MOPS-2010-031: e107 Usersettings loginname SQL Injection Vulnerability (UPDATED)

May 16th, 2010

An SQL Injection vulnerability was discovered in the user settings dialog of e107 that allows any user to become an admin easily.

Affected versions

Affected is e107 <= 0.7.20

Risk

Critical.

Credits

The vulnerability was discovered by Stefan Esser as part of the SQL Injection Marathon.

About e107

e107 is a content management system written in PHP and using the popular open source MySQL database system for content storage. It’s completely free, totally customisable and in constant development.

Detailed information

This vulnerability was discovered during SQL Injection Marathon a PHP code auditing marathon performed by Stefan Esser. The basic idea of this initiative is to select random PHP applications and perform a short code audit on them. The maximum time spent on each application is 30 minutes and after the first found SQL injection usually the next application is audited.

During SQL Injection Marathon e107 was also audited and because the code structure was already known, it was possible to find an SQL injection vulnerability in less than 30 minutes. The offending code is located in usersettings.php.

```
if ($ret == '')
{
  $loginname = strip_tags($_POST['loginname']);
  if (!$loginname)
  {
    $loginname = $udata['user_loginname'];
  }
  else
  {
    if(!check_class($pref['displayname_class'], $udata['user_classlist'], $peer))
    {
      $new_username = "user_name = '{$loginname}', ";
      $username = $loginname;
    }
  }

  ...
  $_POST['signature'] = $tp->toDB($_POST['signature']);
  $_POST['realname'] = $tp->toDB($_POST['realname']);
  ...

  // We can update the basic user record now
  $sql->db_Update("user", "{$new_username} {$pwreset} {$sesschange} user_email='".$tp -> to
```

This code puts the POST variable ‘loginname’ into the SQL query without proper database escaping. However a few lines above there is a little filter that stops some characters from being injected.

```
// Login Name checks
  if (isset($_POST['loginname']))
  {  // Only check if its been edited
    $temp_name = trim(preg_replace('/&nbsp;|\#|\=|\$/', "", strip_tags($_POST['loginname'])));
    if ($temp_name != $_POST['loginname'])
    {
      $error .= LAN_USET_13."\\n";
    }
    // Check if login name exceeds maximum allowed length
    if (strlen($temp_name) > varset($pref['loginname_maxlength'],30))
    {
      $error .= LAN_USET_14."\\n";
  }
    $_POST['loginname'] = $temp_name;
  }
```

The filter disallows some characters like # and = which would help in an SQL injection attack and checks that the loginname does not exceed an allowed maximum length. However this filter is not enough to stop a successful attack, because all that is required for successful attack are the characters '/*. With these characters it is possible to terminate the string context and open a comment that can be closed e.g. in the signature.

Proof of concept, exploit or instructions to reproduce

The following guide explains how to reproduce this vulnerability with only the Firefox browser and the tamper data extension.

1. First visit a vulnerable installation of e107.
2. Register a new user with the name `xpltest`.
3. Login as `xpltest`.
4. Go to user settings.
5. Look into the HTML source code of the page and extract the user id from the hidden field _uid. Let us assume it is 444.
6. Enter `xpltest'-- x` into the realname field.
7. Enter `<emtpy-line> ,user_admin=1,user_perms=0 where user_id=444 -- x` into the signature field. (Replace 444 with your user id)
8. Start tamperdata and activate tampering
9. Submit the user settings and change the variable `realname` into `loginname`.
10. The user `xpltest` is now an admin

Notes

This vulnerability has been patched by the e107 authors with an ineffective patch.

The proof of concept exploit was updated to still work with the patched e107. Aside from that the proof of concept exploit lacked the `,user_perms=0` part, which resulted in not full admin priviledges.