Ci sono buoni tutorial sul Web; questo è uno.)

Ma avere un'*home page* non è ancora abbastanza. Il Web è pieno di *home page*. La maggior parte delle quali sono senza scopo, poltiglia a contenuto zero - vera poltiglia con un look da sballo, penserete, ma sempre poltiglia (per saperne di più The HTML Hell Page).

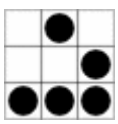
Per essere degna di nota, la vostra pagina deve avere dei contenuti - deve essre interessante ed utile per agli altri *hacker*. E questo ci porta al prossimo tema...

4. Se non conoscete un inglese funzionale, imparatelo..

Da Americano di lingua madre Inglese, ero riluttante nel dare un suggerimento del genere, per timore che fosse interpretato come una sorta di imperialismo culturale. Ma molti altri madrelingua (non inglesi) mi hanno esortato a precisare che l'inglese è il linguaggio operativo della cultura *hacker* e di Internet, e che avrete bisogno di farlo *funzionare* a dovere nell'ambito della comunità degli *hacker*.

Intorno al 1991 appresi che molti hacker che conoscono l'inglese solo come seconda lingua lo usano nelle discussioni tecniche anche quando condividono la stessa lingua di nascita, dato che l'inglese ha un vocabolario tecnico più ricco di qualsiasi altra lingua e che quindi è lo strumento migliore per il lavoro. Per ragioni simili, le traduzioni dei libri tecnici scritte in inglese sono spesso insoddisfacenti (quando vengono fatte).

Linus Torvalds, un finlandese, commenta i suoi codici in inglese (apparentemente non ha mai avuto bisogno di fare altrimenti). La sua fluenza nel parlare inglese è stato un fattore importante nella sua capacità di reclutare una comunità mondiale di sviluppatori per Linux. E' un esempio da seguire.



Status nella *hacker culture*

Come molte culture non basate sull'economia del denaro, lo status nel regno degli *hacker* si basa sulla reputazione. State cercando di risolvere problemi interessanti, ma quanto siano interessanti le vostre soluzioni e se siano veramente buone è qualcosa che solo i vostri pari o superiori, tecnicamente, sono in grado di giudicare.

Di conseguenza, quando giocherete agli *hacker*, imparerete ad acquisire punti in base a ciò che gli altri *hacker* pensano delle vostre capacità (questo è il motivo per cui non sarete veramente un *hacker* finché altri *hacker* non vi ci chiameranno). Tutto ciò è nascosto dall'immagine dell'hacking come lavoro solitario nonché da un taboo della cultura *hacker* (che va lentamente scemando ma è ancora molto potente) contro l'ammissione che l'ego o il riconoscimento esterno non siano coinvolte nelle motivazioni personali.

Specificamente, il regno degli *hacker* è quello che gli antropologi chiamano *gift culture.* Si guadagna lo *status* e la reputazione in essa, né dominando altre persone né essendo bellissimi, né possedendo cose che altre persone vogliono, ma piuttosto mettendole a disposizione, *regalandole*. Per la precisione, dando via il vostro tempo, la vostra creatività , e i frutti delle vostre capacità .

Ci sono principalmente cinque tipi di cose che potete fare per essere rispettati dagli *hacker*:

1. Scrivere software open-source

La prima (quella centrale e più tradizionale) è scrivere programmi che altri *hacker* pensano siano piacevoli o utili, e mettere a disposizione i sorgenti dei programmi all'intera cultura *hacker* affinché li possa utilizzare.

(Di solito chiamiamo questi lavori "software libero", ma questo ha confuso molte persone che non erano sicure di cosa esattamente significasse "libero". Molti di noi, almeno 5:1 secondo le analisi dei contenuti *web*, adesso preferiscono il termine "open-source".)

Nel regno degli hacker i più riveriti semidei sono persone che hanno scritto e messo a disposizione della comunità , grandi e formidabili programmi di bisogno diffuso, che adesso tutti quanti usano.

2. Aiutare nei test e nel debug del software open-source

C'è bisogno di chi sostenga e aiuti nel *debug* del software *open-source*. In questo mondo imperfetto, inevitabilmente impiegheremo la maggior parte del nostro tempo per la fase di *debug*. Questo è il motivo per cui ogni autore di software *open-source* vi dirà che i buoni beta-tester (chi sa come descrivere i sintomi chiaramente, localizzare i problemi, tollerare bachi in una release frettolosa, e disposto ad applicare semplici routine diagnostiche) valgono tanto oro quanto pesano. Uno di loro può fare la differenza tra una fase di *debug* che è un lungo incubo che si protrae e una che è soltanto una *salutare seccatura*.

