Yorick Koster, February 2017

SyntaxHighlight MediaWiki extension allows injection of arbitrary Pygments options

Abstract

A vulnerability was found in the SyntaxHighlight MediaWiki extension. Using this vulnerability it is possible for an anonymous attacker to pass arbitrary options to the Pygments library. By specifying specially crafted options, it is possible for an attacker to trigger a (stored) Cross-Site Scripting condition. In addition, it allows the creating of arbitrary files containing user-controllable data. Depending on the server configuration, this can be used by an anonymous attacker to execute arbitrary PHP code.

Tested versions

This issue was tested on SyntaxHighlight version 2.0 as bundled with MediaWiki version 1.28.0.

Fix

There is currently no fix available.

Introduction

The SyntaxHighlight extension for MediaWiki allows formatting of source code using the <*syntaxhighlight*> tag. Version 2.0 uses the Python Pygments library to format the code. SyntaxHighlight is bundled with MediaWiki version 1.21 and later. Version 2.0 is bundled with MediaWiki 1.26.0 and later (other versions may or may not include this version as well).

The <*syntaxhighlight*> tag supports various parameters. It was found that the *start* parameter is not validated and/or sanitized. This allows an attacker to pass arbitrary options to the Lexer and/or Formatter that is used when Pygments is invoked. By specifying specially crafted options, it is possible for an attacker to trigger a (stored) Cross-Site Scripting condition. In addition, the HTML formatter allows the creating of arbitrary files containing user-controllable data. Depending on the server configuration, this can be used by an attacker to execute arbitrary PHP code.

Details

The SyntaxHighlight extension utilizes Pygments to format source code. Pygments is a Python library, a copy is provided with the extension. In order to use Pygments, the extension invokes it using Symfony's ProcessBuilder component. This component performs escaping of command line arguments to prevent command injection.

SyntaxHighlight_GeSHi.class.php:

```
$optionPairs = array();
foreach ( $options as $k => $v ) {
   $optionPairs[] = "{$k}={$v}";
}
$builder = new ProcessBuilder();
$builder->setPrefix( $wgPygmentizePath );
$process = $builder
   ->add( '-l' )->add( $lexer )
   ->add( '-f' )->add( 'html' )
```

```
   ->add( '-O' )->add( implode( ',', $optionPairs ) )
   ->getProcess();

$process->setInput( $code );
$process->run();
```

The used Lexer is specified through the *lang* parameter, the Formatter is always set to the *HtmlFormatter*. Additional options for the Lexer and/or Formatter are provided using the *-O* command line argument. These options can be controlled by the parameters that are supported by the <*syntaxhighlight*> tag. Each option is a key value pair, the options are comma separated.

It was found that no input validation and/or sanitization is done on the *start* parameter. This parameter is used to define the first line number of a code block. If line numbers are enabled, the numbering will start with the value provided in the *start* parameter. Normally, this value should only contain numbers. Due to the lack of validation/sanitization, it can be set to any value.

SyntaxHighlight_GeSHi.class.php:

```
// Starting line number
if ( isset( $args['start'] ) ) {
   $options['linenostart'] = $args['start'];
}
```

Since Lexer/Formatter options are comma separated, it is possible for an attacker to provide arbitrary options when invoking Pygments. Depending on the options supported by the Lexer or Formatter, this allows the attacker to perform various types of attacks. For example it is possible for an attacker to trigger a (stored) Cross-Site Scripting condition by passing a specially crafted *prestyles* option to the HTML Formatter.

```
<syntaxhighlight lang="java"
start='0,prestyles="&gt;&lt;script&gt;alert(document.cookie)&lt;/script&gt;'>
   string foo="bar";
</syntaxhighlight>
```

