- Se possibile, isolare tutta l'esecuzione dinamica utilizzando un account utente separato e dedicato, che abbia privilegi solo per le operazioni e i file specifici utilizzati dall'applicazione, in base al principio del "Least Privilege". Il principio stabilisce che agli utenti venga attribuito il più basso livello di “diritti” che possano detenere rimanendo comunque in grado di compiere il proprio lavoro.

Esempio:

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {

 char cat[] = "cat ";
 char *command;
 size_t commandLength;

 commandLength = strlen(cat) + strlen(argv[1]) + 1;
 command = (char *) malloc(commandLength);
 strncpy(command, cat, commandLength);
 strncat(command, argv[1], (commandLength - strlen(cat)) );

 system(command);
 return (0);
}
```

L’istruzione system() esegue un commando proveniente dall’input, non verificato né controllato.

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/77.html,
CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection').

7.1.3 Connection String Injection

Come riconoscerla

Un utente malintenzionato potrebbe manipolare la stringa di connessione dell'applicazione al database. Utilizzando semplici strumenti di modifica testo, l'aggressore potrebbe essere in grado di eseguire una delle seguenti operazioni:

- Danneggiare le performance delle applicazioni (ad esempio incrementando il valore relativo al MIN POOL SIZE);
- Manomettere la gestione delle connessioni di rete (ad esempio, tramite TRUSTED CONNECTION);
- Dirigere l'applicazione sul database dell'attaccante al posto dell’originario;
- Scoprire la password dell'account di sistema del database.

Per comunicare con il proprio database o con un altro server (ad esempio Active Directory), l'applicazione costruisce dinamicamente una sua stringa di connessione. Questa stringa di connessione viene costruita dinamicamente con l’input inserito dall’utente. Se tali valori non sono stati verificati né tantomeno sanificati, potrebbero essere utilizzati per manipolare la stringa di connessione.

Come difendersi

L’input deve essere validato, come già evidenziato nei punti precedenti.
Evitare di costruire dinamicamente stringhe di connessione. Se è necessario creare dinamicamente una stringa di connessione, cercare di non includere l'input dell'utente. In ogni caso, utilizzare utilità basate sulla piattaforma, come SqlConnectionStringBuilder di .NET.

Esempio:

```
int main( int argc, char* argv[] )
{
       int result;
```

```
	if ( argc == 3 )
	{
		char* databaseServer = argv[1];
		char* databaseName = argv[2];

		char connString[BUFFER_SIZE] = database_PROTOCOL_STRING;
		strncat( connString, databaseServer, sizeof(connString) -
strlen(connString) - strlen(database_PORT_STRING) );
		strcat( connString, database_PORT_STRING );

		sql::mysql::MySQL_Driver* database_driver =
sql::mysql::get_mysql_driver_instance();
		sql::Connection* database_conn = database_driver->connect(
connString, database_USER, database_PASSWORD );
		database_conn->setSchema( databaseName );

		result = processData( database_conn );

		delete database_conn;
	}
	return result;
}
```

Nell'esempio riportato, la stringa di connessione viene costruita concatenando parametri di input.
Nel codice che segue, la scelta da una white list è obbligata:

```
int main( int argc, char* argv[] )
{
	int result;
	if ( argc == 2 )
	{
		int appId = atoi( argv[1] );

		char* connString;
		char* databaseName;
		switch( appId ) {
			case APP_ID1:
				connString = CONN_STRING_APP1;
				break;
			case APP_ID2:
				connString = CONN_STRING_APP2;
				break;
			case APP_ID3:
				connString = CONN_STRING_APP3;
				break;
			default:
				connString = CONN_STRING_DEFAULT;
		}

		sql::mysql::MySQL_Driver* database_driver =
sql::mysql::get_mysql_driver_instance();
		sql::Connection* database_conn = database_driver->connect(connString,
database_USER, database_PASSWORD);

		result = processData(database_conn);

		delete database_conn;
	}
	return result;
}
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/99.html,
CWE-99: Improper Control of Resource Identifiers ('Resource Injection').