Se siete dei principianti, provate a trovare un programma in corso di sviluppo per cui provate interesse e siate un buon beta tester. C'è una progressione naturale dall'aiutare nei test dei programmi e nel *debug* al modificarli. Imparerete molto in questo modo, e genererete del buon *karma* con le persone che vi aiuteranno più in la.

3. Pubblicare informazioni utili

Un'altra cosa buona è raccogliere e filtrare informazioni utili ed interessanti nelle pagine web o nei documenti come nella lista delle Domande Frequenti (Frequently Asked Questions ovvero FAQ) e renderle disponibili in generale.

I manutentori delle principali FAQs ottengono quasi lo stesso rispetto degli autori di software open-source.

4. Aiutare a mantenere le infrastrutture funzionanti

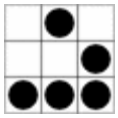
La cultura degli *hacker* (e lo sviluppo ingegneristico di Internet, in questo senso) è alimentata dai volontari.
Per farla progredire c'è bisogno di molto lavoro che non è per niente affascinante - amministrare le *mailing list*, moderare le *newsgroup*, mantenere grandi siti di archivi software, sviluppare RFCs ed altri standard tecnici.

Le persone che fanno queste cose nel modo migliore guadagnano molta stima, perché tutti sanno che questo genere di lavori sono grosse perdite di tempo e che non è divertente come giocare con il codice. Farle mostra dedizione.

5. Servire la cultura hacker

Infine, potete diffondere la cultura hacker (scrivendo, ad esempio, un'accurato manuale su come diventare un *hacker* :-)). Questo non è qualcosa che sarete in grado di fare finché non sarete in circolazione da un po' di tempo e sarete sufficentemente conosciuti per uno dei punti precedenti.

La cultura degli *hacker* non ha grandi leader, ma ha eroi culturali, antenati tribali e storici e oratori. Quando sarete stati in trincea abbastanza a lungo, potrete crescere in una di queste categorie. Attenzione: gli *hacker* diffidano degli ego appariscenti, quindi avere una fama del genere è pericoloso. Piuttosto che combatterla, dovrete scegliere la vostra strada senza aspettare che vi *cada dal cielo*, quindi siate modesti e benigni con il vostro *status*.



La relazione Hacker/Nerd

Contrariamente ad un mito popolare, non dovete essere un *nerd* (reietto) per essere un *hacker*. Aiuta, comunque, e molti hacker sono infatti dei nerd. Essere un reietto vi aiuterà a restare concentrati sulle cose veramente importanti, come pensare e fare dell'hacking.

Per questa ragione molti *hacker* hanno adottato l'etichetta "nerd" e usano anche il termine "geco" come un distintivo d'orgoglio - è un modo di dichiarare la loro indipendenza dalle normali aspettative sociali. Visitare The Geek Page per una discussione esaustiva.

Se riuscite a concentrarvi abbastanza sull'hacking al punto di essere utili e avere ancora una vita è un bene. Questo è molto più facile oggi di quanto non lo fosse negli anni 70, quando io ero un principiante; adesso la cultura dominante è molto più amichevole con i techno nerd. C'è un numero sempre maggiore di persone che si rendono conto che gli *hacker* sono spesso degli amanti di alta qualità e materiale da sposare.

Se siete attratti dall'hacking perché non avete una vita, va bene - almeno non avrete problemi nel concentrarvi. Forse ne avrete una più in la.

Spunti per lo stile

Ancora, per essere un *hacker,* dovrete entrare nel modo di pensare degli *hacker*. Ci sono altre cose che potete fare quando non siete di fronte ad un computer e che sembrano aiutare a diventare *hacker*. Queste non sostituiscono l'hacking (niente lo sostituisce) ma molti *hacker* le fanno, e pensano che siano in qualche modo connesse all'essenza dell'hacking.

- Imparare ha scrivere bene nella vostra lingua. Sebbene sia uno stereotipo comune che i programmatori non sappiano scrivere, un numero sorprendente di *hacker* (inclusi tutti i più ineccepibili che conosco) sono abili scrittori.

- Leggete la fantascienza. Andate ai convegni di fantascienza (un buon modo per incontrare gli *hacker* ed i *proto-hacker*).

