



SECONDA UNIVERSITÀ DEGLI STUDI DI NAPOLI

Facoltà di Ingegneria
Corso di Reti di Calcolatori

Sicurezza della Rete:
Programmi e tecniche di intrusione

Enrico Saviano
Nicola Bottone
Mario De Rosa

Internet e Sicurezza

In rete si è soggetti a molti pericoli.

Per quanto sia possibile darsi da fare per avere un livello di sicurezza il più elevato possibile, nessun computer connesso ad Internet sarà sicuro al 100%.

In rete si è soggetti a molti pericoli: furto di informazioni riservate, intercettazione del traffico, utilizzo di un pc aggredito per sferrare un attacco su altre macchine, defacing (modifica dei contenuti del proprio sito Web) e chi più ne ha più ne metta.

Come evidenziato, il problema della sicurezza in rete è di fondamentale rilevanza tanto da un punto di vista personale (possibili violazioni privacy) che professionale (diffusione di dati aziendali sensibili, perdita d'immagine, ecc...).

L'idea comune dell'utenza media "perché dovrebbero attaccare proprio me?" verte su presupposti privi di fondamento, giacché chiunque può essere vittima di attacchi informatici. Come dichiarò un famoso hacker "su internet non ci sono bocconi grandi e piccoli, bensì tutti sono prede grosse e appetibili".

Gli attacchi informatici avvengono quotidianamente in tutta la rete, coinvolgendo aggressori e vittime situati anche molto distanti tra di loro.

Un aggressore, può scegliere la propria vittima sulla base di particolari interessi personali:

- una azienda potrebbe voler attuare dello spionaggio industriale,
- un pirata informatico potrebbe voler violare una società di servizi bancari per rubare,
- o un amministratore di rete potrebbe voler semplicemente testare la sicurezza del proprio sistema informatico.

Una volta scelto il proprio obiettivo, l'aggressore passa ad una fase di analisi delle possibili vulnerabilità o falle, su tutto il sistema protocollare.

E' possibile individuare tre macro aree nelle vulnerabilità dei sistemi informatici:

- Vulnerabilità nel software, nelle applicazioni o nei sistemi operativi che girano sugli apparati di rete.
- Vulnerabilità nella eccessiva fiducia dei protocolli di rete, dei sistemi operativi e degli utenti.
- Vulnerabilità della di uno strato della rete più basso, che diventa sfruttabile da un aggressore quando questi controlla un nodo importante.

Parte Prima: Vulnerabilità nel software

Internet è costituita di applicazioni, ciascuna delle quali ricopre un ruolo diverso, che può essere l'instradamento, l'offerta di informazioni o la funzione di sistema operativo.

Moltissime nuove applicazioni fanno quotidianamente il proprio ingresso sulle scene, ma per essere davvero utili devono interagire con l'utente: che siano client di chat, siti Web di e-commerce o giochi in linea, tutte le applicazioni modificano dinamicamente l'esecuzione in base all'input degli utenti.

Essendo su Internet, un'applicazione è accessibile in modo remoto da altre persone e, se codificata in modo non corretto, apre il sistema alle vulnerabilità; una realizzazione del codice poco robusta può essere provocata da errori puramente tecnici, dall'inesperienza o da anomalie impreviste.

In ogni caso, è possibile che l'utente, di proposito o meno, invii dati che l'applicazione non prevede: questo evento potrebbe provocare un risultato nullo, modificare lo scopo prestabilito dell'applicazione, far sì che quest'ultima fornisca agli utenti informazioni che costoro di norma non possono raggiungere, oppure ancora modificare l'applicazione stessa o il sistema sottostante.

Port Scanning

Un attacco finalizzato ad ottenere l'accesso ad un host va lanciato contro un servizio vulnerabile ad un problema che permetta all'aggressore di ottenere l'accesso.

È possibile risalire ai servizi adoperati dal sistema con alcuni metodi di raccolta delle informazioni nonché sondare manualmente le porte di un sistema con utilità quali "netcat" per vedere se la connettività al servizio è fattibile.

Il processo di raccolta delle informazioni sui servizi disponibili è agevolato da strumenti come "nmap" (network mapper). Quando è usato su un sistema, nmap, si serve di numerose funzioni avanzate per identificare le caratteristiche di un host; tali funzioni includono la scansione variabile del flag TCP e l'analisi della risposta IP, per intuire il sistema operativo e identificare i servizi in ascolto su un host. Nmap può essere usato per identificare i servizi di un sistema aperti all'uso pubblico e quelli che sono in ascolto ma vengono filtrati attraverso un'infrastruttura, come i wrapper TCP o l'attività dei firewall.

Altri strumenti come Nessus e SAINT comprendono inoltre un database di vulnerabilità specifiche che ricercano tra i servizi dell'host.

