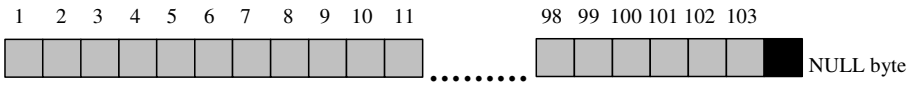
piacimento il flusso di esecuzione dell'applicazione, senza intaccare in modo diretto l'indirizzo di ritorno della funzione vulnerabile. In genere è sufficiente raggiungere l'ultimo byte dell'indirizzo dello stack frame della funzione vulnerabile (il frame pointer puntato ad esempio nell'architettura hardware x86 dal registro EBP) per sfruttare l'attacco eseguendo uno shellcode. Questo genere di errori si verifica molto spesso all'interno di cicli.

Esempio:
Esempio corretto di riempimento di un buffer



In una situazione normale la variabile buffer[104] dovrebbe contenere 103 byte di dati seguiti dal terminatore stringa NULL ('\0')

Esempio errato di buffer sovrascritto di pochi byte oltre le sue reali capacità di contenimento

Contromisure

Gli sviluppatori devono porre la massima attenzione sui loop all'interno degli array, rispettando la lunghezza allocata. I null di terminazione stringa devono essere conteggiati e considerati.

6.5.3 Format string overflow

Il Format string overflow è una tecnica abbastanza recente, descritta nella sua capacità di eseguire istruzioni remote su un sistema durante la prima metà del 2000. Precedentemente nota per i soli effetti di blocco di un'applicazione, questo genere di overflow si può manifestare nelle regioni di memoria stack o heap. Si verifica quando non viene specificato deliberatamente il formato di funzioni che lavorano le stringhe (ad esempio printf, fprintf, sprintf, snprintf), costruendo tale formato a partire dall'input utente.
Tramite il format string "%n", un aggressore può, infatti, scrivere un valore arbitrario in un qualsiasi punto dello spazio di memoria allocato per il processo dell'applicazione.
L'esecuzione di codice malevolo attraverso un format string overflow si sostanzia fondamentalmente in tre step:

- L'aggressore colloca in un certo punto in memoria lo shellcode;
- L'aggressore individua in memoria l'indirizzo di ritorno della funzione vulnerabile e lo sovrascrive con l'indirizzo in cui risiede lo shellcode;
- Al ritorno dalla funzione lo shellcode viene eseguito.

Questa tecnica è soggetta a variazioni nel caso di buffer che risiedono nella regione di memoria heap, dove per eseguire lo shellcode è eventualmente possibile sfruttare indirizzi di chiamata a funzioni di hook, puntatori a funzioni di distruzione (Destructor) invocate all'uscita dell'applicazione, puntatori a gestori delle eccezioni o puntatori a funzioni residenti in librerie esterne linkate con l'applicazione. Un aggressore può utilizzare uno di questi puntatori anche nel caso in cui l'overflow si manifesta nella regione di memoria stack (ad esempio per bypassare restrizioni di tipo stack canary/cookie o in quelle architetture in cui lo stack non risulti essere eseguibile).

Esempio

Se l'applicazione accetta parametri di sostituzione come `%x` e `%s` in istruzioni come la `printf`:

```
printf("valore immesso: %s", valoreInput);
```