- Imparate una forma di arte marziale. Il livello di disciplina mentale richiesta per le arti marziali sembra essere molto simile a quello degli *hacker*. Le forme più popolari tra gli *hacker* sono, per la maggiore, arti asiatiche a *mani nude* come il Tae Kwon Do, Karate, Wing Chun, Aikido, o Jiu Jitsu. L'arte della scherma occidentale e della spada asiatica hanno un grande seguito.

Nei luoghi dove è legale, sparare con la pistola è cresciuto in popolarità nei tardi anni Novanta. Le arti marziali "più *hacke*r" sono quelle per cui vengono enfatizzate le discipline mentali, il rilassamento e il controllo piuttosto che la forza, l'atleticità e la prestanza fisica.

- Studiare una disciplina attuale di meditazione. La perenne favorita tra gli *hacker* è lo Zen (è possibile beneficiare dallo Zen senza professare una religione o deprecare la vostra attuale). Altri discipline possono funzionare, ma fate attenzione a sceglierne una che non vi imponga di credere in cose folli.

- Sviluppare un orecchio musicale analitico. Imparare ad apprezzare particolari tipi di musica. Imparare a suonare uno strumento musicale o a cantare.

- Sviluppare apprezzamento per i giochi di parole e gli indovinelli.

Se già fate molte di queste cose, verosimilmente avete già l'indole dell'*hacker*. Perchè queste cose in particolare non è del tutto chiaro, ma sono connesse con un mix di capacità legate all'emisfero sinistro e destro che sembrano essere importanti; gli *hacker* devono essere in grado di ragionare in modo logico e di andare oltre l'apparenza logica di un problema in esame.

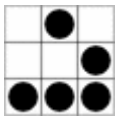
Lavorate intensamente come giocate e giocate intensamente come lavorate. Per i veri *hacker* i limiti tra "gioco", "lavoro", "scienza" e "arte" tendono a scomparire, o a fondersi in alti livelli di gioco creativo. Inoltre non siate soddisfatti di una ristretta gamma di capacità . Sebbene molti *hacker* si descrivano come programmatori, verosimilmente sembrano essere più che competenti anche in altre branche correlate - amministrazione di sistema, *web design*, PC hardware *troubleshooting* sono le più comuni. Un *hacker* che è un amministratore di sistema, d'altro canto, probabilmente sarà abile nella programmazione degli *script* e nel *web design*. Gli *hacker* non fanno le cose a metà e se si gettano in un impresa, tendono poi ad eccellere in essa.

Infine, un paio di cose da *non* fare.

- Non scegliere uno stupido nome altisonante come *user ID* o *screen name*.
- Non mettete a ferro e fuoco la Usenet (o qualsiasi altro luogo).
- Non definitevi "cyberpunk", e non perdete il vstro tempo con chiunque lo faccia.
- Non scrivete lettere o mail piene di errori grammaticali e di ortografia.

La sola reputazione che vi farete facendo una di queste cose sarà quella dello sciocco. Gli *hacker* hanno una lunga memoria e potrebbero volerci anni per cancellare i primi grossolani errori ed essere accettati.

Il problema con gli *screen name* merita qualche chiarimento. Nascondere la vostra identità è un giovanile e stupido comportamento dei *cracker*, *warez d00dz*, e altre basse forme di vita. Gli *hacker* non fanno tutto ciò; sono orgogliosi di ciò che fanno e vogliono che sia associato ai loro *nomi reali*. Quindi, se avete uno pseudonimo abbandonatelo, poiché nella cultura degli *hacker*, vi segnerà sempre come un perdente.



Altre risorse

Paul Graham ha scritto un saggio chiamato Great Hackers (*Grandi hacker*) nel quale parla molto di fede.

Peter Seebach mantiene un eccellente Hacker FAQ per i manager che non sanno come comportarsi con gli hacker.

C'è un documento chiamato How To Be A Programmer (*Come essere un programmatore*) che è un eccellente complemento al presente. Contiene non solo apprezzabili suggerimenti sulla programmazione e sulle capacità , ma anche come convivere in un gruppo di programmazione.

Io stesso ho scritto *A Brief History Of Hackerdom* (*Breve storia del regno degli*

hacker).

Ho scritto un *foglio*, *The Cathedral and the Bazaar* (*La cattedrale ed il bazaar*), che spiega molto circa il modo di operare della cultura Linux e *open-source*. Ho affrontato questo argomento anche più direttamente nel seguito *Homesteading the Noosphere*.

