Security Research Advisory

Simple PHP Blog

Multiple Vulnerabilities

Table of Contents

Summary

Simple PHP Blog is a blogging application that was written with simplicity of installation and maintenance in mind. Unlike other blog software, there is almost no setup because it uses flat text files.

Multiple vulnerabilities have been reported in the latest version of this web application; probably all previous versions are affected to the same issues. The specific issues include multiple cross-site scripting flaws and an arbitrary file upload vulnerability. Various consequences are associated with these issues, such as theft of cookie-based authentication credentials and arbitrary remote code execution.

Date	Details
14/09/2007	Vendor disclosure
14/09/2007	Vendor acknowledgment
23/09/2007	Patch release
25/09/2007	Public disclosure

Arbitrary File Upload

Arbitrary File Upload					Advisory Number
					SN-07-03
Severity	Software	Version	Accessibility	CVE	Author(s)
H	Simple PHP Blog	0.5.0.1, ≤ 0.4.8	Remote	*n/a*	Luca Carettoni Luca De Fulgentis
	Vendor URL		Advisory URL		
	http://www.simplephpblog.com		-		

Vulnerability Details

Simple PHP Blog is prone to an arbitrary file upload flaw because the application fails to check the upload denied files. A potential attacker can upload arbitrary file on the remote operating system.

In order to exploit the arbitrary file upload vulnerability, a regular user should be authenticated. It should be noted that the latest versions of the application haven't multiple users support.

Technical Details

Description

The vulnerability concerns *upload_img_cgi.php* file, as shown below in the Vulnerable Code paragraph.

Using a fake GIF image is possible to bypass the image content control and the file extension check. As an example, create a file called "exploit.php." with the following content.

PoC:

```
GIF89aD
<?php phpinfo(); ?>
```

An attacker could upload the script on the "/images" directory inside the application dir on the webserver. Thanks to "by-design" behaviors of Apache httpd *mod_mime* parsing files with multiple extensions, it's possible to execute the uploaded script.

In Microsoft Windows server environment it's possible too, due to the filename with multiple dot handling.

Exploiting this issue could allow an attacker to upload and execute arbitrary script code in the context of the affected webserver process.

Vulnerable Code

File: *upload_img_cgi.php*

```
[…]

if ( @getimagesize($_FILES['userfile']['tmp_name']) == FALSE ){
                        echo('Image is not valid or not an image file.');
                        exit;
                        // redirect_to_url( 'upload_img.php' );
}

[…]

                $upload_denied_extentions = array( "exe", "pl", "php", "php3", "php4", "php5", "phps",
        "asp","cgi", "html", "htm", "dll", "bat", "cmd" );
                $extension = strtolower(substr(strrchr($uploadfile, "."), 1));
                foreach ($upload_denied_extentions AS $denied_extention) {
                        if($denied_extention == $extension) {
                                echo('That filetype is not allowed');
                                exit;
}}

[…]
```

Multiple Cross-Site Scripting

<table>
<tr><td colspan="5">Multiple Cross-Site Scripting</td><td>Advisory Number</td></tr>
<tr><td colspan="5"></td><td>SN-07-03</td></tr>
<tr><td>Severity</td><td>Software</td><td>Version</td><td>Accessibility</td><td>CVE</td><td>Author(s)</td></tr>
<tr><td rowspan="3">H</td><td>Simple PHP Blog</td><td>0.5.0.1,
≤ 0.4.8</td><td>Remote</td><td>n/a</td><td rowspan="3">Luca Carettoni
Luca De Fulgentis</td></tr>
<tr><td colspan="2">Vendor URL</td><td colspan="2">Advisory URL</td></tr>
<tr><td colspan="2">http://www.simplephpblog.com</td><td colspan="2">-</td></tr>
</table>

Vulnerability Details

Mutiple reflected XSS have been found in the *\themes\<themes name>\user_style.php* file. Exploiting the XSS flaw it is possible to steal the authentication token and then exploit the other vulnerability in order to execute arbitrary code (such a PHP shell).

Technical Details

Description

Looking inside the application source code:

user_style.php:

```
<style type="text/css">
body {
background-color: #<?php echo( $user_colors[ 'bg_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'txt_color' ] ); ?>;
```

It's easy to see that the "*user_colors[bg_color]*" is not validated and it's used directly inside an echo function.

Sending a trivial HTTP request against PHP environments having register global ON is possible to exploit this unvalidated user input flaw. In detail, it's necessary to append a close HTML tag </style> before the malicious JavaScript code.

The same problem arises in different point of the same script, for each different theme template.

user_style.php:

```
background-color: #<?php echo( $user_colors[ 'bg_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'txt_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'inner_border_color' ] ); ?>;
background-color: #<?php echo( $user_colors[ 'inner_border_color' ] ); ?>;
border-color: #<?php echo( $user_colors[ 'border_color' ] ); ?>;
border-color: #<?php echo( $user_colors[ 'border_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'header_txt_color' ] ); ?>;
background-color: #<?php echo( $user_colors[ 'header_bg_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'footer_txt_color' ] ); ?>;
background: #<?php echo( $user_colors[ 'footer_bg_color' ] ); ?>;
border-top: 1px solid #<?php echo( $user_colors[ 'border_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'headline_txt_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'headline_txt_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'date_txt_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'date_txt_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'entry_text' ] ); ?>;
background-color: #<?php echo( $user_colors[ 'bg_color' ] ); ?>;
border-color: #<?php echo( $user_colors[ 'entry_text' ] ); ?>;
border-color: #<?php echo( $user_colors[ 'inner_border_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'link_reg_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'link_hi_color' ] ); ?>;
color: #<?php echo( $user_colors[ 'link_down_color' ] ); ?>;
border-color: #<?php echo( $user_colors[ 'inner_border_color' ] ); ?>;
```

Legal Notices

Secure Network (*www.securenetwork.it*) is an information security company, which provides consulting and training services, and engages in security research and development.

We are committed to open, full disclosure of vulnerabilities, cooperating with software developers for properly handling disclosure issues.

The information in the advisory is believed to be accurate at the time of publishing based on currently available information. This information is provided as-is, as a free service to the community by Secure Network research staff. There are no warranties with regard to this information. Secure Network does not accept any liability for any direct, indirect, or consequential loss or damage arising from use of, or reliance on, this information.

If you have any comments or inquiries, or any issue with what is reported in this advisory, please inform us as soon as possible.

e-mail	info@securenetwork.it
phone	+39 02 917 730 41