- Alterare i permessi di file e directory all'interno del file system (read / create / modify / delete)
- Permettere delle connessioni di rete non autorizzate verso il server da parte dell'attaccante
- Acquisire il controllo dei servizi di sistema, arrestandoli o avviandoli.
- Prendere il pieno controllo del server.

Come difendersi

Rimodulare il codice per evitare una qualsiasi esecuzione diretta di script di comandi. Per effettuare operazioni di basso livello, devono essere preferite specifiche API fornite dalle aziende produttrici di software.
Se è non è possibile rimuovere l'esecuzione del comando, eseguire solo stringhe statiche che non includono l'input dell'utente.
Validare tutti gli input, indipendentemente dalla loro provenienza. La convalida dovrebbe essere basata su una white list: dovrebbero essere accettati solo i dati conformi a una struttura specificata, e scartati i dati che non rientrano in questa categoria. I parametri devono essere limitati a un set di caratteri consentito e i valori non validi devono essere eliminati. Oltre ai caratteri, occorre verificare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list). Configurare l'applicazione da eseguire utilizzando un account utente limitato che non disponga di privilegi non necessari.
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 denominato "Principle of 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:
Nel seguente codice viene utilizzato un input non verificato per lanciare una shell dei comandi:

```
import subprocess

def codifica_file():
  nome_file = raw_input('Inserire nome file da codificare: ')
  comando = 'ffmpeg -i "{source}" file_di_output.mpg'.format(source=nome_file)
  subprocess.call(comando, shell=True) # DA NON FARE
```

Se viene fornito un nome file concatenato con la stringa "; rm -rf /", il comando di cancellazione dell'intero file system verrebbe eseguito automaticamente.
Il paramentro shell deve essere sempre "false" per impedire l'esecuzione di comandi multipli, ma ciò non è sufficiente se la stringa passata, incece di conetenere un nome file, contiene un comando malevolo. Sarebbe meglio non usare affatto la subprocess.call(), ma se proprio dev'essere fatta, il parametro passato a questa funzione dovrebbe essere sottoposto a escaping con la funzione shlex.quote().

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').

7.5.4 Connection String Injection

Come riconoscerla

Questo tipo di attacchi è possibile nel momento in cui l'applicazione affida all'input utente la composiszione dinamica della stringa di connessione al database o a un server LDAP. Un malintenzionato potrebbe inserire una stringa opportunamente artefatta ed 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 fraudolento anziché a quello genuino;