Le due tipologie di vulnerabilità software più diffuse sono l'overflow del buffer e le stringhe di formato.

L'overflow del buffer

Gli attacchi di tipo buffer overflow rappresentano la stragrande maggioranza di tutti gli attacchi ai sistemi informatici in quanto le vulnerabilità del buffer overflow sono comuni e quindi relativamente facili da sfruttare.

Le vulnerabilità del buffer overflow sono la principale causa di intrusioni in sistemi informatici anche perché esse offrono all'attaccante la possibilità di eseguire codice dannoso. Il codice introdotto viene eseguito dal sistema con i privilegi del programma violato e permette all'attaccante di poter controllare altri servizi dell'host vittima che gli interessa controllare.

Questa tecnica viene eseguita spesso per fornire una shell remota sulla macchina di destinazione, utile per un ulteriore sfruttamento, come il furto di password o altre informazioni riservate, l'alterazione delle configurazioni di sistema e/o l'installazione di backdoor e così via.

La maggior parte dei programmi applicativi dispone di buffer di dimensioni fisse che contengono i dati. Se un aggressore invia troppi dati in uno di questi buffer e il programma non controlla le dimensioni dei dati, il buffer è soggetto a overflow.

Durante l'esecuzione di una applicazione, ogni volta che c'è una chiamata ad una funzione (CALL), il processore si occupa di salvare in memoria il valore dell'EIP corrente in modo da potersi riposizionare in quella stessa posizione al termine della funzione che si sta chiamando.

Il programma invece appena viene raggiunto l'inizio della funzione dovrà subito salvare il puntatore EBP che puntava all'inizio del precedente stack e dopodiché lasciare dello spazio proprio prima di esso in modo che venga utilizzato dalle variabili.

Questa semplice operazione permette infatti al programma di poter ripescare il puntatore all'istruzione che abbiamo lasciato prima di chiamare la funzione, non appena quest'ultima ritornerà al chiamante.

In Assembly, quando una funzione inizia, la prima cosa che farà quindi è questo:

```
push ebp
mov ebp, esp
sub esp, MEMORIA_PER_LE_VARIABILI
```

Immagazzina il vecchio EBP, dice ad EBP dove inizia il nuovo stack e successivamente alloca lo spazio per le variabili che appunto verranno posizionate prima di EBP ed EIP.

I problemi con il buffer overflow non si vedranno subito dopo aver sovrascritto il buffer ed i 2 registri salvati, ma si avranno dopo che la funzione tenterà di ritornare alla vecchia posizione precedentemente salvata nello stack (dove si trovano le istruzioni che dovevano essere eseguite dopo la chiamata alla funzione).

Invece di ritrovarsi al vecchio indirizzo, il programma arriverà alla posizione indicata dal registro EIP che, tramite l'istruzione RET alla fine della funzione, si ritroverà al suo interno i bytes che erano stati immessi precedentemente e che hanno causato la sovrascrittura della memoria

adiacente al buffer di destinazione.

Le funzioni più problematiche per quanto riguarda il buffer overflow in C sono:

```
char *strcpy (char *strDestination, const char *strSource)
char *strcat (char *strDestination, const char *strSource)
int sprintf (char *buffer, const char *format [, argument] ...)
char *gets (char *buffer)
```

Esempio pratico:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int bufferoverflow() {
    char buffer[8];
    strcpy (buffer, "AAAAAAAAAAAAAAAAAAAA");
    return 1;
}