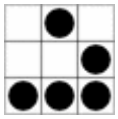
Rick Moen ha scritto un'eccellente documento su how to run a Linux user group.

Rick Moen ed io abbiamo collaborato ad un'altro documento circa How To Ask Smart Questions. Vi aiuterà a chiedere assistenza nel modo più congeniale per ottenerla.

Se avete bisogno di istruzione circa il funzionamento di Internet, Unix ed i personal computer, date un'occhiata a The Unix and Internet Fundamentals HOWTO (*I fondamenti di UNIX e di Internet*).

Quando rilasciate del software o scrivete delle patch per software, provate a seguire le linee guida del Software Release Practice HOWTO.

Se avete gradito il poema Zen, senz'altro vi piacerà Rootless Root: The Unix Koans of Master Foo.



Frequently Asked Questions (Domande più frequenti)

Q: Mi insegnerai a diventare un hacker?
Q: Da dove posso iniziare?
Q: Quando si deve iniziare? E' troppo tardi per me?
Q: Quanto impiegherò per imparare l'hack?
Q: Visual Basic o C# sono buoni linguaggi per iniziare?
Q: Mi aiuterai a crakkare un sistema, o insegnarmi come farlo?
Q: Come posso ottenere la password dell'account di qualcuno?
Q: Come posso leggere/controllare le mail di qualcuno?

Q: Mi insegnerai a diventare un *hacker*?

A: Da quando ho pubblicato questa pagina la prima volta, ho ricevuto svariate richieste ogni settimana (spesso svariate per giorno) da persone che mi dicevano "insegnami tutto riguardo l'hacking". Sfortunatamente non ho il tempo e l'energia per farlo; i miei progetti personali e il viaggiare come avvocato *open-source* impiegano il 110% del mio tempo.

Anche se l'ho fatto, l'hacking è un'attitudine che principalmente dovete insegnare a voi stessi. Vi accorgerete che se da un lato i veri *hacker* vi aiuteranno, dall'altro non vi rispetteranno se pretenderete di essere imboccati con il cucchiaino con quel che conoscono.

Imparate prima alcune cose. Mostrate che ci state provando e che siete in grado di imparare da soli. Poi rivolgetevi agli *hacker* con domande specifiche.

Prima di scrivere *mail* ad un *hacker* chiedendo consigli, ci sono due cose che dovete sapere. Primo, abbiamo notato che le persone pigre e noncuranti nella propria scrittura sono di solito troppo pigre e noncuranti nel loro modo di pensare per essere buoni *hacker* - quindi fate attenzione a parlare correttamente e ad usare correttamente la grammatica e la punteggiatura, altrimenti sarete ignorati. In secondo luogo, non osate chiedere risposte indirizzate ad un ISP (Internet Server Provider, *ndt*) differente da quello da cui scrivete; di solito scopriamo che le persone che lo fanno sono dei ladruncoli che utilizzano account rubati, e non abbiamo nessun interesse nell'assistere o gratificare i ladri.

Q: Come posso iniziare?

A: Probabilmente il modo migliore di iniziare è quello di frequentare un LUG (Linux User Group). Troverete questi gruppi alla pagina LDP General Linux Information Page. Sicuramente c'è un LUG nelle vostre vicinanze, possibilmente associato ad una università . I membri dei LUG vi forniranno una distribuzione Linux e certamente vi aiuteranno ad installarla e ad iniziare.

Q: Quando bisogna iniziare? E' troppo tardi per me?

A: Ogni età in cui siete motivati ad iniziare è buona. Sembra che la maggior parte delle persone abbia iniziato in età compresa tra i 15 e i 20 anni ma conosco eccezioni in entrambe le direzioni.

Q: Quanto impiegherò ad imparare l'hacking?

A: Questo dipende da quanto talento avete e quanto duramente ci lavorerete su. Molte persone possono acquisire delle capacità rispettabili in 18 o 24 mesi se si impegnano. Non pensiate sia finita qui, comunque; se siete dei veri *hacker*, impiegherete il resto della vostra vita imparando e perfezionando i vostri mezzi.

Q: Visual Basic o C# sono dei buoni linguaggi per iniziare?

A: Se state ponendo questa domanda, quasi certamente è perché state pensando di fare dell'hacking sotto Microsoft Windows. Questa è una pessima idea. Quando ho paragonato l'imparare a l'hacking sotto Windows ad imparare a ballare indossando un 'armatura non stavo scherzando. Non fatelo. E' triste e non smetterà mai di esserlo.