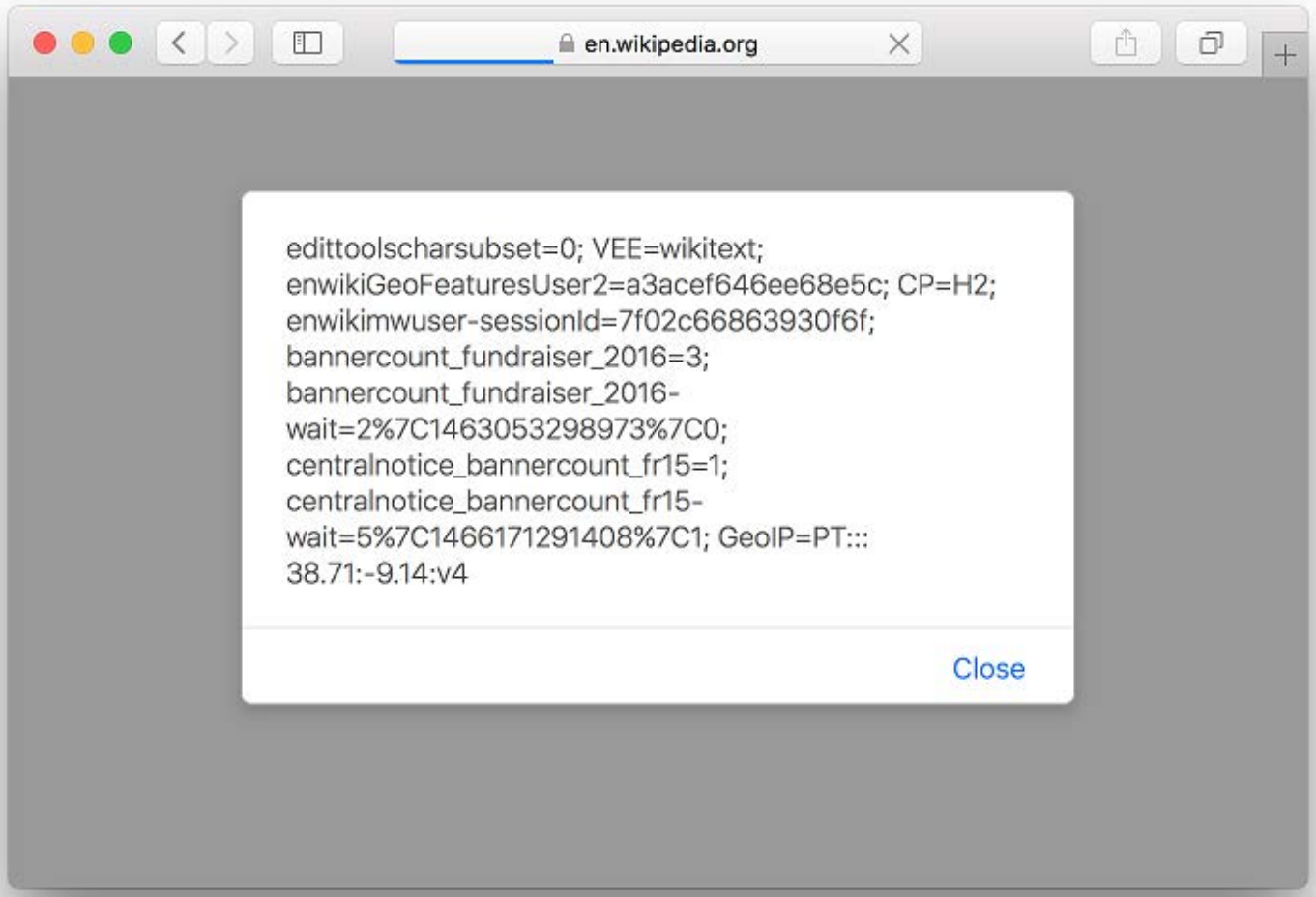

Figure 1: Cross-Site Scripting through *prestyles* option

When the option *full* is passed to the HTML Formatter, it is possible to specify a local CSS file using the *cssfile* option. If the CSS file does not exist it will be created - provided that Pygments has write privileges on the provided path. This CSS file contains the styles that are used for formatting the source code. Providing additional options, it is possible to control parts of the CSS. One such option is the *classprefix* option.

Combining these options can result in execution of arbitrary PHP code, provided that a writeable folder exists within the webserver's document root that allows the execution of PHP files. The proof of concept below will try to create a PHP file name *foo.php* in the *images* folder located within the document root.

```
<syntaxhighlight lang='java' start='0,full=1,title=,cssfile=images/foo.php,classprefix=&lt;?php
phpinfo();exit; ?&gt;'>
</syntaxhighlight>
```

Unless the Wiki is configured as private, it is possible to exploit this issue without logging into the Wiki. If the Wiki is set to private, an account with read access is required to exploit this vulnerability.