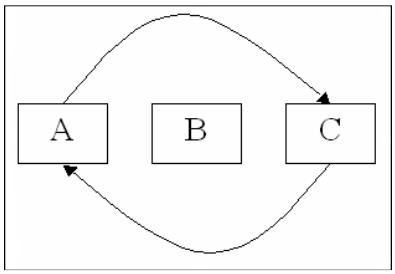
L'attaccante sostituendo il valore del campo in input (`valoreInput`) con `%x` farà perdere all'applicazione il riferimento corretto: l'applicazione cercherà il valore corrispondente nella memoria stack senza riuscire a trovarlo. A questo punto l'attacco ha conseguenze ancora più gravi se all'indirizzo di memoria di quella variabile, l'attaccante fa corrispondere una funzione inserita ad hoc dallo stesso.

Contromisure

Non utilizzare mai l'input dell'utente come stringa di formattazione per le funzioni tipo printf e scanf senza averlo prima verificato.

6.5.4 Heap overflow

I buffer allocati dinamicamente da un'applicazione risiedono nella regione di memoria heap e sono sottoposti a problematiche di overflow così come quelli residenti nello stack. Un luogo comune del passato oramai sfatato era che problematiche di questo tipo non potessero essere sfruttate da un aggressore per eseguire uno shellcode per via dell'assenza di un indirizzo di ritorno che potesse essere utilizzato come puntatore al codice malevolo. Un heap overflow si manifesta solitamente quando un buffer che viene deallocato contiene dati arbitrari provenienti da input utente o quando successivamente ad un overflow ne viene allocato uno nuovo. In entrambi i casi, secondo l'architettura, si viene a creare una condizione adatta per l'esecuzione fraudolenta di uno shellcode. La tecnica è resa possibile manipolando i puntatori alle aree di memoria (chunk) che vengono liberati/allocati.
Presi tre elementi (A, B e C) appartenenti a una lista circolare, per liberare la memoria di B, A dovrà riconoscere C come elemento successivo e C dovrà riconoscere A come elemento precedente:

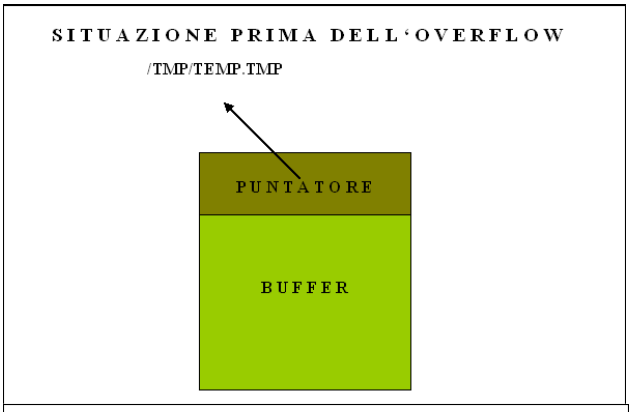


Quando l'applicazione deve allocare un nuovo buffer dinamico, l'Heap Manager osserva questa lista per determinare quale è il prossimo chunk utilizzabile ed aggiorna opportunamente i puntatori. Quando l'applicazione deve liberare un buffer dinamico, l'Heap Manager aggiorna allo stesso modo i puntatori per tenere traccia dei chunk inutilizzati. Gli indirizzi di memoria indirizzati da tali puntatori vengono mantenuti all'interno di strutture apposite (header) anteposte a ciascun chunk. Con il manifestarsi di un Heap Overflow, l'header del chunk adiacente può essere artificiosamente modificato dall'aggressore che, manipolando a piacimento i puntatori della struttura, può scrivere un qualsiasi valore all'interno di qualunque indirizzo residente nello spazio di memoria del processo in esecuzione.
L'esecuzione di codice malevolo attraverso un Heap Overflow si sostanzia fondamentalmente in quattro step:

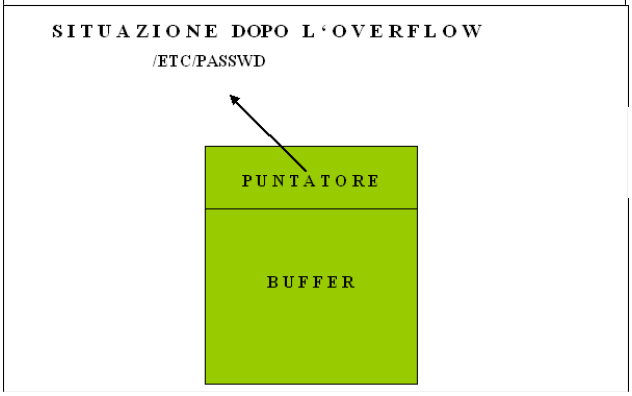
- L'aggressore colloca in un certo punto in memoria lo shellcode e sovrascrive opportunamente il buffer residente nell'Heap;
- L'aggressore sollecita o attende che l'area di memoria sovrascritta venga liberata dall'applicazione o ne venga sequenzialmente allocata una nuova;
- A seguito di uno degli eventi descritti nel punto precedente, l'indirizzo dello shellcode viene collocato in un punto in memoria arbitrariamente scelto dall'aggressore tramite la manipolazione dei puntatori memorizzati nella struttura che descrive il chunk liberato/allocato. Punti validi sono ad esempio gli indirizzi di chiamata a funzioni di hook o i puntatori a gestori delle eccezioni;

Lo shellcode viene eseguito.

Sovrascrivere un puntatore a file - Non tutti gli overflow che si manifestano nella regione di memoria heap possono essere sfruttati per eseguire uno shellcode sul sistema. Ad esempio, quando un heap overflow si manifesta in memoria, in prossimità di un puntatore a un file, l'aggressore può alterarlo e sollecitare la scrittura di dati arbitrari in un punto diverso del disco.
In questo modo, un aggressore potrebbe aggiungere al sistema un nuovo utente con password nulla, cambiare da remoto la configurazione di un'applicazione, disattivando alcune sue funzionalità di sicurezza o aggiungendovi direttive originariamente non previste. Un esempio schematico è rappresentato nelle figure che seguono:



Originariamente il file puntato è: /tmp/temp.tmp



A seguito dell'overflow il file puntato è: /etc/passwd

Esempio:
Le seguenti istruzioni causano un heap overflow:

```
int main(int argc, char **argv) {
    char *p, *q;

    p = malloc(1024);
    q = malloc(1024);
    if (argc >= 2)
        strcpy(p, argv[1]);
    free(q);
    free(p);
    return 0;
}
```

Se `argv[1]` supera, in lunghezza, il buffer dichiarato; viene "scritto" l'indirizzo non mappato dell'heap memory (relativamente ai dati).

Contromisure

Controllare e verificare sempre l'input utente. La lunghezza del buffer accettato non deve superare la lunghezza dell'area di memoria destinato a contenerlo.