int main (int argc, char **argv) {
    bufferoverflow();
    printf("Questo punto non sarà mai raggiunto");
    return 1;
}
```

Questo programma chiama semplicemente la funzione `bufferoverflow()`, quando si trova nella funzione, una stringa di 20 'A' viene copiata in un buffer che può contenere al più 8 byte; si verificherà così un overflow del buffer incontrollato; si noti inoltre che la funzione `printf` non sarà mai chiamata, poiché l'overflow devia il controllo al ritorno di `bufferoverflow()`.

Questo overflow del buffer non è sfruttabile per un aggressore, in quanto non è possibile modificare dall'esterno del programma la stringa che va a riempire il buffer.

Se però sostituissimo a `strcpy` la funzione `gets`, sarebbe possibile dall'esterno immettere una stringa codificata ad hoc, per il controllo dell'overflow del buffer.

Per gestire in modo "utile" questa vulnerabilità, un aggressore crea una stringa composta da byte di saturazione del buffer, più un "carico di lavoro".

I carichi di lavoro sono il codice effettivo che un aggressore tenta di eseguire. Come carico di lavoro è possibile scrivere pressochè tutti i codici; spesso il codice del carico di lavoro consiste semplicemente di istruzioni ridotte in assembly che consentono ad un aggressore di chiamare una funzione di sistema come ad esempio una shell remota.

Vulnerabilità di tipo buffer overflow ne vengono classificate a migliaia e scoperte decine ogni giorno in tutti i più disparati programmi della rete, come il famoso bug di `sendmail`, e le numerose vulnerabilità in Internet Explorer, MSN Messenger, nei numerosi servizi di Microsoft Windows o in Apple iTunes.

Difese

Per difendersi contro vulnerabilità da Buffer Overflow si possono scegliere diverse strade: scrivere codice sicuro, non rendere eseguibile l'area di memoria che contiene le variabili, controllare la dimensione degli array ogni volta che vengono inseriti oppure verificare l'integrità

dei puntatori.

- **Scrivere codice sicuro**

E' sicuramente la scelta migliore anche se la più costosa, in quanto bisogna controllare ogni singola parte del codice, procedimento noioso e molto lungo. Esistono dei veri e propri gruppi di persone che controllano pezzi di codice per verificarne la sicurezza e rilasciano patch. Bisogna fare particolare attenzione a chiamate a funzioni di libreria molto vulnerabili, che non controllano la lunghezza degli argomenti che manipolano. controllare tutti i dati che vengono immessi prima di essere utilizzati.

Per controllare il codice si utilizzano programmi di debug, che immettendo a caso nei buffer utilizzati codice d'attacco servono per individuare eventuali vulnerabilità.

- **Buffer non eseguibile**

Con questo metodo si dovrebbe rendere non eseguibile l'area di memoria utilizzata dalle variabili, in modo da impedire all'attaccante di eseguire il codice inserito. Sui sistemi operativi recenti (sia Windows che Unix) una soluzione del genere è ormai impensabile, dato che per ottimizzare la loro esecuzione deve esserci la possibilità di inserire codice dinamico in memoria.

Tuttavia sono presenti delle patch per Linux e Solaris con cui è possibile rendere lo stack non eseguibile e preservare la compatibilità della maggior parte dei programmi.

La protezione offerta da questo procedimento permette di difendersi dall'inserimento di codice eseguibile nelle variabili automatiche, ma non protegge da altre forme d'attacco.

- **Controllo della dimensione degli array**

Non basta inserire codice per realizzare un buffer overflow, ma è anche necessario modificare il flusso del programma in esecuzione. Controllando la dimensione degli array si previene da questo tipo d'attacco. Se in un array non si può scrivere oltre la sua dimensione non è possibile corrompere i dati adiacenti nello stack: ogni lettura e scrittura in un array è quindi riferita ad uno spazio di memoria pari alla dimensione dell'array stesso.

- **Controllo dell'integrità dei puntatori**

Con questo metodo si controlla se un puntatore è stato sovrascritto prima di essere usato: se un attaccante riuscisse in qualche modo a modificare un puntatore, questo non verrebbe utilizzato. Tuttavia anche questa è una soluzione parziale al problema, in quanto tutti gli attacchi che non riguardano puntatori andranno a segno.

Stringhe di formato

Le vulnerabilità da stringhe di formato rappresentano una grave minaccia per i server e le applicazioni commerciali, esse sono una nuova classe di vulnerabilità del software scoperte nel giugno del 2000.

Una vulnerabilità di tipo stringa di formato può essere sfruttata per mandare in crash un programma o per eseguire localmente o da remoto del codice arbitrario su un sistema.

Il problema delle stringhe di formato si verifica quando a certe funzioni C che effettuano la formattazione del testo, come `printf`, `fprintf`, o `sprintf`, vengono passati dati in ingresso inseriti dall'utente, direttamente come parametri stringa di formato, senza essere prima filtrati.

Un aggressore potrebbe usare i token `%s` e `%x`, per stampare i dati dallo stack o da altre locazioni di memoria. Egli potrebbe anche scrivere dati arbitrari in locazioni di memoria arbitrarie usando il token di formato `%n`, che assegna alla variabile che segue la stringa di formato, il numero di byte che compongono la stringa di formato stessa:

Esempio:

