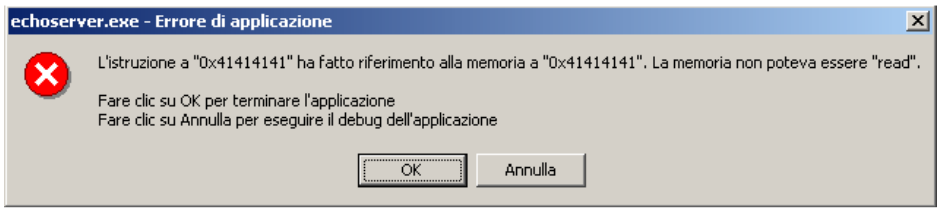
condizione di denial of service viene comunemente causata da un aggressore che sfrutta errori di programmazione riconducibili a problematiche di overflow (descritte nel paragrafo 4.2.6) o come effetto di un attacco non andato a buon fine, che mirava originariamente all'esecuzione di uno shellcode.
Condizioni di denial of service meno pesanti possono ad esempio causare il blocco di un account utente.
Deadlock - Nella programmazione multithread, uno degli errori che più comunemente da origine a problematiche di Denial Of Service è il deadlock. È una circostanza che si verifica quando due o più processi si fermano ad aspettarsi l'un l'altro, a tempo indefinito. La condizione che sbloccherebbe l'attesa, che potrebbe essere il termine di esecuzione di una procedura o il liberamento di una risorsa che causa il blocco, non si verifica mai.

Esempio di crash di un'applicazione che presenta una problematica di Stack Overflow:



L'attacco è andato a buon fine pertanto l'applicazione necessita di essere riavviata per fornire nuovamente il servizio agli utenti.

Contromisure

Dato che il denial of service può essere causato da numerose condizioni inerenti l'applicazione o l'ambiente operativo, le contromisure comprendono una serie di best practises di programmazione che limitino al minimo la superficie d'attacco.
A livello di web server è possibile: definire il numero massimo di richieste accettabili per una connessione TCP; stabilire un timeout e la dimensione massima del body di una singola richiesta; definire un timeout per ogni connessione.

6.4.5 Race condition

La race condition, dove "race " sta per "corsa" è una situazione che si verifica in un ambiente multithreading, dove più processi entrano in competizione per le stesse risorse. Ciò è possibile quando è importante la sequenza delle operazioni, ma l'accesso alle risorse da parte dei vari thread non è soggetto ad alcun vincolo.
La circostanza più classica è riconducibile a quelle applicazioni che devono scrivere dei dati sul disco dopo aver effettuato una serie di controlli preventivi. Un aggressore può usufruire del lasso di tempo in cui questi controlli vengono effettuati, o bloccare per un sufficiente periodo la loro esecuzione, sfruttando una vulnerabilità logica dell'applicazione (ad esempio un deadlock momentaneo), per alterare il dato di destinazione.
Le conseguenze di una modifica malevola del dato possono variare da un errore logico o applicativo, fino al crash dell'applicazione, o addirittura del sistema, se si riesce a generare un errore di overflow.

Esempio:
Il frammento di codice che segue; verifica l'accesso a un determinato file e nel caso in cui l'esito della verifica sia 'truè, apre il file in scrittura:

```
if (access("file", W_OK) != 0) {
  exit(1);
}
fd = open("file", O_WRONLY);
// Actually writing over /etc/passwd
write(fd, buffer, sizeof(buffer));
```

Se fra il controllo e l'apertura del file, l'attaccante riesce a creare un link simbolico a "file" attraverso la seguente sequenza di codice:

```
symlink("/etc/passwd", "file");
```