Ci sono problemi specifici con il VB ed il C#; principalmente perché non sono portabili. Sebbene ci siano prototipi di implementazioni *open-source* di questi linguaggi, gli standard applicabili dell'ECMA non coprono che un piccolo insieme delle loro interfacce di programmazione. Su Windows molte delle librerie di supporto sono proprietarie di un singolo venditore (Microsoft); se non sarete *estremamente* attenti alle caratteristiche che utilizzate - più attenti di quanto ogni principiante sia veramente in grado di essere - finirete costretti alle sole piattaforme che Microsoft sceglie di supportare. Se inizierete con Unix, migliori linguaggi con migliori librerie saranno a vostra disposizione. Python, ad esempio.

Visual Basic è particolarmente orribile. Come tutti gli altri Basic è un linguaggio dal design povero che vi insegnerà cattive abitudini di programmazione. No, non chiedetemi di descriverle in dettaglio perché ci vorrebbe un libro intero. Imparate piuttosto un linguaggio ben progettato.

Una di queste cattive abitudini è quella di essere dipendenti dalle librerie, dagli strumenti di sviluppo e dai gingilli di un singolo rivenditore (Microsoft). In generale, ogni linguaggio che non è completamente supportato in Linux o in

uno dei sistemi BSD, e/o almeno i tre diversi sistemi operativi di rivenditori diversi, è un linguaggio troppo "povero" per imparare l'hacking.

Q: Mi aiuterai a crackare un sistema o mi insegnerai a farlo?

A: No. Chiunque faccia domande del genere dopo aver letto queste FAQ è troppo stupido per essere educato, anche se avessi il tempo di farlo. Ogni email contenente una richiesta del genere che mi sarà recapitata sarà ignorata o otterrà risposte estremamente rudi.

Q: Come posso ottenere la *password* dell'account di qualcuno?

A: Questo è cracking. Idiota.

Q: Come posso leggere/controllare le mail di qualcuno?

A: Questo è cracking. Sparisci deficente.

Q: Come posso rompere i privilegi di un canale op su IRC?

A: Questo è cracking. Va via, cretino.

Q: Sono stato crackato. Mi aiuterai a difendermi da ulteriori attacchi?

A: No. Ogni volta che mi è stata posta questa domanda è stato da parte di qualche povero rimbambito che *gira* Microsoft Windows. Non è possibile, effettivamente, rendere sicuro un sistema Windows contro gli attacchi dei *cracker*; il codice e l'architettura hanno talmente tanti punti deboli che rendere Microsoft Windows sicuro è come voler *svuotare una barca piena d'acqua con un setaccio*. L'unica prevenzione affidabile è quella di passare ad un sistema Linux o qualche altro sistema operativo che sia stato progettato per essere in grado almeno di fornire la sicurezza.

Q: Ho dei problemi con il mio software Windows. Puoi aiutarmi?

A: Si. Vai al prompt del DOS e scrivi "format c:". Qualsiasi tipo di problema tu abbia cesserà in pochi minuti.

Q: Dove posso trovare dei veri *hacker* con cui parlare?

:

A: Il modo migliore è trovare un gruppo locale di utenti Linux o Unix e partecipare ai loro raduni (potete trovare dei link a varie liste di utenti sul sito LDP).

(Di solito a questo punto dico che non troverete nessun vero *hacker* su IRC, ma ultimamente mi sto ricredendo. Apparentemente alcune communities di veri *hacker*, legate a cose come GIMP e Perl, hanno canali IRC.)

Q: Puoi raccomandarmi libri utili su argomenti legati all'hacking?

A: Mantengo una Linux Reading List HOWTO che potrebbe esservi d'aiuto. Anche Loginataka può essere interessante.

Per un'introduzione a Python, visitate materiale introduttivo sul sito web di Python.

Q: Bisogna essere bravi in matematica per diventare un *hacker*?

A: No. L'hacking utilizza poca matematica e aritmetica formale. Di solito non avrete bisogno della trigonometria, del Calcolo o dell'Analisi (ad eccezioni di un piccolo gruppo di applicazioni per aree specifiche, ad esempio la *computer graphic*). Conoscere un po' di logica *formale* e algebra Booleana va bene, così come conoscere alcuni fondamenti di matematica discreta può essere di aiuto (insiemi finiti, calcolo combinatorio e teoria dei grafi).