```
#include <stdio.h>
main () {
int byte;
printf("ABCDE%n\n", &byte);
}
```

In questo esempio all'indirizzo di memoria riservato alla variabile di tipo intero "byte", sarà scritto il valore numerico 5.

Le vulnerabilità di questo tipo sono frequenti quando un programmatore vuole stampare una stringa contenente dati forniti dall'utente. Il programmatore potrebbe commettere l'errore di scrivere `print(buffer)` invece di `printf("%s", buffer)`. La prima versione interpreta `buffer` come una stringa di formato, ed analizza qualsiasi tipo di formattazione che essa possa contenere. La seconda versione semplicemente stampa una stringa allo schermo come il programmatore originariamente voleva fare.

Le vulnerabilità di tipo stringa di formato nascono perchè le convenzioni per il passaggio degli argomenti del C è libera rispetto al tipo di dati.

In particolare, il meccanismo `varargs` consente alle funzioni di accettare un numero qualsiasi di argomenti (come in `printf`).

Esistono tre scopi basilari che gli hacker raggiungono sfruttando le vulnerabilità delle stringhe di formato: innanzi tutto, possono provocare il fallimento di un processo dovuto ad un accesso non valido alla memoria, generando una negazione del servizio; possono leggere la memoria del processo se la stringa formattata viene stampata; infine, gli aggressori possono sovrascrivere la memoria, riuscendo così a eseguire le istruzioni.

Questa specifica vulnerabilità è molto comune poiché nonostante bug di questo tipo fossero noti da molto tempo, erano ritenuti innocui.

La prima vulnerabilità sfruttabile di tipo stringa di formato fu riscontrata nel giugno del 2000, quando fu pubblicato un programma [exploit](#) per il server [WU-FTPD](#). Questo programma consentiva agli aggressori remoti di ottenere l'accesso root agli host che eseguivano WU-FTPD senza autenticazione, se era abilitato FTP anonimo; FTP anonimo era attivato per impostazione predefinita in numerosi sistemi. Si trattava di una vulnerabilità di alto profilo, dato che WU-FTPD

è largamente utilizzato su Internet.

Da allora sono state scoperte migliaia di vulnerabilità di questo tipo, alcune molto rilevanti, a causa della loro diffusione:

IRIX Telnetd: I dati forniti dal client e inclusi nell'argomento della stringa di formato di syslog() hanno consentito agli aggressori remoti di eseguire codice arbitrario senza autenticazione. Di questa stessa vulnerabilità soffrono anche i server telnet di AIX (IBM) e Solaris (Sun Microsystems).

Linux rpc.statd: Anche questa vulnerabilità è stata causata dall'uso improprio di syslog() ed era sfruttabile per ottenere remotamente i privilegi di root.

Cfingerd: Un'altra vulnerabilità delle stringhe di formato dovuta a syslog(). Sfruttandola remotamente, gli aggressori ottengono il controllo dell'host soggiacente.

Vulnerabilità più recenti di tipo stringa di formato sono ad esempio quella del server radio web SHOUTcast v1.9.4 della nullsoft, in entrambe le versioni per linux e windows, nella funzione auth_debug() del diffusissimo server IMAP Courier-IMAP e nel server dhcpcd open source della Internet Systems Consortium (ISC).

Exploit

Alla scoperta di una nuova vulnerabilità segue spesso il rilascio di un particolare software detto "exploit". Un exploit è generalmente un proof-of-concept, ovvero, una prova della effettiva sfruttabilità del bug scoperto.

Spesso accade che gruppi di hacker rilascino exploit di facile utilizzo che permettono di sfruttare vulnerabilità specifiche ed ottenere accesso immediato ai sistemi vulnerabili.

Rootkit

Una volta penetrato nel sistema vittima un aggressore ha lo scopo primario di garantirsi il facile rientro nella macchina colpita.

Benchè possa semplicemente riutilizzare la tecnica adoperata per entrare la prima volta, di solito un pirata informatico preferisce "patchare" la macchina, rattoppando la vulnerabilità appena sfruttata per entrare, al fine di non permettere ad altri pirati informatici di introdursi nel sistema. Per poter agevolmente rientrare nell'host vittima, quindi, egli installa speciali applicativi detti "backdoor" (porta sul retro), che possono essere semplici server telnet o ssh, nascosti, adoperanti su porte non standard, o anche più complessi trojan.

Altra necessità del pirata informatico è quella di nascondere le sue tracce, quelle che lascia nell'utilizzare il sistema, e quelle che ha lasciato nei precedenti tentativi di intrusione (come log, history file, e quant'altro). Lo strumento che viene incontro a questa necessità è detto "cleaner"

(pulitore), che va alla ricerca di particolari stringhe nei file di log del sistema e le rimuove, alterando così i registri di sistema, alcuni dei più famosi sono zap, vanish e wipe.

