6.5.5 Integer overflow ed altri errori logici di programmazione

Inizialmente con il termine integer overflow si tendeva a descrivere una moltitudine di vulnerabilità differenti tra loro. Solo nel 2002 questo tipo di problematica è stata circoscritta a una specifica condizione che si verifica quando un'applicazione effettua un'operazione matematica di addizione, sottrazione o moltiplicazione su un intero con segno, acquisendo un operando da input utente e non considerando i casi in cui il valore numerico ottenuto può essere negativo o minore/maggiore del previsto. Nel caso in cui l'aggressore ha la possibilità di specificare un valore arbitrario, può causare uno stack o un heap overflow secondario, quando il risultato dell'operazione matematica viene utilizzato per specificare la dimensione di un buffer, forzandone un'allocazione non sufficiente a contenere i dati acquisiti in ingresso dalla funzione vulnerabile.
Una problematica simile si verifica anche nei casi in cui un valore numerico acquisito da input utente viene convertito in un formato differente rispetto alla variabile originaria che lo contiene. Secondo il tipo di conversione, il risultato finale può differire notevolmente in eccesso o in difetto dal valore iniziale, causando l'allocazione di buffer insufficienti a soddisfare la necessità di contenimento dei dati o lo spostamento/la copia di un numero di byte eccessivo da un'area di memoria all'altra.
Un terzo fattore di instabilità in un'applicazione può derivare dalla asseganzione di valori non tenendo nella gisuta considerazione il fatto che una variabile numerica sia signed o unsigned.

Esempio:
Nel seguente codice un numero troppo grande causa un overflow della memoria:

```
char variabileChar = '0';
int valoreIntero = 1000;
variabileChar = valoreIntero;
```

`variabileChar`, dichiarato come `char`, può contentere: un valore da -128 a +127, se signed; un valore da 0 a 256, se unsigned. L'attribuzione del valore 1000 causerà un buffer overflow.
Nel seguente esempio, un valore accettabile in un char dichiarato unsigned, causa overflow se il char è dichiarato signed:

```
signed char variabileChar = '0';
int valoreIntero = 200;
variabileChar = valoreIntero;
```

Contromisure

- Controllare l'input dell'utente è indispensabile per verificare la congruità dei dati prima di accettarli.
- L'adozione delle Best practises di programmazione riduce gli errori e quindi l'insorgenza del buffer overflow.

6.6 Processi di tracciamento

Il tracciamento delle operazioni svolte dagli utenti è una delle attività più critiche per un'applicazione, poichè l'implementazione di un meccanismo di logging inadatto o insufficiente permette ad un aggressore di mascherare le sue operazioni, di sospendere il servizio o in taluni casi di eseguire comandi remoti sul sistema che ospita l'applicazione vulnerabile.
Di seguito sono riportate alcune categorie di errori che agevolano l'aggressore in operazioni che portano a sospendere il servizio di tracciamento dell'applicazione o in talune circostanze di eseguire codice da remoto.

6.6.1 Agevolazione delle attività malevole dell'aggressore

Una delle principali preoccupazioni di un aggressore che sferra o porta a termine un attacco a fini intrusivi è di rimuovere ogni traccia delle sue attività, per non essere chiaramente identificato. Qualora abbia la possibilità di manomettere il meccanismo di log, il tracciamento non fornirà all'amministratore alcuna