

Group Project Progress Report

Eric Walker, Mariano Pennini, Xiaoyu Shi

Our group was assigned [CVE-2009-4092](#). The vulnerability exists in the user.php file of Simlog 0.9.3.2, an application used for blogging. Attackers can use cross-site request forgery (CSRF) to hijack the user authentication process by sending a malicious request to change the password. Upon following a malicious URI crafted by the attacker, given that the victim has an active session on the target website, the victim's password will be changed. Since the application cannot tell the difference between a same-site and cross-site request, it leaves itself vulnerable to this type of attack.

Our exploit outline:

1. Set up a web page that integrates Simlog
 - a. We have a copy of the vulnerable code and we will use this to set up the page (install.php)
2. We require that a user is logged in and authenticated to simlog.
3. Then we force the user to click on our crafted link, via an email, and supply the new password that we wish to set.
 - a. For example, if we wish to set the new password of a user to newPass, we will trick the user to click on the following link in order to submit this request on behalf of the logged in user:
 - i. `http://www.example.com/simlog/user.php?pass1=newPass&pass2=newPass&blogid=1&act=change`

```
} elseif($_REQUEST['act'] == "change") {  
  
    if(($_REQUEST['pass1'] == "") or ($_REQUEST['pass2'] == "") or ($_REQUEST['pass1'] != $_REQUEST['pass2'])) {  
        $err = "<font color=red><b>Passwords must match!</b></font><P>";  
    } else {  
        $enc = md5($_REQUEST['pass1']);  
        $sql = "UPDATE blog_users set password='$enc' where login='$_SESSION[login]'";  
        $res = $db->Execute($sql);  
        echo "<b>Password updated</b><br><hr><p>\n";  
    }  
}
```

According to the code above, once user.php identifies the action to be "change" (`$_REQUEST['act'] == "change"`), it does the following:

1. Checks if "pass1" and "pass2" are the same. In this case, pass1 and pass2 are the new password typed twice for confirmation. In order to update the password, they have to be the same, and neither of them could be empty.
2. If pass1 and pass2 are nonempty, and are not the same, user.php hashes the new password, and create a SQL command to update the password of the current blog user.

In order to patch the vulnerability, we seek to provide the application with a means to distinguish same-site and cross-site requests. Using a unique security token generated by the application at the beginning of each new session for each user, every request can be sent with this token to verify that the request is a same-site request. By making the secret token randomly generated and of sufficient length, there is no way for attackers to correctly guess the secret token with noticeable probability.