In generale, per facilitare il suo compito illecito, un pirata informatico installa sulla macchina appena violata, un rootkit: una suite di tool comprendente backdoor, cleaner e altre applicazioni atte a nascondere la presenza dell'intruso nel sistema colpito, nascondendo quindi eventuali porte adoperate, o processi di applicazioni illegittime eseguite dall'intruso.

I rootkit più famosi sono:

Adore: un modulo per il kernel linux che altera le funzioni del kernel al fine di nascondere porte e processi all'amministratore di sistema;

t0rn's: un pacchetto di applicazioni completo per lo stealthing di un intruso in un sistema linux;

X-Org: suite per i sistemi Solaris SunOS.

Per difendersi da questi software, esistono dei tool di rilevazione, per lo più a pagamento, che ricercano la presenza di eventuali rootkit sul sistema. Per i sistemi Unix-Like (Linux, BSD, HP-UX, Solaris, Tru64), il più famoso di tutti è certamente chkrootkit, uno script bash, del tutto libero e gratuito (rilasciato sotto licenza GPL), che riesce a rilevare più di 50 rootkit diversi.

Parte Seconda: Vulnerabilità nella "fiducia"

Molte tecniche di attacco verso sistemi informatici si basano sulla mancanza di controlli, della rete o dei suoi utenti verso informazioni provenienti dall'esterno.

Un aggressore spesso camuffa un attacco nascondendo il pericolo all'interno di un apparentemente innocuo messaggio di posta o in un qualsiasi file eseguibile. In questo caso la macchina dell'utente può essere vittima di un virus o di un programma trojan.

Virus e Trojan

I **virus**, intesi in senso stretto, altro non sono se non dei piccoli "programmi" incaricati di svolgere determinate istruzioni, tendenzialmente votate a creare ostruzionismo o danni al sistema dell'utente aggredito o ai dati dello stesso. Si diffondono nella maggior parte dei casi tramite posta elettronica/trasferimento di file infetti ed hanno la capacità, una volta eseguiti, di occultarsi e moltiplicarsi riproducendo il proprio codice in altre parti del sistema, con fini distruttivi.

Worm - vermicciattoli dannosi

Il **worm**, simile, ma diverso dal virus, è capace di diffondersi rapidamente, utilizzando in particolare la rete per la sua massiccia diffusione.

Possono agganciarsi al sistema operativo per cambiarne alcune impostazioni, di modo che all'avvio successivo il pc possa automaticamente eseguirlo permettendogli le azioni per le quali è concepito.

Naturalmente il principale veicolo di diffusione è Internet. Una volta azionato un Worm scandaglia Internet alla ricerca di altri computer da infettare, cercando e sfruttando bug e vulnerabilità eventualmente presenti. I worm Nimda (2001), Blaster (Agosto 2003) e Sasser (2004), si sono propagati a livello endemico, colpendo milioni di computer nel mondo.

Trojan - tutto cominciò con Ulisse

Il **trojan** (cavallo di troia) non è in grado per sua intrinseca natura di autoreplicarsi, ma può essere altamente pericoloso e dannoso se eseguito nel sistema nel quale viene "ospitato". La sua azione può essere ad esecuzione immediata o ritardata.

In pratica il Trojan tiene costantemente aperte delle porte che i malintenzionati possono tenere sotto controllo al fine di intrufolarsi nel PC colpito.

Sfruttando un Trojan è possibile sbirciare documenti e dati sensibili, rubare password (grazie a programmi chiamati keylogger), distruggere dati e persino svolgere azioni illegali utilizzando il pc infetto come stazione di rilancio. Tra i trojan che hanno fatto epoca, citiamo Netbus e Subseven.

Spesso gli antivirus si rivelano insufficienti a debellare i trojan. Pur riuscendo a riconoscere quelli più importanti e diffusi, non sono in grado di riconoscere e controllare i nuovi arrivati. Questo poichè l'anti-virus si basa essenzialmente sul riconoscimento e la verifica delle "firme" di ogni

Trojan; un qualsiasi programmatore esperto è in grado di creare una versione "modificata" che non sarà identificata dagli antivirus.

A tal fine sono utili altri software specificamente disegnati, predisposti e diretti alla rimozione dei trojan (tra gli anti-trojan più famosi citiamo "Pest Patrol"). Spesso si trascura di abbinare all'antivirus un ottimo antitrojan, pensando erroneamente che il primo sia da solo sufficiente a garantire la massima sicurezza al nostro sistema.

Spyware - Privacy a rischio

Esiste una subdola tipologia di minaccia informatica che di fatto non danneggia il PC. Si definiscono spyware quei programmi che si installano in maniera infida, spesso all'insaputa dell'utente e che hanno come scopo la raccolta di dati sulle proprie abitudini. Quali siti visitiamo, con che frequenza, che oggetti compriamo negli shop online, fino a vere violazioni della privacy, come la raccolta di dati sensibili.

Metodi più diffusi di infezione

