```
  $action = str_replace(array("\n", "\r"), '',$_GET['action']); // Toglie gli "a
capo"
  $filename = $_GET['lognamè]; // Un utente malintenzionato può fornire un
"logname" che si trova sotto la webroot, creando un file che verrebbe servito dal
server
  $file = fopen($filename,'a');
  fwrite($file, $action." was performed successfully.".PHP_EOL); // An attacker
can set $action to "<?php passthru($_GET['c']); ?>", resulting in a basic shell
}
```

Codice bonificato:

```
if (isset($_GET['lognamè]) && isset($_GET['action'])) {
  $action = str_replace(array("\n", "\r"), '',$_GET['action']); // Toglie gli "a
capo"
  $filename = "/var/log/application/".basename($_GET['lognamè]); // // Può creare
file di log arbitrari, ma limitati a una cartella specifica sul sistema.
  $file = fopen($filename,'a');
  fwrite($file, $action." was performed successfully.".PHP_EOL);
}
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/552.html,
CWE- 552: Files or Directories Accessible to External Parties.

7.9.7 LDAP Injection

Come riconoscerla

La LDAP Injection è un tipo di attacco cui sono vulnerabili le applicazioni e che utilizzano l'input, senza verificarlo adeguatamente, per costruire query LDAP (Lightweight Directory Access Protocol).
Se coronato da successo, l'LDAP injection potrebbe consentire un furto di informazioni, un'elevazione dei privilegi e l'autenticazione con un'identità altrui (spoofing).
Per comunicare con la directory delle utenze (ad esempio Active Directory), l'applicazione costruisce dinamicamente delle query. Se utilizza l'input utente senza verificarlo, un malintenzionato può inserire comandi modificati ad arte per carpire informazioni non dovute.

Come difendersi

Validare tutti gli input, indipendentemente dalla loro provenienza. Per la validazione, si consiglia l'approccio white list (sono accettati solo i dati che adottano una struttura specificata nella white list, scartando quelli che non la rispettano). Occorre controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).

Esempio:
Codice vulnerabile:

```
function checkIsUserAdmin() {
  $username = $_POST['usernamè];
  $result = ldap_search($DS, $BASEDN,
"(&(username={$username})(memberOf={$ADMIN_GROUP}))", $LDAP_ATTRIBUTES);
  $foundResults = !($result === FALSE);
  return $foundResults;
}
```

Codice bonificato tramite regular expression:

```
function checkIsUserAdmin() {
  $username = $_POST['usernamè];
  $sanitizedUsername = preg_replace("/[^[:alnum:][:space:]]/u", '', $username);
  $result = ldap_search($DS, $BASEDN,
"(&(username={$sanitizedUsername})(memberOf={$ADMIN_GROUP}))", $LDAP_ATTRIBUTES);
  $foundResults = !($result === FALSE);
  return $foundResults;
}
```