MOPS-2010-007: ClanTiger Shoutbox Module s_email SQL Injection vulnerability

May 4th, 2010

A SQL injection vulnerability was discovered in the shoutbox module of ClanTiger that allows retrieving all the data from the database.

Affected versions

Affected is ClanTiger <= 1.1.3

Credits

The vulnerability was discovered by Stefan Esser during the Month of PHP Security SQL Injecton Marathon.

About ClanTiger

ClanTiger is a gaming/clan CMS with dozens of nifty features.

Are you looking for an easy way to set up your clan website? Looking for a system that will make your gaming website easy to administrate and fun to use? Have a look at the ClanTiger clan cms! Say no to the crappy ones!

This clan cms aims to get you and your website the website you have always dreamed of which is also lightweight and easy to use. And if you have an idea or you simply think something suck (which is doubtable) we will certainly take it in consideration. The more people in the community ask for a feature, the more likely it will make itâ€™s appearance in ClanTiger.

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 ClanTiger was also audited and within 30 minutes it was possible to find a SQL injection vulnerability in the shoutbox module. The offending code is located in the modules/shoutbox.php file within the append() method.

```
// is the email valid?
if(!check_email_address($_POST['s_email']))
{
   message('INFORMATION',translate('Invalid information'),translate('The e-mail address you have e
   exit;
}

// check whether there is a member in the database with the supplied e-mail
$emailMember = $this->db->getrows("SELECT count(1) as found FROM " . DB_PREFIX . "membe
```

One can see here that the POST parameter s_email is directly inserted into the SQL query without any kind of escaping. This can only be safe if POST is previously modified by a database escaping function or a validation has taken place. Looking through the code one can find a magic_quotes_gpc deactivation layer that ensures POST parameters are **not prepared** for SQL, so a validation is required for the statement to be secure. The only validation is a call to the check_email_address() function to verify that the input is in email format. So in order to see that this vulnerability does indeed exist it is necessary to look into this function.

```
function check_email_address($email)
{
  // First, we check that there's one @ symbol, and that the lengths are right
  if (!ereg("^[^@]{1,64}@[^@]{1,255}$", $email)) {
    // Email invalid because wrong number of characters in one section, or wrong number of @ symbo
    return false;
  }
  // Split it into sections to make life easier
  $email_array = explode("@", $email);
  $local_array = explode(".", $email_array[0]);
  for ($i = 0; $i < sizeof($local_array); $i++) {
     if (!ereg("^(([A-Za-z0-9!#$%&'*+/=?^_`{|}~-][A-Za-z0-9!#$%&'*+/=?^_`{|}~\.-]{0,63})|(\"[^(\\|\
      return false;
    }
  }
  if (!ereg("^\[?[0-9\.]+\]?$", $email_array[1])) { // Check if domain is IP. If not, it should be valid doı
    $domain_array = explode(".", $email_array[1]);
    if (sizeof($domain_array) < 2) {
        return false; // Not enough parts to domain
    }
    for ($i = 0; $i < sizeof($domain_array); $i++) {
      if (!ereg("^(([A-Za-z0-9][A-Za-z0-9-]{0,61}[A-Za-z0-9])|([A-Za-z0-9]+))$", $domain_array[$i]))
        return false;
      }
    }
  }
  return true;
}
```

The function itselfs throws several regular expressions against the email address to verify it. However as long the SQL injection payload is shorter than 62 characters, does not contain a \ or ” character and is surrounded by double quote characters it will pass the validation. An example for such a payload would be

"'or id=1 and 0=sleep((substr(password,1,1)='2)*5)#"@xxx.de

Because the answer from the database is posted to the shoutbox and not removeable by an attacker only a time based attack like in the example seems practical.

Proof of concept, exploit or instructions to reproduce

No proof of concept exploit will be provided for this vulnerability.

Notes

This vulnerability was not disclosed to the vendor, because there was no email address mentioned for bug reports or any other way to submit security reports aside from posting it publically in their forum. And in order to contact them somehow one needs to create an account at the forum first.

We therefore strongly recommend to the vendor to have a publically viewable contact address for security bug reports.