Il modo più classico per infettare il proprio sistema con virus e altri codici maligni è l'esecuzione di file allegati con la posta elettronica, ma si vanno sempre più diffondendo metodi alternativi, quali l'esecuzione di uno script infetto durante la navigazione - come per la maggioranza dei dialer.

Le estensioni più pericolose sono ovviamente quelle associate a files eseguibili, come *.exe, *.com, *.bat, *.pif, *.scr o i files con doppia estensione (es. nomefile.doc.pif).

Approfondimento Trojan

Anche se vengono catalogati e riconosciuti come virus dai software antivirus, i trojan horse sono dei programmi che non creano danni in modo autonomo se installati sul computer, in quanto altro non sono che programmi creati appositamente per il controllo e la gestione remota del pc infettato (controllo remoto/controllo a distanza tramite collegamento,internet o di altro tipo), da altri utenti anche dall'altro capo del mondo, grazie al collegamento internet.

Il trojan si compone di due programmi separati: il programma server ed il programma client: il primo si va ad installare nel pc vittima, il secondo viene usato dalla persona che effettua l'attacco, in modo che possa entrare nel pc server (quello infettato) e prenderne il totale controllo. I trojan più evoluti hanno anche una terza parte, l'editor server, che serve appunto per poter settare a distanza la parte server del programma. La diffusione del trojan puo' avvenire in diversi modi, i più classici sono quelli di lasciare il programmino del trojan horse (di pochi Kb), camuffato con altro programma o utilità in vari siti, in attesa che qualcuno lo scarichi, oppure l'invio in allegato con email, lo scambio di file in linea con programmi di chat (tipo con irc o icq, il

file deve essere accettato dall'utente). Il più delle volte il trojan sarà "mascherato" e perfettamente integrato con un programma comune, perfettamente funzionante, e quindi non riconoscibile come un programma che ci potrebbe creare problemi. Al momento dell'installazione il Trojan si maschera ulteriormente nel sistema, non è quindi rilevabile come per gli altri programmi.

Di trojan ne esistono molti tipi, circa 500, e ogni giorno ne vengono creati di nuovi, ma i più noti sono Back Orifice (comunemente chiamato Bo) e il sopracitato Netbus.

Spoofing

Gli attacchi di spoofing consistono nel fornire false informazioni su un'identità principale per ottenere l'accesso non autorizzato ai sistemi e ai loro servizi.

Il concetto di assumere l'identità di un'altra persona è il nocciolo dello spoofing, il cui esempio classico è IP. In sostanza, TCP/IP ed Internet, ripongono fiducia negli utenti, che specificano il loro indirizzo di origine quando comunicano con altri host. Tuttavia, è compito del mittente del messaggio determinare l'indirizzo di origine.

Arp-Spoofing

uno dei più famosi tipi di attacco riguardanti il furto dei dati che viaggiano tra due macchine attraverso la rete ethernet è l'**arp-spoofing**. Questo tipo di attacco sfrutta il protocollo ARP (Address Resolution Protocol), un protocollo utilizzato per conoscere/diffondere il MAC address di una data macchina. Infatti, se un computer volesse spedire dei dati ad un'altra macchina sulla stessa rete, dovrà anzitutto consultare la propria ARP table in cui vi sono delle associazioni **1:1** dei MAC address e dei relativi IP address, se non vi è questa associazione, la macchina manda un ARP request a tutte le macchine della propria rete ethernet, per conoscere il MAC address della macchina interessata, la macchina se è presente risponderà alla richiesta (tramite un ARP reply), e la tabella ARP (residente in una porzione della RAM) sarà aggiornata.

Il MAC address è l'indirizzo della scheda di rete (Nic), che è univoco a livello internazionale, formato attraverso **48 bits** di cui i primi 24 sono forniti attraverso l'OUI, e gli ultimi 24 sono scelti dall'industria che produce le schede di rete; per motivi logici in una rete ethernet ogni indirizzo IP è diverso dall'altro e quindi anch'esso univoco.

Tutto ciò deve avvenire per ovvi motivi di "comunicatività standard", infatti in questo modo si riesce ad osservare le esigenze del Modello OSI, completando i campi richiesti per garantire una connessione con un'altra macchina in rete. La comunicazione all'interno della propria rete ethernet è garantita, ma come può un attaccante aggirare questo sistema a suo favore?

possiamo distinguere due casi:

1) Local Area Network con Hub:

La rete aziendale dispone di un hub (un multi-port repeater), un hub è un dispositivo di rete che lavora basandosi sul Layer 1 del modello OSI, ciò significa che se dovesse ricevere del traffico su

una porta, lo stesso traffico verrebbe ripetuto in uscita sulle altre porte indirizzando ad altri dispositivi o macchine di rete tutto il traffico in entrata dall'hub.