Ben più importante: dovrete essere in grado di pensare in modo logico e seguire la giusta catena di ragionamento, che è ciò che fanno i matematici. Mentre la maggior parte dei concetti della matematica non vi saranno di aiuto, avrete bisogno della disciplina e dell'intelligenza utile per maneggiare con la matematica. Se mancate di intelligenza ci sono poche speranze per voi di diventare un *hacker*; se mancate di disciplina invece, sarà bene che la coltiviate.

Penso che un buon sitema per verificare se avete tutto ciò che vi occorre sia di prendere una copia del libro di Raymond Smullyan's *What Is The Name Of This Book?* (Qual'è il titolo di questo libro?). Gli scherzosi enigmi di logica Smullyan sono molto vicini allo spirito degli *hacker*. Essere in grado di risolverli è un buon segno; divertirsi nel risolverli è un segno anche migliore.

Q: Che linguaggio devo imparare per primo?

A: XHTML (l'ultimo dialetto dell'HTML) se già non lo conoscete. Ci sono un mucchio di splendidi, iper intensivi e *cattivi* libri sull'HTML in circolazione e pochi buoni, purtroppo. Quello che preferisco è *HTML: The Definitive Guide*.

Ma l'HTML non è un linguaggio di programmazione completo. Quando sarete pronti per iniziare a programmare vi suggerisco di iniziare con Python. Molte persone raccomandano Perl, e Perl è ancora più popolare di Python, ma è più difficile da imparare e (secondo me) progettato meno bene.

C è veramente importante, ma è ancora più difficile di Python e Perl. Non provate ad impararlo per primo.

Algi utenti Windows: *non* fermatevi a Visual Basic. Vi insegnerà cattive abitudini e non è portabile fuori da Windows. Evitatelo.

Q: Che tipo di hardware mi occorre?

A: Un tempo i Pc erano poco potenti e poveri di memoria, così tanto da porre dei limiti artificiali al processo di apprendimento degli *hacker*. Tutto questo è finito intorno alla metà degli anni 90; ogni macchina dal 486DX50 in su è più che sufficente per il lavoro di sviluppo, il server grafico X, e la comunicazione Internet, ed anche i dischi più piccoli che potete comprare attualmente sono molto capienti.

La cosa importante nello scegliere una macchina sulla quale imparare è verificare che il suo hardware sia compatibile con Linux (oppure BSD-compatibile, se doveste scegliete quella strada). Sarà così per la maggior parte delle macchine moderne e la sola area per la quale potreste incontrare delle difficoltà è quella dei modem; alcune macchine, invece, hanno dell'hardware specifico per Windows che non funzionerà con Linux.

C'è una pagina di FAQ sulla compatibilità hardware; l'ultima versione si trova qui.

Q: Voglio contribuire. Puoi aiutarmi a trovare un problema su cui lavorare?

A: Dovete essere auto-motivati o non reggerete (non riuscirete), poichè(visto che) lasciare che gli altri scelgano quale deve essere la vostra direzione non funziona quasi mai.

Provate a fare questo. Guardate per qualche giorno gli annunci dei progetti su Freshmeat. Quando ne troverete uno che vi farà pensare "Fico! Mi piacerebbe lavorarci!", allora seguitelo.

Q: Devo disprezzare e odiare Microsoft**?**

A: No, non dovete. Non che Microsoft non sia odioso, ma esiste una cultura *hacker* parecchio prima di Microsoft e ce ne sarà una per molto tempo dopo. Ogni energia che impiegherete per odiare Microsoft sarebbe bene impiegarla per affinare le vostre capacità . Scrivete buon codice, questo "colpirà "

Microsoft a sufficenza senza inquinare il vostro karma.

Q: Ma in qualche modo l'*open-source* non impedisce ai programmatori di farsi una vita?

A: Questo sembra essere molto inverosimile - da tempo l'industria del software *open-source* crea lavoro piuttosto che portarlo via. Se avere un programma già scritto è un guadagno economico netto rispetto a non averlo, un programmatore sarà pagato comunque, che il programma diventi *open-source* o meno una volta finito. E per quanto software "libero" possa essere scritto, ci sarà sempre domanda per applicazioni nuove e personalizzate. Ho scritto di più al riguardo nelle pagine dell'Open Source.

Q: Da dove posso cominciare? Dove posso rimediare uno Unix gratuito?

A: Qua e là in questa pagina ho incluso dei collegamenti su dove trovare gli Unix gratuiti più utilizzati comunemente. Per essere un *hacker* avete bisogno di motivazione, di iniziativa e dell'abilità di autoeducarvi. Iniziate adesso...