

[UNIX] Benchmark Designs' WHM Autopilot Backdoor Allows Plaintext Credential Leakage

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-08/0013.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 08/04/04

To: list@securiteam.com

Date: 4 Aug 2004 00:54:42 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Benchmark Designs' WHM Autopilot Backdoor Allows Plaintext Credential Leakage

SUMMARY

<> Benchmark Designs' WHM Autopilot is a client management system made for webhosts, in order to simplify webhosting business management. It manages CPANEL and WHM accounts, including account creation, maintenance, and removal. It is meant to be a solution to automate account billing and account creation.

Due to a bug in client login code and the builtin login backdoor it is possible to generate the hash required to get a user's username and plain-text password.

DETAILS

Vulnerable Systems:

* Benchmark Designs WHM Autopilot versions 2.4.5 and prior

Benchmark Designs' WHM Autopilot is vulnerable to plain-text credential leakage due to a bug in client logins. In the client login page (/clogin.php) there is a built in backdoor for administrators to login as

Securiteam: [UNIX] Benchmark Designs' WHM Autopilot Backdoor Allows Plaintext Credential Leakage

standard user accounts. This backdoor is accessed using the GET var 'c'. This variable is nothing more than an encrypted user ID, which is an automatically incremented field in the database.

Using WHM Autopilot's encryption functions `clogin_e()`, and the PHP method `base64_encode()`, one can generate the hash required to get a user's username and plain-text password. The required WHM Autopilot functions are found in `/inc/client_functions.php`. Since the user ID field is automatically incremented, one can generate keys for as many accounts as desired. The code to generate these keys would be:

----- Begin Code

```
<?php
$numAccounts = 5; // Set to any #
for($i=1; $i <= $numAccounts; $i++) {
    echo base64_encode(clogin_e($i))."<br />";
}
?>
```

----- End Code

This code creates a list of values to feed to `clogin.php` as the GET variable 'c'. Once the complete URI is requested, including the GET variable:

<http://somedomain/accounts/clogin.php?c=KEY>

The result is that the login form will automatically take on the plain-text values of the account's username and password. Note that the passwords are stored using the same encryption methods as we find for the user ID here. I have found that you do not always get a fully working key on the first try. Sometimes the key you generate is only good enough to get you a plaintext username, but not an encrypted password.

If this is the case, continue generating the keys until you get one that gives you the plain-text password. Once the username and password are achieved, a simple click of the login button accesses an entire user account, including CPanel access, account cancellation access, and payment functions access.

Workaround

This bug can be fixed by removing the backdoor. Since `clogin.php` is thankfully not encoded with the Zend Optimizer, the backdoor code can be removed. The backdoor code needing to be removed is the following:

----- Begin Code

```
if (isset($c))
{
    $c=clogin_d(base64_decode($c));

    $query="select ";
    $query.="username, ";
```

Securiteam: [UNIX] Benchmark Designs' WHM Autopilot Backdoor Allows Plaintext Credential Leakage

```
$query.="password ";
$query.="from ";
$query.="user ";
$query.="where ";
$query.="uid="".addslashes(trim($c))."" ";
$query.="limit 0, 1";

$rs=mysql_fetch_row(mysql_query($query));

$username=$rs[0];
$password=clogin_d(base64_decode($rs[1]));
}
```

----- End Code

In version 2.4.5, the code spans lines 77 to 94. Simply removing this code, and saving the file, will remove this vulnerability. Removing this code will disable Administrative logins for standard users, but the vendor could easily conjure a workaround for that. Ultimately however, user credentials should not be stored in a form that can be resolved to plain-text, one way hashes should be used for added security, and no backdoors should exist.

Disclosure Timeline

Problem Discovered: July 30, 2004

Vendor Notified: August 1, 2004

Public Release: August 1, 2004

ADDITIONAL INFORMATION

The information has been provided by <<mailto:msblows@sdf.lonestar.org>> MS Blows.

=====
This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.