Una volta che il traffico arriva alla scheda di rete, la stessa controlla se il MAC address di destinazione, è **uguale** al proprio MAC address registrato sulla card, agendo in questo senso da **filtro hardware**.

Il traffico all'interno della LAN viene inviato, duplicato ed eventualmente scartato, ma è possibile che un attaccante possa non scartare questo traffico di rete, analizzandolo e registrando tutti i dati; questo procedimento consiste nel porre in **modalità promiscua** la scheda di rete, eliminando il filtro hardware della scheda, e permettendo in questo modo al kernel del Sistema Operativo di ricevere tutti i pacchetti, e gestirli, anche quelli non indirizzati al MAC address appartenente a quella determinata scheda di rete. Infine un attaccante attraverso un programma di analisi del traffico di rete, uno sniffer, è in grado di catturare i pacchetti, limitando eventualmente il traffico da analizzare a certi host e catturando dati che potrebbero essere essenziali per la vita aziendale.

2) Man in the Middle, presenza di uno switch sulla rete:

Con uno Switch la rete è al sicuro da questo genere di attacchi?, la risposta è ovviamente no, vi è un'altra tipologia di attacco che sfrutta il protocollo ARP e le tabelle ARP, per sniffare i dati di una data macchina:

- l'attacco Man in The Middle, è sicuramente la più famosa tecnica di attacco che sfrutta l'**arp-poisoning**, e che non necessita di schede di rete in modalità promiscua. Per arp poisoning si intende l'attività di contraffazione della tabella ARP di una data macchina; come già detto una macchina in una rete ethernet dispone di un tabella ARP in una porzione della RAM, che utilizza per le sue associazioni con gli indirizzi IP; queste tabelle si aggiornano attraverso il solo ricevimento di un ARP reply di una data macchina, modificando o aggiungendo dati; se un'attaccante dovesse spedire ad una macchina un ARP reply con il proprio MAC address (dell'attaccante), e indirizzo IP della macchina obiettivo, il traffico della macchina obiettivo verrebbe reindirizzato sulla macchina dell'attaccante; operando in simmetria sulla macchina obiettivo, si riesce a creare un attacco di tipo Man In the Middle:

Esempio pratico:

Macchina_Attaccante, IP address 192.168.0.1, MAC address, per semplicità poniamo 1,

Macchina_Vittima_1, IP address 192.168.0.2, MAC address 2,

Macchina_Vittima_2, IP address 192.168.0.3, MAC address 3.

La Macchina_Attaccante spedisce un ARP reply alla Macchina_Vittima_1, il pacchetto contiene la nuova associazione 1:1, IP: 192.168.0.3, MAC address: 1;

La Macchina_Attaccante spedisce un ARP reply alla Macchina_Vittima_1, il pacchetto contiene la nuova associazione 1:1, IP: 192.168.0.2, MAC address: 1;

Si è già delineato il procedimento che l'attaccante vuole porre in essere, creare una macchina "di

mezzo", il tramite tra i due computer, ricevendo i loro pacchetti, sniffandoli e reindirizzandoli. Si comprende come non sia necessario che la scheda di rete entri in modalità promiscua, poiché il MAC address dell'attaccante è corretto.

Una breve attività di forging dei pacchetti ARP reply ha consentito ad un attaccante di sniffare una connessione tra due macchine obiettivo, che magari scambiano dati riservati, ed essenziali per l'azienda.

Una ottima soluzione sarebbe quella di settare lo Switch in maniera tale da impedire il traffico di rete in uscita da una determinata porta di cambiare MAC address.

Parte tre: Tecniche di controllo della rete

Quando un aggressore controlla una certa zona della rete, che può essere un host della LAN, un router di internet, o un semplice plug ethernet, può reperire informazioni riservate che viaggiano da un host all'altro mediante tecniche dette di "sniffing".

Sniffing

Ogni sistema su una rete IP scambia informazioni con altri sistemi tramite singoli pacchetti che hanno un IP sorgente e un IP destinazione. Tipicamente un computer analizza e processa solo i pacchetti che arrivano al suo dispositivo di rete (una scheda ethernet, un modem ecc.) che hanno come IP di destinazione il proprio o che sono pacchetti di broadcast, indirizzati cioè ad ogni indirizzo IP attivo nello stesso network IP.

L'attività di sniffing il più delle volte è necessaria per monitorare e diagnosticare problematiche di rete ma può essere impropriamente utilizzata per intercettare informazioni sensibili di terzi, come login e password di accesso ad un determinato servizio.

Si possono sniffare pacchetti su ogni interfaccia di rete, ma tipicamente viene fatto su una scheda ethernet, per la quale possono esistere due tipi di ambienti tipici:

- Rete shareata, dove tutte le schede di rete dei computer nella rete locale ricevono TUTTI i pacchetti, anche quelli destinati ad altri indirizzi IP. In questo caso (rete ad anello con cavo coassiale oppure rete a stella con un hub centrale) le schede di rete dei PC normalmente selezionano solo i pacchetti destinati a loro e scartano tutti gli altri che attraversano il mezzo trasmissivo a cui sono collegati.
- Rete switchata, dove ogni PC riceve solo i pacchetti di broadcast o quelli destinati al proprio IP. Questa è tipicamente una rete a stella con uno switch al centro, che provvede autonomamente a forwardare ad ogni sua singola porta solo i pacchetti destinati all'IP del dispositivo collegato a quella porta, oltre ai broadcast, che vengono sempre propagati su tutte le porte.

Nel primo caso l'attività di sniffing permette di analizzare anche pacchetti destinati e originati da indirizzi terzi, ampliando notevolmente le possibilità di intercettare informazioni sensibili.

In una rete switchata, invece, quando si prova a sniffare i pacchetti che passano per l'interfaccia di rete, si possono solo incontrare pacchetti originati dal o destinati al computer locale, oltre ai soliti broadcast.

Esiste tuttavia una tecnica piuttosto evoluta (l'arp spoofing, di cui abbiamo precedentemente parlato) tramite la quale è possibile sniffare pacchetti di terzi anche in una rete switchata.

Quando si vogliono sniffare pacchetti destinati a terzi, si deve impostare il "PROMISCUOUS MODE" sull'interfaccia di rete, in modo da farle processare tutti i pacchetti indifferentemente.

Esistono diversi strumenti di sniffing, alcuni sono esplicitamente realizzati per attività di hacking (es. Sniffit, Ettercap, Dsniff) e evidenziano le login e le password che sono state intercettate, altri sono più orientati alla risoluzione di problematiche di rete (Ethereal, simile al Windows Network Monitor) e permettono l'analisi di tutti i pacchetti intercettati, altri hanno funzioni di monitoring e analisi a volte limitandosi a considerare solo le intestazioni dei pacchetti (tcpdump, snoop, iptraf, ntop).

Il motivo principale per cui si preferisce cercare di criptare ogni passaggio di login e password in rete (https, ssh, pop3s, sftp ecc.) è proprio per evitare che qualcuno, tramite sniffing, li possa intercettare e facilmente scoprire.

Hijacking

L'hijacking di sessione è solitamente una estensione dello sniffing, con la differenza che quest'ultimo è passivo, mentre l'hijacking di sessione richiede una partecipazione attiva.

Questa tecnica sfrutta le debolezze presenti in gran parte delle reti e dei protocolli non crittografati e quindi utilizza la stessa debolezza sfruttata dallo sniffing. Oltre al monitoraggio, l'attacco di hijacking può anche iniettare un pacchetto o un frame che finge di essere uno degli host comunicanti; questa azione è simile allo spoofing, ma non prevede l'intuizione perché l'aggressore dispone di tutti i dati che gli occorrono.

Hijacking di sessione TCP

Occorre innanzitutto ricordare che in una sessione TCP la connessione ha inizio con l'handshaking a tre fasi. Durante lo scambio i contatori di sequenza aumentano su entrambi i lati e la ricezione del pacchetto viene riconosciuta con i pacchetti ACK; la connessione ha termine sia con uno scambio di pacchetti FIN sia in modo più brusco con RST.

L'aggressore effettua l'iniezione dei pacchetti prima che la connessione abbia termine. Come detto in precedenza il succo dell'hijacking è impadronirsi della fiducia, che non esiste finché l'autenticazione non ha avuto luogo.

Se un aggressore riesce ad eseguire un hijacking di sessione TCP in modo da controllare totalmente la trasmissione dei pacchetti tra due host, sicuramente godrà di un vantaggio notevole e potrà anche emulare perfettamente l'altro capo della conversazione su uno degli host; egli potrà modificare la tabella di instradamento affinché i pacchetti scorrano in un sistema che controlla, modificare le tabelle del bridge eseguendo trucchi con i frame spanning tree o, infine, reinstradare fisicamente i cavi in modo da costringere i frame a scorrere nel sistema dell'aggressore. Quasi sempre la modifica delle tabelle di instradamento avviene in modalità remota.

Hijacking di sessione UDP

Effettuare un hijacking con il protocollo UDP è di gran lunga più facile in quanto questo è sprovvisto di tutte le funzioni di affidabilità incorporate nel protocollo TCP.

I più famosi programmi per l'hijacking di sessione sono:

Juggernaut: è stato uno dei primi programmi a svolgere queste funzioni, e tutt'oggi ben pochi strumenti eseguono la funzioni di hijacking di cui esso è munito.

Hunt: è un progetto più ambizioso di Juggernaut per lo meno nelle ambizioni, Hunt infatti aggiunge alcuni strumenti ARP per eseguire lo spoofing, allo scopo di condurre gli host vittima ad attraversare la macchina di attacco così da eliminare i problemi della tempesta ACK, tipicamente associati all'hijacking di sessione TCP.